



PROGRAMAÇÃO PARA INTERNET RICA

DHTML E O DOCUMENT OBJECT MODEL

Prof. Dr. Daniel Caetano

2011 - 2

Visão Geral

1

- DHTML

2

- Document Object Model

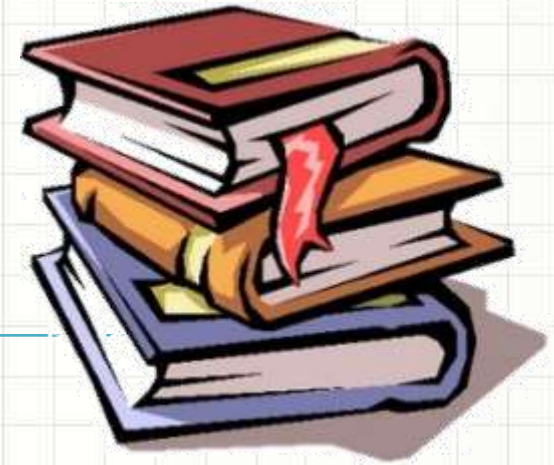
3

- Manipulando o DOM

4

- Manipulando o CSS pelo DOM

Material de Estudo



Material

Acesso ao Material

Notas de Aula

<http://www.caetano.eng.br/aulas/pir/>
(Aula 8)

Apresentação

<http://www.caetano.eng.br/aulas/pir/>
(Aula 8)

Material Didático

Aprenda a Criar Páginas Web c/ HTML, páginas 609 a 648

Google

+“DOM” +tutorial

Web Sites

<http://www.w3.org/>

Objetivos

- Apresentar os conceitos do DHTML
 - Conceituar a estruturação dos objetos do DOM
 - Compreender as diferentes formas de acessar os elementos do DOM
-
- **TRABALHO 2a (turmas de 4ª)!**





DHTML

Introdução

- Já vimos intuitivamente
- DHTML = HTML Dinâmico
- Uso de JavaScript para...
 - Manipular aparência
 - Manipular conteúdo

- Aplicações Web
 - Manipulação precisa e inteligente do HTML/CSS

Introdução

- Manipulação precisa...
- Exige
 - Conhecimento da estruturação dos elementos
 - Conhecimento das formas de acessá-los
 - Conhecimento das formas de modificá-los
- Estruturação dos elementos: DOM
 - **D**ocument
 - **O**bject
 - **M**odel



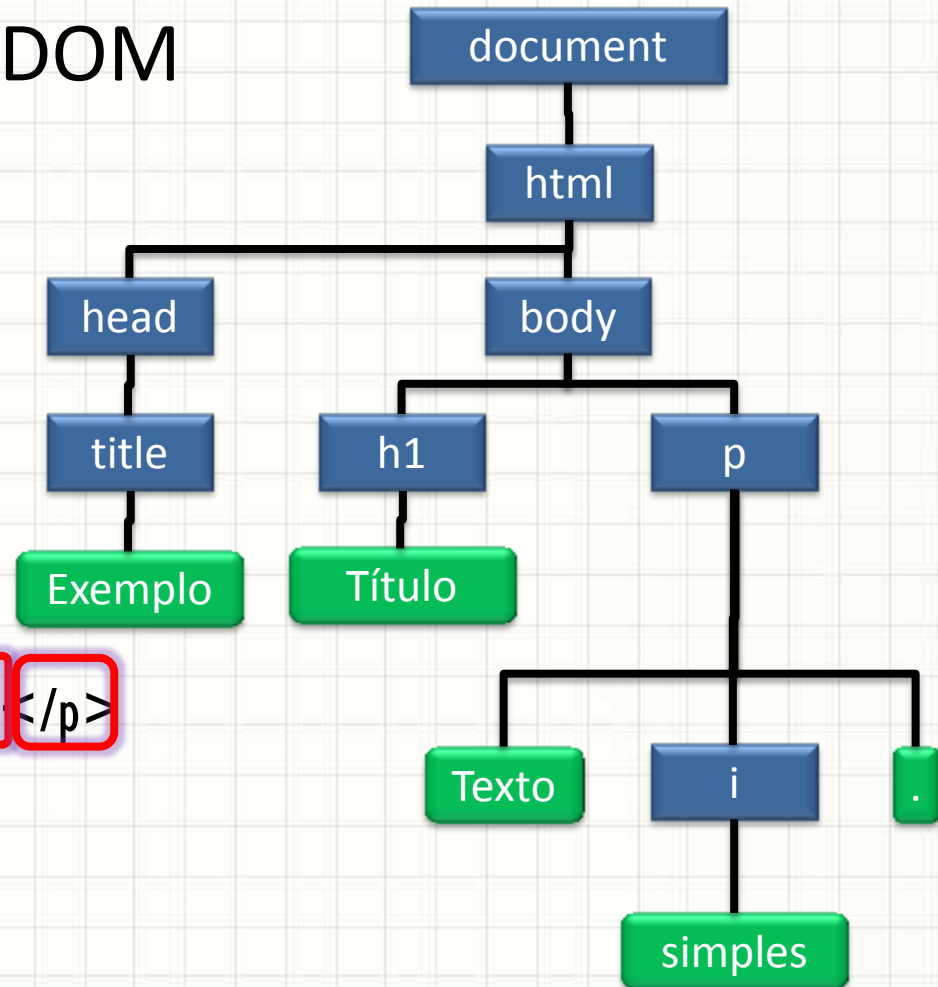
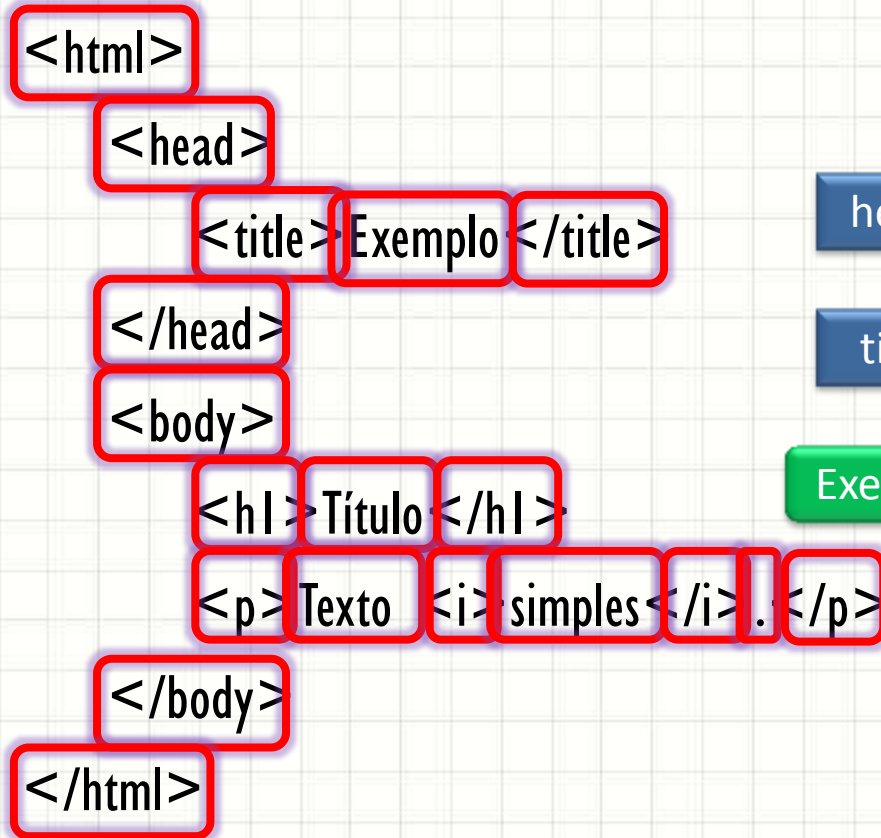
O DOM: MODELO DE OBJETO DE DOCUMENTO

Modelo de Objeto de Documento

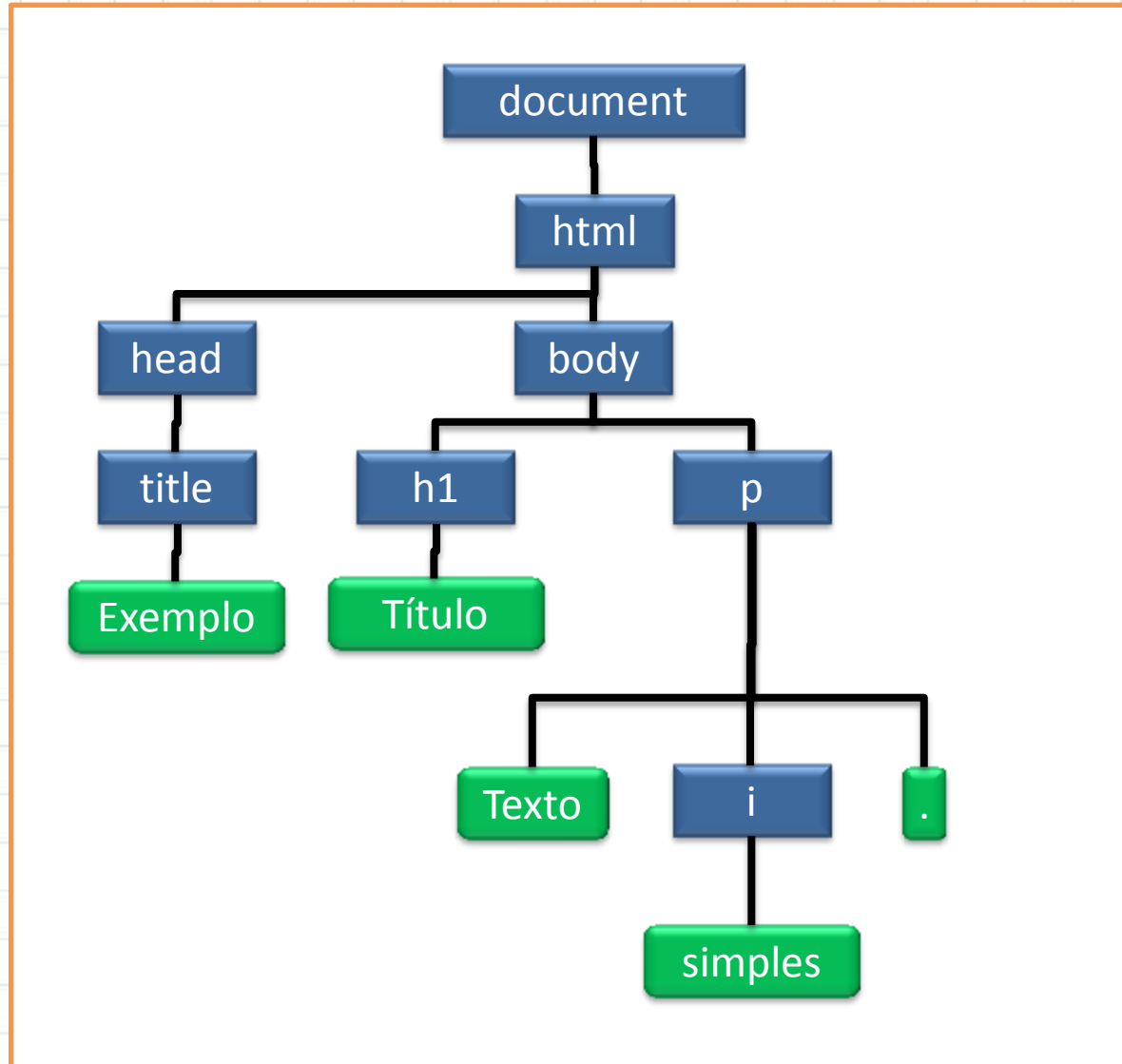
- Maneira de estrutura um documento
 - XML em geral... serve para HTML e XHTML
- Árvore de nós
 - Cada nó é um elemento
 - Cada nó pode conter outros nós
- Nó raiz
 - **document**

Modelo de Objeto de Documento

- Exemplo de Árvore DOM



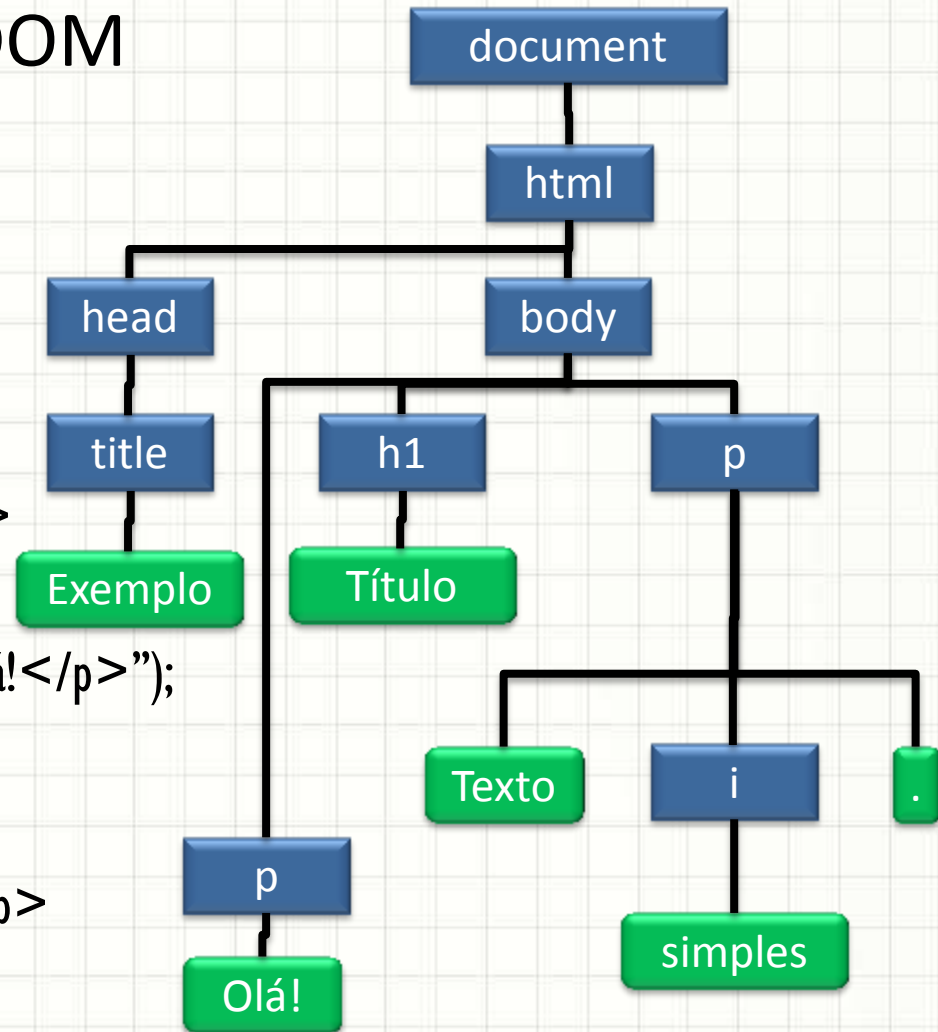
Modelo de Objeto de Documento



Modelo de Objeto de Documento

- Exemplo de Árvore DOM

```
<html>
  <head>
    <title>Exemplo</title>
  </head>
  <body>
    <script type="text/javascript">
      <!--
        document.write("<p>Olá!</p>");
      --> </script>
    <h1>Título</h1>
    <p>Texto <i>simples</i>.</p>
  </body>
</html>
```





MANIPULANDO O DOM

Manipulando o DOM

- As principais funções de manipulação são:
- `getElementById("id")`
 - Retorna um único elemento
 - Existe apenas no objeto **document**
- `getElementsByTagName("nome")`
 - Retorna um vetor de elementos
 - Existe apenas no objeto **document**
- `getElementsByTagName("tag")`
 - Retorna um vetor de elementos
 - Existe em **todos** os elementos do DOM

Manipulando o DOM

- getElementById - Exemplos

```
var tmp = document.getElementById("artigo");  
tmp.style.backgroundColor = "red";
```

```
var tmp2 = document.getElementById("imagem");  
tmp2.src = "nova_imagem.jpg";
```

Manipulando o DOM

- `getElementsByName` - Exemplo

```
var tmp = document.getElementsByName("cpf");  
if (tmp.length > 0) tmp[0].style.backgroundColor = "red";
```

- Cuidado!
- No XHTML... `name` só em formulários!

Manipulando o DOM

- `getElementsByTagName` - Exemplos

```
var tmp = document.getElementsByTagName("p");  
if (tmp.length > 0) tmp[0].style.backgroundColor = "red";
```

- OU

```
var artigo = document.getElementById("artigo");  
var tmp = artigo.getElementsByTagName("p");  
if (tmp.length > 0) tmp[0].style.backgroundColor = "red";
```

Manipulando o DOM

- Todo elemento DOM tem propriedades
- parentNode
 - Retorna o nó “pai” do nó atual
- childNodes
 - Uma matriz de todos os “filhos” do nó atual
- firstChild
 - Primeiro “filho” do nó atual
- lastChild
 - Último “filho” do nó atual
- nodeValue
 - O valor do elemento atual

Manipulando o DOM

- Todo elemento DOM tem propriedades
- `nextSibling` / `previousSibling`
 - Elemento posterior/anterior ao atual
 - Cuidado!

```
<h1 id="a">Teste</h1>[espaço]
```

```
<p>Texto</p>
```

```
var tmp = document.getElementById("a");
```

```
var tmp2 = tmp.nextSibling;
```

```
tmp2.style.backgroundColor = "red";
```

Manipulando o DOM

- Todo elemento do DOM tem propriedades
- nextSibling

ISSO DÁ ERRO!
Texto não tem
propriedade **style!**

```
<p>||  
var tmp = document.getElementsByTagName("a");  
var tmp2 = tmp.nextSibling;  
tmp2.style.backgroundColor = "red";
```

Manipulando o DOM

- Todo elemento DOM tem propriedades
- innerHTML
 - Permite acrescentar um novo ramo ao elemento
 - Pode-se escrever HTML normal, o navegador constrói tudo

Manipulando o DOM

- Todo elemento DOM tem propriedades
- innerHTML

```
<h1 id="a">Teste</h1>
```

```
var tmp = document.getElementById("a");  
tmp.innerHTML = "<em>Novo</em> teste";
```

- Tem o efeito de...

```
<h1 id="a"><em>Novo</em> teste</h1>
```

Manipulando o DOM

- E para apenas **acrescentar** algo?
- `createElement("tag")`
 - Cria um elemento
 - Função do objeto **document**
- `createTextNode("texto")`
 - Cria um elemento de texto
 - Função do objeto **document**
- `appendChild(filhote)`
 - Acrescenta um filhote ao elemento atual
 - Função de **todos** os elementos do DOM

Manipulando o DOM

- Exemplo

```
var par = document.createElement("p");  
var tex = document.createTextNode("Oi!");  
par.appendChild(tex);  
var div = document.getElementById("artigo");  
div.appendChild(par);
```

- No caso, o parágrafo será acrescentado como o último filhote do div "artigo"

Manipulando o DOM

- Mais algumas funções
- `removeChild(elemento)`
 - Remove um filho do elemento atual
 - Função de todos os elementos do DOM
- `replaceChild(original, novo)`
 - Substitui um filho existente (original) por outro (novo)
 - Função de todos os elementos do DOM



MANIPULANDO O CSS PELO DOM

Manipulando o CSS pelo DOM

- Há duas formas padronizadas de mudar CSS
- Modificando o atributo **style** dos elementos
 - Cria/remove estilos **inline** no XHTML
 - Basta alterar o atributo como temos visto

```
document.body.style.color = "blue";
```

```
var tmp = document.getElementById("local");
```

```
tmp.style.backgroundColor = "yellow";
```

Manipulando o CSS pelo DOM

- Há duas formas padronizadas de mudar CSS
- Mudando a classe de um elemento
 - Modifica o atributo “class” do elemento

```
var tmp = document.getElementById("p1");  
p1.className = "outra";
```

Manipulando o CSS pelo DOM

- Para **adicionar** uma classe:

```
var tmp = document.getElementById("p1");  
p1.className += "_outra";
```



COMO LER O CSS?

Lendo o CSS com JavaScript

- Você já deve ter observado que...

```
var cor = document.body.style.color;  
window.alert(cor);
```

- Só imprime a cor se ela estiver **inline**
- Como fazer para atributos do arquivo CSS?

Lendo o CSS com JavaScript

- Alguns atributos podem ser lidos diretamente
- `offsetHeight`
 - Altura em pixels do elemento
- `offsetWidth`
 - Largura em pixels do elemento
- `offsetLeft`
 - Distância em pixels a partir do elemento da esquerda
- `offsetTop`
 - Distância em pixels a partir do elemento de cima

Lendo o CSS com JavaScript

- Mas e se eu quiser ler outros valores do CSS?
- Não tem um método padrão, então...
 - Tem um código que testa os dois (IE x Outros)

Lendo o CSS com JavaScript

```
function getStyle( elem, estilo ) {  
    var valor = "";  
    if (document.defaultView &&  
        document.defaultView.getComputedStyle) {  
        valor = document.defaultView.getComputedStyle(elem, "");  
        valor = valor.getPropertyValue(estilo);  
    }  
    else if (elem.currentStyle) {  
        estilo = estilo.replace(/-(\w)/g,  
            function(match,p1) { return p1.toUpperCase();});  
        valor = elem.currentStyle[estilo];  
    }  
    return valor;  
}
```

Lendo o CSS com JavaScript

- Como usar isso?
- É mais fácil do que parece:

```
var el = document.getElementById("artigo");  
var cor = getStyle(el,"background-color");
```



CONCLUSÕES

Resumo

- O DOM permite uma forma de acessar diretamente todos os elementos do XHTML
- É possível adicionar, editar e remover elementos do XHTML pelo JavaScript, usando a estrutura do DOM
- O acesso direto aos valores do CSS não é padronizado, mas é possível usar um método que funciona em todos os navegadores
- **TAREFA**
 - Trabalho 2a (para as turmas de 4ª)!

Próxima Aula



- Usando tudo isso para fazer algo útil
 - Construir a Interface com o Usuário
 - Aplicação Web: Jogo da Velha



PERGUNTAS?



**BOM DESCANSO
A TODOS!**