JavaScript Arrays

JavaScript arrays are used to store multiple values in a single variable.

```
var cars = ["Saab", "Volvo", "BMW"];
```

تعریف آرایه با استفاده از کلمه کلیدی new

```
var list = new Array(1,2,3,4,5,6)
```

Access the Elements of an Array

```
var cars = ["Saab", "Volvo", "BMW"];
document.getElementById("demo").innerHTML = cars[0];
```

The length Property

The length property of an array returns the length of an array (the number of array elements).

```
var fruits = ["Banana", "Orange", "Apple", "Mango"];
fruits.length;   // the length of fruits is 4
```

JavaScript Array Methods

Popping and Pushing

The `pop()` method removes the last element from an array:

```
var fruits = ["Banana", "Orange", "Apple", "Mango"];
fruits.pop();           // Removes the last element ("Mango") from fruits
```

The `push()` method adds a new element to an array (at the end):

```
var fruits = ["Banana", "Orange", "Apple", "Mango"];
fruits.push("Kiwi");      //  Adds a new element ("Kiwi") to fruits
```

نکته: Popp و Push از انتهای لیست عملیات را انجام می دهد.

Shifting Elements

The `shift()` method removes the first array element and "shifts" all other elements to a lower index.

```
var fruits = ["Banana", "Orange", "Apple", "Mango"];
var x = fruits.shift();   // the value of x is "Banana"
```

The `unshift()` method adds a new element to an array (at the beginning), and "unshifts" older elements:

```
var fruits = ["Banana", "Orange", "Apple", "Mango"];
fruits.unshift("Lemon");   // Adds a new element "Lemon" to fruits
```

نکته: Shift و Unshift از ابتدای لیست عملیات را انجام می دهد.

Indexof Elements

اندیس عناصر آرایه را نمایش می دهد.

```
var list = [1, 2, 3, 4, 5, 6, 7];
list.indexof("3");   // index is 2
```

Splicing an Array

The `splice()` method can be used to add new items to an array:

این متد دو عدد میگیرد که عدد اول ایندکس عنصر در آرایه را مشخص می کند و عدد دوم تعداد عناصری را که می خواهیم جدا کند.

```
var list = [1, 2, 3, 4, 5, 6, 7];
var newlist = list.splice(2, 2);
```

```
Result:
NewList: 3,4
List: 1,2,5,6,7
```

روش دیگر استفاده از متد Splicing اضافه کردن عنصر جدید در مکان مورد نظر در آرایه است.

نوضیح مثال: در اندیس ۲ مقدار ۰ عنصر را جدا کرده و دو مقدار Lemon و Kiwi را به آرایه اضافه می کند.

```
var fruits = ["Banana", "Orange", "Apple", "Mango"];
fruits.splice(2, 0, "Lemon", "Kiwi");
```

```
Result:
Banana,Orange,Lemon,Kiwi,Apple,Mango
```

Slicing an Array

The slice() method slices out a piece of an array into a new array.

این متد دو عدد میگیرد که ایندکس اول و پایان را میگیرد و عناصر را از آرایه جدا می کند.

```
var list = [1, 2, 3, 4, 5, 6, 7];
var newlist = list.splice(2, 2);
```

```
Result:
NewList: 3,4
List: 1,2,3,4,5,6,7
```

تفاوت این متد با متد Splicing در این است که متد Slicing بر خلاف تابع Splicing در آرایه اول تغییری را ایجاد نمی کند.

Converting Arrays to Strings

The JavaScript method `toString()` converts an array to a string of (comma separated) array values.

```
var fruits = ["Banana", "Orange", "Apple", "Mango"];
document.getElementById("demo").innerHTML = fruits.toString();
```

```
Result:
Banana,Orange,Apple,Mango
```

The `join()` method also joins all array elements into a string.

It behaves just like `toString()`, but in addition you can specify the separator:

متد join() هم همه عناصر آرایه را به یک رشته تبدیل می کند.

این دقیقاً مانند toString رفتار می کند ، اما علاوه بر این می توانید جداکننده را مشخص کنید:

```
var fruits = ["Banana", "Orange", "Apple", "Mango"];
document.getElementById("demo").innerHTML = fruits.join(" * ");
```

```
Result:
Banana * Orange * Apple * Mango
```

Merging (Concatenating) Arrays

The `concat()` method creates a new array by merging (concatenating) existing arrays:

```
var myGirls = ["Cecilie", "Lone"];
var myBoys = ["Emil", "Tobias", "Linus"];
var myChildren = myGirls.concat(myBoys);   // Concatenates (joins) myGirls and myBoys
```

این متد دو آرایه را به هم متصل می کند.

Example (Merging Three Arrays)

```
var arr1 = ["Cecilie", "Lone"];
var arr2 = ["Emil", "Tobias", "Linus"];
var arr3 = ["Robin", "Morgan"];
var myChildren = arr1.concat(arr2, arr3);   // Concatenates arr1 with arr2 and arr3
```

Sort Arrays

این متد جهت مرتب سازی آرایه استفاده می شود.

```
var list = [4, 6, 2, 5, 1, 3];
var newlist = list.sort();  // 1,2,3,4,5,6
```

Reverse Arrays

این متد جهت مرتب سازی آرایه ازآخر به اول استفاده می شود.

```
var list = [4, 6, 2, 5, 1, 3];
var newlist = list. reverse ();  // 6, 5, 4, 3, 2, ,
```

Filter Arrays

جهت فیلتر کردن آرایه به کار می رود.

مثال اعدادی را که باقی مانده تقسیم آنها ۱ است را در لیست جدید قرار دهید.

```
var list = [1, 2, 3, 4, 5, 6, 7];
var newlist = list.filter(function(item){
    return item % 2 == 1;
    });   // 1, 3, 5, 7, 9
```

Map Arrays

جهت انجام یک عمل بر روی یک آرایه به کار می رود.

```
var list = [1, 2, 3, 4, 5, 6, 7];
var newlist = list.map(function(item){
    return item * 2;
    });   // 2, 4, 6, 8, 10, 12, 14
```

JavaScript Objects

Objects ها نیز متغیر هستند. اما Objects ها می توانند حاوی مقادیر (properties) زیادی باشند.

این کد مقادیر زیادی را به متغیری با نام car اختصاص می دهد:

```
var car = {type:"Fiat", model:"500", color:"white"};
```

دسترسی به مقادیر (properties) یک Object جهت تغییر (set) مقدار آن.

```
car.model = "600";
```

جهت اضافه کردن یک property می توانید یک نام جدید را (که قبلا در Object تعریف نشده) به شکل زیر وارد نمایید.

```
car.wheels = 4;
```

فاصله و خط شکسته مهم نیست. تعریف شی می تواند چندین خط را شامل شود:

```
var person = {
  firstName: "John",
  lastName: "Doe",
  age: 50,
  eyeColor: "blue"
};
```

مقادیر استفاده شده در Object ها میتواند array و یا function نیز باشد.

```
var car = {
    type: 'fiat',
    model: '500',
    details: {
        colors: ['red', 'blue', 'black'],
        wheels: 4
    }
}
```

```
var person = {
  firstName: "John",
  lastName : "Doe",
  id       : 5566,
  fullName : function() {
    return this.firstName + " " + this.lastName;
  }
};
```

نکته: جهت استفاده از property های یک Object در function باید از کلمه کلیدی this استفاده نمایید.

در تعریف تابع ، this به "مالک" تابع اشاره دارد.

نکته: جهت نمایش حاصل یک تابع می بایست به روش زیر آن را فراخوانی کنید.

```
consol.log(person.fullName());
```

Object Constructors

مثالهای قبل محدود است. آنها فقط اشیاء مجرد ایجاد می کنند.

بعضی اوقات برای ایجاد بسیاری از اشیاء با همان "نوع" به یک "طرح" نیاز داریم.

راه ایجاد "object type" ، استفاده از یک تابع سازنده object است.

در مثال پایین ، تابع Person () یک تابع سازنده object است.

object از همان نوع با فراخوانی تابع سازنده با کلمه کلیدی جدید ایجاد می شوند:

```
function Person(first, last, age, eye){
    this.first = first;
    this.last = last;
    this.age = age;
    this.eye = eye;
}
var ali = new Person("ali", "rezaei", "27", "green");
console.log(ali);
```

Result:

```
Person { first: 'ali', last: 'rezaei', age: '27', eye: 'green' }
```

Prototypes

همه Object های جاوا اسکریپت از یک نمونه اولیه خواص (Prototypes) و method ها را به ارث می برند.

اشیاء تاریخ از Date.prototype به ارث می برند

اشیاء آرایه را از Array.prototype به ارث می برند

اشیاء شخص از Person.prototype به ارث می برند

Object.prototype در بالای زنجیره ارثی نمونه اولیه است:

اشیاء تاریخ ، اشیاء آرایه و اشیاء شخص از Object.prototype به ارث می برند.

ما نمی توانیم یک ویژگی جدید را به سازنده موجود موجود اضافه کنیم.

برای افزودن یک ویژگی جدید به سازنده ، باید آن را به تابع سازنده (constructor function) اضافه کنید.

خاصیت نمونه اولیه (prototype property) به شما امکان می دهد خصوصیات جدیدی را به سازندگان شی اضافه کنید.

```
function Person(first, last, age, eyecolor) {
  this.firstName = first;
  this.lastName = last;
  this.age = age;
  this.eyeColor = eyecolor;
}

Person.prototype.nationality = "English";
```

خاصیت نمونه اولیه (prototype property) به شما امکان می دهد methods های جدیدی به سازندگان اشیاء (objects constructors) اضافه کنید.

```
function Person(first, last, age, eyecolor) {
  this.firstName = first;
  this.lastName = last;
  this.age = age;
  this.eyeColor = eyecolor;
}

Person.prototype.name = function() {
  return this.firstName + " " + this.lastName;
};
```

نکته: فقط نمونه های اولیه خود را اصلاح کنید. هرگز نمونه های اولیه اشیاء استاندارد JavaScript را تغییر ندهید.

جهت اضافه شدن یک methods به تمام Prototype ها کافیست آن Prototype را به Object اضافه کنید

```
Object.prototype.fullname = function(){
   return this.first + " " + this.last
}
```

Inheritance

وراثت جاوا (زیر کلاس و ابر کلاس)

جهت تعریف یک Object به صورتی که هیچ خصوصیتی را به صورت پیش فرض به همراه نداشته باشد به صورت پایین استفاده می کنیم.

```
var car2 = Object.create(null)
```

در صورتی که بخواهیم در ساخت یک Object از خصوصیات یک Object دیگر ارث بری کنیم کافیست به صورت پایین استفاده کنیم.

```
var car = {type:"Fiat", model:"500", color:"white"};
var car2 = Object.create(car1);
```

Function Call

با استفاده از متد call می توانید متدی بنویسید که بتواند روی اشیاء مختلف استفاده شود.

```
var person = {
  fullName: function() {
    return this.firstName + " " + this.lastName;
  }
}
var person1 = {
  firstName:"John",
  lastName: "Doe"
}
var person2 = {
  firstName:"Mary",
  lastName: "Doe"
}
person.fullName.call(person2);  // Will return "Mary Doe"
```

همه توابع، متد هستند.

call یک متد از پیش تعریف شده JavaScript است. می توان از آن برای فراخوانی (call) متدی با یک شیء مالک به همراه آرگومان (پارامتر) استفاده کرد.

```
var person = {
  fullName: function(city, country) {
    return this.firstName + " " + this.lastName + "," + city + "," +
country;
  }
}
var person1 = {
  firstName:"John",
  lastName: "Doe"
}
person.fullName.call(person1, "Oslo", "Norway");
```

```
result:
John Doe,Oslo,Norway
```

Function Apply

متد Apply شبیه به متد Call است.

```
var person = {
  fullName: function() {
    return this.firstName + " " + this.lastName;
  }
}
var person1 = {
  firstName: "Mary",
  lastName: "Doe"
}
person.fullName.apply(person1);  // Will return "Mary Doe"
```

تفاوت بین Call و Apply

تفاوت این است:

روش Call آرگومان ها را جداگانه می گیرد.

روش Apply آرگومانها را به عنوان یک آرایه در نظر می گیرد.

```
var person = {
  fullName: function(city, country) {
    return this.firstName + " " + this.lastName + "," + city + "," +
country;
  }
}
var person1 = {
  firstName:"John",
  lastName: "Doe"
}
person.fullName.apply(person1, ["Oslo", "Norway"]);
// Will return "Mary Doe"
```

Function Bind

متد Bind شبیه به متد Call است. با این تفاوت که متد Bind به صورت خودکار فراخوانی نمی شود و نیاز است که آن را فراخوانی کرد.

```
var person = {
    fullName: function(city, country) {
      return this.firstName + " " + this.lastName + "," + city + "," +
country;
    }
  }
  var person1 = {
    firstName:"John",
    lastName: "Doe"
  }
  var x =person.fullName.bind(person1, "Oslo", "Norway");
console.log(x());
```

همچنین به جای قرار دادن متد در متغییر x می توان تابع را به شکل زیر اجرا کرد.

```
person.fullName.bind(person1, "Oslo", "Norway")();
```

JavaScript Errors

JavaScript try and catch

عبارت try به شما امکان می دهد تا در حین اجرا ، بلوکی از کد را برای آزمایش خطاها تعریف کنید.

عبارت catch به شما امکان می دهد اگر خطایی در بلوک امتحان رخ دهد ، یک بلوک کد را اجرا کنید.

```
try {
  Block of code to try
}
catch(err) {
  Block of code to handle errors
}
```

JavaScript Throws Errors

اگر از Throws با try and catch استفاده می کنید ، می توانید جریان برنامه را کنترل کنید و پیام های خطای سفارشی تولید کنید.

```
function myFunction() {
  var message, x;
  message = document.getElementById("p01");
  message.innerHTML = "";
  x = document.getElementById("demo").value;
  try {
    if(x == "") throw "empty";
    if(isNaN(x)) throw "not a number";
    x = Number(x);
    if(x < 5) throw "too low";
    if(x > 10) throw "too high";
  }
  catch(err) {
    message.innerHTML = "Input is " + err;
  }
}
```

finally Statement

finally به شما امکان می دهد بدون در نظر گرفتن نتیجه ، کد را اجرا کنید ، پس از try and catch.

```
try {
  Block of code to try
}
catch(err) {
  Block of code to handle errors
}
finally {
  Block of code to be executed regardless of the try / catch result
}
```

JavaScript Timing Events

امکان اجرای کد را در فواصل زمانی مشخص فراهم می کند.

دو روش اصلی برای استفاده با JavaScript عبارتند از:

setTimeout (عملکرد ، میلی ثانیه)

بعد از انتظار تعداد مشخصی از میلی ثانیه ، یک عملکرد را انجام می دهد.

setInterval (عملکرد ، میلی ثانیه)

همانند setTimeout () ، اما اجرای کار را به طور مداوم تکرار می کند.

setTimeout Method

اولین پارامتر تابعی است که باید اجرا شود.

پارامتر دوم تعداد میلی ثانیه قبل از اجرا را نشان می دهد.

```
Var setTimeout(myFunction, 3000);
function myFunction() {
  alert('Hello');
}
```

متد clearTimeout () اجرای عملکرد مشخص شده در setTimeout را متوقف می کند.

```
myVar = setTimeout(function, milliseconds);
clearTimeout(myVar);
```

setInterval Method

متد setInterval یک تابع داده شده را در هر بازه زمانی مشخص تکرار می کند.

```
var myVar = setInterval(myTimer, 1000);

function myTimer() {
  var d = new Date();
  document.getElementById("demo").innerHTML = d.toLocaleTimeString();
}
```

متد clearInterval تابع مشخص شده در متد setInterval را متوقف می کند.

```
myVar = setInterval(function, milliseconds);
clearInterval(myVar);
}
```

JavaScript String Methods

متد های رشته به شما کمک می کند تا با رشته ها کار کنید.

String Length

Length ویژگی طول یک رشته را برمی گرداند

```
var txt = "ABCDEFGHIJKLMNOPQRSTUVWXYZ";
var sln = txt.length;
```

Finding a String in a String

متد indexOf موقعیت اولین رخداد یک متن مشخص در یک رشته را برمی گرداند

```
var str = "Please locate where 'locate' occurs!";
var pos = str.indexOf("locate");
```

متد lastIndexOf موقعیت آخرین وقوع یک متن مشخص در یک رشته را برمی گرداند

```
var str = "Please locate where 'locate' occurs!";
var pos = str.lastIndexOf("locate");
```

اگر متن یافت نشد ، indexOf و lastIndexOf گزینه -۱ را برمی گردانند.

هر دو روش یک پارامتر دوم را به عنوان موقعیت شروع جستجو می پذیرند (مثلا از کاراکتر ۱۵ به بعد را جستجو می کند).

```
var str = "Please locate where 'locate' occurs!";
var pos = str.indexOf("locate", 15);
```

متد lastIndexOf به عقب جستجو می کند (از انتهای تا شروع) ، به این معنی: اگر پارامتر دوم ۱۵ باشد ، جستجو در موقعیت ۱۵ شروع می شود و به ابتدای رشته جستجو می کند.

متد search رشته ای را برای یک مقدار مشخص جستجو می کند و موقعیت آن را برمی گرداند:

```
var str = "Please locate where 'locate' occurs!";
var pos = str.search("locate");
```

Extracting String Parts

قطعات استخراج رشته

۳ روش برای استخراج بخشی از رشته وجود دارد:

- slice(start, end)
- substring(start, end)
- substr(start, length)

slice Method

slice بخشی از یک رشته را استخراج می کند و قسمت استخراج شده را در یک رشته جدید برمی گرداند.

این روش ۲ پارامتر را در بر می گیرد: موقعیت شروع و موقعیت انتهایی (پایان درج نشده است).

```
var str = "Apple, Banana, Kiwi";
var res = str.slice(7, 13);
```

Result:

Banana

اگر یک پارامتر منفی باشد ، موقعیت از انتهای رشته محاسبه می شود.

این مثال بخشی از رشته را از موقعیت -۱۲ تا موقعیت -۶ بیان می کند

```
var str = "Apple, Banana, Kiwi";
var res = str.slice(-12, -6);
```

Result:

Banana

اگر پارامتر دوم را حذف کنید ، متد بقیه رشته را قطعه قطعه می کند

```
var res = str.slice(7);
```

Result:

Banana, Kiwi

یا از آخر حساب کنید

```
var res = str.slice(-12);
```

substring Method

substring شبیه به slice است.

تفاوت این است که substring نمی تواند شاخص های منفی را بپذیرد.

substr Method

substr () شبیه به slice است.

تفاوت در این است که پارامتر دوم طول قسمت استخراج شده را مشخص می کند.

```
var str = "Apple, Banana, Kiwi";
var res = str.substr(7, 6);
```

اگر پارامتر دوم را فراموش کنید ، substr بقیه رشته را قطعه قطعه می کند.

اگر پارامتر اول منفی باشد ، موقعیت از انتهای رشته شمارش می شود.

Replacing String Content

متد replace مقدار مشخص شده را با مقدار دیگری در یک رشته جایگزین می کند

```
str = "Please visit Microsoft!";
var n = str.replace("Microsoft", "W3Schools");
```

متد replace رشته ای را که فراخوانی می شود تغییر نمی دهد. رشته جدید را برمی گرداند.

به طور پیش فرض ، متد replace فقط اولین مورد را جایگزین می کند.

```
str = "Please visit Microsoft and Microsoft!";
var n = str.replace("Microsoft", "W3Schools");
```

```
Result:
Please visit Microsoft and Microsoft!
```

به طور پیش فرض ، متد replace حساس به مورد است. نوشتن MICROSOFT (با حروف بزرگ) عملی نخواهد بود

برای جایگزینی موارد غیر حساس ، از یک عبارت معمولی با پرچم / i (غیر حساس) استفاده کنید

```
str = "Please visit Microsoft!";
var n = str.replace(/MICROSOFT/i, "W3Schools");
```

برای جایگزینی همه موارد ، از یک عبارت معمولی با پرچم / g استفاده کنید

```
str = "Please visit Microsoft and Microsoft!";
var n = str.replace(/Microsoft/g, "W3Schools");
```

Converting to Upper and Lower Case

متد Upper جهت تبدیل رشته به حروف بزرگ و متد Lower جهت تبدیل رشته به حروف کوچک استفاده می شود.

```
var text1 = "Hello World!";       // String
var text2 = text1.toUpperCase();  // text2 is text1 converted to upper
var text3 = text1.toLowerCase();  // text3 is text1 converted to lower
```

concat Method

جهت الحاق چند رشته مورد استفاده قرار می گیرد.

```
var text1 = "Hello";
var text2 = "World";
var text3 = text1.concat(" ", text2);
```

Trim

متد Trim فضای سفید را از هر دو طرف یک رشته حذف می کند

```
var str = "       Hello World!        ";
alert(str.trim());
```

charAt Method

متد charAt کاراکتر را در یک موقعیت مشخص شده در یک رشته برمی گرداند

```
var str = "HELLO WORLD";
str.charAt(0);            // returns H
```

The charCodeAt Method

متد charCodeAt یونیکد کاراکتر را در یک شاخص مشخص در یک رشته برمی گرداند

این روش یک کد UTF-16 (یک عدد صحیح بین ۰ تا ۶۵۵۳۵) را برمی گرداند.

```
var str = "HELLO WORLD";
str.charCodeAt(0);         // returns 72
```

JavaScript Date Objects

به طور پیش فرض ، JavaScript از منطقه زمانی مرورگر استفاده می کند و تاریخ را به عنوان یک متن کامل نمایش می دهد:

سه شنبه ۲۱ ژانویه ۲۰۲۰ ۱۴:۲۸:۳۴ GMT + 0330 (زمان استاندارد ایران).

Creating Date Objects

```
new Date()
new Date(year, month, day, hours, minutes, seconds, milliseconds)
new Date(milliseconds)
new Date(date string)
```

۷ عدد سال ، ماه ، روز ، ساعت ، دقیقه ، ثانیه و میلی ثانیه (به ترتیب) را مشخص می کند.

```
var d = new Date(2018, 11, 24, 10, 33, 30, 0);
```

new Date(dateString)

new Date (dateString) یک مورد تاریخ جدید را از یک رشته تاریخ ایجاد می کند.

```
var d = new Date("October 13, 2014 11:13:00");
```

JavaScript Dates Milliseconds

new Date (میلی ثانیه) یک شیء از تاریخ جدید به عنوان زمان صفر به اضافه میلی ثانیه ایجاد می کند.

```
var d = new Date(100000000000);
```

Get Methods

این متد ها می توانند برای بدست آوردن اطلاعات از یک شی تاریخ استفاده شوند.

Method	Description
getFullYear()	Get the **year** as a four digit number (yyyy)
getMonth()	Get the **month** as a number (0-11)
getDate()	Get the **day** as a number (1-31)
getHours()	Get the **hour** (0-23)
getMinutes()	Get the **minute** (0-59)
getSeconds()	Get the **second** (0-59)
getMilliseconds()	Get the **millisecond** (0-999)
getTime()	Get the time (milliseconds since January 1, 1970)

getDay()	Get the weekday as a number (0-6)
Date.now()	Get the time. ECMAScript 5.

Set Date Methods

متد های set date برای تنظیم بخشی از تاریخ استفاده می شود

Method	Description
setDate()	Set the day as a number (1-31)
setFullYear()	Set the year (optionally month and day)
setHours()	Set the hour (0-23)
setMilliseconds()	Set the milliseconds (0-999)
setMinutes()	Set the minutes (0-59)
setMonth()	Set the month (0-11)
setSeconds()	Set the seconds (0-59)
setTime()	Set the time (milliseconds since January 1, 1970)

JavaScript Math Object

JavaScript Math به شما امکان می دهد کارهای ریاضی را بر روی اعداد انجام دهید.

Math.round

Math.round (x)مقدار x گرد شده را به نزدیکترین عدد صحیح خود بازمی گرداند

```
Math.round(4.7);    // returns 5
Math.round(4.4);    // returns 4
```

Math.pow

مقدار x را به توان y بازمی گرداند

```
Math.pow(8, 2);      // returns 64
```

Math.sqrt

ریشه مربع x را برمی گرداند

```
Math.sqrt(64);      // returns 8
```

Math.abs

مقدار مطلق (مثبت) x را برمی گرداند

```
Math.abs(-4.7);     // returns 4.7
```

Math.ceil

مقدار x گرد شده را به نزدیکترین عدد صحیح خود (به بالا) بازمی گرداند

```
Math.ceil(4.4);     // returns 5
```

Math.floor

مقدار x گرد شده را به نزدیکترین عدد صحیح خود (به پایین) بازمی گرداند

```
Math.floor(4.7);    // returns 4
```

Math.sin and Math.cos

محاسبه سینوس و کسینوس

```
Math.sin(90 * Math.PI / 180);     // returns 1 (the sine of 90 degrees)
Math.cos(0 * Math.PI / 180);     // returns 1 (the cos of 0 degrees)
```

Math.min and Math.max

Math.min و Math.max را می توان برای یافتن کمترین یا بالاترین مقدار در لیستی از استدلال ها استفاده کرد

```
Math.min(0, 150, 30, 20, -8, -200);  // returns -200
Math.max(0, 150, 30, 20, -8, -200);  // returns 150
```

Math.random

Math.random عدد تصادفی را بین ۰ (شامل) و ۱ (اختصاصی) برمی گرداند

```
Math.random();     // returns a random number
```

مثال های دیگر از متد Math.random

```
Math.floor(Math.random() * 10);     // returns a random integer from 0
to 9
```

```
Math.floor(Math.random() * 100);     // returns a random integer from 0
to 99
```

```
Math.floor(Math.random() * 100) + 1; // returns a random integer from 1
to 100
```

```
function getRndInteger(min, max) {
  return Math.floor(Math.random() * (max - min) ) + min;
}
```

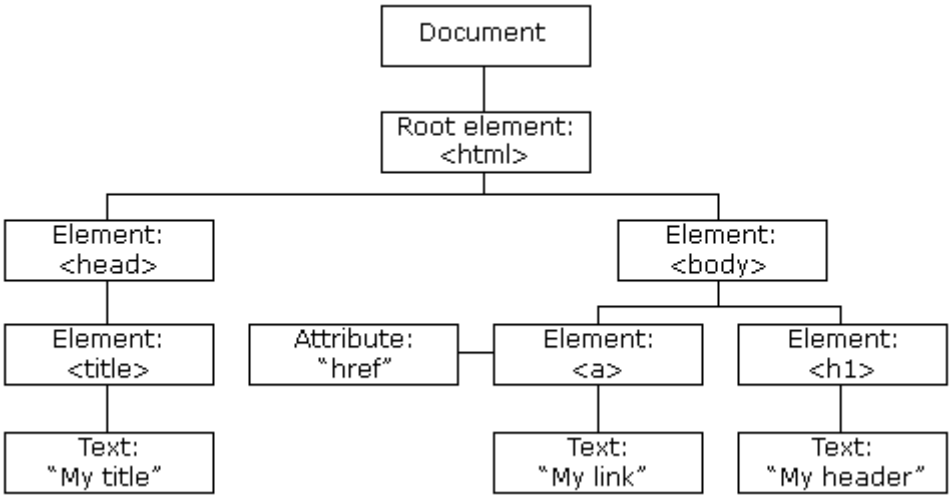
JavaScript HTML DOM

HTML DOM (Document Object Model)

هنگامی که یک صفحه وب بارگیری می شود ، مرورگر یک مدل شیء از صفحه را ایجاد می کند.

مدل HTML DOM به عنوان درخت اشیاء ساخته شده است

The HTML DOM Tree of Objects



DOM چیست؟

DOM یک استاندارد W3C (کنسرسیوم جهانی وب) است.

DOM استانداردی را برای دستیابی به اسناد تعریف می کند:

"W3C Document Object Model (DOM) یک بستر و رابط بی طرف است که به برنامه ها و اسکریپت ها امکان می دهد به طور پویا به محتوا ، ساختار و سبک یک سند دسترسی پیدا کنند و به روز کنند."

استاندارد W3C DOM به ۳ قسمت مختلف تقسیم می شود:

- Core DOM – مدل استاندارد برای انواع سند
- XML DOM – مدل استاندارد برای اسناد XML
- HTML DOM – مدل استاندارد برای اسناد HTML

HTML DOM چیست؟

HTML DOM یک استاندارد برای نحوه دریافت ، تغییر ، اضافه کردن یا حذف عناصر HTML است.

HTML DOM Methods

متد های HTML DOM مقادیر (از عناصرHTML) هستند که می توانید تنظیم یا تغییر دهید.

مثال زیر محتوای (innerHTML) عنصر <p> را با id "demo" تغییر می دهد

```
document.getElementById("demo").innerHTML = "Hello World!";
```

getElementById Method

متداول ترین روش دسترسی به عنصر HTML استفاده از شناسه عنصر است.

The innerHTML Property

ساده ترین راه برای به دست آوردن محتوای یک عنصر استفاده از خاصیت innerHTML است.

خاصیت innerHTML برای بدست آوردن یا جایگزینی محتوای عناصر HTML مفید است.

HTML DOM Elements

Finding HTML Elements

غالباً با JavaScript می خواهید عناصر HTML را دستکاری کنید.

برای این کار ابتدا باید عناصر را پیدا کنید. چندین راه برای انجامش وجود دارد:

یافتن عناصر HTML توسط id

یافتن عناصر HTML بر اساس نام tag

یافتن عناصر HTML بر اساس نام class

یافتن عناصر HTML توسط انتخاب کنندگان CSS

یافتن عناصر HTML توسط مجموعه اشیاء HTML

Finding HTML Element by Id

```
var myElement = document.getElementById("intro");
```

Finding HTML Elements by Tag Name

```
var x = document.getElementsByTagName("p");
```

Finding HTML Elements by Class Name

```
var x = document.getElementsByClassName("intro");
```

Finding HTML Elements by CSS Selectors

```
var x = document.querySelectorAll("p.intro");
```

HTML DOM - Changing CSS

```
document.getElementById("p2").style.color = "blue";
```

HTML DOM Events

در صورت بروز یک رویداد ، مانند هنگامی که کاربر روی یک عنصر HTML کلیک می کند ، می توان JavaScript را اجرا کرد.

نمونه هایی از رویدادهای HTML:

- وقتی کاربر روی ماوس کلیک می کند
- وقتی یک صفحه وب بارگیری شد
- وقتی یک تصویر بارگیری شده است
- وقتی موس روی یک عنصر حرکت می کند
- وقتی یک فیلد ورودی تغییر می کند
- وقتی فرم HTML ارسال شد
- هنگامی که یک کاربر کلید را لمس می کند

The onclick Events

یک رویداد onclick را به یک عنصر دکمه اختصاص دهید.

```
document.getElementById("myBtn").onclick = displayDate;
```

The onload and onunload Events

هنگام ورود کاربر یا ترک صفحه ، رویدادهای بارگذاری و onunload ایجاد می شوند.

از رویداد بارگذاری می توان برای بررسی نوع مرورگر بازدید کننده و نسخه مرورگر استفاده کرد و نسخه مناسب صفحه وب را بر اساس اطلاعات بارگذاری کرد.

از وقایع بارگذاری و بارگیری برای مقابله با کوکی ها می توان استفاده کرد.

```
<body onload="checkCookies()">
```

The onchange Event

رویداد onchange اغلب در ترکیب با اعتبارسنجی فیلدهای ورودی استفاده می شود.

```
<input type="text" id="fname" onchange="upperCase()">
```

The onmouseover and onmouseout Events

```
<div onmouseover="mOver(this)" onmouseout="mOut(this)">
```

The onmousedown, onmouseup and onclick Events

```
<div onmousedown="mDown(this)" onmouseup="mUp(this)">
```

HTML DOM querySelector() Method

اولین عنصر موجود در سند را با کلاس "example" دریافت کنید.

```
document.querySelector(".example");
```

اولین عنصر <p> را در سند بدست آورید.

```
document.querySelector("p");
```

با استفاده از کلاس "example" اولین عنصر <p> را در سند دریافت کنید:

```
document.querySelector("p.example");
```

متن یک عنصر را با id "demo" تغییر دهید.

```
document.querySelector("#demo").innerHTML = "Hello World!";
```

اولین عنصر <p> را در مدرک دریافت کنید که در آن والدین عنصر هستند.

```
document.querySelector("div > p");
```

اولین عنصر <a> را در سندی که دارای ویژگی "target" است ، دریافت کنید.

```
document.querySelector("a[target]");
```

این مثال نشان می دهد که چگونه چندین انتخاب کننده کار می کنند، فرض کنید که شما دو عنصر دارید: یک عنصر <h2> و <h3>.
کد زیر یک رنگ زمینه را به اولین <h2> عنصر در سند اضافه می کند.

```
document.querySelector("h2, h3").style.backgroundColor = "red";
```

اما اگر عنصر <h3> قبل از عنصر <h2> در سند قرار داده شود. عنصر <h3> عنصری است که رنگ پس زمینه قرمز را بدست می آورد.

تمام عناصر موجود در سند را با کلاس "example" دریافت کنید.

```
var x = document.querySelectorAll(".example");
```

تمام عناصر <p> موجود در سند را بدست آورید ، و رنگ پس زمینه اولین عنصر <p> (index0) را تنظیم کنید.

```
// Get all <p> elements in the document
var x = document.querySelectorAll("p");

// Set the background color of the first <p> element
x[0].style.backgroundColor = "red";
```

تمام عناصر <p> موجود در سند را با کلاس "example" بدست آورید ، و زمینه اولین عنصر <p> را تنظیم کنید.

```
// Get all <p> elements in the document with class="example"
var x = document.querySelectorAll("p.example");

// Set the background color of the first <p> element with
class="example" (index 0)
x[0].style.backgroundColor = "red";
```

دریابید که چند عنصر با کلاس "example" در سند وجود دارد (با استفاده از ویژگی طول شیء NodeList):

```
var x = document.querySelectorAll(".example").length;
```

رنگ پس زمینه همه عناصر موجود در سند را با کلاس "example" تنظیم کنید:

```
var x = document.querySelectorAll(".example");
var i;
for (i = 0; i < x.length; i++) {
  x[i].style.backgroundColor = "red";
}
```

رنگ پس زمینه همه عناصر <p> را در سند تنظیم کنید.

```
var x = document.querySelectorAll("p");
var i;
for (i = 0; i < x.length; i++) {
  x[i].style.backgroundColor = "red";
}
```

رنگ پس زمینه همه عناصر h2، div و span را در سند تنظیم کنید.

```
var x = document.querySelectorAll("h2, div, span");
var i;
for (i = 0; i < x.length; i++) {
  x[i].style.backgroundColor = "red";
}
```

JavaScript HTML DOM EventListener

The addEventListener() method

```
element.addEventListener(event, function, useCapture);
```

اولین پارامتر نوع رویداد (مانند "click" یا "mousedown" یا هر رویداد HTML DOM دیگر است.)

پارامتر دوم تابعی است که می خواهیم هنگام رخ دادن رویداد آن را صدا بزنیم.

پارامتر سوم یک مقدار boolean است که مشخص می کند آیا از bubbling یا bubbling استفاده می شود. این پارامتر اختیاری است.

Add an Event Handler to an Element

```
element.addEventListener("click", function(){ alert("Hello World!"); });
```

نکته: *element* را می توان به شکل زیر استفاده کرد

```
Var el = document.getElementById("myP");
el.removeEventListener("mousemove", myFunction);
```

همچنین می توانید به یک تابع "با نام" خارجی مراجعه کنید

```
element.addEventListener("click", myFunction);

function myFunction() {
  alert ("Hello World!");
}
```

Add Many Event Handlers to the Same Element

متد addEventListener به شما امکان می دهد بسیاری از رویدادها را به همان عنصر اضافه کنید ، بدون آنکه رویدادهای موجود را نادیده بگیرید

```
element.addEventListener("click", myFunction);
element.addEventListener("click", mySecondFunction);
```

می توانید وقایع در انواع مختلف را به همان عنصر اضافه کنید

```
element.addEventListener("mouseover", myFunction);
element.addEventListener("click", mySecondFunction);
element.addEventListener("mouseout", myThirdFunction);
```

Passing Parameters

```
element.addEventListener("click", function(){ myFunction(p1, p2); });
```

Event Bubbling or Event Capturing?

دو روش انتشار رویداد در HTML DOM وجود دارد ، bubbling و capturing.

انتشار رویداد راهی است برای تعیین ترتیب عنصر هنگام وقوع یک رویداد. اگر درون یک عنصر یک عنصر <p> دارید ، و کاربر روی عنصر <p> کلیک می کند ، ابتدا باید "کلیک" کدام عنصر را انجام داد؟

در bubbling رویداد اصلی ترین عنصر ابتدا و سپس بیرونی انجام می شود: ابتدا رویداد کلیک عنصر <p> گرفته می شود ، سپس رویداد کلیک عنصر .

در capturing عناصر بیرونی ابتدا به کار گرفته می شود و سپس درونی: ابتدا با کلیک روی عنصر عنصر کلیک می شود و سپس رویداد کلیک عنصر <p>.

با استفاده از متد addEventListener () می توانید نوع انتشار را با استفاده از پارامتر "useCapture" مشخص کنید

```
addEventListener(event, function, useCapture);
```

مقدار پیش فرض false است که از انتشار bubbling استفاده می کند، هنگامی که مقدار true است، رویداد از انتشار capturing استفاده می کند.

```
document.getElementById("myP").addEventListener("click",
myFunction, true);
document.getElementById("myDiv").addEventListener("click",
myFunction, true);
```

The removeEventListener() method

متد removeEventListener حذف کننده های رویدادهای متصل به متد addEventListener را حذف می کند

```
element.removeEventListener("mousemove", myFunction);
```

stopPropagation() Event Method

اگر درون یک عنصر یک عنصر <p> دارید ، و کاربر روی عنصر <p> کلیک می کند ، رویداد هر دو عنصر اجرا می گردد، در صورتی که نیاز ندارید که رویداد والد عنصر اجرا شود از متد زیر می توانید استفاده کنید.

```
function func1(event) {
  alert("Message");
  event.stopPropagation();
}
```

preventDefault() Event Method

از بازکردن پیوند URL جلوگیری کنید

```
document.getElementById("myAnchor").addEventListener("click", function(e
vent){
  event.preventDefault()
});
```