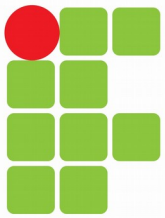


INSTITUTO FEDERAL DE
EDUCAÇÃO, CIÊNCIA E TECNOLOGIA
SUL-RIO-GRANDENSE

Desenvolvimento de Sites para Web

3º Semestre – Aula - 04

Prof. Gill Velleda Gonzales

Agosto, 2019

JavaScript

- **Execução, Sintaxe Básica**
- **Tipos de Dados**
- **Functions e Arrow Functions**
- **Arrays**
  - **Métodos de manipulação e iteração**
- **Objetos**
  - **Formas de Iterar**
- **API DOM**
- **Eventos**
- **Formulários**
- **Referências**

JavaScript

- **O QUE É JAVASCRIPT:**
- Linguagem interpretada criada para rodar no navegador web.
- Não é JAVA, o nome semelhante é apenas uma questão de marketing.
- Criada para o Netscape 2.0 por Bredan Eich
- Evoluiu de acordo com a evolução da Web, a cada nova versão novos recursos são adicionados ou padrões anteriores alterados.
- Padronizada pela ECMA International pela especificação ECMA-262 e ISO/IEC 16262.
- Já esta na sua versão ES8, mas as principais modificações foram feitas na versão ES6.

JavaScript

- **EXECUTAR UM PROGRAMA EM JS:**
- Historicamente executado (interpretado) em navegadores através da tag **<script>** do documento HTML;
- Embarcado no documento HTML dentro das tags ***script***:

  <script> var mensagem = 'Olá Mundo!'; alert(**mensagem**) **</script>**
- Através de um arquivo externo com a extensão .js.
  - **<script src** = 'caminho/do/arquivo.js'**> </script>**
- Em um interpretador como o ***nodejs*** através do terminal ou prompt windows.

  >node nomedoarquivo.js
  - Instruções para instalação de ***npm*** e ***nodejs*** :
    - Em ambiente Linux: https://www.edivaldobrito.com.br/node-js-no-linux/
    - Em Windows: https://dicasdejavascript.com.br/instalacao-do-nodejs-e-npm-no-windows-passo-a-passo/
    - Em MAC: https://tidahora.com.br/instalando-o-npm-nodejs-no-mac-os-x/

JavaScript

- **TIPOS DE DADOS EXEMPLOS:**
  - ***Number***: integers, float, etc;
  - ***String***: vetor de caracteres, definidos por “” ou ‘’;
  - ***Boolean***: **true** ou **false**;
  - ***Null***: sem valor;
  - ***Undefined***: Uma variável declarada mas sem valor definido;
  - ***Symbol (ES6)***: Valor único que não pode ser igual a nenhum outro valor;
  - Qualquer outro tipo é um **Object;**
  - **EXEMPLOS:**
    - **https://repl.it/@g1ll/tiposemjs**
- Um pouco mais sobre tipos primitivos em js
  - Primitivos: https://developer.mozilla.org/pt-BR/docs/Glossario/Primitivo
  - Novos tipos (ES5, ES6, ES7-8 Next): http://modernjs.com/data-types.html

JavaScript

- **DECLARANDO VARIÁVEIS:**

  Para a declaração de variáveis o JavaScript tradicional, antes do ES6, usava apenas a palavra reservada **var**. Com a atualização para o ES 6 foram definidas as palavras chaves ***let*** e ***const***, vejamos as diferenças:

  - **var**: usado até a especificação ES5, permite redeclarar variáveis, pode ser local ou global e independe de escopo de **{ }**, depende apenas de escopo de função.
  - ***let***: grande diferença para o ***var*** é que seu escopo é dependente de bloco, qualquer bloco **{ }**, como por exemplo dentro de um **IF.**
  - ***const***: escopo por bloco, assim como o ***let***, porém não deixa redefinir a variável, ou seja, atribuir novos valores;
    - Mas é possível alterar elementos internos de um Array (vetor) ou Atributos de um Objeto;
    - Na maioria dos casos as variáveis serão declaradas com ***const*** (90%);
    - Links Úteis:
    - VAR, LET e CONST

JavaScript

DECLARANDO VARIÁVEIS - EXEMPLOS:

- Este código funciona sem erros para ***VAR***:

```
var  pi = 3.14
var  pi = 3 //Redeclaração é aceita sem erros
```

Link: https://repl.it/@g1ll/jsvar

- Com ***LET*** :

```
let pi = 3.14
var pi = 3 //Erro {Identifier 'pi' has already been declared}
let pi = 0.14  //Erro {Identifier 'pi' has already been declared}
console.log(pi)
```

Mais exemplos no Link: https://repl.it/@g1ll/jslet

- Com ***const*** : Mais Exemplos : https://repl.it/@g1ll/jsconst

```
const pi = 3.14
      pi = 3.14 //Erro {Assignment to constant variable.}
```

JavaScript

TIPO NUMBER e BIGINT - EXEMPLOS:

- Para representar números o JS usa o tipo ***NUMBER*** que possui a precisão de um ***double*** de 64 bits de acordo com a especificação IEEE754 (- 2^{53} a 2^{53}) :
  - Algumas funções úteis:
    - toFixed(numeroDeCasasDecimais): retorna o número formatado com o número de casas decimais passado como parâmetro. O retorna é uma String.

```
var  pi = 3.1415
console.log(pi.toFixed(2)) //Imprime 3.14
```

Link: https://repl.it/@g1ll/jsnumbertype

JavaScript

TIPO NUMBER e BIGINT - EXEMPLOS:

- O tipo ***BigInt*** representa valores maiores que 2^{53}
- Possui suporte em Chrome e Firefox
  - Proposta do ***BigInt***: https://github.com/tc39/proposal-bigint
  - Exemplos:

```
let aBigInt = 11n
let aNumber = 156
aBigInt = BigInt(aNumber) // converte para BigInt
aBigInt === 156n // true
typeof 156
// "number"
typeof 156n
// "bigint"
```

Mais exemplos no Link: https://repl.it/@g1ll/jsnumbertype

  - Ainda não é recomendado usar se a aplicação não trabalha com valores maiores do que o limite do ***Number*** **(REF)**.

JavaScript

TIPO STRING - EXEMPLOS:

- O objeto global ***String*** é um construtor para **strings**, ou uma sequência de caracteres

```
const  texo1 = ”olá” //Criação literal com “”
const  texto2 = ’mundo’ //Criação literal com ‘’
const  mensagem = String(texto1 + texto2) //Objeto Global
```

- ***Concatenação***

```
console.log( ’1’ + 1) //imprime 11 no console
console.log(texto1 + texto2)
```

- ***Templates literals*** :

```
console.log(`O programa diz: ${mensagem}`)
console.log(`O valor de PI é ${Math.PI.toFixed(2)}`)
```

- Mais Exemplos : https://repl.it/@g1ll/jsstrings

JavaScript

- **FUNCTIONS E ARROW FUNCTIONS:**
  - Mudança na sintaxe
    - **FUNCITON:**

      function nomeDaFuncao(param){return param}

      function (param){return param} //anônima

      Ex: function quadrado(n){return n**2}
    - **ARROW:**

      nomeDaFuncao = (param) => {return param}

      nomeDaFuncao = param => param

      Ex: quadrado = n => n**2
  - ***Function*** possui sua própria referência ***THIS***, uma função também é um objeto.
  - ***Arrow*** Não possui o **THIS,** é sempre **anônimo**, recebendo o *this* do escopo ao qual ela foi declarada.
  - `Exemplos:` https://repl.it/@g1ll/jsfncarrow

JavaScript

ARRAYS:

- Array é um Objeto em JS com a implementação de uma Lista de pares índice/valor ou chave/valor:

```
const carros = [“gol”,”palio”,”celta”]
let frutas = Array(“Laranja”,”Maçã”,”Uva”)
let nuns = Array(“Laranja”,”Maçã”,”Uva”)
```

- O objeto ***Array*** possui métodos que facilitam a manipulação dos seus elementos:

```
carros.push(“corola”) //Adiciona um novo elemento no final
frutas.pop()  //Remove do fim
carros.shift()  //Remove do início
frutas.unshift(“Limão”) //Adiciona no início
carros.splice(0,2) //Remove dois elemento a partir posição 0
//Adiciona os dois elementos a partira da posição 2
frutas.splice(2,1,”corola”,”onix”) //substitui 1 elemento
frutas.splice(2,0,”corola”,”onix”) //Não substitui
```

- Exemplos : https://repl.it/@g1ll/jsarrays

JavaScript

ITERANDO ARRAYS:

- ***forEach()***: ou *paraCada* elemento, recebe uma função que é executada para cada elemento.
  - A função *callback* recebe como parâmetro o próprio elemento da iteração corrente

```
carros.forEach(imprimeCarros);
function imprimeCarros(carro){
    console.log(carro) //Imprime cada elemento do array no console
}
```

- ***Find():*** retorna o primeiro elemento encontrado dentro do array de acordo com um teste.
  - Neste caso a função ***callback*** deve retornar um valor ***true*** se o teste for verdadeiro.

```
let num = [1,2,3];
console.log(num.find(encontraPar)); //Retorna o Elemento
function encontraPar(n){
 if(n%2==0)return true
}
```

  - Existe também o ***findIndex()*** que retorna o índice do elemento e não o próprio elemento.

- ----------

Mais Exemplos : https://repl.it/@g1ll/jsiterarray

JavaScript

ITERANDO ARRAYS:

- ***filter()***: retorna outro ***Array*** com elementos de acordo com retorna ***TRUE*** da função ***callback***.
  - A função ***callback*** recebe como parâmetro o próprio elemento da iteração corrente e deve implementar um teste e retornar ***true***

```
let num = [1,2,3,4,5,6]
let impares = num.filter(filtraImpares); //Retorna um novo array
function filtraImpares(n){
    if(n%2!=0) return true
}
 //Simplificando com arrow function
Simplificando : impares = num.filter(n=>n%2!=0);
```

- ***map():*** retorna um novo array como o método ***filter*** porém baseado nos retornos do ***callback***.

```
console.log(num.map(n=>n*2));
```

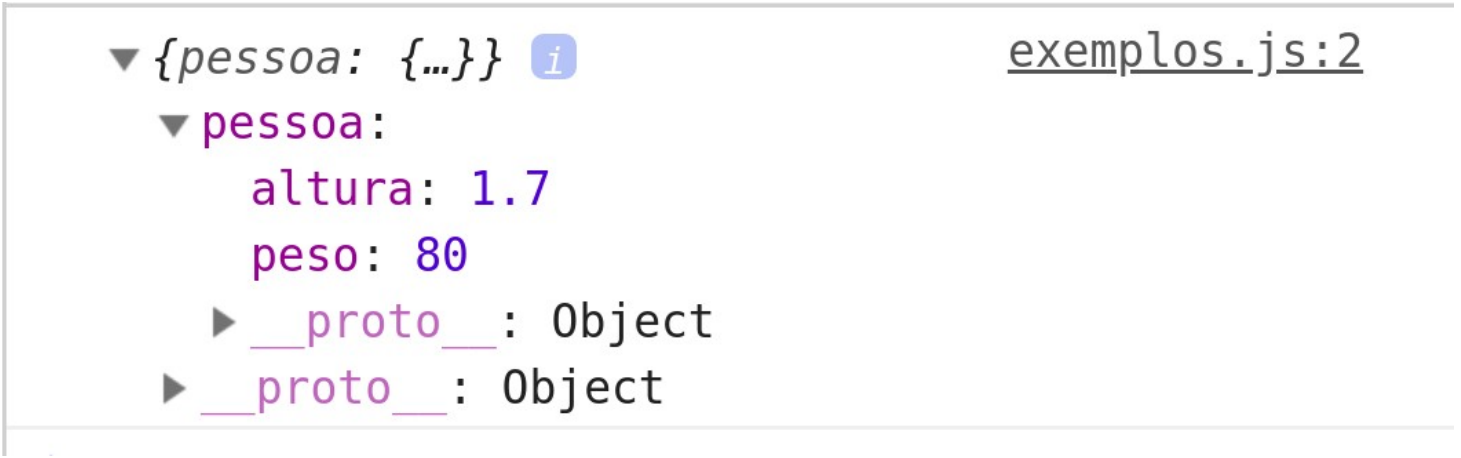
- ----------

  - Mais Exemplos : https://repl.it/@g1ll/jsiterarray

JavaScript

- **OBJETOS:**
  - Em **JS** a **Orientação a Objetos** é baseada em protótipos;
  - Todo o objeto possui a propriedade ***prototype*** **(__proto__)**, ou seja, o protótipo do objeto;
  - Criando um objeto *pessoa* com atributos de **altura** e **peso:**

```
const  pessoa = {altura: 1.7, peso: 80}
console.log({pessoa}) //Mostra o objeto pessoa no console
----------------------------------------------------------
Nodejs: { pessoa: { altura: 1.7, peso: 80 } }
Browser: ctrl+shift+j
```



Link: Referência de Objetos Globais

JavaScript

- **OBJETOS:**
  - É possível adicionar novos atributos a um objeto já declarado
  - Por exemplo adicionar o atributo idade no objeto pessoa

```
const  pessoa = {altura: 1.7, peso: 80}
console.log({pessoa}) //Mostra o objeto pessoa no console
pessoa.idade = 31     //Atributo adicionado ao objeto
console.log({pessoa})
Saída no Node:
-------------------------------------------------------
{ pessoa: { altura: 1.7, peso: 80 } }
{ pessoa: { altura: 1.7, peso: 80, idade: 31 } }
```

- Link: MAIS SOBRE OBJECT

JavaScript

- **OBJETOS:**
  - Mesmo sendo suportado não é recomendado, pois dificulta a leitura de quais são as propriedades do objeto;
  - É melhor já criar o atributo idade na declaração, mesmo que depois seu valor seja modificado

```
const  pessoa = {altura: 1.7, peso: 80, idade: 0 }
console.log({pessoa}) //Mostra o objeto pessoa no console
pessoa.idade = 31  //Atribuindo novo valor ao atributo já existente do objeto
console.log({pessoa})
Saída no Node: ------------------------------------
  { pessoa: { altura: 1.7, peso: 80 } }
  { pessoa: { altura: 1.7, peso: 80, idade: 31 } }
```

  - Exemplos: https://repl.it/@g1ll/jsobjects

JavaScript

- **ITERANDO OBJETOS:**
  - É possível iterar atributos de um objeto usando usando o método ***entries()*** que retorna um par [chave, valor] para cada propriedade do Objeto.
  - Nos casos abaixo temos acesso aos nomes e aos valores dos atributos do objeto como arrays.

```
const  carro = {marca: 'VW', cor: 'Preto', idade: 0 }
//Mostra arrays para cada par chave, valor do objeto
for (let itens of Object.entries(carro)) {
    console.log(itens); //Item é um array [chave, valor]
}
//Neste caso temos acesso direto a chave e valor de cada atributo do
objeto em cada iteração do FOR( OF )
for (let [chave, valor] of Object.entries(carro)) {
   console.log(`atributo: ${chave} | valor: ${valor}`);
}
```

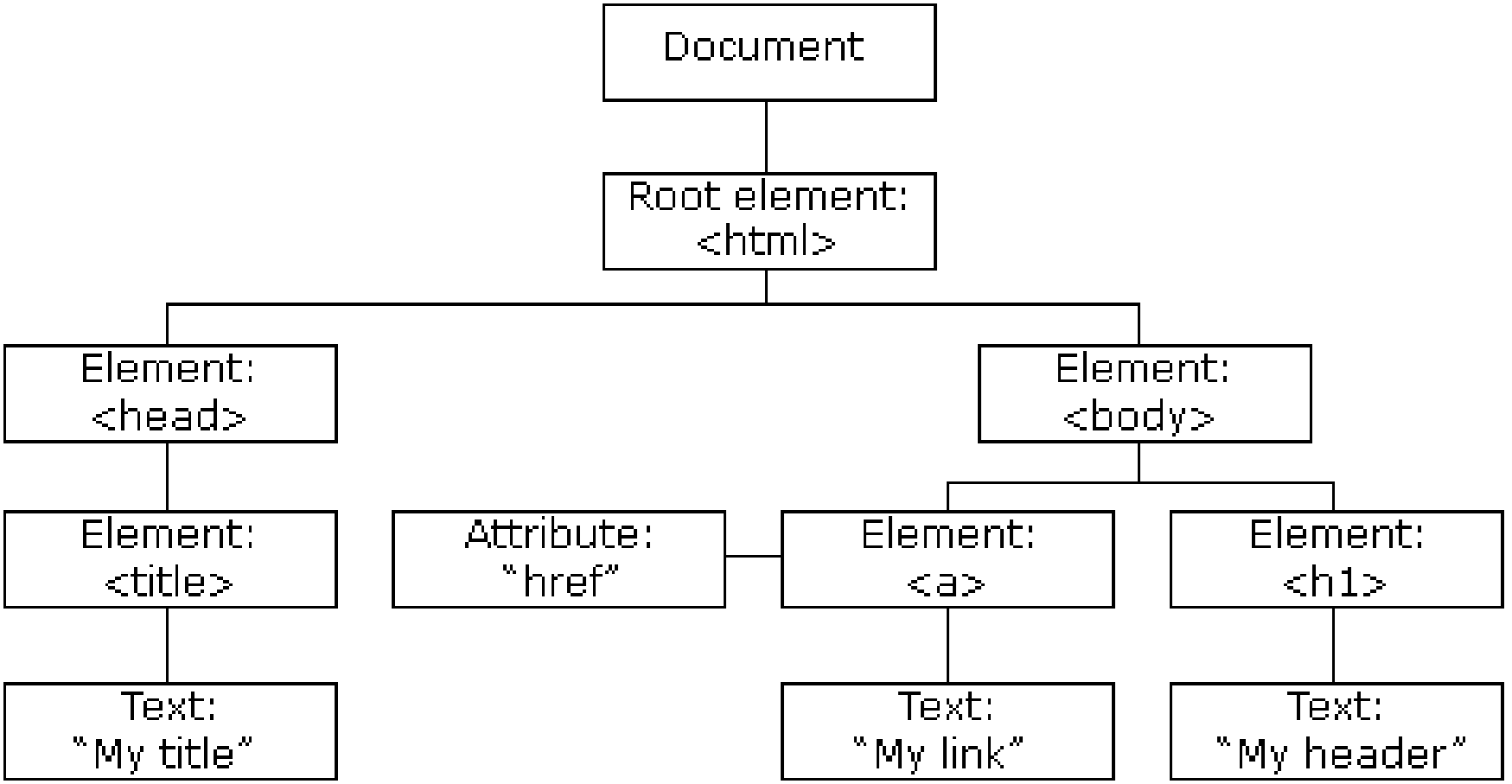
  - Exemplos: https://repl.it/@g1ll/iterobjects
  - Link: Mais sobre iteradores e geradores

JavaScript

- **API DOM:**
  - Representa o documento HTML e seus elementos, também usado para manipulação de XML.
  - Possui quatro objetos principais:
    - **Document:** Representa o próprio documento HTML e é o nó principal
    - **Element:** Representa qualquer elemento HTML
    - **Attribut:** Representando os atributos dos Elementos
    - **Text:** Representa o conteúdo texto
  - **Alguns Métodos do document:**
    - createElement(nomeDaTag): Criar Elementos
    - createTextNode(texto) : Cria um nó representando um texto
    - getElementByID(ID): Retorna um objeto Element representando a tag pelo ID;
    - querySelector(SeletorCSS): Retorna um Elemento de acordo com o Seletor CSS;
    - QuerySelectorAll() Retona uma Lista de Elementos (NodeList)
  - `Exemplos: https://repl.it/@g1ll/jsapidom`
  - `Referência MDN`

JavaScript

- **API DOM:**



JavaScript

- **API DOM:**
  - ***Element*** representa uma tag e possui todas as propriedades e métodos de acordo com a TAG;
  - É possível alterar seus atributos, estilos e utilizar métodos para manipular o seu conteúdo.
  - Exemplo:

```
let div = document.getElementById('demo')
div.setAttribute('style','display:block');
div.appendChild(document.creatTextNode('texto'))
div.style.color = 'blue';
```

JavaScript

- **EVENTOS:**
  - Eventos são ações que ocorrem na página, como o usuário clicar em um botão ou passar o mouse sobre um elemento.
  - Cada objeto Element possui uma variedade de eventos. Os eventos podem variar de tipos de objetos diferentes.
  - Por exemplo, um Element da tag Form suporta o evento submit, enquanto que um objeto Element da tag input suporta o evento change.
  - Eventos são tratados por ouvintes de eventos criados a partir do método addEventListener() ou atribuídos por atributos do tipo on[nomeDoEvento]
    - Ex:
      - form.addEventListener('submit',function(){})
      - form.onsubmit = function(){}
  - **Exemplos:** https://repl.it/@g1ll/jsapidomeventos
  - Mais Sobre Eventos

JavaScript

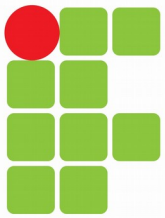
- **FORMULÁRIO:**
  - Usando a API DOM e o tratamento de Eventos podemos capturar os dados de um formulário via JS.
  - Usamos o Evento *submit* passando um callback, ou seja, uma função que deve ser executada ao submeter o formulário.

```
const form = document.querySelector('form');
form.onsubmit = function (e) {
    e.preventDefault()
```

  - Usamos o método ***preventDefault***() para não recarregar a página. Basicamente este método interrompe o comportamento padrão do evento.
  - Dentro da função podemos acessar os campos do formulário de diversas formas:
    - Usando **this**, que neste caso representa o objeto Form
    - Acessando os campos via o atributo name: this.nameDoInput
    - E pegando o valor através do atributo value.
    - Também é possível acessar via ID cada elemento com método getElementByID
  - `Exemplo Comentado: https://repl.it/@g1ll/jsreadform`

Referências

- https://www.w3schools.com/js/js_htmldom.asp
- https://developer.mozilla.org/pt-BR/docs/Web/JavaScript/Reference/Global_Objects
- https://www.alura.com.br/artigos/entenda-diferenca-entre-var-let-e-const-no-javascript
- https://developer.mozilla.org/pt-BR/docs/Web/JavaScript/Reference/Global_Objects/Object
- https://developer.mozilla.org/pt-BR/docs/DOM/Referencia_do_DOM
- https://developer.mozilla.org/en-US/docs/Web/JavaScript/Guide



INSTITUTO FEDERAL DE
EDUCAÇÃO, CIÊNCIA E TECNOLOGIA
SUL-RIO-GRANDENSE

Desenvolvimento de Sites para Web

3º Semestre – Aula - 04

Prof. Gill Velleda Gonzales

Agosto, 2019