**Конструкторы**

// добавлять новые свойства и методы можно только к объектам созданым через оператор new

var objectStr = new String("some String object");

objectStr.customProperty = 123;  
document.write(objectStr.customProperty + "<br />");

Когда интерпретатор читает переменную или литерал, то он временно создает объект, а потом удаляет его.

Кастомные конструкторы, создаются как обычно.

// Функция-конструктор для создания объектов Point.  
function Point(x, y) {  
 this.x = x;  
 this.y = y;  
}

Переменные создаются на лету

// Функция-конструктор для создания объектов Human.  
function Human(name) {  
 // свойство  
 this.firstName = name;  
 // метод - свойство объекта, которое содежит значение в виде функции.  
 this.sayHello = function () {  
 document.write("Hello! My name is " + this.firstName + "<br /> ");  
 }  
}

Еще такое извращение, это эквивалент function Friend(mood){)

var Friend = function (mood) {  
   
 this.mood = mood;  
  
 // если настроение "хорошее"  
 if (mood === "nice") {  
 this.talk = function () {  
 alert("Hello what's up?");  
 };  
 }  
 // если настроение "плохое"  
 else if (mood === "bad") {   
 this.talk = function () {  
 alert("Sorry, I don't want talk with you.");  
 };  
 }  
 // если настроение не определено  
 else {   
 this.talk = function () {  
 alert("Hi");  
 };  
 }  
}  
  
var friendMood = prompt("Type friend mood (nice/bad):");  
var David = new Friend(friendMood);  
  
David.talk();

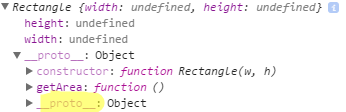
Статические методы у объекта JS

function Point(x, y) {  
  
 // свойства экземпляра  
 this.x = x;  
 this.y = y;  
  
 // метод экземпляра  
 this.print = function () {  
 document.write("(" + this.x + ", " + this.y + ")<br />");  
 }  
}  
  
// свойство функции конструктора (аналог статических свойств).  
Point.MaxPointCount = 100;  
  
// Метод функции-конструктора (аналог статическгого метода).  
Point.getOrigin = function () {  
 return new Point(0, 0) // this использовать недопустимо.  
}  
  
// создание экземпляров и работа с свойствами и методами экземпляров.  
var p1 = new Point(100, 200);  
p1.x = 300;  
p1.y = 400;  
p1.print();  
  
var p2 = new Point(100, 200);  
p2.print();  
  
// работа с свойством и методом функции-конструктора.  
Point.MaxPointCount = 10;  
Point.getOrigin().print();

Прототипы

function Rectangle(w, h) {  
 // свойства  
 this.width = w;  
 this.height = h;  
}  
  
// Метод прототипа Rectangle будет доступен каждому экземпляру, но храниться будет в прототипе соответственно занимать меньше места   
// чем если бы метод принадлежал каждому экземпляру.  
Rectangle.prototype.getArea = function () {  
 return this.width \* this.height;  
}  
  
var rect = new Rectangle(100, 50);  
document.write("Площадь прямоугольника " + rect.getArea());

функции которые в прототипе будут лежать в свойстве \_\_proto\_\_



Прототип создается для функции, и у всех ее объектов будет один прототип, к которому интерпретатор будут обращаться в случае если метод или свойство не будет найдено в объекте функции.

// При попытке смены значения, которое было унаследовано от прототипа, в объекте будет создана копия свойства (свойство прототипа не изменится).   
// Значение в прототипе может быть измененно только черзе операцию Rectangle.prototype.name  
//rect1.name = "first rectangle";  
Rectangle.prototype.name = "first rectangle";

Так rect1.name мы просто у объекта создодом свойство с таки именем. Сначала интерпретатор ищет в объекте и если есть дает его и не идет дальше в прототип, даже если оно пустое.

*constructor - свойство, которое хранит ссылку на функцию-конструктор, через которую был создан объект.*

**Куки**

function setCookie() {  
 // Что бы связать временное значение cookie файла с текущим документом. достаточно выполнить следующую операцию  
 document.cookie = "info=123"  
  
 // Значение Cookie не могут содержать точку с запятой, запятые или символы разделители. Перед сохранением значение в cookie желательно вызвать  
 // функцию encodeURIComponent() при чтении значения надо вызвать decodeURIComponent()  
 // записанный таким способом cookie сохраняются в текущем сеансе браузера но удаляются после его закрытия.  
}  
  
function getCookie() {  
 alert(document.cookie);  
}

Время жизни

function setCookie() {  
 // следующий cookie будет хранится браузером на проятжении 1 недели (60 \* 60 \* 24 \* 7).  
 document.cookie = "info=123; max-age=604800";  
  
 // Существует возможность указать время жизни cookie через устаревшее свойство expires  
 var nextYear = new Date();  
 nextYear.setFullYear(nextYear.getFullYear() + 1);  
 document.cookie = "name=Ivan Ivanv; expires=" + nextYear.toGMTString();  
}

Изменение

// задаю  
document.cookie = "lang=HTML; max-age=120";  
// меняю  
document.cookie = "lang=JavaScript; max-age=120";  
// меняю   
document.cookie = "lang=JavaScript; max-age=0";

URL кодирование

// URL кодирование данных  
encodeURIComponent('фыв'); // %D1%84%D1%8B%D0%B2  
// URL декодирование данных  
decodeURIComponent('%D1%84%D1%8B%D0%B2'); // фыв

document.cookie = "version=" + encodeURIComponent(document.lastModified);

Поиск куков

function findCookieValue(cookieName) {  
 var allcookies = document.cookie;  
 var pos = allcookies.indexOf(cookieName + "="); // version=  
  
 // Если cookie с указанным именем найден, извлечь его значения.  
 if (pos != -1) {  
 var start = pos + cookieName.length + 1;  
 var end = allcookies.indexOf(";", start);  
  
 if (end == -1) {  
 end = allcookies.length;  
 }  
  
 var value = allcookies.substring(start, end);  
  
 return decodeURIComponent(value);  
 }  
}