Javascript

Construções peculiares

"use strict" (disponível a partir de ECMAScript 5)

Example

```
"use strict";
x = 3.14;       // This will cause an error because x is not declared
```

```
"use strict";
myFunction();

function myFunction() {
  y = 3.14;   // This will also cause an error because y is not declared
}
```

Global Scope

Variables declared **Globally** (outside any function) have **Global Scope**.

Example

```
var carName = "Volvo";

// code here can use carName

function myFunction() {
  // code here can also use carName
}
```

Global variables can be accessed from anywhere in a JavaScript program.

Function Scope

Variables declared **Locally** (inside a function) have **Function Scope**.

Example

```
// code here can NOT use carName

function myFunction() {
  var carName = "Volvo";
  // code here CAN use carName
}

// code here can NOT use carName
```

Local variables can only be accessed from inside the function where they are declared.

JavaScript Block Scope

Variables declared with the `var` keyword cannot have **Block Scope**.

Variables declared inside a block **{}** can be accessed from outside the block.

Example

```
{
  var x = 2;
}
// x CAN be used here
```

Before ES2015 JavaScript did not have **Block Scope**.

Variables declared with the `let` keyword can have Block Scope.

Variables declared inside a block **{}** cannot be accessed from outside the block:

Example

```
{
  let x = 2;
}
// x can NOT be used here
```

```
var x = 10;
// Here x is 10
{
  var x = 2;
  // Here x is 2
}
// Here x is 2
```

```
var x = 10;
// Here x is 10
{
  let x = 2;
  // Here x is 2
}
// Here x is 10
```

```
const PI = 3.141592653589793;
PI = 3.14;        // This will give an error
PI = PI + 10;     // This will also give an error
```

Hoisting

Example 1

```
x = 5; // Assign 5 to x

elem = document.getElementById("demo"); // Find an element
elem.innerHTML = x;                     // Display x in the element

var x; // Declare x
```

Example 2

```
var x; // Declare x
x = 5; // Assign 5 to x

elem = document.getElementById("demo"); // Find an element
elem.innerHTML = x;                     // Display x in the element
```

```
var x = 5; // Initialize x

elem = document.getElementById("demo"); // Find an element
elem.innerHTML = x + " " + y;           // Display x and y

var y = 7; // Initialize y
```

Declare as variáveis sempre no Topo!!!

Javascript suporta POO

Real Life Objects, Properties, and Methods

In real life, a car is an **object**.

A car has **properties** like weight and color, and **methods** like start and stop:

Object	Properties	Methods
	car.name = Fiat	car.start()
	car.model = 500	car.drive()
	car.weight = 850kg	car.brake()
	car.color = white	car.stop()

All cars have the same **properties**, but the property **values** differ from car to car.

All cars have the same **methods**, but the methods are performed **at different times**.

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

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

Object Properties

The **name:values** pairs in JavaScript objects are called **properties**:

Property	Property Value
firstName	John
lastName	Doe
age	50
eyeColor	blue

Accessing Object Properties

You can access object properties in two ways:

```
objectName.propertyName
```

or

```
objectName["propertyName"]
```

Método de um Objeto

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

```
name = person.fullName();
```

Construtor de um Objeto

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

```
var myFather = new Person("John", "Doe", 50, "blue");
var myMother = new Person("Sally", "Rally", 48, "green");
```

Adding a Property to an Object

Adding a new property to an existing object is easy:

Example

```
myFather.nationality = "English";
```

Adding a Method to an Object

Adding a new method to an existing object is easy:

Example

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

Adding a Method to a Constructor

Your constructor function can also define methods:

Example

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

Built-in JavaScript Constructors

JavaScript has built-in constructors for native objects:

Example

```
var x1 = new Object();     // A new Object object
var x2 = new String();     // A new String object
var x3 = new Number();     // A new Number object
var x4 = new Boolean();    // A new Boolean object
var x5 = new Array();      // A new Array object
var x6 = new RegExp();     // A new RegExp object
var x7 = new Function();   // A new Function object
var x8 = new Date();       // A new Date object
```

Prototype

The JavaScript `prototype` property also allows you to add new methods to objects constructors:

Example

```
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;
};
```

Class Definition

Use the keyword `class` to create a class, and always add the `constructor()` method.

The constructor method is called each time the class object is initialized.

Example

A simple class definition for a class named "Car":

```
class Car {
  constructor(brand) {
    this.carname = brand;
  }
}
```

Create a method named "present":

```
class Car {
  constructor(brand) {
    this.carname = brand;
  }
  present() {
    return "I have a " + this.carname;
  }
}

mycar = new Car("Ford");
document.getElementById("demo").innerHTML = mycar.present();
```

```
class Car {
  constructor(brand) {
    this.carname = brand;
  }
  present() {
    return 'I have a ' + this.carname;
  }
}

class Model extends Car {
  constructor(brand, mod) {
    super(brand);
    this.model = mod;
  }
  show() {
    return this.present() + ', it is a ' + this.model;
  }
}

mycar = new Model("Ford", "Mustang");
document.getElementById("demo").innerHTML = mycar.show();
```

Funções Javascript

Uma função pode ser definida em qualquer parte do código.

```
function f(arg1, arg2, ...) {
..........código.......
}
```

```
function sayHi(name) {
    alert("Hi, "+name)
}

sayHi('John')
```

Function Hoisting

```
myFunction(5);

function myFunction(y) {
  return y * y;
}
```

Closure (Clausura)

```
let userName = 'John';

function showMessage() {
  userName = "Bob"; // (1) changed the outer variable

  let message = 'Hello, ' + userName;
  alert(message);
}

alert( userName ); // John before the function call

showMessage();

alert( userName ); // Bob, the value was modified by the function
```

```
// Initiate counter
var counter = 0;

// Function to increment counter
function add() {
  counter += 1;
}

// Call add() 3 times
add();
add();
add();

// The counter should now be 3
```

```
// Initiate counter
var counter = 0;

// Function to increment counter
function add() {
  var counter = 0;
  counter += 1;
}

// Call add() 3 times
add();
add();
add();

//The counter should now be 3. But it is 0
```

```
function add() {
  var counter = 0;
  function plus() {counter += 1;}
  plus();
  return counter;
}
```

```
var add = (function () {
  var counter = 0;
  return function () {counter += 1; return counter}
})();

add();
add();
add();

// the counter is now 3
```

Uso de Funções facilita o entendimento

```
1   function showPrimes(n) {
2     nextPrime: for (let i = 2; i < n; i++) {
3
4       for (let j = 2; j < i; j++) {
5         if (i % j == 0) continue nextPrime;
6       }
7
8       alert( i ); // a prime
9     }
10  }
```

```
function showPrimes(n) {

  for (let i = 2; i < n; i++) {
    if (!isPrime(i)) continue;

    alert(i);  // a prime
  }
}

function isPrime(n) {
  for (let i = 2; i < n; i++) {
    if ( n % i == 0) return false;
  }
  return true;
}
```

Parâmetros de Funções

```
function myFunction(x, y) {
  if (y === undefined) {
    y = 0;
  }
}
```

```
function (a=1, b=1) {
  // function code
}
```

```javascript
x = findMax(1, 123, 500, 115, 44, 88);

function findMax() {
  var i;
  var max = -Infinity;
  for (i = 0; i < arguments.length; i++) {
    if (arguments[i] > max) {
      max = arguments[i];
    }
  }
  return max;
}
```

Function expressions

In JavaScript, a function is not a “magical language structure”, but a special kind of value.

The syntax that we used before is called a *Function Declaration*:

```
function sayHi() {
  alert( "Hello" );
}
```

There is another syntax for creating a function that is called a *Function Expression*.

It looks like this:

```
let sayHi = function() {
  alert( "Hello" );
};
```

```
function sayHi() {
  alert( "Hello" );
}

alert( sayHi ); // shows the function code
```

```
let sayHi = function() {
  alert( "Hello" );
};
```

```

let func = sayHi;     // (2) copy

func(); // Hello     // (3) run the copy (it works)!
sayHi(); // Hello    //     this still works too (why wouldn't it)
```

```
function ask(question, yes, no) {
  if (confirm(question)) yes()
  else no();
}

ask(
  "Do you agree?",
  function() { alert("You agreed."); },
  function() { alert("You canceled the execution."); }
);
```

Notação Flecha para Funções

```
1   let sum = (a, b) => a + b;
2
3   /* This arrow function is a shorter form of:
4
5   let sum = function(a, b) {
6     return a + b;
7   };
8   */
9
10  alert( sum(1, 2) ); // 3
```

```
let double = n => n * 2;
// roughly the same as: let double = function(n) { return n * 2 }

alert( double(3) ); // 6
```

```
let age = prompt("What is your age?", 18);

let welcome = (age < 18) ?
  () => alert('Hello') :
  () => alert("Greetings!");

welcome();
```

```
let sum = (a, b) => {  // the curly brace opens a multiline function
  let result = a + b;
  return result; // if we use curly braces, then we need an explicit "return"
};

alert( sum(1, 2) ); // 3
```

Mais sobre Javascript

https://www.w3schools.com/js/default.asp

http://javascript.info/