



Treinamento Web Avançando

JavaScript
jQuery

Treinamento de Web

JavaScript Parte III

- Oque é JavaScript
 - Exemplo
 - Como usar
 - Boas Práticas
 - Padrões
 - Características da linguagem
 - Orientação a Objetos com JS
 - Alguns Objetos
 - Eventos
 - Oque é bom saber
 - Tarefa II
-

Quiz da W3School

W3Schools JavaScript Quiz

Result:

23 of 25

92%

You can be proud of yourself!

Time Spent

4:16

Check your answers

Try again

Hoje, sem meme idiota



Porque estudar JavaScript?

- HTML define o conteúdo e estrutura da página.
 - CSS define o estilo
 - **JavaScript programa por trás da página.**
 - JavaScript tem NADA haver com JAVA
-
- `<script src="https://www.w3schools.com/js/myScript1.js"></script>`
 - `<script src="myScript1.js"></script>`
 - `<script> </script>`

Onde adicionar o JS ?

```
<html>  
  <head> <title>Example JS</title> </head>  
  <body>  
    ...  
    <script> JS AQUI </script>  
  </body>  
</html>
```

- // Comentário de linha
- /* Comentário de bloco */

Primeiro Código

- 1º Passo, criar página HTML
- 2º Aplicação Estilos
- 3º Aplicar JS

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

Mais exemple

```
document.getElementById("demo").style.fontSize = "25px";  
or  
document.getElementById('demo').style.fontSize = '25px';
```

```
document.getElementById("demo").style.display = "none";  
or  
document.getElementById('demo').style.display = 'none';
```


Características

- Ponto e vírgula no final de cada “sentença”
 - `a = 5; b = 6; c = a + b;`
- Espaços em branco, são ignorados
- Quebra de linhas e tamanho da linha
 - Quebra de linha não muda o código
 - recomenda 80 caracteres por linha (Best-pratics)
 - quebra deve ocorrer após um operador (Best-pratics)
- Chaves define os blocos de Código.

Syntax

- Atribuir variável
 - `var x, y, z;`
 - `y = 6;`
- Statement
 - Values, Operators, Expressions, Keywords, and Comments.
- Valores
 - Literal
 - `"Luiz Felipe"`
 - `10.50`
 - Variável
 - `var x ;`
 - `x = 6;`

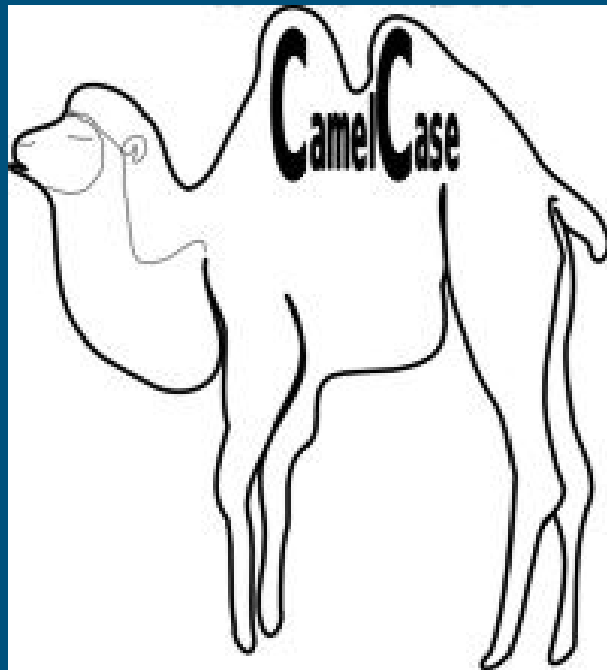
Operadores

- Operadores Aritméticos
- +
- -
- *
- /
- ()
- =
- "Luiz" + " " + "Felipe";
- $x * 10$

Como dar nome às variáveis

- first-name (não permitido)
- first_name (OK)

- Padrão CamelCase
- Upper Camel Case (Pascal Case)
 - FirstName
- Lower Camel Case
 - firstName
- JavaScript usa o Unicode como “tabela” caracteres



Operadores

Operator	Description
+	Addition
-	Subtraction
*	Multiplication
/	Division
%	Modulus
++	Increment
--	Decrement

Alguns “atalhos” de operações

Operator	Example	Same As
=	<code>x = y</code>	<code>x = y</code>
+=	<code>x += y</code>	<code>x = x + y</code>
-=	<code>x -= y</code>	<code>x = x - y</code>
*=	<code>x *= y</code>	<code>x = x * y</code>
/=	<code>x /= y</code>	<code>x = x / y</code>
%=	<code>x %= y</code>	<code>x = x % y</code>

Operadores de Comparação

Operator	Description
==	equal to
===	equal value and equal type
!=	not equal
!==	not equal value or not equal type
>	greater than
<	less than
>=	greater than or equal to
<=	less than or equal to
?	ternary operator

Comparadores Lógico

Operator	Description
<code>&&</code>	logical and
<code> </code>	logical or
<code>!</code>	logical not

Comparadores de bit ... não veremos

Palavras-Chaves

Keyword	Description
break	Terminates a switch or a loop
continue	Jumps out of a loop and starts at the top
debugger	Stops the execution of JavaScript, and calls (if available) the debugging function
do ... while	Executes a block of statements, and repeats the block, while a condition is true
for	Marks a block of statements to be executed, as long as a condition is true
function	Declares a function
if ... else	Marks a block of statements to be executed, depending on a condition
return	Exits a function
switch	Marks a block of statements to be executed, depending on different cases
try ... catch	Implements error handling to a block of statements
var	Declares a variable

Precedência de Operadores

- `()`
- membro ou `[]` `obj.name` ou `obj['name']`
- Chamada de função
- `new`
- Postfix Increment
- Prefix Increment ou `!` ou `typeof`
- `*` ou `/` ou `%` ou `**`
- `+` ou `-`
- `<<` ou `>>`
- `<` ou `>` ou `<=` ou `=>`
- `==` ou `===` ou `!=` ou `!==`
- `&&` ou `||`

Tipos de Dado

- Fracamente tipado
- undefined
- Number
- String
- Object
- Booleans
- Arrays
- function

```
typeof "John"           // Returns "string"  
typeof 3.14             // Returns "number"  
typeof true             // Returns "boolean"  
typeof false            // Returns "boolean"
```

```
typeof undefined        // undefined  
typeof null             // object  
null === undefined      // false  
null == undefined       // true
```

Como fazer chamada de funções

- `<button type="button" onclick="myFunction()">Try it</button>`

```
<script>
```

```
function myFunction() {
```

```
    document.getElementById("demo").innerHTML = "Paragraph changed.";
```

```
}
```

```
</script>
```

Função

- Função é um bloco de código.
- É executado quando alguém o “invoca”

```
function name(parameter1, parameter2, parameter3) {  
    code to be executed  
}
```

Condição

- If
- else
- else if
- switch
- break
- continue
- label

Laços

- for
- for/in
- while
- do/while

Orientação a Objetos por JavaScript

Scope

- 2 tipos
 - Global
 - Local

```
// code here can not use carName
```

```
function myFunction() {  
    var carName = "Volvo";
```

```
    // code here can use carName
```

```
}
```

```
var carName = "Volvo";
```

```
// code here can use carName
```

```
function myFunction() {
```

```
    // code here can use carName
```

```
}
```


- Global Automática ->
- Global no HTML
- Parâmetros de função são locais

```
myFunction();
```

```
// code here can use carName
```

```
function myFunction() {  
    carName = "Volvo";  
}
```

Conceito

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

Definição

- Simples valor é uma variável
 - Um objeto é como se fosse um conjunto de variáveis.
-
- `var car = {name: '500', year: 2017, color: "white"};`
-
- Método são funções do objeto

Métodos

- Acessar o atributos de 2 maneiras
 - `obj.attr`
 - `obj['attr']`
- Acessar os Métodos de 2 maneiras
 - `obj.method()`
 - `obj.method`

Declarando String, Números e Boolean

```
var x = new String();           // Declares x as a String object
var y = new Number();           // Declares y as a Number object
var z = new Boolean();           // Declares z as a Boolean object
```

Strings

Code	Outputs
\'	single quote
\"	double quote
\\	backslash

Code	Outputs
\b	Backspace
\r	Carriage Return
\f	Form Feed
\t	Horizontal Tabulator
\v	Vertical Tabulator

https://www.w3schools.com/jsref/jsref_obj_string.asp

Números

- parseInt
- parseFloat

Property	Description
<u>constructor</u>	Returns
<u>MAX_VALUE</u>	Returns
<u>MIN_VALUE</u>	Returns
<u>NEGATIVE_INFINITY</u>	Represent
<u>NaN</u>	Represent
<u>POSITIVE_INFINITY</u>	Represent
<u>prototype</u>	Allows y

Method
<u>isFinite()</u>
<u>isInteger()</u>
<u>isNaN()</u>
<u>isSafeInteger()</u>
<u>toExponential(x)</u>
<u>toFixed(x)</u>
<u>toPrecision(x)</u>
<u>toString()</u>
<u>valueOf()</u>

Math

- `Math.PI;` // 3,144592653589793
- `Math.round(4.7)` // 5
- `Math.pow(8, 2)` // 64
- `Math.sqrt(64)` // 8
- `Math.abs(-4.7)` // 4.7
- `Math.ceil(4.4)` // 5
- `Math.floor(4.7)` // 4
- `Math.min(0, 15, -30)` // -30
- `Math.random()` // random Number entre 0 e 1

```
Math.E      // returns Euler's number
Math.PI     // returns PI
Math.SQRT2  // returns the square root of 2
Math.SQRT1_2 // returns the square root of 1/2
Math.LN2    // returns the natural logarithm of 2
Math.LN10   // returns the natural logarithm of 10
Math.LOG2E  // returns base 2 logarithm of E
Math.LOG10E // returns base 10 logarithm of E
```


Dates

- Wed Aug 30 2017 22:12:32 GMT-0300 (-03) ou 1504141952401
- Iniciada em 1º Janeiro de 1970 00:00:00
- `new Date()`
- `Date.toString()`
- `Date.toUTCString()`
- Timezone: se não especificar usa a do Browser se não com GMT(Greenwich) e é convertida em CDT(Central US Daylight Time)

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

Formato de Datas

- ISO 8601
(YYYY-MM-DD)

Type	Example
ISO Date	"2015-03-25" (The International Standard)
Short Date	"03/25/2015"
Long Date	"Mar 25 2015" or "25 Mar 2015"
Full Date	"Wednesday March 25 2015"

- Z significa UTC ->
Podemos usar +HH:MM ou -HH:MM

```
var d = new Date("2015-03-25T12:00:00Z");
```

Funções de Data

Method	Description
getDate()	Get the day as a number (1-31)
getDay()	Get the weekday as a number (0-6)
getFullYear()	Get the four digit year (yyyy)
getHours()	Get the hour (0-23)
getMilliseconds()	Get the milliseconds (0-999)
getMinutes()	Get the minutes (0-59)
getMonth()	Get the month (0-11)
getSeconds()	Get the seconds (0-59)
getTime()	Get the time (milliseconds since January 1, 1970)

Vetores

- `var cars = ["Saab", "Volvo", "BMW"];`
- `var cars = new Array("Saab", "Volvo", "BMW");`

- `var name = cars[0];` //name têm valor de "Saab"
- `cars[0] = "Fiat";` //["Fiat", "Volvo", "BMW"];
- Arrays podem ser objetos
 - Arrays => Index numerados
 - Objetos => Index nomeados

Vetores Funções

- `Array.toString()`
- `Array.join(String)`
- `Array.pop()` `//Ultimo`
- `Array.push(String)`
- `Array.shift()` `//Primeiro`
- `Array.unshift(String)`
- `Array.length`
- `delete Array[index]`
- `Array.splice(where add, how many remove, valor pra add, valor pra add, ..)`
- `Array.splice()`
- `Array.concat(Array)`

Funções “especiais” de vetor

- `sort`
- `max.apply`
- `min.apply`

Vetores Boas-práticas

```
var points = new Array();           // Bad  
var points = [];                    // Good
```

```
var points = new Array(40, 100, 1, 5, 25, 10); // Bad  
var points = [40, 100, 1, 5, 25, 10];          // Good
```

- Uso de new, além de deixar o código completo de ler, pode causar um efeito inesperado.

Eventos



O que são ?

- HTML eventos são “**things**” que acontecem nos elementos HTML
- JavaScript pode “**reagir**” a estes eventos
- Evento pode ser algo que o Browser faz ou que o usuário faça
 - HTML é finalizado
 - HTML Input é modificado
 - HTML Button é clicado
- HTML permite tratar os eventos com JavaScript

With single quotes:

```
<element event='some JavaScript'>
```

With double quotes:

```
<element event="some JavaScript">
```

Exemplos

- `<button onclick="document.getElementById('demo').innerHTML = Date()">The time is?</button>`
- `<button onclick="displayDate()">The time is?</button>`

Alguns Eventos

Event	Description
onchange	An HTML element has been changed
onclick	The user clicks an HTML element
onmouseover	The user moves the mouse over an HTML element
onmouseout	The user moves the mouse away from an HTML element
onkeydown	The user pushes a keyboard key
onload	The browser has finished loading the page

Quando usamos JavaScript

- Things that should be done every time a page loads
- Things that should be done when the page is closed
- Action that should be performed when a user clicks a button
- Content that should be verified when a user inputs data
- And more ...

Many different methods can be used to let JavaScript work with events:

- HTML event attributes can execute JavaScript code directly
- HTML event attributes can call JavaScript functions
- You can assign your own event handler functions to HTML elements
- You can prevent events from being sent or being handled
- And more ...

Testar

- Datas
- Comparações
- Bitwise
- Strict Mode
- Style Guide
- Performance
- Palavras reservadas

Bom saber..

-
- Expressões Regulares
 - Tratamento de erros (try, catch, finally, throw)
 - console.log
 - debugger (palavra chave)
 - Style Guide
 - Nomear e declarar regras e convenções
 - Regras de Espaços brancos, indentação e comentários
 - Princípios e práticas de programação
 - Fazer código “readability”
 - Fazer código de fácil manutenção

Boas práticas

- Sempre declarar variáveis locais
- Declaração no topo
 - Código limpo
 - Único lugar para olhar as variáveis
 - Fácil identificar variáveis globais;
- Inicializar as variáveis
- Não use **new** Object()
- Use ===
- Use parameter defaults (undefined)
- Switch com “default”
- Evite eval()

Erros comuns

- Acidentalmente utilizar atribuição ao lugar de comparação
 - `if(x = 10)`
 - `if (x === 10)`
- Adição e concatenação
 - `x = 10 + 5;`
 - `x = 10 + '5';`
- Desentendimento de floats
- Quebrando String em JS
- Retornar um função
- Acessar Array com nome nós index
- Finalizar Definição com vírgula
- `undefined != null`

JSON

- É JSON é um formato para armazenar e transportar dados
- É JSON é utilizado quando dados tem que usado para enviar dados para servidor.
- JSON = JavaScript Object Notation
- É leve e independe da linguagem e é auto descritivo
- Syntax
 - Par de nome/valor
 - Dados são separados por vírgula
 - Chaves separam os objetos
 - Colchetes delimitam os arrays.

Como ele é

```
"employees":[
  {"firstName":"John", "lastName":"Doe"},
  {"firstName":"Anna", "lastName":"Smith"},
  {"firstName":"Peter", "lastName":"Jones"}
]
```

```
var text = '{ "employees" : [' +
  '{ "firstName":"John" , "lastName":"Doe" },' +
  '{ "firstName":"Anna" , "lastName":"Smith" },' +
  '{ "firstName":"Peter" , "lastName":"Jones" } ]}';
```

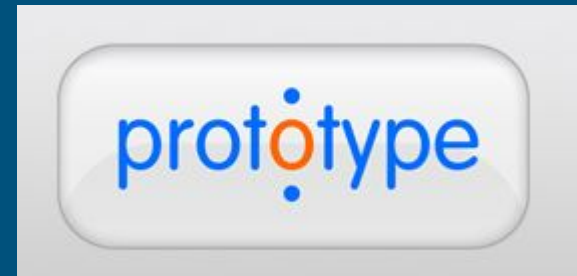
```
var obj = JSON.parse(text);
```

```
<p id="demo"></p>
```

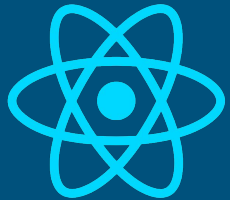
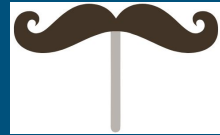
```
<script>
```

```
document.getElementById("demo").innerHTML =
obj.employees[1].firstName + " " + obj.employees[1].lastName;
</script>
```

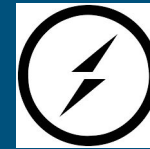
Algumas maravilhas em JS



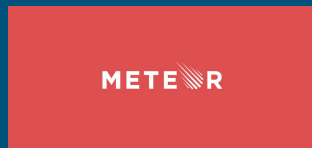
{dust}



Stapes.js



ember



LET'S GO WORK

