Desenvolvimento de Software para WEB

Aula 5 - Javascript

Professor: Anderson Almada

Introdução

- **JavaScript** ou **JS** é uma linguagem de programação interpretada.
- Serve para provê interação ao usuário com a aplicação web
- Programação client-side em navegadores web.
- Também utilizada do lado do servidor através de ambientes como o node.js.

Elemento <script> em <head>

```
<!DOCTYPE html>
<html>
<head>
   <meta charset="utf-8">
   <title>Aula de JS</title>
   <script>
       alert("Olá, Mundo!");
   </script>
</head>

<body>
   <h1>JavaScript</h1>
   <h2>Linguagem de programação</h2>
</body>

</html>
```

Js externo

```html
<!DOCTYPE html>
<html>
<head>
    <meta charset="utf-8">
    <title>Aula de JS</title>
    <script src="js/hello.js"></script>
</head>

<body>
    <h1>JavaScript</h1>
    <h2>Linguagem de programação</h2>
</body>

</html>
```

js/hello.js

```
alert("Olá, Mundo!");
```

Console

- **Chrome**
  - Control + Shift + C
- **Firefox**
  - Control + Shift + K

- JS
  - console.log("hello, world")

Comentários

- `// this is a comment`

- ```
  /* this is a multi-line
  or block comment */
  ```
  ```

Tipos de dados

- Number
- String
- Boolean
- Object
- Null
- Undefined

Declaração de variável

- **var** nome_da_variavel;
- **Number**
  - var number1 = 2;
  - number1 = 3 + 5.3
  - number1 = 28 % 6

- **Objeto Math**
  - Math.sin(3.5);
  - d = Math.PI * r * r;

parseInt

- Conversão de string para número (defina a base):
  - parseInt("123", 10)
    - 123
  - parseInt("010", 10)
    - 10
  - parseInt("11", 2)
    - 3

parseFloat

- Conversão de string para número flutuante
    - parseFloat("123")
        - 123

NaN (Not a Number) e Infinity

- parseInt("hello", 10)
  - NaN
- NaN + 5
  - NaN
- isNaN(NaN)
  - true
- 1 / 0
  - Infinity

NaN (Not a Number) e Infinity

- parseInt("hello", 10)
  - NaN
- NaN + 5
  - NaN
- isNaN(NaN)
  - true
- -1 / 0
  - -Infinity

String

- "Hello".**length**
  - 5
- “hello".**charAt**(0)
  - h
- "hello, world".**replace**("hello", "goodbye")
  - goodbye, world
- "hello".**toUpperCase**()
  - HELLO

String

String	`length`	Returns the number of characters in a string
	`concat( )`	Joins two or more strings
	`indexOf( )`	Returns the position of the first occurrence of a specified string value in a string
	`lastIndexOf( )`	Returns the position of the last occurrence of a specified string value, searching backward from the specified position in a string
	`match( )`	Searches for a specified string value in a string
	`replace( )`	Replaces some characters with others in a string
	`slice( )`	Extracts a part of a string and returns the extracted part in a new string
	`split( )`	Splits a string into an array of strings
	`substring( )`	Extracts the characters in a string between two specified indexes
	`toLowerCase( )`	Displays a string in lowercase letters
	`toUpperCase( )`	Displays a string in uppercase letters

Boolean

- true ou false
- **false**: 0, "", NaN, null, undefined
- O restante é **true**
- Operações: &&, || e !

Conversão de tipos

- var textoInteiro = "10";
- var inteiro = parseInt(textoInteiro);
- var textoFloat = "10.22";
- var float = parseFloat(textoFloat);
- var milNumber = 1000;
- var milString = milNumber.toFixed(2); // recebe o retorno da função
- console.log(milString); // imprime a string "1000.00"

Operadores

- Numéricos: +, -, *, / e %
- Atribuição: =, +=, -=, *=, /=, %=
- Incremento/decremento: a++, ++a, b--, --b
- Concatenação de string
  - "hello" + " world"
    - hello world
- Coerção de tipos:
  - "3" + 4 + 5 -> 345
  - 3 + 4 + "5" -> ?

Operadores

- Numéricos: +, -, *, / e %
- Atribuição: =, +=, -=, *=, /=, %=
- Incremento/decremento: a++, ++a, b--, --b
- Concatenação de string
  - "hello" + " world"
    - hello world
- Coerção de tipos:
  - "3" + 4 + 5 -> 345
  - 3 + 4 + "5" -> 75

Comparação

- Para números e strings: <, >, <= e >=
- Igualdade: == e !=
  - Faz conversão de tipos se necessário
    - "dog" == "dog" -> true
    - 1 == true -> true
- Identidade: === e !==
  - Não faz conversão de tipos
  - Se forem de tipos diferentes, o resultado será falso
    - 1 === true -> false
    - true === true -> true

typeof

number	'number'
string	'string'
boolean	'boolean'
function	'function'
object	'object'
array	**'object'**
null	**'object'**
undefined	'undefined'

Estrutura de controle - if

```
var name = "kittens";

if (name == "puppies") {
    name += "!";
} else if (name == "kittens") {
    name += "!!";
} else {
    name = "!" + name;
}

name == "kittens!!"
```

Estrutura de controle - while e do-while

```
while (true) {
    // an infinite loop!
}

do {
    // an infinite loop!
} while (true)
```

Estrutura de controle - for

```
for (var i = 0; i < 5; i++) {
    // Will execute 5 times
}
```

Estrutura de controle - switch

```
switch(action) {
    case 'draw':
        // draw
        break;
    case 'eat':
        // eat
        break;
    default:
        // default
        break;
}
```

Operador ternário

```
var allowed = (age > 18) ? "yes" : "no";
```

Exception

```
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
}
```

Objetos

- **Simples pares key-value**

- **Criação de objetos:**
  - var obj = new Object();

- **Manipulação de objetos**
  - obj.name = "Simon"

    Ou
  - obj["name"] = "Simon";

Iteração de um objeto

- **Pode-se iterar pelas chaves de um objeto:**

```
var obj = { 'name': 'Simon', 'age': 25 };

for (attr in obj) {
   console.log (attr + ' = ' + obj[attr]);
}
```

Arrays

- Tipo especial de objeto: as chaves são números e não strings.
- Sintaxe [ ]:

```
var a = new Array();
a[0] = "dog";
a[1] = "cat";
a[2] = "hen";
a.length
    3
```

Arrays

```
var a = ["dog", "cat", "hen"];
var palavras = ["UFC", "Ensino"];
palavras.push("Inovação");
// adiciona a string "Inovação"
```

Arrays - Cuidado !!

```
var a = ["dog", "cat", "hen"];
a[100] = "fox";
a.length
    101

typeof(a[90])]
    undefined
```

Forma seguro:

```
    a[a.length] = item;
```

Iteração em um array

```
for (var i = 0; i < a.length; i++) {
   // Do something with a[i]
   }

for (var item in a) {
   // Do something with item
}
```

Funções

```
function add(x, y) {
   var total = x + y;
   return total;
   }
```

- **Passagem de parâmetros:**

  add(2, 3) -> 5

Funções - Objeto arguments

```
function add() {
    var sum = 0;
    for (var i = 0, j = arguments.length; i < j; i++) {
        sum += arguments[i];
    }
    return sum;
}
```

add(2, 3,4) -> 9

Funções anônimas

```
var somaDoisNumeros = function (numero1, numero2) {
    return numero1 + numero2;
};

somaDoisNumeros(10, 20);
```

Funções temporais

```
var somaDoisNumeros = function (numero1, numero2) {
    result = numero1 + numero2;
    alert(result);
};

// inicia
var timer = setInterval(somaDoisNumeros, 1000, 1, 2);
// remove
clearInterval(timer);
```

Classes?

- JavaScript não possui classes
- Funcionalidade semelhante é obtida através de **protótipos de objetos**.
- JavaScript usa funções como classes.
- A palavra reservada **new** cria um novo objeto e o atribui a palavra chave **this** de dentro do escopo da função invocada.
- Pode-se então adicionar atributos a esse objeto.

Construtores

```
function Person(first, last) {
    this.first = first;
    this.last = last;

    this.fullName = function () {
        return this.first + ' ' + this.last;
    };
    this.fullNameReversed = function () {
        return this.last + ', ' + this.first;
    };
}

var s = new Person("Lemmy", "Kilmister");
```

Protótipo

- Qualquer atributo ou função adicionado ao protótipo de uma dessas funções ficará disponível em qualquer objeto do tipo gerado por elas.

```
String.prototype.paraNumero = function () {
    if (this == "um") {
        return 1;
    }
}

console.log("um".paraNumero());
```

Protótipo

```
var Pessoa = function (nome, email) {
    this.nome = nome;
    // verifica se o e-mail foi preenchido
    if (email) {
        this.email = email;
    }
}
Pessoa.prototype.email = "contato@ufc.br"
var ricardo = new Pessoa("Ricardo");
console.log(ricardo.email); // contato@ufc.br
var joao = new Pessoa("Joao da Silva", "joao@da.silva");
console.log(joao.email); // joao@da.silva
```

Protótipo

```
var Pessoa = function (nome, email) {
    this.nome = nome;
    // verifica se o e-mail foi preenchido
    if (email) {
        this.email = email;
    }
};
Pessoa.prototype.fala = function () {
    console.log("Olá, meu nome é " + this.nome + " e meu email é " +
this.email);
};
Pessoa.prototype.anda = function () {
    console.log("Estou andando");
};
```

Closures

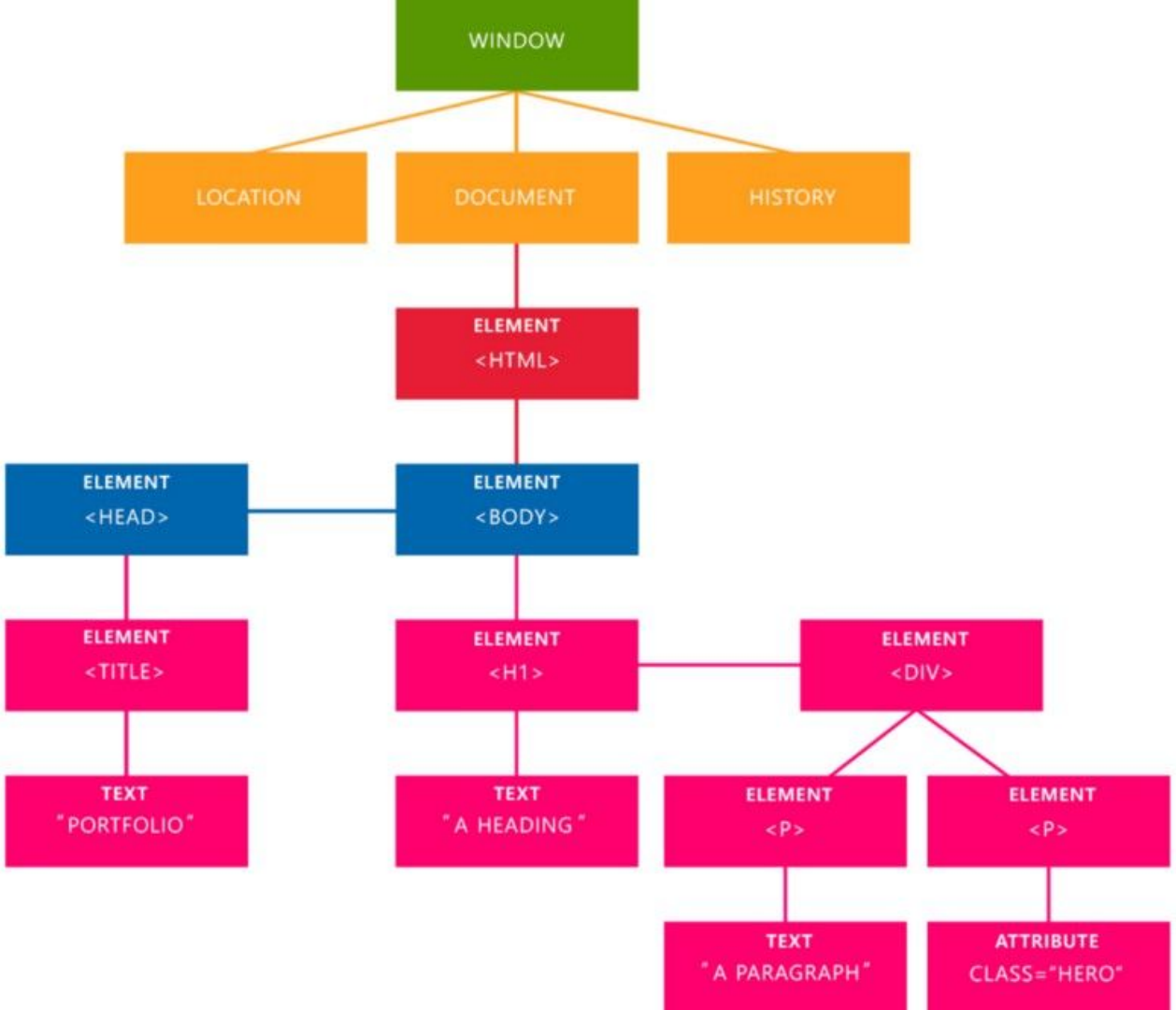
```
function makeAdder(a) {
    return function (b) {
        return a + b;
    };
}
var x = makeAdder(5);
var y = makeAdder(20);
x(6); // 11
y(7); // 27
```

DOM

- **D**ocument **O**bject **M**odel é uma interface que representa como os documentos HTML e XML são lidos pelo seu *browser*.

- Após o *browser* ler o HTML, ele cria um objeto que faz uma representação estruturada do seu documento e define meios de como essa estrutura pode ser acessada.

- O JavaScript server para acessar e manipular o DOM


WINDOW
LOCATION
DOCUMENT
HISTORY
ELEMENT
<HTML>
ELEMENT
<HEAD>
ELEMENT
<BODY>
ELEMENT
<TITLE>
ELEMENT
<H1>
ELEMENT
<DIV>
TEXT
"PORTFOLIO"
TEXT
"A HEADING"
ELEMENT
<P>
ELEMENT
<P>
TEXT
"A PARAGRAPH"
ATTRIBUTE
CLASS="HERO"

getElementById

JS

```
function ok() {
    var myStart = document.getElementById('name');
    alert(myStart.value);
}
```

HTML

```
<button onclick="ok()">OK</button>
<input id="name" type="text">
```

getElementsByClassName

JS

```
function ok() {
    var myContainer = document.getElementsByClassName('container');
    alert(myContainer['idade'].value);
}
```

HTML

```
<input class="container" type="text" name="idade">
<input class="container" type="text" name="peso">
```

getElementsByTagName

JS

```
function ok() {
    var buttons = document.getElementsByTagName('button');
    alert(buttons[0].value);
}
```

HTML

```
<button onclick="ok()" value="ok">OK</button>
```

querySelector

JS

```
function ok() {
    var resetButton = document.querySelector('#reset');
    alert(resetButton.value);
}
```

HTML

```
<button id="reset" onclick="ok()" value="ok">OK</button>
```

querySelector e querySelectorAll

JS

```
function ok() {
    var resetButton = document.querySelector('#reset');
    alert(resetButton[0].value);
}
```

HTML

```
<button id="reset" onclick="ok()" value="ok">OK</button>
<button id="reset" onclick="ok()" value="ok">OK</button>
```

querySelector - textContent

JS

```
function ok() {
var titulo = document.querySelector("#titulo");
   titulo.textContent = "Agora o texto do elemento mudou!";
}
```

HTML

```
<p id="titulo">OK</p>
<a href="#" onclick="ok()">OK</a>
```

querySelector - innerHTML

JS

```
function myFunction() {
   document.getElementById("demo").innerHTML = "<div style='color:
red'>OK Google</div>";
}
```

HTML

```
<p id="demo" onclick="myFunction()">Click me to change my HTML
content (innerHTML).</p>
```

addEventListener - click

JS

```
function ok() {
    var myStart = document.getElementById('name');

    myStart.addEventListener('click', function(event) {
        alert(myStart.value);
    });
}
```

HTML

```
<input type="text" id="name" value="ok">
<a href="#" onclick="ok()">OK</a>
```

addEventListener - select

JS

```
function ok() {
    var myStart = document.getElementById('name');

    myStart.addEventListener('select', function(event) {
        alert(myStart.value);
    });
}
```

HTML

```
<input type="text" id="name" value="ok">
<a href="#" onclick="ok()">OK</a>
```

Links importantes

- **Desenvolvimento Web com HTML, CSS e JavaScript**. Disponível em: https://www.caelum.com.br/apostila-html-css-javascript/
- https://www.w3schools.com/js/
- https://www.w3schools.com/js/js_htmldom.asp
- https://tableless.com.br/entendendo-o-dom-document-object-model/

Dúvidas??

E-mail: almada@crateus.ufc.br