JavaScript Parte 2

PROFESSOR WELLINGTON TELLES CUNHA

JavaScript Objetos

JavaScript é uma linguagem OO

Com isto, algumas classes utilitárias padrões estão disponíveis, as quais possuem métodos e propriedades:

String: manipulação de strings no script

JavaScript Objetos

- Objetos referem-se a janelas, documentos, imagens, tabelas, formulários, botões ou links, etc.
- Os objetos devem ser nomeados
- Objetos tem propriedades que atuam como modificadores

JavaScript Objetos - Propriedades

Propriedade são atributos de objetos

Propriedades de objetos são definidas pelo uso do nome do objeto, ponto, e o nome da propriedade

Exemplo:

- `document.bgcolor`
- `document` é o **objeto**
- `bgcolor` é a **propriedade**

JavaScript Métodos

Métodos são ações aplicadas para objetos

Métodos são o que os objetos podem fazer

Exemplo:

- `document.write (“Hello World!”)`
- `document` é **objeto**
- `write` é um **método**

Funções

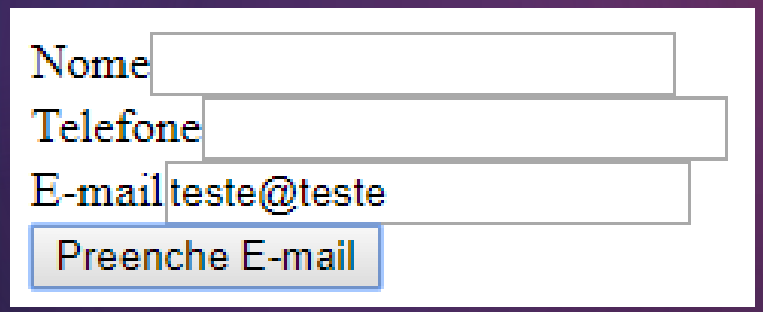
Funções são declarações nomeadas em acordo com a tarefa

```
function FaçaIsso() {
}
```

Os colchetes contém as declarações da função

JavaScript tem funções **built-in**, e você pode criar a sua própria.

Funções e Objetos (preenche_js.html)




```
<!DOCTYPE html>
<html>
<head>
<meta content="text/html; charset=utf-8" http-equiv="Content-Type">
<title>Formulário</title>
<script>
function valor(){
     document.cliente.email.value = 'teste@teste';
}
</script>
</head>
<body>
     <form name="cliente">
          Nome<input name="nome"> <br />
          Telefone<input name="telefone"> <br />
          E-mail<input name="email"> <br />
          <button type="button" onclick="valor()">Preenche E-
mail</button>
     </form>
</body>
</html>
```

Valores

- Tipos
  - Númericos: 1,2,3, etc
  - String: caracteres marcados por aspas
  - Boolean: true e false
  - Objeto: image, form
  - Funções: validação, FazerQualquerCoisa

Valores

Variáveis contêm valores e usam o sinal de igual para especificar seu valor.

Variáveis são criados pela declaração usando o comando var com ou sem um valor inicial.

Exemplos:

- `var month;`
- `month = April;`
- `var month = April;`

Variáveis

- variáveis são declaradas com a palavra-chave **var** (diferencia maiúsculas de minúsculas)
- tipos não são especificados, mas JS tem tipos ("fracamente digitado")
  - Número, Booleana, Sequência de caracteres, Matriz, Objeto, Função, Nulo, Indefinido
  - pode descobrir o tipo de uma variável chamando **typeof**

Variáveis

```
var name = expression;
```

```
var clientName = "Connie Client";
```

```
var age = 32;
```

```
var weight = 127.4;
```

Tipo numérico

- inteiros e números reais são do mesmo tipo (sem int vs. duplo)
- mesmos operadores: + - * /% ++ - = + = - = * = / =% =
- precedência semelhante ao Java
- muitos operadores convertem automaticamente os tipos: "2" * 3 é 6

Tipo Numérico

```
var valorA = 99;
var Media = 2.8;
var creditos = 5 + 4 + (2 * 3);
```

Tipo Numérico (multi_js.html)

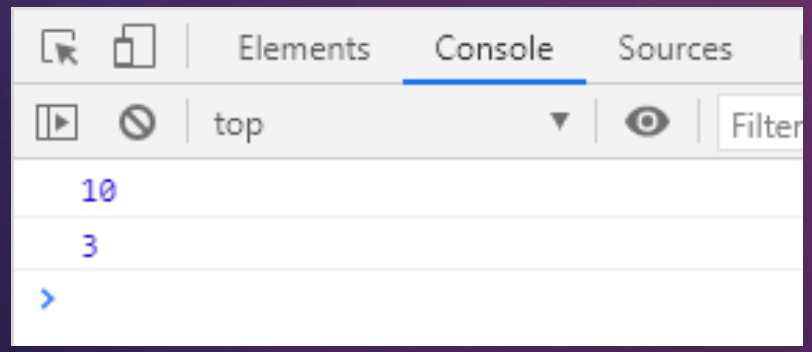
```
<!DOCTYPE html>
<html>
<head>
<meta content="text/html; charset=utf-8" http-equiv="Content-Type">
<title>Formulário</title>
<script>
function contas(){
     document.matematica.soma1.value = "2"+3;
     document.matematica.soma2.value = 2+3;
     document.matematica.multi.value = "2"*3;
}
</script>
</head>
<body>
     <form name="matematica">
          Soma1:<input name="soma1"><br>
          Soma2:<input name="soma2"><br>
          Múltiplica:<input name="multi"><br>
     </form>
     <button type="button" onclick="contas()">Contas</button>
</body>
</html>
```

Objetos Matemáticos

métodos: abs, ceil, cos, piso, log, max, min, pow, aleatório, redondo, sin, sqrt, tan

Propriedades: e(exponencial), PI

Objeto de matemática (mate_js.html)




```
<!DOCTYPE html>
<html>
<head>
<meta content="text/html; charset=utf-8" http-equiv="Content-Type">
<title>Matemática</title>
<script>
    var sorteia1a10 = Math.floor(Math.random() * 10 + 1);
    var three = Math.floor(Math.PI);
    console.log(sorteia1a10);
    console.log(three);
</script>
</head>
<body>
</body>
</html>
```

Valores especiais: nulo e indefinido

indefinido: não foi declarado, não existe

null: existe, mas foi especificamente atribuído um valor vazio ou nulo

Valores especiais: nulo e indefinido

```
var ned = null;
var benson = 9;
var caroline;
// at this point in the code,
// ned is null
// benson's 9
// caroline is undefined


var x;
if (typeof x === "undefined") {
  txt = "x is undefined";
} else {
  txt = "x is defined";
}
```

Operadores lógicos

- `> <> = <= && || ! ==! = ===! ==`

- a maioria dos operadores lógicos converte automaticamente os tipos:
  - `5 <"7" é verdadeiro`
  - `42 == 42.0 é verdade`
  - `42 === 42.0 é falso`
  - `"5.0" == 5 é verdade`

- === e !== são testes rigorosos de igualdade; verifica o tipo e o valor
  - `"5.0" === 5 é falso`

Tipo String

```
var s = "Connie Client";
var fName = s.substring(0, s.indexOf(" ")); // "Connie"
var len = s.length; // 13
var s2 = 'Melvin Merchant';
```

Tipo String

- métodos: charAt, charCodeAt, fromCharCode, indexOf, lastIndexOf, substituir, dividir, substring, toLowerCase, toUpperCase
  - charAt retorna uma string de uma letra (não há nenhum tipo de caractere)
- propriedade length (não é um método como em Java)
- Strings podem ser especificados com '"' ou "'"
- concatenação com +:
  - `1 + 1 é 2, mas "1" + 1 é "11"`

Mais sobre String

- sequências de escape se comportam como em Java: \ '\ "\ & \ n \ t \\
- convertendo entre números e Strings:

```
var count = 10;
var s1 = "" + count; // "10"
var s2 = count + " bananas, ah ah ah!"; // "10 bananas, ah ah ah!"
var n1 = parseInt("42 is the answer"); // 42
var n2 = parseFloat("booyah"); // NaN
```

Mais sobre String

- acessando as letras de uma string:
- var firstLetter = s[0]; // falha no IE
- var firstLetter = s.charAt(0); // funciona no IE
- var lastLetter = s.charAt(s.length - 1);

Dividindo strings: dividir e juntar

- split divide uma string em uma matriz usando um delimitador
- também pode ser usado com expressões regulares (visto mais tarde)
- junção mescla um array em uma única string, colocando um delimitador entre eles

Dividindo strings: dividir e juntar


```
var s = "the quick brown fox";
var a = s.split(" "); // ["the", "quick", "brown", "fox"]
a.reverse(); // ["fox", "brown", "quick", "the"]
s = a.join("!"); // "fox!brown!quick!the"
```

if / else statement (o mesmo que Java)

- estrutura idêntica à instrução if / else do Java
- JavaScript permite quase qualquer coisa como condição

if / else statement

```
if (condition) {
   statements;
} else if (condition) {
   statements;
} else {
   statements;
}
```

Tipo booleano

qualquer valor pode ser usado como um booleano

valores "falsos": 0, 0.0, NaN, "", null e indefinido

valores "verdadeiros": qualquer outra coisa

convertendo um valor em um booleano explicitamente:

```
var boolValue = Booleano (outroValor);
var boolValue = !! (otherValue);
```

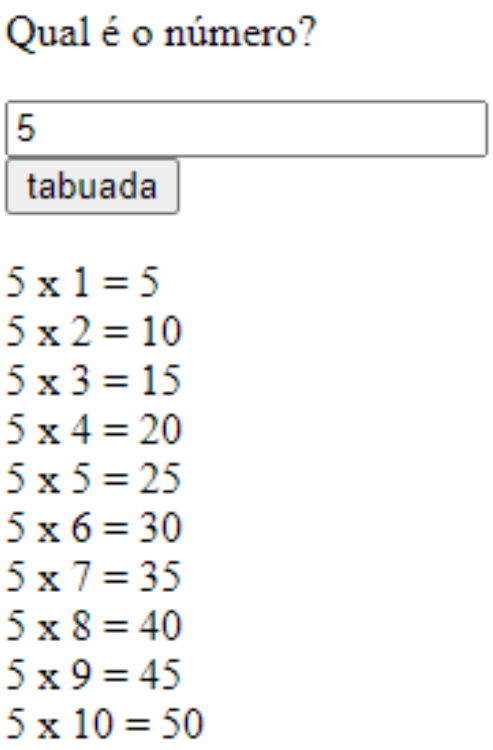
Tipo booleano


```
var banana = true;
var ieehbom = "IE6" > 0; // false
if ("desenvolvimento Web é bom") { /* true */ }
if (0) { /* false */ }
```

for loop


```
var sum = 0;
for (var i = 0; i < 100; i++) {
   sum = sum + i;
}
```

for loop (tabuada_js.html)




```
<!DOCTYPE html>
<html>
<head>
<meta content="text/html; charset=utf-8" http-equiv="Content-Type">
<title>Tabuada</title>
<script>
	function calctabuada(){
		n = document.tabuada.numero.value;
		console.log (n);
		document.getElementById("resultado").innerHTML = '';
		for (i=1; i<11; i++){
			document.getElementById("resultado").innerHTML += n + ' x '+i+' = '+n*i+'<br>';
		}
	}
</script>
</head>
<body>
	<p>Qual é o número?</p>
	<form name="tabuada">
		<input name="numero"><br>
		<input type="button" onclick="calctabuada()" value="tabuada" />
	</form>
	<p id="resultado"></p>
</body>
</html>
```

While loops


```
while (condition) {
   statements;
}

do {
   statements;
} while (condition);
```

Matrizes

```
var name = []; // matriz vazia
var name = [valor, valor, ..., valor]; // pré definida
name[index] = valor; // armazena um elemento
```

Matrizes


```
var ducks = ["Huey", "Dewey", "Louie"];
var stooges = []; // stooges.length is 0
stooges[0] = "Larry"; // stooges.length is 1
stooges[1] = "Moe"; // stooges.length is 2
stooges[4] = "Curly"; // stooges.length is 5
stooges[4] = "Shemp"; // stooges.length is 5
```

Métodos de matriz


```
var a = ["Stef", "Jason"]; // Stef, Jason
a.push("Brian"); // Stef, Jason, Brian
a.unshift("Kelly"); // Kelly, Stef, Jason, Brian
a.pop(); // Kelly, Stef, Jason
a.shift(); // Stef, Jason
a.sort(); // Jason, Stef
```

Métodos de matriz

- array serve como muitas estruturas de dados: list, queue, stack, ...
- métodos: concat, join, pop, push, reverse, shift, fatia, sort, splice, toString, unshift
  - push e pop adicionar / remover de volta
  - Unshift e shift adicionar / remover da frente
  - shift e pop retornam o elemento que é removido

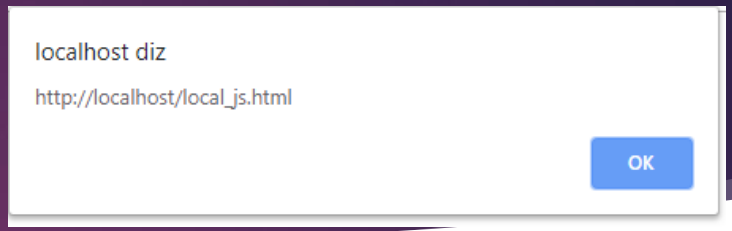
JavaScript Objetos


```
/* Exemplo de função de script que calcula o tamanho de uma string */
function tamanho(msg)
{
  return msg.length;
}
```

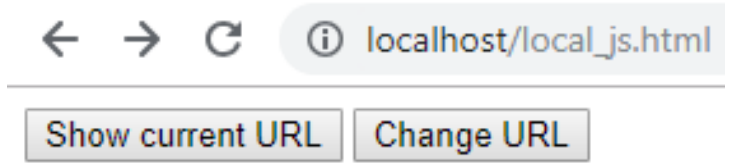
JavaScript
Objetos - Exemplo


```
/* Exemplo de função de script que manipula objetos de JavaScript */
function exemplo_objetos()
{
    var d=new Date();
    var dias=new Array(7);
    dias[0]="Domingo"; dias[1]="Segunda"; dias[2]="Terça";
    dias[3]="Quarta"; dias[4]="Quinta"; dias[5]="Sexta";
    dias[6]="Sábado";
    document.write("Hoje é" + dias[d.getDay()]);
    document.write("Dia qualquer: " + dias[Math.round(Math.random()*6)]);
    document.write(navigator.appCodeName);
}
```

Objetos: Exemplo com Window (window_js.html)




```
<html>
<head>
    <script type="text/javascript">
    function currLocation() {
        alert(window.location);
    }
    function newLocation() {
        window.location="http://www.google.com.br";
    }
    </script>
</head>
<body>
    <input type="button" onclick="currLocation()" value="Show current URL">
    <input type="button" onclick="newLocation()" value="Change URL">
</body>
</html>
```



HTML DOM (Document Object Model)

DEFINE UM PADRÃO PARA ACESSO A ELEMENTOS HTML

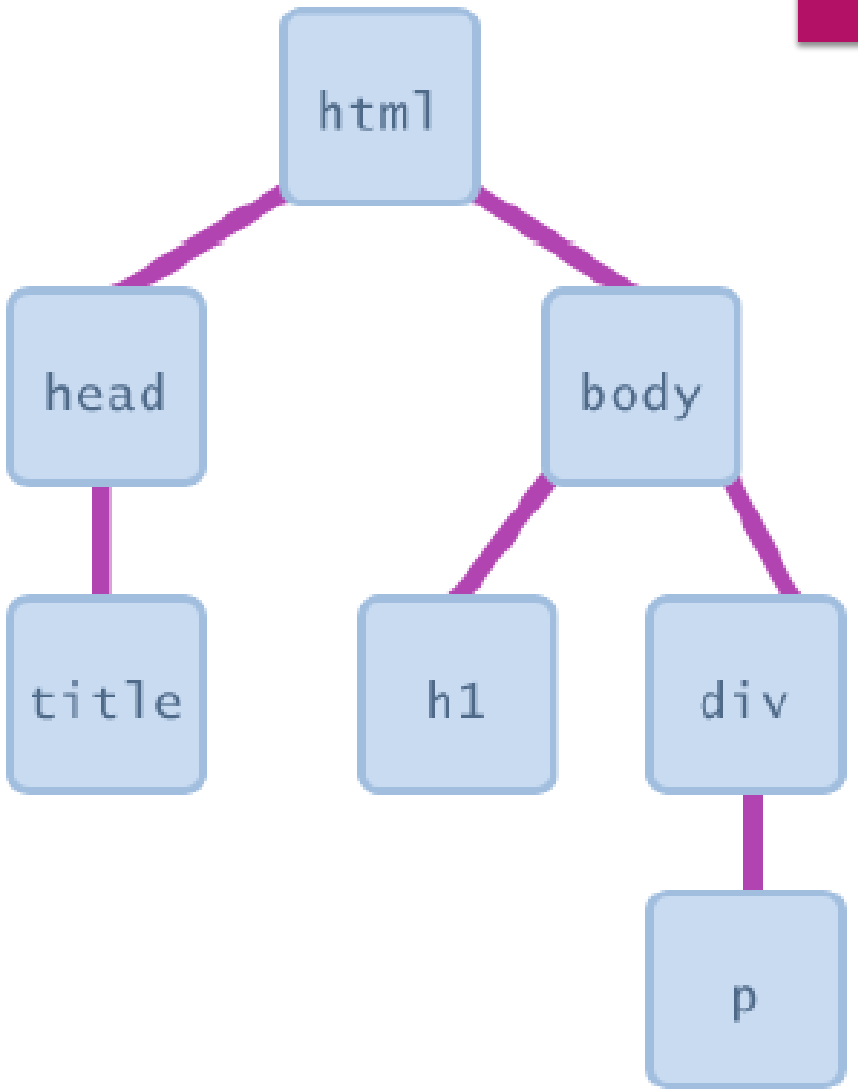
O DOM APRESENTA UM DOCUMENTO HTML COMO UMA ESTRUTURA EM ÁRVORE

MODELO DE DOCUMENTO POR OBJETOS

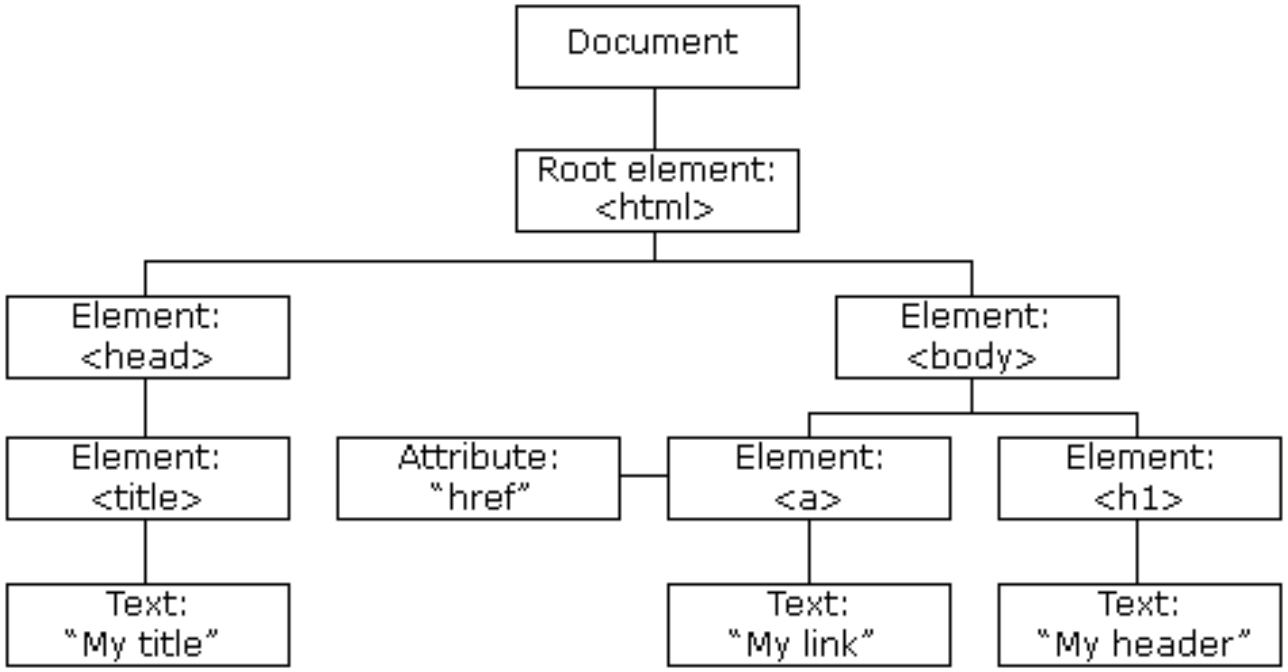
Document Object Model (DOM)

Modelo de objeto de documento

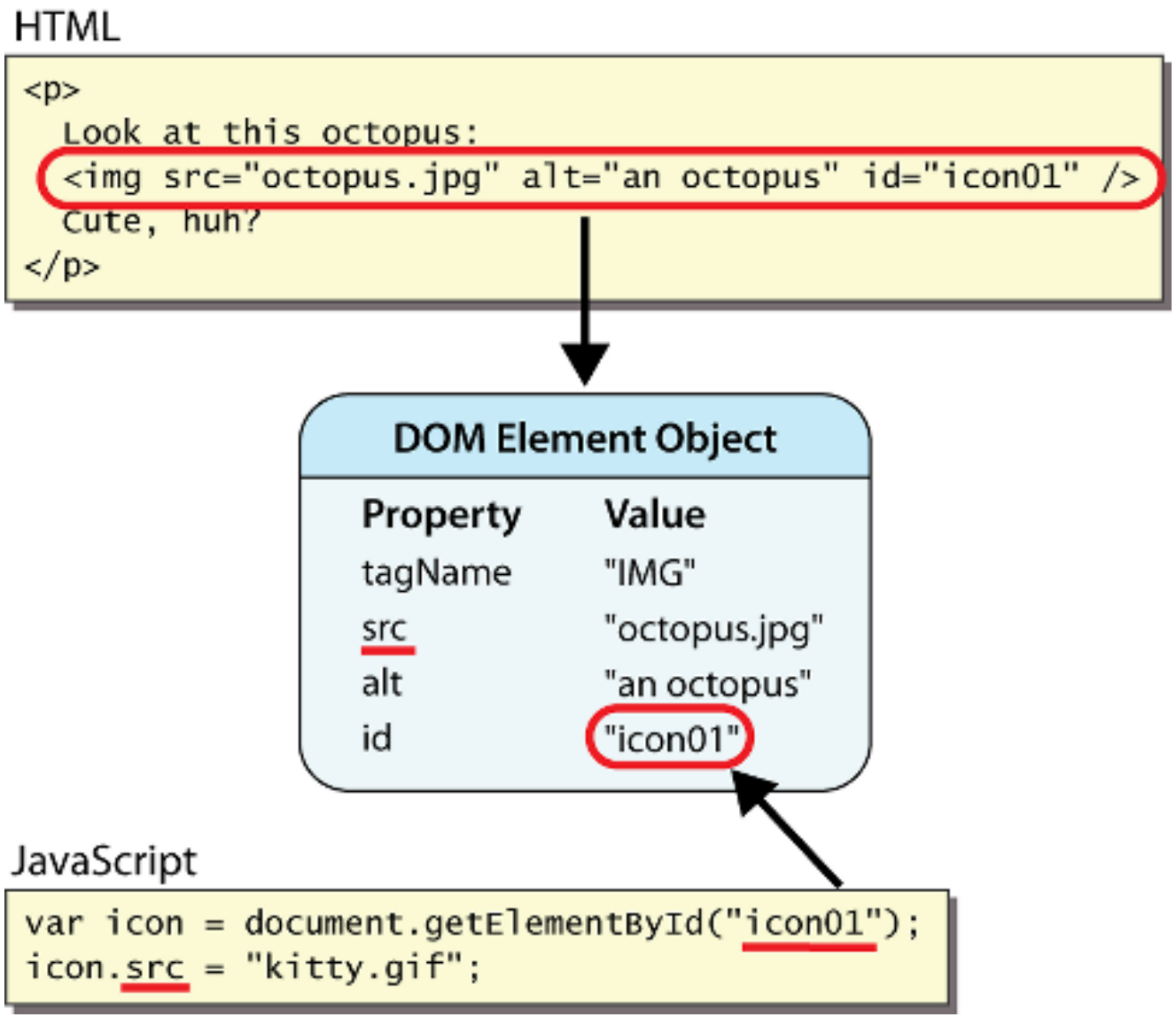
- a maioria do código JS manipula elementos em uma página HTML
- podemos examinar o estado dos elementos
  - por exemplo. ver se uma caixa está marcada
- podemos mudar de estado
  - por exemplo. insira um novo texto em um div
- podemos mudar estilos
  - por exemplo. fazer um parágrafo vermelho



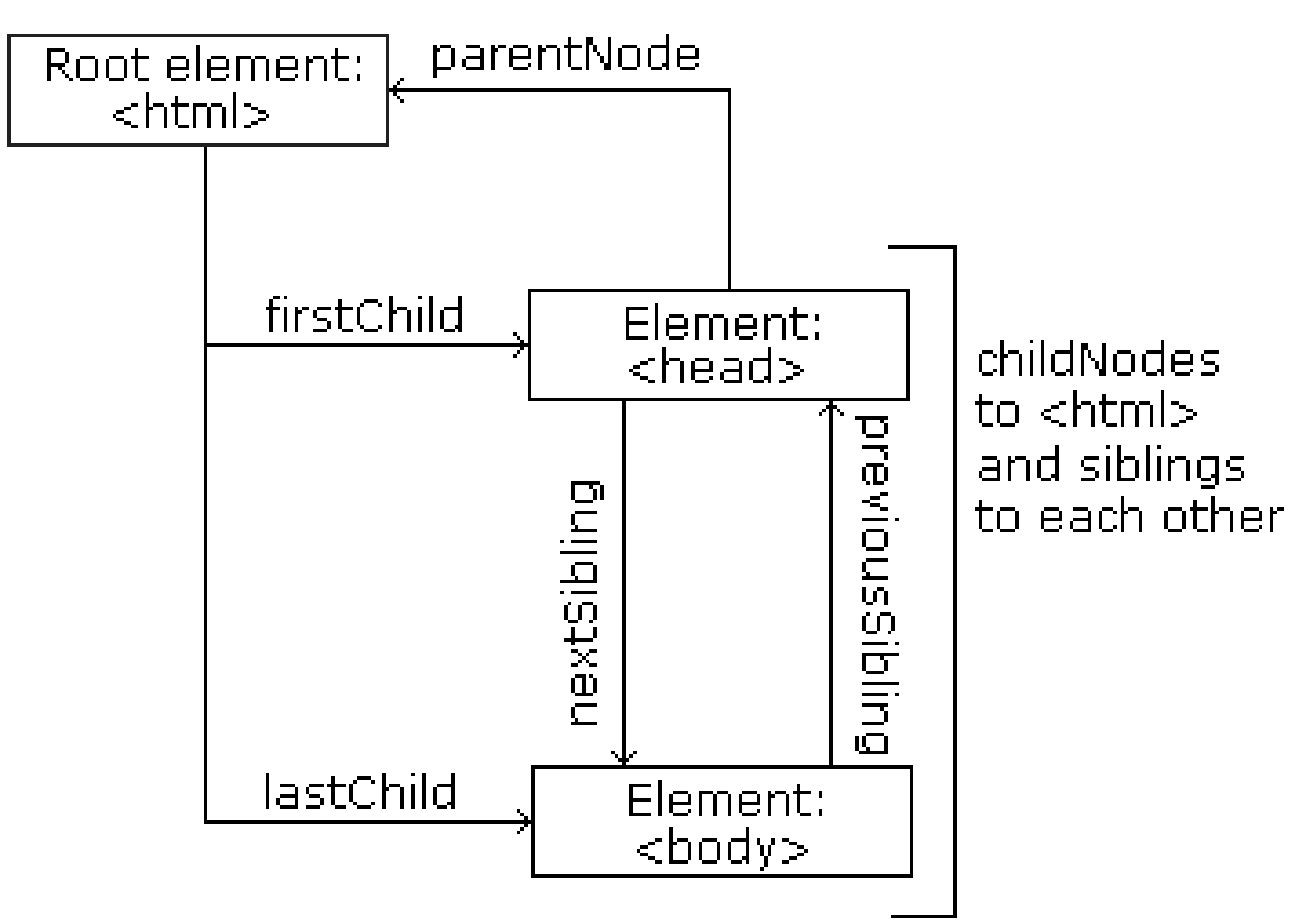
HTML DOM Árvore do documento HTML



DOM element objects

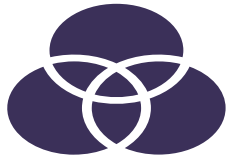


HTML DOM Relacionamento / Nó



HTML DOM - API

É definida por um conjunto de propriedades e métodos



Algumas propriedades DOM

x.innerHTML: o valor text de x

x.nodeName: o nome do elemento x

x.nodeValue: o valor de x

x.nodeType: o tipo de x *(1 – elemento; 2 – atributo; 3 – texto; ...)*

x.parentNode: o nó pai de x

x.childNodes: os nós filhos de x

x.attributes: os nós atributos de x

HTML DOM - API

- Alguns métodos DOM
  - **x.getElementById(*id*)**: obtém o elemento com o *id* fornecido
  - **x.getElementsByTagName(name)**: obtém todos os elementos com a tag *name*
  - **x.appendChild(*node*)**: insere um nó filho *node* em x
  - **x.removeChild(*node*)**: remove o nó filho *node* de x

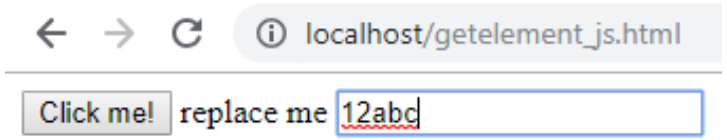
Acessando elementos: document.getElementById

```
var name = document.getElementById("id");
```

Acessando elementos: document.getElementById (dom_js.html)


```
<button onclick="changeText();">Click me!</button>
<span id="output">replace me</span>
<input id="textbox" type="text" />
```

HTML

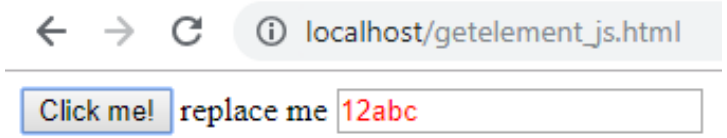



```
function changeText() {
    var span = document.getElementById("output");
    var textBox = document.getElementById("textbox");

     textbox.style.color = "red";

}
```

JS



Acessando elementos: document.getElementById

- document.getElementById retorna o objeto DOM para um elemento com um determinado ID
- pode alterar o texto dentro da maioria dos elementos, definindo a propriedade innerHTML
- pode alterar o texto em controles de formulário, definindo a propriedade de valor

HTML DOM – API
Procurando Elementos HTML


```
Procurando elemento HTML pelo Id:

Exemplo:

    var x=document.getElementById("intro");

Procurando elemento HTML pelo nome da TAG:

Exemplo:

    var x=document.getElementById("main");
    var y=x.getElementsByTagName("p");
```

Mudando o estilo do elemento: element.style

Atributo	Propriedade ou objeto de estilo
Cor	color
Preenchimento	padding
Cor de fundo	backgroundColor
Largura da borda de cima	borderTopWidth
Tamanho da fonte	fontSize
Família da fonte	fontFamily

Embelezar (dom2_js.html)


```
function changeText() {
    //grab or initialize text here

    // font styles added by JS:
    text.style.fontSize = "13pt";
    text.style.fontFamily = "Comic Sans MS";
    text.style.color = "red"; // or pink?
}
```

Usando arquivos separados em JavaScript

- Vincular pode ser vantajoso se muitas página usarem o mesmo script
- Use o elemento SRC (Source=fonte) para vincular o arquivo Script

```
<script src="meujavascript.js" language="JavaScript1.2" type="text/javascript">
</script>
```

Vinculando a um arquivo JavaScript: `script`


```
<script src="filename" type="text/javascript"></script>
```

- tag de script deve ser colocada na cabeça da página HTML
- código de script é armazenado em um arquivo .js separado
- O código JS pode ser colocado diretamente no corpo ou na cabeça do arquivo HTML (como CSS)
  - mas isso é um estilo ruim (deve separar conteúdo, apresentação e comportamento

HTML DOM – API (api_js.html)


```
<!DOCTYPE html>
<html>
<head>
<meta content="text/html; charset=utf-8" http-equiv="Content-Type">
<title>Hello 2</title>
</head>
<body>
     <p id="intro">Hello World!</p>
     <script type="text/javascript">
          txt=document.getElementById("intro").innerHTML;
          document.write("<p>Texto do parágrafo intro: " + txt + "</p>");
     </script>
</body>
</html>
```

HTML DOM – API (api2_js.html)


```
<!DOCTYPE html>
<html>
<head>
<meta content="text/html; charset=utf-8" http-equiv="Content-Type">
<title>Hello 3</title>
</head>
<body>
     <p id="intro">Hello World!</p>
     <script type="text/javascript">
        txt=document.getElementById("intro").childNodes[0].nodeValue;
        document.write("<p>Texto do parágrafo intro: " + txt + "</p>");
     </script></body>
</html>
```