JAVASCRIPT

1 - JAVASCRIPT

Javascript é uma das linguagens mais importantes de nosso tempo, justamente por ser entendida pelos mais diversos navegadores (desktop/mobile) sem a necessidade de instalação de plugins ou qualquer outro artifício. O nome Javascript esta no coletivo imaginário dos desenvolvedores, mas na verdade a linguagem segue o padrão ECMA-262. Este padrão foi criado para tentar unificar a sintaxe e garantir a compatibilidade entre os diversos navegadores do mercado.

Javascript é uma linguagem interpretada, isto é, os scripts são lidos linha a linha e cada instrução é traduzida por um interpretador. Para facilitar seu aprendizado e dispensar qualquer configuração extra em sua máquina, utilizaremos o Google Chrome como plataforma, ele será nosso ambiente de execução de Javascript. Escolhemos o Chrome, mas nada impede que você escolha outro navegador como Firefox, Safari, entre outros. O importante é que o navegador escolhido suporte o ECMASCRIPT 5, padrão suportado por navegadores modernos.

O JavaScript é uma linguagem interpretada, (script) precisamos apenas digitar o código-fonte e o interpretador irá ler esse código e executar as instruções, comando por comando, a partir do próprio texto do código-fonte, cada vez que o script for rodado; assim, não énecessário a criação de um arquivo estático, para alterar o programa basta alterar o código e ele já estará pronto para rodar novamente.

A Tipagem Dinâmica também conhecida como Duck Typing, indica como os tipos das variáveis são definidos. Nesse modo de tipagem, as variáveis podem assumir qualquer tipo ou objeto definido pela linguagem, por exemplo, se uma variável X receber um valor inteiro ela irá se comportar como uma variável inteira, e se mais tarde X receber uma string passará a se comportar como uma string daquele ponto em diante. Assim, não é preciso definir o tipo da variável no momento da sua declaração. O tipo da variável é definido implicitamente pelo seu valor.

Quanto a Segurança, por ser uma linguagem que é executada no computador do cliente, o JavaScript precisa ter severas restrições para evitar que se façam códigos maliciosos que possam causar danos ao usuário.

As principais limitações do JavaScript para garantia de segurança são a proibição de:

• Abrir e ler arquivos diretamente da máquina do usuario

• Criar arquivos no computador do usuário (exceto cookies)

• Ler configurações do sistema do usuario

• Acessar o hardware do cliente

• Iniciar outros programas

• Modificar o valor de um campo de formulário do tipo <input>

No entanto, essas limitações interferem muito pouco no desenvolvimento de aplicações web sérias com a linguagem.

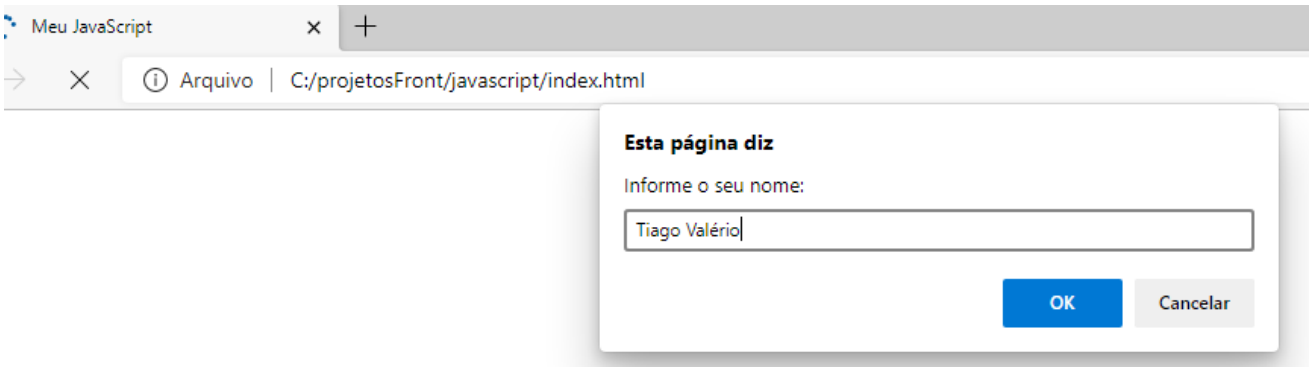
1.1 - Utilizando JavaScript no HTML

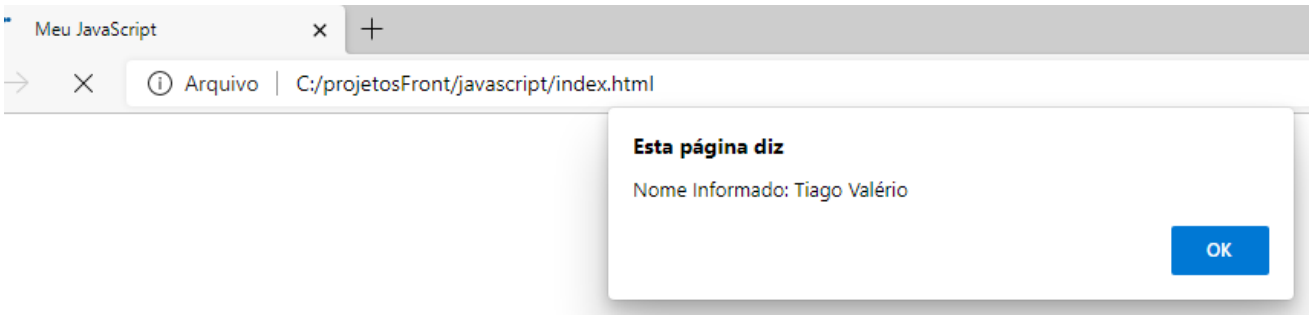
Para adicionar um código JavaScript ao nosso HTML. Precisamos utilizar a tag <script>. Quando o navegador identificar esta tag, será reconhecido como um script dinâmico pelo browser e o mesmo será executado. Para testarmos o JavaScript, iremos criar em nossa pasta projetosFront a pasta javascript. Nesta pasta iremos criar o arquivo index.html.

Neste caso temos a tag <script>, dentro temos um código JavaScript. Será demonstrada uma janela com o texto (Informe o seu nome) e a informação adicionada pelo usuário será atribuída a variável nome. O alert apenas vai demonstrar o conteúdo da varável nome.

```
<html>
    <head>
        <meta charset="utf-8">
        <title>Meu JavaScript</title>
    </head>

    <body>
        <h1>JavaScript</h1>
        <script>
            var nome = prompt("Informe o seu nome:");
            alert("Nome Informado: " + nome);
        </script>
    </body>
</html>
```





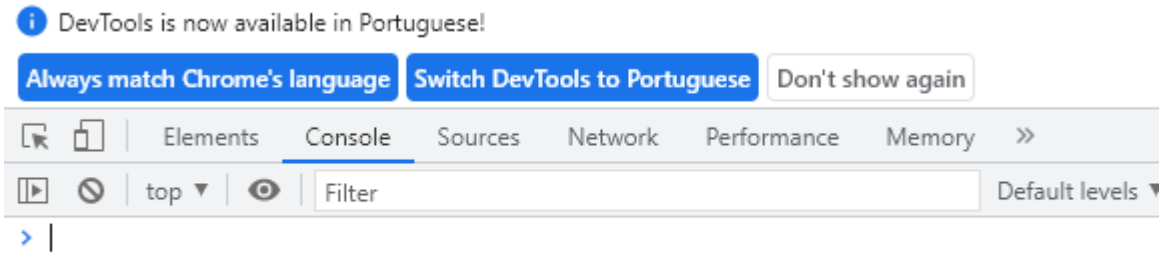
Da mesmo forma que o CSS, o ideal é termos um arquivo separado do HTML e depois realizar o vínculo de nossa página com o arquivo JavaScript. Iremos criar a pasta js, neste iremos criar o scriptInicial.js com o conteúdo da tag <script>.

No HTML iremos alterar a tag <script>, para executar o nosso arquivo JavaScript.

```
<script src="js/scriptInicial.js"></script>
```

2 - Tipos de Dados

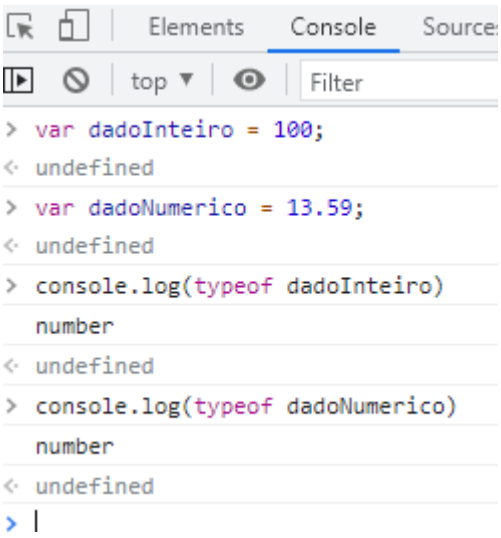
Para entendermos as operações do JavaScript, iremos utilizar o console do Google Chrome. Para isso utilize F12 no Browser, neste iremos utilizar o console para testarmos o JavaScript.



Tipos Numéricos

Em JavaScript os números são representados pelo padrão IEEE 754. Todos os valores numéricos são declarados pela simples atribuição dos valores a uma variável. Temos os valores inteiros e os decimais (com ponto flutuante).

Pode ser verificado que criamos a varíavel dadoInteiro com um valor inteiro e dadoNumerico com valor decimal. Porém ao utilizar o typeof, em ambas foi demonstrado como tipo number.

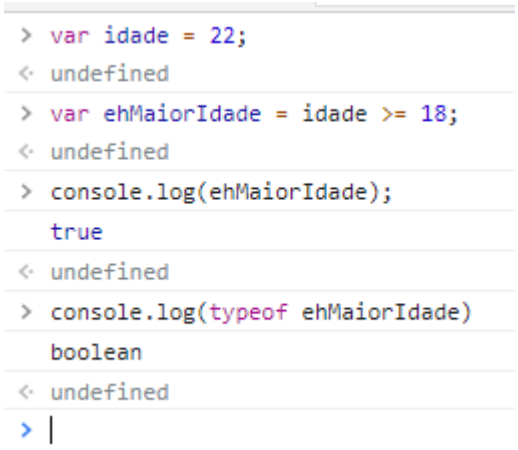


Boleano

Uma variável do tipo booleano pode assumir apenas dois valores: true e false. Os valores deste tipo são em geral usados pela linguagem como resultado de comparações e podem ser usados

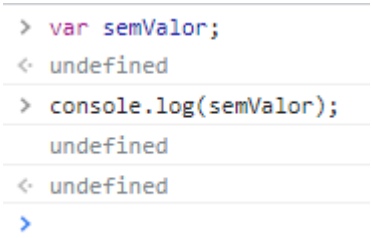
pelo usuário para valores de teste ou para atributos que possuam apenas dois estados. Equivale ao uso de um inteiro com valores 0 ou 1 na linguagem C.

O JavaScript converte automaticamente true para 1 e false para 0 quando isso for necessário.



Indefinido

Uma variável é indefinida quando ela foi declarada de alguma forma mas não possui nenhum valor concreto armazenado. Quando tentamos acessar uma variável que não teve nenhum valor associado a ela teremos como retorno "undefined" (indefinido).



Null

O null é a ausência de valor; quando atribuimos null a um objeto ou variável significa que essa variável ou objeto não possui valor válido. Para efeito de comparação, se usarmos o operador de igualdade "==", JavaScript irá considerar iguais os valores null e undefined. E isso não afeta o uso da comparação (var.metodo == null) quando queremos descobrir se um objeto possui determinado método. No entanto, se for necessário diferenciar os dois valores é recomendável o uso do operador "===" de identicidade. Assim, para efeito de comparação, undefined e null são iguais, mas não idênticos.

```
> var valorNulo = null;
< undefined
> console.log(valorNulo);
  null
< undefined
>
```

Strings

Strings são sequências de caractéres. Em JavaScript a string pode ser tanto um tipo primitivo de dado como um objeto; no entanto, ao manipulá-la temos a impressão de que sejam objetos pois as strings em JavaScript possuem métodos que podemos invocar para realizar determinadas operações sobre elas. Essa confusão ocorre porque quando criamos uma string primitiva, o JavaScript cria também um objeto string e converte automaticamente entre esses tipos quando necessário.

Este conceito será explicado melhor adiante, quando tratarmos de objetos. Para se declarar uma string, basta colocar uma sequência de caractéres entre aspas simples ou duplas.

```
> var nome = "Tiago da Rosa Valério";
< undefined
> console.log(nome);
  Tiago da Rosa Valério
< undefined
> console.log(typeof nome);
  string
< undefined
>
```

Arrays

Os Arrays são pares do tipo inteiro-valor para se mapear valores a partir de um índice numérico. Em JavaScript os Arrays são objetos com métodos próprios. Um objeto do tipo Array serve para se guardar uma coleção de ítens em uma única variável.

Para acessar as variáveis dentro de um array basta usar o nome do array e o índice da variável que se deseja acessar.

Do mesmo modo, pode-se fazer atribuições ou simplesmente ler o conteúdo da posição. Em JavaScript os arrays podem conter valores de tipos diferentes sem nenhum problema; podemos colocar em um mesmo array inteiros, strings, booleanos e qualquer objeto que se desejar.

```
> var alunos = ["Tiago", "João", "Maria"];
< undefined
> console.log(alunos);
  ▶ (3) ['Tiago', 'João', 'Maria']
< undefined
> console.log(typeof alunos);
  object
< undefined
> console.log(alunos[0]);
  Tiago
```

3 - Operadores

Aritméticos

Operador	Operação	Exemplo
+	Adição	x+y
-	Subtração	x-y
*	Multiplicação	x*y
/	Divisão	x/y
%	Módulo (resto da divisão inteira)	x%y
-	Inversão de sinal	-x
++	Incremento	x++ ou ++x
--	Decremento	x-- ou --x

```
> var a = 10;
< undefined
> var b = 5;
< undefined
> var soma = a + b;
< undefined
> var subtracao = a - b;
< undefined
> console.log(soma);
  15
< undefined
> console.log(subtracao);
  5
```

Comparação

Operador	Função	Exemplo
==	Igual a	(x == y)
!=	Diferente de	(x != y)
>	Maior que	(x > y)
>=	Maior ou igual a	(x >= y)
<	Menor que	(x < y)
<=	Menor ou igual a	(x <= y)

```
> var ano = 1981;
undefined
> var anoAtual = 2021;
undefined
> var ehAnoAtual = ano == anoAtual;
undefined
> console.log(ehAnoAtual);
  false
```

Atribuição

Operador	Exemplo	Equivalente
=	x = 2	Não possui
+=	x += y	x = x + y
-=	x -= y	x = x - y
*=	x *= y	x = x * y
/=	x /= y	x = x / y
%=	x %= y	x = x % y

Lógicos

Operador	Função	Exemplo
&&	E Lógico	(x && y)
||	OU Lógico	(x || y)
!	Negação Lógica	!x

```
> var notaFinal = 9;
< undefined
> var frequencia = 25;
< undefined
> var aprovado = notaFinal >= 7 && frequencia >= 20
< undefined
> console.log(aprovado)
  true
```

4 - DOM

O DOM (Document Object Model) é uma interface que representa como os documentos HTML e XML são lidos pelo seu browser. Após o browser ler seu documento HTML, ele cria um objeto que faz uma representação estruturada do seu documento e define meios de como essa estrutura pode ser acessada. Nós podemos acessar e manipular o DOM com JavaScript, é a forma mais fácil e usada.

Com ele você tem infinitas possibilidades, você pode criar aplicações que atualizam os dados da página sem que seja necessário uma atualização. Pode também criar aplicações que são customizáveis pelo usuário, mudar o layout da página sem que seja necessário atualização. Arrastar, mover, excluir elementos. Ou seja, você tem infinitas possibilidades, milhares de coisas que você pode fazer manipulando o DOM, basta você usar sua criatividade.

4.1 - Manipulando o DOM

Utilizando o objeto document pode-se ter acesso a um grande número de propriedades. Segue abaixo algumas propriedades que podem ser utilizadas com o objeto document:

Propriedade	Descrição
documentElement	Captura o elemento raiz <html> de um documento HTML.
getElementById	Busca um elemento da página Web com o uso do atributo id do elemento.
createElement	Cria um nodo elemento na página.
createAttribute	Cria um nodo atributo na página.
createTextNode	Cria um nodo texto na página.
getElementsByTagName	Retorna um array dos elementos com o mesmo nome.
appendChild	Insere um novo elemento filho.
removeChild	Remove um elemento filho.
parentNode	Retorna o nodo pai de um nodo.

Todas as páginas Web são de alguma forma uma árvore. Isso se deve ao fato de podermos ver uma página Web como uma árvore, com uma raiz como o elemento HTML e os seus filhos como o <HEAD> e o <BODY> que por sua também possuem elementos filhos e assim sucessivamente. Os elementos que não possuem filhos são chamados de nós folhas, como por exemplo os elementos <TITLE>,<STYLE>,<SCRIPT>,<LI>,<H1>,<P>,<TD> demonstrados acima. Note que Text é um texto que está dentro de um elemento. O nó <TD> por exemplo também é considerado um nó, mas um nó de tipo diferente (tipo texto).

Essa estrutura de árvore é a forma que o navegador organiza as marcações do HTML, é dessa forma que o navegador Web enxerga um documento HTML. A leitura da árvore se dá sempre da esquerda para a direita, assim teremos a página Web original.

Uma boa prática para evitar que os navegadores apresentem a pagina Web de formas diferentes é sempre escrever HTML padrão, se possível sempre validando a página HTML. Fechar os elementos corretamente, utilizar tags atuais evitando as tags desatualizadas ajudam os navegadores a exibirem uma página Web de maneira correta. As especificações do Object Document Model são publicadas pela World Wide Web Consortium (W3C). Portanto, o DOM é um padrão de fato.

Iremos criar em nosso projeto o arquivo olaMundo.html. Neste adicionamos o elemento h1, com o id “titulo”.

```html
<html>
    <head>
        <meta charset="utf-8">
        <title>Olá JavaScript</title>
    </head>

    <body>
        <h1 id="titulo">Hi, We are JavaScript!!!!</h1>
    </body>
</html>
```

Então iremos criar um script para pegar o conteúdo da tag <h1> e iremos demonstrar o seu conteúdo com o console.log. Iremos criar na pasta js, o arquivo olaMundo.js. O comando querySelector, permite pegar um dos elementos de nosso HTML, neste caso iremos pegar pelo id, por este motivo temos #titulo. Com a tag selecionado, iremos pegar o seu texto e demonstrar uma mensagem ao usuário.

```javascript
var tagTituloDaPagina = document.querySelector("#titulo");
varTexto = tagTituloDaPagina.textContent;
alert(varTexto);
```

Depois precisamos apenas vincular nosso javascript ao HTML.

```html
<body>
    <h1 id="titulo">Hi, We are JavaScript!!!!</h1>
    <script src="js/olaMundo.js"></script>
</body>
```

Então, iremos criar um script para calcular os dados de uma table. Neste caso temos uma tabela com nome, peso, altura e IMC. Iremos realizar o calculo pegando as informações por id.

```html
<table class="table">
    <tr>
        <th>Nome</th>
        <th>Peso</th>
        <th>Altura</th>
        <th>IMC</th>
    </tr>
    <tr>
        <td id="nome-1">Leonardo</td>
        <td id="peso-1">50.0</td>
        <td id="altura-1">1.60</td>
        <td id="imc-1"></td>
    </tr>
    <tr>
        <td id="nome-2">Paulo</td>
        <td id="peso-2">100</td>
        <td id="altura-2">2.00</td>
        <td id="imc-2"></td>
    </tr>
</table>
```

Iremos criar na pasta js, o arquivo tabela.js. Neste iremos pegar as informações pelo id, iremos realizar o calculo e depois devolver o valor calculado no campo do imc.

```javascript
var peso1 = document.querySelector("#peso-1").textContent;
var altura1 = document.querySelector("#altura-1").textContent;
var imc1 = peso1 / (altura1 * altura1);
document.querySelector("#imc-1").textContent = imc1;

var peso2 = document.querySelector("#peso-2").textContent;
var altura2 = document.querySelector("#altura-2").textContent;
var imc2 = peso2 / (altura2 * altura2);
document.querySelector("#imc-2").textContent = imc2;
```

No arquivo tabela.html, adicionar o JavaScript.

```html
<script src="js/tabela.js"></script>
```

5 - Estruturas de Controle

if / else

A estrutura if é usada quando se deseja verificar se determinada expressão é verdadeira ou não, e executar comandos específicos para cada caso.

```
var idade = 16;
if (idade >= 18){
    console.log('Maior de Idade')
}else{
    console.log('Menor de Idade')
}
```

Assim, se a expressão for avaliada como verdadeira, o primeiro bloco de comandos é executado. Caso seja avaliada como falsa, o bloco de comandos que segue o else será executado. Também é possível aglomerar mais testes, utilizando-se o comando else if.

```
var nota = 9;
if (nota >= 7){
    console.log('Aluno Aprovado')
}else if(nota >= 5){
    console.log('Aluno em Recuperação')
}else{
    console.log('Aluno Reprovado')
}
```

Outra forma possível de se utilizar o if é com sua forma abreviada como na linguagem C, usando o operador ternário ?. Ele pode criar estruturas de decisão simples em apenas uma linha de comando, porém, muitas vezes isso pode prejudicar a clareza do seu código, tornando-o complicado de entender para alguém que não esteja familiarizado com o uso desde operador condicional.

```
var salario = 1500;
var mensagem = salario >= 2000 ? "Precisa pagar Imposto de Renda" : "Isento de Impostos";
console.log(mensagem);
```

switch / case

As estruturas do tipo switch são usadas quando queremos selecionar uma opção dentre várias disponíveis.

```
var estado = 'SC';
var descricao = '';

switch(estado){
    case "SC":
        descricao = "Estado de Santa Catarina";
        break;
    case "RS":
        descricao = "Estado do Rio Grandedo Sul";
        break;
    case "PR":
        descricao = "Estado do Paraná";
        break;
    default:
        descricao = "Estado Desconhecido";
        break;
}
console.log(descricao);
```

Ao contrário de outras linguagens, os valores de comparação podem ser strings além de valores numéricos. O comando break faz com que o switch pare de verificar as outras possibilidades abaixo e pode ser omitido caso se deseje uma estrutura que tornará mais de uma opção como verdadeira. Por fim, default é opcional e corresponde a uma sequência de comandos que será executada quando nenhum dos outros case o forem.

while

Os laços do tipo while são usados quando se deseja que uma sequência de ações seja executada apenas no caso da expressão de condição ser válida. Assim, primeiro a expressão é testada, para depois o conteúdo do laço ser executado ou não.

```
var tabuada = 8;
var numero = 0;

while(numero < 10){
    numero = numero + 1;
    var multiplicacao = tabuada * numero;
    console.log(numero + ' * ' + tabuada + ' = ' + multiplicacao);
}
```

do / while

Diferentemente do while, o do ... while primeiro executa o conteúdo do laço uma vez e, depois disso, realiza o teste da expressão pare decidir se continuará executando o laço ou irá seguir o resto do programa.

```
var tabuada = 8;
var numero = 0;

do{
    numero = numero + 1;
    var multiplicacao = tabuada * numero;
    console.log(numero + ' * ' + tabuada + ' = ' + multiplicacao);
}while(numero < 10)
```

for

Na maioria das vezes, quando usamos um laço do tipo while também construímos uma estrutura com um contador que é incrementado a cada passo para controle do laço e manipulação interna de objetos, arrays como nos exemplos anteriores. Os laços for oferecem a vantagem de já possuírem em sua estrutura essa variável de contador e incrementá-la de maneira implícita.

```
var valorASomar = [2,8,3,5];
var soma = 0;

for(var x = 0; x < 4; x++){
    soma = soma + valorASomar[x];
}
console.log('Soma = ' + soma);
```

O significado do comando é for(variável de contador inicializada ; condição de parada ; incremento da variável de contador).

6 - Funções

Funções possuem um papel muito importante na programação estrutural pelo fato de ajudar muito na modularização no programa, ou seja, viabiliza a divisão do programa em partes menores e logicamente relacionadas. Em JavaScript, existem diversas maneiras de se declarar uma função; mostraremos todas elas aqui com exemplos.

Um ponto importante é que em JavaScript as funções são consideradas como dados, ou seja, podemos atribuir uma função a uma variável ou propriedade de um objeto e a partir desde momento usar a variável ou a propriedade da mesma forma que se usaria a função. Elas também podem ser passadas como argumentos para outras funções e por isso funções de JavaScript são chamadas funções de alta ordem, elas podem tanto receber funções como argumento quanto retornar uma função.

Expressão function

A primeira maneira de se declarar uma função é através do uso da palavra-chave function de maneira similar a como elas são declaradas na linguagem C, com as diferenças de que em JavaScript não definimos o tipo de retorno e nem mesmo o tipo dos argumentos. Uma função complexa pode ser capaz de tratar argumentos diferentes e retornar argumentos diferentes dependendo das circunstâncias nas quais foi invocada. Deve-se definir seu nome e seus argumentos conforme mostra o exemplo a seguir.

```
var soma = somar(3,5);
console.log(soma);

function somar(a, b){
    var valores = a + b;
    return valores;
}
```

Funções como literais

Também podemos declarar uma função em JavaScript é através de literais.

```
var funcaoSoma =
     function (a, b){
          var valores = a + b;
          return valores;
}
console.log('Soma 1 =' + funcaoSoma(4,5));
console.log('Soma 2 =' + funcaoSoma(3,2));
```

Essa forma é basicamente a mesma que declarar através do construtor Function(). No entanto, ela é melhor porque os comandos podem ser declarados com a sintaxe normal de JavaScript ao invés de ser uma string como é o caso do construtor. Com literais não há necessidade de manter a função em uma linha, dentro das chaves podemos construir a função usando um comando por linha normalmente.

7 - Objetos

Ao contrário de uma variável, um objeto pode conter diversos valores e de tipos diferentes armazenados nele (atributos) e também possuir funções que operem sobre esses valores (métodos). Tanto os atributos, quanto os métodos, são chamados de propriedades do objeto. Para criar um objeto é muito simples, basta invocar seu construtor através do operador new.

```
var carro = {
    modelo : "BMW",
    cor : "preto",
    ano: 2021
}

console.log(carro.modelo);
console.log(carro.cor);
console.log(carro.ano);
```

Para acessar uma propriedade de um objeto, basta usar objeto.propriedade e no caso de métodos adicionar o operador (). Podemos definir, como já foi dito, um construtor para um objeto, assim podemos inicializar atributos logo no momento da instanciação do objeto. Para que um construtor inicialize um atributo, ele precisa ser referenciado através da palavra-chave this.

```
function Pessoa(nome, idade, endereco){
    this.nome = nome;
    this.idade = idade;
    this.endereco = endereco;
}

var professor = new Pessoa("Tiago Valério", 40, "Içara - SC");

console.log("Nome: " + professor.nome);
console.log("Idade: " + professor.idade);
console.log("Endereço: " + professor.endereco);
```

Métodos

No paradigma de orientação a objetos, os métodos são simplesmente funções que são invocadas por meio de um objeto! E em JavaScript isso é levado tão a sério que a maneira de se criar métodos para seus objetos leva isso ao pé da letra. Basta criarmos uma função e atribuí-la a uma propriedade do objeto.

```
function Pessoa(nome, idade, endereco){
    this.nome = nome;
    this.idade = idade;
    this.endereco = endereco;

    this.maiorDeidade = function(){
        if(this.idade >= 18){
            return 'Maior de Idade';
        }else{
            return 'Menor de Idade';
        }
    }
}

var professor = new Pessoa("Tiago Valério", 10, "Içara - SC");

console.log("Situação Idade: " + professor.maiorDeidade());
```

Também podemos definir os métodos dentro do próprio construtor de uma função, tanto definindo a função fora e atribuindo no construtor, como definindo a própria função dentro do próprio construtor uma vez que JavaScript suporta o aninhamento de funções.

8 - JavaScript em Form

Para concluirmos nossa introdução ao JavaScript, iremos adicionar uma função a ser executada pelo botão, para simular gravar os dados do formulario em nosso server em Spring Boot.

Iremos criar o nosso cadastro.html, este arquivo é o template do BootStrap, com o seguinte formulário no body. Importante que no botão temos a função gravarCadastro() no evento onClick. A função será criado no Javascript cadastro.js na pasta js.

```
<h1>Cadastro de Pessoas</h1>

<div class="container">
    <form>
        <div class="form-group">
          <label>Nome</label>
          <input type="text" class="form-control" id="infNome">
        </div>
        <div class="form-group">
            <label>Endereço</label>
            <input type="text" class="form-control" id="infEndereco">
        </div>
        <button type="submit" onclick="gravarCadastro()" class="btn btn-primary">Gravar</button>
      </form>
</div>

<script src="js/cadastro.js"></script>
```

Em nosso cadastro.js, iremos criar a estrutura do objeto Pessoa, com a informação do nome e endereço. Então com o query Selector, pegamos as informações de nosso formulário e criamos um objeto com as informações.

O próximo passo seria indicar a URL de nosso serviço Spring Boot e passar este objeto na requisição. Para visualizarmos as informações no console, será necessário alterar o <button> de submit para botão. Pois o submit vai enviar os dados do formulário e resetar as informações.

```
function Pessoa(nome, endereco){
    this.nome = nome;
    this.endereco = endereco;
}

function gravarCadastro(){
    var nomeTela = document.querySelector("#infNome").value;
    var enderecoTela =  document.querySelector("#infEndereco").value;

    var novoCadastro = new Pessoa(nomeTela, enderecoTela);

    console.log('Agora é só montar a requisição e passar o objeto!!!!')
    console.log("Nome: " + novoCadastro.nome);
    console.log("Endereço: " + novoCadastro.endereco);
}
```

9 - Material Complementar

https://www.youtube.com/watch?v=Ptbk2af68e8

https://www.youtube.com/watch?v=093dIOCNeIc&list=PLQCmSnNFVYnT1-oeDOSBnt164802rkegc

https://www.caelum.com.br/apostila/apostila-html-css-javascript.pdf

https://www.scriptbrasil.com.br/download/apostila/837/

10 - Exercícios

Criar um formulário, que será informado o nome do alunos e 3 notas. O botão deve invocar um JavaScript, que vai pegar a informação das três notas e indicar se o mesmo está Aprovado ou Reprovado. Considerar média 7, use a sua criatividade.