Sh.A. Nazirov, F.M. Nuraliev

DHTML yordamida Web-sahifa

“Dasturlash II” fanidan dasturlashga yo’naltirilgan IKT tizimilari muttahasislari uchun

TOSHKENT, 2006

Tastiqlangan: Samarqand sanoat va axborot texnologiyalar kasb-hunar kolleji pedagogik kengashi

Mazkur material “axborot va telekommunikatsiya texnologiyalar soxasida professional ta’limni ko’llash – kooperatsion loyixa” uzbek-nemis eksperimental loyiha miqiyosida professional kollejlarda ishlatish maqsadida ishlab chikilgan.
Material “axborot va kommunikatsion tizimlar” muttaxasisligi kasblariga o’qitishda qo’llaniladi. (kod 3521916).

Taqrizchilar:
1. L.A. Kokcheev, Samarqand sanoat va axborot tehnologiyalar kasb-hunar kolleji pedagogik uqituvchisi

2. prof. B. Kurmanbaev, f-m. f. d, Uzbekisnton Milliy Universiteti

Mundarija

1. DHTMLga kirish
Skriptlar tili (JavaScript)
Shartli (nisbiy) o'tuvchi funktsiyalar
Takrorlanuvchi funktsiyalar

Gipermatnli ma'lumotlar tizimi ma'lumotlar bo'g'inlari to'plamlari, shu bo'g'inlarda aniqlangan gipermatnli aloqalar to'plamlari va bo'g'inlar va aloqalarni boshqarish instrumentlaridan tashkil topgan. World Wide Web texnologiyasi bu – gipermatnli taqsimlangan sistemalarni Internetga kiritish texnologiyasi va shundan kelib chiqib, u bunday tizimlarning umumiy ta'rifiga mos kelishi kerak. Bu shuni bildiradiki, gipermatnli tizimlarning yuqorida keltirilgan barcha komponentalari Web da ham bo'lishi kerak.

Webga, gipermatnli tizim sifatida, ikki xil nuqtai nazardan qarash mumkin. Birinchidan, o'zaro gipermatnli o'tishlar (ANCHOR konteyneri) vositasida bog'langan tasvirlanishi kerak bo'lgan sahifalar t'plami sifatida. Ikkinchidan, tasvirlanayotgan sahifalar (matn, grafika, uyali kod va hokazolar) ni tashkli qiluvchi elementar ma'lumot ob'ektlarining to'plami sifatida. So'nggi holatda sahifaning gipermatnli o'tishlar to'plami – bu matnga ichki qo'yilgan rasm kabi ma'lumot bo'lagi.

Ikkinchi yondashuvda gipermatnli tizim elementar ma'lumot ob'ektlari to'plami uchun gipermatnli aloqalar rolini o'ynovchi HTML-sahifalarning o'zi tomonidan aniqlanadi. Bu yondashuv tayyor komponentalardan tasvirlanayotgan sahifalarni qurish nuqtai- nazaridan ancha serhosilroqdir.

Webda sahifalarni ayratishda "klient-server" arxitekturasi bilan bogliq muammo yuzaga chiqadi. Sahifalarni ham klient tomonida, ham server tomonida yaratish mumkin. 1995 yilda Netscape kompaniaysi muttaxasislari *JavaScript* dasturlash tilini ishlab chiqib, sahifalarni klient tomonida boshqarish mexanizmini yaratishdi.

Shunday qilib, *Javascript* – bu Webni gipermatnli sahifalarini klient tomonida ko'rish tsenariyalarini boshqarish tili. Yanada aniqroq aytadigan bo'lsa, *Javascript* – bu na faqat klient tomonidagi dasturlash tili. Liveware *Javascript* tilining avlodi bo'lib, Netscape serveri tomonida ishlovchi vosita bo'ladi. Ammo *Javascript* tilini mashhur qilgan narsa bu klient tomonida dasturlashdir.

*Javascript*ning asosiy vazifasi – *HTML-konteynerlar* atributlarining qiymatlarini va ko'rsatuvchi muhitining *hossalirini* HTML-sarlavxalarni ko'rish jarayonida foydalanuvchi tomonidan o'zgartirish imkoniyatlarida, boshqacha aytganda ularni dinamik sarlavxalar qilish (DHTML). Yana shuni aytish joizki, sarlavxalar qayta yuklanmaydi

Amalda buni, masalan, quydagicha ifodalash mumkin, sarlavxaning fonini rangini yoki xujjatdagi rasmni o'zgartirish, yangi oyna ochish yoki ogoxlantirish oynasini chiqarish.

"*JavaScript*" nomi Netscape kompaniyasining hususiy maxsuloti. Microsoft tomonidan amalga oshirilgan til rasman *Jscript* deb nomlanadi. *Jscript* versiyalari *Javascript*ning mos versiyalari bilan mos keladi (aniqroq qilib aytganda oxirigacha emas).

Javascript – ECMA (European Computer Manufacturers Association – Evropa Kompyuter Ishlab Chiqaruvchilar Assotsiyatsiyasi) tomonidan standartlashtirilgan. Mos standartlar quydagicha nomlanadi ECMA-262 va ISO-16262. Ushbu standartlar bilan *Javascript* 1.1ga taqriban ekvivalent ECMAScript tili aniqlanadi. Eslatish joizki, bugungi kunda *Javascript* ning hamma versiyalari ham ECMA standartlariga mos kelavermaydi. Mazkur kurs yoki qo'llanmada barcha hollarda biz *Javascript* nomidan foydalanamiz.

Javascriptning asosiy hususiyatlari. Javascript – bu Internet uchun katta bo'lmagan klient va server ilovalarni yaratishga mo'ljallangan nisbatan oddiy ob'ektga yo'naltirilgan til. *Javascript* tilida tuzilgan dasturlar HTML-xujjatning ichiga joylashtirilib ular bilan birga uzatiladi. Kurish dasturlari

(*brauzerlar – browser* ingliz suzidan) Netscape Navigator va Microsoft Internet Explorer xujjat matniga joylashtirilgan dasturlarni (*Scriptkod*) uzatishadi va bojarishadi.

Shunday qilib, *Javascript* – interpritatorli dasturlash tili xisoblanadi. *Javascript*da tuzilgan dasturlarga foydalanuvchi tomonidan kiritilayotgan ma'lumotlarni tekshirayotgan yoki xujjatni ochganda yoki yopganda biror bir amallarni bagaruvchi dasturlar misol bo'lishi mumkin.

JavaScript da yaratilgan dasturlarga misol sifatida foydalanuvchi tomonidan kiritilgan ma'lumotlarn tekshiruvchi, dokumentni ochish yoki yopish vaqtida qandaydir amallarni bajaruvchi dasturlarni keltirish mumkin. Bunday dasturlar foydalanuvchi tomonidan berilgan kursatmalarga – sichqoncha tugmachasini bosilishiga, ma'lumotlarni ekran orqali kiritishiga yoki sichqonchani sahifa buylab siljitilishiga kura ish bajaradi. Bundan tashqari JavaScript dagi dasturlar brauzerning uzini va dokumentning atributlarini ham boshqarishi mumkin.

JavaScript dasturlash tili sintaktik jihatdan Java dasturlash tiliga, ob'ektli modellashni istisno qilgan holda, o'hshab ketsada, lekin ma'lumotlarni statik tiplari va qat'iy tiplashtirish kabi hususiyatlarga ega bulmaydi. JavaScript da Java dasturlash tilidan farq qilib, sinf (klass) tushunchasi bu tilning asosiy sintaktik qurilmasi hisoblanmaydi. Bunday asos sifatida foydalanilayotgan tizim tomonidan qullab-quvvatlanayotgan, oldindan aniqlangan ma'lumot tiplari: sonli, mantiqiy va satrli; mustaqil ham bulishi, ob'ektning metodi (JavaScripda metod tushunchasi funktsiya/qism-dastur ning uzi) sifatida ham ishlatilishi mumkin bulgan funktsiyalar; katta sondagi uz hossalariga va metodlariga ega bulgan oldindan aniqlangan ob'ektlardan iborat ob'ektli model va yana dastur ichida foydalanuvchi tomonidan yangi ob'ektlarni berish qoidalari hisoblanadi.

JavaScript da dasturlar yaratish uchun hech qanday qo'shimcha vositalar kerak bulmaydi – faqatgina tegishli versiyadagi JavaScript qullanishi mumkin bulgan brauzer va DHTML-dokumentlarni yaratishga imkon beruvchi matn muharriri kerak bo'ladi. JavaScript dagi dastur bevosita HTML –dokumentlarni ichiga joylashtirilganligi uchun dastur natijasini dokumentni brauzer yordamida kurish orqali tekshirish mumkin va kerakli holda uzgartirishlar kiritilishi mumkin.

JavaScript dasturlash tilining imkoniyatlari. Uning yordamida HTML –dokumentlarning ko'rinishi va tuzilishini dinamik ravishda boshqarish mumkin. Ekranda tasvirlanayotgan dokumentga brauzer tomonidan yuklangan dokumentning sintaktik tahlil qilish jarayonida istalgan HTML-kodlarni joylashtirish mumkin. "Dokument" ob'ekti yordamida foydalanuvchining oldingi bajargan amallari yoki boshqa bir faktorlarga kura yangi dokumentlarni avtomatik hosil qilish mumkin.

JavaScript yordamida brauzer ishini boshqarish mumkin. Masalan, Window ob'ekti suzib yuruvchi oynalarni ekranga chiqarish, brauzerning yangi oynalarini yaratish, ochish va yopish, oynalarning yugurdagi va o'lchamlarining rejimlarini o'rnatish va hokazolarni imkoniyatini beruvchi metodlarga ega.

JavaScript dokumentdagi ma'lumotlar bilan bog'lanish imkoniyatini beradi. Document ob'ekti va undagi mavjud ob'ektlar dasturlarga HTML-dokumentlarning qismlarini o'qish va bazida ular bilan bog'lanish imkoniyatini beradi. Matnning o'zini o'qish mumkin emas, lekin masalan berilgan dokumentdagi gipermatnli o'tishlar ro'yhatini olish mumkin. Hozirgi vaqtda Form ob'ekti va undagi mavjud bo'lishi mumkin bo'lgan ob'ektlar: Button, Checkbox, Hidden, Password, Radio, Reset, Select, Submit, Text va Textarealar dokumentdagi ma'lumotlar bilan bo'g'lanish uchun keng imkoniyatlar beradi.

JavaScript foydalanuvchi bilan aloqa qilishga imkon beradi. Bu tilning eng muhim hususiyati unda amalga oshirilgan hodisalarni qayta ishlashni aniqlsh imkoniyati – ma'lum bir hodisaning (odatda foydalanuvchi tomonidan bajarilgan amal) ro'y berish vaqtida bajariladigan dastur kodining ihtiyoriy qismi hisoblanadi. JavaScript hodisalarni qayta ishlovchi sifatida ihtiyoriy yangi oldindan berilgan funktsiyalardan foydalanish imkoniyatini beradi. Masalan, foydalanuvchi sichqoncha ko'rsatkichini gipermatnli o'tishlar ustiga keltirsa, holatlar satrida mahsus habarni chiqaruvchi yoki

ma’lum bir amalni bajarishni tasdiqlashni so’rovchi dialogli oynani ekranga chiqaruvchi yoki foydalanuvchi tomonidan kiritilgan qiymatlarni tekshiruvchi va hatolik yuz bergan holda kerakli kursatmalarni berib, tug’ri qiymatni kiritishni so’rovchi dasturlarni yaratish mumkin.

JavaScript ihtiyoriy matematik hisoblashlarni bajarish imkoniyatini beradi. Bundan tashqari bu tilda vaqt va sanalarning qiymatlari bilan ishlovchi yuqori darajada rivojlangan vositalar mavjud. JavaScript CGI-dasturlarga va Perl dasturlash tiliga va to’diruvchi sifatida ayrim hollarda Java tiliga muqobil til sifatida yaratilgan.

Har bir boshlovchi dasturchining asosiy savoli: “Dasturlar qanday tuziladi va bajariladi? ”. Bu savolga iloji boricha soddaroq, lekin *JavaScript*-kodlarini qo’llanilishining barcha usullarini unutmagan holda javob berishga harakat qilamiz.

Birinchidan, *JavaScript*-kodlari brauzer tomonidan bajariladi. Unda mahsus *JavaScript* interpritatori mavjud. Unga kura programmaning bajarilishi interpretator tomonidan boshqaruvni qachon va qay tarzda olishiga bog’liq bo’ladi. Bu esa, o’z navbatida kodning funktsiyaviy qo’llanilishiga bo’g’liq bo’ladi. Umuman olganda *JavaScript* ning funktsional qo’llanishining 4 hil usulini ajratib ko’rsatish mumkin:

1. *gipermatnli o’tish (URL sxemasi);*
2. *hodisalarni qayta ishlash (handler);*
3. *o’rniga qo'yish(entity)*
4. *qo’yish (*SCRIPT *konteyneri).*

JavaScript bo’yicha o’quv qo’llanmalarida *JavaScript* ni qo’llashning bayoni odatda SCRIPT konteyneridan boshlanadi. Lekin dasturlash nuqtai nazaridan bu unchalik ham to’g’ri emas, chunki bunday tartib asosiy savol: “*JavaScript*-kodi boshqaruvni qanday oladi?” ga javob bermaydi. Ya’ni *JavaScript* da yozilgan va HTML-dokumentning ichiga joylashtirilgan dastur qanday tarzda chaqiriladi va bajariladi.

HTML-sahifa muallifining kasbi va uning dasturlash asoslaridan habardarligining darajasiga qarab *JavaScript* ni o’zlashtirishga kirishishni bir necha hil variantlari mavjud. Agar siz klassik tillar (C, Fortran, Pascal va h.) bo’yicha dasturlovchi bo’lsangiz, u holda dokument matni ichida dasturlashdan boshlagan ma’qul, agar siz Windows sistemasida dasturlashga o’rgangan bo’lsangiz, u holda hodisalarni qayta ishlashni dasturlashdan boshlaganingiz ma’qul, agar siz faqat HTML bo’yichagina tajribaga ega bo’lsangiz yoki anchadan beri dasturlash bilan shug’ullanmayotgan bo’lsangiz, u holda gipermatnli o’tishlarni dasturlashdan boshlaganingiz ma’qul.

URL-sxemali JavaScript. URL sxemasi (Uniform Resource Locator)- bu Web-texhologiyalarning asosiy elementlaridan biri. Web dagi har bir informatsion resurs uzining o’ziga hos URLiga ega bo’ladi. URL A konteynerining HREF atributida, IMG konteynerining SRC atributida, FORM konteynerining ACTION atributida va h.larda ko’rsatiladi. Barcha URL lar resursga ruhsatning protokoliga bog’liq bo’lgan ruhsat sxemalariga bo’linadi, masalan, FTP-arxiviga kirish uchun ftp sxemasi, Gopher-arxiviga kirish uchun gopher sxemasi, elektron maktublarni jo’natish uchun smtp sxemasi qo’llaniladi. Sxemaning tipi URLning birinchi komponentasiga ko’ra aniqlanadi: http://directory/page.html

Bu holatda URL http bilan boshlanayapti- mana shu kirish sxemasini aniqlashdir (sxema http).

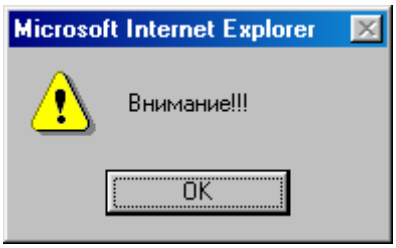
Gipermatnli sistemalar uchun dasturlash tillarining asosiy vazifasi gipermatnli o’tishlarni dasturlashdir. Bu shuni bildiradiki, u yoki bu gipermatnli o’tishlarni tanlashda gipermatnli o’tishni amalga oshiruvchi dastur chaqiriladi. Web-texnologiyalarida standart dastur sifatida sahifani yuklash dasturi hisoblanadi. HTTP protokoli bo’yicha standart o’tishni *JavaScript* da dasturlangan o’tishdan farq qilish uchun yaratuvchilar yangi URL sxemasi – *JavaScript* ni kiritishdi:

```
<A HREF="JavaScript:JavaScript_код">...</A>
<IMG SRC="JavaScript:JavaScript_код">
```

Bu holatda "JavaScript_код" matni birinchi holatda gipermatnli o'tishni tanlanganda chaqiriladigan va ikkinchi holatda rasmni yuklashda chaqiriladigan *JavaScript* da yaratilgan dasturiy-qayta ishlovchilarni bildiradi.

Masalan, Внимание!!! gipermatnli o'tishga keltirilgan holda ogohlantiruvchi oynani chiqarish mumkin:

```
<A HREF="JavaScript: alert('Внимание!!!');"> Внимание!!!</A>
```



Formadagi submit tipidagi tugmachani bosish orqali shu formadagi matnli maydonni to'ldirish mumkin:

```
<FORM NAME=f METHOD=post
ACTION="JavaScript: window.document.f.i.VALUE='Нажали кнопку Click';void(0);">
<TABLE BORDER=0>
<TR>
<TD><INPUT NAME=i></TD>
<TD><INPUT TYPE=submit VALUE=Click></TD>
<TD><INPUT TYPE=reset VALUE=Reset></TD>
</TABLE>
</FORM>
```

URLda murakkab dasturlar va funktsiya chaqirilishlarini joylashtirish mumkin. Faqatgina shuni yodda tutish kerakki, *JavaScript* sxemasi hamma brauzerlarda ham ishlamaydi, faqatgina Netscape Navigator va Internet Explorer larning to;rtinchi versiyalaridan boshlab ishlaydi.

Shunday qilib gipermatnli o'tishlarni dasturlashda interpretator boshqaruvni foydalanuvchi sichqoncha tugmasini gipermatnli o'tishga "bosgandan" keyingina oladi.

Hodisalarni qayta ishlovchilar. Hodisalarni qayta ishlovchi dasturlar (handler) shu hodisa bog'liq bo'lgan konteynerlarning atributlarida ko'rsatiladi. Masalan, tugmachani bosishda onClick hodisasi ro'y beradi:

```
<FORM><INPUT TYPE=button VALUE="Кнопка"
onClick="window.alert('Внимание!!!');"></FORM>
```

O'rniga qo'yish. O'rniga qo'yish (entity) Web-sahifalarda anchagina kam uchraydi. Shunga qaramay u brauzer tomonidan HTML-sahifani hosil qilish uchun ishlatiladigan kerakli darajadi kuchli instrument hisoblanadi. O'rniga qo'yishlar HTML-konteynerlarning atributlari qiymatlari

sifatida ishlatiladi. Masalan, foydalanuvchining shahsiy sahifasini aniqlovchi forma maydonining boshlang'ich qiymati sifatida joriy sahifaning URLi ko'rsatiladi:

```
<SCRIPT>
function l()
{
  str = window.location.href;
  return(str.length);
}
</SCRIPT>
<FORM><INPUT VALUE="&{window.location.href};" SIZE="&{l()};">
</FORM>
<SCRIPT>
<!-- Это комментарий ...JavaScript-код...// -->
</SCRIPT>
<BODY>
 Тело документа ...
</BODY>
</HTML>
```

HTML-shathlar bu yerda sahifaning berilgan bo'lagini eski brauzerlardagi HTML-parserlar tomonidan interpretatsiya qilinishidan saqlanish uchun qo'yilgan (yuqori boshqaruvdagilarda hali ham uchraydi). O'z navbatida HTML-sharhining ohiri JavaScript interpretatori tomonidan interpretatsiya qilinishidan himoya qilingan (satr boshidagi // belgisi). Bundan tashqari konteiner boshlanishidagi tegning LANGUAGE atributining qiymati sifatida *"JavaScript"* ko'rsatilgan. *JavaScript* ga muqobil sifatida ko'riluvchi VBScript keng qo'llaniluvchi tildan ko'ra ko'proq ekzotika hisoblanadi, shuning uchun bu atributni tushirib qoldirish mumkin – *"JavaScript"* qiymati o'z-o'zidan qabul qilinadi.

O'z –o'zidan ayonki, dokument sarlavhasida matnni hosil qilishni joylashtirish mantiqsizdir – u brauzer tomonidan namoyish qilinmaydi. Shuning uchun sarlavhada keyinchalik dokument ichida foydalaniladigan umumiy o'zgaruvchilar va funktsiyalar e'lon qilinadi. Bunda Netscape Navigatorda Internet Explorerga nisbatan talab qat'iyroqdir. Agar funktsiya sarlahada e'lon qilinmagan bo'lsa, dokument ichida bu funktsiya chaqirilsa, bu funktsiya aniqlanmaganligi to'g'risidaga habarni olish mumkin.

Funktsiyani joylashtirish va ishlatishga oid misolni ko'ramiz:

```
<HTML>
<HEAD>
<SCRIPT>
function time_scroll()
{
  d = new Date();
  window.status = d.getHours()+":"+d.getMinutes()+":"+d.getSeconds();
  setTimeout('time_scroll();',500);
}
</SCRIPT>
</HEAD>
<BODY onLoad=time_scroll()>
<CENTER>
```

```
<H1>Часы в строке статуса</H1>
</CENTER>
</BODY>
</HTML>
```

Shuni ta’kidlab o’tish kerakki, o’rniga qo'yishlar Internet Explorer 4.0 da ishlamaydi, shuning uchun ulardan foydalanishda ehtiyotkorroq bo’lish kerak. Brauzerga подстановки bo’lgan sahifani berishdan avval shu brauzer tipini tekshirib olish kerak.

O’rniga qo'yishlar bo’lgan holda brauzer (parser komponent) tomonidan HTML-dokumentni tahlil qilish vaqtida interpretator boshqaruvni o’z qo’liga oladi. Parser konteyner atributida &{..} konstruktsiyasini uchratishi bilan boshqaruvni *JavaScript* interpretatoriga beradi, u esa o’z navbatida bu kodni bajargandan so’ng boshqaruvni yana parserga qaytaradi. Shunday qilib bu operatsiya HTML-sahifaga grafikani yuklashga o’hshab ketadi.

Qo’yish(SCRIPT konteyneri- interpretatorni majburiy chaqirish). SCRIPT konteyneri bu o’rniga qo'yishlarni *JavaScript*-kod tomonidan dokument matnini hosil qilish imkoniyati darajasigacha rivojlantirilishidir. Bu ma’noda SCRIPTni qo’llanilishi Server Side Includesga o’hshab ketadi, ya’ni server tomonidan dokumentlarning sahifalarini hosil qilishga. Lekin bu yerda biz sal ilgarilab ketdik. Dokumentni tahlil qilish vaqtida HTML –parser SCRIPT konteineri boshlanishi tegini uchratgandan so’ng boshqaruvni interpretatorga beradi. Interpretator SCRIPT konteynerining ichidagi barcha kod bo’lagini bajaradi va SCRIPT konteyneri ohirini ko’rsatuvchi tegdan so’ng sahifa matnini qayta ishlash uchun boshqaruvni HTML-parserga qaytaradi.

SCRIPT konteyneri quyidagi 2 ta asosiy funktsiyani bajaradi:

1. HTML-dokument ichiga *kodni joylashtirish*;
2. brauzer tomonidan HTML-belgilarni *shartli hosil qilish*.

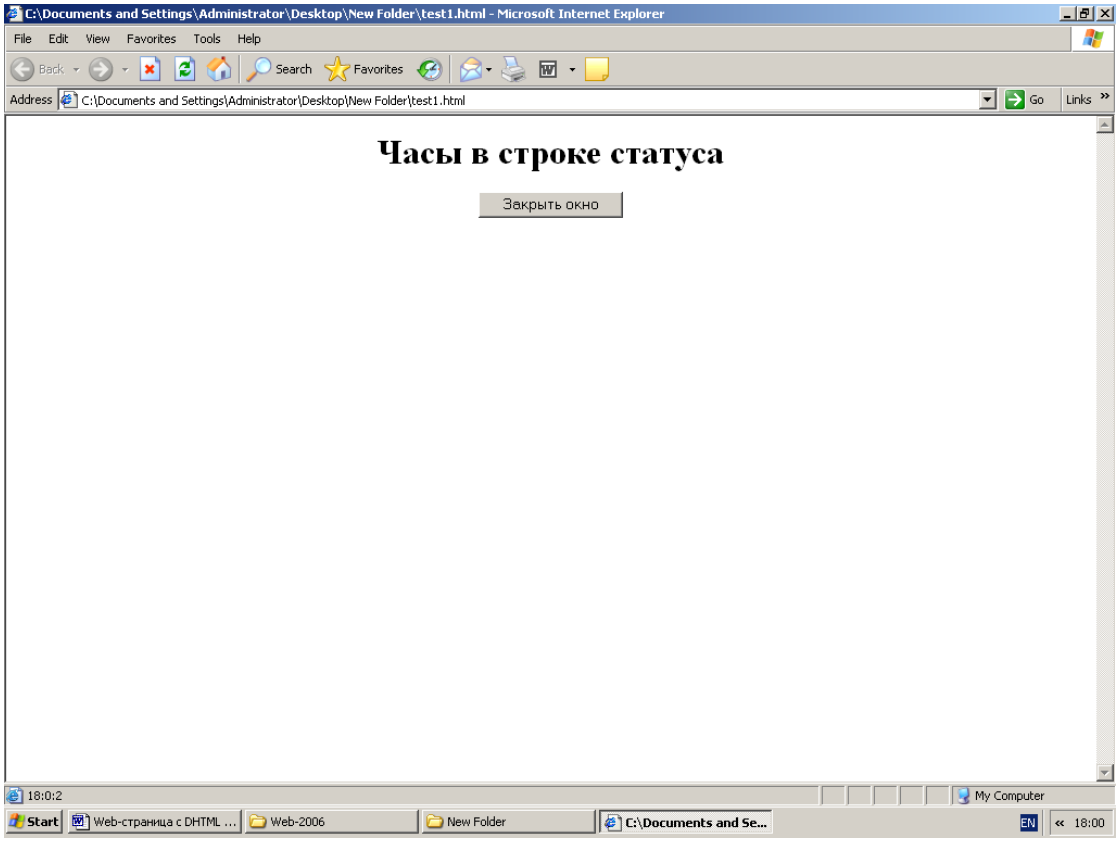
Birinchi funktsiya keyinchalik o’tishlar dasturi, hodisalarni qayta ishlovchilar va almashtirishlar sifatida ishlatish uchun o’zgaruvchilar va funktsiyalarni e’lon qilinishiga o’hshab ketadi. Ikkinchisi bu dokumentni yuklash yoki qayta yuklash paytida JavaScript-kod bajarilishi natijalarini o’rniga qo’yishdir.

HTML-dokument ichiga kodni joylashtirish. Hususan olganda bu yerda hech qanday ajralib turadigan har hillik yuq. Kodni dokument sarlavhasida, HEAD konteynerining ichida yoki BODY ning ichiga joylashtirish mumkin. So’nggni usul va uning hususiyatlari “Brauzer tomonidan HTML-bo’limlarini shartli hosil qilish” bo’limida ko’rib chiqiladi. Shuning uchun diqqatimizni dokument sarlavhasiga qaratamiz.

Sarlavhada kod SCRIPT konteinerining ichiga joylashtiriladi:

```
<HTML>
<HEAD>
<SCRIPT>
function time_scroll()
{
  d = new Date();
  window.status = d.getHours()+":"+d.getMinutes()+":"+d.getSeconds();
  setTimeout('time_scroll();',500);
}
</SCRIPT>
</HEAD>
<BODY onLoad=time_scroll()>
<CENTER>
```

```
<H1>Часы в строке статуса</H1>
<FORM>
<INPUT TYPE=button VALUE="Закрыть окно" onClick=window.close()>
</FORM>
</CENTER>
</BODY>
</HTML>
```



Bu misolda biz time_scroll() funktsiyani dokument sarlavhasida e'lon qildik, keyinchalik esa uni BODY konteineri boshining tegida hodisalarni qayta ishlovchi load sifatida chaqirdik (onLoad=time_scroll()).

O'zgaruvchilarni e'lon qilishga misol sifatida avvalgi-oyna tomonidan keyingi-oynaning statusini o'zgartirishni ko'ramiz.

Quyidagi funktsiya yordamida hosilaviy oynani e'lon qilish va keyin chaqirish orqali yaratamiz:

```
function sel()
{
  id = window.open("","example","width=500,height=200,status,menu");
  id.focus();
  id.document.open();
  id.document.write("<HTML><HEAD>");
  id.document.write("<BODY>");
  id.document.write("<CENTER>");
```

```
  id.document.write("<H1>Change text into child window.</H1>");
  id.document.write("<FORM NAME=f>");
  id.document.write("<INPUT TYPE=text NAME=t SIZE=20 MAXLENGTH=20 VALUE='Это
тест'>");
  id.document.write("<INPUT TYPE=button VALUE='Закрыть окно'
onClick=window.close()></FORM>");
  id.document.write("</CENTER>");
  id.document.write("</BODY></HTML>");
  id.document.close();
}
<INPUT TYPE=button VALUE="Изменить поле статуса в окне примера"
  onClick="id.defaultStatus='Привет'; id.focus();">
```

Keyingi oynani ochishda id o'zgaruvchiga oyna *ob'ekt*iga bo'lgan ko'rsatkich id=window.open() ni joylashtirdik. Endi biz uni Window sinfining *o'bekt*i identifikatori sifatida ishlatishimiz mumkin. Bizning holatda id.focus() dan foydalanish shart. "Misoldagi oynadagi status maydonini o'zgartirish" tugmasini bosish orqali e'tiborni asosiy oynaga o'tkaziladi. U ekran o'lchamida bo'lishi mumkin. Bunda asosiy oyna tomonidan berkitib turilgan ikkinchi oynada o'zgarishlar ro'y beradi. O'zgarishlarni ko'rish uchun diqqatni yana ikkinchi oynaga berish kerak. id o'zgaruvchi qandaydir funktsiya tashqarisida aniqlangan bo'lishi kerak. Bu holatda u oynaning *hususiyati* ga aylanadi. Agar biz uni keyingi oynani ochish funktsiyasi ichiga joylashtirsak, u holda hodisalarni qayta ishlovchi click orqali unga murojaat qila olmaymiz.

Brauzer tomonidan HTML-bo'limlarini shartli hosil qilish. Serverdan o'zimizning brauzer imkoniyatlariga yoki foydalanuvchiga moslashtirilgan sahifalarn olish doimo yoqimli. Bunday sahifalarni hosil qilishning faqatgina 2 imkoniyati bor: server tomonidan yoki bevosita klient tomonidan. *JavaScript* –kod klient tomonida bajariladi (aslida Netscape kompaniyasi serverlari *JavaScript*-kodni server tomonida ham bajarishi mumkin, lekin bu holda u LiveWire-kod nomini oladi; LiveConnect bilan adashtirilmasin), shuning uchun faqatgina klient tomonida hosil qilinishini ko'rib chiqamiz.

HTML-bo'limlrini hosil qilish uchun SCRIPT konteyneri dokument asosiy qismining ichiga joylashtiriladi. Oddiy misol- mahalliy vaqt sahifasiga moslashtirish:

```
<BODY>
...
<SCRIPT>
d = new Date();
document.write("<BR>");
document.write("Момент загрузки страницы:
"+d.getHours()+":"+d.getMinutes()+":"+d.getSeconds());
document.write("<BR>");
</SCRIPT>
...
</BODY>
```

JavaScriptda operatorlar, ifodalar, funktsiyalar

Operatorlar: arifmetik amallar, o'zlashtirish, orttirish, kamaytirish. Shartli ifodalar:

Qo'shish "+", ayirish "-", ko'paytirsh "*", bo'lish "/", qoldiqli bo'lish "%".
Bu ifodalar har qanday sonli ifodalarda uchrashi mumkin.

Shuni ko’rish mumkinki, JavaScriptdagi o’zlashtirish operatorlari C va Java dagilar bilan bir hil: “=”, “+=”, “-=”, “*=”, “/=”, “%=”.

x=y ifoda boshqa ko’pgina dasturlash tillaridagi kabi “x” o’zgaruvchiga “y” qiymatni berishni bildiradi.

Quyidagi operatorlar C dagi mos operatorlar bilan bir hil sintaktik qoidalarga ega:

y+=x bilan y=y+x ekvivalent
y-=x bilan y=y-x ekvivalent
y*=x bilan y=y*x ekvivalent
y/=x bilan y=y/x ekvivalent
y%=x bilan y=y%x – y ni x ga butun bo’lgandagi qoldiq- ekvivalent.
Shartli ifodalar quyidagi ko’rinishga ega

(shart)?qiymat1:qiymat2

Agar *shart* ning qiymatu true bo’sa, shartli ifodaning qiymati *qiymat1* ga teng bo’ladi, aks holda *qiymat2* ga teng bo’ladi. Shartli ifodalarni oddiy ifodalarni qo’llash mumkin bo’lgan hamma joyda qo’llash mumkin.

Misol:
a=(b<1)?0:(x-1)+c

Orttirish va kamaytirish operatorlari ham C dagi kabi sintaksisga ega: "x++", "++x", "x--", "--x".

Ifodalar:
y=x++ ikki o’zlashtirishga ekvivalent: y=x; y=y+1,
y=++x ikki o’zlashtirishga ekvivalent: x=x+1; y=x,
y=x-- ikki o’zlashtirishga ekvivalent: y=x; x=x-1,
y=--x ikki o’zlashtirishga ekvivalent: x=x-1; y=x.

Satrli operatorlar:

Satrlar bilan ishlash uchun bir nechta operatorlar mavjud:

"+" – satrlani qo’shish s1+s2 (konkatenatsiya) s1 satrning simvollari ketma-ketligidan keyin s2 satrning simvollari ketma-ketligi kelgan satrni beradi.

eval(s) – JavaScriptda qurilgan ichki funktsiya. U berilgan argument- bir yoki bir nechta JavaScript operatorlarini (nuqtali vergul bilan ajratilgan holda) o’z ichiga olgan s satr bilan berilgan kodni bajaradi. Berilgan funktsiyani nafaqat operatorni bajarish uchun, balki ifodalarni hisoblash uchun ham ishlatish mumkin. U berilgan koddagi ohirgi hisoblangan ifodaning qiymatini qaytaradi. *eval(s)* funktsiaysi foydalanuvchi tomonidan kiritish punktida kiritilgan qiymatni hisoblash va yana JavaScript dasturda bajarilayotgan kodni dinamik ravishda moslashtirish imkoniyatini beradi. U *parseInt* va *parseFloat* funktsiyalariga nisbatan umumiyroqdir.

parseFloat(s) – JavaScriptda ichki qurilgan funktsiyadir. U s satrdan birinchi belgidan son bo’lmagan dastlabki belgigacha oraliqdan sonni qidiradi (Float tipidagi). Agar son topilmasa NaN (“Not a Number”) qiymatini qaytaradi.

parseInt(s) – butun sonlar uchun huddi o’sha amalni bajaradi (Integer). Bunda avtomatik ravishda asos topiladi.

parseInt(s,n) – n asos buyicha huddio’sha amal bajariladi (n 2 dan 36 gacha). n=0 bo’lgan hol – bu *parseInt(s)* ni beradi. Bunda avtomatik ravishda asos topiladi.

Bitli o'zlashtirish operatorlar:

Bitli o'lashtirishning bir nechta operatorlari mavjud:

x<<=n x=(x<<n) ga ekvivalent — ikkilik ko'rinishdagi x sonda chapga n bit siljish;

x>>=n x=(x>>n) ga ekvivalent— ikkilik ko'rinishdagi x sonda ishora bitini saqlagan holda o'ngga n bit siljish (manfiy sonlarning qo'shimcha kodida birinchi bit 1 ga teng bo'ladi. Siljishdan keyin birinchi bitning o'rniga 1 yoziladi va son manfiyligicha qoladi);

x>>>=n x=x>>>n ga ekvivalent — ikkilik ko'rinishdagi x sonda birinchi bitiga 0 ni qo'ygan holda o'ngga n bit siljish (Siljishdan keyin birinchi bitning o'rniga 0 yoziladi va son musbat songa aylanadi);

Chat tomondagi barcha ifodalar (bizda u — "x") 32-bitli butun sonlarga aylantirib olinadi, keyin siljish bajariladi, keyin hosil bo'lgan son ifodaning- natijaning (bizda u yana "x") tipiga aylantiriladi va o'zlashtirish bajariladi.

Misollar:

1) 9<<2 ifoda 36ni beradi, chunki 1001 (9 sonining 2 lik ko'rinishdagi tasviri) da chapga 2 bit siljish 100100ni , ya'ni 36 ni beradi. Yetishmayotgan bitlar 0 lar bilan to'ldiriladi.

9>>2 ifoda 2 ni beradi, chunki 1001 (9 sonining 2 lik ko'rinishdagi tasviri)da o'ngga 2 bit siljish 10 ni, ya'ni 2 ni beradi. Yetishmayotgan bitlar 0lar bilan to'ldiriladi.

"&" — bitli AND — "VA";

"|" — bitli OR — "YOKI";

"^" — bitli XOR — "INKOR QILUVCHI YOKI".

Barcha operatsiyalar sonlarning 2 lik ko'rinishida bajariladi, lekin natija oddiy o'nli ko'rinishda qaytariladi.

Misollar:

15&9 ifoda 9 ni beradi, ya'ni (1111) AND (1001) ifoda 1001 ga teng;

15|9 ifoda 15 ni beradi, ya'ni (1111) OR (1001) ifoda 1111 ga teng;

15^9 ifoda 6 ni beradi, ya'ni (1111) XOR (1001) ifoda 0110 ga teng.

Mantiqiy ifodalar.

"&&" — mantiqiy AND — "VA";

"||" — mantiqiy OR — "YOKI";

"!" — mantiqiy NOT — "YO'Q".

Misol:

(a>b)&&(b<=10)||(a>10)

JavaScripda mantiqiy ifodalarni qisqartirilgan tekshirilishi deb ataluvchi amal doimo qo'llaniladi: B1&&B2 operandda B1=false bo'lgan holda B2 ni baholash bajarilmaydi va false qiymati qaytariladi. Shunga o'hshab B1||B2 B1=true holatda true deb baholanadi. Bunda mantiqiy ifodalarni tahlili chapdan o'ngga qarab olib boriladi va faqatgina to'la ifoda baholanib bo'lishi bilan natija qaytariladi. Shuning uchun agar B2 sifatida funktsiya bo'lsa, u chaqirilmaydi, va agar u aks ta'sirga ega bo'lsa, bu hatolikka olib kelishi mumkin.

Taqqoslash operatorlari:

"= =" -"teng";

">" -"katta";

"<" -"kichik";

">=" -"katta yoki teng";

"<=" -"kichik yoki teng";

"!=" -"teng emas".

Taqqoslash operatorlari nafaqat sonli ifodalarga, balki satrli ifodalarga ham qo'llanilishi mumkin. Bunda satrlar teng hisoblanadi qachonki ulardagi barcha simvollar ustma-ust tushsa va bir hil tartibda kelsa (bo'sh belgi ham simvol deb qaraladi). Agar satrlar turli hil uzunliklarga ega

bo’lsa, u holda uzunroq satr katta hisoblanadi. Agar ularning uzunliklari teng bo’lsa, u holda chapdan o’ngga brogan sari kattaroq nomerdagi simvolga ega bo’lgan satr katta hisoblanadi.

(a < b < c < … < z < A < … < Z).

Catrlarni qo’shish mumkin, agar S1="bu ", S2="mening satrim" bo’lsa, u holda S!+S2 "bu mening satrim" ni beradi.

Operatorlarning bajarilish tartibi (kichigidan boshlab; bir satrdagilarning tartibi bir hil):

"+=", "-=", "*=", "/=", "%=", "<<=", ">>=", ">>>=", "&=", "^=", "|=".

Operatorlarning tartibi. Quyida operatorlarning bajarilish tartibi o’sish tartibida berilgan:

shartli operator: "?:";
mantiqiy "YOKI": "||";
mantiqiy "VA": "&&";
bitli "YOKI": "|";
bitli "XOR": "^";
bitli "VA": "&";
tenglikka solishtirish: "= =", "!=";
solishtirish: "<", "<=", ">", ">=";
bitli siljish: "<<", ">>", ">>>";
qo’shish, ayirish: "+", "-";
ko’paytirish, bo’lish: "*", "/", "%" ;
inkor qilish, orttirish, kamaytirish: "!", "~", "++", "--";
qavslar: "( )", "[ ]".

Funktsiyalar:

JavaScriptda huddi C yoki Java dagi kabi protseduralar va protsedura-funktsiyalar funktsiya deb ataladi. Funktsiyalarni e’lon qilish quyidagilarni o’z ichiga oladi:

Qabul qilingan function so’zi;
Funktsiayning nomi;
Yumaloq qavs ichiga olingan va vergul bilan ajratilgan funktsiya argumentlari;
Figuraviy qavs ichiga olingan funktsiyaning asosiy qismi.

Ya’ni:

```
function myFunction(arg1, arg2, ...)
{
...
Operatorlar ketma-ketligi
...
}
```

Bu yerda myFunction — funktsiyaning nomi, arg1, arg2 — formal parametrlar ro’yhati

Misol:

```
function Factorial(n)
{
if((n<0)||(round(n)!=n))
{
alert("функция Factorial не определена при аргументе "+n);
return NaN;
}
else
{
result=(n*Factorial(n-1));
return result;
```

```
}
}
```

Funktsiya qabul qilingan return so’zi yordamida qiymatni qaytarmasligi mumkin.
Misol:

```
function Greeting(s)
{document.write("Hello,"+s+"!");
}
```

Funktsiyani chaqirish aniq bir qiymatlar orqali bajariladi.
Misol:

Factorial(2+1);— 6 ni qaytaradi,
Greeting("world");— ekranga "Hello, world!" satrini chiqaradi.

Har bir funktsiya, masalan, myFunction funktsiyasi argumentlari arguments deb nomlangan massiv ko’rinishida saqlanuvchi myFunction deb nomlanuvchi ob’ekt hisoblanadi, bunda uning argumentlariga quyidagicha murojaat qilinadi:

myFunction.arguments[i],

bu yerda i — argument nomeri (nomerlash 0 dan boshlanadi).

Funktsiayning sonli argument qiymatlari soni funktsiya e’lon qilingandagi formal parametrlari soniga teng yoki undan ortiq bo’lishi kerak. Bunda funktsiayni chaqirishdagi argumentlar soni myFunction.arguments.length maydonida saqlanadi va bu maydonning qiymatini qaytadan berish orqali dinamik ravishda o’zgartirish mumkin.

Misol: Dokumentda HTML formatida ro’yhatni chiqarish.

Bu yerda birinchi argument (ListType) tartiblangan ro’yhat uchun "o" yoki "O" qiymatni olish va tartiblanmagan ro’yhat uchun "u" yoki "U" qiymatni olishi mumkin. Keyin ro’yhatning elementlari keladi.

```
function myList(ListType)
{
document.write("<"+ListType+"L");
for(var i=1; i < myList.arguments.length; i=i+1)
{document.write("<LI>"+myList.arguments[i]);
}
document.write("</"+ListType+"L>");
}
```

HTML dokumentda bu funktsiyani chaqirish:

```
<script>
myList("0", "bir", 2, "3")
</script>
```

Quyidagi matnni chiqaradi:

bir
2
3

Shartli (nisbiy) o’tishli funktsiyalar

if shartli o’tish operatori
if operatori sintaksisining birinchi varianti:

```
if(shart)
{
tasdiq
}
```

Bu yerda tasdiq – operator yoki operatorlar ketma-ketlig

Bu holatda shartli operator quyidagicha ishlaydi: avval shart tekshiriladi va agar uning qiymati true ko'rinishida bo'lsa, *tasdiq* bajariladi. Aks holda if dan keyin keluvchi operator bajariladi.

if operatori sintaksisining ikkinchi varianti:

```
if(shart)
{
        tasdiq1
}
else
{
        tasdiq2
}
```

Bu holda avval *shart* tekshiriladi va agar uning qiymati "true" ko'rinishda bo'lsa, *tasdiq1* bajariladi, aks holda, ya'ni "false" bo'lsa, *tasdiq2* bajariladi.

Shartli o'tish operatorining ishlatilishiga misol:

```
function checkData()
{
if (document.form1.threeChar.value.length==3)
{return true;
}
else
{alert('Введите ровно 3 символа');
return false;
}
}
```

switch tanlash operatori:

```
switch (ifoda)
{
        qiymat1: operator1 break;
        qiymat2: operator2 break;
        //...
        default: operatorN break;
}
```

Tanlash operatori quyidagi tartibda ishlaydi: avval *ifoda* ning qiymati hisoblanadi, keyin uning *qiymat1* bilan tengligi tekshiriladi va agar u teng bo'sa, *operator1* bajariladi, keyin *ifoda* qiymatining *qiymat2* bilan tengligi tekshiriladi va agar u teng bo'lsa *operator2* bajariladi va hokazo. Agar *ifoda* qiymati hech bir qiymat: *qiymat1, qiymat2*, v.h. larga teng bo'lmasa, u holda o'z-o'zidan *operatorN* bajariladi.

Tsiklli funktsiyalar

for tsikli

for(boshlang'ich qiymat sektsiyasi;shart sektsiyasi; hisoblagich o'zgarishi sektsiyasi)
{
tasdiq
}

Bu sektsiyalardan har biri ham bo'sh bo'lishi mumkin. Boshlang'ich qiymat berish va hisoblagich o'zgarishi sektsiyalarida ifodalar ketma-ketligini vergul bilan ajratgan holda yozish mumkin. Tsiklni bajarilishi quyidagi tartibda bo'ladi. Birinchi boshlang'ich qiymat sektsiyasi bajariladi. Keyin shart tekshiriladi. Agar shartning qiymati true bo'lsa, u holda tsiklning asosiy qismi (*tasdiq*) bajariladi, keyin hisoblagich o'zgartirgich sektsiyasi bajariladi. Agar shartning qiymati false bo'lsa, tsikldan chiqiladi.

Misol:

```
function HowMany(SelectObject)
{
var numberSelected=0
for (i=0; i< SelectObject.options.leght; i++)
{
if (SelectObject.options[i].selected==true) numberSelected++;
}
return numberSelected;
}
```

for operatori ob'ektdagi barcha maydonlarni ko'rib chiqish uchun ishlatilishi mumkin (keyingi ob'ektli model haqidagi bo'limni qarang).

Sintaksis:

for(o'zgaruvchi in ob'ekt)
{
ifoda
}

Bunda ob'ektdagi ko'rsatilgan o'zgaruvchining barcha mumkin bo'lgan qiymatlari hosil qilinadi va ularning har biri uchun tasdiq bajariladi.

Misol: student sinfini va shu sinfning ob'ekti (ekzemplyar) Helen ni yaratamiz.

```
function student(name, age, group)
{
this.name=name;
this.age=age;
this.group=group;
}
function for_test(myObject)
{
for(var i in myObject)
{
document.write("i="+i+" => "+myObject[i]+"/n");
}
};
```

```
Helen=new student("Helen K.", 21, 409);
for_test(Helen);
```

Вывод на экран:

```
i=0 => Helen K.
i=1 => 21
i=2 => 409
```

while tsikli

```
while(shart)
{
    ifoda
}
```

while tsiklinig bajarilishi shartni tekshirishdan boshlanadi. Agar uning qiymati true ga teng bo'lsa, tsikl bajariladi, aks holda boshqaruv tsikldan keyingi operatorga beriladi.

while operatorining ishlatilishiga misol:

```
n1=10
n=0
x=0
while(n<n1)
{
n=n+1;
x=x+n;
}
```

Tsikllarni bajarilishini to'htatib qo'yuvchi operatorlar

for va while tsikllarining joriy bajarilishlarini to'htatish uchun *break* operatori ishlatiladi.

break operatorining ishlatilishiga misol:

```
function test(x)
{
var j=0;
var sum=0;
while(n<n1)
{
if(n==x)
{ sum=x;
break;
}
sum=sum=n;
n=n+1;
}
return sum;
}
```

continue operatori for va while larning ichida joriy iteratsiyani bajarilishini to'htatadi va keyingi iteratsiyaga o'tishni ta'minlaydi.

continue operatorining ishlatilishiga misol:

```
function test1(x)
{
var j=0;
while(n<n1)
{ if(n==x)
{ sum=x;
continue;
}
sum=sum=n;
n=n+1;
}
return sum;
}
```

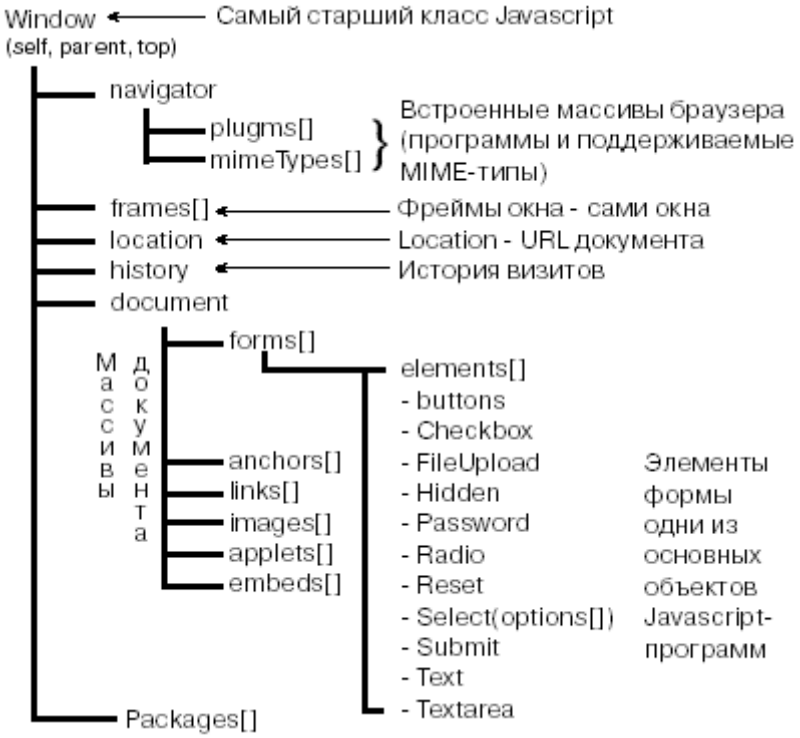
Bu yerda biz JavaScript tilining asoslari, ma’lumotlarning tiplari, ma’lumotlar ustida bajariladigan asosiy operatorlar va dasturning bajarilishini boshqarishni kabi tushunchalarni hamda JavaScript- kodlarni HTML-dokument ichiga joylashtirish usullarini ko’rib chiqdik.

1. HTML ob’ektlari

HTML ob’ektlari tushunchasi
ID-hossa
Barcha ob’ektlar uchun standartlar: hossa va hodisalar uchun metodlar
Window ob’ekti (oyna)
Document ob’ekti (dokument)
Ikkilamchi ob’ektlar

Bu yerda biz Javascriptda ob’ektlarga asoslangan dasturlash va dokument (Web-sahifa)ning ob’ektli modelini ko’rib chiqamiz.

Ob’ektlarga asoslangan dasturlash tillarida sinflar ob’ektlarining ierarhiyasi mavjudligi faraz qilinadi. JavaScriptda bunday ierarhiya Window sinfining ob’ektlaridan boshlanadi, ya’ni har bir ob’ekt u yoki bu oynaga biriktirilgan bo’ladi. Har bir ob’ektga yoki uning hossasiga murojaat qilish uchun bu ob’ektning yoki uning hossasining shu ob’ekt kirgan ierarhiyadagi eng yuqori turgan ob’ektdan boshlab to’liq yoki qisman nomi ko’rsatiladi.



Hoziroq shuni aytishimiz mumkinki, ob’ektli modelning keltirilgan sxemasi Netscape Navigator ning 4 va undan yuqori versiyalari hamda Microsoft Internet Explorer ning 4- va undan yuqori versiyalari uchun to’g’ri. Yana bir marta ta’kidlab o’tamizki, Internet Explorer va Netscape Navigator larda ob’ektli modellar turlichadir, keltirilgan sxema esa ularning umumiy qismlari asosida tuzilgan.

Umuman olganda *JavaScript* klassic ob’ektlarga asoslangan til hisoblanmaydi (uni yana yengillashtirilgan ob’ektli til deb ham atashadi). Unda meros qilib olish va polimorfizm yo’q. Dasturchi uzining *ob’ektlari* sinfini function operatori yordamida aniqlab olishi mumkin, lekin ko’pincha standart *ob’ektlar*, ularning konstruktorlaridan foydalanadi va sinflarning

destruktorlaridan umuman foydalanmaydi. Bu shu bilan tushintiriladiki, *JavaScript*-dasturlarning bajarilish doirasi joriy oynadan tashqariga chiqmaydi.

Ba'zida *JavaScript* dagi turli ob'ektlarda bir hil nomdagi *hossalar* aniqlangan bo'ladi. Bu holatda dasturchi qaysi *ob'ektning hossasidan* foydalanmoqchi ekanligini aniq ko'rsatishi kerak. Masalan, Window va Document lar location *hossasiga* ega bo'ladi. Faqat Window uchun bu Location sinfining ob'ekti, Document uchun esa qiymat sifatida yuklanayotgan dokumentning URLini oluvchi satrli literaldir.

Yana shuni hisobga olish kerakki, ko'pgina ob'ektlar uchun ob'ektlar hossalarining qiymatlarini oddiy o'zgaruvchilarga aylantirish uchun standart metodlar mavjud. Masalan, barcha ob'ektlar uchun simvollar satriga aylatirish uchun metod aniqlangan: toString().location bilan misolda agar satrli kontekstda window.location ga murojaat qilinsa, aylantirish o'z-o'zidan bajariladi va dasturchi buni payqamaydi:

```
<SCRIPT>
document.write(window.location);
document.write("<BR>");
document.write(document.location);
</SCRIPT>
```

Lekin baribir farq bor va anchagina. Huddi shu misolda satrli konstantaning uzunligini olamiz:

```
<SCRIPT>
w=toString(window.location);
d=toString(document.location);
h=window.location.href;
document.write(w.length);
document.write(d.length);
document.write(h.length);
</SCRIPT>
```

Shunga osongina ishonch hosil qilish mumkinki, URL tipidagi ob'ektning hossasiga murojaat qilinsa, location hossasi aynan shu tipdagi ob'ekt hisoblanadi, almashtirishdan keyin simvollar satri uzunligi boshqacha bo'ladi.

Barcha ob'ektlar uchun standartlar: hossa va hodisalar metodlari

Klient tomonida sahifalar ustidan boshqaruv mexanizmini yaratish uchun dokumentning ob'ektli modelidan foydalanish taklif qilingan. Modelning asosiy ma'nosi shundaki, har bir HTML-konteyner – bu quyidagi uchlik bilan harakterlanuvchi ob'ekt:

1. *hossalar,*
2. *metodlar,*
3. *hodisalar.*

Ob'ektli modelni sahifalar va brauzer orasidagi bog'lanish usuli sifatida tasavvur qilish mumkin. Ob'ektli model – bu brauzerning dasturiy ta'minotida ishtirok etuvchi va ro'y beruvchi, ular yordamida HTML kodi va sahifadagi stsenariy matnlari bilan ishlash qulay bo'lgan ko'rinishdagi ob'ektlar, metodlar va hodisalarning tasvirlanishidir. Biz uning yordamida brauzerga nima hohlashimizni bildirishimiz va unga mos ravishda ekrandagi sahifani o'zgartirishimiz mumkin.

Bir hil hossa, metodlar va hodisalar to'olamlariga ega bo'lgan ob'ektlar bir hil tipli ob'ektlar sinflariga birlashtiriladi. Sinflar – bu bo'lishi mumkin bo'lgan ob'ektlarning tariflanishidir.

Ob'ektlarning o'zlari esa faqatgina brauzer tomonidan dokumentni yuklagandan so'ng yoki dastur ishi natijasi sifatida paydo bo'ladi. Buni yuq ob'ektga murojaat qilmaslik uchun doimo esda tutish kerak.

Hossalar.

Ko'pgina HTML-konteynerlar atributlarga ega bo'ladi. Masalan, yakor konteyneri <A ...>...</A> uni gipermatnli o'tishga aylantiruvchi HREF atributiga ega bo'ladi:

<A HREF=tuit.htm>tuit</A>

Agar yakor konteyneri <A ...>...</A> ni ob'ekt sifatida ko'radigan bo'lsak, u holda HREF atributi "yakor" ob'ektining *hossasini* beradi. Dasturchi *atribut* qiymatini, shu bilan *ob'ektning hossasini* o'zgartirishi mumkin:

document.links[0].href="tuit.html";

Barcha atributlarda ham qiymatlarni o'zgartirish mumkin emas. Masalan, grafik rasmning balandligi va bo'yi sahifada rasmni tasvirlashda dastlabki yuklanganiga ko'ra aniqlanadi. Barcha keyingi rasmlar avvalgisiga ko'ra masshtablashtiriladi. Shuni ta'kidlab o'tish kerakki, Microsoft Internet Explorerda rasmning o'lchamlari o'zgartirilishi mumkin.

Tasvirning umumiyligi uchun JavaScriptda hossalar bilan HTML-bo'laklarda muqobili bo'lmagan ob'ektlar beriladi. Masalan, Navigator ob'ekti yoki brauzer oynasi deb ataluvchi, *JavaScript*dagi umuman eng yuqori ob'ekt hisoblanuvchi bajarish muhiti.

Metodlar.

JavaScript terminologiyasida *ob'ekt metodlari* deganda uning *hossalarini* o'zgartiruvchi funktsiyalar tushiniladi. Masalan, "dokument" ob'ekti bilan open(), write(), close() metodlari bog'langan. Bu metodlar dokumentni hosil qilish yoki uning mazmunini o'zgartirish imkoniyatini beradi. Quyidagi oddiy misolni ko'ramiz:

```
function hello()
{ id=window.open("","example","width=400, height=150");
  id.focus(); id.document.open();
  id.document.write("<H1>Salom!</H1>");
  id.document.write("<HR><FORM>");
  id.document.write("<INPUT TYPE=button VALUE='Oynani yopish' ");
  id.document.write("onClick='window.opener.focus();window.close();'>");
  id.document.close();
}
```

Bu misolda open() metodi dokumentga yozish oqimini ochadi, write() metodi bu yozishni amalga oshiradi, close() metodi dokumentga yozish oqimini yopadi. Bularning barchasi oddiy faylga yozishdagi kabi ro'y beradi. Agar oynada status maydonchasi bo'lsa (odatda unda dokumentni yuklash darajasi tasvirlanadi), yopilmagan dokumentga yozish oqimi bo'lgan holda unda huddi dokumentni yuklashdagi kabi yozishni davom etayotganligini ko'rsatuvchi to'rtburchak tasvirlanadi.

Hodisalar.

Metodlar va *hossalardan* tashqari ob'ektlar hodisalar bilan ham xarakterlanadi. Hususan, *JavaScript*da dasturlashning mazmuni ana shu hodisalarni qayta ishlovchilarni yozishdan iboratdir. Masalan, button tipidagi ob'ekt (button tipidagi INPUT konteyneri – "Tugmacha") bilan click hodisasi bo'lishi mumkin, ya'ni foydalanuvchi tugmachani bosishi mumkin. Buning uchun INPUT konteynerining atributlari click hodisasini qayta ishlovchi atribut – onClick bilan kengaytirilgan. Bu atributning qiymati sifatida HTML-dokumentning muallifi *JavaScript*da yozadigan hodisani qayta ishlash dasturi ko'rsatiladi:

<INPUT TYPE=button VALUE="Нажать" onClick="window.alert('Пожалуйста, нажмите еще раз');">

Hodisalarni qayta ishlovchilar shu hodisalar bog'langan konteynerlarda ko'rsatiladi. Masalan, BODY konteyneri butun dokumentni hossalarini aniqlaydi, shuning uchun butun dokumentni yuklanib bo'lish hodisasini qayta ishlovchi bu konteynerda onLoad atributining qiymati sifatida ko'rsatiladi.

Qat'iy qilib aytganda har qanday brauzer, hoh u Internet Explorer yoki Netscape Navigator, va yoki Opera bo'lsin, o'zining ob'ektli modeliga ega bo'ladi. Turli hil brauzerlar (va hatto bitta brauzerning turli hil versiyalari) ning ob'ektli modellari bir-birlaridan farq qiladi, lekin printsipial jihatdan bir hil strukturaga ega bo'ladi. Shuning uchun ularning har biriga alohida to'htalib o'tishdan ma'no yuq. Biz barcha brauzerlarga qollash mumkin bo'lgan umumiy yondashuvlarni va ayrim hollarda ular orasidagi farqlarni ko'rib chiqamiz.

Window (oyna) ob'ekti

Window ob'ektlar sinfi – bu JavaScriptdagi ob'ektlar ierarhiyasidagi eng yuqori sinf. Unga Window ob'ekti va Frame ob'ekti kiradi. Window ob'ekti brauzer-dastur oynasi bilan, Frame ob'ekti esa HTML-sahifa muallifi tomonidan FRAMESET va FRAME konteynerlaridan foydalanganda brauzer oynasi tomonidan hosil qilinuvchi va uning ichida joylashgan oynalar bilan bog'lanib ketadi.

JavaScriptda dasturlashda ko'pincha Window tipidagi ob'ektlarning quyidagi hossalari va metodlari ishlatiladi:

Hossalar	Metodlar	Hodisalar
status	open()	Hodisalar yuq
location	close()	
history	focus()	
navigator		

Window ob'ekti faqatgina oynani ochish vaqtidagina yaratiladi. Sahifani oynaga yuklashda paydo bo'ladigan barcha qolgan ob'ektlar Window ob'ektining hossalaridir. Shunday qilib turli hil sahifalarni yuklashda Windowning hossalari turlicha bo'lishi mumkin.

Status maydoni (holatlar paneli). **Status maydoni** — bu HTML-sahifa mualliflari JavaScriptdan foydalanishni boshlagan birinchi narsadir. Kalkulyatorlar, o'yinlar, matematik hisoblashlar va boshqa elementlar judayam sun'iy ko'rindi. Buning natijasida status maydonidagi yugurib yuruvchi satr Web dan foydalanuvchilarning diqqatini haqiqatdan torta oladigan durdona bo'lgan edi. Asta sekin uning ommaviyligi yuqola bordi. Yuguruvchi satrlar juda kam qo'llaniladigan bo'ldi, lekin status maydonchasini dasturlash ko'pgina Web-bo'g'imlarida uchraydi.

Status maydoni (status bar) deb HTML-sahifani tasvirlovchi sohaning shundoq ostidagi brauser oynasining quyi qismidagi o'rta maydonga aytiladi. Status maydonida brauzer holati (dokumentni yuklash, grafikani yuklash, yuklashni tugatish, appletni bajarish va h.) haqidagi ma'lumot tasvirlanadi. JavaScript dagi dastur bu maydon bilan oynaning uzgaruvchi hossasi kabi ishlash imkoniyatiga ega. Bunda amalda u bilan 2 ta turli hildagi hossalar bo'g'langan bo'ladi:

1. *window.status;*
2. *window.defaultStatus.*

Ular o'rtasidagi farq shundaki, amalda brauzer ba'zi bir hodisalar bilan bog'liq bo'lgan bir nechta holatlarda bo'ladi. Brauzer holati status maydonidagi habarda aks etadi. Umuman olganda faqatgina 2 ta holatlar mavjud: hech qanday hodisalar yuq (defaultStatus) va qandaydir hodisalar ro'y berayapti (status).

statusni dasturlaymiz. status hossasi sahifani oddiy yuklashdan farq qiluvchi hodisalar to'g'risidagi habarlarni tasvirlash bilan bog'liq. Masalan, sichqoncha kursori gipermatnli o'tish ustidan o'tayotganda, HREF atributida ko'rsatilgan URL status maydonchasida aks ettiriladi. Sichqoncha kursori gipermatnli o'tishdan holi bo'lgan maydonga o'tishi bilan status maydonchasida boshlang'ich habar tiklanadi (Document: Done). Bu texnika berilgan sahifada status va defaultStatus larni bayon qilishda amalga oshirilgan.

```
<A HREF=#status onMouseover="window.status='Jump to status description';return true;"
onMouseout="window.status='Status bar programming';return true;">window.status</A>
```

JavaScriptning dokumentatsiyasida ko'rsatilganki, mouseover va mouseout hodisalarni qayta ishlovchilar true qiymatini qaytarishi kerak. Bu brauzer boshlang'ich harakatlarni bajarmasligi uchun kerak bo'ladi. Tajriba shuni ko'rsatadiki, Netscape Navigator 4.0 true qiymati holatida ham juda yahshi ishlaydi.

defaultStatusni dasturlaymiz. defaultStatus hossasi hech qanday hodisa ro'y bermayotgan vaqtda status maydonida aks ettirilayotgan matnni aniqlaydi. Bizning dokumentda biz uni dokumentni yuklash vaqtida aniqladik:

```
<BODY onLoad="window.defaultStatus='Status bar programming';">
```

Bu habar sahifaning barcha komponentalari (matn, grafika, appletlar va h.) yuklab bo'lingandan keyin paydo bo'ladi. U dokumentni ko'rishda ro'y berishi mumkin bo'lgan hohlagan hodisadan qaytgandan keyin status maydonchasida qayta tiklanadi. Qizig'i shundaki, sichqonchani gipermatnli o'tishlardan holi bo'lgan sohalar bo'ylab harakatlantirilishi defaultStatus ni doimo aks ettirilishiga olib keladi.

location maydoni (adres satri). Adres maydonida yuklangan dokumentning URLi aks ettiriladi. Agar foydalanuvchi o'zicha qaysidir sahifaga o'tmoqchi bo'lsa (uning URLini terib), u buni location maydonida amalga oshiradi. Maydon brauzer oynasining yuqori qismida, instrumentlar panelidan quyida, lekin shahsiy tanlovlar panelidan yuqorida joylashgan.

Umuman olganda Location – bu ob'ekt. JavaScript versiyalaridagi o'zgarishlar tufayli Location quyi sinf sifatida Window sinfiga ham, Document sinfiga ham kiradi. Biz Location ni faqatgina window.location sifatida ko'rib chiqamiz. Bundan tashqari Location – bu Area va Link sinfining ob'ektlari kiruvchi URL sinfining quyi sinfi hamdir. Location URL ning barcha hossalarini meros qilib oladi va bu unga URL sxemasining istalgan qismiga kirishga imkon beradi.

Location ob'ektining xarakteristikalarini va ishlatish usullarini ko'rib chiqamiz:

- hossalar;
- metodlar;
- Locationni xarakterlovchi hodisalar yuq.

Ko'rib turibmizki, Location ob'ektining xarakteristikalari ro'yhati to'liq emas.

Hossalar. Aytaylik, brauzer quyida berilgan adresdagi sahifani aks ettirayotgan bo'lsin: http:// tuit.uz:80/r/dir/page?search#mark

U holda Location ob'ektining hossalari quyidagi qiymatlarni qabul qilishi mumkin:

```
window.location.href = http://tuit.uz:80/r/dir/page?search#mark
window.location.protocol = http;
window.location.hostname = tuit.uz;
window.location.host = tuit.uz:80;
window.location.port = 80
window.location.pathname = /r/dir/;
window.location.search = search;
window.location.hash = mark;
```

Metodlar. Location metodlari sahifani yuklashni va qayta yuklashni boshqarish uchun mo'ljallangan. Bu boshqaruv shuni bildiradiki, dokumentni qayta yuklash (reload) yoki yuklash (replace) mumkin. Bunda sahifalarni ko'rish trassasiga (history) ma'lumot kiritilmaydi:

```
window.location.reload(true);
window.location.replace('#top');
```

reload() metodi instrumentlar panelidagi Reload tugmachasini bosgandagi brauzer harakatini to'laligicha modellaydi. Agar bu metodni argumentsiz yoki unga true qiymat bergan holda chaqirilsa, brauzer dokumentning ohirgi o'zgartirilgan vaqtini tekshiradi va uni yoki keshdan (agar dokument o'zgartirilmagan bo'lsa) yoki serverdan yuklaydi. Bunday harakat Reload tugmasining oddiygina bosish bilan mos keladi. Agar argument sifatida false ko'rsatilsa, u holda brauzer dokumentni har qanday holatda ham serverdan yuklaydi. Bunday harakat Reload va Shift tugmachalrini birgalikda bosish bilan mos keladi (Reload+Shift).

replace() metodi bir sahifani ikkinchisi bilan shunday almashtirishga imkon beradiki, bu almashtirish HTML-sahifalarni ko'rish trassasi (history)da aks ettirilmaydi va instrumentlar panelidagi Back tugmachasini bosish orqali foydalanuvchi doimo dastlabki oddiy usulda (gipermatnli o'tish bo'yicha) yuklangan sahifaga qaytadi. Eslatib o'tamizki, Location hossasini o'zgartirishda ham sahifani qayta yuklash ro'y beradi, lekin bu holda bu o'tish haqidagi ma'lumot history ga kiritiladi.

Kirishlar tarihi (History). World Wide Web sahifalariga kirishlar tarihi (trassa) faoydalanuvchiga u bir necha minut (soat, kun) oldin ko'rgan sahifaga qaytish imkoniyatini beradi. Kirishlar tarihi JavaScriptda history sinfining ob'ektiga aylantiriladi. Bu ob'ekt foydalanuvchi ko'rgan va brauzer menyusidagi GO rejimini tanlagan holda olishi mumkin bo'lgan URL-sahifalar massivini ko'rsatadi. history ob'ekti metodlari shu massivdagi URLdan foydalangan holda sahifalarni yuklashga imkon beradi.

Brauzer havfsizligi bilan muammolar bo'lmasligi uchun History bo'yicha faqatgina URLning indeksi bo'yicha sayr qilish mumkin. Bunda URL matnli qator sifatida daturchiga berilmaydi. Ko'pincha bu ob'ekt bir nechta turli hil sahifalarga o'tishlar bo'lgan misollar yoki sahifalarda misol yuklanadigan sahifaga qaytish mumkin deb faraz qilgan holda foydalaniladi:

```
<FORM><INPUT TYPE=button VALUE="Назад" onClick=history.back()></FORM>
```

Berilgan kod bosish orqali oldingi sahifaga qaytishimiz mumkin bo'lgan "Orqaga" tugmasini aks ettiradi.

Brauzer tipi (Navigator ob'ekti). Brauzerlar o'rtasidagi "urush" tufayli (uni allaqachon Microsoft Internet Explorer foydasiga hal bo'ldi deb hisoblash mumkin) sahifani aniq bir ko'rish dasturiga moslashtirish muammosi vujudga keldi. Bunda ikki hil variant bo'lishi mumkin: server tomonidagi brauzer tipini aniqlash va klient tomonidagi brauzer tipini aniqlash. Ohirgi variant uchun JavaScriptda Navigator ob'ekti mavjud. Bu ob'ekt – Window ob'ektining hossasi.

Ko'rish dasturining tipini aniqlashga oddiy misolni ko'ramiz:

```
<FORM><INPUT TYPE=button VALUE="Тип навигатора"
onClick="window.alert(window.navigator.userAgent);"></FORM>
```

Tugmachani bosish bilan ogohlantirish oynasi aks ettiriladi. Unda tegishli brauzer HTML-sarlavhaga joylashtiradigan userAgent satri bo'ladi.

Bu satrni komponentalar bo'yicha bo'laklash mumkin, masalan:

```
navigator.appName = Microsoft Internet Explorer
navigator.appCodeName = Mozilla
navigator.appVersion = 4.0 (compatible; MSIE 5.5; Windows 98)
```

navigator.userAgent = Mozilla/4.0 (compatible; MSIE 5.5; Windows 98)

Navigator ob’ektini dasturlash tuqtai nazaridan qiziqarli bo’lgan bir necha hil qo’llash usullari mavjud. Masalan, Javani qo’llash mumkinligini tekshirish.

Bu imkoniyatni misolda ko’rsatamiz:

```
<SCRIPT>
document.write("<P ID=red>");
if(navigator.javaEnabled()==true)
  document.write("Ваша программа поддерживает исполнение Java-апплетов");
if(navigator.javaEnabled()==false)
  document.write("<FONT COLOR=red>Ваша программа не поддерживает исполнение Java-
апплетов</FONT>");
</SCRIPT>
</example>
```

Shunga o’hshab sizning brauzeringizda ishlarsa bo’ladigan grafik fayllar formatlarini hamtekshirish mumkin:

```
<SCRIPT>
if(navigator.mimeTypes['image/gif']!=null)
  document.write("Ваш браузер поддерживает GIF<BR>");
if(navigator.mimeTypes['image/tif']==null)
  document.write("Ваш браузер не поддерживает TIFF");
</SCRIPT>
```

Afsuski, bunday tekshiruv grafikani avtomatik yuklashni mavjudligini aniqlashga imkon bermaydi.

Oynalarni boshqarish. Oynalar bilan nimalar qilish mumkin? Ochish (yaratish), yopish (yo’qotish), uni boshqa barcha ochiq oynalar ustiga joylashtirish (diqqatni berish). Bundan tashqari oynaning va unga bo’ysungan ob’ektlar hossalarini boshqarish mumkin. Asosiy hossalarning bayoni “Brauzer oynasining hossalarini dasturlaymiz” bo’limida berilgan, shuning uchun asosiy e’tiborni oddiy va ko’proq ishlatiladigan oynalarni boshqarish metodlariga qaratamiz:

- *alert();*
- *confirm();*
- *prompt();*
- *open();*
- *close();*
- *focus();*
- *setTimeout();*
- *clearTimeout().*

Bu yerda faqat ikki metod: scroll() и blur() lar ko’rsatilmagan.

Birinchisi oynani berilgan pozitsiyaga joylashtirishga imkon beradi. Lekin uni oynaning koordinatalarini bilmagan holda ishlatish juda qiyin. Keyingisi esa oddiy ish hisoblanadi agarki qatlamlamlarni dasturlash texnologiyalari yoki CSS (Cascading Style Sheets) lar ishlatilmayotgan bo’lsa.

Ikkinchi metod diqqatni oynadan oladi. Bunda diqqatni qaerga qaratish umuman ma’lum bo’lmaydi. Yahshisi diqqatni yo’qotgandan ko’ra uni aniq maqsadli yo’naltirgan ma’qul.

window.alert()

alert() metodi ogohlantirish *oynasini* chiqarishga imkon beradi:

```
<A HREF="javascript:window.alert('Diqqat')">
So’rovni qaytaring!</A>
```

Hammasi juda oddiy, lekin shuni nazarda tutish kerakki, habarlar sistema fontlarida chiqariladi, shuning uchun rus tilidagi habarlarni olish uchun OSning mahalliylashtirilgan varianti bo’lishi kerak.

window.confirm()

confirm() metodi foydalanuvchiga u ma’qullashi yoki rad qilishi mumkin bo’lgan savollarni berishga imkon beradi:

```
<FORM>
<INPUT TYPE=button VALUE="Вы знаете JavaScript?"
onClick="if(window.confirm('Знаю все')==true)
{ document.forms[0].elements[1].value='Да'; }
else {
  document.forms[0].elements[1].value='Нет';
};"><BR>
</FORM>
```

Rus tilidagi habarlar uchun alert() metodi uchun bayon qilingan barcha cheklashlar confirm() metodi uchun ham o’rinli.

window.prompt()

prompt() metodi foydalanuvchidan informatsion oynaning kiritish maydoniga teriladigan qisqa matn satrini qabul qilishga imkon beradi:

```
<FORM>
<INPUT TYPE=button VALUE="Открыть окно ввода"
onClick="document.forms[1].elements[1].value=window.prompt('Введите сообщение');">
<INPUT SIZE=30>
</FORM>
```

Foydalanuvchi tomonidan kiritilgan satr istalgan o’zgaruvchi tomonidan o’zlashtirilishi mumkin va keyin JavaScript-dasturda ishltish mumkin.

window.open()

Oynaning bu metodida atributlar boshqa ob’ektlarnikiga qaraganda ko’proq. Open() metodi yangi oynalarni yaratishga mo’ljallangan. Umumiy holda uning sintaksisi quyidagi ko’rinishda bo’ladi:

```
open("URL","window_name","param,param,...", replace);
```

bu yerda: URL — yangi oynaga yuklanadigan sahifa, window_name — A va FORM konteynerlaridagi TARGET atributida ishlatish mumkin bo’lgan oyna nomi.

Parametrlar	Ishlatilishi
replace	*Oynani* ochish vaqtida History massiviga yozishni boshqarishga imkon beradi
param	Parametrlar ro’yhati
width	*Oynaning* piksellardagi kengligi
height	*Oynaning* piksellardagi balandligi
toolbar	Brauzerning sistema tugmalari bo’lgan

	oynani yaratadi
location	location maydonli *oynani* yaratadi
directories	Foydalanuvchining tanlovlari menyusili *oynani* yaratadi
status	Status maydonli *oynani* yaratadi
menubar	Menyuli *oynani* yaratadi
scrollbar	Yugurdak polosali *oynani* yaratadi
resizable	O'lchamini o'zgartirsa bo'ladigan *oynani* yaratadi

Quyidagi misolni keltiramiz:

```
<FORM>
<INPUT TYPE=button VALUE="Простое окно"
onClick="window.open('about:blank','test1','directories=no,height=200,location=no,menubar=no,res
izable=no,scrollbars=no, status=no,toolbar=no,width=200');">
<INPUT TYPE=button VALUE="Сложное окно"
onClick="window.open('about:blank','test2','directories=yes,
height=200,location=yes,menubar=yes,resizable=yes,
scrollbars=yes,status=yes,toolbar=yes,width=200');">
</FORM>
```

"oddiy oyna" tugmasiga bosib, quyidagi parametrli *oynani* hosil qilamiz:

- directories=no – menyusiz *oyna*
- height=200 – balandligi 200 px
- location=no – location maydoni yuq
- menubar=no - menyusiz
- resizable=no – o'lchamini o'zgartirish mumkin emas
- scrollbars=no – yugurdak polosasi yo'q
- status=no – status satri yo'q
- toolbar=no – brauzerning sistema tugmachalri yo'q
- width=200 – kengligi 200

"murakkab oyna" tugmasini bosib, quyidagi *oynani* olamiz:

- directories=yes – menyuli oyna
- height=200 - balandlik 200 px
- location=yes - location maydoni mavjud
- menubar=yes – menyu mavjud
- resizable=yes – o'lchamni o'zgartirish mumkin
- scrollbars=yes – yugurdak polosasi mavjud
- status=yes – status satri mavjud
- toolbar=yes – brauzerning sistema tugmachali mavjud
- width=200 - kenglik 200

window.close()

close() metodi – bu open() metodining aks tomonidir. U *oynani* yopish imkoniyatini beradi. Ko'pincha aynan qaysi *oynani* yopish kerak degan savol tug'iladi.

Agar joriy oynani yopish kerak bo'lsa, u holda:

```
window.close();
self.close();
```

Agar bosh *oynani*, yani joriy oyna ocilgan oynani yopish kerak bo'lsa, u holda:

```
window.opener.close();
```

Agar ihtiyoriy *oynani* yopish kerak bo'lsa, u holda avval uning identifikatorini olish kerak bo'ladi:

```
id=window.open();
...
id.close();
```

Ohirgi misoldan ko'rinib turibdiki, oynani nomi bo'yicha emas (TARGET atributining qiymati bu yerda ahamiyatga ega emas), balki ob'ektning ko'rsatkichi bo'ycha yopiladi.

window.focus()

focus() metodi diqqatni u ishlatilgan *oynaga* qaratish uchun ishlatiladi. Diqqatni berish oynani qaysi vaqtda tanlash holatlarini eslamasdanoq, oynani ochishda ham, yopishda ham juda foydali. Misol ko'ramiz.

Oynani ochamiz va uni yopmasdan turib yana shu nomdagi, lekin boshqacha matnli oynani ochamiz. Yangi oyna asosiy oynaning ustida paydo bo'lmaydi, chunki diqqat unga berilmadi. Endi oyna ochilishini takrorlaymiz, faqat bu safar diqqatni berish orqali:

```
function myfocus(a)
{
id = window.open("","example","scrollbars,width=300,height=200");
//oynani ochaniz va unga ko'rsatkich bo'lgan o'zgaruvchini kiritamiz
//agar shu nomli oyna mavjud  bo'lsa, yangi oyna yaratilmaydi,
//faqatgina shu oynaga yozish uchun oqim ochiladi
if(a==1)
{
id.document.open();
//yaratilgan oynaga yozish uchun oqim ochamiz
id.document.write("<CENTER>>Oynani birinchi marta ochdingiz");
//Bu oqimga yozamiz
}
if(a==2)
{
id.document.open();
id.document.write("<CENTER>Oynani ikkinchi marta ochdingiz");
}
if(a==3)
{
id.focus();
//diqqatni beramiz, keyin oldingi holdagi amallarni bajaramiz
id.document.open();
id.document.write("<CENTER>Oynani uchinchi marta ochdingiz");
}
id.document.write("<FORM><INPUT TYPE=button
onClick='window.close();' VALUE='Oynani yopish'></CENTER>");
id.document.close();
}
```

Yangi oynani eski (bosh) oynadan turib yozayotganligimiz uchun yangi ob'ektga ko'rsatkich sifatida id o'zgaruvchining qiymatidan foydalanamiz.

window.setTimeout()

setTimeout() metodi bajarilishi ikkinchi argumentda ko'rsatilgan millisekundlarcha keyingi qolditiluvchi yangi hisoblash oqimini ochish uchun foydalaniladi:

idt = setTimeout("JavaScript_код",Time);

Bu funktsiyaning tipik qo'llanilishi – ob'ektlar hossalarini avtomatik o'zgarishini tashkil qilish. Masalan, forma maydonida soatni qo'yish mumkin:

```
var flag=0;
var idp=null;
function myclock()
{
if(flag==1)
{
d = new Date();
window.document.c.f.value = d.getHours()+":"+d.getMinutes()+":"+d.getSeconds();
}
idp=setTimeout("myclock();",500);
}
function flagss()
{
if(flag==0) flag=1; else flag=0;
}
...
<FORM NAME=c>
Joriy vaqt:<INPUT NAME=f size=8><INPUT TYPE=button VALUE="Start/Stop"
onClick="flagss();myclock();">
</FORM>
```

Shuni nazarda tutish kerakki, oqim har doim, hattoki soat to'htagan holatda ham paydo bo'laveradi. Agar u flag o'zgaruvchisining 1 ga teng qiymatidagina yaratilganida edi, u holda flag ning 0 ga teng qiymatida yo'qolgan bo'lar edi va u tugmachani bosishda soatlar to'htab turishda davom etgan bo'lardi.

window.clearTimeout

clearTimeout() metodi setTimeout() metodi tufayli hosil qilingan oqimni yo'q qilishga imkon beradi. Aniqki, uni ishlatilishi hisoblash qurilmalari resurslarini effektivroq taqsimlash imkoniyatini beradi. Bu metodni soatlar bilan bo'lgan misolda foydalanish uchun, biz funktsiyani va formani moslashtirishimiz kerak:

```
var idp1 = null;
function start()
{
d = new Date();
window.document.c1.f1.value = d.getHours()+":"+d.getMinutes()+":"+d.getSeconds();
idp1=setTimeout("start();",500);
}
function stop()
{
clearTimeout(idp1);idp1=null;
}
```

```
...
<FORM NAME=c1>
Joriy vaqt:<INPUT NAME=f1 size=8>
<INPUT TYPE=button VALUE="Start" onClick="if(idp1==null)start();">
<INPUT TYPE=button VALUE="Stop" onClick="if(idp1!=null)stop();">
</FORM>
```

Berilgan misolda soatlarni to'xtatish uchun clearTimeout() metodi ishlatiladi. Bunda ko'plab oqimlarni paydo b'lmasligi uchun oqim ob'ektiga bo'lgan ko'rsatkichning qiymati tekshiriladi.

Document ob'ekti (window.document). Oynalarni yaratish.

Brauzerda yangi oynalarni yaratish – JavaScriptning juda katta imkoniyati. Siz yangi oynaga yangi dokumentlarni yuklashingiz (masalan, huddi o'sha HTML dokumentlarini) yoki (dinamik ravishda) yangi materiallarni *yaratishingiz* mumkin. Avval yangi oynani qanday ochish mumkinligini, keyin esa bu oynaga qanday qilib HTML-sahifani yuklash mumkinligini va nihoyat, uni qanday qilib yopish mumkinligini ko'rib chiqamiz.

Quyida keltirilgan script brauzerning yangi oynasini ochadi va unga qandaydir web-sahifani yuklaydi:

```
<html>
<head>
<script language="JavaScript">
<!-- hide
function openWin() {
  myWin= open("abc.html");
}
// -->
</script>
</head>
<body>
<form>
<input type="button" value="Yangi oynani ochish" onClick="openWin()">
</form>
</body>
</html>
```

Keltirilgan misolda yangi oynaga open() metodi yordamida *abc.html* sahifasi yoziladi.

Shunki ko'rishingiz mumkinki, siz oyna yaratish protsessining o'zi ustidan nazorat o'rnata olasiz. Masalan, siz yangi oyna status satri, instrumentlar paneli yoki menyuga ega bo'lishi kerakligini ko'rsata olasiz. Bundan tashqari siz oynaning o'chamini ham bera olasiz. Masalan, keyingi skriptda 400x300 piksel o'lchamli oyna ochiladi. U status satriga ham, instrumentlar paneliga ham, menyuga ham ega bo'lmaydi.

```
<html>
<head>
<script language="JavaScript">
<!-- hide
function openWin2() {
  myWin= open("abc.html", "displayWindow",
    "width=400,height=300,status=no,toolbar=no,menubar=no");
}
```

```
// -->
</script>
</head>
<body>
<form>
<input type="button" value="Yangi oynani ochish " onClick="openWin2()">
</form>
</body>
</html>
```

Ko’rib turganingizdek, oynaning hossalarini quyidagi satrda hosil qilamiz *"width=400,height=300,status=no,toolbar=no,menubar=no"*.

Yana shu narsaga e’tibor beringki, siz bu satrda bo’sh o’rin simvollarini joylashtirishingiz kerak emas.

Siz boshqara olishingiz mumkin bo’gan oyna hossalarining ro’yhati:

directories	yes|no
height	Piksellar soni
location	yes|no
menubar	yes|no
resizable	yes|no
scrollbars	yes|no
status	yes|no
toolbar	yes|no
width	Piksellar soni

JavaScript tilining 1.2 versiyasida bir nechta yangi hossalar qo’shildi (yani Netscape Navigator 4.0 da). Siz bu hossalardan Netscape 2.x, 3.x yoki Microsoft Internet Explorer 3.x lar uchun materiallar tayorlashda foydalanmasligingiz kerak, chunki bu brauzerlar 1.2 JavaScript tilini tushinmaydi. Oynalarning yangi hossalari:

alwaysLowered	yes|no
alwaysRaised	yes|no
dependent	yes|no
hotkeys	yes|no
innerWidth	*Piksellar soni* (width ning o’rniga)
innerHeight	*Piksellar soni* (height ning o’rniga)
outerWidth	*Piksellar soni*
outerHeight	*Piksellar soni*
screenX	*Piksellar soni*
screenY	*Piksellar soni*
titlebar	yes|no
z-lock	yes|no

Siz bu hossalarning ma’nolarini JavaScript 1.2 tilining bayonidan topishingiz mumkin. Bundan keyin ulardan ayrimlariga tushintirishlar va ishlatishga misollar keltiramiz.

Masalan, endi bu yangi hossalardan foydalangan holda yangi oyna ekrannning qaysi yerida joylashish kerakligini aniqlashingiz mumkin. JavaScript tilining eski versiyasida siz buni amalga oshira olmas edingiz.

Oynaning nomi. Ko'rib turganingizdek, oyna ochish vaqtida biz uchta argumentdan foydalanishimiz kerak:

```
myWin= open("abc.html", "displayWindow",
"width=400,height=300,status=no,toolbar=no,menubar=no");
```

Ikkinchi argument nima uchun kerak? Bu oynaning nomi. Oldinroq biz uni target parametrida qanday ishlatilganini ko'rib o'tdik. Shunday qilib agar siz oynaning nomini bilsangiz, u holda unda quyidagi yozuv yordamida yangi sahifani yuklashingiz mumkin

```
<a href="abc.html" target="displayWindow">
```

Bunda siz mos oynaning nomini ko'rsatishingiz kerak (agar bunday nomdagi oyna mavjud bo'lmasa, shu nomdagi yangi oyna yaratiladi).

E'tibor beringki, *myWin* – bu yangi oyna nomi emas. Lekin shu o'zgaruvchi yordamidagina siz oynaga kirishingiz mumkin. Va u oddiy o'zgaruvchi bo'lganligidan uning qo'llanish sohasi u aniqlangan skript holos. Shu bilan birga oyna nomi (berilgan holatda u *displayWindow*) – bu brauzerning istalgan oynasida foydalanish mumkin bo'lgan o'ziga hos identifikator.

Oynalarni yopish. Siz yana JavaScript tili yordamida oynalarni yopishingiz ham mumkin. Buni amalga oshirish uchun sizga close() metodi kerak bo'ladi. Keling, oldinroq ko'rganimiz kabi yangi oyna ochamiz. Va unga navbatdagi sahifani yuklaymiz:

```
<html>
<script language="JavaScript">
<!-- hide
function closeIt() {
  close();
}
// -->
</script>
<center>
<form>
<input type=button value="Yopish" onClick="closeIt()">
</form>
</center>
</html>
```

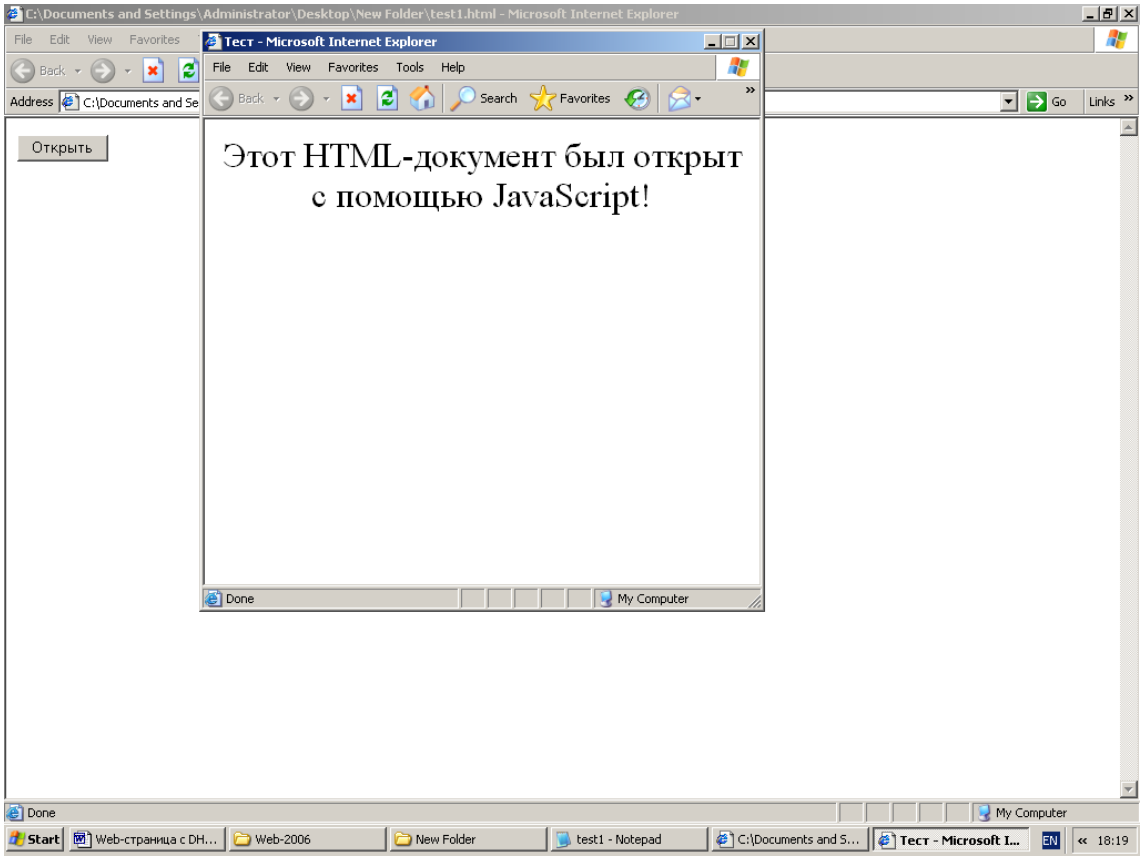
Endi agar siz yangi oynadagi tugmachani bossangiz, u yopiladi. open() va close() –bu window ob'ektining metodlaridir. Biz shuni yodda tutishimiz kerakki, oddiygina qilib *open()* va *close()* shaklida emas, balki *window.open()* va *window.close()* ko'rinishida yozish kerak. Ammo bizning holatda *window* ob'ektini tushirib qoldirishimiz mumkin - siz agar bu o'bektning metodlaridan birini chaqirmoqchi bo'lsangiz (va u faqatgina shu ob'ekt uchungina mumkin), u holda siz window prefiksini yozishingiz shart emas.

Dokumentni dinamik ravishda yaratish. Endi biz dokumentlarni dinamik ravishda yaratish kabi JavaScript ning ajoyib imkoniyatini ko'rib chiqishga tayyormiz. Yani JavaScriptda tuzilgan skript ning o'ziga yangi HTML-sahifalarni yaratishga ruhsat berish mumkin. Bundan tashqari shu yul bilan webning VRML-ko'rinishlar va h. kabi boshqa dokumentlarini ham yaratish mumkin. Qulaylik uchun bu dokumentlarni alohida oyna yoki freimga joylashtirish mumkin.

Avval biz oddiy HTML-dokument yaratamiz va uni yangi oynada namoyish qilamiz.

Quyidagi skriptni ko'ramiz.

```
<html>
<head>
<script language="JavaScript">
<!-- hide
function openWin3() {
  myWin= open("", "displayWindow",
    "width=500,height=400,status=yes,toolbar=yes,menubar=yes");
  // keyinchalik bosib chiqarish uchun document ob’ektini ochish
  myWin.document.open();
  // yangi dokumentni ochish
  myWin.document.write("<html><head><title>Тест");
  myWin.document.write("</title></head><body>");
  myWin.document.write("<center><font size=+3>");
  myWin.document.write("Этот HTML-документ был открыт ");
  myWin.document.write("с помощью JavaScript!");
  myWin.document.write("</font></center>");
  myWin.document.write("</body></html>");
  // dokumentni yopish - (lekin oynani emas!)
  myWin.document.close();
}
// -->
</script>
</head>
<body>
<form>
<input type=button value="Открыть" onClick="openWin3()">
</form>
</body>
</html>
```



Keling winOpen3() funktsiaysini ko'rib chiqaylik. Aniqki, biz avval brauzerning yangi oynasini ochamiz. Open() funktsiaysining birinchi argumenti – bo'sh satr (“”), bu esa shuni bildiradiki, biz bu holatda aniq bir URL adresini ko'rsatishni hohlamaymiz. Brauzer faqatgina mavjud dokumentni qayta ishlamasligi kerak – JavaScript qo'shimcha yangi dokument yaratishi kerak.

Skriptda biz yangi *myWin* o'zgaruvchini aniqlaymiz. Va uning yordamida biz yangi oynaga kirib bora olamiz. Shunga e'tibor beringki, bu holda bu maqsad uchun oyna nomi (*displayWindow*) dan foydalana olmaymiz.

Yangi oynani ochganimizdan keyin, document ob'ektini yozish uchun navbat keladi. Bu quyidagi komanda yordamida amalga oshiriladi:

```
// keyinchalik bosmaga chiqarish uchun document ob'ektini ochish
myWin.document.open();
```

Bu yerda biz open()ga- document ob'ektining metodiga murojaat qilamiz. Lekin u window ob'ektining open() metodi bilan bir hil narsa emas! Bu komanda yangi oyna ochmaydi – u bosib chiqarish uchun documentni tayorlaydi. Bundan tashqari biz *document.open()* ning oldiga *myWin* ni yangi oynaga yozish imkoniyatini olish maqsadida qo'yishimiz kerak.

Skriptning keyingi satrlarida *document.write()* ni chaqirish yordamida yangi dokumentning matni hosil qilinadi:

```
// yangi dokumentni hosil qilish
myWin.document.write("<html><head><title>On-the-fly");
myWin.document.write("</title></head><body>");
myWin.document.write("<center><font size=+3>");
myWin.document.write("Этот HTML-документ был открыт ");
myWin.document.write("с помощью JavaScript!");
```

```
 myWin.document.write("</font></center>");
 myWin.document.write("</body></html>");
```

Ko’rinib turibdiki, biz dokumentga HTML tilining oddiy teglarini yozamiz. Yani amalda biz HTML bo’laklarini hosil qilamiz! Bunda HTML teglarining istalganidan foydalanish mumkin.

Buni yakunlagandan keyin biz dokumentni yana yopishimiz kerak. Bu quyidagi komanda orqali amalga oshiriladi:

```
 // dokumentni yopish - (oynani emas!)
 myWin.document.close();
```

Nafaqat dokumentlarni dinamik ravishda yaratish mumkin, balki ularni o’z hohishicha u yoki bu freimlarga joylashtirish ham mumkin. Masalan, *frame1* va *frame2* nomli ikki framega ega bo’lsak va f*rame2* ga yangi dokumentni hosil qilish kerak bo’lsa, u holda *frame1*ga quyidagini yozib qo’yish yetarli:

```
parent.frame2.document.open();
parent.frame2.document.write("Здесь находиться ваш HTML-код");
parent.frame2.document.close();
```

Ikkilamchi ob’ektlar. JavaScriptda ichki qurilgan ob’ektlarni ko’rib chiqamiz. Foydalanuvchi tomonidan yaratilgan ob’ektlar va brauzerning ob’ektli modelini (bu model to’g’risida boshqa bir bo’limda hikoya qilinadi) tashkil qilgan ob’ektlardan farq qilib, ichki qurilgan ob’ektlar hohlagan kontekstda – hoh u Microsoft Internet Explorer yoki Netscape Navigator bo’lsin- chaqirilishi mumkin.

JavaScript tiliga bo’lgan asosiy talablarni belgilab beruvchi ECMAScript (ECMAScript Language Specification, Standard ECMA-262 ni qarang) spetsifikatsiyasiga ko’ra tilda quyidagi ob’ektlar amalga oshirilgan bo’lishi kerak: Global, Object, Function, Array, String, Boolean, Number, Math va Date.

Array, Boolean, Date, Function, Math, Number va String ichki qurilgan ob’ektlarini ko’rib chiqamiz.

Array ob’ekti

JavaScript tilida massivlarni yaratish uchun ichki qurilgan ma’lumotlar tipi yo’q, shuning uchun bunday masalalarni hal qilishda Array ob’ektidan foydalaniladi. U massivlarni birlashtirish, tartiblash va o’rin almashtirish uchun metodlarga ega, yana massivning o’lchamini aniqlash imkoniyati ham mavjud.

Massiv – bu nomi va tartib raqami (indeksi) bo’yicha chqiriladigan qiymatlarning tartiblashgan to’plamidir. Masalan, dasturda har biri o’z tartib raqamiga ega bo’lgan habarlar to’plami –allMsg dan tuzilgan massivni yaratish mumkin. Shunday qilib, allMsg[0] birinchi habar, allMsg[1] –ikkinchi habar bo’ladi va hokazo.

Array ob’ektini hosil qilish uchun ikki bir birini o’rnini olishi mumkin bo’lgan usulni qo’llsh mumkin.

New konstruktorini chaqiring va massiv o’lchami (undagi elementlar soni)ni bering. Massivni to’ldirish keyinroq ro’y beradi.

Quyidagi misolni ko’rib chiqamiz.

```
<html>
<head><title>Тест JavaScript.</title>
<script language="JavaScript">
// создание нового массива
  allStr = new Array(5);
```

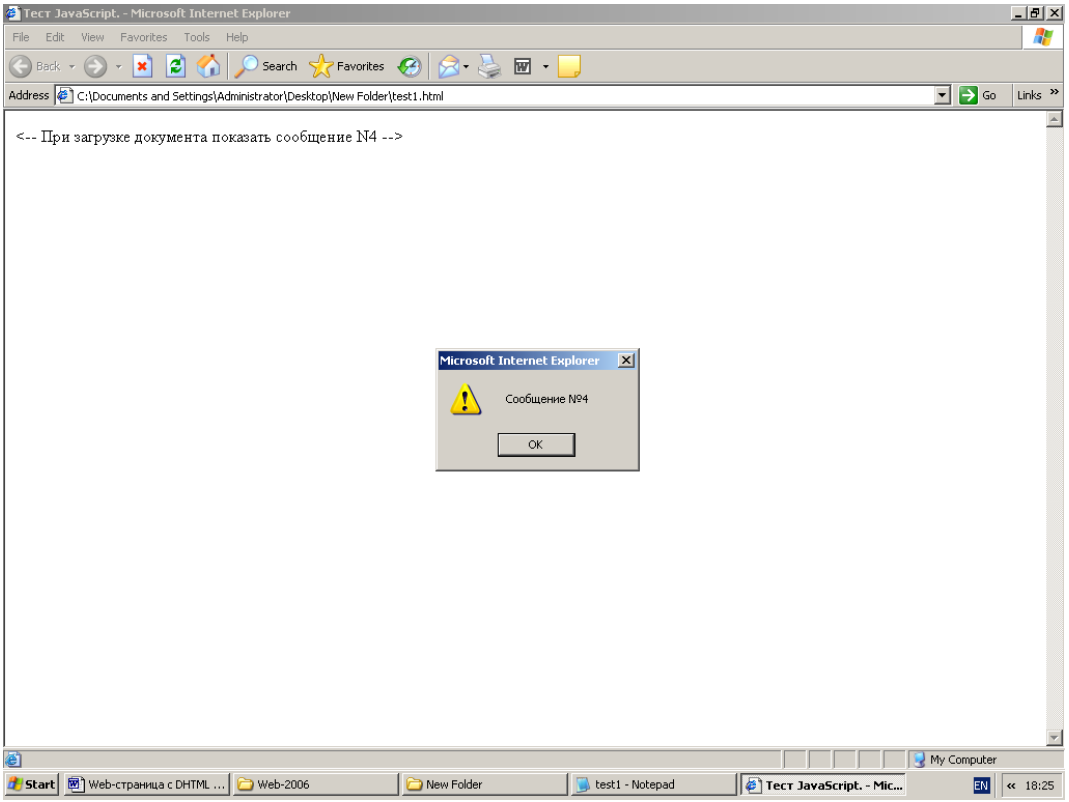
```
// заполнение массива
  allStr[0] = "Сообщение №1";
  allStr[1] = "Сообщение №2";
  allStr[2] = "Сообщение №3";
  allStr[3] = "Сообщение №4";
  allStr[4] = "Сообщение №5";
// функция для отображения элемента массива
  function showMsg(ndx)
  {
    alert(allStr[ndx]);
  }
</script>
</head>
<-- При загрузке документа показать сообщение №4 -->
<body onLoad="showMsg(3);">
</body>
</html>
```

Yuqorida keltirilgan misolda 5 ta elementdan tashkil topgan massiv yaratiladi, keyin u to'ldiriladi.

New konstruktorini chaqirasiz va massivning barcha elementlarining qiymatlarini berasiz. Massiv o'lchami bu holda oshkora holda ko'rsatilmaydi.

Quyidagi misolni ko'ramiz.

```
<html>
<head><title> Тест JavaScript.</title>
<script language="JavaScript">
// создание нового массива и его заполнение
  allStr = new Array("Сообщение №1", " Сообщение №2", " Сообщение №3",
    " Сообщение №4", " Сообщение №5");
// функция для отображения элемента массива
  function showMsg(ndx)
  {
    alert(allStr[ndx]);
  }
</script>
</head>
<-- При загрузке документа показать сообщение N4 -->
<body onLoad="showMsg(3);">
</body>
</html>
```



Bu yerda massiv elementlarining qiymatlari new konstruktorini chaqirish vaqtidayoq bevosita beriladi.

Array ob'ektining metodlari

Array ob'ekti quyidagi metodlarga ega.

Metod	Bayoni
join	Massivning barcha elementlarini satrga birlashtiradi
reverse	Massivdagi elementlar tartibini o'zgartiradi – birinchi element ohirgi elementga, ohirgisi esa birinchi elementga aylanadi
Sort	Massiv elementlarini tartiblaydi

Array ob'ekti metodlarining ishlatilishlarini ko'rib chiqamiz. Aytaylik bir nechta elementga ega bo'lgan massiv va ikkita –showAll va showElement funktsiaylari berilgan bo'lsin. Birinchi, massivning barcha elementlarini namoyish qiluvchi funktsiya tsikl ichida massivning berilgan elementini namoyish qiluvchi showElement funktsiaysini chaqiradi:

```
<html>
<head><title>Тест JavaScript.</title>
<script language="JavaScript">
 myArray = new Array("Отец", "Мать", "Брат", "Сестра", "Другой");
 function showElement(ndx)
 {
  alert(myArray[ndx]);
 }
 function showAll()
```

```
 {
  for( i = 0; i <= myArray.length-1; i++)
  {
   showElement(i);
  }
 }
</script>
</head>
<body onLoad="showAll();">
</body>
</html>
```

Agar yuqorida keltirilgan faylni yuklasak, har birida myArray massivining bitta elementi - Mother, Father, Sister, Brother va Uncle tasvirlangan habarlar paneli ketma-ketligini ko'rishimiz mumkin.

Quyidagi kod yozilgan test funktsiaysini yaratamiz:

```
 function test()
 {
  alert(myArray.join());
 }
```

Va <body> tegini o'zgartiramiz:

```
 <body onLoad="test();">
```

Joint metodi massiv elemetlarini ajratib turuvchini berish mumkin bo'lgan majburiy bo'lmagan parametrga ega. Ko'rsatilmagan holda "," belgisi ishlatiladi. Masalan

```
 function test()
 {
  alert(myArray.join(" _|_ "));
 }
```

reverse metodi massive elementlarini o'rin almashtirishlari uchun foydalaniladi. test funktsiaysiga shu metodni chaqirilishini qo'shamiz:

```
 function test()
 {
  myArray.reverse();
  alert(myArray.join(";"));
 }
```

Massivning birinchi elementi ohirgi o'rinni egalladi, ikkinchisi- ohiridan oldingi va hokazo.

Sort metodi massiv elementlarini tartiblash uchun foydalaniladi. test funktsiyasiga shu metodni chqirilishini qo'shamiz:

```
 function test()
 {
  myArray.sort();
  alert(myArray.join(";"));
 }
```

Ko'p o'lchamli massivlarni yaratish.

Array ob’ekti ko’p o’lchamli massivlarni yaratishga imkon beradi. Quyida ko’p o’lchamli massivni yaratishga misol keltirilgan.

```
<html>
<head><title>Тест JavaScript. </title></head>
<body>
<center>
<font size=5><b>Многомерный массив</b></font><p>
<script language="JavaScript">
  a = new Array(4);
  for( i=0; i < 4; i++)
  {
    a[i] = new Array(4);
    for( j=0; j < 4; j++)
    {
      a[i][j] = "["+i+","+j+"]";
    }
  }
  for( i=0; i < 4; i++)
  {
    str = "Строка "+i+":";
    for( j=0; j < 4; j++)
    {
      str += a[i][j];
    }
    document.write( str, "<br>");
  }
</script>
</center>
</body>
</html>
```

Yuqorida keltirilgan dasturda to’rtta elementdan tashkil topgan massiv hosil qilinadi: uning elementlarining har biri ham to’rtta elementdan iborat massivdir. Har bir elementga element indeksini beruvchi i,j juftikning qiymatlari kiritiladi. Keyin tsikl ichida berilgan massiv elementlari namoyish qilinadi.

Boolean ob’ekti

Ichki qurilgan Boolean ob’ekti mantiqiy bo’magan qiymatlarni mantiqiy qiymatlarga aylantirish uchun foydalaniladi. Boolean ob’ektining namunasini hosil qilish uchun new konstruktori ishlatiladi**:**

```
bfalse = new Boolean(false);
btrue  = new Boolean(true);
```

valueOf metodi mantiqiy o’zgaruvchining mantiqiy ko’rinishdagi qiymatini ko’rsatadi, toString metodi esa satrli ko’rinishdagi qiymatini ko’rsatadi. Bu metodlarni ishlatilishiga misol quyida keltirilgan.

```
<html>
```

```
<head><title>Тест JavaScript.</title></head>
<body>
<script language="JavaScript">
  // ikki mantiqiy o’zgaruvchi yaratamiz
  bfalse = new Boolean(false);
  btru = new Boolean(true);
  // ularning qiymatlarini chqaramiz (mantiqiy qiymatlar)
  document.write(bfalse.valueOf()+"<br>");
  document.write(btrue.valueOf()+"<br>");
  // satrli qiymatlarni chiqaramiz
  document.write(bfalse.toString()+"<br>");
  document.write(btrue.toString()+"<br>");
</script>
</body>
</html>
```

Date ob’ekti.

Date ob’ekti va uning metodlaridan skript dasturlarda sana va vaqt bilan ishlashda foydalaniladi. Bu ob’ekt sanani o’rnatish, uning qiymatini olish va turli hil almashtirishlarni bajarish uchun katta sondagi metodlar to’plamiga ega. Date ob’ekti hossalarga ega emas.

Shuni ta’kidlab o’tamizki, JavaScriptda sana huddi Javadagi singari saqlanadi – u 1 yanvar 1970 yildan buyon o’tgan millisekundlar sonini ko’rsatadi. Shunday qilib undan oldingi sanalar qo’llanmaydi.

Date ob’ektining namunasini hosil qilish uchun Date konstruktoridan foydalaniladi:

```
MyDate = new Date([parametrlar]);
```

Quyidagi parametrlarni ko’rsatish mumkin:

- parametrlarsiz – namuna joriy sana va vaqtni ko’rsatadi. Masalan, today = new Date();
- sanani quyidagi formatda ko’rsatuvchi satr: "Oy, kun, yil vaqt:minutlar:sekundlar". Masalan, someDate = new Date("May 15, 1996"). Agar soat, minut yoki sekundlar soni ko’rsatilmagan bo’lsa, ularning qiymatlari 0 ga teng deb olinadi;
- yil, oy va kunlarning butun sondagi qiymatlari to’plami. Masalan, otherDay = new Date(96, 4, 15);
- yil, oy, kun, soat, minut va sekundlarning butun sondagi qiymatlari to’plami. Masalan, sameDay = new Date(96, 4, 15, 15, 30, 0);

Turli hil parametrlarning ishlatilishiga oid misol quyida keltirilgan.

```
<html>
<head><title> Тест JavaScript.</title></head>
<body>
<center>
<script language="JavaScript">
  today = new Date();
  document.write("today="+today+"<br>");
  someDate = new Date("May 16, 1996");
  document.write("someDate="+someDate+"<br>");
  otherDay = new Date( 96, 4, 15);
  document.write("otherDay="+otherDay+"<br>");
  sameDay = new Date( 96, 4, 16, 15, 30, 0);
```

```
 document.write("sameDay="+sameDay+"<br>");
</script>
</center>
</body>
</html>
```

Date ob'ektining metodlari

Date ob'ekti tomonidan sana va vaqtni boshqarish uchun ishlatiladigan metodlarni quyidagi kategoriyalarga bo'lish mumkin:

- *o'rnatish metodlari (set)* – Date ob'ekti namunalarida sana va vaqtni o'rnatish uchun metodlar;
- *aniqlsh metodlari (get)* - Date ob'ekti namunalaridan sana va vaqtni olish uchun metodlar
- *shakl almashtirish metodlari (to)* – sana va vaqtni satrga aylantirish uchun metodlar;
- *sanani qayta ishlash uchun metodlar*.

O'rnatish va aniqlash metodlari sekund, minut, soat, oy kuni, hafta kuni, oy va yil qiymatlarining olish/ o'zgartirsh uchun foydalanilishi mumkin. Masalan, hafta kunini aniqlash mumkin bo'lgan getDay metodi mavjud, lekin setDay metodi mavjud emas, chunki hafta kuni avtomatic ravishda o'rnatiladi.

Bu metodlarning barchasi quyidagi jadvalda keltirilgan butun sonli qiymatlardan foydalanadi.

Qiymati	Oralig'i
Minut va sekundlar soni	0..59
Soatlar soni	0..23
Hafta kuni	0..6
Sana	1..31
Oy	0..11 (Yanvar..Dekabr)
Yil	1900 yildan boshlab

Quyidagi misolni ko'ramiz. Aytaylik, siz sanani 15 may 1996 yil deb berdingiz. Quyida Date ob'ektining bir nechta metodlarining ishlatilishi namoyish qilingan.

```
<html>
<head><title>JavaScript </title></head>
<body>
<center>
<p>
<script language="JavaScript">
  someDate = new Date( "May 15, 1996" );
  document.write("someDate="+someDate+"<br>");
  document.write("getDay  ="+someDate.getDay()+"<br>");
  document.write("getMonth="+someDate.getMonth()+"<br>");
  document.write("getYear ="+someDate.getYear()+"<br>");
</script>
</center>
</body>
</html>
```

getTime va setTime metodlari sanalarni taqqoslash uchun qulay. getTime metodi millisekundlar sonini beradi. Masalan, bu yildagi qolgan kunlar sonini qanday aniqlash quyida keltirilgan.

```
<html>
<head><title>JavaScript </title></head>
<body>
<center>
<br><br><br>
<script language="JavaScript">
  today = new Date();
  // sanani berish
  endYear = new Date("December 31, 1990");
  // yilni o'zgartirish
  endYear.setYear(today.getYear());
  // kundagi millisekundlar sonini hisoblash
  msPerDay = 24 * 60 * 60 * 1000;
  // kunlar sonini olish
  daysLeft = (endYear.getTime() - today.getTime()) / msPerDay;
  // yahlitlash
  daysLeft = Math.round(daysLeft);
  // ko'rsatish
  document.write("Number of days left in the year: "+daysLeft);
</script>
</center>
</body>
</html>
```

Parse metodi vaqtning satrli tasvirini shakl almashtirish uchun foydalanilishi mumkin. Masalan,

```
someDate = new Date();
someDate.setTime(Date.parse("May 15, 1996"));
```

Date ob'ektini ko'rishni kichkina bir amaliy misol bilan yakunlaymiz. Biz joriy vaqtni ekranga chiqruvchi dasturni yaratamiz – hohlasangiz uni siz o'z HTML-sahifangizga joylashtirishingiz mumkin. Vaqt "tugma" interfeis elementining sarlavhasi sifatida chiqariladi. Mashq sifatida dasturni shunday to'ldirishni taklif qilamanki, tugmani bosishingiz bilan joriy sana namoyish qilinsin.

Vaqtni namoyish qiluvchi dastur matni quyida keltirilgan.

```
<html>
<head><title>JavaScript </title>
<script language="JavaScript">
  function showTime()
  {
    // joriy vaqtni olamiz
    var time = new Date();
    // soatlar sonini bilib olamiz
    var hour = time.getHours();
```

```
   // minutlar sonini bilib olamiz
   var minute = time.getMinutes();
   // sekundlar sonini bilib olamiz
   var second = time.getSeconds();
   // soatlarni ko'rsatamiz
   var temp = hour;
   // minutlarni ko'rsatamiz ( 0 ni  0-9ga qo'shib)
   temp += ((minute < 10) ? ":0" : ":") + minute;
   // sekundlarni ko'rsatamiz (0 ni  0-9ga qo'shib)
   temp += ((second < 10) ? ":0" : ":") + second;
   // namoyish qilamiz
   document.forms[0].clock.value = temp;
   // har bir sekundni ko'rsatamiz
   id = setTimeout( "showTime()", 1000);
  }
</script>
</head>
<body onLoad="showTime();">
<center>
<font size=5><b>
 JavaScript Date & Time Demo
</b></font>
<form name="DateDemo">
 <input type="button" name="clock" value="xxxxxxxx">
</form>
</center>
</body>
</html>
```

Sanani namoyish qiladigan funktsiya quyidagi ko'rinishda b'lishi mumkin:

```
 function showDate()
 {
   //joriy sanani olamiz
   var D = new Date();
   // hafta kunini bilib olamiz
   var DOW = D.getDay();
   // sanani bilib olamiz
   var Day = D.getDate();
   // oyni bilib olamiz
   var Month = D.getMonth();
   // yilni bilib olamiz
   var Year = D.getYear();
   // namoyish qilamiz
   temp = Day + "/" + Month + "/" + Year;
   document.forms[0].clock.value = temp;
 }
```

DOW o’zgaruvchisida hafta kunining nomeri saqlanadi. Yuqorida ko’rib o’tilgan Array ob’ektidan foydalangan holda, hafta kunlari nomlarining massivini hosil qilishimiz mumkin:

```
dayNames = new
      Array("воскресенье","понедельник","вторник","среда",
                  "четверг","пятница","суббота");
```

и добавить в предпоследней строке функции showDate следующее:

```
temp = Day + "/" + Month + "/" + Year + dayNames[DOW];
```

Function ob’ekti

Ichki qurilgan Function ob’ekti satrga funktsiya deb baholash mumkin bo’lgan JavaScriptdagi kodni berishga imkon beradi. Function ob’ektining namunasini yaratish uchun new konstruktoridan foydalaniladi:

```
var newBgColor = new Function("c", "document.bgColor=c");
```

Bu funktsiyadan foydalanishga misol quyida berilgan.

```
<html>
<head><title>JavaScript </title></head>
<body>
<center>
<script language="JavaScript">
 var newBgColor = new Function("c", "document.bgColor=c");
 function fnDemo()
 {
  newBgColor("#a7b7c7");
 }
</script>
<form>
<input type="button" value="bgColor" onClick="fnDemo();">
</form>
</center>
</body>
</html>
```

Yana bir misolni ko’rib chiqamiz. Aytaylik, bizga ikki argumentning qiymatlarini ko’paytiradigan funktsiya kerak bo’lsin. Bu funktsiyani myMult deb ataymiz:

```
var myMult = new Function("x", "y", "return x*y");
```

Undan quyidagicha foydalanish mumkin:

```
<html>
<head><title>JavaScript </title></head>
<body>
<center>
<script language="JavaScript">
 var myMult = new Function("x", "y", "return x*y");
 document.write("10*20="+myMult(10, 20));
</script>
</center>
```

```
</body>
</html>
```

Biz hattoki kichkina forma hosil qilishimiz va uni turli hil sonlarning ko'paytmasini topish uchun foydalanishimiz mumkin:

```
<html>
<head><title>JavaScript 12.97</title></head>
<body>
<center>
<script language="JavaScript">
  var myMult = new Function("x", "y", "return x*y");
  function calc()
  {
    document.forms[0].sum.value =
      myMult(document.forms[0].x.value, document.forms[0].y.value);
  }
</script>
</center>
<form>
<input type="text" name="x"   value=10  size=4>
<input type="text" name="y"   value=10  size=4>
<input type="text" name="mul" value=" " size=4>
<input type="button" value="calc" onClick="calc();">
</form>
</body>
</html>
```

Math ob'ekti

Ichki qurilgan Math ob'ekti turli hil matematik konstantalarni olish va matematik funktsiyalarni bajarish uchun hossa va metodlarga ega. Quyida shu qiymatlarni ekranga chiqaradigan dasturning matni berilgan.

```
<html>
<head><title>JavaScript </title></head>
<body bgcolor="#a7b7c7">
<center><font size=4><b>Math</b> Object Properties</font><p>
<script language="JavaScript">
  s = "<TABLE BORDER=2>" +
      "<TR><TD><B>E</B></TD><TD>"       + Math.E       + "</TD></TR>" +
      "<TR><TD><B>LN2</B></TD><TD>"     + Math.LN2     + "</TD></TR>" +
      "<TR><TD><B>LN10</B></TD><TD>"    + Math.LN10    + "</TD></TR>" +
      "<TR><TD><B>LOG2E</B></TD><TD>"   + Math.LOG2E   + "</TD></TR>" +
      "<TR><TD><B>LOG10E</B></TD><TD>"  + Math.LOG10E  + "</TD></TR>" +
      "<TR><TD><B>PI</B></TD><TD>"      + Math.PI      + "</TD></TR>" +
      "<TR><TD><B>SQRT1_2</B></TD><TD>" + Math.SQRT1_2 + "</TD></TR>" +
      "<TR><TD><B>SQRT2</B></TD><TD>"   + Math.SQRT2   + "</TD></TR>" +
      "</TABLE>";
  document.write(s);
</script>
</center>
```

```
</body>
</html>
```

Keyingi jadvalda Math ob’ektida mavjud bo’lgan metodlar sanab o’tilgan.

Metod	Bayoni
abs	Argument qiymatining modulini beradi
sin, cos, tan	Standart trigonometrik funktsiyalar, argumentlar radianda beriladi
acos, asin, atan	Teskari trigonometrik funktsiyalar, radiandagi qiymatlarni beradi
exp, log	Eksponenta va natural logarifm, asos -e
ceil	Argumentga teng yoki undan katta bo’lgan butun sonni beradi
floor	Argumentga teng yoki undan kichik butun sonni beradi
min, max	Ikki argumentlardan kichigi yoki kattasini beradi
pow	Argument darajasini beradi
round	Argumentni unga eng yaqin butun songacha yaxlitlaydi
sqrt	Argumentning kvadrat ildizini beradi

Math ob’ektining ko’plab sondagi hossa va metodlaridan foydalanganda, masalan, qandaydir hisoblashlarni bajarganda, with operatoridan foydalanish qulay:

```
 with( Math )
 {
   a = PI * r*r;
   b = floor(c);
   x = r * sin( theta );
   y = r * cos( theta );
   z = round( b );
 }
```

Number ob’ekti

Ichki qurilgan Number ob’ektining hossalari maksimal qiymat, “not-a-number” (NaN) va cheksizliklar kabi tipdagi turli hil sonli o’zgarmaslarning qiymatlarini olish uchun foydalaniladi. Quyida Number ob’ekti hossalari qiymatlarini chiqaradigan dasturning matni berilgan.

```
<html>
<head><title>JavaScript </title></head>
<body bgcolor="#a7b7c7">
<center><font size=4><b>Number</b> Object Properties</font><p>
<script language="JavaScript">
 s = "<TABLE BORDER=2>" +
    "<TR><TD><B>MAX_VALUE</B></TD><TD>"        +
      Number.MAX_VALUE        + "</TD></TR>" +
```

```
    "<TR><TD><B>MIN_VALUE</B></TD><TD>"        +
      Number.MIN_VALUE        + "</TD></TR>" +
    "<TR><TD><B>NaN</B></TD><TD>"             +
      Number.NaN              + "</TD></TR>" +
    "<TR><TD><B>NEGATIVE_INFINITY</B></TD><TD>" +
      Number.NEGATIVE_INFINITY + "</TD></TR>" +
    "<TR><TD><B>POSITIVE_INFINITY</B></TD><TD>" +
      Number.POSITIVE_INFINITY + "</TD></TR>" +
    "</TABLE>";
  document.write(s);
</script>
</center>
</body>
</html>
```

String ob'ekti

JavaScript tilida string ko'rinishidagi ichki qurilgan ma'lumotlar tipi yo'q. Shunga qaramasdan String ob'ekti va uning metodlaridan foydalangan holda siz skript dasturlarda satrlar bilan ishlashingiz mumkin. String ob'ekti satrlar ustida bajariladigan juda ko'plab metodlarga va satrning uzunligini aniqlashga imkon beruvchi bitta hossa (length)ga ega.

String ob'ektining namunasi new konstruktorini chaqirish orqali hosil qilinadi:

```
myString = new String("myString Object");
```

String ob'ekti ikki hil tipdagi metodlarni namoyish qiladi. Birinchi tipga shakl almashtirishlar yoki satrlar ustida boshqa operatsiyalarni bajaradigan metodlar kiradi: substring, toUpperCase va h. shuni ta'kidlab o'tamizki, HTML-versiyasi satrini beruvchi funktsiyalar ECMAScript standartiga kirmaydi.

String ob'ekti metodlari quyidagi jadvalda sanab o'tilgan.

Metod	Bayoni
anchor	anchor- HTML-elementini yaratadi
big, blink, bold, fixed, italics, small, strike, sub, sup	Mos HTML-elementni yaratadi
charAt	Satrning ko'rsatilgan o'rnida joylashgan belgini beradi
indexOf, lastIndexOf	Mos ravishda satrdagi qism-satrning o'rnini yoki satrdagi ohirgi qism-satrning o'rnini beradi
link	HTML-o'tishni yaratadi
split	String ob'ektini satrlarni qism-satrlarga aylantirgan holda satrlar massiviga bo'ladi
substring	Ko'rsatilgan qism-satrni beradi
toLowerCase, toUpperCase	Satrdagi belgilarni mos ravishda quyi va yuqori registrdagi belgilarga aylantiradi

Bu ob'ekt metodlaridan foydalanishga bir nechta misollar ko'rib o'tamiz.

```
<html>
<head><title>JavaScript </title></head>
<body bgcolor="#a7b7c7">
<center>
<script language="JavaScript">
  //yangi satrni yaratamiz
  var s = new String('Netscape Navigator');
  //qism-satrni olamiz
  var n = s.substring(0,8);
  //satr va qism satrni ko’rsatamiz
  document.write("string="+s+"<br>substring="+n);
  //quyi registrga aylantiramiz
  var l = s.toLowerCase(s);
  // ko’rsatamiz
  document.write("<br>"+l);
  // yuqori registrga aylantiramiz va ko’rsatamiz
  document.write("<br>"+s.toUpperCase(s));
</script>
</center>
</body>
</html>
```

Bular shakl almashtirish metodlari edi. Endi HTML-versiyalar satrlarini beruvchi metodlarni ko’rib chiqamiz:

```
<html>
<head><title>JavaScript </title></head>
<body bgcolor="#a7b7c7">
<center>
<script language="JavaScript">
  // yangi satr yaratamiz
  var s = new String('Netscape Navigator');
  //  <b> va </b> larni qo’shamiz
  document.write(s.bold(s)+"<br>");
  //  <i> va </i> larni qo’shamiz
  document.write(s.italics(s)+"<br>");
  // <tt>  va </tt> larni qo’shamiz
  document.write(s.fixed(s));
  // <sub>  va  </sub> larni qo’shamiz
  document.write(s.sub(s)+" ");
  // <sup> va  </sup> larni qo’shamiz
  document.write(s.sup(s)+"<br>");
  //  <strike>  va  </strike> larni qo’shamiz
  document.write(s.strike(s)+"<br>");
</script>
</center>
</body>
</html>
```

Satrni o’tishga aylantirish uchun link metodini chaqirish kerak:

```
var s = new String('Netscape Navigator');
document.write(s.link("http://www.netscape.com"));
```

s o’zgaruvchi uchun dokumentda HTML-matn quyidagicha ko’rinishga ega bo’ladi:

```
<A HREF="http://www.netscape.com">Netscape Navigator</A>
```

2. Kollektsiyalar

Kollektsiyalar to'g'risida umumiy tushuncha
All, Forms, Images kollektsiyalari
Boshqa kollektsiyalar

Agar JavaScriptda dasturlashni tarihiy nuqtai-nazardan turib qaralsa, hossa va metodlar yaratilgan birinchi ob'ektlar forma maydonlari bo'ldi. Odatda FORM konteineri va forma maydonlari nomlangan bo'ladi:

```
<FORM NAME=f_name METHOD=get
ACTION="javascript:void(0);">
<INPUT NAME=i_name SIZE=30 MAXLENGTH=30>
</FORM>
```

Shuning uchun JavaScriptdagi dasturlarda ularga nomlari orqali murojaat qilinadi:
window.document.f_name.i_name.value="Текстовое поле";

Yuklangan dokumentning formalari massividan foydalangan holda ham huddi shu natijaga erishish mumkin:

window.document.forms[0].elements[0].value="Текстовое поле";

berilgan misolda biz nafaqat formaga, balki forma maydoniga ham massiv elementi sifatida qaraymiz.

FORM konteineriga mos keluvchi *Form* ob'ekini to'laroq ko'rib chiqamiz.

Hossalar	Metodlar	Hodisalar
action method target elements[] encoding	reset() submit()	onReset onSubmit

Form ob'ektining metodlari, hossalari va hodisalari o'zicha kam ishlatiladi. Ularni qayta aniqlsh odatda forma maydoni qiymatining o'zgarishlariga ta'siri bilan bog'liq bo'ladi.

action

action hossasi skript (CGI-skript) chqiruviga javob beradi. Unda uning (skriptning) URLi ko'rsatilgan bo'ladi. Lekin URLni ko'rsatish mumkin bo'lgan holda uning javascript sxemasini ham ko'rsatish mumkin:

```
<FORM METHOD=post
    ACTION="javascript:window.alert('We use JavaScript-code as an URL');
   void(0);">
<INPUT TYPE=submit VALUE="Продемонстрировать JavaScript в action">
</FORM>
```

Shu narsaga e’tibor beringki, FORM konteinerida METHOD atributi ko’rsatilgan. Bu holatda bu narsa action ga berilgan URLga “?” simvoli yozilmasligi uchun qilingan. Gap shundaki, oshkora ko’rsatilmagan holda kirish metodi deb GET metodi hisoblanadi. Bu metodda formadan turib resursga murojaat qilinganda search nomli URL elementi hosil qilinadi. Bu element URL skriptiga, bisning holatda esa JavaScript –kodga yoilgan “?” belgisi bilan yaqinlashib ketadi.

Quydagi konstruktsiya

window.alert("String");void(0);?

JavaScriptda hatolikni keltirib chiqaradi.

POST metodi HTTP-habarning ichida berilgan formalarni skriptga uzatadi, shunng uchun “?” simvoli URLga qo’shilmaydi va hatolik hosil bo’lmaydi. Bunda void(0)ning qo’llanilishi document qayta yuklanishini bekor qiladi va brauzer submit hodisasini hosil qilmaydi, yani tugmachabi bosilganda formalarni standart qayta ishlashdagi kabi serverga murojaat qilmaydi.

method

method hossasi brauzer-dasturdan turib HTTP-serverdagi resurslarga *kirish metodlarini* belgilaydi. HTML-sahifa muallifi qanday qilib formadan ma’lumotlarni olishi va qayta ishlashiga qarab, u u yoki bu *kirish metodini* tanlashi mumkin. Amaliyotda ko’pincha GET va POST metodlari ishlatiladi.

JavaScript-dastur bu hossaning qiymatini o’zgartirishi mumkin. Oldingi bo’limda (action) formadagi *kirish metodi* oshkora ko’rsatilgan edi. Endi biz uni dastur bajarilayotgan vaqtda qaytadan belgilaymiz:

```
<FORM NAME=m ACTION="javascript:window.alert('Мы используем JavaScript-код в
качестве an URL');void(0);">
<SCRIPT>
document.write("<FONT COLOR=navy>По умолчанию установлен
метод</FONT>"+document.m.method+".");
</SCRIPT>
<INPUT TYPE=button onClick="window.document.main.document.m.method='post';"
VALUE="Метод POST">
<INPUT TYPE=button onClick="window.document.main.document.m.method='get';"
VALUE="Метод GET">
<INPUT TYPE=submit VALUE="JavaScript в ACTION">
</FORM>
```

Oshkora ko’rsatilmagan holda GET metodi o’natiladi.

Keltirilgan misolda ikki narsaga e’tibor berish kerak:

1. 1. Ogohlantirish oynasini ochishdan avval “POST metodi” tugmasini bosish kerak. Agar buni qilinmasa, JavaScript hatoligi to’g’risidagi habar paydo bo’ladi. Bu yerda hammasi yeatrlicha mantiqiy. URLni tuzish submit hodisasini hosil qilish paytida ro’y beradi, skriptni chaqirilishi esa hodisa hosil qilib bo’lingandan keyin ro’y beradi. Shuning uchun hodisani qayta ishlovchida metodni qayta aniqlash mumkin emas, chunki bu vaqtda URL hosil qilib bo’lingan bo’ladi va u o’z navbatida ohirida “?” simvoli bo’lgan JavaScript-dastur bo’ladi. Metodni qayta aniqlash submit hodisasi ro’y berishidan oldin bajarilgan bo’lishi kerak.
2. Dokument ichida SCRIPT konteineri orqali forma uchun oshkora ko’rsatilmagan holdagi kirish metodini habar beruvchi JavaScript-kod joylashtirilgan. Bu konteiner FORM konteineridan keyinoq joyashtirilgan. Uni FORM konteineridan oldin joylashtirish mumkin emas, chunki interpretator tomonidan boshqaruvni qo’lga olingan vaqtda FORM ob’ekti yaratilmagan bo’ladi va shuning uchun uning hossalari bilan ishlash imkoniyati bo’lmaydi.

Method hossaning boshqa uziga hosliklari yuq. Bu hossada GET va POST dan tashqari boshqa kirish metodlarini ham ko'rsatish mumkin, lekin bu serverni qo'shimcha sozlashni talab qiladi.

target

target hossasi CGI-skriptga murojaat qilish natijasini yuklash kerak bo'lgan oyna nomini belgilaydi. Bu hossa qiymatini JavaScript-dastur ichida qo'llash o'zini oqlamaydi, chunki doimo oyna identifikatorini olish mumkin yoki ichki qurilgan frames[0] massivi va opener, top, parent va h. kabi oyna hossalaridan foydalanish mumkin. Tashqi faylni qandaydir oynaga yuklash uchun doimo window.open() metodini qo'llash mumkin. Lekin shunga qaramay bu metodni qo'llash mumkin:

```
for(i=1;i<id.frames.length;i++)
{
if(id.frames[i].name==
id.frames[0].document.f0.s0.options[id.frames[0].document.f0.s0.selectedIndex].text)
{
id.frames[i].document.open();
id.frames[i].document.write("<CENTER>Выбрали этот фрейм</CENTER>");
id.frames[i].document.close();
}
else
{id.frames[i].document.open();
id.frames[i].document.write("<CENTER>Этот фрейм не выбрали</CENTER>");
id.frames[i].document.close();
} }
```

Misolda freym nomlarini tartiblash tsikli tashkil qilingan. Agar nom ko'rsatilgan nom bilan ustma-ust tushsa, freym tanlangan hisoblanadi. Bu yerda quyidagini ta'kidlash kerak: Internet Explorer bilan islaganda freymlarga iloji boricha indeks orqali murojaat qilmaslik kerak.

elements[]

Dokumentga ichki qurilgan *Form* ob'ektini hosil qilishda brauzer u bilan birga forma maydonlari massivini ham yaratadi. Odatda maydonlarga ularning nomlari orqali murojaat qilinadi, lekin forma mydonlari massivining indekslari orqali ham murojaat qilish ham mumkin:

```
<FORM NAME=fe>
<INPUT NAME=fe1 SIZE=30 MAXLENGTH=30>
<INPUT TYPE=button VALUE="Ввести текст по имени"
    onClick="document.fe.fe1.value='Ввести текст по имени';">
<INPUT TYPE=button VALUE="Ввести текст по индексу"
    onClick="document.fe.elements[0].value='Ввести текст по индексу';">
<INPUT TYPE=reset VALUE="Очистить">
</FORM>
```

Bu misoldan ko'rinib turibdiki, massivda maydonlarni indekslashtirish "0" raqamidan boshlanadi. Formadagi maydonlarning umumiy sonini murojaat qilishlar natijasi sifatida olish mumkin:

document.forms[i].elements.length.

encoding

Form obe'ktida shunday hossa mavjud, lekin undan qanday foydalanish esa unchalik tushunarli emas. Encoding hossasini o'zgartirish faqatgina formada file tipidagi maydon mavjud

bo’lgandagina o’zini oqlaydi. Bu holda foydalanuvchiga fayllarni o’zining mahalliy diskidan serverga berishga ruhsat berilgan deb faraz qilinadi. Bunda agar multipart/form-data kodlashtirish ko’rsatilmagan bo’lsa, faqatgina faylning nomi uzatiladi, agar u ko’rsatilgan bo’lsa, u holda faylning o’zi ham uzatiladi.

Bu to’g’risida dastlabki keladigan fikr shu bo’ladiki – ma’lum bir holatlarda fayllarni uzatishni bekor qilish. Skriptni o’zini foydalanuvchi uning kodini o’zgartirmasligi uchun tashqi faylga joylashtirish kerak.

reset()

reser() metodi, uni hodisalarni qayta ishlovchi onReset bilan chalkashtirmaslik kerak, forma maydonlari qiymatlarini oshkora bo’lmagan holda o’rnatishga imkon beradi. Bunda Reset tipidagi tugmachadan foydalanish talab qilinmaydi:

```
<FORM NAME=r>
<INPUT VALUE="Значение по умолчанию" SIZE=30 MAXLENGTH=30>
<INPUT TYPE=button VALUE="Изменим текст в поле ввода"
    onClick="document.r.elements[0].value='Изменили текст';">
</FORM>
<A HREF="javascript:document.r.reset();void(0);">
Установили значение по умолчанию</A>
```

Keltirilgan misolda giprmatnli o’tish bo’yicha formadagi oshkora ko’rsatilmagan qiymatlarga qaytish ro’y beradi.

submit()

submit() metodi formaga kiritilgan ma’lumotlarni serverga uzatishni amalga oshirishga imkon beradi. Bunda submit() metodi orqali Submit tipidagi tugmani bosishdagi protsesning o’zi amalgam oshiriladi. Bu ma’lumotlarni serverga uzatishni kechiktirishga imkon beradi:

```
<FORM NAME=s METHOD=post
ACTION="javascript:window.alert('Данные подтверждены');void(0);">
Введите цифру или букву:<INPUT SIZE=1 MAXLENGTH=1>
</FORM>
<A HREF="javascript:document.s.submit();">Отправить данные</A>
```

Umuman olganda ma’lumotlarni foydalanuvchining ishtirokisiz serverga uzatadigan skriptlarni submit() metodi yordamida yozish mumkin. Lekin brauzer kodning bu harakatlari to’g’risida sahifada ogohlantirish beradi.

onReset

reset hodisasi (forma maydonlaridagi qiymatlarning oshkora ko’rsatilmagandagi qiymatlarini qayta qabul qilishi) Reset tipidagi tugmani bosilganda yoki reset() metodining bajarilishi natijasida ro’y beradi. FORM konteinerida ma’lumotlarni qayta ishlash funktsiyasini qayta aniqlash mumkin. Buning uchun unga onReset atributi kiritilgan:

```
<FORM onReset="javascript:window.alert(
'Event Reset');return false;">
<INPUT VALUE="Значение по умолчанию">
<INPUT TYPE=reset VALUE="Восстановить">
</FORM>
```

Bu misolda shu narsaga e’tibor berish kerakki, reset hodisasini qayta ishlovchisi false mantiqiy qiymatni beradi. Bu reset hodisasini qayta ishlashni to’la egallash uchun qilingan. Agar

hodisani qayta ishlovchi false qiymatini bersa, u holda maydonlarning qiymatini oshkora ko'rsatilmagandagi qiymatlariga o'rnatish ro'y bermaydi; agar qayta ishlovchi true qiymatini bersa, u holda maydon qiymatlari oshkora ko'rsatilmagandagi qiymatlariga qayta o'rnatiladi.

onSubmit

submit hodisasi Submit tipidagi tugmani bosganda, grafik tugmani (image tipida) bosganda yoki submit() metodini chaqirganda ro'y beradi. Submit hodisasini qayta ishlash metodini belgilash uchun FORM konteineriga *onSubmit* atributi qo'shilgan. Bu atributda aniqlangan funktsiya ma'lumotlarni serverga jo'natishdan oldin bajariladi. Bunda funktsiya qiymat sifatida nimani berishiga ko'ra ma'lumotlar jo'natiladi yoki jo'natilmaydi.

```
function test()
{
if(parseInt(document.sub.digit.value).toString()=="NaN")
{
window.alert("Некорректные данные в поле формы.");
return false;
}
else
{
return true;
}
}
...
<FORM NAME=sub onSubmit="return test();" METHOD=post
ACTION="javascript:window.alert('Данные подтверждены');void(0);">
<INPUT NAME=digit SIZE=1 MAXLENGTH=1><INPUT TYPE=submit
VALUE="Отправить">
</FORM>
```

Bu misolda return test() konstruktsiyasiga e'tibor berish lozim. test() funktsiyasining o'zi true yoki false qiymatlarini qabul qiladi. Mos ravishda ma'lumotlar serverga jo'natiladi yoki jo'natilmaydi.

Kiritish maydonidagi matn

Kiritish maydoni (TEXT tipidagi INPUT konteyneri) JavaScriptdagi dasturlashning eng keng qo'llaniladigan ob'ektlaridan hisoblanadi. Buni shu bilan tushintirish mumkinki, ularni korsatilgan maqsadda ishlatishdan tashqari bu maydonlarga o'zgaruvchi va ob'ektlarning hossalarining oraliq qiymatlarini kiritib, dastur hatoliklarini to'g'rilash maqsadida ham qollaniladi.

```
<FORM>Число гипертекстовых ссылок:
<INPUT SIZE=10 MAXLENGHT=10 VALUE="&{document.links.length};">
до момента обработки формы.
<INPUT TYPE=button VALUE="Число всех гипертекстовых ссылок в документе"
onClick="window.document.forms[0].elements[0].value=document.links.length;">
<INPUT TYPE=reset VALUE="Установить по умолчанию">
</FORM>
```

Keltirilgan misoldagi formaning dastlabki maydoni – bu kiritish maydoni. Подстановкydan foydalangan holda biz unga oshkora ko'rsatilmagan qiymatni beramiz, keyinchalik tugmacha yordamida uning qiymatini o'zgartiramiz.

Text ob’ekti (kiritishning matnli maydoni) quyidagi hossa, metod va hodisalar bilan xarakterlanadi:

Hossalar	Metodlar	Hodisalar
defaultValue form name type value	blur() focus() select()	onBlur onChange onFocus

Text ob’ektining hossalari – bu forma maydonlari hossalarining standart to’plamidir. Kiritish maydonlarida faqatgina value hossasining qiymatini o’zgartirish mumkin.

Odatda kiritish maydonlarini dasturlashda ikki tipik masalani hal qilinadi: maydonni foydalanuvchi tomonidan qiymat kiritishidan himoya qilish va kiritish maydoni qiymatining o’zgarishiga bo’gan aks ta’sir.

Kiritish maydonini himoyalash

Kiritish maydonini unga simvollar kiritishdan saqlash uchun blur() metodi hodisalarni qayta ishlovchisi onFocus bilan birgalikda ishlatiladi:

```
<FORM>
<INPUT SIZE=10 VALUE="1-е значение"
    onFocus="document.forms[0].elements[0].blur();">
<INPUT TYPE=button VALUE=Change
    onClick="document.forms[0].elements[0].value=
'2-е значение';">
<INPUT TYPE=reset VALUE=Reset>
</FORM>
```

Bu misolda kiritish maydonining qiymatini o’zgartirish mumkin, faqatgina Change va Reset tugmalarini bosga holda. Kursorni kiritish maydoniga o’rnatishga urinilgan holda u tezda bu yerdan qochadi va shuning uchun maydonning qiymati foydalanuvchi tomonidan o’zartirila olmaydi.

Kiritish maydonining qiymatini o’zgartirish

Kiritish maydoni qiymatining o’zgarishiga aks ta’sir *onChange* atributida ko’rsatilganidek dasturiy yul bilan qayta ishlanadi:

```
<FORM METHOD="post" onSubmit="return false;">
<INPUT SIZE="15" MAXLENGHT="15" VALUE="Тест"
    onChange="window.alert(document.forms[0].elements[0].value);">
<INPUT TYPE="button" VALUE="Изменить"
    onClick="document.forms[0].elements[0].value='Change';">
</FORM>
```

Agar diqqatni kiritish maydoniga berilsa va Enter tugmasi bosilsa, xech nima ro’y bermaydi. Agar kiritish maydonidan yuqorida joylashgan maydonga nimadir kiritib, so’ng Enter tugmasi bosilsa, kiritilgan matnli ogohlantirish oynasi paydo bo’ladi (Netscape Navigator uchun) yoki xech nima ro’y bermaydi (ohirgi versiyadagi Internet Explorer uchun). Agar siz ohirgi versiyadagi

Internet Explorerdan foydalanayotgan bo'lsangiz, u holda ogohlatiruvchi oyna faqatgina diqqatni kiritish maydoni tashqarisiga bergandan keyingina paydo bo'ladi. Buni quyidagicha tushuntirish mumkin: birinchidan, *onChange* qayta ishlovchisi maydonga kiritish yakunlangandan keyingina chaqiriladi. Hodisa maydonga matnni kiritishda klaviatura tugmasini har safar bosilganida ham chaqirilavermaydi. Ikkinchidan, hodisani qayta ishlovchi VALUE atributining qiymati JavaScript-dastur tomonidan o'zgartirilganida chaqirilmaydi. Bunga Change tugmasini bosib ko'rib, ishonch hosil qilish mumkin – ogohlantirish oynasi paydo bo'lmaydi. Lekin agar maydonga nimadir kiritib, keyin Change tugmasi bosilsa, oyna paydo bo'ladi.

Ta'kidlab o'tamizki, u Internet Explorer va Netscape Navigatorlar uchun turlicha ishlaydi, aynan onChange hodisasi turlicha qayta ishlanadi. Internet Explorer uchun maydonning har qanday o'zgarishlarida hodisa darhol qayta ishlanadi, Netscape Navigatorda – aktiv maydon tomonidan diqqatni yuqotgandan so'ng.

Ro'yhatlar va pastga tushuvchi menyu

Bu yerda gap forma kontekstidagi, qatlam kontekstidagi va CSS texnologiyasidagi emas, pastga tushuvchi meyu to'g'risida boradi.

Foydalanuvchi interfeysining eng muhim elementlaridan biri meyu hisoblanadi. HTML-formalarda menyudan foydalanish uchun select (o'z navbatida o'z ichiga OPTION konteynerlarini joylovchi SELECT konteyneri) tipidagi maydonlar ishlatiladi. Bu maydonlar tanlov variantlari ro'yhati ko'rinishida bo'ladi. Bunda ro'yhat "tushishi" yok oyna ichida aylantirilishi mumkin. Select tipidagi maydonlar ro'yhatdan faqatgina bitta variantni tanlash yoki bir nechta variantlarni belgilash imkonini beradi. Select tipidagi maydonlarni boshqarish uchun JavaScriptda Select va Option ob'ektlari mavjud.

Bu ob'ektlar quyidagi hossa, metod va hodisalar bilan xarakterlandi:

Select ob'ekti		
Hossalar	Metodlar	Hodisalar
form length name options[] selectedIndex type	blur() click() focus()	onBlur onChange onFocus

Option ob'ekti		
Hossalar	Metodlar	Hodisalar
defaultSelected index selected text selectedIndex value	нет	нет

Biz bu ikki ob'ektning barcha hossa, metod va hodisalarini bayon qilib o'tirmaymiz. Faqatgina ular kombinatsiyalarini qo'llashning tipik ususllarigagina to'htalib o'tamiz. Bizning jadvallarimizda Option ob'ekti quyida joylashganiga- bu uning Select ga nisbatan buysingan holatda turishini bildiradi- qaramasdan, biz uning hossa va o'ziga hosliklarining bayonidan boshlaymiz.

Option ob'ekti

Option ob'ekti shunisi bilan qiziqki, JavaScriptning boshqa ko'pgina ob'ektlaridan farq qilgan holda u konstruktorga ega. Bu shuni bildiradiki, dasturchi o'zi Option ob'ektini yaratishi mumkin:

```
opt = new Option([ text, [ value,
    [ defaultSelected, [ selected ] ] ] ]);
```

Bu yerda:

text — <LI> (<LI>текст) konteynerida joylashuvchi matn satri;
value — Option ob'ekti bilan bog'liq muqobilini tanlashda serverga beriladigan qiymat;
defaultSelected — oshkora ko'rsatilmagan holda tanlangan muqobil (true/false);
selected — muqobil tanlangan (true/false).

Birinchi qaraganda dasturchiga bunday ob'ekt nima uchun kerakligi tushunarli emas, ahir Select tipidagi ob'ektni yaratish mumkin emas va shundan unga yangi OPTION ob'ektini biriktirish mumkin emas. Hammasi dokumentga ichki qurilgan Select ob'ektlari muqobillari ro'hatini o'garishi tugrisida gapirganda tushunarli bo'ladi. Buni qilish mumkin, chunki Select muqobillari ro'yhatining uo'zgarishi dokumentni qayta formatlashga olib kelmaydi. Muqobillar ro'hatining o'zgarishi HTML-formalarda bo'lmagan ichma-ich qo'yilgan menyular yaratish muammosini *oddiy menyularni dasturlash* (options[]) yuli bilan hal qilishga imkon beradi.

Muqobillarni dasturlashda (Option ob'ekti) shu narsaga e'tibor berish kerakki, Option ning hossalari orasida name hossasi yo'q. Bu shuni bildiradiki, ob'ektga nomi orqali murojaat qilish mumkin emas. Hossaning yo'qligi shu bilan tushuntiriladiki, OPTION konteynerida NAME atributi yo'q. Dokumentga ichki qurilgan Option ob'ektlariga faqatgina Select ob'ektining options[] massive elementlari sifatidagina murojaat qilish mumkin.

options[]

options[] massivi – bu Select ob'ektining hossasidir. Bu massivning elementlari Option ob'ektlariniki kabi hossalarga ega bo'ladi. Hususan ular dokumentga ichki qurilgan Option ob'ektlaridir. Ular brauzer tomonidan sahifani yuklanishi bilan hosil qilinadi. Dasturchi nafaqat Optionning yangi ob'ektlarini yaratish imkoniyatiga ega, balki brauzer tomonidan yaratilgan ob'ektlarni yo'qotishi ham mumkin:

```
<FORM NAME=f0>
<SELECT NAME=s0>
<OPTION>Первый вариант
<OPTION>Второй вариант
<OPTION>Третий вариант
</SELECT>
<INPUT TYPE=button VALUE="Удалить последний вариант"
    onClick="document.f0.s0.options[document.f0.s0.length-1]=null;">
<INPUT TYPE=reset VALUE=Reset>
</FORM>
```

Keltirilgan misolda sahifani serverdan yuklashda uchta muqobil aniqlangan. Agar select maydoni tanlansa, ular paydo bo'ladi. Ohirgi variantni o'chirish tugmasi ("Delete last option")ni

bosgandan keyin faqatgina ikkita muqobil qoladi. Muqobilni o'chirich tugmasini yana bir bor bosilsa, faqatgina bitta muqobil qoladi va hokazo. Va nihoyat variantlar umuman qolmaydi, yani faydalanuvchi tanlash imkoniyatidan ayriladi. Reset tugmasi muqobillar izsiz yuqotilganligini ko'rsatadi, chunki bu tugmani bosgandan keyin SELECT maydoni qayta tiklanmaydi.
Endi Option konstruktoridan foydalangan holda protsesni teskarilaymiz:

```
function def_f1()
{
document.f1.s1.options[0] = new Option("вариант Один","",true,true);
document.f1.s1.options[1] = new Option("вариант Два");
document.f1.s1.options[2] = new Option("вариант Три");
return false;
}
...
<FORM NAME=f1 onReset="def_f1();">
<SELECT NAME=s1>
<OPTION>вариант Один
<OPTION>вариант Два
<OPTION>вариант Три
</SELECT>
<INPUT TYPE=button VALUE="Удалить последний вариант"
    onClick="document.f1.s1.options[document.f1.s1.length-1]=null;">
<INPUT TYPE=reset VALUE=Reset>
</FORM>
```

Berilgan holda biz reset hodisasi (FORM konteyneri)ni qayta ishlaymiz. Bunda Option tipidagi yangi ob'ektlar yaratamiz va ularni Select ob'ektiga bo'ysundiramiz. Bunda massivning birinchi elementi sahifani dastlabki yuklashdagi harakatni modellashtirish uchun oshkora ko'rsatilmagan holda tanlanishi kerak.

HTML-formalarda menyu osti menyularini qilish mumkin emas. JavaScript bu cheklovni chetlab o'tishga imkon beradi va select maydonini dasturlash orqali almashtirishni bajarishga imkon beradi.

length

options[] ni qayta dasturlashga oid misollarda Select ob'ektining length hossasi ko'p ishlatiladi. U tanlash maydoni uchun berilgan muqobillar sonini belgilaydi. Bu hossaning yordamida ro'yhatlarni yo'qotish va qayta tiklash mumkin.

Keyingi misolda shu hossa yordamida variantlar sonini aniqlaymiz:

```
<FORM NAME=f3>
Число вариантов: <INPUT NAME=i0 SIZE=1 MAXLENGTH=1 onFocus="out();">
</FORM>
<SCRIPT>
document.f3.i0.value=document.f1.s1.length;
</SCRIPT>
```

SCRIPT konteyneriga e'tibor bering. U formadan keyin joylashgan. Agar uni oldinroqqa qo'ysak, u holda forma maydonlari aniqlanmay qoladi va natijada biz hatolik to'g'risidagi habarni olamiz.

selectedIndex

Select ob’ektining tanlangan variant qiymatini beruvchi hossasi selectedIndex kabi belgilanadi.

```
<FORM>
Вариант:
<SELECT NAME=s0 onChange="form.elements[1].value=selectedIndex;">
<OPTION>Один
<OPTION>Два
</SELECT>
Выбрали индекс:
<INPUT SIZE=1 maxlength=1>
</FORM>
```

Bu misolda hodisalarni qayta ishlovchilarga e’tibor bering. *onChange* qayta ishlovchini o’zini keyinroq tushuntiramiz. Hozircha asosiysi bu emas. Joriy formaning elementlariga qanday murojaat qilayotganimizni ko’ring. Birinchidan, biz *form* nomidan foydalanayapmiz. U maydon qarashli bo’lgan *Form* ob’ektini ko’rsatadi. Ikkinchdan, biz formaning ikkinchi elementini ko’rsatayapmiz. Bu vaqtda u aniqlangan emas, lekin hodisa biz variantni tanlaganimizdan keyingina ro’y beradi. Bu vaqtga kelganda maydon aniqlangan bo’ladi. Uchinchidan, biz formani to’la ismini ko’rsatmasdan, selectedIndexni ko’rsatamiz. Bu yerda u joriy formaga tegishli bo’ladi.

onChange

change hodisasi Select ob’ektidagi tanlangan indeksning qiymati o’zgartirilgan paydta sodir bo’ladi. Biz bu sahifadagi yagona variantni tanlash maydonidagi bu indeksni o’zgarishi bilan birnecha marta duch keldik (selectedIndex va options[]). Bu hodisa SELECT konteynerining *onChange* atributida ko’rsatilgan JavaScript-dastur tomonidan qayta ishlanadi. Bu jihatdan agar biz SELECT konteynerining multiple bilan ishlasak nima bo’lishini kuzatish qiziqarli:

```
<FORM>
```

Kantselyariya tovarlari to’plami:

```
<SELECT onChange="form.elements[1].value=";
for(i=0;i<form.elements[0].length;i++)
if(form.elements[0].options[i].selected==true)
form.elements[1].value = form.elements[1].value+i;"multiple>
<OPTION>Вариант 1
<OPTION>Вариант 2
<OPTION>Вариант 3
<OPTION>Вариант 4
<OPTION>Вариант 5
<OPTION>Вариант 6
<OPTION>Вариант 7
</SELECT>
Выбраны позиции:
<INPUT NAME=s1 SIZE=7 MAXLENGTH=7
onFocus="form.elements[1].blur();">
</FORM>
```

Shunga e’tibor beringki, change hodisasi tanlsh yoki muqobilni rad qilish ro’y berayotgan hollarda bo’ladi. Variantlar ketma-ket belgilab boriladigan holgina bundan mustasno. Bu holda

hodisa foydalanuvchi sichqoncha tugmasini qo'yib yuborishi bilan ro'y beradi va barcha belgilangan muqobillar tanlangan hisoblanadi.

selected

Option ob'ektining selected hossasi, uning yordamida kantselyariya asboblari to'g'risidagi misol qurilgan edi, ikki qiymatni qabul qilishi mumkin: to'g'ri (true) yoki yolg'on (false). Misolda biz tanlangan muqobilning indeksini bosib chiqaramiz, agar Option ob'ektining selected hossasining qiymati true bo'lsa:

```
if(form.elements[0].options[i].selected==true)
 ...
```

Umuman olganda, selected hossasi aynan ko'p tanlovli maydonlar bo'lgan holatlar uchun qiziqarli. Yagona variantni tanlash kerak bo'lgan hollarda uni Select ob'ektining selectedIndex hossasiga ko'rsatgan holda olish mumkin.

text

text hossasi muqobilga mos keluvchi, meyuda aks ettiriluvchi matnni bildiradi:

```
<SELECT onChange= "form.elements[2].value=
form.elements[0].options [form.elements[0
].selectedIndex].text;">
</SELECT>
```

Keltirilgan misolda text hossasi formaning matnli maydoniga kiritiladi.

value

ma'lumotlarni brauzerdan serverga uzatishda so'rovda ihtiyoriy matn uzatiladi, agar OPTION konteynerining VALUE atributida qiymat ko'rsatilmagan bo'lsa.

Tugmachalar

Tugmachalarda Web da JavaScripni qo'llamagan holda foydalanish umuman mantiqqa ega emas. Tugmachali formani hosil qiling va tgmachaga bossangiz nima bo'lishini kuzating – tugmacha bosiladi, lekin hech nima ro'y bermaydi. Formaning hech bir standart hodisasi (reset yoki submit) chaqirilmaydi. Albatta, bu fikr Submit va Reset tugmachalariga tegishli emas. Tugmacha formaga asosan click hodisasini qayta ishlash uchun kiritiladi:

```
<FORM>
<INPUT TYPE=button VALUE="Окно предупреждения"
     onClick="window.alert('Открыли окно');">
</FORM>
```

Tugmachada aks ettirilayotgan metn INPUT konteynerining VALUE atributi orqali aniqlanadi. Bu atribut bilan Button ob'ektining value hossasi bog'langan. Qizig'i shuki, spetsifikatsiyaga ko'ra bu atributning qiymatini o'zgartirish mumkin emas. Lekin 4 versiyadagi Netscape Navigator va Internet Explorerlarda bu mumkin.

Shuni ta'kidlab o'tish kerakki, Netscape Navigatorda tugmachaning o'chami o'zgarmas (birinchi qiymati eng uzun bo'lishi kerak, aks holda unchalik chiroyli chiqmaydi), Internet Explorerda esa o'lcham matnning uzunligiga bog'liq ravishda o'zgaradi.

Rasmlar

Rasm –tugmachalar – bu oddiy tgmachalar, faqat ular ma'lumotlarni serverga uzatish imkoniyatiga ega. Hususan, JavaScriptdagi bunday tugmachalar INPUT konteynerining ikki turini tashkil qiladi: image va submit. JavaScriptda bunday tugmalar bilan bog'langa ob'ektlar Submit deb ataladi..

```
<FORM>
Активная кнопка:
<INPUT TYPE=image SRC=images.gif onClick="return false;">
</FORM>
```

Ta’kidlab o’tganimizdek, berilgan ob’ekt Button ob’ekti bilan bir hil hossa, metod va hodisalarga ega. Lekin har hil brauzerlarda hodisalarni qayta ishlashda shu hodisalarga aks harakatlar har hil bo’lishi mumkin. Masalan, onClick hodisasida Internet Explorer false qiymatni qaytarib, ma’lumotlarni serverga uzatishni bekor qilishi mumkin. Netscape Navigator esa bu hodisaga umuman javob bermasligi mumkin va u serverga ma’lumotlarni uzatishni faqatgina FORM konteynerining *onSubmit* atributi orqali bekor qilishi mumkin:

```
<FORM onSubmit="return false">
Активная кнопка:
<INPUT TYPE=image SRC=images.gif border=0>
</FORM>
```

Grafik tugmachalarning qiziqarli hossalaridan biri bu ularning kerakli vaqtda serverga foydalanuvchi sichqoncha tugmasini bosish orqali ko’rsatgan nuqta koordinatalarini bera olish qobiliyatidir. Afsuski, tugmaning bunday harakatkarini JavaScript-dastur orqali qayta ishlash mumkin emas.

Ma’lumotlar almashish

Ma’lumotlarni formadan serverga uzatish submit hodisasi asosida amalga oshiriladi. Bu hodisa foydalanuvchining quyida berilgan harakatlaridan birida sodir bo’ladi:

- Submit tugmachasi bosilganda;
- Grafik tugmacha bosilganda;
- Formada biror bir maydondan turib Enter tugmachasi bosilganda;
- submit() metodi chaqirilganda.

FORM konteynerini JavaScript ob’ektlariga tasvirini bayon qilishda submit hodisasini qayta ishlash to’g’risida yetarlicha ma’lumot berildi. Bu bo’limda biz JavaScript-dasturlarni maydon atributlari va haodisalarni qayta ishlovchilardagi qollanishiga to’htalib o’tamiz. Submit hodisasini ushlash/haosil qilish imkoniyatiga alohida e’tibor berish kerak.

Submit tugmachasi

Submit tugmachasi kiritish maydonining bir turi hisoblanadi. U o’zini oddiy tugmacha kabi tutadi, lekin u yana submit hodisasi (ma’lumotlarni serverga uzatish)ni ham hosil qiladi. Bu bilan u JavaScriptda dasturlash nuqtai-nazaridan qaraganda grafik tugmachalar bilan bir hildir:

```
<FORM>
<INPUT TYPE=submit VALUE=submit>
</FORM>
```

Keltirilgan misolda biz sahifani oddiygina qayta yuklaymiz.

Dasturlash nuqtai nazaridan submit hodisasini tutib turish imkoniyati va bunday qilishda standartdan tashqari harakatlarni bajarish ko’proq qiziqish tug’diradi.bu maqsadda tugmachada click hodisasini qayta ishlash uchun atribut mavjud (*onClick*):

```
<FORM>
<INPUT TYPE=submit VALUE=Submit onClick="return false;">
</FORM>
```

Bu misoldan ko’rinib turibdiki, Submit tugmachasi o’zini Netscape Navigatordagi grafik tugmachaga nisbatan boshqacharoq, lekin Internet Explorer dagi grafik tugmacha bilan bir hil (vaqt o’tishi bilan farq yuqolsa kerak) tutadi. Tugmachani bosish bilan sahifani qayta yuklash yuz bermaydi – ma’lumotlarni serverga uzatish bekor qilingan. Qayta ishlovchi FORM konteyneridagi submit hodisasini qayta ishlovchi kabi harakat qiladi.

Endi uzimizning submit hodisasini qayta ishlovchi dasturimizni yozishimiz mumkin:

```
function my_submit()
{
if(window.confirm("Хотите перегрузить страницу?")) return true;
else return false;
}
...
<FORM>
<INPUT TYPE=submit VALUE=Submit onClick="return my_submit();">
</FORM>
```

Agar sahifani qayta yuklash zarurligini tasdiqlansa, u haqiqatda qayta yuklanadi, rad qilingan holda esa (cancel), siz qayta yuklashsiz joriy sahifaga qaytasiz. Harakatlar yanada murakkab bo’lishi ham mumkin. Har qanday holatda ham agar qayta ishlash funktsiyasi true qiymatni qaytarsa, ma’lumotlarni serverga uzatish (bizning misolda – sahifani qayta yuklash) yuz beradi, aks holda (false qiymatida) – ma’lumotlar uzatilmaydi.

Formadagi yagona maydon

Agar formada bitta –yagona maydon bo’lsa va biz unga kiritib keyin Enterni bossak, u holda brauzer submit hodisasini hosil qiladi:

```
<FORM onSubmit="window.alert('Сделано');return false;">
<INPUT SIZE=10 MAXLENGTH=10>
</FORM>
```

Bunday hodisani huddi misolda qilinganidek FORM konteyneridagi submit hodisasini qayta ishlash dasturi yordamidagina tutish va qayta ishlash mumkin.

Bu misolda formada kiritish maydonidan tashqari meyu ham ishtirok etadi. Agar tanlangan muqobillar qiymatlarini o’zgartirilsa, qayta yuklash sodir bo’lmaydi, lekin kiritish maydonidagi qiymatni o’zgartirish va Enter tugmasi bosilsa, submit hodisasi ro’y beradi va sistema ogohlantirish oynasini chiqaradi.

submit() metodi

submit metodi - bu forma metodidir. Agar dasturda submit metodi chaqirilsa, shu metod qo’llanilayotgan formadagi ma’lumotlar serverga uzatiladi. Kiritish maydoni va tanlash menyuli misolni rivojlantiramiz (muqobilni tanlashdan oldin misol ostidagi sharxni o’qing):

```
<FORM onSubmit="window.alert('Сделано');return false;">
<INPUT SIZE=10 MAXLENGTH=10>
<SELECT onChange="form.submit();">
<OPTION>Вариант 1<OPTION>Вариант 2</SELECT>
</FORM>
```

Muqobillarni tanlashda foydalanuvchi birdaniga server bilan ma’lumotlar almashishni boshlaydi. Bu holda submit hodisasi Enter tugmasini bosishdan farq qilgan holda, hodisalarni qayta ishlovchi tomonidan ushlanmaydi. Brauzerning bunday harakati anchagina mantiqiy. Agar dasturchi submit() metodini chaqirgan bo’sa, u holda u serverga jo’natilayotgan ma’lumotlarni tekshirgan bo’lishi kerak.

Image ob’ekti

JavaScriptda dasturlashda koproq effektlarga *grafika* bilan ishlash orqali erishiladi. Bunda dasturchida unchalik ko’p instrumentlar yo’q: dokumentga ichki qurilgan rasmlar, Image ob’ektini hosil qilish imkoniyati, rasmlarni gipermatnli o’tishlar va jadvallar bilar birga qo’llash. Shunga qaramasdan bu soda usullar orqali erishiladigan effektlar juda ajoyib.

JavaScriptda grafikani dasturlash quyidagi hossa, metod va hodisalar bilan xarakterlanuvchi Image ob’ektiga asoslanadi:

Hossalar	Metodlar	Hodisalar
border complete height hspace name src vspace width lowsrc	Yo’q	onAbort onError onLoad

Hossalarning ko’pligiga qaramasdan ularning ko’pchiligini faqatgina o’qish mumkin, lekin o’zgartirib bo’lmaydi. Bunga metodlarning yo’qligi ham guvoh bo’la oladi. Lekin baribir ikki hossani o’zgartirish mumkin: src va lowsrc. Shularning o’zi ham rasmlar bilan ko’pgina effektlarni bajarish uchun yetarli.

Image sinfining barcha ob’ektlarini ichki qurilgan va dasturchi tomonidan yaratilganlarga bo’lish mumkin. Ichki qurilganlari – bu *IMG* konteyneridagi rasmlar. Agar bu rasmlar nomlansa, u holda ularga nomlari bo’yicha murojaat qilish mumkin:

```
<A HREF="javascript:void(0);"
onClick="window.alert('Image name:'+
      document.images[0].name)">
<IMG NAME=tuit SRC=images.gif BORDER=0>
</A>
```

Rasm aktiv. Agar unga bosilsa, u holda *IMG* konteynerining nomini olamiz. document.images[0].name ga murojaat qilish bun omni ogohlantirsih oynasida chop qilish imkonini beradi. Bunda nomning o’zi *IMG* konteynerida name=tuit kabi ko’rsatiladi.

Ichki qurilgan grafik ob’ektlarga indekslari bo’yicha ham murojaat qilish mumkin:
document.images[0];

bu holda images[0] – dokumentning birinchi rasmi.

src va lowsrc

src va lowsrc hossalari dokumentning ichiga joylashtiriladigan rasmlarning URLlarini aniqlaydi. Bunda lowsrc asosiy rasm yuklanayotgan paytda tasvirlanuvchi, odatda kichkina bo’ladigan, vaqtinchalik rasmni aniqlaydi, uning URLi *IMG* konteynerining SRC atributi qiymatida

ko’rsatiladi. src hossasi IMG konteynerining SRC atributi qiymatini qabul qiladi. Dasturchi src ni ham, lowsrc ni ham qiymatlarini o’zgartira oladi. src bilan misolni ko’ramiz:

```
document.i2.src="images2.gif";
```

bu misoldan ko’rinib turibdiki, ichki qurilgan Image ob’ektining src hossasi qiymatini o’zgartirish hisobiga joylashtirilgan rasmni moslashtirish imkoniyati mavjud. Agar siz berilgan sahifani birinchi marta ko’rayotgan bo’lsangiz (yani rasm brauzer tomonidan keshga joylashtirilmagan), u holda rasmning asta-sekin o’zgarishi sezilarli bo’ladi. Bu o’zgarishni qanday tezlashtirishni biz keyingi bo’limda ko’rib chiqamiz.

Rasmni o’zgartirish

Rasmni o’zgartirish mumkin, faqatgina ichki qurilgan Image ob’ektining src hossasiga yangi qiymat bergan holda. “Grafikani dasturlash” sahifasida oddiy holatda buni qanday amalgam oshirish ko’rsatilgan edi. Aniqki, rasmni serverdan sekin yuklash tez varaqlab ko’rishga imkon bermaydi. Bu muammoni hal qilishga urinib ko’ramiz.

Hususan, masalani yechimi rasmni va uning *tasvirlarini* vaqt bo’yicha ko’paytirib borishdir. Buning uchun Image ob’ektining konstruktoridan foydalaniladi:

```
<TABLE>
<TD>
<A HREF="javascript:void(0)";
onMouseover="document.m0.src=color[0].src;
        return true;"
onMouseout="document.m0.src=mono[0].src;
       return true;">
<IMG NAME=m0 SRC="images0.gif" border=0>
</A>
</TD>
...
</TABLE>
```

Kod bo’lagi sichqoncha kursorining o’tishida rasmning almashishi va qayta tikkalanishining tipik usulini ko’rsatadi. Tabiiyki, faqat bitta emas, balki birdaniga bir nechta rasmlarni almashtirish mumkin.

Shunga qaramay asosiysi rasmlarni almashtirishda emas, balki bu qanday tezlikda amalga oshirilishidir. Kerakli natijaga erishish uchun sahifa boshida tasvirlanishidan oldin grafika yuklanadigan rasmlar massivi hosil qilinadi (sahifani yuklashda status satriga e’tibor bering):

```
color = new Array(32);
mono = new Array(32);
for(i=0;i<32;i++)
{ mono[i] = new Image();
color[i] = new Image();
if(i.toString().length==2)
{
mono[i].src = "images0"+i+".gif";
color[i].src = "images0"+i+".gif";
}
else
{
```

```
mono[i].src = "images00"+i+".gif";
color[i].src = "images00"+i+".gif";
}
}
```

Yana bir xarakterli usul – JavaScript-kodni kechiktirib bajarish funktsiyasini qo'llanilishi (eval()):

```
function def()
{
for(i=0;i<32;i++)
{
eval("document.m"+i+".src=mono["+i+"].src");
}
for(i=0;i<5;i++)
{
eval("document.r"+i+".src=rm["+i+"].src");
}
}
```

Berilgan holda eval() bizni o'zlashtirish operatsiyasini terishdan ozod qiladi.

Multiplikatsiya

IMG konteyneridagi SRC atributi qiymatini almashtirish g'oyasining tabiiy davomi bu multiplikatsiya hisoblanadi, yani bu atribut qiymatini vaqt bo'yicha ketma-ket o'zgartirib boorish. Multiplikatsiyani amalga oshirish uchun Window ob'ektining setTimeout() metodidan foydalaniladi. Hususan, multiplikatsiyani ishga tushirishning ikki hil usuli bor: ии:

- onLoad();
- onClick(), onChange() ...

Ko'proq qo'llaniladigani – onLoad dagi setTimeout().

onLoad() hodisasi

onLoad() hodisasi brauzer tomonidan dokumentni yuklash tugallangan vaqtda sodir bo'ladi.

Hodisani qayta ishlovchi BODY konteynerida ko'rsatiladi:

```
...
<BODY onLoad="JavaScript_code">
...
```

Bisning holatda dokumentni yuklash vaqtida rasmlarni o'zgartirish tsikli boshlanishi kerak:

```
function movie()
{
eval("document.images[0].src='clock"+
    i+".gif';");
i++;if(i>6) i=0;
setTimeout("movie();",500);
}
...
<BODY onLoad="movie();">
...
```

Chekli sondagi almashtirishlarni ham bajarish mumkin bo'lsada, misolda cheksiz tsikldan foydalaniladi:

```
function movie()
{
eval("document.images[0].src='clock"+
    i+".gif';");
i++;
if(i<7)
{
setTimeout("movie();",500);}
}
...
<BODY onLoad="movie();">
```

Ikkala misolda ham setTimeout() metodning ishlatilishiga e'tibor berish kerak. Birinchi qaraganda u oddiy rekursiyadek ko'rinadi. Lekin amalda hammasi sal murakkabroq. JavaScript ko'p oqimli operatsion sistemalargacha ishlab chiqilgan, shuning uchun skriptlarning ishlashini quyidagicha tasvirlash to'g'ri bo'ladi:

- Skript onLoad() hodisasida boshqaruvni o'z qo'liga oladi.
- Rasmni o'zgartiradi.
- Yangi skriptni vujudga keltiradi va uning bajarilishini 500 millisekundga kechiktiradi.
- Joriy skript JavaScript-interpretator tomonidan yo'qotiladi.

Kechikish muddati tugagandan keyin bajarilish yana qaytariladi. Birinchi misolda (cheksiz takrorlanish) funktsiya o'zini-o'zi hosil qiladi va shu yo'l bilan o'zining uzuliksiz bajarilishini ta'minlaydi. Ikkinchi misolda (chekli sondagi iteratsiyalar) o'n marta takrorlashlardan keyin funktsiya yangidan hosil bo'lmaydi. Bu yangi rasmlarni namoyish qilishni tugatishga olib keladi.

Multiplikatsiyani boshlash va tugatish

Doimiy multiplikatsiyaga boshqa usullar bilan ham erishish mumkin, masalan, ko'p kadrli *grafik* fayllar orqali. Lekin sahifadagi harakat doimo ham yahshi emas. Ko'pincha foydalanuvchining talabi bo'yicha harakatni boshlash va to'xtatishni amalga oshirish istagi paydo bo'ladi. Oldingi misoldan foydalanib, bu hohish(multiplikatsiyani boshlash yoki to'htatish) ni bajaramiz:

```
var flag1=0;
function movie()
{
if(flag1==0)
{
eval("document.images[0].src='clock"+
    i+".gif';");
i++;if(i>6) i=0;
}
setTimeout("movie();",500);
}
...
<BODY onLoad="movie();">
...
```

```
<FORM>
<INPUT TYPE=button VALUE="Start/Stop"
onClick="if(flag1==0) flag1=1; else flag1=0;">
</FORM>
```

Bu holatda biz faqatgina rasmning o'zgarishini chetlab o'tayapmiz, lekin yangi oqimning paydo bo'lishini to'xtatmayapmiz. Agar setTimeout() ni if() konstruktsiyasini ichiga joylashtirsak, u holda Start/Stop tugmasini bosgandan keyin oqim paydo bo'lmaydi va harakatni boshlash mumkin bo'lmay qoladi.

Multiplikatsiyani boshlash va to'xtatish muammosini yechishning yana bir usuli mavjud. U setTimeout() metodini qo'llashga asoslangan. Tashqaridan qaraganda hammasi oldingidagidek ko'rinadi, lekin jarayon butunlay boshqacha ketadi:

```
var flag1=0;
var id1;
function movie()
{
eval("document.images[0].src='clock"+
     i+".gif';");
i++;if(i>6) i=0;
id1 = setTimeout("movie();",500);
}
...
<BODY onLoad="movie();">
...
<FORM>
<INPUT TYPE=button VALUE="Start/Stop"
onClick="if(flag1==0)
  { id1=setTimeout('movie();',500); flag1=1;}
  else {clearTimeout(id1); flag1=0;};">
</FORM>
```

Ikki o'zgarishga e'tibor bering. Birinchidan, oqim identifikatori (id1) e'lon qilingan va foydalaniladi; ikkinchidan, clearTimeout() metodi ishlatiladi va, hususan, unga argument sifatida oqim identifikatori beriladi. movie() funktsiyasini qaytadan paydo bo'lishini to'xtatish uchun oqimni "o'ldirish" yetarli.

Tasvirlarni optimallashtirish

Grafikani dasturlashda sahifani namoyish qilishga va grafik tasvirlarning o'zgarish tezligiga ta'sir qiluvchi juda ko'p omillarni hisobga olish kerak. Bu yerda dasturni optimallashning oddiy dilemmasi – tezlik yoki egallagan hajm – faqatgina tezlikni oshirish hisobiga yechiladi. Hotira o'lchami to'g'risida o'lash JavaScriptda dasturlashda negadir qabul qilinmagan.

Rasmlarni tasvirlashni optimallashtirishning barcha usullaridan biz faqat bir nechtasiga to'xtalib o'tamiz:

- Yuklash vaqtida tasvirlashni optimallash;
- Oldindan yuklash hisobiga tasvirlashni optimallash;
- Rasmni kesish hisobiga optimallash.

Agar dastlabki ikkisi statistik rasmlarni tasvirlashga ham, multiplikatsiyaga ham tegishli bo'lsa, uchunchisi asosan multiplikatsiyaga xarakterli usul hisoblanadi.

Yuklashdagi optimallash

Deyarli HTML-sahifalarni yaratishga oid barcha qo'llanmalarda shu narsa ta'kidlanadiki, HTML-sahifaning ichida IMG konteyneridan foydalanganda WIDTH va HEIGHT atributlarini ko'rsatish kerak. Bu narsa sahifa komponentalarini serverdan yuklash tartibi va HTML-parserining ishlash algoritmi tufayli kelib chiqadi. Eng avval ***разметки*** matni yuklanadi. Undan keyin parser matnni tahlil qiladi va qo'shimcha komponentalar, jumladan *grafikani* yuklashni boshlaydi. Bunda rasmlarni yuklash HTTP-protokolining tipiga ko'ra ketma-ket yoki parallel ravishda borishi mumkin.

Parser yuklash parallel ravishda ishlashda davom etadi. Agar rasmlar uchun kenglik va balandlik parametrlari berilgan bo'lsa, u holda matnni formatlash va brauzer oynasida aks ettirish mumkin. Bu parametrlar aniqlanmaguncha matn namoyish qilinmaydi.

Shunday qilib rasmni kengligi va bo'yini ko'rsatish dokumentni rasmlar serverdan olinishidan avvalroq namoyish qilishga imkon beradi. Bu foydalanuvchida to'la yuklanib bo'lguncha dokumentni o'qishga yoki undagi gipermatnli o'tishlar bo'yicha harakat qilishga imkon beradi (load hodisasi).

JavaScript nuqtai-nazaridan qaraganda rasmlarning o'lchamlarini ko'rsatish dokument ichidagi *grafikani* tasvirlash oynasiga boshlang'ich parametrlarni beradi. Bu esa to'la rasm bilan almashtirish uchun kichkina shaffof oynadan foydalanish imkoniyatini yaratadi. G'oya shundaki, talab qilinganda katta ob'ekt bilan almashtirish uchun kichkina ob'ekt uzatiladi.

Oldindan yuklab olish

Bitta timsolni ikkinchisi bilan almashtirish faqatgina bu yetarlicha tez sodir bo'lgandagina o'zini oqlaydi. Agar qayta yuklash uzoq vaqt davom etsa, uning foydasi bo'lmaydi. Tez almashtirish uchun dokumentni avval maxsus yaratilgan Image sinfi ob'ektiga yuklab olish imkoniyatidan foydalaniladi.

Haqiqiy ta'sirini faqatgina klient tomoni (brauzer) da sahifani keshlashtirishni o'chirib qo'yilgandagina sezish mumkin. Keshlashtirishdan Web-tarmoqlarida sahifalar bilan ishlashni tezlashtirish uchun tez-tez foydalaniladi. Qoidaga binoan, birinchi sahifani yuklash – bu anchagina uzoq vaqt davom etadigan jarayon. Eng asosiysi foydalanuvchi bu vaqtda ozgina kuta olishi kerak. Shuning uchun faqatgina birinchi sahifada kerak bo'ladigan *grafikadan* tashqari u yana bu sahifada ko'rsatilmaydigan *grafikani* ham uzatishi kerak. Lekin tarmoqning boshqa sahifalariga o'tilganida u serverdan uzatilmasdan, kechikmay tasvirlanadi.

Yuqorida bayon qilingan usul bir qiymatli emas. Uni shu narsa bilan oqlash mumkinki, agar foydalanuvchi kutib tura olmasa, u holda u *grafikani* uzatishni butunlay uzib qo'ishi mumkin.

Rasmlarni kesib olish

Rasmlarni qirqib olish anchagina keng qo'llaniladi. U tasvirlanayotgan rasmni qisman o'zgartirish hisobiga natijaga erishishga imkon beradi. U ko'pincha menyularni yaratishda qo'llaniladi.

Bunday natijadan tashqari qirqib olish katta rasmlardan multiplikatsiyani amalga oshirishga imkon beradi. Bunda to'la obraz emas, faqatgina ayrim qismlari o'zgartiriladi.

Grafika va jadvallar

Web-tarmoqlarini yaratishning keng tarqalgan usullaridan biri bu rasmlarni qismlarga qirqib olish texnikasidir. Sahifaning navigatsion komponentalarini tashkil qilish uchun bu texnikani qo'llashning quyidagi usullarini ajratib ko'rsatish mumkin:

- gorizontal va vertikal menyular;
- ichma-ich qo'yilgan menyular;
- navigatsion *grafik* bloklar.

Qirqish grafikasini ishlatishdagi eng asosiy muammo bu uni HTML-parser tomonidan sahifani kontekstli formatlashdan himoya qilishdir. Masala shundaki, agar ular bir satrga joylashmasa, u разметки elementlarini avtomatik ravishda yangi satrga ko'chiradi. Qirqilgan rasmning tashkil

qiluvchi bo’laklari aniq bir tartibda joylashgan bo’lishi kerak, shuning uchun ularni qatorga o’tkazish istalgan natijani bermaydi:

```
<IMG SRC="image1.gif"><IMG
SRC="image2.gif"><IMG
SRC="image3.gif"><IMG
SRC="image4.gif">
```

Elementlar yangi satrga ko’chiriladi, chunki bo’limning kengligi rasmlarning umumiy kengligidan kichikroqdir. Muammo parserdan himoya -<PRE> qo’llanilsa, yechiladi:

```
<PRE>
<IMG SRC="image1.gif"><IMG
SRC="image2.gif"><IMG
SRC="image3.gif"><IMG
SRC="image4.gif">
</PRE>
```

Bunday menyudan foydalanish unda gipermatnli o’tishlarni aniqlshni talab etadi va bu quyidagi natijaga olib keladi:

```
<PRE>
<A HREF="javascript:void(0);"><IMG
SRC="image1.gif"></A><A
HREF="javascript:void(0);"><IMG
SRC="image2.gif"></A><A
HREF="javascript:void(0);"><IMG
SRC="image3.gif"></A><A
HREF="javascript:void(0);"><IMG
SRC="image4.gif"></A>
</PRE>
```

Bunga 0ga teng BORDER atributini qo’llash hisobiga erishish mumkin:

```
<PRE>
<A HREF="javascript:void(0);"><IMG
SRC="image1.gif" BORDER="0"></A><A
HREF="javascript:void(0);"><IMG
SRC="image2.gif" BORDER="0"></A><A
HREF="javascript:void(0);"><IMG
SRC="image3.gif" BORDER="0"></A><A
HREF="javascript:void(0);"><IMG
SRC="image4.gif" BORDER="0"></A>
</PRE>
```

Endi shu usul bilan ko’p satrli menyuni hosil qilishga harakat qilamiz:

```
<PRE>
<IMG SRC=tree.gif WIDTH=27 HEIGHT=21 BORDER=0><A
```

```
HREF="javascript:void(0);"><IMG SRC="image1.gif" WIDTH=103 HEIGHT=21
BORDER=0></A>
<IMG SRC=tree.gif WIDTH=27 HEIGHT=21 BORDER=0><A
HREF="javascript:void(0);"><IMG SRC="image2.gif" WIDTH=103 HEIGHT=21
BORDER=0></A>
<IMG SRC=tree.gif WIDTH=27 HEIGHT=21 BORDER=0><A
HREF="javascript:void(0);"><IMG SRC="image3.gif" WIDTH=103 HEIGHT=21
BORDER=0></A>
<IMG SRC=tree.gif WIDTH=27 HEIGHT=21 BORDER=0><A
HREF="javascript:void(0);"><IMG SRC="image4.gif" WIDTH=103 HEIGHT=21
BORDER=0></A>
</PRE>
```

Tola qoplagan rasm hosil bo'lmaydi, chunki satrning balandligi rasmning balandligiga teng emas. Bu parametrlarni qo'ish deyarli mumkin emas. Har bir foydalanuvchi brauzerni o'z hohishicha moslab oladi. Echim quyidagi jadvalni ishlatilishidadir:

```
<TABLE BORDER=0 CELLPADDING=0 CELLSPACING=0 ALIGN=center>
<TR>
<TD><IMG SRC=tree.gif WIDTH=27 HEIGHT=21 BORDER=0></TD>
<TD><A HREF="javascript:void(0);"><IMG SRC="image1.gif" WIDTH=103 HEIGHT=21
BORDER=0></A></TD>
</TR>
<TR>
<TD><IMG SRC=tree.gif WIDTH=27 HEIGHT=21 BORDER=0></TD>
<TD><A HREF="javascript:void(0);"><IMG SRC="image2.gif" WIDTH=103 HEIGHT=21
BORDER=0></A></TD>
</TR>
<TR>
<TD><IMG SRC=tree.gif WIDTH=27 HEIGHT=21 BORDER=0></TD>
<TD><A HREF="javascript:void(0);"><IMG SRC="image3.gif" WIDTH=103 HEIGHT=21
BORDER=0></A></TD>
</TR>
<TR>
<TD><IMG SRC=tree.gif WIDTH=27 HEIGHT=21 BORDER=0></TD>
<TD><A HREF="javascript:void(0);"><IMG SRC="image14.gif" WIDTH=103 HEIGHT=21
BORDER=0></A></TD>
</TR>
</TABLE>
```

Bu holda barcha rasmlarni o'tkazib yubormasdan birlashtirish imkoniyati bo'layapti va shu yo'l bilan navigatsion daraxtning uzuliksizligiga erishilayapti. Oraliqlarni BORDER,CELLSPACING va CELLPADDING atributlarini qo'llash hisobiga yo'qotiladi. Birinchisi yacheykalar orasidagi chegarani yo'qotadi, ikkinchisi yacheykalar orasidagi masofani 0 pikselga o'rnatadi, uchinchisi esa yacheyka chegarasi va unga joylashtirilgan element orasidagi chekinishni 0 pikselga o'rnatadi.

Grafika va hodisani qayta ishlash

Bu bo'limda *IMG* konteyneri hodisalar qayta ishlovchilarini qaramaymiz. Biz hodisalarni qayta ishlovchilar va *grafik* obrazlarni o'zgartirishlarni birgalikda ishlatishning tipik usuliga

to’xtalib o’tamiz. Hususan, qirqilgan grafikani qo’llash ma’noga ega bo’lmasdi, agar tasvirning alohida qismlari o’zarishlarini qayta ishlovchilarni qo’llashning imkoniyati bo’lmaganda. Navigatsion daraxt misolini muhokama qilishda davom etib, uni sichqonchani ob’ektga tugrilash va rasmni o’zgarishlarini qayta ishlovchi bilan rivojlantirilishini ko’rsatamiz:

```
<TABLE BORDER=0 CELLPADDING=0 CELLSPACING=0 ALIGN=center>
<TR>
<TD><IMG SRC=image.gif WIDTH=20 HEIGHT=20 BORDER=0></TD>
<TD><IMG SRC=image1.gif WIDTH=20 HEIGHT=20 BORDER=0></TD>
</TR>
<TR>
<TD><IMG SRC=image2.gif WIDTH=20 HEIGHT=20 BORDER=0></TD>
<TD><A HREF="javascript:void(0);" onMouseover="document.manual.src='image3.gif';return true;"
onMouseout="document.manual.src='image4.gif'; return true;">
<IMG SRC=image5.gif BORDER=0 WIDTH=20 HEIGHT=20></A></TD>
</TR>
<TR>
<TD><IMG SRC=image6.gif WIDTH=20 HEIGHT=20 BORDER=0></TD>
<TD><A HREF="javascript:void(0);" onMouseover="document.desk.src='image7.gif';return true;"
onMouseout="document.desk.src='image8.gif';return true;">
<IMG SRC=image9.gif BORDER=0 WIDTH=20 HEIGHT=20></A></TD>
</TR>
</TABLE>
```

Berilgan misolda sichqoncha kursori rasmlar ustidan o’rayontganda ularning menyulari o’zgaradi. Bu narsa ikki hodisani qo’llnishi hisobiga ko’ra amalga oshiriladi: onMouseover va onMouseout. Birinchi hodisada rasm pozitivdan negativga o’zgaradi, ikkinchi hodisada u o’zining dastlabki variantiga qaytadi. Shuni ta’kidlash kerakki, hodisalar yakor (A) konteynerida, *IMG* konteynerida emas, aniqlangan. Bu brauzerlarning mosligi nuqtai-nazaridan eng maqbul variant.

Vertikal va gorizontal menyular

“Grafika va jadvallar” va “Grafika va hodisalarni qayta ishlash” bo’limlarida bayon qilinganlarning deyarli barchasi bitta darajali menyularni qurish masalalariga tegishli. Shuning uchun bu bo’limda biz bunday menyularga oid ko’proq yoki ozroq darajada amaliy bo’lgan misollarni keltirishga harakat qilamiz. Grafik menyular shunisi bilan qulayki, mualiff uning komponentalarini doimo yeatrlicha aniqlikda ekranga joylashtirishi mumkin. Bu esa, o’z navbatida, sahifaning boshqa elementlarini ham menyu elementlariga nisbatan aniqroq joylashtirishga imkon beradi:

```
<TABLE BORDER="0" CELLPADDING="0" CELLSPACING="0" ALIGN="center">
<TR ALIGN="center">
<TD><IMG NAME="e0" SRC="empty.gif" WIDTH="15" HEIGHT="8" BORDER="0"></TD>
<TD><IMG NAME="e1" SRC="empty.gif" WIDTH="15" HEIGHT="8" BORDER="0"></TD>
<TD><IMG NAME="e2" SRC="empty.gif" WIDTH="15" HEIGHT="8" BORDER="0"></TD>
<TD><IMG NAME="e3" SRC="empty.gif" WIDTH="15" HEIGHT="8" BORDER="0"></TD>
</TR>
<TR>
<TD><A HREF="javascript:void(0);" onMouseover="document.e0.src='arrowdw.gif';return true;"
```

```
 onMouseout="document.e0.src='empty.gif';return true;">
 <IMG SRC="image1.gif" BORDER="0"></A></TD>
<TD><A HREF="javascript:void(0);" onMouseover="document.e1.src='arrowdw.gif';return true;"
onMouseout="document.e1.src='empty.gif';return true;">
<IMG SRC="image2.gif" BORDER="0"></A></TD>
<TD><A HREF="javascript:void(0);" onMouseover="document.e2.src='arrowdw.gif';return true;"
onMouseout="document.e2.src='empty.gif';return true;">
<IMG SRC="image3.gif" BORDER="0"></A></TD>
<TD><A HREF="javascript:void(0);" onMouseover="document.e3.src='arrowdw.gif';return true;"
 onMouseout="document.e3.src='empty.gif';return true;">
 <IMG SRC="image4.gif" BORDER="0"></A></TD>
</TR>
</TABLE>
```

Bu holatda ko'rsatkich sichqoncha ko'rsatib turgan elementlar ustidan aniq yuguradi. Keng ma'noda olganda IMG dagi ALT atributini qo'llash va status satrida undan nusxa olish yangi *grafik* elementni qo'shishga qaraganda ko'proq informativ hisoblanadi. Tug'ri, ALTdagi bor narsalar sal kechikibroq tasvirlanadi.

Endi hozir, shunday qilish qabul qilinganidek, matnning grafik bloklari asosida qurilgan vertikal menyuni amalga oshirilishini ko'rib chiqamiz:

```
<TABLE BORDER="0" CELLPADDING="0" CELLSPACING="0" ALIGN="center">
<TR>
<TD><A HREF="javascript:void(0);" onMouseover="document.evente1.src='corner.gif';"
onMouseout="document.evente1.src='clear.gif';">
<IMG SRC="image1.gif" border="0"></A></TD>
<TD><A HREF="javascript:void(0);" onMouseover="document.evente1.src='corner.gif';"
onMouseout="document.evente1.src='clear.gif';">
<IMG NAME="evente1" SRC="clear.gif" border="0"></A></TD>
</TR>
<TR>
<TD><A HREF="javascript:void(0);" onMouseover="document.evente2.src='corner.gif';"
onMouseout="document.evente2.src='clear.gif';">
<IMG SRC="image2.gif" border="0"></A></TD>
<TD><A HREF="javascript:void(0);" onMouseover="document.evente2.src='corner.gif';"
onMouseout="document.evente2.src='clear.gif';">
<IMG NAME="evente2" SRC="clear.gif" border="0"></A></TD>
</TR>
<TR>
<TD><A HREF="javascript:void(0);" onMouseover="document.evente3.src='corner.gif';"
onMouseout="document.evente3.src='clear.gif';">
<IMG SRC="image3.gif" border="0"></A></TD>
<TD><A HREF="javascript:void(0);" onMouseover="document.evente3.src='corner.gif';"
onMouseout="document.evente3.src='clear.gif';">
<IMG NAME="evente3" SRC="clear.gif" border="0"></A></TD>
</TR>
<TR>
<TD><A HREF="javascript:void(0);" onMouseover="document.evente4.src='corner.gif';"
 onMouseout="document.evente4.src='clear.gif';">
```

```
 <IMG SRC="image4.gif" border="0"></A></TD>
<TD>
<A HREF="javascript:void(0);" onMouseover="document.evente4.src='corner.gif';"
onMouseout="document.evente4.src='clear.gif';">
<IMG NAME="evente4" SRC="clear.gif" border="0">
</A></TD>
</TR>
</TABLE>
```

Sichqoncha harakatlanganda uning ostiga tushgan mos komponentaning “burchagi egiladi”. Berilgan holda “burchak” – bu mustaqil rasm. Barcha burchaklar jadvalning o’ng ustunida amalga oshirilgan. Gipermatnli o’tish har ikkala rasm (matn va “burchak”) bo’yicha ham ishlashi uchun grafik obrazlarni qamrab oluvchi A ning bir xil konteinerlari qo’llaniladi. Bu yechimda bitta kamchilik bor: matndan “burchakka” o’tishda keyingisi “miltillaydi”. Rasmlarni jadvalning bitta yacheykasiga joylashtirish ham mumkin, lekin bu holda uning kengligini berish kerak bo’ladi, aks holda brauzer oynasining o’lchamlari o’zgartirilganda, rasmlar “siljib ” ketishi mumkin. “Miltillashni” yo’qotish uchun to’la qonli rasm almashtirishlar qilish kerak.

“Miltillash” konteiner разметкasining bir elementidan ikkinchisiga o’tganida ro’y beradi. Bunda element tasvirlanishining hossalari qaytadan aniqlanadi.

Ichma ich qo’yilgan menyular

Formalarni dasturlashni muhokama qilinganda ta’kidlangan ediki, HTML da ichma ich qo’ilgan menyularni yaratishning standart usuli yo’q. Shunga qaramasdan grafikaning hisobiga uning o’xshashini yaratish mumkin. Bunda shuni esda tutish lozimki, grafika yotgan joyni metn bilan to’ldirish mumkin emas:

```
<SCRIPT>
function submenu(a)
{
if(a==1)
{
document.menu00.src="image1.gif"; // 1 (активна)
document.menu10.src="image2.gif"; // 2
document.menu01.src="image3.gif"; // 1 пункт вложенного меню 1
document.menu02.src="image4.gif"; // 2 пункт вложенного меню 1
}
if(a==2)
{
document.menu00.src="image1.gif"; // 2
document.menu10.src="image2.gif"; // 1 (активна)
document.menu01.src="image3.gif"; // 1 пункт вложенного меню 2
document.menu02.src="image4.gif"; // 2 пункт вложенного меню 2
}
}
</SCRIPT>

<TABLE BORDER="0" CELLPADDING="0" CELLSPACING="0" ALIGN="center">
<TR>
<TD><A HREF="javascript:void(0);" onMouseover="submenu(1);return true;">
<IMG NAME=menu00 SRC=image1.gif BORDER=0></a></td>
```

```
<TD><IMG NAME=menu01 SRC=image3.gif BORDER=0></TD>
</TR>
<TR>
<TD><A HREF="javascript:void(0);" onMouseover="submenu(2);return true;">
<IMG NAME=menu10 SRC=image2.gif BORDER=0></td>
<TD><IMG NAME=menu02 SRC=image4.gif BORDER=0></TD>
</TR>
</TABLE>
```

Bu misolda ichma ich qo'yilgan menyu asosiydan o'ng tomonda joylashgan. Ichma ich qo'ilganlikka rangning o'zgarishi hisobiga erishiladi. Menyuning bo'ysunishini uning vaziyatini asosiy menyuga nisbatan o'zgarishida ko'rsatish mumkin.

Bu holatda menyuni pastga harakatga keltirish uchun korinuvchi yoki ko'rinmas rasmlar vositasida joyni band qilish kerak. Bular hech qanday yuklanishni olmaydigan illyustrativ rasmlar bo'lishi shart emas.

Qatlamlardan foydalangan holda haqiqiy ichma ich qo'yilgan menyularni yaratish mumkin.

Frames kollektsiyasi.

Freymlar — bu qanchadir ko'rinishi o'zgargan oynalar. Ular oddiy oynalardan shunisi bilan farq qiladiki, ular oddiy oynalarning ichida joylashadi. Freymda, oddiy oynalardan farq qilib, instrumentlar paneli ham, menyu ham bo'lishi mumkin emas. Status maydoni sifatida freym o'zi joylashgan oynaning status maydoni sifatida ishlatiladi. Boshqa bir nechta sezilarli farqlar ham mavjud.

Biz quyidagilarga to'xtalamiz:

- *freymlar* ierarxiyasi;
- *freymlarni* nomlanishi;
- *freymga* ma'lumotlarni uzatish.

Tabiiyki, ierarxiya freymlarning nomlanishini ham, rfreymga diqqatni berishni ham aniqlaydi.

Freymlar ierarxiyasi

Avval oddiy misolni ko'ramiz. Ekranni ikkita vertikal ustunlarga bo'lamiz:

```
<HTML>
<HEAD>
</HEAD>
<FRAMESET COLS="50%,*">
<FRAME NAME=left SRC=left.html>
<FRAME NAME=right SRC=right.html>
</FRAMESET>
</HTML>
```

Freymlar joylashtiriladigan *oynani* _top(_parent) deb ataymiz.

Misolni murakkablashtiramiz: birinchi freymni gorizontal bo'yicha ikkiga bo'amiz.

```
<HTML>
<HEAD>
</HEAD>
<FRAMESET COLS="50%,*">
<FRAME NAME=left SRC=left.html>
<FRAMESET ROWS="50%,*">
<FRAME NAME=top SRC=top.html>
```

<FRAME NAME=bottom SRC=bottom.html>
</FRAMESET>
</FRAMESET>
</HTML>

Rasm. O'ng fraym gorizontal bo'yicha ikkiga bolingan.

Ikkita holatga e'tibor bering: birinchidan, _top va top ni farqlash kerak, ikkinchidan, right freymi yo'qoldi. Birinchi ta'kid bo'yicha: _top – bu katta oyna uchun rezervlangan nom, top esa - katta oynaga sahifa muallifi tomonidan berilgan freimning nomi. Ikkinchi ta'kid bo'yicha: barcha freymlar uchun brauzerning barcha oynalari katta oyna hisoblanadi, berilgan holatda right nomli freym mavjud emas.

Uning paydo bo'lishi uchun har ikkala misolimizni birlashtirishimiz kerak. Bu shuni bildiradiki, right freymiga biz yana freymli dokumentni yuklashimiz kerak.

Birinchi dokument:

<HTML>
<HEAD>
</HEAD>
<FRAMESET COLS="50%,*">
<FRAME NAME=left SRC=left.html>
<FRAME NAME=right SRC=right.html>
</FRAMESET>
</HTML>
Второй документ (right.htm):
<HTML>
<HEAD>
</HEAD>
<FRAMESET ROWS="50%,*">
<FRAME NAME=top SRC=top.html>
<FRAME NAME=bottom SRC=bottom.html>
</FRAMESET>
</HTML>

Bu holatda sahifalarni bo'ysunishi uchta freymli misoldagidan boshqacha ko'rinishda bo'ladi.

Shunday qilib biz uchta freym va bitta katta oynali hol bilan bir xil natija oldik. Lekin bu variant anchagina egiluvchan: u gorizontal bo'linmaga ega freymga ta'sir o'tkazishga imkon beradi. Huddi shunday usul "Web-injiniring"ning bosh sahifasida qo'llaniladi.

Sahifaning shunday fraymli interpretatsiyasi JavaScriptdagi freymlarni nomlashda ham uz aksini topadi.

Freymlarning nomlanishi

Freymga uning nomi orqali yoki frames[] massivining elementi kabi murojaat qilish mumkin. HTML dokumentni ko'rib chiqamiz:

```
<HTML>
<HEAD>...
</HEAD>
<FRAMESET COLS="20%,*">
<FRAME NAME=left SRC=frame1.htm>
<FRAME NAME=right SRC=frame2.htm>
</FRAMESET>
</HTML>
```

Faraz qilaylik, o'ng tomondagi freymga yuklangan sahifada ikkita rasm bo'lsin. Ulardan ikkinchisining src hossasini o'zgartirish uchun quyidagi yozuvdan foydalanish mumkin:

```
top.frames[1].images[1].src="pic.gif";
```

yoki

```
top.right.images[1].src="pic.gif";
```

freymlarning indekslanishi bilan bog'liq Window ob'ektining bir o'lchamli ichki qurilgan freymlar massivida ular qanday nomerlanadi degan savol tug'iladi. Buni misol yordamida namoyish qilamiz:

```
<FRAMESET ROWS="50,*,50">
<FRAME NAME=top SRC=top.html>
<FRAMESET COLS="100,*,100">
<FRAME NAME=left SRC=left.html>
<FRAME NAME=center SRC=center.html>
<FRAME NAME=right SRC=right.html>
</FRAMESET>
<FRAME NAME=bottom SRC=bottom.html>
</FRAMESET>
```

Rasm. Uchta vertikalga bo'lingan markaziy freym

Endi oldingi misoldaginga perpendikulyarni - uchta freymdan tashkil topgan ustunni quramiz:

```
<FRAMESET COLS="100,*,100">
<FRAME NAME=left SRC=top.html>
<FRAMESET ROWS="60,*,60">
<FRAME NAME=top SRC=left.html>
<FRAME NAME=center SRC=center.html>
<FRAME NAME=bottom SRC=right.html>
</FRAMESET>
<FRAME NAME=right SRC=bottom.html>
</FRAMESET>
```

Rasm. Ikkita gorizontalga bo'lingan markaziy freym

Shunday qilib, sahifada freymlarni bir o'lchamli freymlar massivida nomerlashda sistema "chapdan o'ngga, yuqoridan pastga" qoidasiga amal qiladi. Misollarimizni birgalikda qo'ygan holda freymlarning har qanday murakkab sistemasida ham sahifalarni to'gri indekslashtirishni olish mumkin.

Diqqatni freymga berish

Tipik Web-tarmog'ini yaratishdagi odatdagi masala bu CGI-skript bajarilishi natijalarini bu skrip uchun ma'lumotlar kiritilayotgan freymdan boshqa freymga yuklash hisoblanadi. Agar natijalarni yuklash yo'li aniq belgilab qo'yilgan bo'lsa, u holda oddiygina qilib formaning TARGET atributidan foydalanish mumkin. Agar ish natijalari, masalan, tanlangan tugmachaga bog'liq holda turlicha freymlarga yuklanishi kerak bo'lsa, bu ancha murakkabdir.

Bu masalani turlicha usulda hal qilish mumkin: oldinroq ochilgan oynani ochish yoki target hossasini qayta belgilash. Oxirgi usul yaxshiroq ko'rinadi va biz o'shandan boshlaymiz:

```
function load()
{
 if(self.document.f.s.options[document.f.s.selectedIndex].text=="top")
 {
  document.f.target = "mytop";
  self.top.frames[2].document.open();
  self.top.frames[2].document.close();
 }
 else
 {
  document.f.target = "mybottom";
  self.top.frames[1].document.open();
  self.top.frames[1].document.close();
 }
 return true;
}
```

load() funktsiyasi submit hodisasini qayta ishlovchi sifatida chaqiriladi, u mantiqiy funktsiya hisoblanadi. True qiymatini qaytarilishi dokumentni qayta yuklanishini amalga oshirishga imkon beradi.

Endi ikkinchi variantni ko'rib chiqamiz. Uning asosiy g'oyasi shundaki, mavjud *oyna* bilan bir nomdagi *oynani* ochishga urinilganda, *oyna* ochilmaydi, balki mavjud *oynadan* foydalaniladi. *Freym* - bu *oyna*, shuning uchun unga ham shu qoida qo'llaniladi, lekin buni amalga oshiruvchi funktsiya oldingisidan farq qiladi:

```
function load()
{
 if(self.document.f.s.options[document.f.s.selectedIndex].text=="top")
 {
  window.open("./framer.htm","mytop");
  self.top.frames[2].document.open();
  self.top.frames[2].document.close();
 }
 else
 {
  window.open("./framer.htm","mybottom");
```

```
   self.top.frames[1].document.open();
   self.top.frames[1].document.close();
  }
 return false;
}
```

Shunga e’tibor beringki, berilgan funktsiya false qiymatini qaytaradi. Bu narsa submit hodisasini berkitish uchun sodir bo’ladi. Axir dokument yuklab bo’lingan va uni qayta yuklashga hojat you’q.

3. Web-sahifalarni yaratish bo’yicha tavsiyalar
Sahifalarning tiplari
Grafik dizayn
Psixologik tavsiyalar

Sahifalarning tiplari.

Web-tarmoqlarni yaratishning umumiy tamoyillari.

Aytaylik, siz WWW (*World Wide Web*, Dunyo tarmog’i)da o’z bo’g’iningizni yatamoqchi va joylashtirmoqchi bo’dingiz. Uning qiziqarli, foydali va muximi ko’p kiriladigan bo’lishi uchun nimalar qilish kerak. Aniq javob berish kerak bo’lgan birinchi savol: Web-bo’g’in nima uchun yaratilayapti? Bun savolga ko’p narsa bog’liq: bezash usuli, yaratish va keyingi faoliyati uchun sarflanadigan harajatlar, Web ga o’rnatish uchun ma’lumotlarni tasvirlash formati, Web-serverning va Internet aloqasi kanalining dasturiy ta’minotiga bo’lgan talablar va instrumentlar. Bu yerda bir necha variantlar bo’lishi mumkin.

Agar siz qandaydir maxsulotni realizatsiya qilayotgan kompaniya uchun Web-bo’g’in yaratayotgan bo’lsangiz, u holda asosiy maqsad firma to’g’risida ma’lumot tarqatish va maxsulot reklamasi va yana Web-magazin tashkil qilish bo’ladi. Bunda quyidagi vazifalar yechiladi:

- kompaniya imijini o’zgartirish va obro’yini ko’tarish;
- maxsulot markasini oldinga surish;
- maxsulot tug’risidagi ma’lumotlarning yetarliligi va xaridorlar uchun narxlar;
- dilerlar tarmog’ini qo’llab-quvvatlash, maxsulot tug’risidagi ma’lumotlarning yetarliligi va dilerlar uchun narxlar;
- maxsulotni Internetda to’g’ridan-to’g’ri sotish, Web-magazin tashkil qilish;
- ofisdan tashqaridagi xodimlar uchun ichki ma’lumotlarning yetarliligi.

Boshqacha variant – bu internetda tijorat bilan shug’ullanmaydigan, balki ma’lumot tarqatadigan ilmiy yoki umumta’lim tashkilotlari uchun Web-bo’g’in yaratish. Bu holatda masala

qidiruvni tashkil qilish va unga kirish uchun katta ma’lumotlar massivlarini yig’ish, qayta ishlash va Web-bo’g’inga joylashtirish to’g’risida boradi.

Va so’nggi variant – siz o’zingizning shaxsiy sahifangizni Internetga joylashtirmoqchisiz.

Qo’yilgan savollarga to’g’ri javob berish uchun Web-bo’g’in mo’ljallangan foydalanuvchilar kategoriyasini aniqlash lozim. Klientlardi jalb qiluvchi va ularni ushlab turuvchi ma’lumotlar qurilmasi psixologiya asosida qurilishi kerak. Keyinchalik Web-bo’g’in bilan bog’liq bo’lgan har qanday harakatlarlarning maqbulligi sahifaga kiruvchilarni ularga nisbatan aks harakatlariga va bu harakatlar ko’zlangan maqsadga yetishga qanchalik xizmat qilishiga ko’ra baxolanmog’i lozim.

Maqsadlar belgilab va foydalanuvchilar kategoriyalari aniqlab olingandan keyin tayorlangan ma’lumotlarni Web-dokumentlar bo’yicha taqsimlab olish, ular orasidagi bog’liqliklarni o’ylab ko’rish va qo’shimcha navigatsion imkoniyatlarni, masalan, Web-bo’g’im bo’yicha qidiruv sistemasini ko’rib chiqish kerak.

Firma Web-bo’g’imining tipik tuzilishi odatda quyidagicha tasvirlanadi:

Kompaniya to’g’risida ma’lumot. Firmaning maqsadlari va ish yuritishi usullari, uning tarixi va h. To’g’risida bayon qilish kerak. Klientlar aynan siz bilan, boshqa kompaniyalar bilan emas, hamkorlik qilib qanday foyda olishlarini ko’rsating.

Maxsulot va xizmatlar yuzasidan ma’lumot. Web-sahifaga o’zingizning maxsulotingizning rasmlarini yoki fotorasmlarini joylashtiring. Uning xususiyatlari va ustunliklarini bayon qiling va undan foydalanishga misollar keltiring. Agar maxsulotning qog’ozdagi katalogi mavjud bo’lsa, u holda uning mazmunini va strukturasini Web-bo’g’inga joylashtirish mumkin. Bu katalogning elektron variantini yaratish va yangilab turishini yengillashtiradi. Agar maxsulotga yoki xizmatlarga Internet orqali buyurtmalar qabul qilish rejalashtirilayotgan bo’lsa, u holda elektron pochta orqali qabul qilish uchun buyurtma blankasini joylashtirish kerak.

Informatsion qo’llab-quvvatlash. Bu bo’limda qo’shimcha texnik ma’lumotlar, ko’p beriladigan savollar, noto’kisliklarni bartaraf qilish uchun maslaxatlar va hokazolar e’lon qilinadi.

Yangiliklar. Xaridorlarni firma tomonidan chiqarilayotgan yangi maxsulotlar va ko’rastilayotgan yangi xizmatlar to’g’risida xabardor qilib turing, press-rilizlarni chop qiling va hokazo.

Teskari aloqa. Siz bilan qanday aloqaga kirishish mumkinligini va qaerda joylashganingiz to’g’risida xabar bering. Izoxlar formasi, mexmonlar kitobi, xaridor o’z fikrlarini jo’natishi uchun elektron pochta adreslari va h. ni joylashtiring.

Web-bo’g’inni to’ldirishda doimo ikki tamoyilni yodda tutish kerak: o’ziga xoslik va e’lon qilinayotgan materiallarning ishonchliligi.

O’ziga xoslik mazmunga qo’iladigan eng birinchi talabdir. *WWW* da shunga o’xshash ko’plab sahifalar bo’lishi mumkin. Sizning Web-bo’g’iningiz shunga o’xshash tematikadagi serverlardan farq qilishi kerak, xech bo’lmaganda diqqatni o’ziga jalb qilish uchun. O’ziga xos materialllarni joylashtirilishi sahifangizga kiruvchilar sonini orttiradi. O’ziga xos ma’lumotlar resursi yaratish uchun printsipial jihatdan yangi narsani yaratish shart emas, mavjud resurslarni boshqacha tartibda joylashtirish mumkin, lekin bunda xaridorga kerakli materialn qidirishda ko’p vaqt sarflashga majbur qilmaslik kerak. Resurslarning o’ziga xosligini zamonaviy qidiruv sistemalari yordamida tekshirib ko’rish mumkin. Obro’ jihatdan kelsak, u siz ma’lumotlarni qanchalik sinchiklab tanlashingiz, tekshiringiz va o’z vaqtida yangilab turishingizga bog’liq bo’ladi.

Web- bo’g’imni yaratishda shu narsani esda tutish kerakki, uni tashkil qiluvchi alohida dokumentlar bir xil bezash usuli va navigatsion vositalar bilan birlashtirilishi kerak. Yagona bezash usuli – bu professional Web-bo’g’inni havaskor Web-bo’g’inlardan farqlovchi ko’rsatkichlardan biridir. Bir xil ko’rinishda qilinga dokumentlar tufayli foydalanuvchilar sizning Web-bo’g’imizni boshqalardan ajrata olishadi va eslab qolishadi. Bu dokumentlar xuddi ikki tomchi suvdek bir-biriga o’xshashi kerakligini bildirmaydi, lekin umumiy g’oya, yagona usul doimo bo’lishi kerak.

Shu narsa sahifalar bo'ylab navigatsion vositalarga ham tegishli. Web-bo'g'inga kiruvchi uni xuddi siz kabi yaxshi biladi deb o'ylamaslik kerak. U hozir o'zining qaerda turganligini va boshqa hohlagan joyga qanday o'tishi mumkinligini qiyinchiliksiz bila olishi kerak. Birinchi sahifaga, qidiruv dasturiga yoki Web-bo'g'in sxemasiga o'ta olish imkoniyati ko'rib qo'yilishi kerak.

Bundan tashqari usulning bir xilligi bekash va navigatsiya (ma'lumotlar to'ldirmasdan)ning faqatgina oddiy elementlariga ega shablon- sahifalardan foydalanish imkoniyatini yaratadi. Ularning yordamida yangi sahifalarni tez va unumli tarzda yaratish va uni yaratishda ishni bir necha kishiga taqsimlash mumkin. Tayyor sahifani olishda shablondan foydalanishda unga faqatgina kerakli ma'lumotlarni kiritiladi. Ketma-ketlilik, mantiqiylik, doimiylik – yaxshi Web-bo'g'in uchun kerak bo'ladigan xususiyatlar. HTML 4.0 da paydo bo'lgan usullarning kaskadli jadvallari sizning Web-bo'g'iningiz stilini hosil qilish va o'zgartirish bo'yicha qilinadiga ishlarni anchagina soddalashtiradi. Ularning bazi bir imkoniyatlari to'g'risida quyida hikoya qilinadi, alohida bo'lim esa ulaga bag'ishlanadi.

Maqsadlar aniqlanib, struktura berilgan va matnli va grafik ma'lumotlar yig'ilgandan keyin Web-bo'imning tashqi ko'rinishini ishlab chiqish kerak. U yana erishish kerak bo'lgan maqsadlarga ham bog'liq bo'ladi. Mumkin bo'lgan yechimlar cpektri juda keng: mavjud sahifalarni ko'rish va unga o'xshashlarini yaratishdan tortib profissional dizayner va rassomlarga murojaat qilishgacha. Shu bilan bir vaqtda Web-bo'g'in tashkil topgan Web-dokumentlarni qurishning o'rnashib ulgurgan qoidalarini ham yodda tutish kerak.

Struktura. Bugungi kunda dokument strukturasi to'grisidagi qarashlar qat'iylashdi. Web-dokument o'zida quyidagi bo'limlarga ega bo'lishi kerak: sarlavha, kompaniya nomi, navigatsion panel, hususiy mazmun, kontakt ma'lumotlari, yangilanish sanasi va vaqti, mualliflik xuquqi va dokument statusi.

Logotip. Web-sahifani yaratishda firmaning nomi doimo ekranda bo'lishi to'g'risida qayg'urish kerak. Buning uchun har bir Web-dokumentning boshida odatda chiroyli qilib bezatilgan firma logotipi joylashtiriladi. Bundan tashqari kompaniya nomi barcha dokumentlarga kiruvchi ma'lumotlarda ham bo'lishi kerak.

Navigatsion panel. Web-dokumentdagi eng muhim bo'limlardan biri bu navigatsion panel yoki boshqaruv panelidir. *WWW* undagi gipermatnli o'tishlar e'lon qilinayotgan metriallar orasida bog'lanishlarni yuzaga keltirganligi hisobiga butub dunyoni egalladi. Lekin xuddi shu o'tishlar to'la xaos havfini ham yuzaga keltiradi, zanjir bo'ylab uch-to'rtta dokumentni o'gandan keyin, orqaga qaytolmay qolasiz. Sizning web-bo'g'iningiz aniq va intuitive tushunarli navigatsion yo'nalishlar bilan ta'minlashi kerak.

Ko'plab o'tkazilgan izlanishlar ko'rsatadiki, Web-serverlarga kiruvchilar juda sabrsizdir va serverdagi ma'lumotlar bo'yicha ikki darajadadan ortiq ichkari kirishni istashmaydi. Shuning uchun katta hajmdagi Web-bo'g'imlarini yaratayotganda har qanday ma'lumot bir-ikki o'tishlardan uzoq jiylashmagan oraliq, odatda birinchi-ikkinchi darajada joylashuvchi dokumentlarni ko'rib chiqish kerak.

Sizning Web-bo'g'iningizning navigatsion paneli har bir dokumentda bo'lishi kerak. Eng avval unda Web-bo'g'im strukturasidagi qo'shni dokumentlarni ko'rsatuvchi "Oldinga"-"Orqaga" ("Navbatdagi"-"Oldingi") tipidagi yo'naltiruvchi o'tishlar bo'lishi kerak. Yana boshqaruv panelidan Web-bo'g'imning barcha yirik bo'limlari – birinchi darajali bo'limlariga o'tishlar bo'lishi kerak. Va nihoyat foydalanuvchi doimo Web-bo'g'imning asosiy sahifasiga qaytish imkoniyatiga ega bo'lishi kerak. O'tishlardan tashqari mahalliy qidiruv sistemasiga va indeksga bo'lgan yo'lni ko'rsatish kerak

Mazmun. Eng oldin shuni ta'kidlash kerakki, Web-dokumentlar oddiy gazeta yoki jurnallarga qo'yiladigan talablarga to'la javob berislari kerak: Grammatik va orfografik xatolardan holi bo'lish, berilayotgan materiallarning aniqligi va ishonchliligi va boshqalar. Bundan tashqari Web-dokument qanoatlantirishi kerak bo'lgan ko'plab maxsus talablar paydo bo'ladi.

Ko’pincha dokumentning o’chamlari to’g’risida savol tug’iladi: qanday sondagi sahifalar yetarli bo’ladi? Javob birinchi qaraganda g’alati tuyulishi mumkin: bitta ekranli oyna yoki xech qanday chegara qo’ilmaydi. Ko’plab izlanishlar shuni ko’rsatadiki, foydalanuvchilar brauzerning yugurdakli polosasi bilan ishlashni yoqtitishmaydi. Ularga hammasidan ko’proq bitta ekranga sig’adigan sahifa yoqadi. Haqiqatdan ham siz xech qachon foydalanuvchiga bitta sahifaga joylashtirib yozilgandan ko’proq ma’lumot bera olmaysiz. Agar bitta sahifaga sig’dira olmasangiz, u holda yan bitta dokument yarating.

Bir ekranli sahifa ma’lumotlarni tasvirlash uchun mos o’lchov bo’lib chiqdi. Agar dokumentning o’lchami bitta sahifadan o’tib ketsa, ko’pchilik hollarda u har biri bitta sahifadan ortiq joy egallamaydigan bir nechta mantiqiy bo’laklarga bo’linishi mumkin. Agar ma’lumotni mantiqiy qismlarga bo’lishning iloji bo’lmasa, bayon qilish usulini, ba’zan esa materialni o’zini ham qayta ko’rib chiqish kerak bo’ladi. Hozirgi vaqtda Web-serverni bir ekranli dokumentlar asosida qurish kerak degan umumiy fikr hosil bo’lgan. Bu qoidadan ikki holdagina chekiniladi. U WWWda e’lon qilinayotgan maqolalarga qo’llanilmaydi va ikkinchi hol- bo’lsih mumkin bo’lmagan anketa formalari.

Grafika. Web-sahifani ishlab chiqishda grafik va matnli materiallarning optimal nisbatini diqqat bilan tanlash kerak. Bitta yahshi rasm ming qator metnning o’rnini olishi mumkin, lekin u tarmoqda ming marta yzoqroq yuklanadi ham. Shuning uchun grafikadan extiyot bo’lib foydalanish kerak. Shu narsadan kelib chiqish mumkinki, sahifadagi grafika Web-ustasi xohlagandan ko’ra ozroq bo’lishi kerak. Foydalanuvchilarda sabr yetishmasligi mumkin va ular dokumentni to’la ochilmasidan turiboq yopib yuborishadi. Sismtema javobining kechikishi foydalanuvchining g’ashini keltiradi. Hozir Internetning kanallar infrastrukturasi ishlash qanchalik og’irligini hamma yaxshi tushunadi. Shuning uchun kechikish vaqti sutka vaqtiga qarab 15-60 sekundgacha o’zgarib turadi. Endi tasavvur qiling, klientda faqat 19200 bit/s uchun modem bo’lsin. Bundan oshig’iga Rossiya telefon liniyalarida erishish qiyin. U holda bir minut ichida, yani klientning sabri tugaguncha, faqatgina 170 Kbayt ma’lumot uzatish mumkin. Shundan kelib chiqib, dokument o’lchami bundan katta bo’lmasligi lozim.

Shuni ta’kidlash kerakki, boshqaruv paneli, firma nomi va logotipi odatda grafik elementlar shaklida bajariladi. Maketini yasagandan keyin *HTML* va *WWW* ning samonaviy texnologiyalari tomonidan tavsiya qilinadigan boshqa vositalar yordamida amalga oshirishni boshlash mumkin.

Web-bo’g’imni yaratib bo’lib, uni Internetga joylashtirish kerak. Bu yerda ikkita variant bo’lishi mumkin: birinchisi – Web-server va Web-bo’g’im bilan birga ofisingizda turgan kompyuterdan foydalanish va ajratilgan yoki kommutirli liniya bo’yicha Internetga ulanish; ikkinchisi – Web-bo’g’imni joylashda maxsus tashkilotlar xizmatidan foydalanish.

Ikkinchi variantni ko’rib chiqamiz. Web-sahifaga kirishni ta’minlovchi providerni to’g’ri tanlash sizning klientlaringizga maksimal qulaylik bilan ma’lumotlarni olish imkoniyatini beradi. Bundan tashqari Web-server tomonidan maxsus imkoniyatlar berilishi Web-bo’g’imni ishlab chiqishni engillashtiriladi.

Sizning Web-bo’g’imingizni o’z serveriga joylashtiradigan provayderni tanlashda nimaga e’tibor berish zarur?

Kanalning o’tkazish qobiliyati. Sizning xaridorlaringiz sahifani yuklanishini ko’p kutib qolmasliklari uchun provayder sekundiga 1-2 Mbit tartibidagi ishonchli yuqori tezlikdagi aloqaga ega bo’lishi kerak.

Server tomonidan SSI (Server Side Includes, server tomonidan qo’ilma) *provayderini qo’llab-quvvatlash.* SSI dan foydalanish Web serverga foydalanuvchiga jo’natilayotgan HTML-dokumentga kichik hajmli dinamik ma’lumotlarni bevosita qo’shish imkoniyatini beradi. So’ralgan HTML-sahifa SSI elementlarini qidirib, “ko’rib chiqiladi”. Bunday elementni topsa, server talab qilingan dinamik ma’lumotni qo’shadi. SSI yordamida bir faylni ikkinchisining ichiga joylashtirish,

CGI-stsenariyalarni bajarishi va boshqa ma’lumotlarni uzatishi mumkin. SSI ning qanday funktsiyalari provayder serveri tomonidan qo’llanishini aniqlash lozim.

Provayder serveri tomonidan CGI-stsenariyalarini qo’llab- quvvatlanishi. CGI (Common Gateway Interface, umumiy shlyuzli interfeys) — Web-serverga ixtiyoriy amaliy dasturlarni bajarishga imkon beruvchi spetsifikatsiya. Bunday dasturlarning (stsenariya yoki “skriptlar”) ishlashi natijasida HTML-dokumentlar yaratiladi. CGI-stsenariyalar yordamida foydalanuvchidan ma’lumotlar qabul qilinishi, Web-sahifalarda muloqotni tashkil qilish, ma’lumotlar bazalariga so’rovlar qilish kabilarni amalga oshirish mumkin. CGI-stsenariyalarini istalgan keng tarqalgan dasturlash tillari: Perl, Basic, C, C++, Pascal va boshqalar yordamida yaratish mumkin.

O’z vaqtida qayta kodlashtirishning qo’llab-quvvatlanishi. Afsuski, turli xil platformalar(Windows, Mac, Unix va boshqalar)da Internetda ishlaganda rus tili uchun turli xil kodlashtirishlar qabul qilingan. Foydalanuvchiga sahifalarni ko’rish oson bo’lishi uchun provayder Web-serveri tushayapgan talabga binoan dokumentni avtomatik tarzda qayta kodlashtira olishi kerak. Aks holda yoki sizning Web-bo’g’imingiz ayrim- kiruvchilar uchun o’qib bo’lmaydiga bo’lib qoladi, yoki Web-bo’g’imning bir nechta nusxalarini – har bir kodlash usuli uchun bittadan-tayorlash kerak bo’ladi.

Sahifani yangilab turish usuli. Odatda sahifalar FTP (File Transfer Protocol, Fayllarni uzatish protokoli) protokoli bo’yicha yangilab turiladi. Ayrim FTP-klientlar provayder kompyuteridagi fayllar bilan xuddi o’zining kompyuteridagi fayllar singari ishlash imkoniyatini beradi, yani nusxa olish, o’chirish, qayta nomlash va hokazo.

Odatda Web-bo’g’inni joylashtirish xizmatini provayder ozgina to’lov evaziga yoki butunlay bepul amalga oshiradi.

Web-bo’g’in ostida elektron pochta adresi va boshqa xizmatlar bilan joy ajratadigan xizmatlar mavjud. Qoidaga binoan bunday “bepul” joylashtirishning sharti bu reklama uchun sizning sahifangizdan joy olishdir. Undan tashqari sizning fayllaringizning o’lchamlariga chegaralar qo’yiladi.

Grafik dizayn.

Bu yerda biz dizayn haqida gapiramiz. Rasm chizish yoki kompozitsiyalar qila olish to’g’risida emas – buning uchun qobiliyat va ta’bning o’zi yetarli. Va bundan tashqari ko’gina foydalanuvchilar Photoshop yoki CorelDRAWni ozgina o’ragnib olib, o’zalarini dizayner deb hosoblasalarda, biz xech qanday grafik paket bilan ishlash xususiyatlari to’g’risida gapirmaymiz. Dizayn to’g’risida huddai fan singari gapiramiz. Yana aniq, aksioma va qoidalar yordamida tuzilgan, adashishlar va kashfiyotlar qilingan fan singari.

Siz faqat shu bobni o’qibgina dizayner bo’lmaysiz. Lekin biz o’laymizki, sizlar asosiy narsani tushunasiz: agar biror narsa yaxshi qilingan bo’lsa, u ob’ektiv sabablarga ko’ra yaxshi qilingan va boshqa savol, yaratuvchi bu narsaga nimani hisobiga erishdi – o’z qobiliyati yoki nazariy bilimlari hisobiga.

Biz o’quvchi shu narsani tushunishini hoxlaymiz: rassomlik qobiliyatiga ega bo’masdan turib ham dizayner bo’lish mumkin. Lekin buning uchun inson “nima yaxshiyu nima yomon”ligini aniqlaydigan qoidani juda yaxshi tushunib olish kerak.

Biz o’lcham, shakl, tekstura rangi, joylashishi va shrift kabi tushunchalar to’g’risida fikr yuritamiz. Yana kompozitsiya masalasiga ham tegib o’tamiz. Shakllantirishga urinib ko’raylik, tanlash, ayniqsa tanlashdan voz kechish nimaga asoslanadi. O’zingiz tushunasiz, ko’pincha variantlar juda ko’p buladi, yaxshilari esa sanoqli. Biz boshlovchi dizaynerlar yo’l qo’yadigan asosiy hatolarni va ularning kattaroq hamkasblarida bo’ladigan adashishlarni aks ettirishga harakat qilamiz.

Lekin bu dizaynga soddalshtirilgan qarash xolos. Agar bu sizga qiziqarli ko’rinsa, dizaynning asosiy savoli – “qanday qilib chiroyli qilish mumkin”ga bundan ko’ra yaxshiroq javob

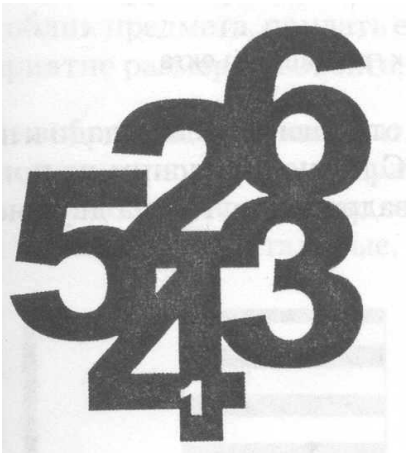
bera oladigan ko’plab kitoblar mavjud.

O’lcham. Olcham nimaligi hammamizga u yokibu darajada yaxshi ma’lum – maktab davrlarida geometriya bu to’g’rida aniq ta’rif bergan. Lekin inson his to’yg’usi uchun o’lcham aniq matematik ifodalangan tushuncha hisoblanmaydi. Maslan agar biz evkaliptning balandligini 100 metrga yaqinligini bilsak, bu bizga unchalik ko’p narsa bermaydi. Lekin agar bu 30-qavatli uyning balandligi ekanligini bilsak, u holda bu daraxtning balandligi to’g’risida aniq tasavvur hosil qila olamiz.

Shunday qilib o’lcham- nisbiy tushuncha. Biz uni 20cm, 3m, 5km sifatida emas, balki “mitti”, “o’rtacha”, “katta”, “ulkan”, “bahaybat” kabi qabul qilamiz. Hammasi inson qabul qiladigan sezgilarga asoslanadi, inson sezgisi esa juda egiluvchan. Biz jurnaldagi minatyurani ko’rayotganimizdagi katta va kichik tushunchalar katta kartinani ko’rayotganimizdagi katta va kichik to’g’risidagi tushunchalardan farq qiladi.

Ishdagi aniq bir ob’ekning o’lchamlarini tanlashda kompozirsiya birgalikda qabdaydir g’oyaga (umuman olganda, zamonaviy dizaynning asosiy vazifasi – ma’lumot yoki buyg’uni maksimal darajada yetkazib berish) ega bo’lishi to’g’risida o’ylash kerak. Shuning uchun, aytaylik, uyali telefon reklamasida aynan uyali telefon ajralib turishi kerak.

Kompozitsiyadagi biror bir ob’ektga e’tiborni nafaqat uning hajmini boshqa ob’ektlarga nisbatan kattaroq qilib jalb qilish mumkin, balki buning teskarisi: katta hajmdagi detallarga keskin qarshi turuvchi kichkina detalga ham jalb qilish ham mumkin. Bu holatda aynan u asosiy ma’lumot sifatida, qolganlari esa fon sifatida qabul qilinadi. Bu narsa biz tomoshabinlarning e’tiborini bitta raqamga jalb qilgan 1-rasmda ko’rsatilgan.

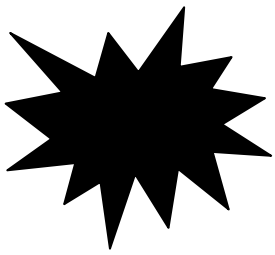


1-rasm. Kompozitsiyadagi ob’ektga e’tiborni nafaqat uning hajmini kattaroq qilib jalb qilish mumkin, balki buning aksi ham bo’lishi mumkin

Shuning uchun bitta kompozitsiyada yaxshi “ishlovchi” ob’ekt (ko’pincha) boshqasiga o’lchamlarini yazshilab o’gartirmasdan turib, mos kelmaydi. Buda ishni bajarayotganda nafaqat ko’zga (ko’z bilan olingan ma’lumotni qanchalik aldamchiligini professioanallar juda yazshi bilishadi), balki shakl, tekstura, va ranglarning odamning o’lchamni qabul qiluvchi sezgilariga ta’siri to’g’risidagi o’z bilimlariga ham tayanish kerak.

Shakl va o’lcham. O’lchamni qabul qilish ob’ekning shakliga bog’liq bo’ladi. Bu inson ko’zining yorug’lik interferentsiyasini qabul qilish xuxusiyatlari bilan bog’liq. Amalda murakkab tuzilishga ega, asosan bo’laklardan iborat ob’ektlarning o’lcham jihatidan qiyin baholanishiga ko’p bor duch kelish mumkin. Albatta, agar ularning nisbatan katta, qolgan kompozitsiya ob’ektlariga nisbatan gabaritlari solishtirsa bo’ladigan bo’lsa, bunday muammo yuz bermaydi. Lekin murakkab

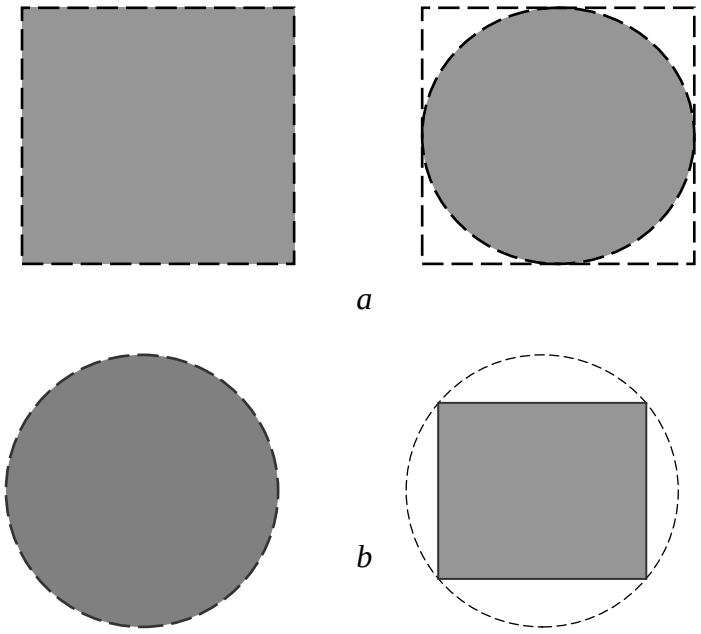
shaklning kichik bo'laklari umuman ob'ektning o'chamiga ta'sir qiluvchi qismi sifatida qabul qilinmaydi.



2-rasm. Bu figuradagi nurlar bu ob'ektning qirralari sifatida qabul qilinmaydi

Bu misol bizni shaklning o'lchamni qabul qilishga munosabatini belgilovchi asosiy tushunchaga olib keladi. Bu kompaktlilik yoki zichlik. Ba'zida eng zich figura sifatida adabiyotlarda doira keltiriladi. Bizningcha, bu juda ham to'g'ri emas.

Biz shaklning kompaktliligini uning maydonini belgilangan chegaralardagi to'ldirilmagan sohaga nisbati sifatida aniqladik. Murakkabmi? Misolda tushuntiramiz. Aytaylik, sizga ikki figura-to'rtburchak va doirani ikki tanga va markalarga joylashtirish kerak bo'lib qoldi (3-rasm).



3-rasm. Doira va kvadrat shaklidagi ob'ektlarni joylashtirish: *a* - markaga; *b* – tangaga

Ko'rinib turibdiki, marka uchun kvadrat kompaktroq figura hisoblanadi, tangu uchun esa doira.

Qabul qilishning mana shu nisbiyligini doimo yodda tutish kerak, chunki aynan kopaktlilik o'lchamlarni qabul qilishni belgilaydi. Sichroq figura doimo yirikroq ko'rinadi. Bunarsa 3-rasmda aniq ko'rinib turibdi, ayniqsa markada.

Bu hossadan amalda quyidagicha foydalaniladi. Aytaylik, ikkita masala turgan bo'lsin:

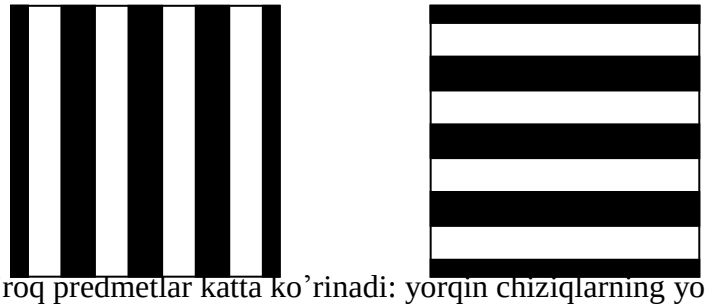
vizitkalarda foydalanish uchun shartli ravishda “AGGW” deb nomlanuvchi kompaniyaning logotipini chizish kerak bo’lsin. Agar siz kartochkada bu elementning muhimligini ta’kidlamoqchi, unga e’tiborni jalb qilmoqchi bo’lsangiz, u holda logotipning kompakt shaklda tayorlashga to’g’ri keladi (4-rasm, *a*). Agar logotip e’tiborni jalb qilmasligini, balki faqat fon sifatida ishlatilishini hohlasangiz, u holda sizning tanlovingiz (4-rasm,*b*) bo’lishi kerak. Va bu, e’tibor bering, vizitkaning bir hil o’lchamlarida.

a *b*

4-rasm. Logotip shakli: a – kompakt; b – kompakt bo’lmagan

Tekstura va o’lcham. Teksturaning qo’llanilishi yangi effektlarni yasash, predmet qiyofasini murakkablashtirish, unga ma’no berish imkoniyatini yaratadi. Lekin tekstura yana ob’ektning o’lchovini qabul qilishga ham ta’sir qilishi mumkin va buni albatta hisobga olish kerak.

5-rasmda bir xil geometrik o’lchamga ega ikkita kvadrat keltirilgan. Lekin shunga qaramasdan tik polosali kvadrat ancha “og’ir”, gorizontal tekislikka nisbatan yirikroq ko’rinadi. Gorizontal polosali kvadrat esa yengil ko’rinadi.



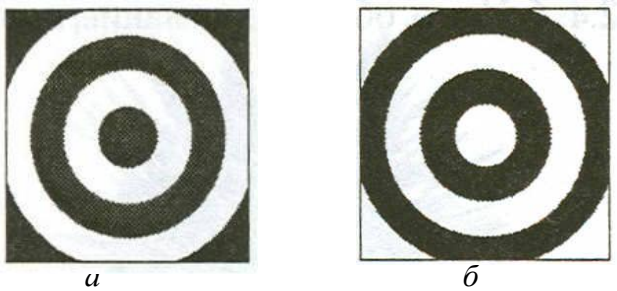
5-rasm. Yorug’roq predmetlar katta ko’rinadi: yorqin chiziqlarning yo’nalishi shu yo’nalish bo’yicha predmetning uzayishi kabi illyuziyani hosil qiladi

Bu effekt anchadan beri ma’lum. Eski jurnallardayoq to’la ayollarga vertikal polosali ko’ylaklar kiyish to’g’risidagi tavsiyalarni o’qish mumkin – ular ayollarni aslidaginga nisbatan oriqroq ko’ratish xususiyatiga ega.

Amalda bu hodisadan ko’p foydalaniladi. Ob’ektga qo’shimcha “mustaxkamlik” ebrish kerak bo’lsa, gorizontal yo’nalganligi yorqin ravishda ko’rinib turgan rasmlardan (aytaylik, g’isht taxlami teksturasi) foydalaning. Agar aksincha, bosib turuvchi o’lchamdan qutulmoqchi bo’lsangiz, vertikal yo’nalishga asoslaning.

Tekstura bilan yana bitta effekt bo’g’langan. 6-rasmdagi birinchi tasvir go’yoki bizdan uzoqlashib borayapti, ikkinchisi esa, aksincha, yaqinroq ko’rinadi. Bu effekt oq rangning o’zining fonidagi barcha predmetlarni “egishi”ga asoslangan. Bu effekni bilish va uning asosida ku kitobda ko’plab misollar qilingan aylanaviy gradiyentlar bilan ishlashda qo’llay bilish kerak.

Shunday qilib tekstura ob’ektlarni tasvirlashda ijobiy ham, salbiy ham (uni noto’g’ri tanlanganda) ta’sir qilishi mumkinekan. Bunga juda e’tibor bering va shunda bunga o’xshash narsa siz bilan sodir bo’lmaydi.

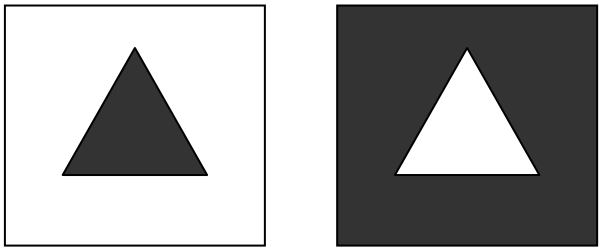


a *б*

6-rasm. Oq va qora aylanaviy polosalarning almashishiga asoslangan effektlar: *a* - uzoqlashish; *b* - yaqinlashish

Rang va o'lcham. Bu biz uchun o'lchamlarga bag'ishlangan eng murakkab qismdir. Masala yana o'sha – uni qabul qilishning nisbiyligidir. Ko'pincha biz yirikroq, boshqalari orasida ajralib tradiganlariga, garchi bunga ob'ektiv sabablar bo'lmasada, e'tibor beramiz.

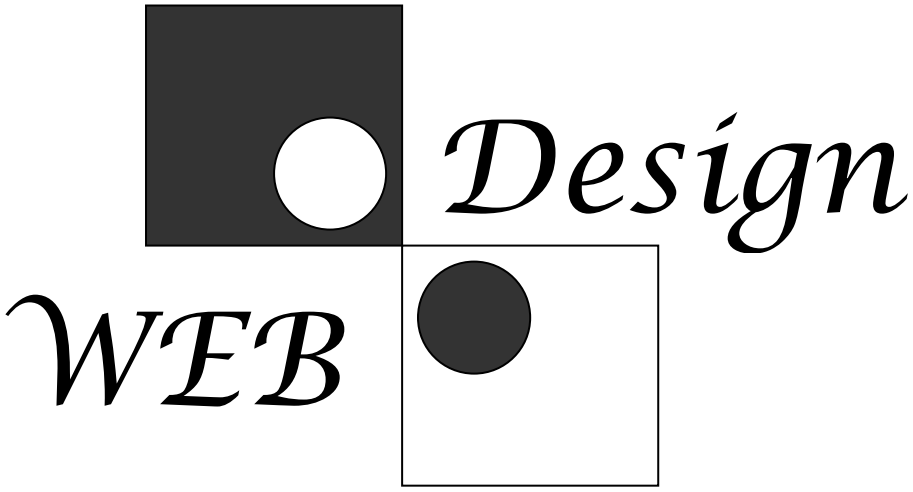
Shunday qilib, birinchi va eng asosiy hossa: qora fondagi yorqin ob'ektlar yorqin fondagi qora ob'ektlarga nisbatan yirikroq ko'rinadi (7-rasm).



7-rasm. Qora fondagi yorqin ob'ektlar yorqin fondagi qora ob'ektlarga nisbatan yirikroq ko'rinadi

Bir biridan yorqinligi bo'yicha keskin ajralib turuvchi, masalan, logotipdagi ob'ektlarning simmetrikligini ta'kidlamoqchi bo'lsangiz, buni albatta hisobga olishingiz kerak. Tomoshabinga siz o'z ishingizni unchalik sifatli bajarmagandek ko'rinishingiz uchun qora elementni kattaytirishga to'g'ri kelishi mumkin.

8-rasmda biz qora doira radiusini 2%ga orttirdik – yo'qsa logotip o'ylanganichalik chiqmasdi.



8-rasm. Bu ishda doiralar bir xil ko'rinishi uchun qora doiraning radiusi oqinikiga nisbatan 2% ga oshirilgan

Shunga o'xshash, yorqin-qizil element o'zi bilan teng to'q-ko'k elementga nisbatan ancha kattaroq ko'rinadi. Buni ham ish jarayonida e'tiborga olish kerak.

Rang. Biz rangli dunyoda yashaymiz. Har kuni, hatto uydan chiqmasdan turib ham, inson juda katta sondagi ranglarni ko'radi. Miz bunga ko'nikib qolganmiz va ranglarning tabiati, ular bizga va bizning hatti-harakatlarimizga qanday ta'sir qilishi haqida o'ylamaymiz ham. Garchi bu, balki unchalik to'g'ri bo'lmasada: har bir ayol qanday holda yorqin-qizil ko'ylak kiyishni, kulrang kostyum esa qanday holda to'g'ri kelishini juda yaxshi biladi.

Biz ranglarning asiri bo'lib yashaymiz. Bizning fikrlarimiz, his-tuyg'ularimiz – hammasi o'zining rangiga ega. Qora fikrlar, ko'k sog'inch, "zangori kayfiyat" kabilarni esga olish quyidagilarni tushunish uchun yetarli: predmetning psixologik qabul qilinishi uning rangi bilan yetarlicha qattiq bog'langan.

Odatiy hayotda bu unchalik katta ahamiyatga ega emas. Lekin hozir gap ranglarning kompyuter grafikasida qo'llanilishi to'g'risida borayapti va bu yerda biz nafaqat ranglardagi farq, balki bir rangning darajalarini ham his qila olishimiz, ularni qo'llanilishi inson tomonidan qabul qilishga qanday ta'sir qilishini tushunishimiz kerak.

To'g'ri tenlangan ranglar tasvirga e'tiborni jalb qilishi ham, qaytarishi ham mumkin. Siz quvonch, qiziqish, sog'inch, qo'rquv, zerikish kabilarni ranglarnigina o'zgartirgan holda hosil qilishingiz mumkin.

Ranglar juda ko'p, lekin har bir kishining, psixologlarning ta'kidlashlaricha, tanlanishi har bir shaxsning xususiyatlariga bilan bog'liq bo'lgan sevimli ranglari bor. Shuning uchun ham ranglarni tanlashda o'racha tomashibinning tavsiya qilingan psixologik portretiga ham suyanish kerak.

Qo'shimcha murakkablik shundan ham kelib chiqadiki, rang ob'ektiv fizikaviy kattalik sifatida tabiatda bo'lmaydi. Ranglarni his qilish, garchi elektromagnit nurlanishlarning ob'ektiv faktorlari ta'sirida ro'y bersa ham, subektiv narsa hisoblanadi. Bundan tashqari ranglarni tasvirlashda turli mamlakatlarda milliy-madaniy an'analarga asoslangan turlicha rang modellaridan foydalaniladi. Kompyuter grafikasi bilan professional tarzda shug'ullanadigan har qanday kishi duch keladigan ranglarni tasvirlashning rang-barangligi shu bilan tushuntiriladi.

Shakldan farq qilib, subektiv tushsunchaligiga qaramay, ranglar olamida dizayner bilishi va amalda qo'llay olishi kerak bo'lgan umumiy qonuniyatlar mavjud.

Biz oldin rang qanday tuzulganligini tahlil qilib olishimiz kerak. Aslida bu jiddiy monografiyaning mavzusi, shuning uchun biz fiziologik va spektral xususiyatlariga to'xtalib o'tirmaymiz, balki hammasini soddalashtiramiz.

Dastlab rangning tashkil etuvchilarga ajratamiz. Barcha mavjud modellardan faqat HSV (Hue – Saturation – Value, Rang – To'yinganlik – Yorqinlik) gina uni biz odatlangan holda tasvirlaydi va ko'nikishni talab qilmaydi.

HSV tizimi uchta komponentaga ajratiladi.

♦ *Rang* (Hue) – bu bevosita rang to'g'risidagi ma'lumot. Buni oddiy tilda tushuntirish qiyin, shuning uchun o'z intuitsiayngizga ishoning.

♦ *To'yinganlik* (Saturation) – ranglar, kundalik hayotdan siz bilasizki, u yoki bu darajada to'yingan bo'ladi. Oddiy tilda ko'proq to'yingan rangni ko'proq suvli deyishadi.

♦ *Yorqinlik* (Value) – osongina topish mumkinki, yorqinroq rang yorug'roqdek qabul

qilinadi.

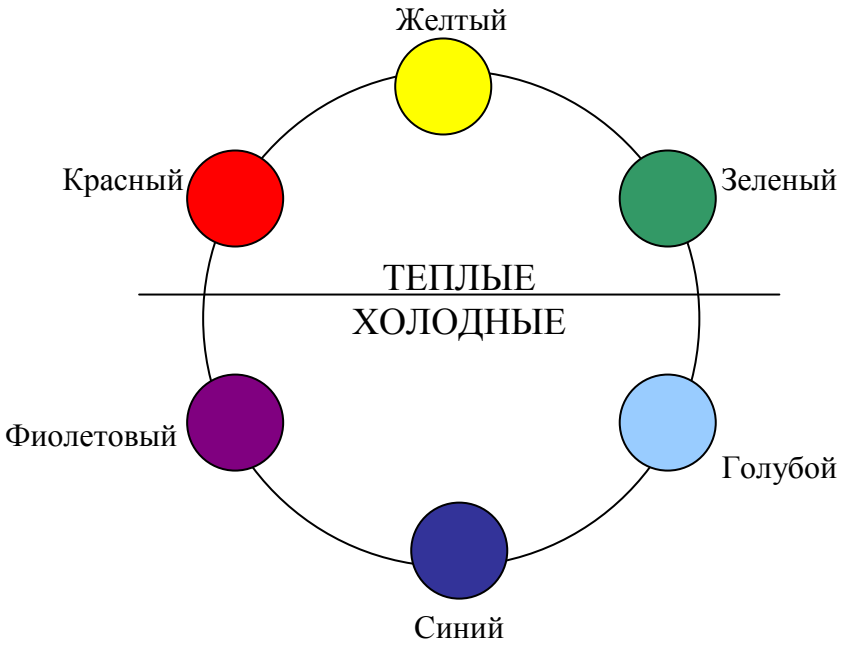
Shuni ta’kidlash kerakki, nafaqat turli kishilar, balki butun xalqlar bir xil rangga turlicha munosabat bildirishadi. Bu shu xalq tarbiyalangan an’analar bilan bog’liq. Masalan, evropa mamlakatlarida oq rang – tozalik va begunohlik rangi, bazi bir sharq xalqlarida esa u motam rangi hisoblanadi.

Asosiy ranglarni xarakterlashga urinib ko’ramiz.

Ranglarning issiq va sovuqqa bo’linishi 9-rasmda tasvirlangan. Bu bo’linish ma’lum darajada shartli: rang qanchalik yuqori va quyi yarim doiralarga yaqin bo’lsa, uni issiq yoki sovuqqa ajratishga ishonch shuncha kamroq.

Qizil

Ko’pchilik kishilarda qizil rang olov bilan to’g’ridan-to’g’ri bog’lanish hosil qiladi. Shuning uchun ham uning ta’siri ham har xil bo’lishi mumkin – iliqlik tuyg’usidan tortib qo’rquvgacha. U pulsni tezlashtirish va qorachig’ni kengaytirish hossasiga ega, lekin o’zining yorqinligi tufayli tez charchatadi, ayniqsa ko'p miqdorda bo'lsa (masalan, agar xona devorlari shu rangda bo’lsa).



9-rasm. Ranglarni sovuq va issiqqa bo’linishini ko’rsatuvchi aylana doirasi

Toza qizil rang – bu havotirlanish va hayajonlanish rangi, lekin uning turlari (jigarrang, qizg’ish-) tnchlantiruvchi ta’sir qiladi. Qizil o’ziga diqqatni jalb etadi (deyarli barcha ogohlantiruvchi belgilar qizil fonda yoki qizil yozuvda qilinishi tasodifiy emas).

Bu rang kompyuter grafikasida ko’p qullaniladi, lekin charchatib qo’ymasligi uchun u bilan ishlashda ehtiyot bo’lish kerak.

Qizil rang to’g’ri keladi:

- Agressivlikni, faollikni ta’kidlash uchun. Masalan, eng yaxshi sport mashinalari, jumladan Ferrari asosan qizil rangda ishlab chiqariladi;
- Kuchli hisni, seksual hoxishni. Buni shu bilan tasdiqlash mumkinki, erotik dasturli tungi klublar bezagida juda ko’p qizil rang ishlatiladi;
- Ayovsizlikni ifodalash uchun (qizil – qon rangi);
- Boylik, dabdabalilikni ta’kidlash uchun (ayniqsa qora rang bilan birgalikda).

Sariq.

Bu rang doirasidagi eng yorqin rang. U yaxshi kayfiyat, umidvorlikni uzatish uchun juda mos rang. Aynan shuning uchun sariq – turistik kompaniyalar reklamasida eng ko’p qo’llaniladigan ranglardan biri.

Bundan tashqari sariq rang – bu oltin rangi. Shuning uchun u ko’pchilikda omad, boylik va dabdaba tuyg’ularini vujudga keltiradi.

Binafsha rang.

Iliq, musbat, yorqin va zamonaviy rang, tetiklantiruvchi ta’sir qiladi. Pulsni tezlashtiradi va qorachig’ni kengaytiradi. Zamonaviy duzaynda, ayniqsa web-dizaynda ko’p qo’llaniladigan ranglardan biri.

Binafsha rang, agar hoxlasangiz, quyidagilarni ta’kidlash uchun ishlatilishi mumkin:

- zamonaviylik. O’zining brendining asosiy qirrasi sifatida zamonaviylikni tanlagan ko’plab kompaniyalar “firma rangi” sifatida binafsha rangni tanlshadi. Bu ayniqsa uyali telefon operatorlari orasida keng tarqalgan;
- harakatchanlik;
- optimizm. Yorqin plakatlarni yaratish uchun binafsha rang bachkana, lekin amaliy jihatdan ideal variant.

Tajrbalarning ko’rsatishicha, binafsha rang yashil rang bilan birga juda yaxshi ko’rinadi.

Lolarang.

Ranglar doirasini qizil rangdan ko’k rangga borgani sari biz bu sekin-asta o’tishni payqamay qolamiz, axir huddi o’sha yerda ajoyib rang berkinib turibdi – lolarang. Bu qabul qilishga juda qiyin rang, chunki u tabiatda deyarli uchramaydi. U huddi ko’k va yashil ranglarga o’xshab sovuq guruxga kiritiladi. U torlik, chegaralanganlik tuyg’usini hosil qilish xususiyatiga ega. U yana tez charchatadi va faollikni susaytiradi.

Lolarang “erlik emas”, unga qandaydir yashirinlik xos. Yaqingacha mistikada lolarangga alohida o’rin ajratilgan. Agar siz ko’zboylog’ichlarning tamoshalarini ko’rgan bo’lsangiz, ularning kiyimi, narsalari, pardalarida bu rang juda ko’p. Lolarang kishilarda hurofiy, tushunatsiz qo’rquvni hosil qila oladi.

Shunday qilib lolarang quyidagilar uchun to’g’ri keladi:

- mistik kayfiyatni hosil qilish uchun;
- yashirinuvchanlikni ta’kidlash uchun.

Ko’k

Ko’k rang sovuq guruxga kiradi, rang doirasining eng quyisida joylashadi. U tinchlantiradi, ba’zan melanxolik kayfiaytni tarqatadi.

Bu rangni “doimiy muzlik” zonasi ranglariga kiritish mumkin: u sovuq va tozalik sezgisini juda yaxshi uzatadi. Tinchlantiruvchi ta’sir qiladi, yotoqxonalarning dizayni uchun juda mos.

Toza ko’k rangning to’yinganligi va yorqinligini o’zgarishi bizga juda ko’p ranglarni berishi mumkin (to’g’ri, CMYK-tizimning o’ziga xosligi tufayli aynan ko’k rang bosib chiarishda yaxshi tasvirlanmaydi).

- Sokinlikni ta’kidlsh uchun ko’k rangdan foydalaning;
- tozalikni. Siz e’tibor bergan bo’lsangiz kerak, barcha sifatli tozalovchi vositalar ko’k yoki yashil rangda bo’ladi. Bu tasodifiy emas: olimlarning isbotlashicha aynan shu rang ko’pgina kishilarda tozalik tuyg’usi bilan mos keladi.

- mustaxkamlik.

Zangori

Ajoyib rang. U ham iliq, ham sovuq bo'lishi mumkin, ammo ko'pincha u birlashgan joyda yotadi, shuning uchun ham uni ham sovuq guruxga, ham iliq guruxga kiritish mumkin.

Zangori rang tinchlantiruvchi ta'sir qiladi, qon bosimini pasaytiradi, qon aylanishini normallashtiradi. Bu eng tabiiy va eng "tirik" rang. Uning dizayndagi asosiy vazifasi aynan shunda – ob'ekt bilan tabiat aloqasini uzatish. Umuman olganda, logotiplar ko'rib chiqilsa, zangori rang asosan tabiat resurslarini qaziboladigan kompaniyalarda yoki ekologik tashkilotlarda uchraydi.
Agar siz zamonaviy blokbasterlarni ko'rsangiz, u holda biologik qurollar, boshqa dunyo jonzotlari va boshq biologik mavjudotlar yzshil (ayniqsa yorqin yashil) rangda bo'lishiga e'tibor bergan bo'lishingiz kerak.

Bundan tashqari zangori rang – mistika va yashirinlilik rangi.

Zangori rang quyidagilarni uzatish uchun mos keladi:

- barcha biologik ko'rinishdagi tiriklik;
- tabiat bilan aloqa;
- yashirinlik.

Yashil.

Yashil rang tinchlantiradi va sobutadi. Bunday effect sovuq suv va muzga o'xshashligi bilan tushuntiriladi. Ba'zan yolg'izlik tuyg'usini vujudga keltiradi.

Asosiy ranglar: qora va oq.

Siz sezgan bo'lsangiz kerak, biz barcha asosiy ranglarni ko'rib chiqdik, lekin rang aylanasining hamma yerida bo'lgan va shu bilan hech qaerda ko'rinmaydigan ikkita rang – qora va oq ranglarga tegishmadik.

Qora rang o'zining takrorlanmas yaxlitligi bo'yicha – o'zida zerikish, alamni olib yuruvchi og'ir rang. Charchoq va o'ng'aysizlikni keltirib chiqaradi. Shunga qaramasdan tez-tez kishilar shu rangdagi kiyimlarni tanlashadi. Bu holda u klassikaga, shu bilan birga aniq bir stil hosil qilgan holda, kiradi. Yana bu rang qolgan barcha ranglar bilan u yoki bu darajada kelisha oladi. Qora rang –bu dabdaba rangi, ayniqsa qizil rang bilan birgalikda. Bizning an'analarda uni motam rangiga kiritish qabul qilingan.

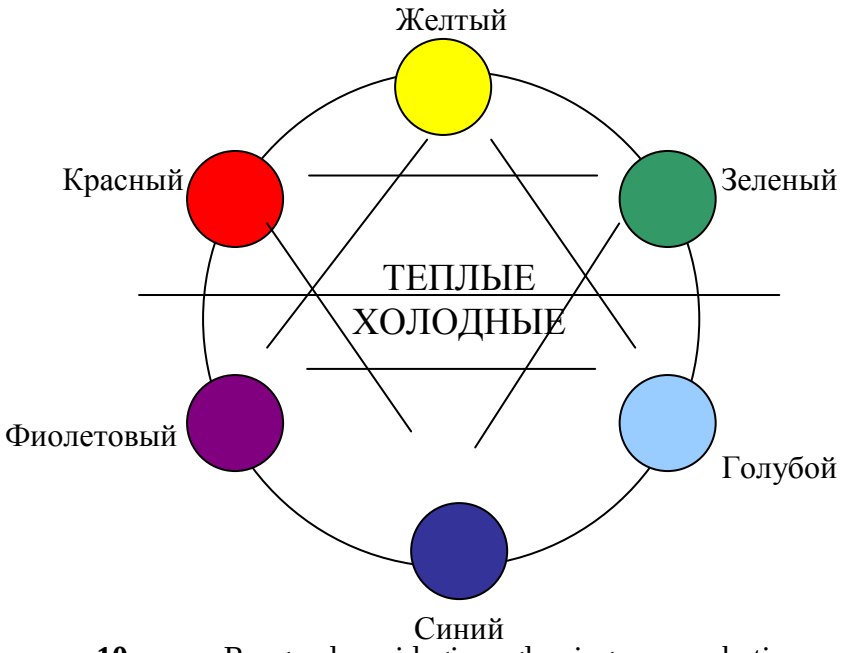
Oq rang – juda quvnoq rang. Toza havo bilan bog'liqligi tufayli u engillik, erkinlik va vaznsizlik tuyg'ularini hosil qiladi. Pulsni tezlashtiradi va qorachig'ni kengaytiradi. Oq rang fonni yaratishda anchagina ko'p qo'llaniladi. Oq rang o'zicha xech qanday ma'lumot yetkazmaydi, boshqa barcha ranglar bilan yazshi moslashib, yorqinroq rang ohanglarini hosil qiladi. Bu rangni begunohlik va tozalik rangiga kiritishadi.

Ranglarning o'zaro mosliklari.

Ranglarning ozaro mosligi masalasi – bu dizayndagi eng baxsli va turli xilcha talqin etiladigan savollardan biri. Haqiqatda bu yerda "har kimning ta'bi har xil" tamoyili hamma yerdagidan kuchli. Shuning uchun universal qaoidalar to'g'risida, afsuski, gap ham bo'lishi mumkin emas.

Lekin qabdaydir qonuniyatlarni topish mumkin. Shunday qilib, birinchi va eng sodda tamoyil – raqin ranglarni tanlash. Kiyimni "ohang" bo'yicha tanlanganidek, web-sahifaga ham ranglarni tanlash mumkin.

To'g'ri, bunday yondashuv juda qoloq va prfessional ishlar uchun to'g'ri kelmaydi. O'zaro mos keladigan ranglarni qidirish uchun ranglar doirasidan foydalanish qiziqarliroq. 10-rasmga qarang.



10-rasm. Rang aylanasidagi ranglarning munosabati

- Aralash ranglar yomon o'rin almashmaydi, ammo bunday variantni tanlash odatda zerikarli va bachkana hisoblanadi.
- Bir-biriga qarama-qarshi joylashgan ranglar yomon kelishadigan rangklar hisoblanadi. Yagona istisno- ko'k rang sariq bilan juftlikda yomon ko'rinmaydi.
- Eng yaxshi tanlov – 2.10 rasmda to'g'ri chiziq bilan tutashgan ranglardan bitta keyingi rang. Bu ranglar o'zlarining ko'plab turlarida juda yaxshi mos keladi, asosiysi yorqinlikda juda katta farq bo'lmasligi.

Yana bir muxim narsa, qora va oq qolgan barcha ranglar bilan, ayniqsa bir-biri bilan juda yaxshi mos keldi. Shuning uchun, agar tanlash imkoniyati bo'lsa, ishni aynan shu rangllar yordamida qilish osonroq.

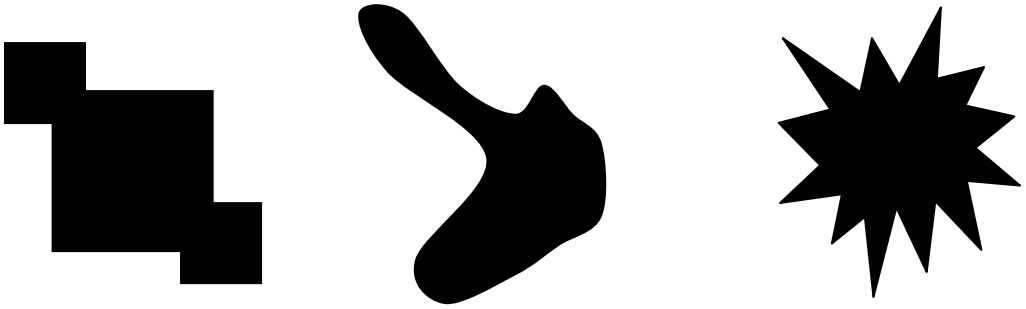
Lekin ranglarni tanlashda asosiy orientir bo'lib baribir ta'b (kimdadir bu tug'ma bo'ladi) va tajriba (vaqt o'tishi bilan orttirilgan) bo'lishi kerak. Ahir xech qanday ranglar to'g'risidagi nazariyalarga tusmaydigan, lekin sifatli va talantli dizaynerlik ishining namunasi bo'la oladigan juda ko'plab yechimlar mavjudku.

Shakl.

Shakli – bu har qanday ob'ektning eng muxim qismidir. Biz rang, tekstura, o'lcham to'g'risidagi ma'lumotlarni tushurib qoldirishimiz mumkin, lekin shakl to'g'risida doimo eslashga to'g'ri keladi. Har qanday dizaynerlik ishi ob'ektlar uchun shakllarni tanlab olishdan boshlanmog'i lozim.

"Shakl" tushunchasiga ta'rif berish ancha murakkab. Agar ozgina orttirib gapirsak, bu ob'ektning barcha geometric munosabatlari yig'indisidir. Judayam ko'p miqdordagi shakllar bor, shuning uchun ularni faqat qurilishiga ko'ragina sinflarga ajratish mumkin.

- *Ko'pburchaklar*. Bunday figuralar to'g'ri chiziqlardan quriladi. Ularga biz o'rganib qolgan to'g'ri chiziq, uchburchak, kvadrat, yulduzchalar ham, undan murakkabroq figuralar ham kiradi (11-rasm, a).
- *Egri chiziqlilar*. Bu figuralar silliqlangan chiziqlarga asoslangan figuralardir (11-rasm, b). Ularga doira, oval, yoy va boshqalar kiradi.
- *Amorf* . Bular aniq bo'lmagan murakkab shakllardir (11-rasm, c). Amorf figuralar teksturalarga juda yaqin bo'ladi, shuning uchun ba'zida ularni differentsiallash qiyin bo'ladi.



а *б* *в*

11-rasm. Figuralarning uch tipi:
a – ko’pburchak; *b* – egri chiziqlilar; *c* - amorf

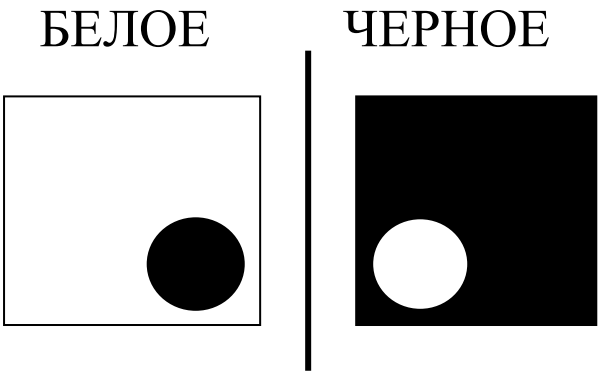
Shakllar bilan ishlash – bu dizaynerlik ishining eng oddiy va eng qiziqarli qismidir. Shakllarning mosligi to’g’risda ko’p avlodlarning tajribalari mavjud, shuning uchun ko’pchilikda shakllar bilan ishlaganda (rang bilan ishlagandagidan farq qilib) o’z intuitsiyasi yetarli bo’ladi. Lekin shakllarning asosiy formalari to’g’risida baribir aytamiz.

Chiziq.

Chiziq – geometrik figuralarning eng soddasi. Nazariy jihatdan u faqat bitta o’lchamga – uzunlikka ega bo’ladi. Garchi amalda ularning rangi, tipi (aytaylik, punktir chiziqlardan foydalanish keng tarqalgan) va yo’g’onligini ham berishga duch kelsak ham. Ba’zida chiziq bilan to’g’ri to’rtburchak o’rtasidagi farqni bilish qiyin.

Chiziqlarning ikkita asosiy funktsiyalarin bor: boshqa ob’ektlarni ajratish va birlashtirish.

Ajratish – kitoblarni bezashda keng rivojlangan va rivojlanayotgan klassik usul hisoblandi. Dizaynda ham u juda faol ravishda ishlatiladi (12-rasm).

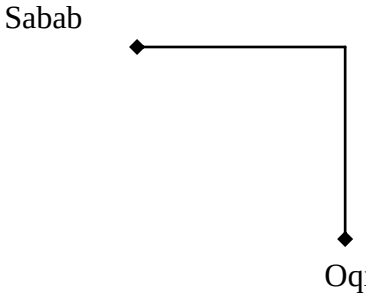


12-rasm. Chiziqlarni ajratish uchun foydalanish

Lekin chiziqlarning qo’llanilishi (tipografiya terminologiyasida ajratish chiziqlari deb ataladi) bachkana va judayam oshkora usul hisoblanadi. Agar sizga ishni oddiy va tez qilish kerak bo’lsa, bu yaxshi variant, lekin agar ajratish chiziqlari yordamida original nimadir olmoqchi bo’lsangiz, ishlashga to’g’ri keladi.

Aytayluk, matn bloklarini ajratish uchun fondagi ranglar farqidan yoki bo’sh bloklardan foydalanish qiziqarliroq.

Chiziqlarning birlashtirish funktsiyasi qiziqarliroq. Aytish mumkinki, bu zamonaviy dizaynning eng yorqin va keng tarqalgan qirralaridan biri. Chiziqlar bu holda ma’lumotlar bliklari yoki grafik komponentalar o’rtasidagi muxim o’zaro ta’sir funktsiyasini bajaradi (13-rasm).



13-rasm. Chiziqlarning birlashtiruvchi funktsiyasi

Chiziqlar –bu hohlagan vaqtda yordamga kelishga tayyor bo'lgan o'ziga xos "kaltak-qutqaruvchi". Bu yaxshi mutaxassisning ishida doimo bo'lishi kerak bo'ladigan ikki narsa: yaxlitlik va dinamizmni kompozitsiyaga kiritishning eng oddiy usuli.

Ishda chiziqlar tomoshabinlar uchun bir elementdan ikkinchisiga o'tishni ko'rsatib turuvchi yo'naltiruvchilar rolini o'ynashlari kerak. Shuning uchun ulardan foydalanishda nafaqat tashki jozibani, balki mantiqiy asoslanganlikni ham o'ylash kerak.

To'g'ri to'rtburchak.

To'g'ri to'rtburchak – dizayn uchun butunlay o'ziga xis figura, kompyuter dizayni uchun ayniqsa. Va buning sababi oddiy. Ko'pgina saqlovchilar: plakatlar, kitob sahifalari- deyarli barcha poligrafik maxsulotlari, kompyuter ekrani to'g'risida gapirmayoq, to'g'ri to'rtburchak shaklida bo'ladi. Bunday holat bu figurani avtomatik tarzda eng ko'p ishlatiladigan, eng keng tarqalgan figura qilib qo'yadi.

To'g'ri to'rtburchakka asoslangan dizayn – bu eng oddiy va oshkora ko'rinib turgan yo'l. Buning isboti sifatida siz o'nlab web-sahifalarni ko'rishingiz mumkin va ishonch hosil qilishingiz mumkinki, ularning barchasi shu formadan kelib chiqqan.

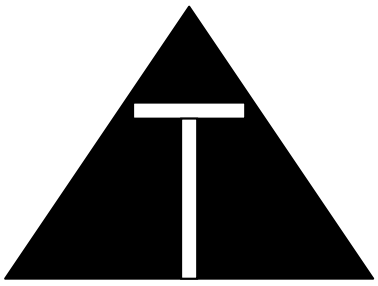
To'g'ri to'rtburchaklar bilan ishlaganda eng muximi – proportsiyani to'g'ri tanlash. Kvadratga yaqin keluvchi figuralar unchalik yaxshi variant hisoblanmaydi: bunday yechim o'zining simmetrikligi to'fayli modadan qolgan hisoblanadi. Lekin judayam jo'zilganlari ham yomon: agar gorizontal bo'yicha bo'lsa, "yerlashganligi" tufayli, vertikal bo'yicha cho'zilgan bo'lsa, muvozanatsizligi tufayli.

Qadimdan oltin kesim deb ataluvchi to'g'ri to'rtburchak tomonlarining eng yaxshi nisbati ma'lum. Bu nisbat 0,618 ga teng. Albatta, bu har qanday vaziyatdan chiqish degani emas, lekin uni albatta hisobga olish kerak – butun klassik arxitektura oltin kesimga asoslangan.

Uchburchak.

Bu figura ko'p jihatdan yaxshi, lekin chiziqlar yoki to'g'ri to'rtburchaklar singari keng tarqalmagan. Sababi shundaki, uchburchak kompozitsiyadagi qolgan figuralar bilan qiyin moslashadi.

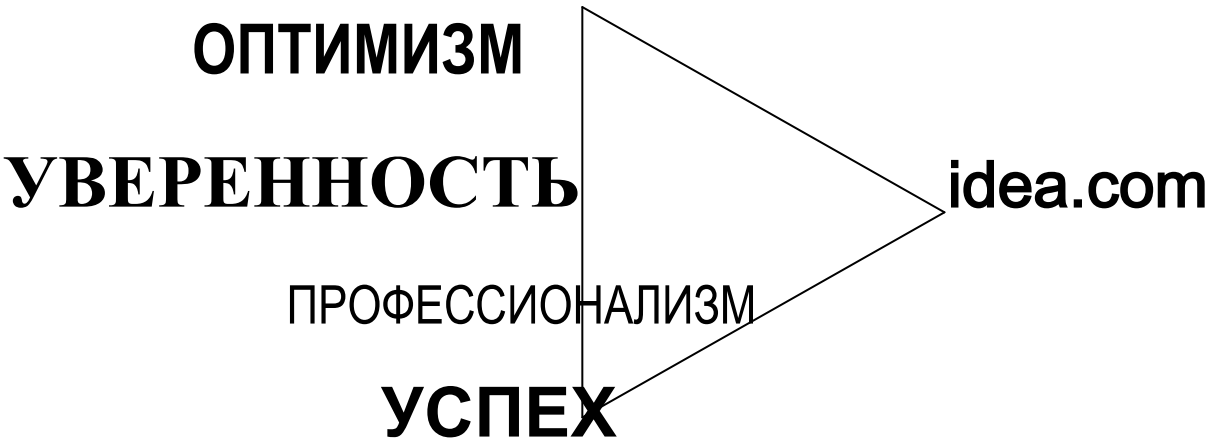
Uchburchak, aytaylik, logotip asosida yomon ko'rinmaydi (14-rasm).



14-rasm. Uchburchak asosidagi logotipga misol

Bu holatda figura asosining pastga qo'yilgankigi ajoyib tuyg'uni beradi – figuraning, va bu esa o'z navbatida kompaniyaning mustaxkamligini. Bunday effekt anchadan beri ma'lum, shuning uchun butun dunyodagi ko'plab firmalar uchburchakni o'z belgilari qilib tanlaganlar.

Bu figura bilan erishiladigan ikkinchi effekt – "ko'rsatuvchi " effekti. Uchburchakning strelkani – biz ko'nikkanimizdek, yo'nalishning standart ko'rsatkichini eslatishi uni tomoshabinlar e'tiborini yo'naltiruvchi element sifatida ishlatishga imkon beradi. Masalan, 15-rasmda shartli idea.com saytidagi so'zlarning qiymatlariga e'tibor qaratilgan.



15-rasm. Fondagi uchburchak bu misolda ko'rsatkichli strelka vazifasini o'ynaydi

Fon sifatida ishlatiladigan uchburchak yordamida biz bir tomondan saytning sifatlarnini ajoyib tarzda gruppalay oldik, ikkinchi tomondan – bu ma'lumotni birdaniga sayt nomiga yo'naltira oldik. Boshqa usullar bilan bunga erishish oson emas.

Doira.

Doira ko'pgina rivojlangan jamiyatlarda eng mukammal, ilohiy figura hisoblanilgan. Doira quyoshning belgisi bo'lgan, maqbara va qurbonlik uchun mexroblar doira shaklida qurilgan.

Zamonaviy dizaynda bu figuraga xurmatdan iz ham qolmagan. Doira sahifalarning va ekranning to’g’ri to’rtburchak shakli bilan judayam kontrastga kirishadi.

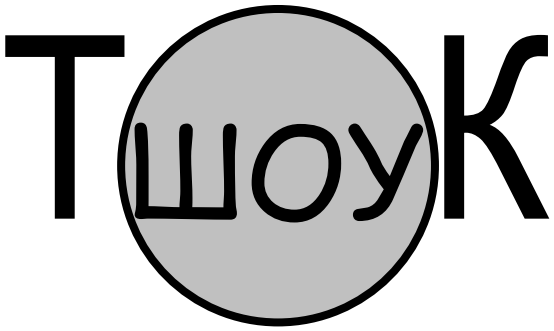
Albatta, kontrast – bu unchalik yomon emas, lekin kompyuter dizayni sohasida kontrast asosida sifatli ishlar bajarish – bu haqiqiy professional mutaxassislargina bajara oladigan narsadir.

16-rasmda bosmoxonalardan birining saytini birinchi sahifasi ko’rsatilgan. Sahifaning yaxshichiqishiga sabab shuki, rubrikator aylananing butun uzunligi bo’yicha tarqatilmagan, uning bir tomoni bo’ylab kompakt tarzda joylangan. Bundan tashqari yuqoridagi yirik gorizontal matnli blok kompozitsiyaga stabillik qo’shadi. Lekin baribir brauzer oynasida sahifa unchalik muxim ko’rinish olmaydi.

16-rasm. Aylana asosida qurilgan web-sahifa

Siz o'zingiz eslashga urinib ko'ring, aylana asosida qurilgan saytlarni ko'p ko'rganmisiz. Bu sahifani original tarzda bezatishning eng yaxshi usuli, lekin shu bilan birga uni buzub qo'yishning ham eng oddiy usuli hisoblanadi.

Hozirgacha doira o'zini to'la oqlaydigan yagona narsa bu logotiplarni yaratishdir. Mimkinki, deyarli barcha firmalarning belgilarining yarmi o'zining asosida doirani qabul qilgan. GARCHI, faqat ba'zilarini istisno qilganda, bu ishlar ma'no jihatdan ham, bajarilishi jihatdan ham originalligi bilan ajralib turmasada. Eng keng tarqalgan misol – aylana yordamida "O" harfiga o'xshatish (17-rasm).

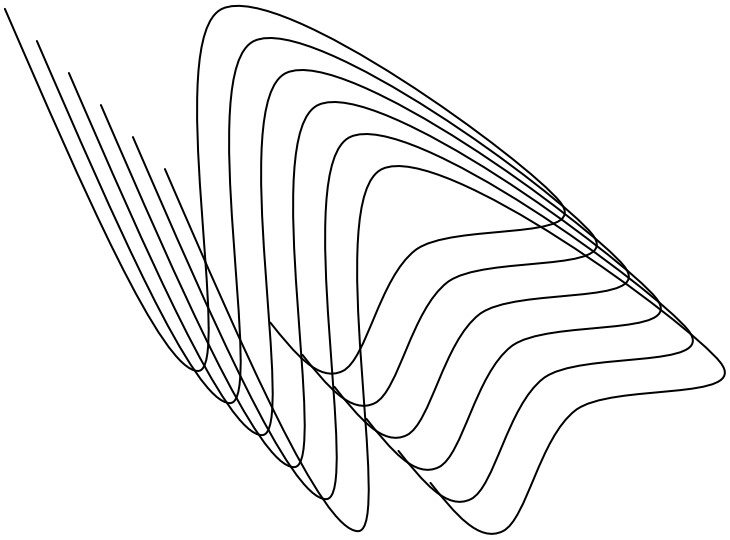


17-rasm. Aylana yordamida "O" harfiga o'xshatish – bu figuradan dizaynda muvaffaqiyatsiz foydalanishga misol

Bir so'z bilan aytganda, agar sizda tajriba kam bo'lsa, bu figura bilan hozircha ishlamagan ma'qul. Qachon yahshi mutaxassis bo'lganingizda aylana bilan ishlash siz uchun bartaraf qilib, original natijalarni olsa bo'ladigan qiyinchiliklarni chaqiruvchisi bo'lib qoladi.

Egri chiziqlar.

Egrilik - bu bir nechta har xil darajadagi egrilikka (matematik til bilan u ikkinchi tartibli egri chiziq deyiladi) ega bo'lgan chiziq (18-rasm).



18-rasm. Ikkinchi tartibli egri chiziqlardan tuzilgan figura

Zamonaviy kompyuter dizaynida bu figuralar yetarli darajada keng qo'llaniladi (asosan ular to'g'ri chiziqlar bilan bir xil maqsadlarda ishlatiladi). Ammo ular bilan haddan tashqari qiziqib bo'lmaydi: siz asrning boshlarida urf bo'lgan Modern stilida bajarilgan ishni olishingiz mumkin, bu esa unchalik yaxshi emas.

Formaning yo'qligi.

Albatta, formaning yo'qligi tug'risida gapirish unchalik ham to'g'ri emas- forma doimo mavjud bo'ladi. Lekin uning murakkabligi tahlil qilish imkoniyatini yo'q qilsa, bunday ob'ektni formasiz deb atash mumkin.

Formasiz ob'ektlardan foydalanish – to'g'ri burchakli formalarning qat'iy mantiqiyligiga qarama –qarshi qo'yiluvchi zamonaviy dizaynning yana bitta ustunligidir. Va u undan u yog'iga borish mumkin bo'lmagan simmetriyaga qarshi kurashning chegarasidir.

Formasizlikni quyidagilarni ta'kidlash uchun ishlatish mumkin:

- mustaqillik;
- ultra zamonaviylik;
- ommaviylik;
- etiroz;
- nostandartlilik.

Formasizlik ayniqsa shrift bilan hamkorlikda yorqin namoyon bo'lishi mumkin. Bu ko'rinishda u insonning originalligi, agressivligini juda yaxshi aks ettiradi.

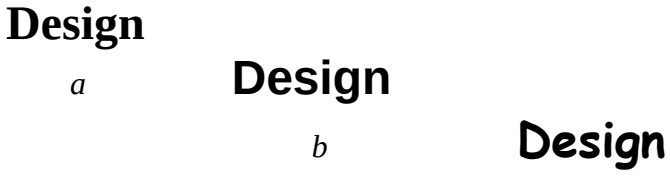
Formaning yo'qligi kamchilik hisoblanmaydi, ko'proq aksincha – axir boshqa xech nima original yechimni topishda bunchalik keng imkoniyatlar bermaydi. Bu yerda ob'ektlarning kompanovkasi figuralarni tanlashdan kam ahamiyatga ega bo'lmaydi. Ammo bu haqda keyinroq gapiramiz.

Shriftli dizayn.

Shriftlar biln ishlash – dizayner ishining eng qiziqarli, ko'proq ko'prejali va eng murakkab qismidir. Hoziroq aytamiz, agar siz matnlar bilan u yoki bu darajada professional ishlamoqchi bo'lsangiz, shriftlarni dizayni bo'yicha maxsus adabiyotlarni sotib olish va ularni o'rganishga ancha vaqt sarflash kerak bo'ladi.

Shunday qilib shriftlar asosan uch tipli bo'ladi:

- kesimlar bilan (20-rasm, *a);*
- maydalangan (20-rasm, *b*);
- erkin stilda (20-rasm, *c).*



a *b* *c*

20-rasm. Shriftlarning uch tipi:
a- kesimlar bilan; *b* - maydalangan; *c* – erkin stilda

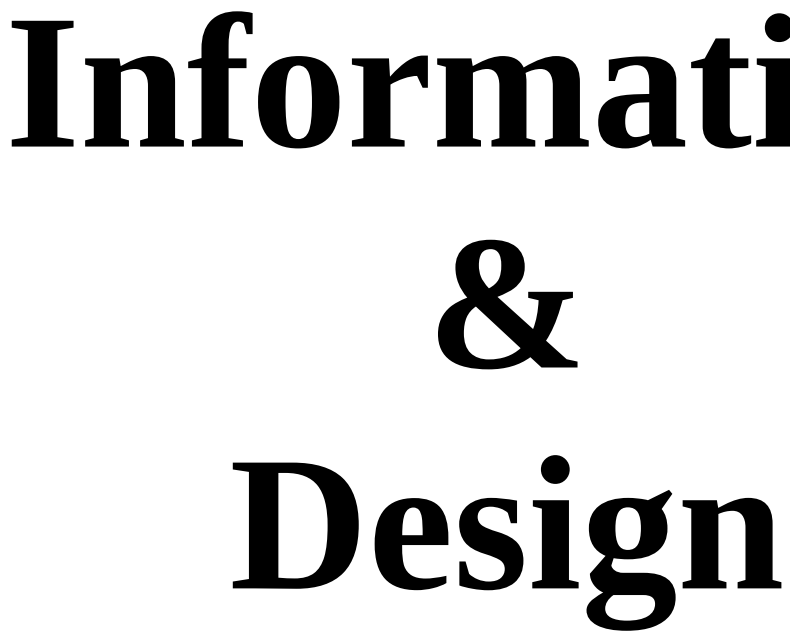
Garnitura

Shriftlarning dizaynida biz yuqorida gapirgan yaxlitlik va kontrastning o'sha tamoyillari ishlaydi. Kesimli va maydalangan shriftlar yaxshiroq mos keladi. Bunga ular professional ishlarda qanchalik ko'p ishlatilishini ko'rib, ishonch hosil qilish qiyin emas.

Maydalangan shriftlar qo'yozma ko'rinishida ishlangan shriftlar bilan ham yomon mos kelmaydi, lekin bu absolyut qoida emas. Dekorativ va kesimli shriftlar bir-biri bilan juda yomon moslashadi- bunday qo'shnilikdan qochishga harakat qiling.

O'lchov.

Yozuvlarning o'lchovini tanlash anchagina murakkab hisoblanadi. Dastlabki keladigan fikr – agar birorta yozuv katta bo'lsa, u holda ko'ruvchilar tomonidan ham unga qiziqish katta bo'ladi. Lekin bu unchalik emas.



21-rasm. Judayam yirik matn ko'ruvchi uchun informativlik funktsiyasini yo'qotadi va dizayn elementi sifatida qabul qilina boshlanadi.

Ammo yirik harflar logotiplar, sarlafhalar, ko'rsatkichlar- bir so'z bilan aytganda dekorativ funktsiaysi informativlik funktsiyasidan kam bo'lmagan hamma elementlar tarkibida juda yaxshi ko'rinadi.

Mayda shriftlar o'zida havfni ham jamlaydi. Bizning ishda u o'quvchi qiynalmay o'qiy olishi uchun unchalik ko'p bo'lmasligi kerak. Bundan tashqari alohida matnli bloklar yetarli darajada mustaqil, yani bo'shliq bilan ajratilgan bo'lishlari kerak.

Rang

Umuman olganda bu bo'limda aytilgan rangli dizaynga bag'ishlangan barcha qoidalar matnlar uchun ham o'rinli. Faqat bir fikrni qo'shib qo'yish kerak: agar siz bitta so'zdagi bitta

harfni yoki bitta gapdagi bitta so’zni ajratib ko’rsatmoqchi bo’lsangiz, u holda yoki rangdan foydalaning, yoki garnituradan. Ularni birgalikda ishlatish hayratlanarli darajada yomon natijaga olib keladi (22-rasm).

PHOTOSHOP PHOTOSHOP

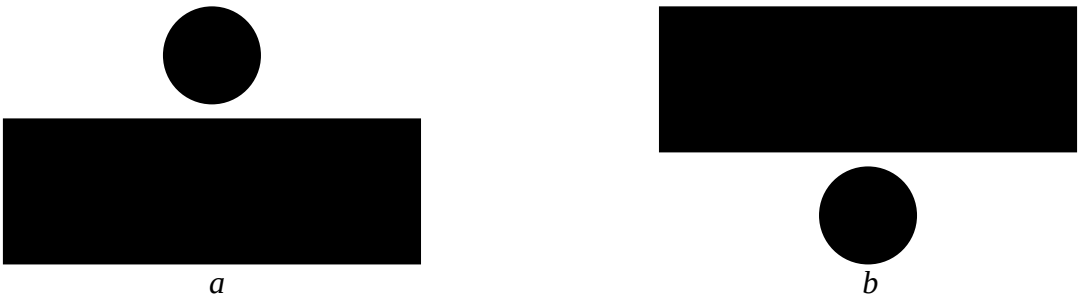
PHOTOSHOP

22-rasm. Fragmentni ikki marta ajrtib ko’rsatish (garnitura va rang bilan) – yomon usul

Qisqacha kompozitsiya to’g’risida.

Kompozitsiya to’g’risida suhbat boshlash xech bo’lmasa shuning uchun murakkabki, bu mavzu alohida e’tibor talab qiladi. Biz kompozitsiyalar mavzusini ochishga talabgor emasmiz-buning uchun, agar siz haqiqatda professional dizayer bo’lmoqchi bo’lsangiz, o’qishingiz mumkin bo’lgan maxsus kitoblar bor. Ammo agar siz kompozitsiyalar nazariyasi bilan umuman tanish bo’lmasangiz, bu bo’limni diqqat bilan o’qib chiqing.

Kompozitsiyalar nazariyasining asosida biz ko’p natsani ixtiyorsiz, instinkt va reflekslar darajasida qabul qilishimizni tushunish yotadi. Ob’ektlarning bir xilda joylashishida barcha kishilarda bir xil reaktsiya hosil bo’ladi. Masalan, 23-rasmdagi aylana bilan to’rtburchakning qanday joylashishida sizda havf, o’ng’aysizlik tuyg’usi hosil bo’ladi? Mualliflarning ishonchi komilki, bu *b* variantdir.



23-rasm. Kompozitsiyadagi ob’ektlarning turlicha joylashuvlari qarama-qarshi tuyg’ularni vujudga keltirishi mumkin: *a* - stabillik; *b* – o’ng’aysizlik

Shunday qilib har qanday kompozitsion ish asoslangan bo’lishi kerak. Dizaynda esa, ko’pchiligimizni hayratlantirib, nazariyani bilmasdan turib, kerakli natijaga deyarli erishish mumkin emas.

Kompozitsiyaning asosida ikki tushuncha yotadi: yaxlitlik va kontrast.

Yaxlitlik.

Bu tushuncha o’zida bitta maqsadga olib kelishi kerak bo’lgan ko’plab talablarni jamlaydi. Hatto aytish mumkinki, yaxlitlik – bu usul emas, balki dizayndagi maqsad, yakuniy natijada ish, u logotip yoki Web-sahifa bo’ladimi, yaxlit butun ko’rinishiga bo’lgan talabdir.

Amalda esa bu o’zni vositalar tomonida juda chegaralashni bildiradi. Sodda qilishdan qo’rqmang! Aksincha, sizning dizayningizdan soddaroq nimadir o’ylab topish qiyin bo’ladigan darajada soddalikka intiling. Agar shriftning bitta garniturasidan foydalanishga imkoniyat bo’lsa, bitta garnitura ishlating, butun dizaynni bitta figuraga keltirish imkoniyati bo’lsa, keltiring.

Yaxlitlik tamoyiliga ko’pincha boshlovchi web-ustalar amal qilishmaydi. Bizningcha, siz GIF-animatsiyalar, katta hajmli elementlar, ulkan sarlavhalar va juda yorqinranglarda yaratilgan ijod namunalarini ko’p bor ko’rgansiz. Ayniqsa o’larini professional dezayner hisoblovchi va o’z

xizmatlarini anchagina katta mablag’ evaziga taklif qiluvchi bu asarlarning mualliflari juda o’ng’aysiz ko’rinishadi. Ularning xatolarini takrorlamang, aqlsiz yoki uquvsiz ko’rinishdan qo’rqmang- oddiylikka intiling.

Kompozitsiyada yaxlitlikka erishishning eng muxim komponenti – bu muvozanatga erishish. Muvozanat – sizning ishingisni ko’rgan kishilarda “bu yerda nimadir yetishmayapti”, degan tuyg’u hosil bo’lmasligining garovi.

Kompozitsiya muvozanatiga nafaqat ob’ektlarning nisbatan bir tekis raqsimlanishi hisobigagina emas, balki ularning o’lchamlari, rangi, yorqinligi, formasi hisobiga ham erishiladi. Masalan, web-sahifaning bir tomonida sizda kattagina matnli blik bo’lsa, u holda ikkinchi tomondan bezak yoki qora fon hisobiga muvozanatga keltirish mumkin.

Muvozanat asosan ikki tipli bo’ladi: rasmiy va norasmiy.

Rasmiy muvozanat kompozitsiyaning optik markazi atrofidagi simmetriyani nazarda tutadi. Rasmiy muvozanatga ko’proq simmetriyaning hisobiga erishiladi.

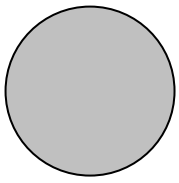
Rasmiy muvozanat – bu kompozitsiyadagi garmoniyaga erishishning eng oddiy usullaridan biridir. Lekin u simmetriyani nazarda tutadi, simmetriya esa, bu kitobda bir necha bor ta’kidlanganidek, zamonaviy dizaynda deyarli aqqli yechim deb hisoblanilmaydi. Har holda usiz yomon qilgandan ko’ra simmetriya bilan yaxshi qilgan ma’qulroq. Bu haqda xech bo’lmasa shu narsa bildiraki, klassik san’at, ayniqsa klassik arxitektura, o’z asosida rasmiy muvozanat tamoyillariga amal qilib keladi: oltin kesim proportsiyasi, formalarning qat’iyligi, absolyut simmetriya.

Rasmiy muvozanatni quyidagi sifatlar aniq ta’kidlashi mumkin:

- konservatizm;
- doimiylik;
- mustaxkamlik;
- munosiblik.

Ammo hozir ikkinchi tip- norasmiy muvozanat ko’proq ishlatiladi. U rasmiy muvozanat kabi aniq tiniq belgilarga ega emas, shuning uchun unga intuitsiya va ta’bga asoslangan holda erishish mumkin.

Norasmiy muvozanatga ko’plab usullar bilan erishish mumkin, lekin ularning mazmuni bitta anchagina sodda fikrga olib keladi: simmetriyadan tashqari hamma narsa mumkin. Yani agar siz web-sahifada yoki reklama varaqchasida yirik figurani joylashtirgan bo’lsangiz va u boshqalardan kuchli darajada ustunlik qilayotgan bo’lsa, boshqa figuralarning ranglarini o’zgartirish hisobiga uning yorqinligini kamaytirish mumkin. Va teskarisi, yirik matnli blok o’lchami bo’yicha ancha kichik, lekin butun figuralar bilan engil muvozanatga kirishadi (25-rasm).



Norasmiy muvozanatga ko’plab usullar bilan erishish mumkin, lekin ularning mazmuni bitta anchagina sodda fikrga olib keladi: simmetriyadan tashqari hamma narsa mumkin. Yani agar siz web-sahifada yoki reklama varaqchasida yirik figurani joylashtirgan bo’lsangiz va u boshqalardan kuchli darajada ustunlik qilayotgan bo’lsa, boshqa figuralarning ranglarini o’zgartirish hisobiga uning yorqinligini kamaytirish mumkin. Va teskarisi, yirik matnli blok o’lchami bo’yicha ancha kichik, lekin butun figuralar bilan engil muvozanatga kirishadi

25-rasm. Yirik matnli blok o’lchami bo’yicha ancha kichik, lekin butun figuralar bilan engil muvozanatga kirishadi.

Norasmiy muvozanatga erishishda eng murakkabi biz shu bobni boshlagan qoidamizni buzmaslik – hammasini oddiy qilish. Variantlarning ko’pligi shunga olib keladiki, biz u yerga ozgina qo’shamiz, bu yerga ozgina qo’shamiz, va natijada zamonaviy dizaynni o’rniga ob’ektlar uyulmasini olamiz.

Kompozitsiyaning yaxlitligi uchun barcha elementlar (iloji boricha) yagona *ritmga* rioya qilgan holda joylashsin. Ritm – bu kompozitsiyada ob’ektlar bilan bo’sh o’rinlarning yoki ob’ektlar bilan boshqa ob’ektlarning belgilangan tartibiga rioya qilishdir (26-rasm).

Rasm . 26. Logotip dizaynida ritmdan foydalanish

Aniq bir ritmga erishishga intilishda (axir u ishning umumiy “kayfiyatiga” kuchli ta’sir qiladi) eng oddiy echimdan – oshkora simmetriyadan qochishga harakat qiling.

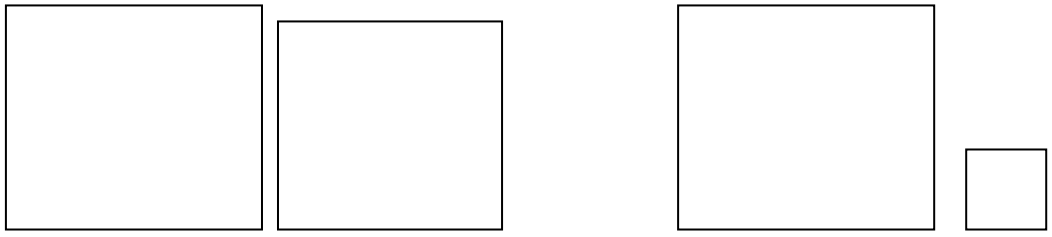
Kontrast.

Kontrast to’g’risida biz oldin gapirib o’tdik. Kontrast- bu zamonaviy dizaynning asosi. U yoki bu darajadagi professional ishlar yoki kontastga asoslanadi, yoki komponenta sifatida kontrastga ega bo’ladi.

Bir tomondan, kontrast- yaxlitlikning qarama-qarshisi. U bilan kompozitsiyadagi butunlikka erishilmaydi, aksincha, kontrastning maqsadi ob’ektlar o’rtasidagi farqlarni ta’kidlashdir. Lekin bu ikki tushunchalarni bir-biriga qarshi qo’ymaslik va ular o’zaro kelisholmaydi deb o’ylamaslik kerak. Aynan kontrast bilan yaxlitlik o’rtasidagi “oltin oraliq”qa intilish kerak.

Kontrast to’g’risidagi asosiy fikrlar oldinroq aytib o’tildi. Shuning uchun hozir biz oldinroq aytilganlarni kompozitsiyadagi kontastning har biri to’g’risida bir nechta so’z qo’shgan holda jamlaymiz.

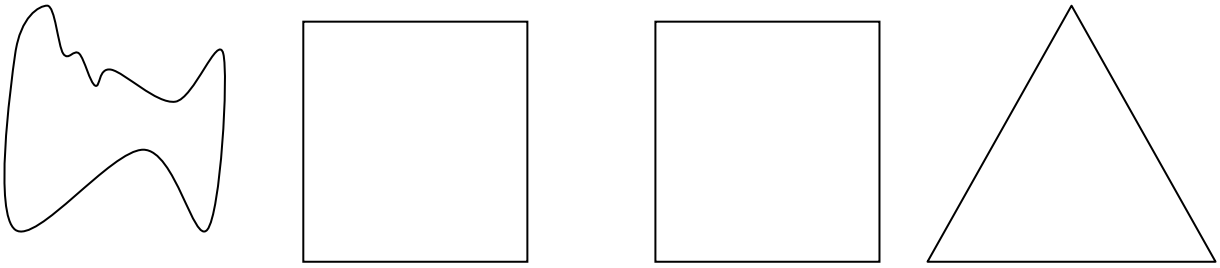
- *O’lcham.* Kontrastga erishishning eng yaxshi usullaridan biri – bir xil geometrik figuralar o’lchamlari o’rtasidagi farq. Lekin shuni esda tutish kerak: agar bu farq yetarli darajada katta bo’lmasa, tomoshabin uni dizaynerning ishi deb emas, balki xatolik deb qabul qiladi (27-rasm).



a *b*

Rasm. 27. Kontrast: a –figuralar orasidagi yetarlicha bo’lmagan farq xatolik singari qabul qilinadi; *b* – va faqat bu farqni orttirilishi holatni tuzatishi mumkin

- *Forma.* Formani yaxshisi kompositsiyani yaxlitligini ta’kidlash uchun foydalangan ma’qul. Masala shundaki, egri chiziqli va to’g’ri chiziqli figuralar o’zaro oshkora, qo’pol, naturalistik kontrastga egaki, faqaqt ko’pchilik bunda professional ishlarni bajara olmaydi. Agar figuralar bir-birlariga o’xshamasa (masalan, kvadrat va uchburchakdek), u holda ularning birlashtirilishi yakunlanmaganlik tuyg’usini beradi, tomoshabinlarni g’ashini keltiradi (28-rasm).



a *b*

Rasm . 2.28. Kontrast: a – egrichiziqli va to’g’richiziqli figuralar juda qo’pol kontrast yaratadi; b – bir tipdagi formalar orasidagi farq unchalik sezilarli ifodalanmagan

- *Rang.* Ranglardagi farqlar yordamida to’la kontrastga erishib bo’lmaydi. Lekin u to’ldiruvchi sifatida, “ikkinchi ovoz” uchun juda ajoyib variant.
- *Shrift.* Shriftlar o’z-o’zidan formalarning oshlora kontrastlarini namoyon qiladi. Shuning uchun garnituralar o’rtasidagi ortiqcha oshkora farqlarni qidirish dizayndagi tavtalogiya kabi rolni o’naydi. Ehtiyotkorroq bo’ling!

Butun kompozitsiya bo’yicha kontrast – bu hamma sanab o’tilgan kontrastlarning yig’indisi. Unga erishishda meyorga e’tibor bering: masalan, o’zining o’lchamlari tufayli shundoq ham yaxshi kontrast berayotgan elementni alohida rang bilan ajratmang.

Foydalanilgan adabiyotlar

1. Ш. Пауэрс Динамический HTML "Лори", 1999. 384 с.
2. Вайк А., Джиллиам Дж.JavaScript: Полное руководство: Перевод с английского."Вильямс", 2004. 719 с.
3. Лещев Д.Создание интерактивного web-сайта: Учебный курс. "Питер", 2003.
4. Дронов В. JavaScript в Web-дизайне. СПб-Петербург, 2002. 880 с.
5. Джейсон Кренфорд Тиге DHTML и CSS для Internet. НТ Пресс, 2005. 520 с.
6. Гаевский А.Ю., Романовский В.А. Самоучитель по созданию Web-страниц
7. Матросов, Сергеев, Чаунин. HTML 4.0 в подлиннике. BHV-СПб, 2000, 672 с.
8. Уилтон П. JAVASCRIPT. Основы. Символ-плюс. 2002. 1056 с.
9. Кингли-Хью Э., Кингли-Хью К. JAVASCRIPT 1.5: Учебный курс. Питер. 1-е издание. 2002.
10. Бранденбау. JAVASCRIPT. Сборник рецептов для профессионалов. СПб. 2001.
11. Рейнбоу В. Компьютерная графика СПб-Питер, 2003. 768 с.