



Usabilidade Desenvolvimento WEB e Mobile



JS

Java Script

Introdução

- Linguagem Interpretada
- Executada no lado cliente (Executada pelo navegador).
- Sua função é tornar páginas WEB interativas/dinâmicas.

Inserção

- Inserção em uma página HTML:
 - TAG `<script>` dentro das tags `<head>` ou `<body>`
 - Em um arquivo externo usando a TAG `<script>` e definindo o caminho ou nome do arquivo no atributo "src"

Inserção

```
<!DOCTYPE html>  
<html>  
  <head>  
    <style>  
      /* ... código JS ... */  
    </style>  
  </head>  
  <body>  
  </body>  
</html>
```

Inserção

- Inserção em um arquivo externo de extensão JS.
- Método recomendado.
- Evita replicações e inconsistências.
- Permite organizar em diretórios.

Inserção

```
<!DOCTYPE html>  
<html>  
  <head>  
    <script src="nome-arquivo.js"></script>  
  </head>  
  <body>  
  </body>  
</html>
```

Comentários

Única Linha:

```
// nome = "Java Script"
```

Múltiplas linhas:

```
/* nome = "HTML"  
   . . .  
   . . .  
   . . .  
   nome = "CSS"  
*/
```


Variáveis

Formas de declarar variáveis:

- Usando `var`;
- Usando `let`;
- Usando `const`;
- Usando nada.

Variáveis

```
var x = 10
```

```
let y = 10
```

```
const z = 10
```

```
w = 10
```

Variáveis

	var	let	const
Escopo Global	✓	×	×
Escopo de função	✓	✓	✓
Escopo de bloco	×	✓	✓
Declaração irrestrita	✓	✓	×
Atribuição irrestrita	✓	×	×
Sem declaração	✓	×	×

Tipagem dinâmica

- Não é necessário definir o tipo da variável.
- Para visualizar um tipo use: `typeof`
- Problemas:

Tipagem dinâmica

```
x = 1
```

```
x = 1.5
```

```
x = "Java Script"
```

```
x = true
```

```
x = false
```

```
x = null
```

```
x = function(){} 
```

Coerção Automática

Coerção automática de tipos:

```
1 + "1" //11
```

```
1 - "1" //0
```

```
1 == "1" //true
```

Quando há um operador entre variáveis de tipos distintos, uma delas é convertida para o tipo da outra variável.

Coerção Automática

O que fazer para evitar coerção automática:

Evitar:

```
1 == "1" //false
```

Usar:

```
1 === "1" //true
```

Sempre que possível, faça conversões:

```
var c = Number(a) + Number(b)
```

```
var cpf = String(04568409032)
```

Tipos Existentes

- string:
- number
- boolean
- function: definido pela sintaxe
- object
- undefined

Tipos Existentes

- Nunca compare dois objetos, eles nunca serão iguais, compare um valor do objeto.
- Um array é um object
- Para saber se uma variável é um Array use:
 - `Array.isArray(nome_variavel)`
- Objetos e Array tem valores de referência.

Referência e Valor

pass by reference




pass by value




Referência e Valor

Pass by reference

Square square = 

fillColor()

Pass by value

Square square = 

fillColor()



Referência e Valor

Value



Reference



Operadores Aritméticos

Adição	+
Subtração	-
Multiplicação	*
Exponenciação	**
Dvisão	/
Modulo	%
Incremento	++
Decremento	--

Operadores Atribuição

=	a = b
+=	a = a + b
-=	a = a - b
*=	a = a * b
/=	a = a / b
%=	a = a % b
**=	a = a ** b

Operadores Lógicos

==	igual
!=	diferente
&&	and
 	or
>, >=	maior que, maior ou igual que
<, <=	menor que, menor ou igual que

Condicional

```
if (condição) {  
    // bloco de instruções  
}
```


Condicional

```
if (condição) {  
    // bloco de instruções  
}else{  
    //bloco de instruções  
}
```

Condicional

```
if (condição) {  
    // bloco de instruções  
}else if (condição){  
    //bloco de instruções  
}else{  
    //bloco de instruções  
}
```

Condicional

```
switch(expressão) {  
    case x:  
        //bloco de instruções  
        break;  
    case y:  
        //bloco de instruções  
        break;  
    default:  
        //bloco de instruções  
}
```

Repetição

```
for (expressão 1; expressão 2; expressão 3) {  
    //bloco de instruções  
}
```

Repetição

```
for (objeto of objetos) {  
    //bloco de instruções  
}
```

Repetição

```
for (chave in objeto) {  
    //bloco de instruções  
}
```

Repetição

```
while (condição) {  
    //bloco de instruções  
}
```

Repetição

```
do {  
    //bloco de instruções  
}  
while (condição);
```


Funções

```
function nomeFuncao(parametro1, parametro2)  
    //bloco de instruções  
}
```

Objetos

```
let pessoa = {  
  nome: "Anne",  
  sobrenome: "Willians",  
  idade: 32,  
  dataNascimento: "01/10/1990"  
}
```

Objetos

```
let pessoa = {}  
pessoa.nome: "Anne";  
pessoa.sobrenome: "Willians";  
pessoa.idade: 32;  
pessoa.dataNascimento: "01/10/1990";  
}
```

Eventos

onchange	Um elemento HTML foi alterado
onclick	Ao clicar
onmouseover	Ao passar o mouse
onmouseout	Ao mover o mouse para longe de um elemento
onkeydown	Ao pressionar uma tecla do teclado
onload	Ao carregar a página

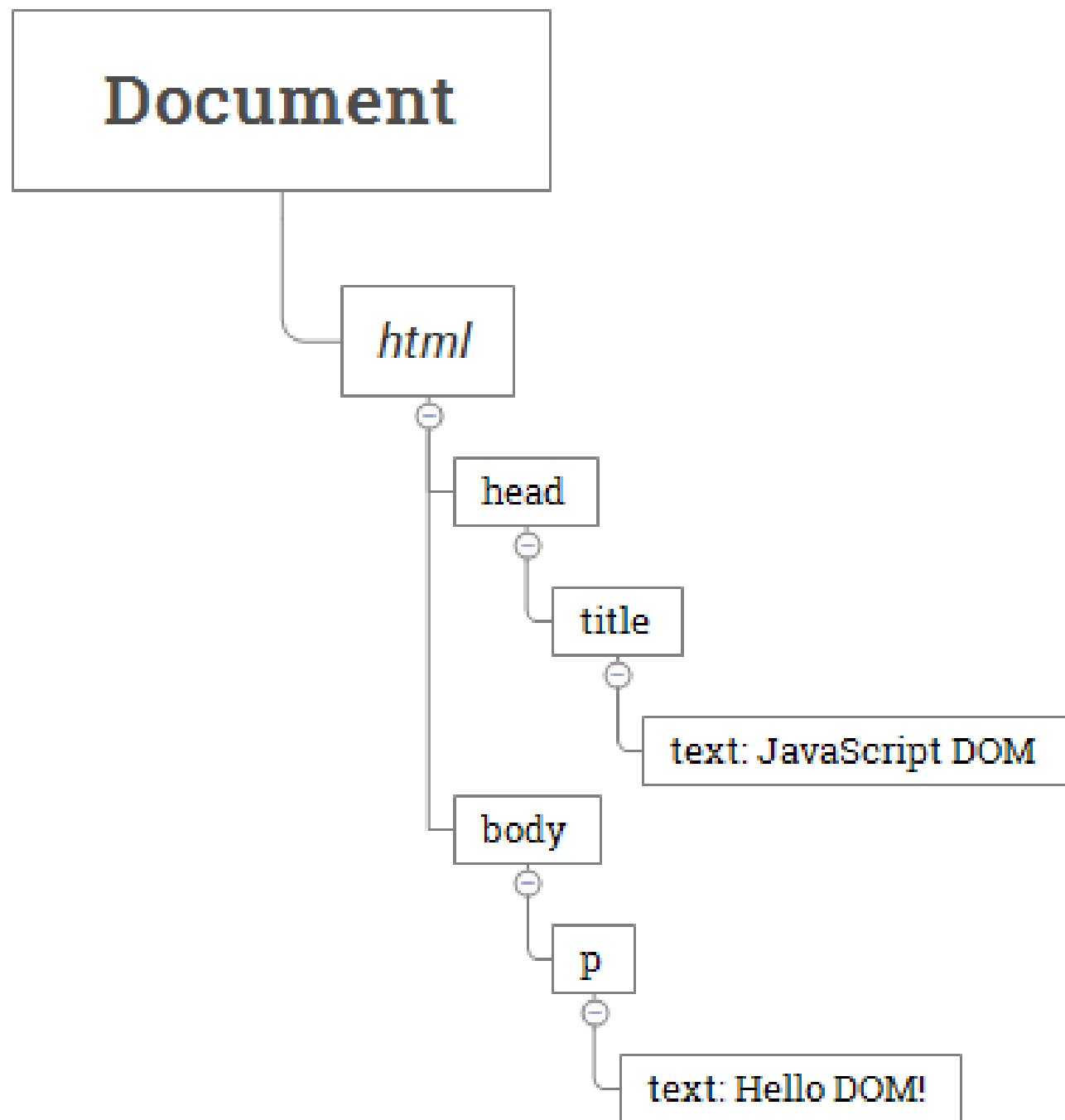
Log

Inserir mensagens de log no console do navegador:

```
console.log("mensagem")
```

DOM (*Document Object Model*)

- É uma API para manipulação de documentos HTML.
- O documento é representado como uma árvore.



DOM (*Document Object Model*)

Como selecionar um elemento:

```
<html>
<body>
<p id="paragrafo"></p>
<script>
document.getElementById("paragrafo").innerHTML = "Hello DOM!";
</script>
</body>
</html>
```


DOM (*Document Object Model*)

```
<html>
  <head>
    <title>JavaScript DOM</title>
  </head>
  <body>
    <p>Hello DOM!</p>
  </body>
</html>
```

DOM (*Document Object Model*)

Como selecionar um elemento:

- Por ID:

`getElementById()`

- Por tag:

`getElementsByTagName()`

- Por Classe:

`getElementsByClassName()`

- Por nome:

`getElementByName()`

DOM (*Document Object Model*)

Selecionar elemento por Seletores CSS

```
querySelectorAll()
```

DOM (*Document Object Model*)

Alterar um elemento HTML:

Propriedade:

```
element.innerHTML = new html content
```

```
element.attribute = new value
```

```
element.style.property = new style
```

Método:

```
element.setAttribute(attribute, value)
```

DOM (*Document Object Model*)

Criar e Remover elemento HTML:

Criar Elemento:

```
document.createElement(element)
```

Remover Elemento:

```
document.removeChild(element)
```

Adicionar um elemento dentro de um elemento pai:

```
document.appendChild(element)
```

Substituir elemento:

```
document.replaceChild(new, old)
```

Erros Comuns

- **RangeError:**
 - Ocorre quando um número informado não é compatível com o intervalo de números válidos.
- **ReferenceError:**
 - Os erros de referência ocorrem quando ocorre uma referência inválida, ou seja, se tenta acessar uma variável/metodo/propriedade que não existe.
- **SyntaxError:**
 - Ocorrem quando é detectado um erro de sintaxe na programação.
- **TypeError:**
 - Estes erros ocorrem quando um valor tem um tipo diferente daquele que é esperado.
- **URIError:**
 - Ocorre quando passamos uma URI inválida.

Erros Comuns

- Uncaught TypeError: Cannot Read Property:
 - Este erro ocorre quando você quer acessar uma propriedade ou método de um objeto que não existe.

Erros Comuns

- **TypeError: 'undefined' Is Not an Object:**
 - Este erro é o mesmo que foi descrito acima, a diferença é que esta mensagem de erro aparece no Chrome.

Erros Comuns

- **TypeError: Object Doesn't Support Property:**
 - Se você utiliza o navegador Internet Explorer, este erro ocorre quando você chama um método que não foi definido.

Erros Comuns

- **TypeError: 'this.<>' Is Not a Function:**
 - Este erro ocorre quando você chama uma função que não foi definida, esta mensagem pode aparecer no Chrome e no Firefox.

Erros Comuns

- Uncaught RangeError
 - Este erro ocorre no Chrome por dois motivos, o primeiro pode ser uma função recursiva que não terminou, e o segundo, um valor fora do intervalo esperado para uma função.

Funções úteis

onebitc 1/2

{JavaScript} Manipulando Arrays

 `.map(☞☞) →` 

 `.filter(☞) →` 

 `.find(☞) →` 



 `.indexOf(☞) →` 3

 `.fill(1, ☞) →` 















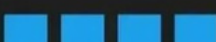









 `.some(☞) →` True

 `.every(☞) →` False

Funções úteis



JAVASCRIPT ARRAY METHODS CHEATSHEET

 .push()	→	
 .unshift()	→	
 .pop()	→	
 .shift()	→	
 .map( =>  = )	→	
 .filter()	→	
 .reverse()	→	
 .at(2)	→	
 .slice(2)	→	

@alenvarazdinac

Save it, so you don't lose it.

Funções úteis

```
const today = new Date()  
today          // 2020-08-09T01:40:51.017Z  
today.getDate()    // 9  
today.getDay()     // 0 sunday  
today.getMonth()   // 7 (0 is jan)  
today.getFullYear() // 2020  
today.getYear()    // 120 ?  
today.getHours()   // 11  
today.getMinutes() // 40  
today.getSeconds() // 51  
today.getMilliseconds() // 24  
today.getTime()    // 1596937251025  
today.getTimezoneOffset() // -600
```

AJAX

- **Atualizar uma página da web sem recarregar a página.**
- **Enviar dados para um servidor web - em segundo plano.**

AJAX

- Atualizar uma página da web sem recarregar a página.
- Enviar dados para um servidor web - em segundo plano.

AJAX

Exemplo:

[https://www.w3schools.com/xml/tryit.asp?
filename=tryajax_suggest_php](https://www.w3schools.com/xml/tryit.asp?filename=tryajax_suggest_php)

JQUERY

- jQuery é uma biblioteca JavaScript com o lema: "escreva menos, faça mais".
- O jQuery pega muitas tarefas comuns que requerem muitas linhas de código JavaScript para serem realizadas e as agrupa em métodos que você pode chamar com uma única linha de código.
- jQuery também simplifica muitas das coisas do JavaScript, como chamadas [AJAX](#) e manipulação de [DOM](#).
- A biblioteca jQuery contém os seguintes recursos:
 - Manipulação de HTML/DOM.
 - Manipulação de CSS.
 - Métodos de evento HTML.
 - Efeitos e animações.
 - AJAX.
 - Serviços de utilidade pública.



COMO USAR

Para incluir em seu projeto:

```
<head>  
<script  
src="https://ajax.googleapis.com/ajax/libs/jquery/3.6.4/j  
query.min.js"></script>  
</head>
```

SINTAXE

A sintaxe jQuery é feita sob medida para selecionar elementos HTML e executar alguma ação no(s) elemento(s).

A sintaxe básica é: \$(seletor). ação ()

- Um sinal \$ para definir/acessar jQuery
- A (seletor) para "consultar (ou localizar)" elementos HTML
- Uma ação jQuery () a ser executada no(s) elemento(s)

```
$(document).ready(function(){
```

```
    // implementação.
```

```
});
```

PRINCIPAIS SELETORES

Por elementos:

```
$( "p" )
```

Por ID:

```
$( "#test" )
```

Por classe:

```
$( ".test" )
```

Principais Funções

`ready()`: elemento carregado.

`hide()`: oculta elemento.

`show()`: mostra elemento.

`click()`: quando há um clique.

`dblclick()`: duplo clique.

`hover()`: quando o mouse está sob o elemento

Exemplos - Atributos

Executa uma função quando o documento HTML é carregado.

```
$(document).ready(function(){
```

```
    // implementação.
```

```
});
```

ou

```
$(function(){
```

```
    // implementação.
```

```
});
```

Exemplos - Adicionar classe CSS

Quando há um evento de clique no botão uma classe CSS é adicionada a div selecionada.

```
$( "button" ).click(function(){  
    $( "#div1" ).addClass( "important blue" );  
});
```


Referências

FLANAGAN, David. JavaScript: o guia definitivo. Bookman Editora, 2004.

<https://www.alura.com.br/artigos/tipagem-dinamica-com-javascript>

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

<https://api.jquery.com/>