**OBYEKT ANLAYIŞI**

Obyekt, mürəkkəb verilən tipidir. Obyektin tərkibinə dəyişən çoxluğu - xassə və bu dəyişənlər üzərində manipulyasiya etmək üçün funksiya çoxluğu - metod daxildir. Xassə verilən saxlayır, metod isə onları emal edir. Bu şəkildə obyektə proqramın digər fraqmentlərindən asılı olmayaraq, ayrıca baxmaq olar.  
  
Obyektə nümunə kimi, HTML sənədə baxaq. O, xassələr çoxluğuna malikdir: internet-ünvan, ölçü, simvol kodlaşdırılması və s. Bəzi xassələrini dəyişə bilərsiniz, bəzilərini isə yox. O, həmçinin metodlar çoxluğuna da malikdir: Web browser pəncərəsində göstərmək, disk üzərində saxlamaq, çap etmək və s. Belə bir analogiya vermək də olar: xassə - obyektin atributu, metod - idarəetmə vasitəsi.  
  
Obyekt digər verilənlərin tiplərindən əlavə olaraq bir də onunla fərqlənir ki, onun istifadəsi üçün uyğun obyekt sinfinin nüsxəsini yaratmaq lazımdır. Bu zaman sinif dəyişən tiplərinə oxşar olaraq obyekt tipinin nümunəsidir, nüsxə isə onunla işləyə biləcəyiniz konkret obyektdir. Nüsxə obyekt "new" operatoru vasitəsilə yaradılır. Bu zaman obyektə istinad dəyişənə mənimsədilir. İstinad (və ya göstərici) yaddaşda bir yerə nişanlanır, hansı ki, yaradılan nüsxə obyekt saxlanılır. Dəyişəndə saxlanılan bu göstərici ilə obyektə müraciət edə bilərik.

var obj;

obj = new SomeObject();

Burada obj dəyişəndir, hansı ki, yaradılmış obyektə istinad edən göstərici mənimsədilib, SomeObject isə, obyekti yaradan sinifdir.  
Obyekti isifadə edib qurtardıqdan sonra, onu silmək olar. Bunun üçün "delete" operatorundan istifadə edilir.

delete obj;

**Qeyd:** "delete" operatoru yalnız Internet Explorer 6.0 və Navigator 4.0 versiyalarından başlayaraq dəstəklənir. Internet Explorer 5 və Navigator 3.0 versiyalarında dəyişənə null qiymətini mənimsədərək, obyekti silə bilərsiniz. Navigatorun çox köhnə versiyaları obyekti silməyi ümumiyyətlə dəstəkləmir.  
  
Yaxşı, biz obyekt yaratdıq və onun xassə və metodlarına müraciət etmək istəyirik. Bunu necə etməliyik? Çox sadə. Aşağıdakı çağırma formatından istifadə edərək, xassələrə müraciət edə bilərsiniz:  
**{Obyektə istinad edən dəyişənin adı}.{Xassənin adı}**  
  
Bu şəkildə xassəyə müraciət etmək dəyişənə müraciət etmək kimidir, fərq yalnız ondan ibarətdir ki, burada dəyişənin - xassənin əvvəlinə obyekti göstərən dəyişənin adını və nöqtə işarəsini qoymaq lazımdır.

obj.SomeProperty = 9;

encoding = HTMLDocument.CodePage;

Metod da oxşar şəkildə çağırılır:  
**{Obyektə istinad edən dəyişənin adı}.{Metodun adı}([{Parametrlər}])**  
  
Yəni demək olar ki, adi funksiya kimi çağırılır, istisna olaraq, obyektin adını (obyektin adı - obyektə istinad edən dəyişənin adı mənasındadır) yazmaq lazımdır.

void HTMLDocument.SaveToDisk('somefile.htm');

Obyekt massivin elementinə mənimsədilə və funksiyaya göndərilə bilər. Bu zaman funksiyanın gövdəsində onun xassələrini dəyişməyə və metodunu çağırmağa icazə verilir.  
Obyekt həmçinin obyekt tipində xassəyə malik ola bilər, yəni xassə daxili obyekt adlandırılan obyektə göstərici saxlaya bilər. Analoji qaydada daxili obyektin xassə və metodlarına müraciətə icazə verilir:

encoding = someObject.internalObject1.property1;

v = someObject.internalObject1.method1;

Koddan göründüyü kimi, əvvəlcə obyektin adı, sonra daxili obyektin adı, daha sonra isə, daxili obyektin xassəsinin və ya metodunun adı göstərilib.  
Yeri gəlmişkən, obyektin özü bu şəkildə yaradılır:

obj = new SomeObject();

O, funksiya çağırılmışına çox oxşayır, elə deyilmi? Bu da öz növbəsində funksiyadır, konstruktor adlanan obyektin xüsusi metodudur. Bu metod digərlərindən onunla fərqlənir ki, bu metod obyekt yaradılarkən çağırılır və onun xassələrinin ilkin qiymətlərini verir. Konstruktorun adı həmişə sinifin (class) adı ilə eyni olur.  
Konstruktor parametr ala bilər.

obj2 = new SomeObject2(param1, param2, x + y, 3);

Konstruktorsuz yaradılan obyektlər də mövcuddur. Onlar inisiallaşdırıcı vasitəsilə yaradılır:

obj3 = new SomeObject3(prop1: param1, prop2: param2, prop3: x + y, prop4: 3);

Yəni bu cür obyekt yaradarkən, biz onun xassələrini qiymətlərlə doldururuq.  
  
**Qeyd:** İnisiallaşdırıcı Internet Explorer və Navigator browserlərində 3.0 versiyalarından başlayaraq dəstəklənir. Navigatorun (JavaScript-i Internet Explorer 3.0 versiyasından başlayaraq dəstəkləyir) daha əvvəlki versiyalarında, siz obyekti yalnız konstruktor vasitəsilə yarada bilərsiniz.  
  
JavaScript-də olan bütün mümkün obyektlər daxili və istifadəçi yaradan olmaqla iki qrupa bölünür. Daxili siniflərə misal olaraq Array, Boolean, Date, Function, Math və s. misal göstərmək olar. Daxili siniflər burada araşdırılmayacaq.  
typeof operatoru obyekt dəyişəni üçün "object" sətirini qaytarır.

### BİR NEÇƏ YENİ OPERATOR

Obyektlərlə işləyərkən tətbiq olunan bir neçə yeni operatora baxmaq zamanı gəldi.  
**in** operatoru verilmiş obyektdə verilmiş xassənin olub-olmadığını yoxlayır və true və ya false qiymətini qaytarır.  
**{xassənin adı} in {obyektin adı}**  
  
Burada xassənin adı sətir şəklində verilir, obyektin adı isə, obyekt tipli dəyişən kimi verilir.  
Bu operatoru məsələn, aşağıdakı kimi istifadə etmək olar:

if ("SomeProperty" in obj) ...

**Qeyd:** in operatorunu yalnız Internet Explorer dəstəkləyir.  
  
**instanceof** operatoru verilmiş obyektin verilmiş sinifin nüsxəsi olub-olmadığını yoxlayır, true və ya false qiymətini qaytarır.  
**{obyekt} instanceof {sinif}**  
  
Burada obyekt obyekt tipli dəyişən şəklində verilir, sinif isə sinifin adı şəklində verilir.

if (obj instanceof SomeObject) ...

**Qeyd:** **instanceof** operatoru Internet Explorer-in 5.5-ci versiyasından başlayaraq dəstəklənir.  
  
**for-in** operatoru obyektin bütün xassələrinə və ya massivin bütün elementlərinə baxmağa və hər biri üçün əməliyyat aparmağa imkan verir. Bu operator xüsusiləşdirilmiş dövr operatorudur.

for({sayğac} in {obyekt və ya massiv})

... dövrün gövdəsi

sayğacı obyektin xassəsinə və ya massivin elementinə müraciət etmək üçün istifadə etmək olar.

for(k in obj)

{

s += k + ‘ ’;

st += obj[k] + ‘ ’;

}

Bu proqram fraqmenti obj obyektinin bütün xassələrinə baxır və xassələrin adlarını boşluq işarəsi ilə ayıraraq, s dəyişəninə yazır, st dəyişəninə isə, xassələrin qiymətlərinin yenə də boşluq işarəsi ayıraraq saxlayır.  
  
**Qeyd:** **for-in** operatoru Internet Explorer-in 5.5-ci versiyasından başlayaraq dəstəklənir.  
  
**with** operatoru, əgər proqramda hər hansı obyektin (qeyd: bir obyektin) xassə və metodları istifadə olunursa, JavaScript ifadələrinin uzunluğunu xeyli qısaltmağa imkan verir. Misala baxaq.

someobject.property1 = 1;

someobject.property2 = 2;

someobject.property3 = 3;

someobject.method1;

Bu, bizim **with** operatoru istifadə olunmayan proqram fraqmentimizdir. Baxın ifadələr necə uzun sətirdir. İndi isə, **with** operatorunu istifadə edək.

with(someobject)

{

property1;

property2;

property3;

method1;

}

Bu proqram fraqmenti isə, olduqca qısadır.

### YARADILMA

Bəzən JavaScript-in tərkibində olan Class-ların (siniflərin) imkanları məsələlərin həlli üçün kifayət etmir. Bu zaman istifadəçi öz class-larını yaradır.  
İstifadəçi class-ını yaratmaq üçün nə etmək lazımdır? O qədər çox nə isə tələb olunmur. Əgər class konstruktora malik deyilsə, onda onun nüsxəsini sadəcə inisiallaşdırıcı istifadə etməklə **new** operatoru ilə yaratmaq lazımdır.

var somePoint;

somePoint = new SomePoint(x: 100, y: 50, color: "black");

Hamısı bu qədər. Bu şəkildə biz bir qrafiki nöqtəni göstərən, koordinat və rəng xassələri olan obyekt yaratdıq. Artıq indi biz onu istifadə edə bilərik.

somePoint.y = somePoint.x / 2 + 20;

Hətta yaradılmış bu obyektə xassə əlavə etmək mümkündür.

somePoint.radius = 0.5;

Lakin biz öz sinifimizi yaratmadıq. Öz sinifimizi həqiqətən yaratmaq üçün konstruktor təyin etmək lazımdır.  
Əvvəlcədən bildiyimiz kimi konstruktor sinif üçün xassə təyin edir və onlara susmaya görə (default) qiymət verir. (Susmaya görə qiymət proqram kodunda sərt olaraq təyin oluna bilər və ya konstruktora parametr kimi verilə bilər.) Həmçinin konstruktor sinfə metod təyin edir (əlavə edir).  
  
Gəlin, qrafiki nöqtə sinfini yaradaq. Bunun üçün aşağıdakı konstruktoru yazaq.

function Point(x, y, color)

{

this.x = x;

this.y = y;

this.color = color;

}

Biz cari obyektə istinad etmək (müraciət etmək) üçün yeni **this** operatorundan istifadə etdik. Bu operatoru yaddan çıxarmayın, o bizə çox lazım olacaq.  
  
İndi biz Point sinfindən obyekt yarada bilərik.

var somePoint;

somePoint = new Point(100, 50, "black");

Gəlin, yenicə yaradılmış Point sinfinə metod əlavə edək. Yadda saxlayın ki, dəyişən funksiyaya göstərici saxlaya bilər. Bu dəyişən sinfin xassəsi də ola bilər.  
Əvvəlcə, funksiya təyin edək. Bu metod koordinatların müəyyən aralıqda olub-olmadığını yoxlayır və buna uyğun olaraq, true və ya false cavabını qaytarır.

function isPointInBounds()

{

if(this.x>0 && this.x<400)

{

return (this.y>0 && this.y<200);

}

else

{

return false;

}

}

Yuxarıda elan etdiyimiz Point sinfini yenidən təyin edək.

function Point(x, y, color)

{

this.x = x;

this.y = y;

this.color = color;

this.isPointInBounds = isPointInBounds;

}

Sonuncu sətirdə biz isPointInBounds xassəsinə isPointInBounds funksiyasına olan göstərici mənimsədirik. İndi biz bu xassəni isPointInBounds funksiyasını çağırmaq üçün istifadə edə bilərik. Faktiki olaraq, bu funksiya obyektin bir hissəsi olur, yəni metodu.

if(somePoint.isPointInBounds())

{

...

}

Əgər istifadəçi xassələrə mənimsətmək üçün lazım olan parametrləri konstruktora verməyibsə, bu zaman, xassə üçün susmaya görə olan qiymətdən istifadə etmək olar. Bunu etmək üçün || (məntiqi "və ya") operatorunun xüsusi formasından istifadə olunur.

function Point(x, y, color)

{

this.x = x || 100;

this.y = y || 50;

this.color = color || “black”;

this.isPointInBounds = isPointInBounds;

}

Əgər || operatorundan sol tərəfdə olan arqument təyin olunubsa, o qaytarılır. Əks halda, || operatorundan sağ tərəfdə olan arqument qaytarılır, bizim halda susmaya görə qiymət. Buradan belə çıxır ki, biz obyekti aşağıdakı şəkildə yarada bilərik:

somePoint = new Point();

İndi bir az əvvələ qayıdaq. İnisiallaşdırıcı ilə yaradılmış obyektə yeni xassə əlavə etdiyimizi yada salın. Biz eyni hərəkəti konstruktor ilə yaradılmış somePoint obyekti üçün də edə bilərik.

somePoint.radius = 0.5;

Amma burada qeyd etmək lazımdır ki, əlavə etdiyimiz xassə yalnız somePoint obyektinə məxsus olur, digər Point sinfindən olan obyektlərdə bu xassə olmur. Bəs bu xassənin Point sinfində olan bütün obyektlərdə olmasını necə təmin etmək olar?

### PROTOTİP İSTİFADƏ ETMƏK

Çıxış var - obyektin prototipini istifadə edin və xassə əlavə edin.  
Prototip - bu bir növ obyektin sinfinə istinaddır. JavaScript-də olan bütün siniflərin (hətta JavaScript-in daxili siniflərinin) prototype adlı xassəsi var və onun prototipini göstərir. Prototipi sinfi genişləndirmək üçün istifadə edirik.  
  
Obyektə xassə əlavə etmək üçün aşağıdakı kodun əvəzinə

somePoint.radius = 0.5;

yeni xassəni onun prototipinə daxil edək (istənilən sinfə məxsus olan prototype xassəsini istifadə edərək):

Point.prototype.radius = 0.5;

İndi isə, Point sinfindən olan bütün obyektlərin radius xassəsi var. Lakin ancaq somePoint obyektinin radius xassəsi 0.5-ə mənimsədilib, digər obyektlərdə isə, bu qiymət təyin olunmayıb.  
Biz həmçinin daxili sinifləri də dəyişə bilərik, xassə və metod əlavə edə bilərik. Məsələn, Math sinfinə lazımi riyazi funksiyalar və konstantlar əlavə edə bilərik.  
  
Lakin prototipi digər məqsədlər üçün də istifadə etmək olar. Məsələn, biz sinfi irsən ala bilərik. Misal üzərində izah edək.  
Tutaq ki, biz Point sinfinə bir neçə yeni xassə əlavə etmək istəyirik. Bunu hansı şəkildə edək? Əlavə olunacaq bütün yeni xassələri həmin sinifdən olan hər hansı obyektə növbə ilə əlavə etmək olar, lakin bu, çox da yaxşı deyil. Bunu başqa cür edək. Yeni SuperPoint nöqtə sinfi yazaq və bu sinifdə köhnə Point sinfinin bütün xassə və metodlarını irsən alaq. SuperPoint sinfi Point sinfinin varisi olacaq və valideyninin (və ya əcdadının) bütün xassə və metodlarını istifadə edə biləcək. Bu şəkildə biz iki nöqtə sinfinə malik olacağıq: sadə və təkmilləşdirilmiş.  
  
Əvvəlcə SuperPoint sinfini və onun bütün yeni xassələrini təyin edək. Və bizim koda sətir-sətir baxaq.

function SuperPoint(x, y, color, radius, edgeColor)

{

Bura qədər hər şey aydındır. Yeni sinif üçün konstruktor funksiyası təyin etdik. Qeyd edək ki, konstruktorun parametr siyahısında həm yeni xassələri elan etdik, həm də irsən valideyn sinifdən alınan xassələri elan etdik.

this.base = Point;

this.base(x, y, color);

Biz SuperPoint sinfinin yeni base xassəsini təyin etdik və ona Point sinfinin konstruktoruna göstərici mənimsətdik. Bundan sonra, valideyn sinfin konstruktorunu tələb olunan parametrləri verərək çağırdıq. Bu çağırmanın nəticəsində valideyn sinifdən irsən alınan bütün xassələr qiymət aldı. Base xassəsinin adı burada bir rol oynamır, yalnız rahatlıq üçün belə seçilib.

this.radius = radius || 0.5;

this.edgeColor = edgeColor || "black";

Burada isə, yeni xassələrə ilkin qiymət veririk. Bu xassələr üçün susmaya görə qiymət təyin olunduğuna diqqət yetirin. Bu susmaya görə təyin olunan qiymətlər, əgər bizim bu sinfi istifadə edən proqramçı təyin olunan qiymətlərdən birini (və ya ümumiyyətlə heç birini) konstruktora verməzsə, istifadə olunur. Susmaya görə qiymət yaxşı stil proqramlaşdırmadır.

}

SuperPoint.prototype = new Point;

Qeyd edək ki, bu sətir konstruktorun gövdəsindən kənarda yerləşir.  
Artıq indi yeni sinifdən obyekt yarada və onun yeni xassələri kimi, irsən alınan xassələrini də istifadə edə bilərik.

var obj;

obj = new SuperPoint(100, 100, “green”, 0.6);

Burada biz konstruktora bütün parametrləri vermədik, lakin bu qəribə deyil, axı biz xassə üçün susmaya görə qiymətə baxmışıq.

obj.radius = Math.SQRT(5);

İndi biz SuperPoint sinfinin yeni xassəsinə müraciət etdik və ona 5-in kvadratı qiymətini mənimsətdik.

a = obj.x + 20;

İndi isə, Point sinfindən irsən alınan xassəyə müraciət etdik.  
Biz valideyn sinfindən irsən alınan istənilən xassə və ya metodu yenidən təyin (override) edə bilərik. Daha bir sadə misala baxaq.

function SuperPuperPoint(x, y, color, radius, edgeColor)

{

this.base = SuperPoint;

this.base(x, y, color, radius, edgeColor);

this.isPointInBounds = isPointInBounds2;

Burada biz isPointInBounds xassəsinə digər isPointInBounds2 funksiyasına göstərici mənimsədərək, isPointInBounds metodunu yenidən təyin etdik.

}

SuperPuperPoint.prototype = new SuperPoint;

Və bizim yeni sinfin valideynini göstərməyi unutmayın.  
Point, SuperPoint və SuperPuperPoint siniflərinin tam kodları aşağıda verilmişdir.

<html>

<head>

<title>

Class Test

</title>

<script language="JavaScript">

// Begin Declaration Point class

function Point(x, y, color)

{

this.x = x || 100;

this.y = y || 50;

this.color = color || “black”;

this.isPointInBounds = isPointInBounds;

}

function isPointInBounds()

{

if(this.x>0 && this.x<400)

{

return (this.y>0 && this.y<200);

}

else

{

return false;

}

}

// End Declaration Point class

// Begin Declaration SuperPoint class

function SuperPoint(x, y, color, radius, edgeColor)

{

this.base = Point;

this.base(x, y, color);

this.radius = radius || 0.5;

this.edgeColor = edgeColor || “black”;

}

SuperPoint.prototype = new Point;

// End Declaration SuperPoint class

// Begin Declaration SuperPuperPoint class

function SuperPuperPoint(x, y, color, radius, edgeColor)

{

this.base = SuperPoint;

this.base(x, y, color, radius, edgeColor);

this.isPointInBounds = isPointInBounds2;

}

SuperPuperPoint.prototype = new SuperPoint;

function isPointInBounds2()

{

if(this.x>0 && this.x<300)

{

return (this.y>0 && this.y<300);

}

else

{

return false;

}

}

// End Declaration SuperPuperPoint class

</script>

</head>

<body>

</body>

</html>