Tecnologia para Front-end II

Prof. M.Sc. Gláucio Bezerra Rocha

História do Javascript

- **Tornar a Web mais interativa;**
- **Desenvolvida por Brendan Eich;**
- **Baseada no ECMAScript;**
- **Inicialmente chamada de Mocha, depois oficialmente batizada de LiveScript;**
- **Em setembro de 1995 foi renomeada para JavaScript;**
- **Versão inicial implementada no Netscape Navigator;**
- **Back-end e Front-end;**

Ambiente - Configuração

- **Visual Studio Code;**
- **SublimeText;**
- **Navegadores;**
- **Noje.js**

Fundamentos de Javascript

- **Características Gerais**

  É uma linguagem de programação interpretada;
  Possui tipagem fraca e dinâmica.

- **Sintaxe Básica**

  Toda instrução deve ser finalizada com “;” (ponto e vírgula);
  Valores são divididos em LITERAIS e VARIÁVEIS.

  Exemplo de valores LITERAIS: “joão e maria”, 10.5, 19, true;
  Exemplo de valores variáveis: var nome = “joão e maria”;

Fundamentos de Javascript

- **Variáveis no Javascript**
    - São declaradas usando a palavra reservada **var (escopo local e global)** e **let (escopo de bloco)** ;
    - Podem ser do tipo: undefined, null, number, string, boolean, array, Object, function;

Constantes (const)

- **Comentários**
    - // e /**/
- Operador **typeof**
- **Operador de Atribuição (=)**
- **Operadores Aritméticos (%, ++, --, +, -, /, *, **)**
- **Operadores de Comparação (==, !=, ===, !==, >, <, >=, <=)**
- **Operadores Lógicos (&&, ||, !)**

Controles de Fluxo

- **IF (se)**

```
if (teste lógico) {
//conjunto de instruções a serem executadas caso o teste lógico seja verdadeiro
}
```

Exemplo:

```
var nome = “João”;
if (nome == “João”) {
        console.log(“Olá”, nome);
}
```

Controles de Fluxo

- **ELSE (senão)**

```
        if (teste lógico) {
        //conjunto de instruções a serem executadas caso o teste lógico seja verdadeiro
} else {
        //conjunto de instruções a serem executadas caso o teste lógico seja falso
}
```

Exemplo:

```
        var nome = "João";
        if (nome == "João") {
                console.log("Olá", nome);
        } else {
                console.log("Pessoa não identificada");
}
```

Controles de Fluxo

- **ELSE IF (senão se)**

```
        if (teste lógico) {
        //conjunto de instruções a serem executadas caso o teste lógico seja verdadeiro
} else if (teste lógico 2) {
        //conjunto de instruções a serem executadas caso o teste lógico 2 seja verdadeiro
} else {
        //conjunto de instruções a serem executadas caso o teste lógico 2 seja falso
}
```

Exemplo:

```
        var nome = "João";
        if (nome == "João") {
                console.log("Olá", nome);
        } else if (nome == "Pedro") {
                console.log("Olá", nome);
} else {
         console.log("Pessoa não identificada");
}
```

Controles de Fluxo

- **SWITCH**

```
switch (valor a ser testado) {
        case “valor 1”:
                //instruções a serem executadas
                break;
        case “valor 2”:
                //instruções a serem executadas
                break;
        default:
                //instruções a serem executadas
                break;
}
```

Controles de Fluxo

- **SWITCH**

Exemplo:

```
var nome = “João”;
switch (nome) {
        case “João”:
                console.log(“Olá João”);
                break;
        case “Pedro”:
                 console.log(“Olá Pedro”);
                break;
        default:
                 console.log(“Pessoa não identificada”);
                break;
}
```

Controles de Repetição

- **FOR**

```
for (variável de inicialização; teste lógico de continuidade; incremento da variável) {

    //conjunto de instruções a serem executadas
}
```

Exemplo:

```
for (var i = 0; i<= 10; i++) {
    console.log("Número", i);
}
```

Controles de Repetição

- **FOR IN**

```
for (variável que receberá o valor IN array de valor) {

        //conjunto de instruções a serem executadas
}
```

Exemplo:

```
var valores = [1,2,3,4,5]
for (var numero in valores) {
        console.log("Número", numero);
}
```

Controles de Repetição

- **WHILE**

```
while (condição lógica) {

        //conjunto de instruções a serem executadas
}
```

Exemplo:

```
var teste = true;
var cont = 0;
while (teste == true) {
        cont++;
        console.log("Olá turma", cont);
        if (cont >= 10) teste = false;
}
```

Controles de Repetição

- **DO WHILE**

```
do {

        //conjunto de instruções a serem executadas

} while (condição lógica);
```

Exemplo:

```
var teste = true;
var cont = 0;
do {
        cont++;
        console.log("Olá turma", cont);
        if (cont >= 10) teste = false;
} while (teste == true);
```

Atividade 01

Obtenha dados da altura e o sexo (M ou F) de 15 pessoas e apresente os seguintes resultados:

- A maior e a menor altura do grupo;
- A média de altura dos homens;
- O número de mulheres.

Fundamentos de Javascript

- **Hoisting**

  - Comportamento padrão onde o Javascript "***içа***" todas as declarações de variáveis (**var**) e funções para o topo do escopo (local ou global);

  - ATENÇÃO: as inicializações não são "***içadadas***", apenas as declarações.

Array em Javascript

- **Características Gerais:**

  É um tipo de variável que contem um ou mais valor;
  É um Object “especial”; (typeof array)
  São definidos utilizando colchetes [];
  Não necessita delimitar o tamanho (o limite é a memória);
  Os índices de um array inicia na posição 0 (zero);
  Comporta diversos tipos de dados.

- **Definindo um Array**

```
var meuArray = [‘Aluno 1’, ‘Aluno 2’];
meuArray[5] = “Aluno 6”;
```

- **Acessando um elemento do Array**

```
console.log(meuArray[1]);
```

Array em Javascript

- **Funções de um Array:**

  meuArray.length: informa o tamanho do array;
  meuArray.push: adiciona elemento no final de um array;
  meuArray.unshift: adiciona elemento no ínicio de um array;
  meuArray.pop: remove o último elemento de um array e o retorna;
  meuArray.shift: remove o primeiro elemento de um array e o retorna;
  meuArray.toString(): retorna os elementos do array no formato string separados por “,” (vírgula);
  meuArray.join(): retorna os elementos do array no formato string separados por valor configurado;
  outros: https://www.w3schools.com/js/js_array_methods.asp

- **Atenção: usar Array ou Object?**

```
> var animalLeao = ['Jonh', 'Mamífero', '4', '23/10/2005'];
```

```
> var animal = {nome: 'Jonh', tipo: 'Mamífero', quantPatas: 4, dataNascimento: '23/10/2005'}
```

Array em Javascript

- **Atenção: usar Array ou Object?**

```
> var animalLeao = ['Jonh', 'Mamífero', '4', '23/10/2005'];
```

```
> var animal = {nome: 'Jonh', tipo: 'Mamífero', quantPatas: 4, dataNascimento: '23/10/2005'}
```

- **Iterando um Array:**

for: laço de repetição já estudado anteriormente;

```
> for (int i = 0; meuArray.length, i++) { console.log(meuArray[i]);}
```

forEach();

```
> meuArray.forEach(function(elemento){console.log(elemento);});
```

Funções em Javascript

- **Definição**

  Conjunto de instruções agrupadas com o objetivo específico;
  Uma Function é um tipo de dado;
  Reaproveitamento de código.

- **Function Declaration**

  function nome(a,b,c) { return a+b+c; }

- **Function Expression**

  Var somar = function(a,b,c) { return a+b+c; }

- **Observação:** não é obrigatório o uso do ***return*.**

Funções em Javascript

- **Objeto Arguments**

  Objeto que contêm todos os parâmetros passados a uma função;
  Os parâmetros podem ser acessados através dos índices;
  Os parâmetros podem ter seus valores alterados;
  Possui a propriedade ***length*** que informa a quantidade de parâmetros.

Objetos

- **É um container de dados que armazena uma estrutura complexa de dados;**
- **É um tipo de dados do Javascript;**
- **É definido por meio de chaves {};**
- **Sua estrutura é formada de Chave x Valor;**

Exemplo:

```
var aluno = {
        nome: “João Marcos da Silva Filho”,
        idade: 26,
        nacionalidade: “Brasileiro”,
        menorIdade: false
}
```

```
var aluno = new Object();

aluno.nome= “João Marcos da Silva Filho”;
aluno.idade= 26;
aluno.nacionalidade= “Brasileiro”;
aluno.menoridade= false;
```

Objetos

- **Tipo de dados das propriedades: number, string, boolean, function, objetos etc**

Exemplo:

```
var aluno = {

        nome: “João Marcos da Silva Filho”,
        idade: 26,
        nacionalidade: “Brasileiro”,
        menorIdade: false,
        filiacao: [“João Marcos da Silva”, “Maria Costa da Silva”],
        endereco: {

                rua: “Rua das palmeiras”,
                bairro: “Manaira”,
                cidade: “João Pessoa

            }
        estaRegular: function() { return true; }

}
```

Objetos

- **Lendo valor de uma propriedade:**

```
var aluno = {
        nome: “João Marcos da Silva Filho”,
        idade: 26,
        nacionalidade: “Brasileiro”,
        menorIdade: false
}

aluno.nome;    OU    aluno[“nome”]
```

ATENÇÃO: Se a propriedade acessada não existir, será retornado um valor UNDEFINED.

Objetos

- **Atribuindo valor a uma propriedade:**

```
var aluno = {
        nome: “João Marcos da Silva Filho”,
        idade: 26,
        nacionalidade: “Brasileiro”,
        menorIdade: false
}

aluno.nome = “Pedro Henrique Albuquerque”;

                OU

aluno[“nome”] = “Pedro Henrique Albuquerque”;
```

Objetos

- **Adicionando uma propriedade:**

```
var aluno = {
        nome: “João Marcos da Silva Filho”,
        idade: 26,
        nacionalidade: “Brasileiro”,
        menorIdade: false
}

aluno.curso = “Sistema para Internet”;

                OU

aluno[“curso”] = “Sistema para Internet”;
```

Objetos

- **Removendo uma propriedade:**

```
var aluno = {
        nome: “João Marcos da Silva Filho”,
        idade: 26,
        nacionalidade: “Brasileiro”,
        menorIdade: false
}

delete aluno.menorIdade;
        OU
delete aluno[“menoridade”]
```

Objetos

- **Lendo e atribuindo valor uma propriedade complexa:**

```
var aluno = {
        nome: “João Marcos da Silva Filho”,
        filiacao: [“João Marcos da Silva”, “Maria Costa da Silva”],
        endereco: {
                rua: “Rua das palmeiras”,
                bairro: “Manaira”,
            }
}
```

Lendo propriedade OBJETO: aluno.endereco.rua; OU aluno[“endereco”][“rua”]
Atribuindo propriedade OBJETO: aluno.endereco.rua = “Rua do sertão”; OU aluno[“endereco”][“rua”] = “Rua do sertão”;

Lendo propriedade ARRAY: aluno.filiação[0];
Atribuindo propriedade ARRAY: aluno.filiação[1] = “João Marcos Pai”;

Objetos

- **Iterando as propriedades do objetos:**

```
var aluno = {

        nome: “João Marcos da Silva Filho”,
        filiacao: [“João Marcos da Silva”, “Maria Costa da Silva”],
        endereco: {

                rua: “Rua das palmeiras”,
                bairro: “Manaira”,
            }
}


for (var prop in aluno) {

        console.log(aluno[prop]);

}
```

Objetos

- **Comparando objetos: tem que definir um critério**

```
var aluno = {
        nome: “João Marcos da Silva Filho”,
        filiacao: [“João Marcos da Silva”, “Maria Costa da Silva”],
        equals: function(obj) {
                if (this.nome == obj.nome)
                        return true;
                else
                        return false;
        }
}

var alunoB = {
        nome: “João Marcos da Silva Filho”,
        filiacao: [“João Marcos da Silva”, “Maria Costa da Silva”],
        equals: function(obj) {
                if (this.nome == obj.nome)
                        return true;
                else
                        return false;
        }
}

console.log(aluno.equals(alunoB));
```

DOM

- **Definição**

  DOM (Document Object Model) é uma interface de programação para documentos HTML.

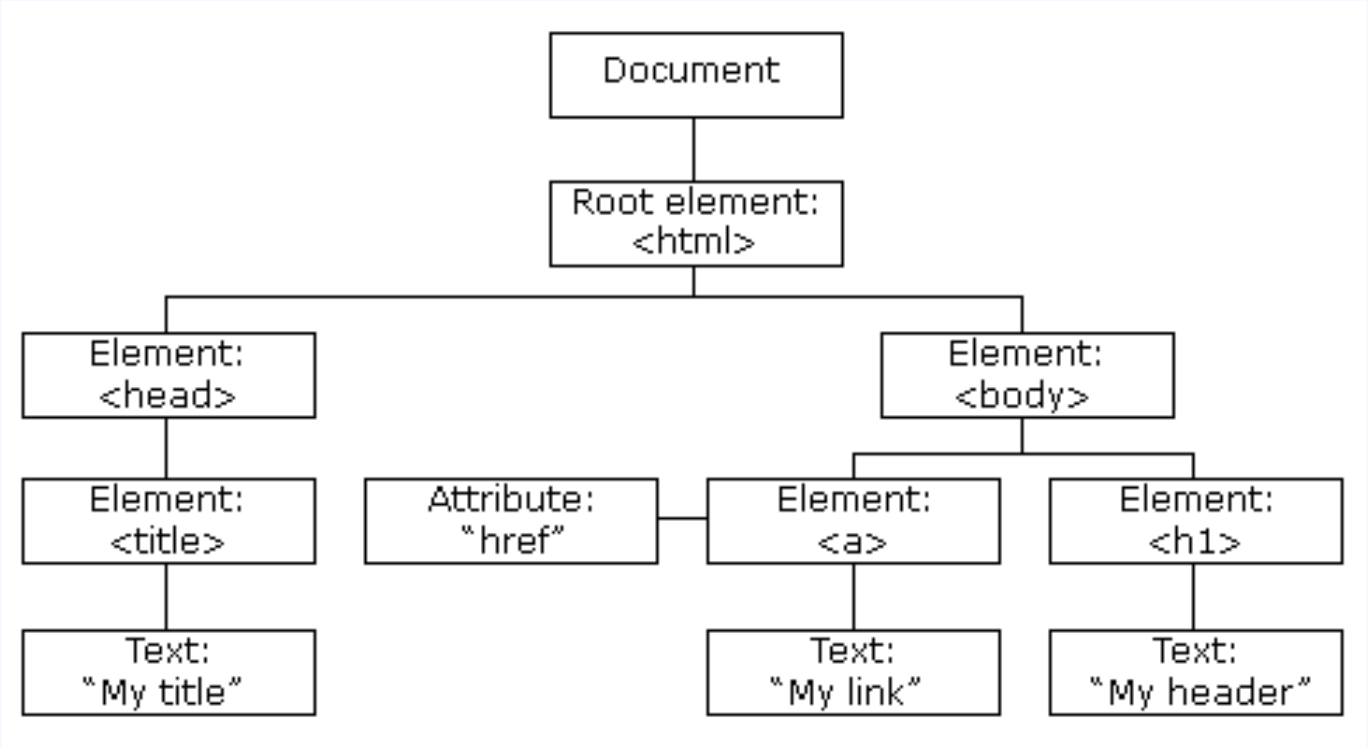
  Representa, em memória, a página HTML, com seus conjuntos de objetos que podem ser manipulados por meio da linguagem Javascript.

  Com o DOM é possível que os desenvolvedores modifiquem dinamicamente o conteúdo, estilo e comportamento da página, proporcionando um ambiente interativo para o usuário.

Document Object Model - DOM

- **Quando uma página é carregada o navegador cria o DOM**

  O DOM é construído como uma árvore de objetos



Document Object Model - DOM

“Todos os elementos HTML existentes são transformados em Objetos Javascript”

HTML

```
<div id=“principal” class=“cor_azul”>

    <img src=“bola.jpg” />
    <p>O Brasil foi desclassificado da copa do mundo pela seleção ...</p>

</div>
```

DOM

```
{
   tipo: “div”
   id: “principal”,
   class: “cor_azul”,
   objects: [
      {tipo: “img’, src: “bola.jpg”},
      {tipo: “p”, conteudo: “O brasil foi desclassificado ...”}
                ]
}
```

Document Object Model - DOM

- **Com o DOM é possível:**
  - Alterar todos os elementos da página;
  - Alterar todos os atributos da página;
  - Alterar todo o estilo CSS da página;
  - Remover elementos e atributos existentes;
  - Adicionar novos elementos e atributos;
  - Reagir a eventos HTML existentes na página;

Document Object Model - DOM

- **Objeto Window**

  Representa a janela de exibição da página HTML.

  Contem todos os objetos, funções e variáveis Javascript globais.

  Principais objetos: History, Location e Document.

  Principais atributos e métodos:

  * window.innerHeight, window.innerWidth, window.confirm(), window.alert(), windows.prompt(), window.open(), window.close(), setItnterval(), clearInterval().

- **Objeto History**

  Objeto que contem lista com todas as Urls visitadas.

  Principais atributos e métodos: length, go(), back(), forward().

- **Objeto Location**

  Objeto que contem informações sobre a URL atual.

  Principais atributos e métodos: hash, hostname, href, pathname, reload(), replace(), search.

- **Objeto Navigator**

  Objeto que contem todas as informações sobre o browser.

  Principais atributos e métodos: appCodeName, appName, appVersion, geoLocation, javaEnabled().

- **Objeto Document**

  Objeto que representa todo o conteúdo HTML.

Document Object Model - DOM

- **Objeto Document**

  Principais métodos: getElementById, getElementsByClassName, getElementsByName, getElementsByTagName e querySeletor;

- **Eventos em JavaScript**

  São ações que podem ser disparados por elementos HTML: click de botões, carregamento de página, movimentos de mouse, pressionamento de teclas etc.

  Exemplos:

  "onclick": dispara quando um elemento é clicado;

  "onload": dispara quando uma página é carregada;

  "onmouseover": dispara quando o cursor do mouse passa sobre um elemento;

  "onkeydown": dispara quando uma tecla é pressionada;

  "onsubmit": dispara quando um formulário é submetido.

  "onblur“:

  "onfocus“:

  "onchange“:

  "ondblclick“:

Document Object Model - DOM

- **Usando jQuery**

  Fazer o download do jQuery:

  Importar o jQuery para dentro do projeto:

```
<script src="js/jquery-3.6.0.min.js"></script>
```

- **Realizando a consulta**

```
var consulta = $.ajax({
      url: urlConsulta,
      method: 'GET',
   });
resposta = consulta.done(function(data) {
      aplicarDados(data);   });
```

Document Object Model - DOM

- **Realizando a Post**

```
var post = $.ajax({
        url: urlPost,
        method: 'POST',
        dataType: 'json',
        data: objJSon
    });
resposta = post.done(function(data) {
        aplicarDados(data);    });
```