



JS4GIRLS

Florianópolis - SC



Javascript e Lógica de Programação

Introdução

Quando desenvolvemos páginas web, utilizamos como base a linguagem de marcação de hipertexto (HTML), que forma toda a estrutura do documento a partir de tags, esse código HTML é interpretado pelo navegador para montar algum elemento na tela.



Aliando essa estrutura ao CSS (Cascading Style Sheets), conseguimos estilizar os elementos da página, definindo sua formatação visual. Temos assim uma página bem estruturada, com vários elementos visuais (e não visuais) como formulários, caixas e textos, com bordas, cores, plano de fundo, imagens e etc.

Porém, não vamos muito além disso utilizando só HTML e CSS, pois obtemos apenas documentos ESTÁTICOS, sem interação com o usuário, como caixas de diálogo e validação de entrada de dados.

A Linguagem Javascript

História da Linguagem

JavaScript foi originalmente desenvolvido por **Brendan Eich** da *Netscape* sob o nome de Mocha. Posteriormente teve seu nome mudado para LiveScript e por fim **JavaScript**. LiveScript foi o nome oficial da linguagem quando foi lançada pela primeira vez na versão beta do navegador Netscape 2.0 em setembro de 1995, mas teve seu nome mudado em um anúncio conjunto com a Sun Microsystems em dezembro do mesmo ano, quando foi implementado no navegador Netscape versão 2.0B3.

A mudança de nome de LiveScript para JavaScript coincidiu com a época em que a Netscape adicionou suporte à tecnologia Java em seu navegador (Applets).

A escolha final do nome causou confusão dando a impressão de que a linguagem foi baseada em java, sendo que tal escolha foi caracterizada por muitos como uma estratégia de marketing da Netscape para aproveitar a popularidade do recém-lançado Java. JavaScript rapidamente adquiriu ampla aceitação como linguagem de script client-side de páginas web.



A Linguagem Javascript

Como consequência, a Microsoft desenvolveu um dialeto compatível com o próprio JavaScript, mas que levou o nome de JScript para evitar problemas de trademark. JScript foi incluído no Internet Explorer 3.0, liberado em Agosto de 1996. Em novembro de 1996 a Netscape anunciou que tinha submetido o JavaScript para Ecma internacional como candidato a padrão industrial e o trabalho subsequente resultou na versão padronizada chamada ECMAScript.

O JavaScript tem se transformado na linguagem de programação mais popular da web. Inicialmente muitos profissionais denegriram a linguagem, pois a mesma tinha como alvo principal o público leigo. Com o advento do Ajax, o JavaScript teve sua popularidade de volta e recebeu mais atenção profissional. O resultado foi a proliferação de frameworks e bibliotecas, práticas de programação melhoradas e o aumento no uso do JavaScript fora do ambiente de navegadores bem como o uso de plataformas de JavaScript server-side.



A Linguagem Javascript

Resumindo...

- ❖ JS foi criado por Brendan Eich da Netscape com nome de Mocha. Depois mudaram para LiveScript, quando foi lançado pela primeira vez na versão beta do navegador Netscape 2.0 em setembro de 1995. Por fim, oficializou o nome para **JavaScript**.
- ❖ Tinha-se a impressão que a linguagem era baseada no java, porém, foi uma estratégia de marketing da Netscape para aproveitar a popularidade do recém-lançado Java.
- ❖ O JavaScript tem se transformado na linguagem de programação mais popular da web.
- ❖ Com a criação do Ajax, o JS teve sua popularidade em alta e recebeu mais atenção profissional. O resultado foi a proliferação de frameworks e bibliotecas, práticas de programação melhoradas e o aumento no uso do JavaScript fora do ambiente de navegadores bem como o uso de plataformas de JavaScript server-side.



A Linguagem Javascript

Principais características

- ❖ Tipagem dinâmica
- ❖ Funcional
- ❖ Orientada a Objetos
- ❖ Baseada em protótipos

Possui tipos e operadores, objetos e métodos. Sua sintaxe vem das linguagens Java e C, por isso tantas estruturas dessas linguagens se aplicam a JavaScript também. Uma das principais diferenças é que o JavaScript não tem classes; em vez disso, a funcionalidade de classe é realizada por protótipos de objetos. A outra diferença principal é que as funções são objetos, dando as funções a capacidade para armazenar código executável e serem "passadas" como parâmetro para qualquer outro objeto.



A Linguagem Javascript

JAVASCRIPT \neq JAVA

Não confunda! **Javascript** é diferente de **JAVA**:

“Java está para o Javascript assim como Bola está para Bolacha.”

Principais diferenças

- ❖ Java é uma linguagem compilada, enquanto JavaScript é uma linguagem interpretada;
- ❖ Java é fortemente tipado, enquanto que o JavaScript é fracamente tipado.
- ❖ Isso só para citar as maiores diferenças, sem contar a sintaxe.



A Linguagem Javascript

Onde usar?

O uso primário de JavaScript é escrever funções que são embarcadas ou incluídas em páginas HTML e que interagem com o Modelo de Objeto de Documentos (DOM) da página.

Inicialmente o Javascript era utilizado apenas nos navegadores. Hoje em dia, com a evolução das engines(motor) de Javascript como SpiderMonkey(Firefox) e V8(Google), eles levaram o Javascript também para o lado do servidor com o Node.js e bancos NoSQL que utilizam o Javascript como CouchDb e MongoDB.

MongoDB é um banco de dados orientado a documentos que armazena dados em documentos JSON com esquema dinâmico. Isso significa que você pode armazenar seus registros sem se preocupar com a estrutura de dados, como o número de campos ou tipos de campos para armazenar valores.



A Linguagem Javascript

Alguns exemplos de uso:

- ❖ Abrir uma nova janela com controle programático sobre seu tamanho, posição e atributos;
- ❖ Validar valores de um formulário para garantir que são aceitáveis antes de serem enviados ao servidor;
- ❖ Mudar imagens à medida que o mouse se movimenta sob elas.



A Linguagem Javascript

O que oferece?

O JavaScript além de ser a linguagem mais usada no Universo nos oferece algumas coisas interessantes que sem elas a Internet como existe hoje não seria possível:

- ❖ Dinamismo
- ❖ Validação de Formulários
- ❖ Interatividade
- ❖ Controle de Comportamento
- ❖ Personalização da página

Atualmente o JavaScript é o motor da Internet, principalmente com o advento do **AJAX** que fez nossas interfaces ficarem mais ricas e interativas.



Introdução à Lógica de Programação

O que é?

Técnica de desenvolver sequências de ações para atingir um objetivo.

Essas sequências são adaptadas para linguagem de computador pelo programador a fim de produzir um sistema.

Essa sequência lógica é denominada **algoritmo**.

Exemplo: O passo a passo de uma receita de bolo.

Algoritmo

Uma sequência finita de passos que levam a execução de uma tarefa. Estas tarefas não podem ser redundantes nem subjetivas na sua definição, devem ser claras e precisas.

Instruções

Em informática uma instrução é a informação que indica a um computador uma ação para se executar, e obter um processo completo, é necessário um conjunto de instruções colocadas em ordem sequencial lógica.



Introdução à Lógica de Programação

Exemplo de Algoritmo

inicio

Pergunta a idade

Grava em um local chamado
idade

Se a idade é maior que 18

Mostra na tela: Maior de idade

Se não

Mostra na tela: Menor de idade

fim

inicio

escreva("Qual sua idade?")

leia(idade)

se idade > 18 então

 escreva("Maior de idade")

senão

 escreva("Menor de idade")

fimse

fim



Inserindo o JS no HTML

- ❖ Inserção dentro da tag head, como um arquivo externo:

```
<script type="text/javascript" src="pasta/arquivo.js"></script>
```

- ❖ Ou incorporado, direto no arquivo:

```
<script type="text/javascript">  
    alert("Olá mundo!");  
</script>
```

- ❖ Fazer testes rápidos:

Usando o console do navegador (tecla F12), execute:

```
alert("Olá mundo!");
```



Variáveis e seus Tipos de Dados

Em qualquer linguagem utilizamos variáveis para armazenar valores e para nos ajudar a manipular e mostrar resultados. Podemos armazenar diferentes tipos de dados nelas, a seguir vamos descobrir como criar uma variável e quais são cada um de seus tipos:

Declarando uma variável (usamos a notação "var" seguido de um sinal de igualdade que significa "atribuir", depois entre aspas o valor a ser guardado):

```
var nomedavariavel = "JS4Girls";
```



Variáveis e seus Tipos de Dados

NULL - O valor null é um literal em JavaScript que representa um valor nulo ou "vazio" (ex: que aponta para um objeto que existe, mas com valor inexistente).

Ex: `var nomedavariavel = null;`

UNDEFINED - Representa um valor indefinido. O tipo undefined é retornado caso uma propriedade de um determinado objeto seja consultada e não exista.

Ex:

```
var nomedavariavel;  
console.log(nomedavariavel); // undefined
```



Variáveis e seus Tipos de Dados

STRING - Tipo utilizado para armazenar um conjunto de caracteres, sempre atribui-se o valor entre aspas.

Ex: `var texto = "1. Meu conjunto de caracteres!!";`

NUMBER - Armazenar números, atribui-se o valor sem aspas:

Ex:

```
var nota1, nota2, soma;  
nota1 = 8.5;  
nota2 = 10;  
soma = nota1 + nota2;  
console.log(soma); // 18.5
```



Variáveis e seus Tipos de Dados

ARRAY (Matriz) - Um array é uma coleção de um ou mais objetos, do mesmo tipo, armazenados em endereços adjacentes de memória. Cada objeto é chamado de elemento do array. O tamanho do array é o seu número de elementos.

Ex: `var frutas = ['uva', 'maçã', 'tomate'];`

BOOLEAN - Boolean é um tipo lógico de variável, dado em homenagem da lógica booleana, de George Boole.

Ela possui apenas 2 valores, sendo estas palavras reservadas da linguagem: `true` ou `false`.

Ex: `var clicado = false;`



Exercício

Escreva um código onde você inicie um array com o nome de 5 dos seus amigos e depois faça ele mostrar a mensagem: "Eu gosto muito da(o) " + nome.

Dicas:

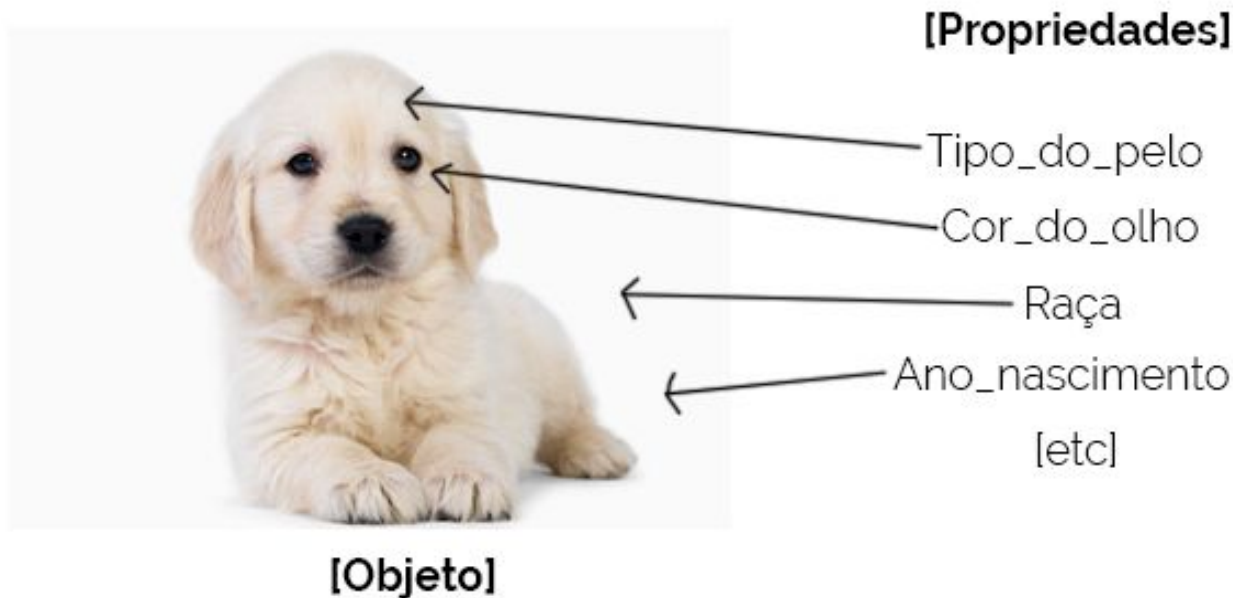
```
console.log(frutas[0]); // uva  
console.log(frutas[1]); // maçã  
console.log(frutas[2]); // tomate
```

```
console.log(frutas.length); // 3
```



Objetos


O que é um objeto? - Um objeto é uma coleção dinâmica de chaves e valores de qualquer tipo de dado. É possível adicionar ou remover propriedades do mesmo. Propriedade é uma associação entre um nome e um valor.



Criação e Manipulação de Objetos

1 - Acessa-se as propriedades de um objeto com uma simples notação de ponto:

Nome_do_objeto.nome_da_propriedade



```
var meuAviao = {}; ou var meuAviao = new Object(); // Criando um objeto
meuAviao.fabricante = "Airbus"; // Criando uma propriedade e atribuindo um valor para ela
meuAviao.modelo = "A380";
meuAviao.ano = 2012;
```

2 - Ou essa segunda notação, lê-se meuAviao na "posição" fabricante, utilizando chaves:

```
meuAviao["fabricante"] = "Airbus";
meuAviao["modelo"] = "A380";
meuAviao["ano"] = 2012;
```

Criação e Manipulação de Objetos

Uma das operações mais usadas nas strings é checar seu tamanho:

```
var palavra = "JS4Girls";  
palavra.length // 8
```

Para 'concatenar' (juntar) valores, usamos os operadores "+" e "+=":

```
var titulo = "JS4Girls";  
var subtítulo = " - Aula de JS e Acessibilidade na Web";  
palavra + subtítulo; // "JS4Girls - Aula de JS e Acessibilidade na Web"
```



Criação e Manipulação de Objetos

Verificando o tipo de dados da variável:

```
var palavra = "JS4Girls";  
typeof palavra; // "string"
```

Selecionando apenas uma posição(índice) do valor:

Índice (**index**) é a posição onde esse valor dessa variável se encontra. Podemos pesquisar o índice de uma letra dentro de uma **string** e também dentro de um **array**.

```
palavra[3]; // "G"  
palavra[0] > palavra[1] // "false"
```

```
var frutas = ["pera", "uva", "maçã"];  
frutas[1]; // "uva"
```



Exercício

Crie um objeto e chame ele de "**MinhaRoupa**".

Em seguida crie as seguintes propriedades:


"**Cor**", "**Tamanho**", "**Marca**" e "**Tipo**" (Camisa ou Calça).



Funções

Um conjunto de instruções utilizadas para executar uma determinada tarefa. Seu principal objetivo é evitar que um trecho de código seja repetido sempre que for preciso efetuar uma mesma operação.

Sintaxe



```
function nome ( parametro ) {  
    //código a ser executado  
}
```

Parâmetros/Argumentos

```
function multiplicar (x, y) {  
    console.log("resultado: " + x * y);  
}
```

```
multiplicar(10, 5); // Atenção! Apenas com essa chamada que o navegador  
irá executar a sua função!
```

Funções

Funções com Retorno

```
function retorna(algo){  
    return algo;  
}  
function multiplicar (x, y) {  
    return x * y;  
}
```

Return além de retornar o resultado, faz com que a função pare sua execução.

Chamando/Invocando a Função

```
nomeDaFuncao(arg1, arg2);  
var resultado = nomeDaFuncao(arg1, arg2);  
console.log(resultado) // retorno da funcao
```



Exercício

Crie uma função para calcular o IMC e invoque-a passando dois argumentos.

*Cálculo do IMC: peso / (altura * altura)*



Operadores Lógicos

AND/E - &&

- ❖ Todas as premissas comparadas devem ser **verdadeiras** para resultar verdadeiro.

Exemplo hipotético: Para se cadastrar no site do Grêmio, o usuário só é aceito se:

`usuarioValido = nome && email // falso`

`usuarioValido = falso && falso // falso`

`usuarioValido = falso && verdadeiro // falso`

`usuarioValido = verdadeiro && falso // falso`

`usuarioValido = verdadeiro && verdadeiro // verdadeiro`



Operadores Lógicos

OR/OU - ||

- ❖ Pelo menos uma das premissas deve ser verdadeira para resultar verdadeiro.

```
usuarioValido = RG || CPF
```

```
usuarioValido = falso || falso // falso
```

```
usuarioValido = falso || verdadeiro // verdadeiro
```

```
usuarioValido = verdadeiro || falso // verdadeiro
```

```
usuarioValido = verdadeiro || verdadeiro // verdadeiro
```



Operadores Lógicos

NOT/Negação - !

- ❖ Pegamos a nossa premissas, caso ela for verdadeira, se tornará falsa, caso for falsa, se tornará verdadeira.

```
usuarioValido = !(colorado)
```

```
usuarioValido = falso // verdadeiro (usuário não é torcedor do Inter)
```

```
usuarioValido = verdadeiro // falso (usuário é torcedor do Inter)
```



Operadores Lógicos - Tabela

Operador	Nome	Exemplo	Resultado
&&	E	<code>(10 > 7) && (9 == 9)</code>	Verdadeiro se 10 for maior que 7 e 9 for igual a 9
 	Ou	<code>(10 > 7) (9 == 9)</code>	Verdadeiro se 10 for maior que 7 ou 9 for igual a 9
!	Negação	<code>!(10 > 7)</code>	Verdadeiro se 10 for menor que 7



Operadores Lógicos - Tabela

Operador	Nome	Exemplo	Resultado
+	adição	9 + 9	18
-	subtração	20 - 20	0
*	multiplicação	12 * 2	24
/	divisão	100/4	25
%	Resto/Módulo	10 % 2 === 0 5 % 2 !== 0	Caso o resto de 10/2 seja 0 é verdadeiro. Caso o resto de 5/2 seja diferente de 0 é verdadeiro.

Potência (função nativa), veja o exemplo:

```
Math.pow(4, 3); // Primeiro número é a base e o segundo o expoente
```


Operadores de Comparação

Operador	Nome	Exemplo	Resultado
<code>==</code>	Igual	<code>a == b</code>	Verdadeiro se <code>a</code> for igual a <code>b</code>
<code>!=</code>	Diferente	<code>a != b</code>	Verdadeiro se <code>a</code> não for igual a <code>b</code>
<code>===</code>	Idêntico	<code>a === b</code>	Verdadeiro se <code>a</code> for igual a <code>b</code> e for do mesmo tipo
<code>!==</code>	Não idêntico	<code>a !== b</code>	Verdadeiro se <code>a</code> não for igual a <code>b</code> , ou eles não são do mesmo tipo
<code><</code>	Menor que	<code>a < b</code>	Verdadeiro se <code>a</code> for menor que <code>b</code>
<code>></code>	Maior que	<code>a > b</code>	Verdadeiro se <code>a</code> for maior que <code>b</code>
<code><=</code>	Menor ou igual	<code>a <= b</code>	Verdadeiro se <code>a</code> for menor ou igual a <code>b</code>
<code>>=</code>	Maior ou igual	<code>a >= b</code>	Verdadeiro se <code>a</code> for maior ou igual a <code>b</code>



Comandos de Controle de Fluxo - Estrutura Condicional

Assim como a maioria das linguagens de alto nível(longe do código de máquina e mais próximo à linguagem humana), Javascript possui estruturas condicionais e de repetição para controle de fluxo.

- ❖ **IF-ELSE** (da tradução: SE - SE NÃO) - Testa se algo é verdadeiro

Exemplo da sintaxe da estrutura IF-ELSE:

```
if(condição 1){  
    //Faz Ação se condição 1 for verdadeira  
}
```

```
if(condição 1){  
    //Faz Ação se condição 1 for verdadeira  
}else if (condição 2){  
    //Faz ação se condição 2 verdadeira  
}else{  
    //Faz ação se nenhuma anterior é verdadeira  
}
```



Exemplo

```
var idade = 18;

if(idade == 18) {
    console.log("Maior de idade");
} else {
    console.log("Menor de idade");
}
```



Comandos de Controle de Fluxo - Estrutura Condicional

- ❖ **SWITCH** - Exemplo de sintaxe da estrutura do Switch (tradução interruptor) - Se estadoCivil seja correspondente a algum caso, ele retornar a mensagem e sai do laço:

```
var estadoCivil = prompt("Qual seu estado civil?");

switch(estadoCivil) {
  case 'solteira':
    console.log("Bora pra festa?");
    break;
  case 'casada':
    console.log("Parabéns pelo casamento!");
    break;
  case 'divorciada':
    console.log("Deve ser um alívio!");
    break;
  case 'viúva':
    console.log("Meus pesames!");
    break;
  default: console.log("Complicado");
}
```

Utilizamos instrução 'default' para executar um código quando nenhum dos outros cases foi verdadeiro. Dessa forma deixando nosso código mais simples e claro, ao invés de usar muitos comandos else e if.



Comandos de Controle de Fluxo - Estrutura de Repetição

Estrutura que facilita nossa vida, chamamos de laços de repetição. Temos o While, Do While e For, ambos são comandos que servem para manter um pedaço do script executando repetidamente. Também conhecidos como estrutura de repetição, iteração ou loop, esses comandos mantêm a execução até que seu argumento seja falso.

❖ **WHILE** (tradução: enquanto)

```
while( proposição ) {  
    //comandos  
}
```

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

[Exercício] Escreva um código onde inicie um número com o valor 0 e vá até 20, mostrando apenas os valores pares. Sabendo que o operador % retorna o resto da divisão, por exemplo:

```
var numero = 5;  
var verificasepar = 5 % 2 == 0; // false
```



Comandos de Controle de Fluxo - Estrutura de Repetição

Muito parecido com o while, porém sempre irá executar a primeira vez, e só depois começará a conferir a condição depois.

❖ **DO WHILE** (tradução: faça, enquanto):

```
do {  
    //comandos  
} while( proposição );
```

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



Comandos de Controle de Fluxo - Estrutura de Repetição

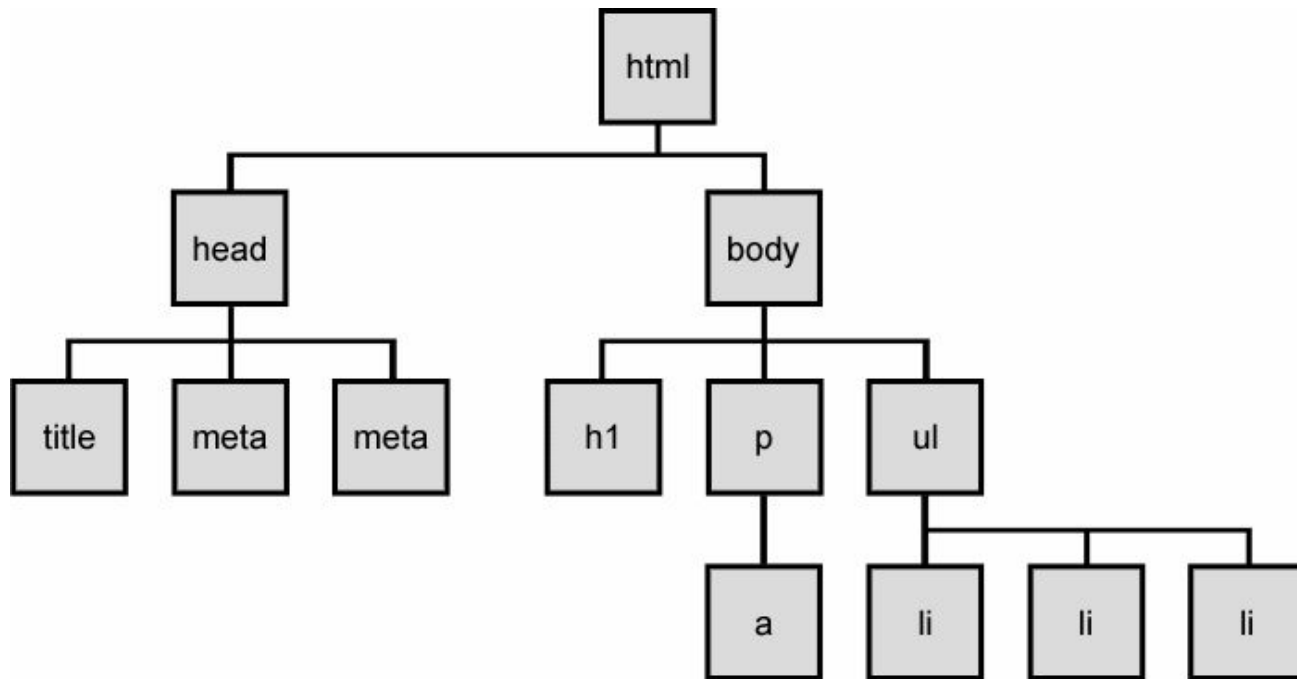
❖ **FOR** (tradução: para)

```
for (inicialização; condição; expressão final) {  
    // seus comandos  
}  
  
for (var numero = 1; numero <= 10; numero++) {  
    console.log(numero);  
}
```



Javascript e o Navegador - O que é DOM?

O Modelo de Objeto de Documento (do inglês Document Object Model - DOM) é uma convenção multiplataforma e independente de linguagem para representação e interação com objetos em documentos HTML, XHTML e XML. Os nós de cada documento são organizados em uma estrutura de árvore, chamada de árvore DOM.



Localizando Elementos na Página

❖ getElementById

```
var selecionaElemento = document.getElementById("id-do-elemento");
```

❖ getElementsByTagName

```
var list = document.getElementsByTagName("UL")[0];
```

(Pode nos retornar um array de tags UL, portanto selecionamos a primeira encontrada)

❖ getElementsByClassName

```
var x = document.getElementsByClassName("example");
```

❖ getElementsByName

```
var selecionaElemento = document.getElementsByName("namedoinput");
```



Localizando Elementos na Página

❖ **querySelector**

`document.querySelector(".example").style.backgroundColor = "red";` (Retorna sempre o primeiro elemento encontrado)

Mais opções: https://www.w3schools.com/jsref/met_document_queryselector.asp

❖ **querySelectorAll** (Retorna um array com todos os elementos)

`var x = document.querySelectorAll(".example");`

```
function myFunction() {  
    var x = document.querySelectorAll("h2, .a, .b");  
    var i;  
    for (i = 0; i < x.length; i++) {  
        x[i].style.backgroundColor = "red";  
    }  
}
```



Validação de Formulário - Parte 1

➤ No código HTML:

```
<form name="form1">
  <div class="form-group">
    <label for="name">Nome:</label>
    <input id="name" type="text" />
  </div>
  <div class="form-group">
    <label for="email">E-mail:</label>
    <input id="email" type="email" />
  </div>
  <button onclick="formValidate()">Enviar</button>
</form>
```



Validação de Formulário - Parte 2

➤ No código Javascript:

```
function formValidate() {  
    var nome = form1.nome.value; // pega valor do input  
    var email = form1.email.value;  
    if (email == "" || nome == "") {  
        alert("Campos de Nome e E-mail requeridos!");  
        form1.email.focus(); // foca com o mouse no input  
        return false;  
    } else {  
        alert("Cadastrado com sucesso!");  
        return true;  
    }  
}
```



Manipulando Elementos na Página

Utilizando o **innerHTML** ou **innerText** para inserir conteúdos dentro de elementos HTML dinamicamente.

➤ No código Javascript:

```
<script type="text/javascript">
  function nameMyFunction() {
    document.getElementById("test").innerHTML = "Parágrafo modificado";
  }
</script>
```

➤ No código do HTML:

```
<p id="test" onclick="nameMyFunction()">
  Clique e mude o conteúdo do elemento.
</p>
```



Manipulando Elementos na Página

Utilizando o **Value**, para capturar o valor digitado no input:

➤ No código Javascript:

```
<script type="text/javascript">
    function nameMyFunction2() {
        var getInputValue = document.getElementById("name").value;
        alert("Esse é o valor do input: "+ getInputValue);
    }
</script>
```

➤ No código HTML:

```
<label for="name">Nome completo:</label>
<input id="name" type="text" />
<button onclick="nameMyFunction2()">Enviar</button>
```



Eventos

Avisam quando algo acontece, exemplos: (Quando o evento de clicar no botão acontece, realiza determinada ação)

Alguns tipos eventos:

- ❖ Um clique no mouse (**onclick**)
- ❖ O carregamento de uma página ou imagem web (**onLoad**) - utilizado na tag body para executar alguma função assim que a página é carregada ou em tags img para verificar se a mesma já foi carregada.
- ❖ Quando o mouse passa sobre algum element (**onmouseover**, **onmouseenter**, **onmousemove**)
- ❖ Selecionar um campo de entrada em um formulário HTML (**onfocus**)
- ❖ Submeter um formulário HTML (**onsubmit**)
- ❖ Pressionar uma tecla (**onkeydown**)



Eventos

❖ onclick - exemplo 1

➤ No código Javascript:

```
<script type="text/javascript">
  function clickFunction() {
    document.getElementById("test").innerHTML = "Parágrafo modificado";
  }
</script>
```

➤ No código do HTML:

```
<p id="test" onclick="clickFunction()">
  Clique e mude o conteúdo do elemento.
</p>
```



Eventos

❖ onLoad - exemplo 2

➤ No código Javascript:

```
<script type="text/javascript">
  function loadFunction() {
    alert("Página carregada, portanto executando esta ação!");
  }
</script>
```

➤ No código do HTML:

```
<body onLoad="loadFunction()">
  // aqui haveria o código de conteúdo
</body>
```



Eventos

❖ onmouseover - exemplo 3

➤ No código Javascript:

```
<script type="text/javascript">
  function mouseOverFunction() {
    var x = 0;
    document.getElementById("number").innerHTML = x;
  }
</script>
```

➤ No código do HTML:

```
<p onmouseover="mouseOverFunction()">
  Passe o mouse em cima para ver a ação. <span id="number"></span>
</p>
```



Eventos

Exercícios:

Use o evento `onmouseover` para chamar uma função que deverá mostrar em um elemento na tela o número de vezes que o evento é chamado.



Criando elementos dinamicamente

```
//Selecionando o elemento que irá receber o novo elemento  
var conteudo = document.getElementById('conteudo');
```

```
//Criando um novo elemento  
var text = document.createElement('span');  
text.innerText = 'Eu não estava aqui antes';
```

```
//Usando o método append para colocar o elemento que criamos  
na div selecionada pela variável conteúdo que selecionamos  
conteudo.appendChild(text);
```



Criando elementos dinamicamente

Exercícios:

Dentro de um elemento `<div id="dinamic">` crie um elemento `` com o seguinte texto: “Lugar de mulher é onde ela quiser!”. Esse texto deve aparecer utilizando o seguinte evento javascript: `onLoad`.



Considerações

Javascript é uma linguagem que requer aprendizado constante. Diariamente são criados novos frameworks baseado na linguagem e eles com certeza farão parte do seu dia a dia.

Alguns links úteis:

<https://developer.mozilla.org/pt-BR/docs/Web/JavaScript>

<https://css-tricks.com/>

<https://tableless.com.br/>



Agradecemos sua presença
e esperamos que tenham gostado!

