14-лабораториялық жұмыс

JavaScript тілі. Объект = тәсілдер + қасиеттер

1. Объект түсінігі

Объект — бұл мәліметтер мен функциялар жиынынан тұратын бірыңғай конструкция немесе, JavaScript терминологиясында, қасиеттер мен тәсілдер жиыны.

Функция = тәсіл (метод).

Айнымалы = қасиет (свойства).

Инкапсуляция термині «қара жәшік» ретінде қарастырылатын объектінің ішкі құрылымын жасыру деген сөз. Объектінің қасиеттері белгілі болып саналады, яғни олар - сырттан қол жеткізуге болатын айнымалылар. Бірақ бұл функциялар қалай құрылған, олар қандай алгоритммен жұмыс істейді, ол туралы программалаушыға айтылмайды. Программалаушы немесе объектіні тұтынушы адам объектінің қосымша ішкі функция-лары мен айнымалылары бар ма, олар қол жеткізуге болатын қасиеттер мен тәсілдермен қалай байланысқан, ол жағын білмейді.

1.1. Объект және объектінің бір данасы (экземпляры)

Мысалы, нақты телевизор — бұл JavaScript терминологиясында объект емес, ол объектінің бір данасы (экземпляры). Объект болып зауытта нақты өнім шығаруға арналған құжаттамалар комплектісі саналады. Конвейерден шығып жатқан барлық телевизорлар бейнелерінің *қасиеттері* бірдей және оларды басқаратын *тәсілдер* де бірдей экземпляр болып табылады.

Программалауда да осылай. Объект — бұл шаблон, құжаттар жиыны. Объектінің бір данасы (экземпляры) — ол оның жұмыстық көшірмесі ғана.

1.2. Объект интерфейсі

және объектінің ішкі құрылымы

Rectangle объектісін қарастырайық. Ол мынадай информацияны сақтайды.

Rectangle объектісі

Объект тіктөртбұрыштармен жұмыс істеуге арналған. Объектінің бір данасын жасау үшін былай жазу керек:

```
var x = new Rectangle(a,b);    // Мұндағы x Rectangle
    // объектінің бір данасы (экземпляры).
   // a мен  b тіктөртбұрыш ені мен биіктігі.
   // new сөзі бір дана жасау үшін керек.
```

Объектінің бір данасы жасалған соң, келесі тәсілдер мен қасиеттерді пайдалауға болады:

Қасиеттер

width

height

Тәсілдер

square()

perimeter()

radius()

Пайдалану мысалы:

```
var p =x.perimeter();
var r = x.radius();
var m = x.width;
if(x.height > m)
  m = x.height;
```

Мұнда объект *интерфейсі* келтірілген, яғни объектімен қатынасуға қажет информация берілген.

Мұнда square, perimeter, radius функция-ларының программалық кодтары келтірілмеген. Басқаша айтқанда, объектінің ішкі құрылымы көрсетілмеген. Жұмыс кезінде тек интерфейсті пайдаланып, оның ішкі құрылымын қажет етпеуге болады.

Объект интерфейсі – бұлар пайдалануға болатын объектінің айнымалылары мен функциялары.

Объектінің ішкі құрылымы — программалау тілінде объектінің ішкі айнымалылары мен функцияларын сипаттау.

JavaScript тілінде *Rectangle* объектісінің **x** экземпляры-ның қасиеттері мен тәсілдері жазылады:

```
x.height;    x.perimeter();
```

Жалпы жазылу форматы мынадай болады:

экземпляр_аты.объект_қасиеті_не_тәсілі_

Нүкте - сатылы бөлу таңбасы: ол атасын баласынан (тегін мұрагерінен) бөліп тұрады.

1.3. Құрамдас ішкі объектілер және тұтынушы объектісі

JavaScript тілінде ішкі құрамдас объектілер көп. Оларды программалау қажет емес, олар тіл ішінде орнатылған. Бұл – браузердің программалық кодына осы объектілер коды кіреді деген сөз. Програм-малаушы осы объектілердің интерфейсін білуі тиіс, олардың бір экземплярын жасай білуі керек, сонда ол өз қалауынша ішкі объектілерді пайдалана алады.

JavaScript жаңа объектілерді программалауға және олардың ішкі объектілерін өзгертуге мүмкіндік береді. Енді бірнеше ішкі объектілер жұмысын қарастырайық.

2. *Date* объектісі

JavaScript тілінің ішкі объектілерін қарастыруды өте пайдалы болып саналатын **Date** объектісінен бастайық. Бұл объект күн-ай мерзімімен (датамен) және уақытпен жұмыс істеу үшін керек.

Объект экземплярын жасау үшін (Date объектісі-нің ғана емес, одан басқасының да) JavaScript тілінде **new** түйінді сөзі қолданылады:

var now = new Date();

Енді **now** айнымалысы **Date** объектісі экземпляры болып табылады да, ол үстіміздегі дата мен уақытты береді.

Жалпы экземпляр жасау былай орындалады:

var айнымалы = new Date(параметрлер);

Келесі параметрлерді көрсетуге болады:

Мысал

```
var now = new Date();
var birthday =
  new Date(1954,1,8);

var bell = new Date(2003,
     0,14,12,20,0);
```

Date объектісі экземпляры құрылғаннан кейін, оның ішкі мәліметтерін көруге болады, оны өзгерту мүмкіндігі де бар. Ол үшін көптеген тәсілдер бар, олардың тізімі тілге арналған кітап қосымшаларында келтіріледі.

Объект тәсілі аты (тәсіл – JavaScript терминологиясында функция) экземпляр атынан нүктемен бөлініп жазылып тұрады.

Былай жазуға болады:

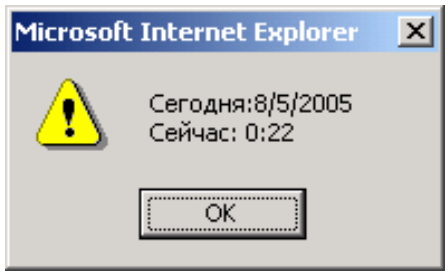
var year = bell.getYear();

year айнымалысы мәні 2003 болады.

Date объектісінің бірнеше қарапайым скриптерін қарастырайық.

Ағымдағы дата және уақыт

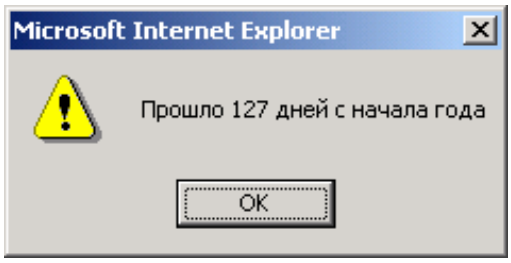
```
var now = new Date();
alert("Сегодня:"+now.getDate()+"/"+
(now.getMonth()+1)+"/"+
 now.getYear()+"\nСейчас:
 "+now.getHours() +":"
   +now.getMinutes());
```



Осы кодтарды жазған кезде скрип-ті орындау мынадай хабарлама шығуына себепші болады.

Жыл басынан бергі күндер саны

```
var now = new Date(); // Текущая дата и время.
var begin = new Date(now.getYear(),0,1);
    // Начало текущего года.
          // Число миллисекунд от начала года:
var num = now.getTime()-begin.getTime();
var msPerDay = 24 * 60 * 60 * 1000;
          // Число миллисекунд в сутках.
   num /= msPerDay;
          // Число дней с начала года.
          // Покажем результат:
alert("Прошло " + Math.floor(num) + " дней с начала года");
```



Осы кодтарды жазған кезде скрипті орындау мына-дай хабарлама шығуына себепші болады.

2.1. Date ішкі объектісі және оның тәсілдері

1. Тәсіл түрі – getYear(). Жыл нөмірінің соңғы екі цифрын береді.

Мысалы:

```
var d = new Date(2005,1,2);
var y = d.getYear();
alert(y);
```

2. Тәсіл түрі – setYear(). Жыл нөмірін тағайындайды.

Мысалы:
```
var d = new Date();
d.setYear(2004);
alert(d.getYear());
```

3. Тәсіл түрі – getMonth(). Ай нөмірі мәнін береді.
```
var d = new Date(2005,1,2);
var m = d.getMonth();
alert(m);
```

4. Тәсіл түрі – setMonth(). Айды енгізеді.
Мысал:
```
var d = new Date();
d.setMonth(4);
alert(d.getMonth());
```

5. Тәсіл түрі – getDate(). Ай күнін береді.
```
var d = new Date(2005,1,2);
var m = d.getDate();
alert(m);
```

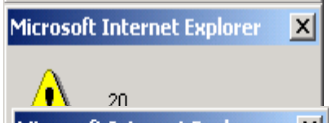
6. Тәсіл түрі – setDate(). Ай күнін енгізеді (тағайындайды).
Мысал:
```
var d = new Date();
d.setDate(2);
alert(d.getDate());
```

7. Тәсіл түрі – getDay(). Апта күнін береді (воскресенье – 0, понедельник – 1,
…, суббота - 6)
```
var d = new Date(2005,1,2);
var m = d.getDay();
alert(m);
```

8. Тәсіл түрі – getHours(). Уақыттың сағатын береді
```
var d = new Date();
var t = d.getHours();
alert(t);
```

9. Тәсіл түрі – setHours().
Сағатты енгізеді (тағайындайды).
Мысал:
```
var d = new Date();
d.setHours(22);
alert(d.getHours());
```

10.Тәсіл түрі – getMinutes(). Уақыт минутын береді.
```
var d = new Date();
var m = d.getMinutes();
alert(m);
```



11.Тәсіл түрі – setMinutes(). Уақыт минутын енгізеді.
```
var d = new Date();
d.setMinutes(32);
var t = d.getMinutes();
alert(t);
```

12.Тәсіл түрі – getTime().
1 янв 0 сағатынан 1970 ж. бергі миллисекунд мөлшерін береді.
Мысал:
```
var d = new Date();
```

```
var t = d.getTime();
alert(t);
```

3. *Array* объектісі

Бұл объект мәліметтер жиымын (массивін) жасау үшін керек. *Массив — элементтердің реттелген жиыны*. Жеке элементті көрсету оның аты мен индексін (нөмір) көрсету арқылы орында-лады. JavaScript тілінде элементтерді нөмірлеу нөлден басталады.

Мысал: апта күндерінің аттары жиымы.

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

Жиымның жеке элементтерін пайдалану үшін былай жазылады:

массив__аты [индекс]

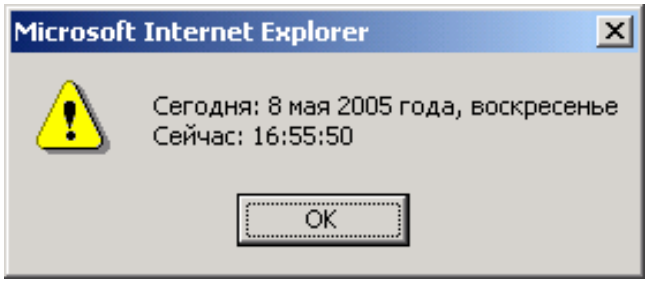
Төмендегі скрипт:

```
var dayNames = new Array("воскресенье", "поне-дельник", "вторник", "среда", "четверг", "пят-ница","суббота"); alert(dayNames[0]);
```

жұмысы нәтижесінде alert терезесіне «воскресенье» мәтіні шығады.

Мысалы: ағымдағы дата мен уақытты көрсету.

```
// Название месяцев в родительном падеже:
var monthNames = new Array ("января", "февраля",
   "марта","апреля","мая","июня","июля", "августа",
   "сентября","октября", "ноября", "декабря");
// Название дней недели:
var dayNames = new Array ("воскресенье","понедельник",
   "вторник","среда","четверг","пятница", "суббота");
var today = new Date(); // Текущая дата и время.
// Начало формирования строки вывода:
var str = "Сегодня: "
// Добавим день месяца:
str += today.getDate() + " ";
// Добавим название месяца:
str+=monthNames[today.getMonth()]+" ";
// Добавим год:
str += today.getYear() + " года, ";
// Добавим день недели:
str += dayNames[today.getDay()] + "\n";
// Добавим время:
str += "Сейчас: " + today.getHours() + ":" +
            today.getMinutes () +":"+
            today.getSeconds();
// Покажем результат:
alert(str);
```

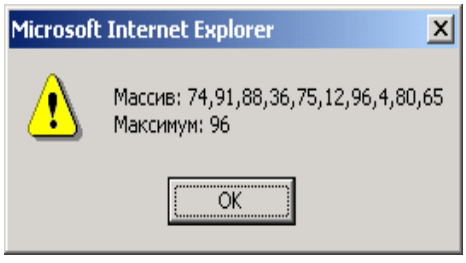


Жиым ұзындығы (оның элементтері саны) программа жұмысы кезінде өзгере алады:

```
var f = new Array(); //Сейчас массив пустой,
            //элементов в нем нет.
f[0] = 1;                        //В массиве один элемент.
f[l] = 1;                  //В массиве два элемента.
f[2] = f[0] + f[1];  //В массиве три элемента.
f[5] = 8;          //В массиве шесть элементов
            // f[0]...f[5]
```

Мысал: Жиымның максималь элементін анықтау.

```
// Создадим массив из num случайных чисел,
// каждое из которых входит в диапазон [a, b]
var num = 10; // число случайных чисел
var a = 1;    // левая граница интервала
var b = 100;  // правая граница интервала
var set = new Array(); // создан массив
// Заполнение массива случайными числами
for (i=0; i<num; i++)
  set[i] = Math.round(a+(b-a)*Math.random());
```



```
// Найдем максимальный элемент
var max = a;
for (i=0; i<num; i++)
  if(set[i] > max) max = set[i];
// Покажем массив и найденный максимум:
alert("Массив: "+set+"\nМаксимум: "+max);
```

Қарастырылған мысалда екі цикл бір циклге біріктірілген:


```
var max = a;
for(i = 0; i < num; i ++}
{
 set[i]-Math.floor(a+(b-a+1)*Math.random());
 if(set[i] > max) max = set[i];
}
```

Осыған дейін біз объектілер тәсілдерімен таныстық, бірақ — олардың қасиеттерін қарастырмадық. Объект қасиеттері деге-німіз — JavaScript терминологиясы бойынша объект тұ-тынушысына арналған интерфейстік айнымалылар болып табы-лады. Негізінде, мұнда объект туралы емес, тек объектінің нақ-ты экземпляры жайлы сөз болады. Қасиеттерді пайдалану үшін объект экземпляры аты мен нүкте арқылы бөлінген қасиет аты жазылады, мысалы:

```
var set = new Array("Горбунков","Семен",
      "Семенович");
alert (set.length);
```

length қасиеті массив элементтерінің санын — оның ұзындығын береді.

JavaScript тілінде объектілермен жұмыс істеу кезінде бірсыпыра «еркіндіктерге» жол беріледі. Мысалы, массив экземплярын new түйінді сөзісіз және Array объектісін де көрсетпей жазуға болады:

```
var set = [1, 4, 9, 16, 25, 36];
```

Әрине, браузер, мұндай жазбаны кездестіріп, бәрі бір Array объект экземплярын төмендегі жазба бергендей түрде ашады:

```
var set = new Array(1, 4, 9, 16, 25, 36);
```

Осыған дейін біз басқа тәсілдермен string объектісі экземплярын құрған болатынбыз. Мысалы, мынадай жазба:

```
var title = "Старик, упавший с каланчи";
```

төмендегі жазбамен бірдей болып табылады:

```
var title = new String("Старик, упавший с
                  каланчи");
```

JavaScript ішкі объектілерімен жұмыс істеу үшін анықтамалық материалдар болғаны дұрыс, олар көбінесе кітап қосымшаларында келтіріледі.

4. Объектіге бағытталған программалау

Объектіге бағытталған программалау (ООП) —програм-малық кодтар жасаудың қазіргі тәсілі, ол структуралық программалаудан кейін келді. ООП структуралық програм-малауды ауыстырған жоқ, ол оны ары қарай логикалық жетілдіру кезеңінен өткізді.

Структуралық программалау негізі — есепті шешудің сатылы бұтақ тәріздес құрылымын жасау және оның прог-раммалық кодын жекелеп жазып шығу болып табылады. Структуралық программалаушылар бір логикалық бірлікке қатысты процедуралар мен мәліметтерді жеке файлға құрылым (Си-де struct) арқылы жинақтай отырып жасап шығады.

ООП ортасында тек жаңа мәліметтерді құру ғана емес, оларды өңдеу функциялармен біріктіруге болады. JavaScript тілінде ол объект деп аталады.

Мысал.

```
// Конструктор объекта Rectangle.
function Rectangle(a, b)
{ // Сипатталуы свойств объекта (его данных).
  this.a = a;       // Ширина тіктөртбұрыша,
  this.b = b;       // Высота тіктөртбұрыша. }
//— Конец документации на объект Rectangle.
```

Бұл мысале **Rectangle** объектісі C++ тіліндегі құрылым сияқты сипатталған. Бұл объектіде мәлімет бар да, функцияның өзі жоқ. Сондықтан жаңа тіктөртбұрыш керек болғанда, оны былай анықтау керек болады:

```
var rec = new Rectangle(10,20);
// Создан экземпляр объекта rec.
```

Кейіннен rec айнымалысы мәнін өрнектерде пайдалануға болады:

```
var perimeter = 2 * (rec.a + rec.b);
```

// Периметр тіктөртбұрыша.
Объекта экземплярын функция аргументі ретінде де пайдалану-ға болады:
function Perimeter(x)
{ return 2 * (x.a + x.b); }
var p = Perimeter(rec);
Осы объектіге оның мәліметтерін өңдейтін функция (тәсіл) қосайық.
Мысалы.
// Объект Rectangle.
// Конструктор объекта.
function Rectangle(a,b)
{ // Сипатталуы свойств объекта (его данных)
this.a = a; // Ширина тіктөртбұрыша
this.b = b; // Высота тіктөртбұрыша
// Сипатталуы тәсілов объекта (его функций)
this.perimeter = _perimeter;
// Ссылка на функцию _perimeter
}
// Сипатталуы тәсіла perimeter.
function _perimeter()
{ return 2 * (this.a + this.b); }
//-- Конец документации объекта Rectangle
Енді Rectangle объектісі тек мәліметтерден емес, функциядан да тұрады. Мынадай код жазуға болады:
// Создадим первый тіктөртбұрыш:
var p1 = new Rectangle(10,20);
// Создадим второй тіктөртбұрыш:
var p2 = new Rectangle(35,70);
// Найдем сумму периметров:
var sum = p1.perimeter() + p2.perimeter();
perimeter тәсілін анықтайтын жолға назар аударыңдар:
This.perimeter = _perimeter;

Мұнда **perimeter** атты тәсіл анықталған және бұл тәсілді **_perimeter** атты функция жүзеге асырады. Тәсіл және функции аттары әр түрлі бола береді. Бірақ түсінбеушілік тудырмас үшін функция атын тәсіл атына төменгі сызықша «_» қою арқылы анықтау ұсынылады.

Объект түсінігі — қиын емес, бірақ оған үйрену керек. Тағы да бір рет Rectangle объектісіне бір мысал келтірейік. JavaScript тіліндегі объект *кәдімгі функция* сияқты , **function** *түйінді сөзімен* сипатталады**:**

//Объект Rectangle.
function Rectangle (a,b)
{ // Свойства.
this.width = a; // Ширина тіктөртбұрыша.
this.height = b; // Высота тіктөртбұрыша.
// тәсілы.
this.square= _square;//Площадь тіктөртбұрыша
this.perimeter = _perimeter;// Его периметр
this.radius=_radius;//Радиус опис-ной окр-ти
}
// Реализация тәсілов объекта.
function _square()
{ return this.width * this.height; **}**
function _perimeter()
{ return 2 * (this.width + this.height);**}**
function _radius()
{ var temp = this.width * this.width +
this.height * this.height;
return Math.sqrt(temp)/2; **}**
//Конец документации объекта Rectangle
Мынадай сипатталу:
function Rectangle (a,b)
{
…
}
объект *конструкторы* деп аталады, оның ішіндегі айнымалылар this сөзі арқылы жазылады:

this.width = a; // Ширина тіктөртбұрыша. **this.height = b;** // Высота тіктөртбұрыша.

this түйінді сөзі конструктор арқылы жасалатын объект экземплярының көрсеткіші болып табылады. Яғни ол айнымалылар мен функцияларға болашақ объект экземплярлары қасиеттері мен тәсілдері ретінде мағына береді.

Шартты түрде **this** түйінді сөзі конструктор сипатталу-ындағы айнымалыны қасиетке, ал объект тәсіліндегі сілтемені функцияға «айналдырады».

Кәдімгі функцияны объект конструкторынан айырудың оңай тәсілі: «егер **function {...}** ішінде **var** сөзі орнына **this** пайдаланылса — ол объект конструкторы болғаны».

```
var r1 = new Rectangle(3,4);
// Первый экземпляр тіктөртбұрыша.
var r2 = new Rectangle(10,20);
// Второй экземпляр тіктөртбұрыша.
```

Осы екі команда және объект «документациясы» арқылы браузер екі тіктөртбұрыш экземплярын (rl және r2) жасайды. Әрбір экземплярдың өз айнымалылары: **width** және **height** болады. Оларды былай пайдалануға болады:

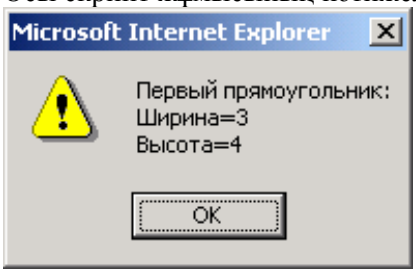
```
var x = rl.width; // x получает значение 3.
var y = r2.height; // y получает значение 20.
```

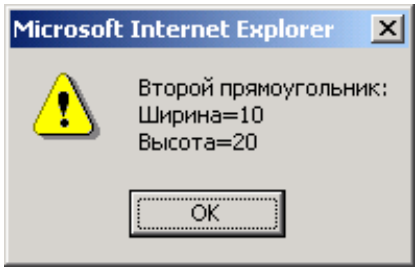
Келесі слайдта браузерде тексеруге болатын мысалдың толық мәтіні келтірілген.

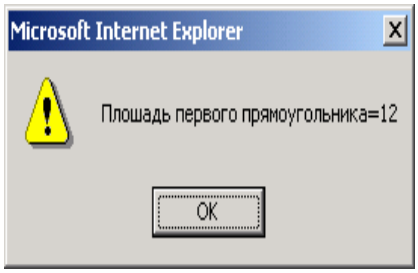
```
<HTML>
 <HEAD>
  <TITLE>Проверка объекта</TITLE>
 </HEAD>
 <BODY bgcolor=white text=black link=blue
      alink=red vlink=purple>
 <H1>Проверка объекта</H1> <HR>
 <SCRIPT language=JavaScript>
 <!--
 // -- Документация объекта Rectangle.--
 // Конструктор объекта.
 function                                                    Rectangle(a,b}
{
  // Свойства.
  this.width = a; // Ширина тіктөртбұрыша.
  this.height = b; // Высота тіктөртбұрыша.
  // тәсілы.
  this.square =_square;//Площадь тіктөртбұрыша
  this.perimeter =_perimeter; //Его периметр
  this.radius = _radius; //Радиус опис-ной окр.
 }
// Реализация тәсілов объекта
function _square()
{ return this.width * this.height;    }
function _perimeter()
{ return 2*(this.width + this.height); }
 function _radius()
          { var temp = this.width*this.width +
       this.height*this.height;
 return Math.sqrt(temp)/2;          }
  //Конец документации объекта Rectangle.
  // Мысалы работы с построенным объектом.
var rl = new Rectangle(3,4);
alert("1-ый тіктөртбұрыш:\nШирина="+
     r1.width+"\nВысота=" + rl.height);
var r2 = new Rectangle(10,20);
alert("2-ой тіктөртбұрыш:\nШирина="+
     r2.width+"\nВысота=" + r2.height);
alert("Площадь 1-тіктөртбұрыша="+r1.square());
alert("Периметр 2-тіктөртбұрыша="+r2.perimeter());
alert("Радиус для 1-тіктөртбұрыша=" +r1.radius());
//-->
 </SCRIPT>
 </BODY>
```

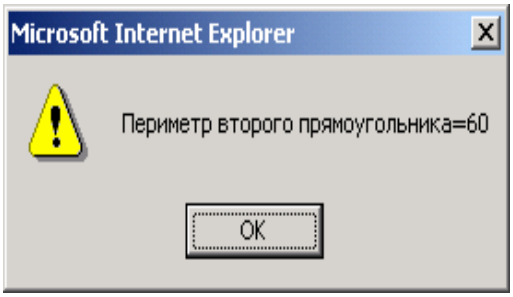
</HTML>

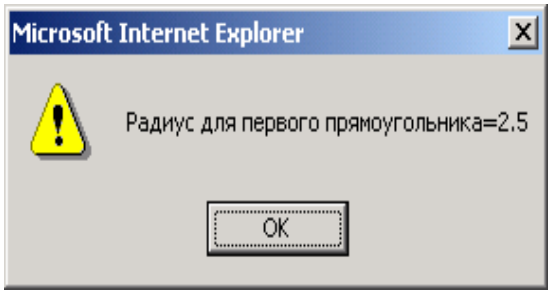
Осы скрипт жұмысының нәтижелері:







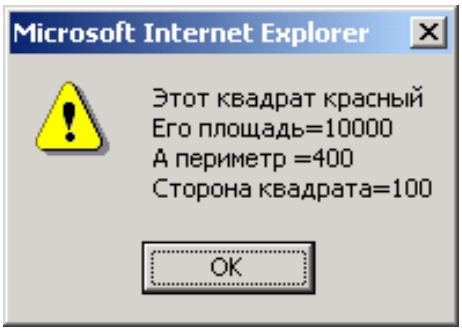




5. Мұралау

Тіктөртбұрыш негізінде жаңа объект Kvadrat. Для жасайық. Тіктөртбұрыштың балық қасиеттері мен тәсілдері мұра ретінде квадратқа өтеді, айырмасы болуы үшін түс элементін қосайық:

```
//Описание конструктора(a - сторона, c - цвет)
function Kvadrat(a,c)
{
 this.parent = Rectangle;
        //Родительский объект.
 this.parent(a,a);
        //Вызвали его конструктор.
 this.color = c;
        // Определили қасиет "цвет"
}
```



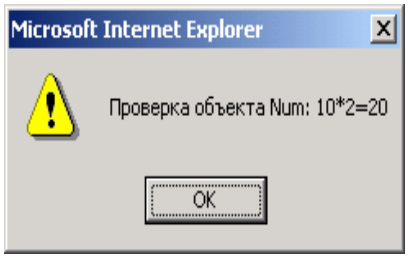
Енді былай жазуға болады:

```
var kv = new
    Kvadrat(100,"красный"); //Квадрат
var s = kv.square();//Нашли его площадь
var p = kv.perimeter();//Нашли периметр
alert("Этот квадрат " + kv.color + "\nЕго площадь=" + s + "\nА периметр =" + p + "\nСторона квадрата="
+ kv.width);
```

Осы сипатталған механизм ООП ортасында *мұралау* (наследование) деп аталады.

6. Статикалық және динамикалық мұралау

Мынадай объект жасайық:




```
//-- Объект Num --
function Num(a)
{ this.number = a; this.mul2 = _mul2; }
function _mul2()
{ return this.number * 2; }
//-- Конец документации на объект Num. --
```

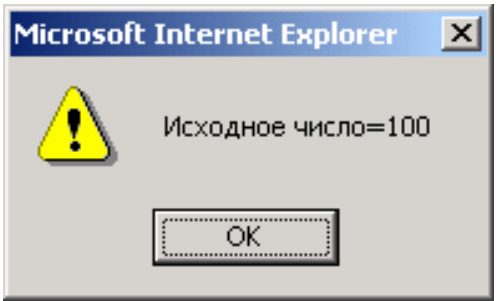
Объект санды сақтап, оны 2-ге көбейте алады.

Енді былай жазамыз:

```
var x = new Num(10);
var y = x.mul2();
alert("Проверка объекта
Num: 10*2="+y);
```

Енді Num объектісі үшін мұрагер жасаймыз:

```
 // Объект Numa (наследован от Num)
function Numa(a)
{ this.parent = Num; // родителем объявлен Num
  this.parent(a); // вызван конструктор родителя
```

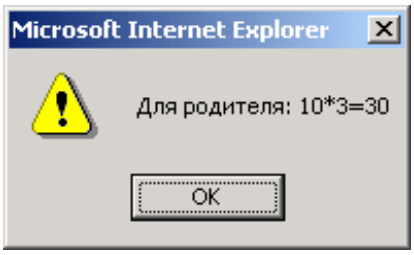


```
  this.put = _put;// объявлен новый тәсіл у потомка }
function _put()
{  alert("Исходное число=" + this.number);}
// Конец документации на объект Numa.
```

Мұрагерді тексеру үшін былай жазамыз:

```
 // Проверка Num
var z = new Numa(100);
z.put();
```

Экранда мынадай жазу пайда болады:



«Исходное число=100»

Num объектісіне жаңа тәсіл қосып былай жазамыз:

```
 Num.prototype.mul3 = _mul3;
function _mul3()
{  return this.number*3;}
```

Тексереміз:

```
var t = new Num(10);
alert("Для родителя: 10*3=" +t.mul3());
```

Мұрагердің өз тегінің жаңа тәсілін қалай түсінетінін тексерейік, ол үшін былай жазайық:

```
var k = new Numa(10);
alert(k.mul3());
```

Браузер қате жайлы хабарлама береді.

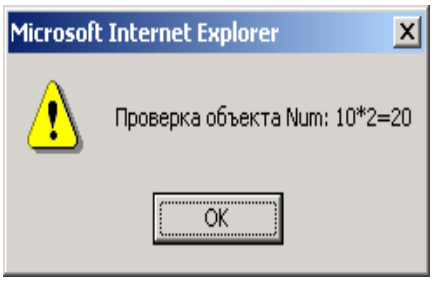
Мұндай мұралау *статикалық* деп аталады. Мұрагер жасалу кезінде оның ішкі құрамына тегінің барлық қасиеттері мен тәсілдері көшіріледі де, содан кейін туыстық байланыс жойылады. Егер оның ата тегі жаңа тәсілге ие болатын болса, оның мұрагері ол туралы ешнәрсе білмейді.

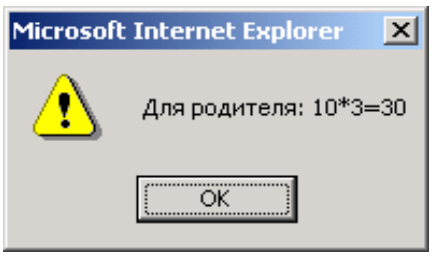
Мұрагер жасау кезінде оның ата тегімен байланысын үзбеуге болады. Ол *динамикалық* мұралау арқылы іске асырылады. Егер объект мұралау тәсілімен жасалса, онда ата тегінің құжаты (документациясы) көшірілмейді, тек оған *сілтеме* жасалады. Егер ата тегінде бір өзгеріс болса, онда ол барлық «динамикалық» түрде құрылған мұрагерлерге ол қатысты болып саналады.

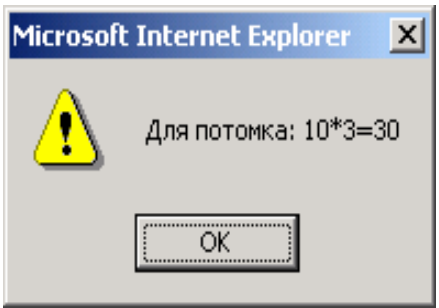
Динамикалық мұралауы бар объектіге мысал:

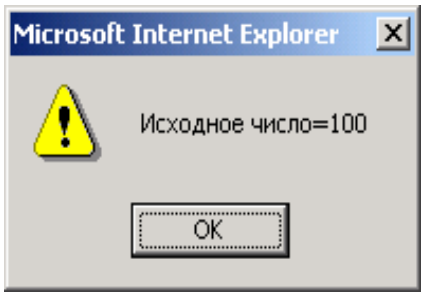
```
//Объект Numa (динамически наследован от Num)
function Numa(a)
{  this.parent = Num;
   // родителем объявлен Num
   this.parent(a);
   // вызван конструктор родителя
   this.put = _put;
   // объявлен новый тәсіл у потомка
}
Numa.prototype = new Num;
  //Задана динамическая связь с родителем
function _put()
{
alert("Исходное число="+this.number);
}
//-- Конец документации на объект Numa.--
```

Динамикалық мұралауы бар объект мысалдары нәтижелері:









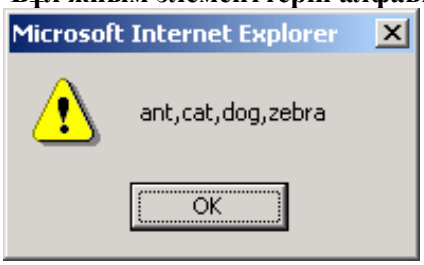
7. *Array* объектісінің *sort* тәсілі

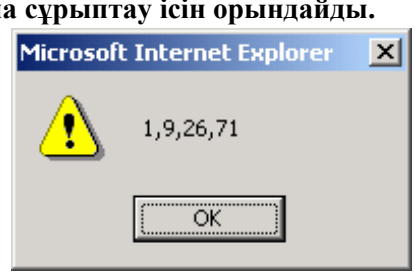
Array объектісінің sort тәсілін қарастырайық.

Ол үшін sort(function) немесе sort() тәсілі енгізілген, мұнда function параметрі екі элементті салыстыру ережесін береді, ол болмаған жағдайда сұрыптау лексографикалық тәртіпте жүргізіледі. Мысалы:

```
var set = new
    Array("zebra","ant","dog","cat");  set.sort();                    alert(set);
```

Бұл жиым элементтерін алфавит бойынша сұрыптау ісін орындайды.





Бұл sort(function) тәсілінде function функциясы-ның екі аргументі болуы тиіс, ол мынадай мәндер қайтарады:

— теріс сан, реттелуі бойынша бірінші аргумент екіншісінен сол жақта орналасқанда;

— 0, аргументтері тең мәнді болғанда;

— оң сан, реттелуі бойынша бірінші аргумент екіншісінен оң жақта орналасқанда. Мысалы:

```
var set = new Array(26,71,9,1);
function Compare(a,b)
{ return a - b; }
set.sort(Compare);
alert(set);
```

2.1. Array құрамдас объектісі және оның тәсілдері

1. Тәсіл түрі – concat(array). Бұрынғы массивке array массиві қосылған жаңа массив жасап береді. Бастапқы массив өзгермейді.

Мысал:

```
var set1 = new Array(1,2);
var set2 = new Array(3,4);
var set = set1.concat(set2);
alert(set1+"\n"+set2+"\n"+set);
```

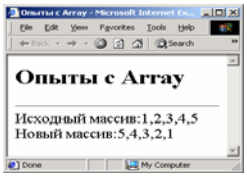
2.Тәсіл түрі – join(разделитель). Разделитель арқылы бөлініп орналасқан массив элементтері жолын береді. Бастапқы массив өзгермейді.

Мысал:

```
var set= new Array(1,2,3,4,5);
var set2 = set.join("+");
alert(set1+"\n"+set2);
```

3.Тәсіл түрі – reverse(). Массив элементтерін оның 1-элементі соңғысы болатындай етіп кері бағытта орын ауыстырып береді.

```
var set= new Array(1,2,3,4,5);
alert("исходный массив:"+set);
var y = set.reverse();
alert(" Новый массив:"+y);
```

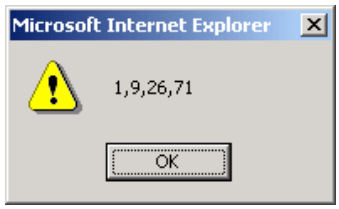


4.Тәсіл түрі – slice(ind1,ind2)не slice(ind1) Бастапқы массивтен позициясы ind1 -ден бастап ind2-1 позициясына дейінгі элементтерден тұратын жаңа массив жасап береді. Егер 2-индекс жоқ болса, онда массив соңына дейінгі элементтер алынады.

```
var set= new Array(0,1,2,3);
var set1= set.slice(1,3);
var set2= set.slice(1);
```

alert(”set=“+set+”\nset1=”+
set1+”\nset2=”+set2);

5.Тәсіл түрі – sort(function)не sort(). Массивті сұрыптайды. function параметрі екі элементті салыстыру ережесін береді, ол жоқ болса, сұрыптау лексографикалық тәртіппен жүргізіледі. Мысалы:
var set = new Array("zebra","ant","dog","cat"); set.sort(); alert(set);
sort(function) тәсілінде function функциясының екі аргументі болуы тиіс, оның қайтаратын мәндері:
теріс сан, реттелуі бойынша бірінші аргумент екіншісінен сол жақта орналасқанда;
— 0, аргументтері тең мәнді болғанда;
— оң сан, реттелуі бойынша бірінші аргумент екіншісінен оң жақта орналасқанда. Мысалы:
var set = new Array(26,71,9,1);
function Compare(a,b)
{ return a-b; }
set.sort(Compare);
alert(set);



6. length тәсілі қасиеттері – массив ұзындығы (оның элементтері саны). Мысалы:
var set = new Array(0,1,2,3,4,5,6,7,8,9,10); alert(set.length);