# Programação para Internet Rica

## DHTML e o Document Object Model

Prof. Dr. Daniel Caetano

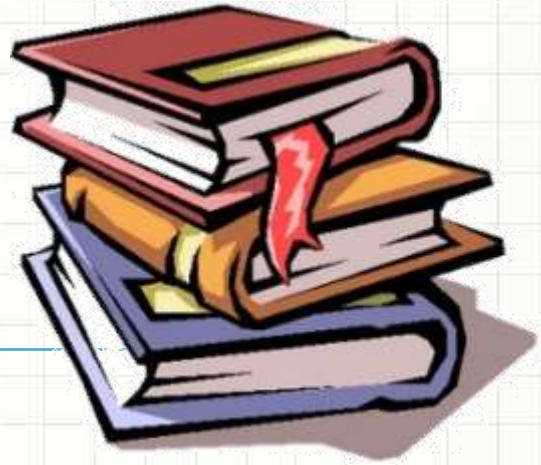
2012 - 2

# Objetivos

- Apresentar os conceitos do DHTML
- Conceituar a estruturação dos objetos do DOM
- Compreender as diferentes formas de acessar os elementos do DOM

- **AV1!**

# Material de Estudo



| Material | Acesso ao Material |
|---|---|
| Notas de Aula | http://www.caetano.eng.br/ (Aula 8) |
| Apresentação | http://www.caetano.eng.br/ (Aula 8) |
| Material Didático | Aprenda a Criar Páginas Web c/ HTML, páginas 609 a 648 |
| Google | +“DOM” +tutorial |
| Web Sites | http://www.w3.org/ |

# 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

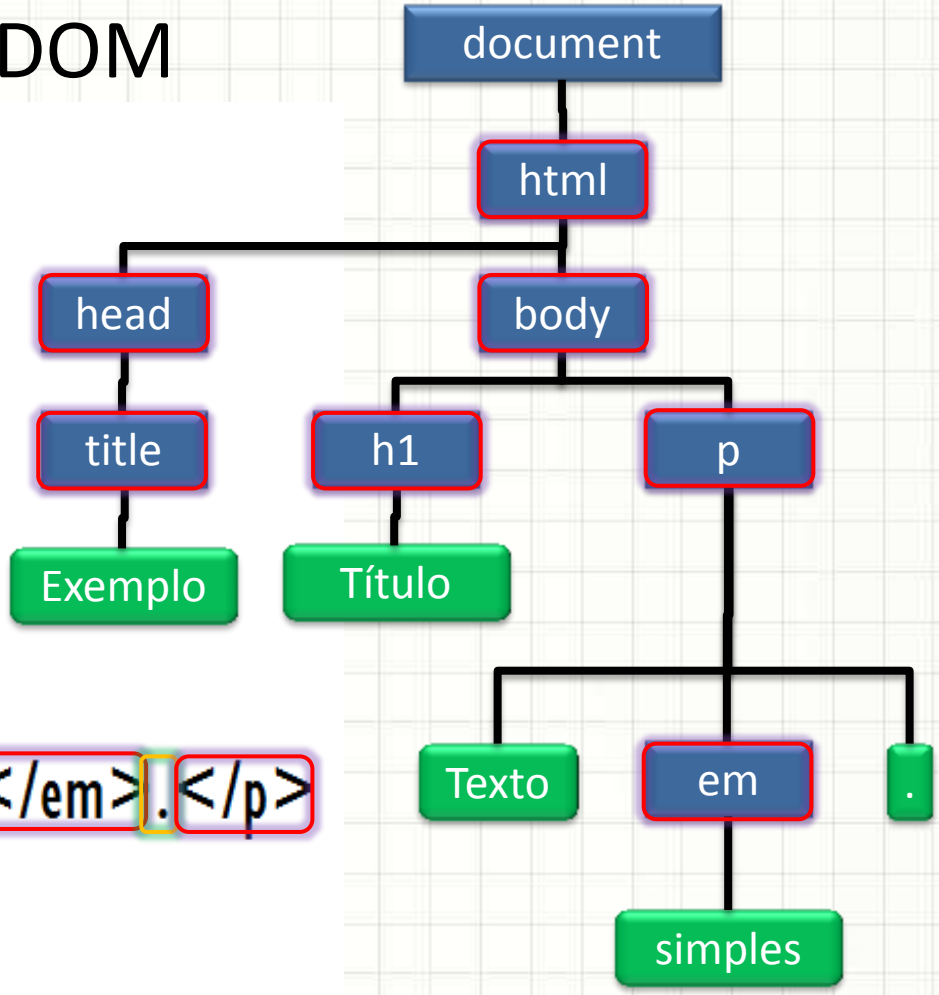
# Modelo de Objeto de Documento

- 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**

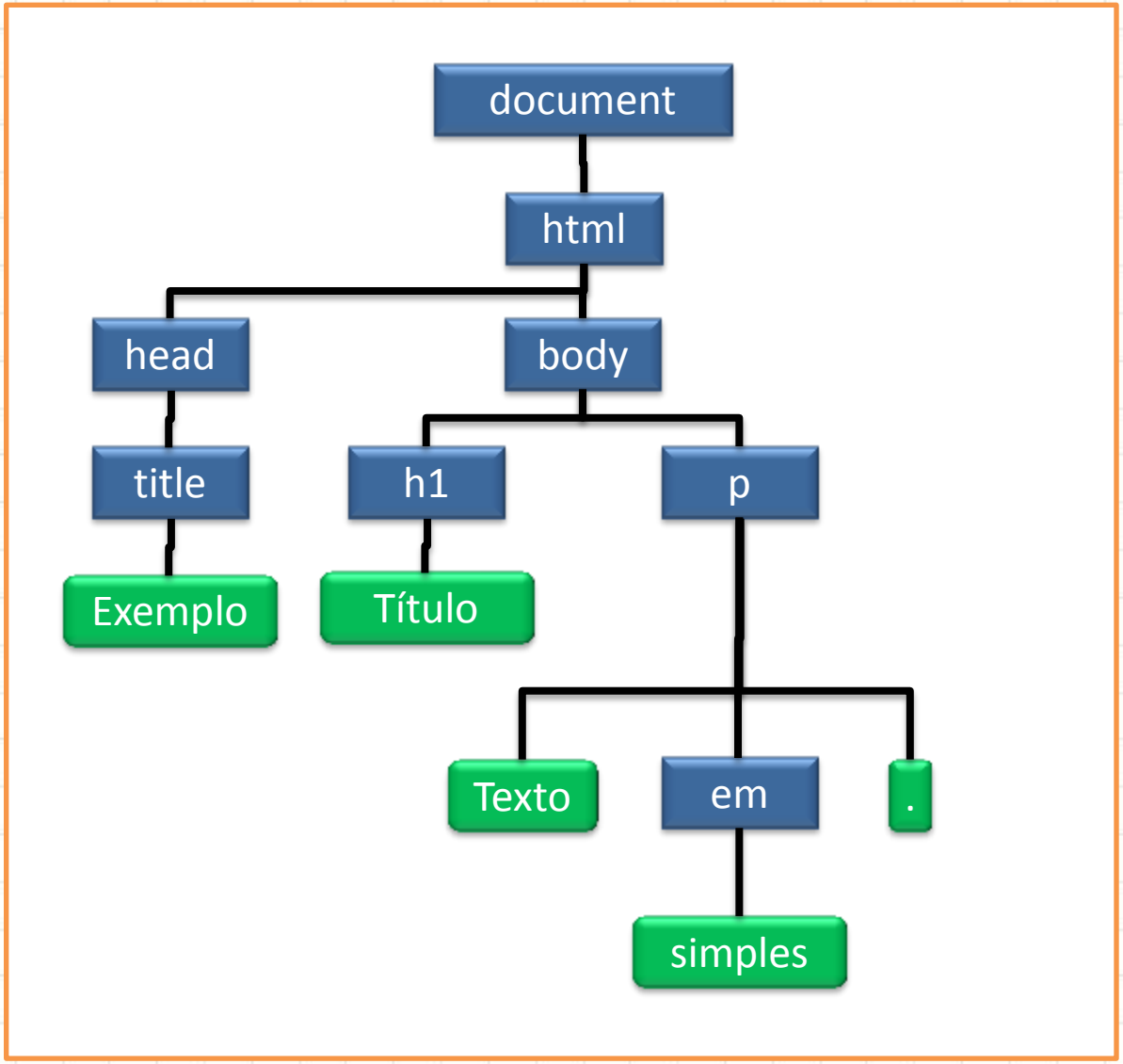
# Modelo de Objeto de Documento

- Exemplo de Árvore DOM

```
<html>
    <head>
        <title>Exemplo</title>
    </head>
    <body>
        <h1>Título</h1>
        <p>Texto <em>simples</em>.</p>
    </body>
</html>
```

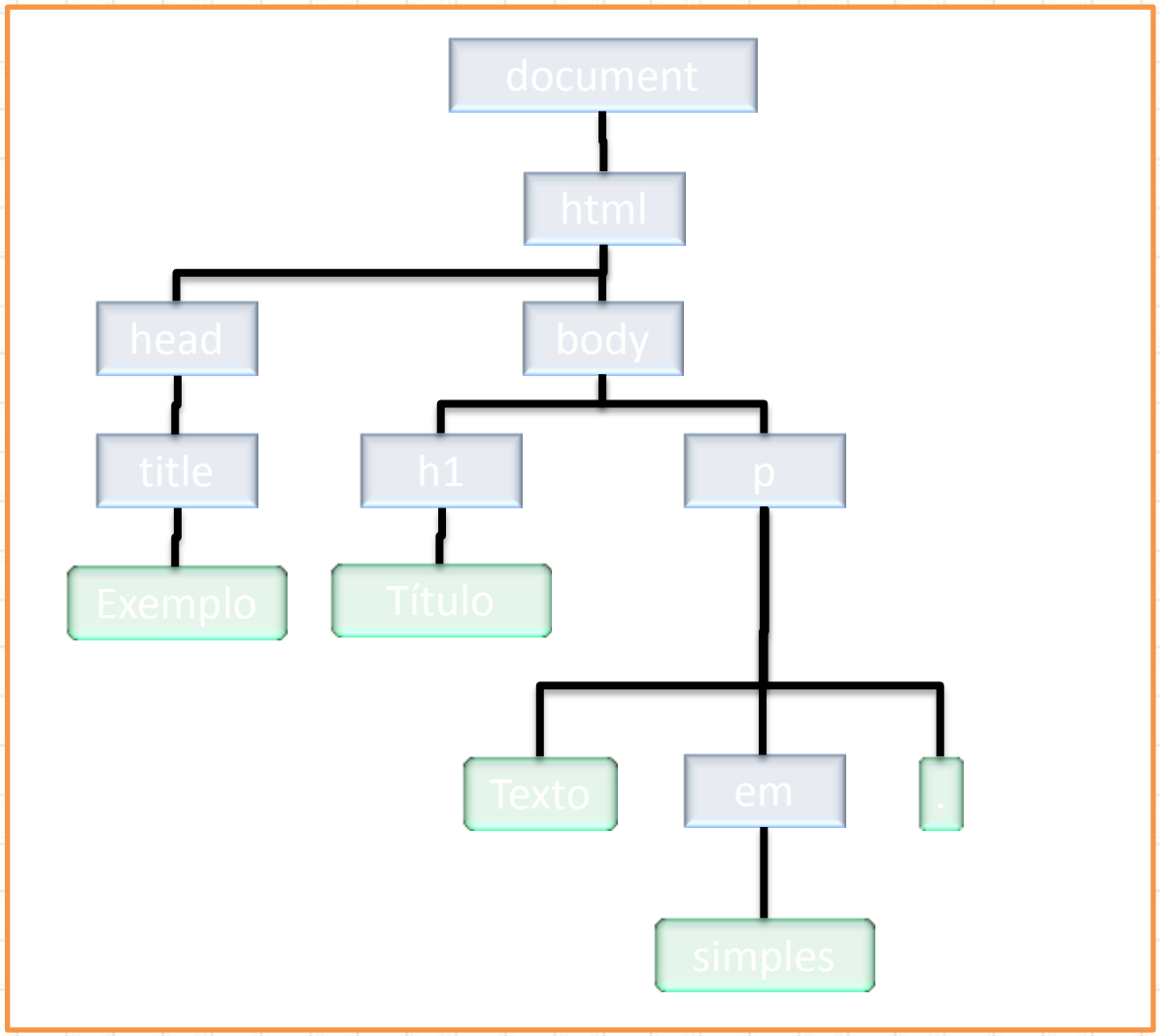


# Modelo de Objeto de Documento



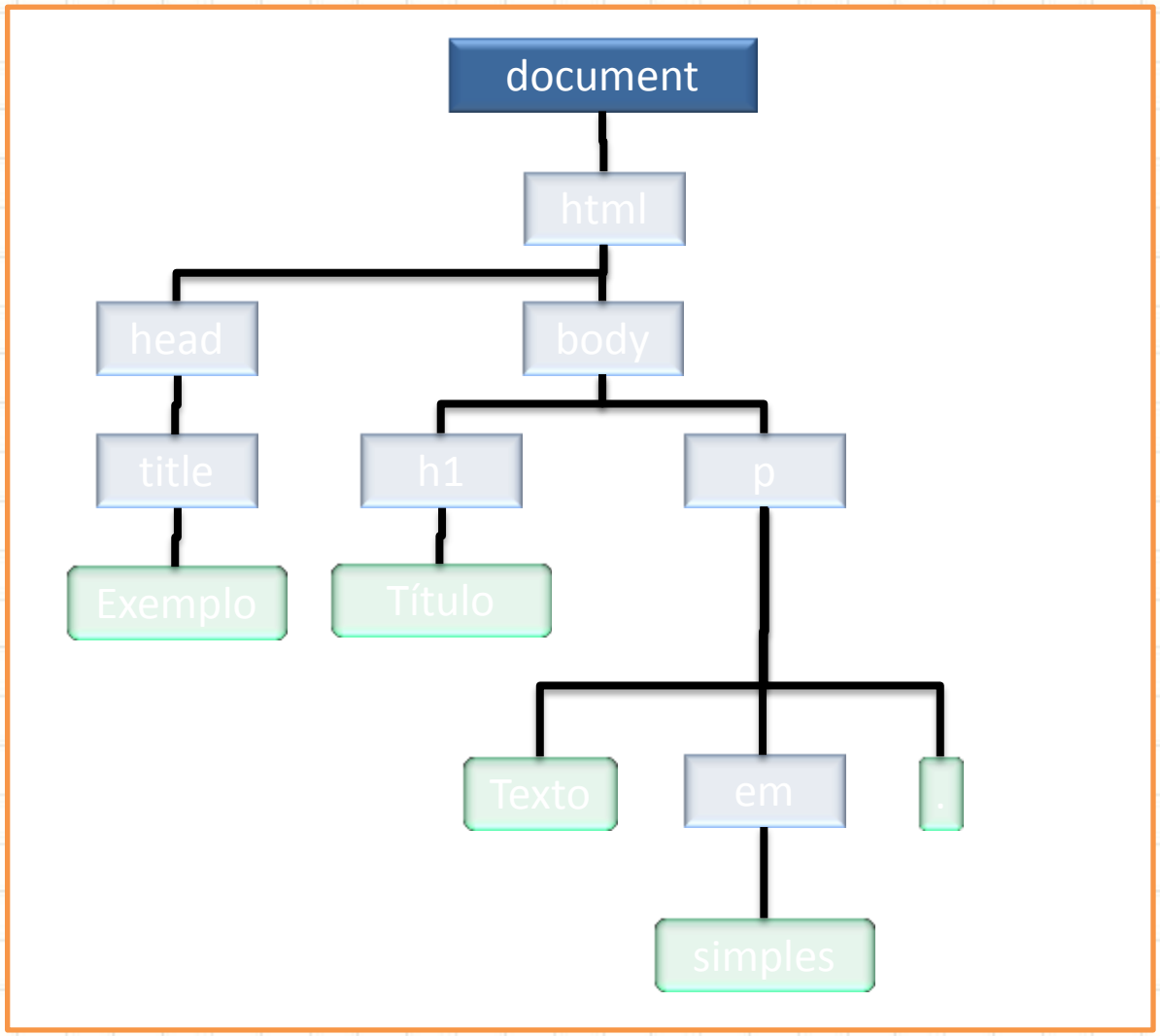
# Modelo de Objeto de Documento

- Busca pelo 



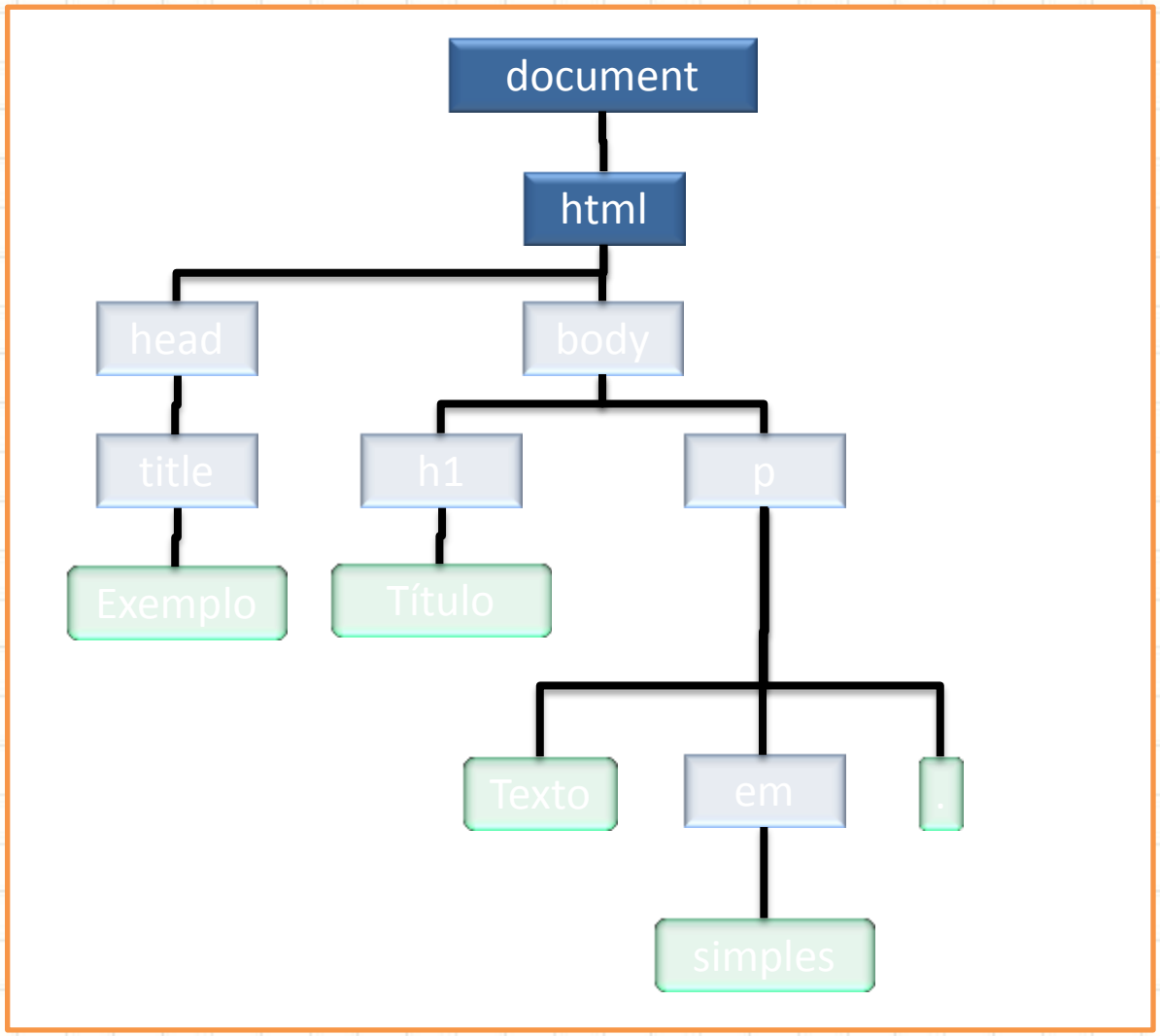
# Modelo de Objeto de Documento

- Busca pelo 



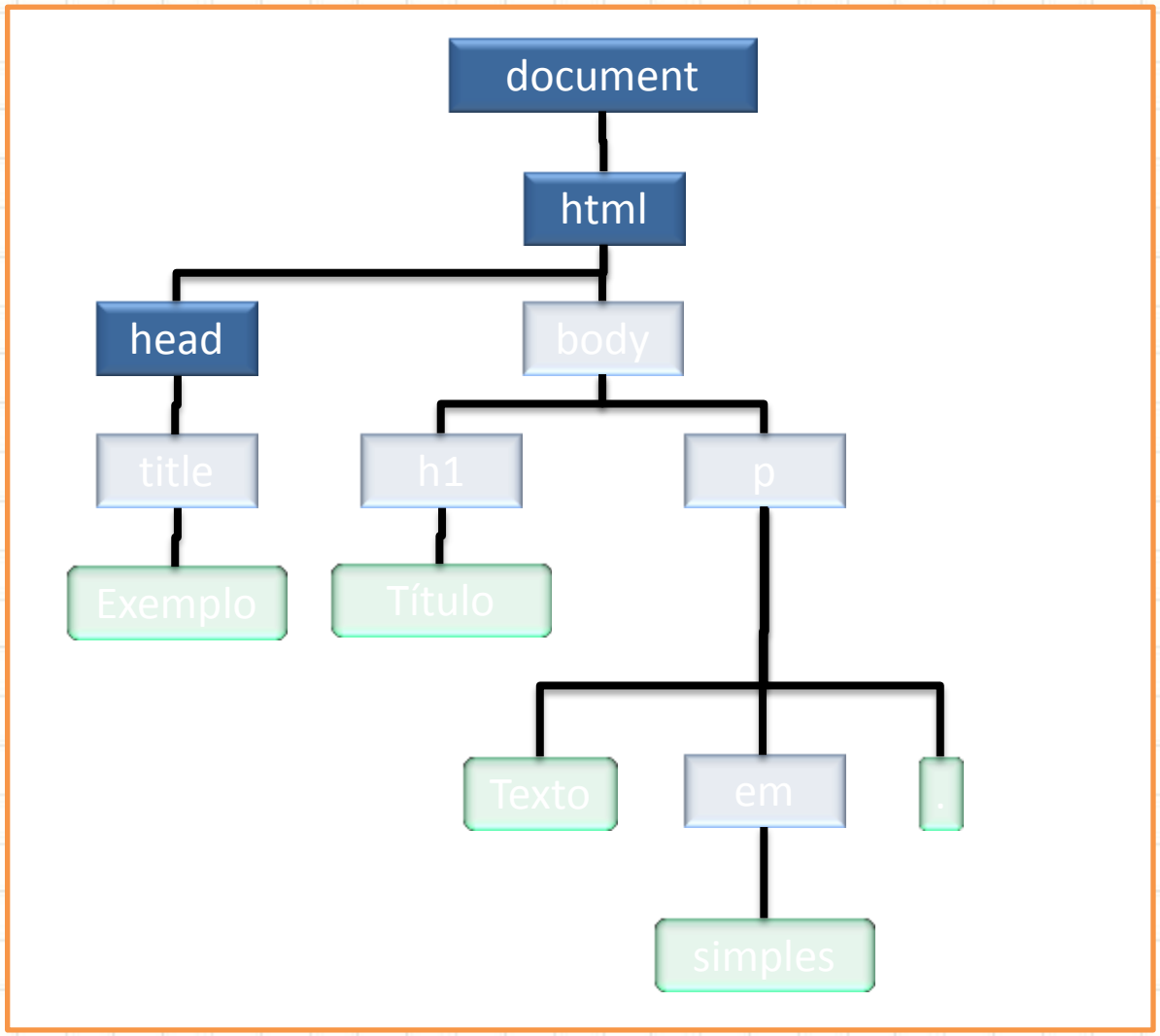
# Modelo de Objeto de Documento

- Busca pelo 



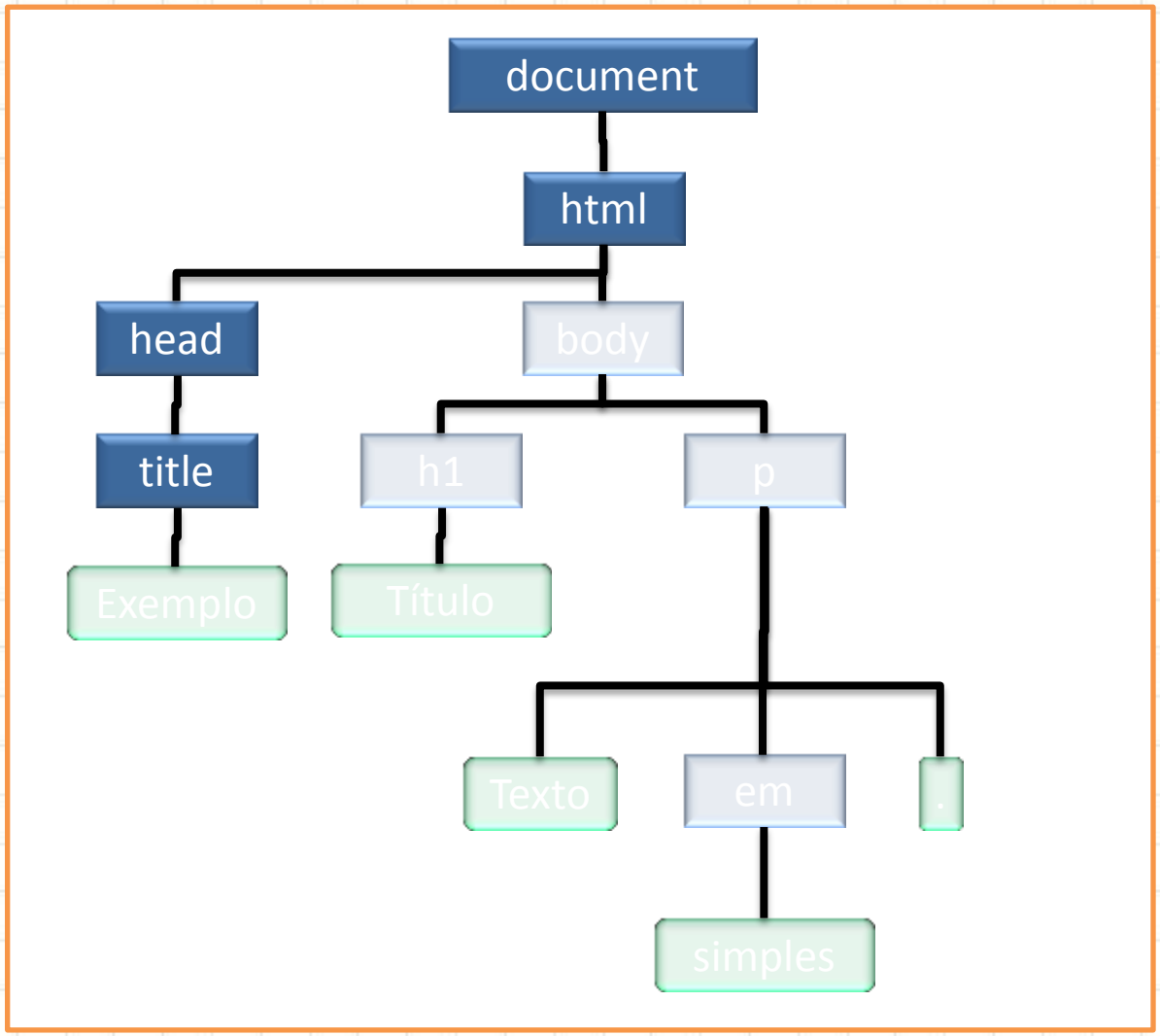
# Modelo de Objeto de Documento

- Busca pelo 



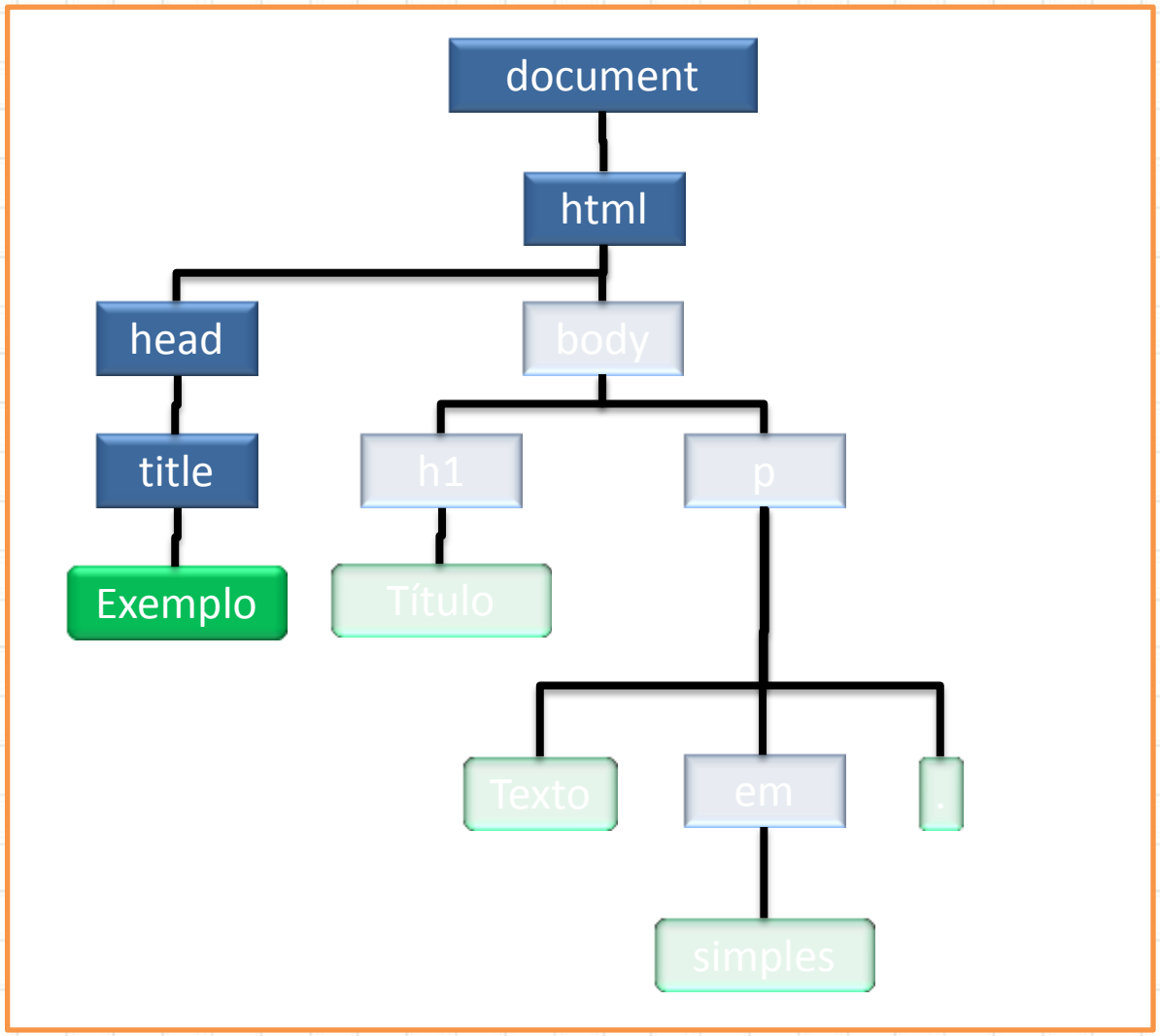
# Modelo de Objeto de Documento

- Busca pelo 



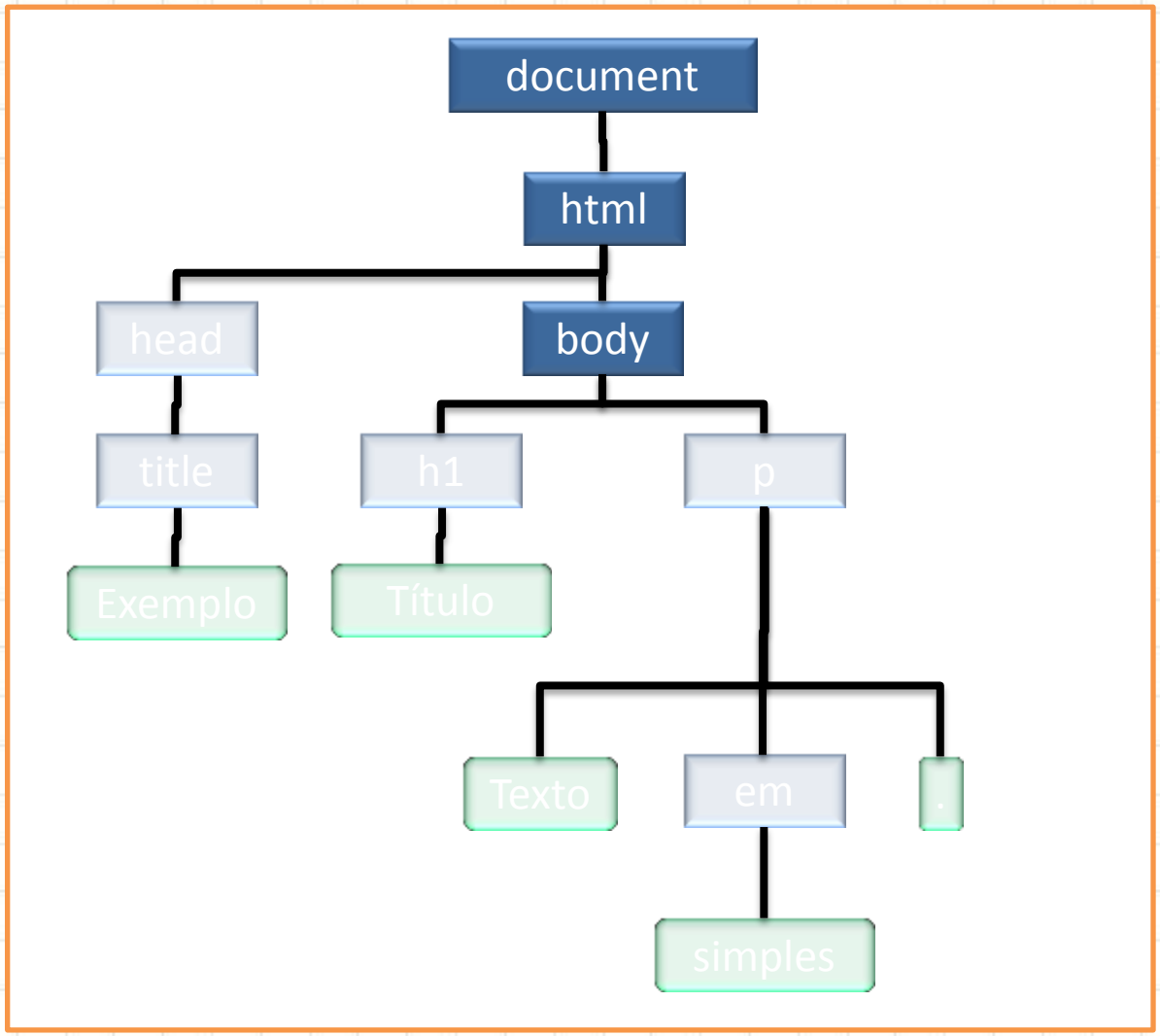
# Modelo de Objeto de Documento

- Busca pelo 



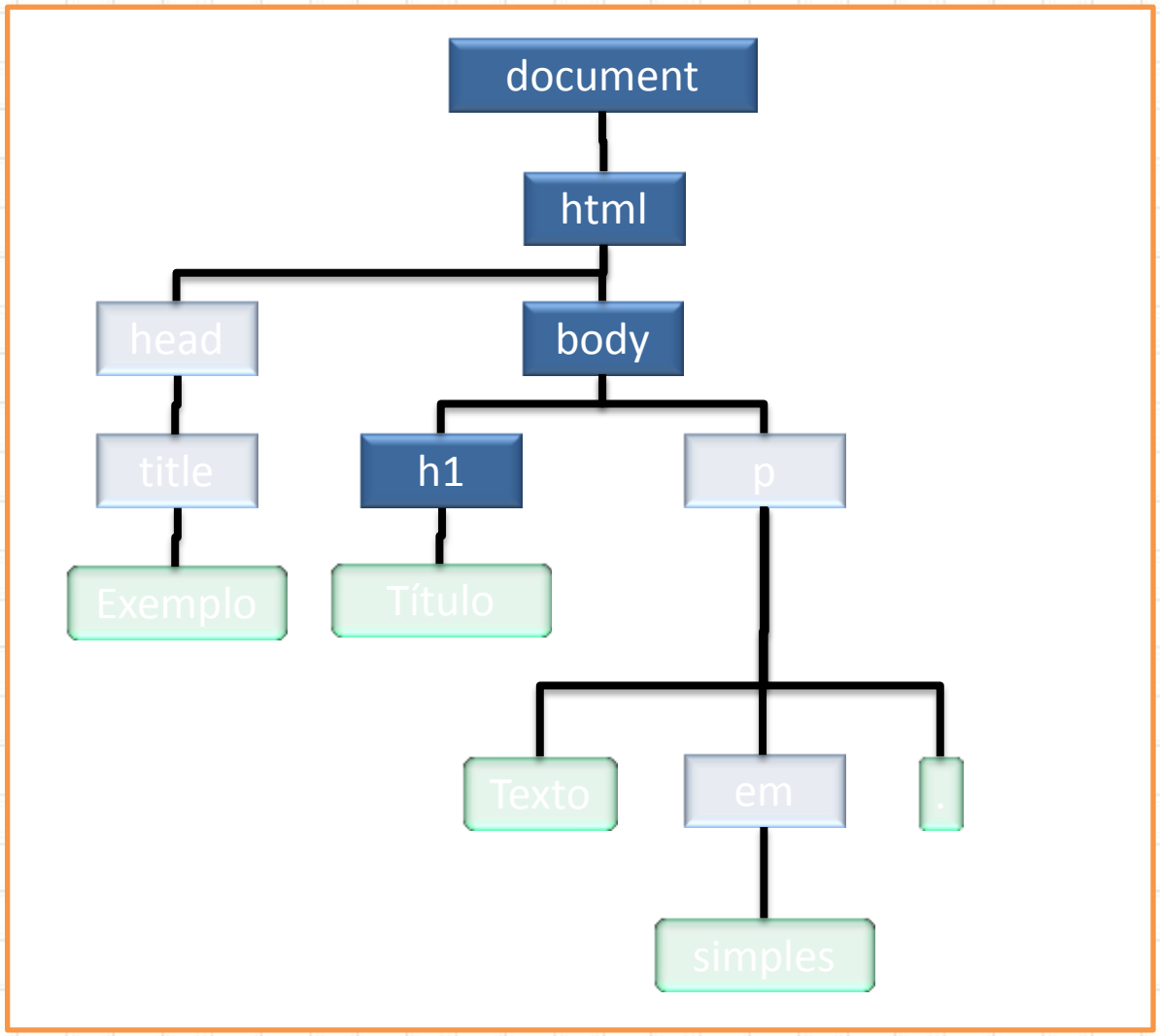
# Modelo de Objeto de Documento

- Busca pelo 



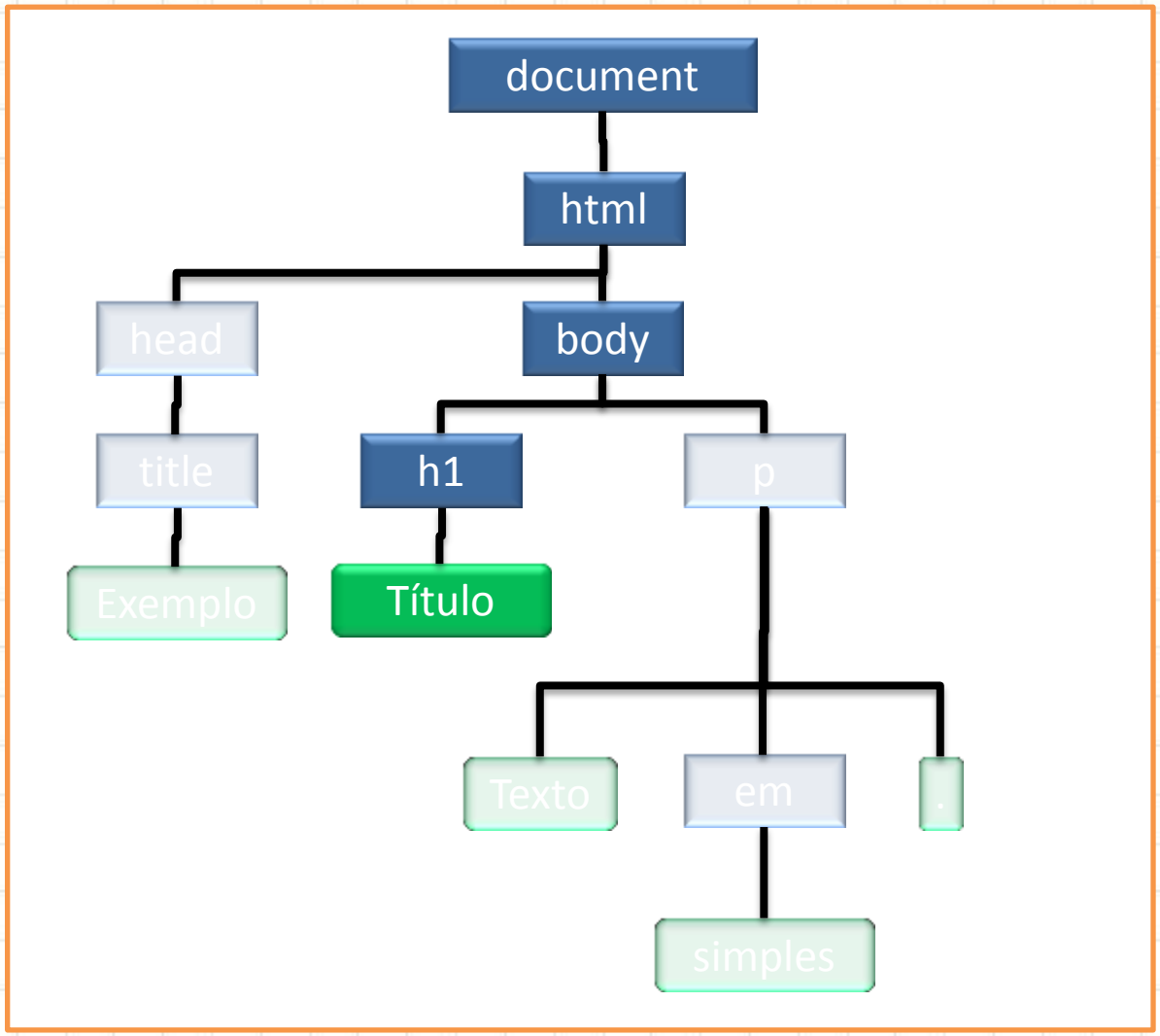
# Modelo de Objeto de Documento

- Busca pelo 



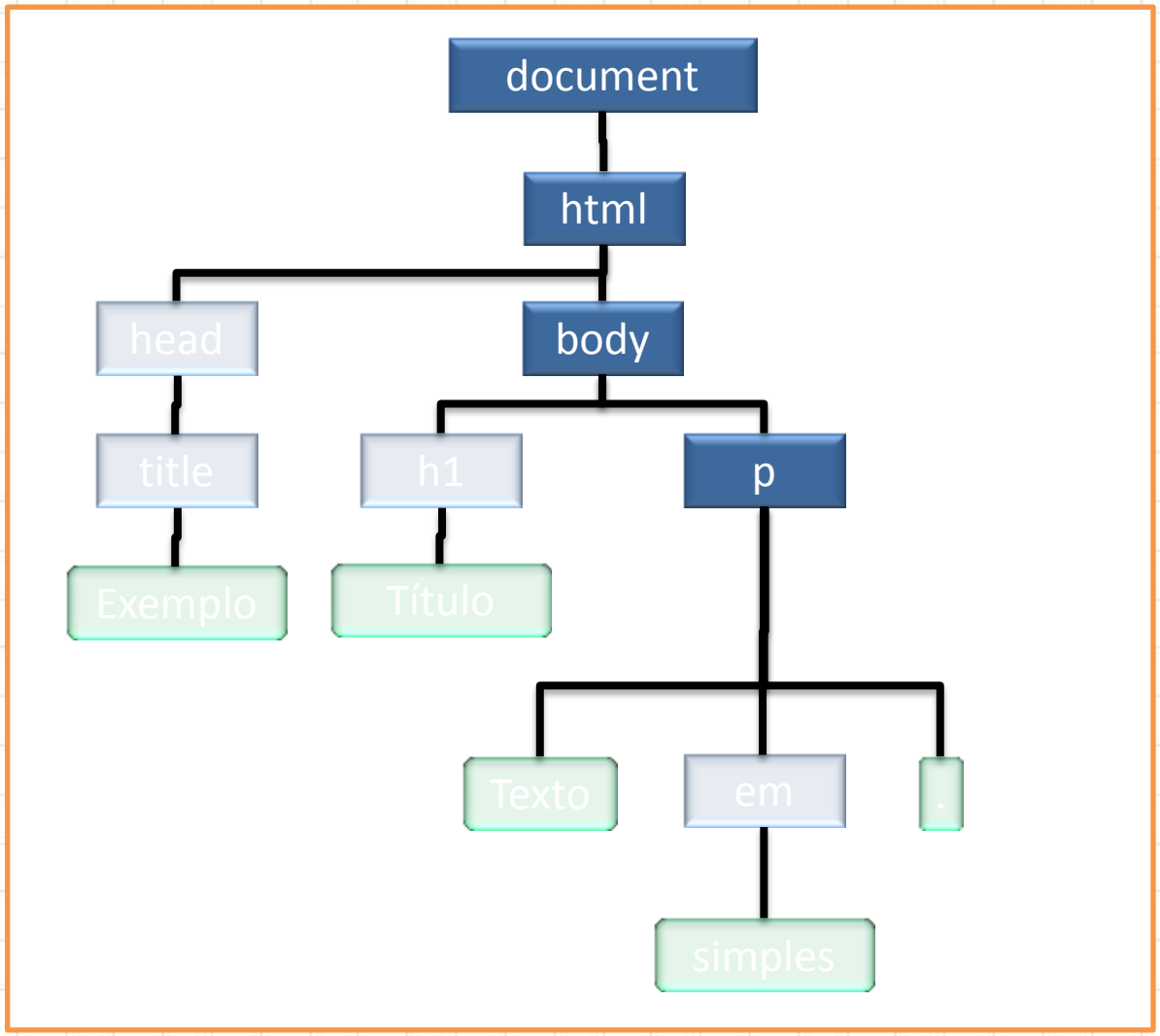
# Modelo de Objeto de Documento

- Busca pelo 



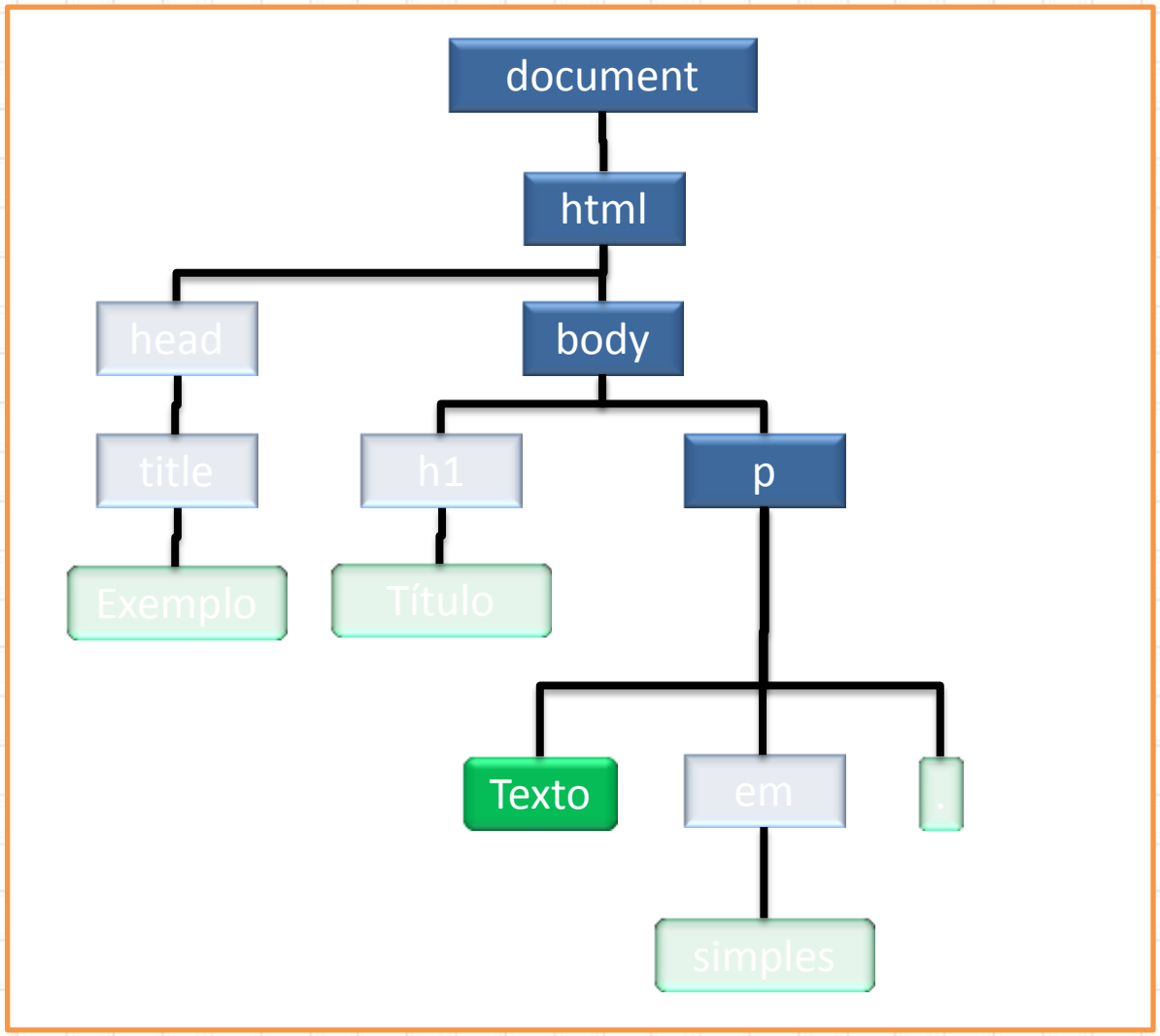
# Modelo de Objeto de Documento

- Busca pelo 



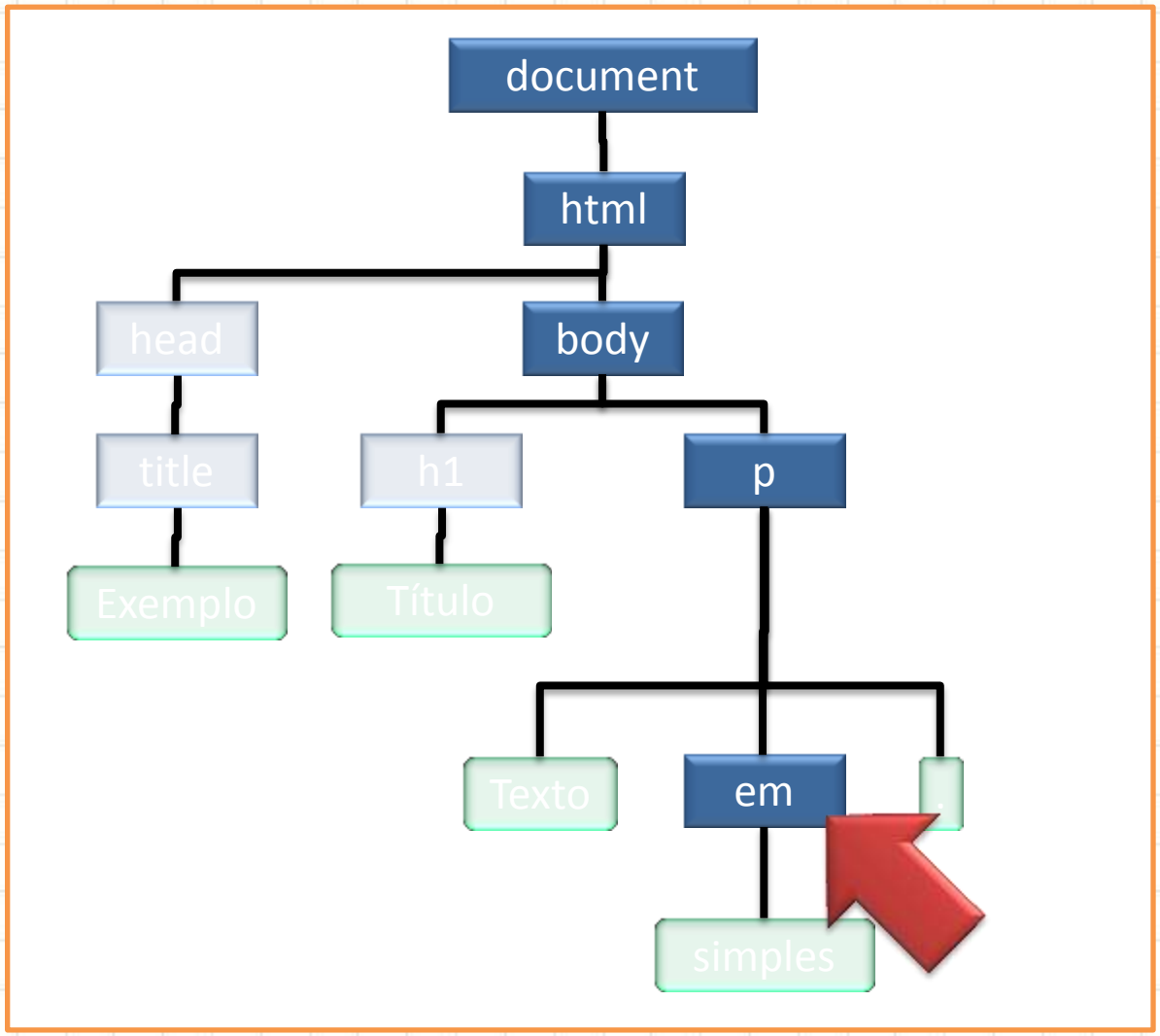
# Modelo de Objeto de Documento

- Busca pelo 



# Modelo de Objeto de Documento

- Busca pelo 



# Manipulando o DOM

# Manipulando o DOM

- As principais funções de manipulação são:

- getElementById(“id”)
  - Retorna um único elemento
  - Existe apenas no objeto **document**
- getElement**s**ByName(“nome”)
  - Retorna um vetor de elementos
  - Existe apenas no objeto **document**
- getElement**s**ByTagName(“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

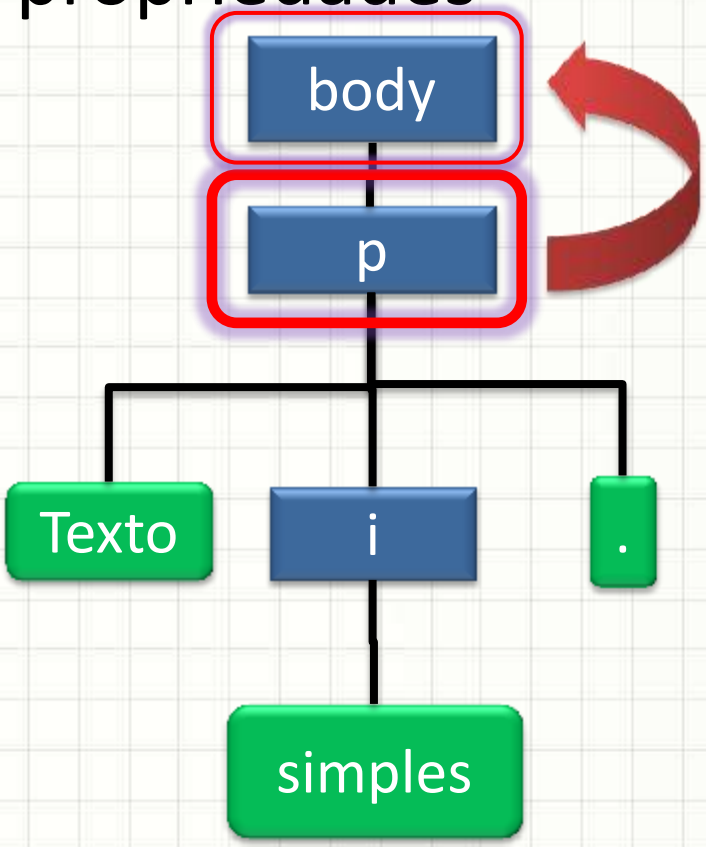
```
var tmp = document.getElementsByTagName(“p”);
if (tmp.length > 0) tmp[0].style.backgroundColor = “red”;
```

- OU

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

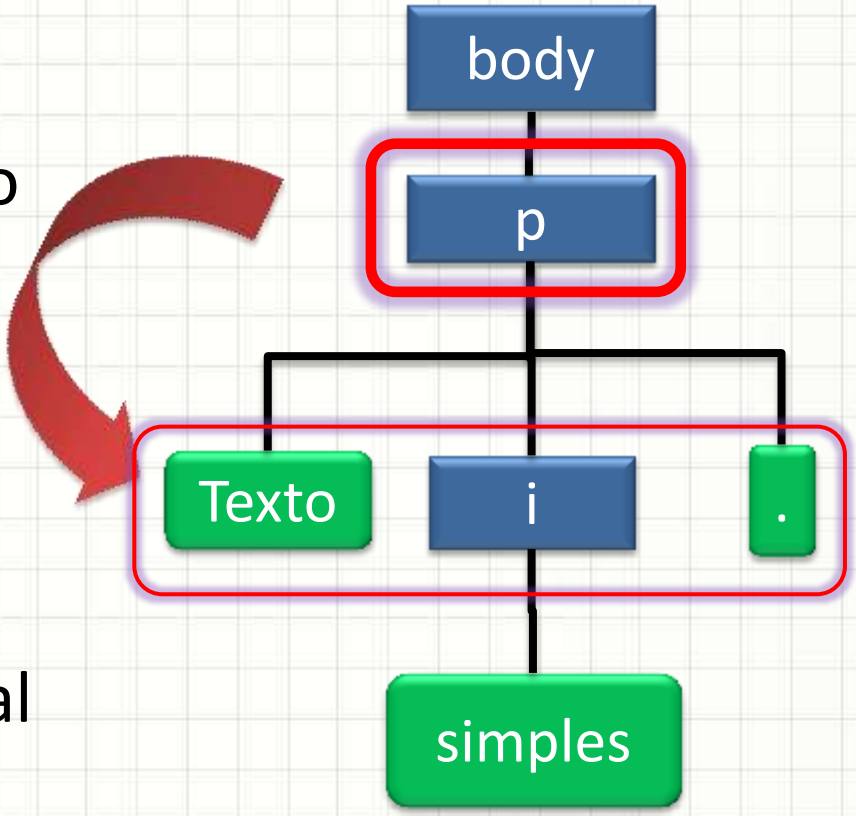
# Manipulando o DOM

- Todo elemento DOM tem propriedades
- parentNode
  - Retorna o nó “pai” do nó atual



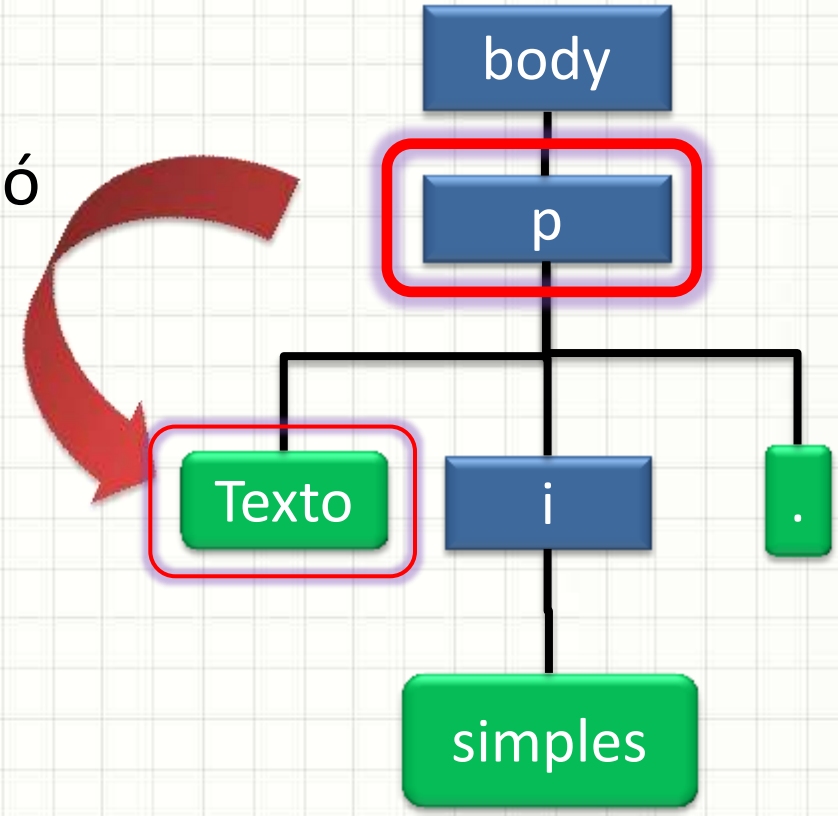
# Manipulando o DOM

- Todo elemento DOM tem propriedades
- parentNode
  - Retorna o nó “pai” do nó atual
- childNodes
  - Uma matriz de todos os “filhos” do nó atual



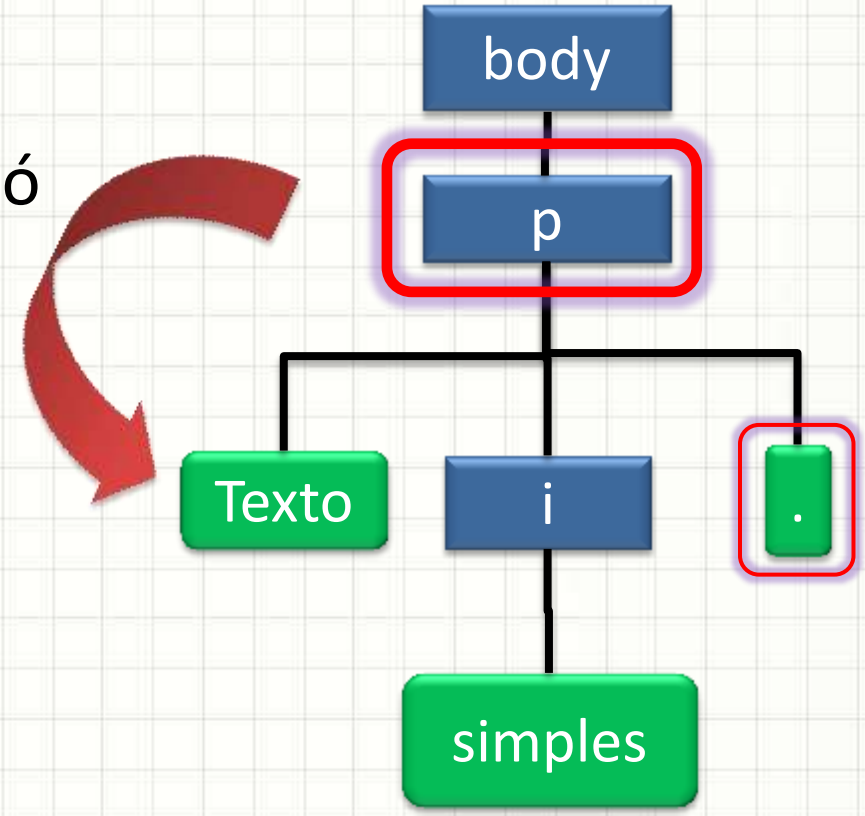
# Manipulando o DOM

- Todo elemento DOM tem propriedades
- firstChild
  - Primeiro “filho” do nó atual



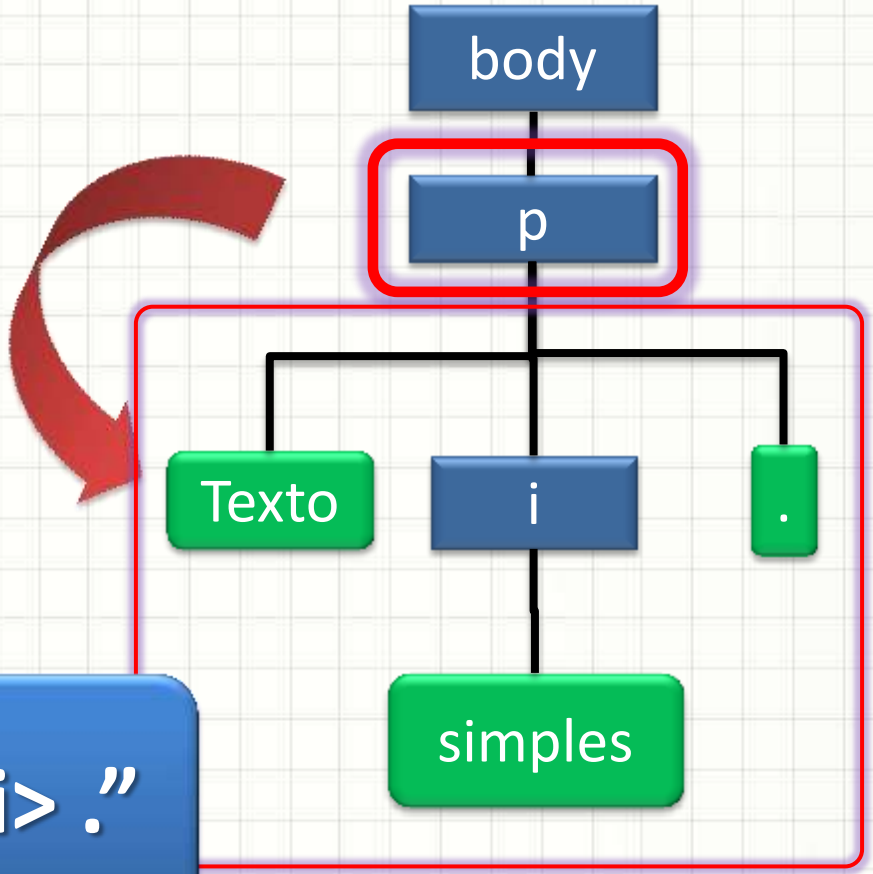
# Manipulando o DOM

- Todo elemento DOM tem propriedades
- firstChild
  - Primeiro “filho” do nó atual
- lastChild
  - Último “filho” do nó atual



# Manipulando o DOM

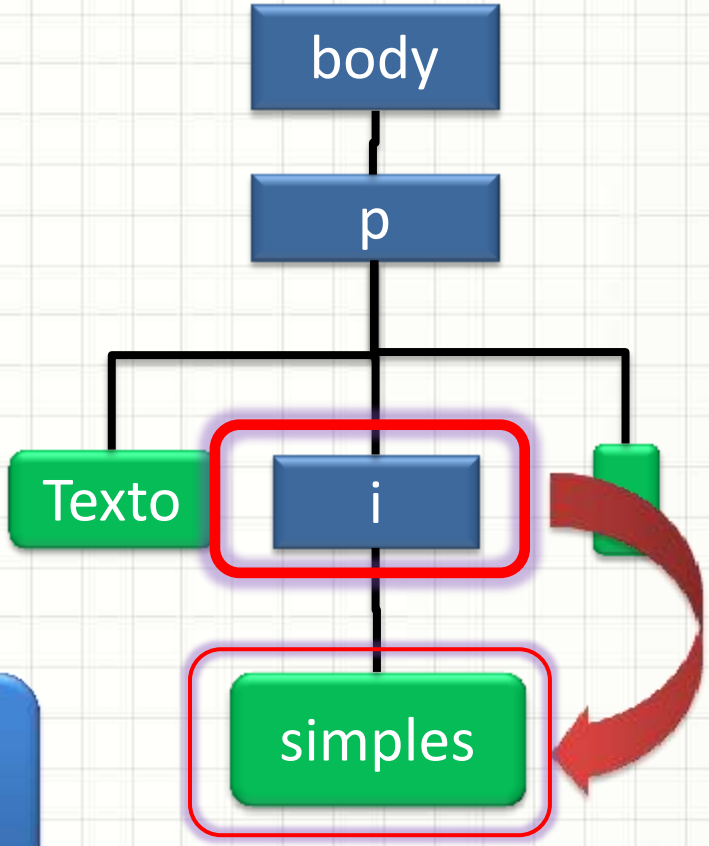
- Todo elemento DOM tem propriedades
- nodeValue
  - O valor do elemento atual




**“Texto simples .”**

# Manipulando o DOM

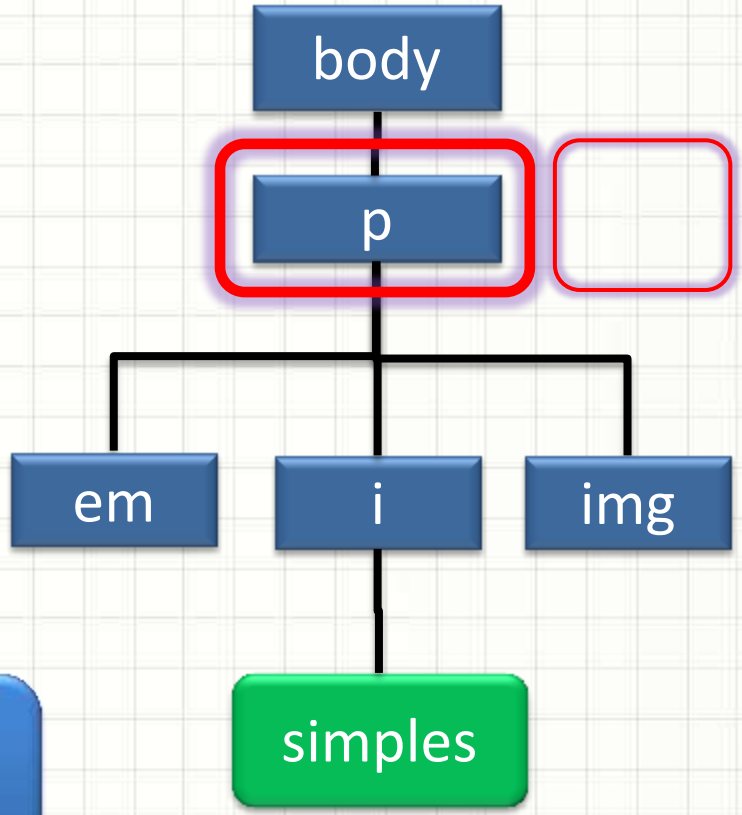
- Todo elemento DOM tem propriedades
- nodeValue
  - O valor do elemento atual




**“simples”**

# Manipulando o DOM

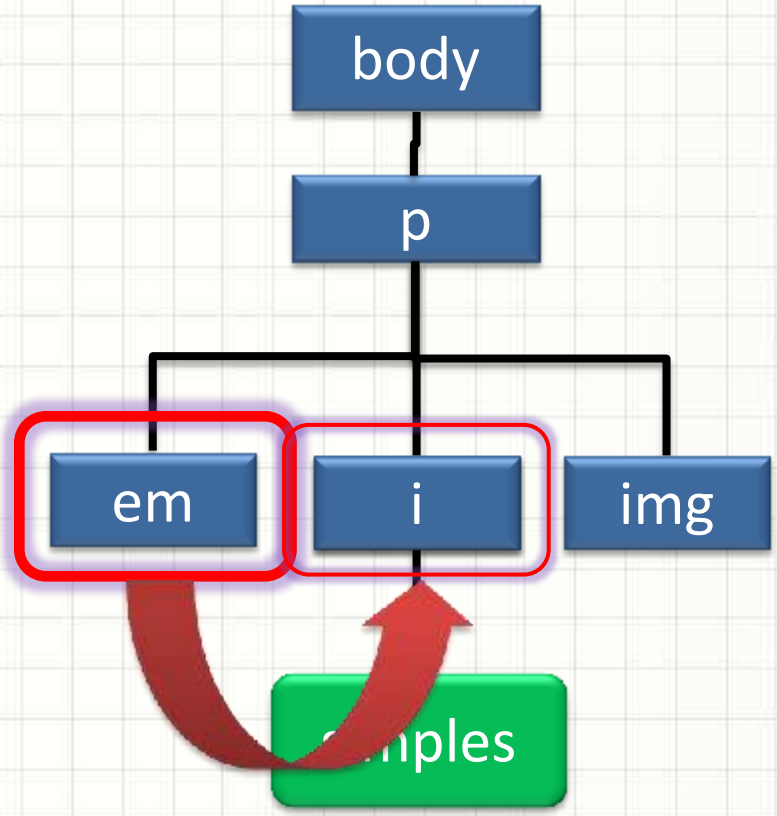
- Todo elemento DOM tem propriedades
- nextSibling
  - Elemento posterior




**null**

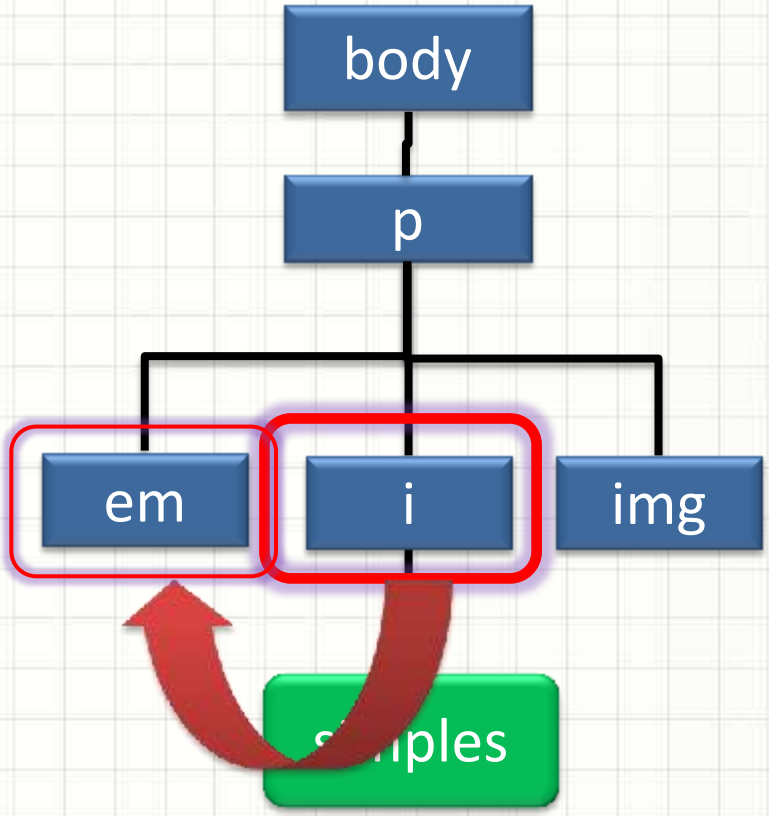
# Manipulando o DOM

- Todo elemento DOM tem propriedades
- nextSibling
  - Elemento posterior



# Manipulando o DOM

- Todo elemento DOM tem propriedades
- nextSibling
  - Elemento posterior
- previousSibling
  - Elemento anterior



# 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";
```

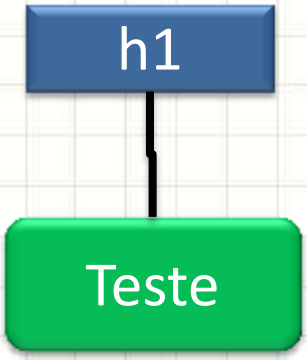
- Tem o efeito de...

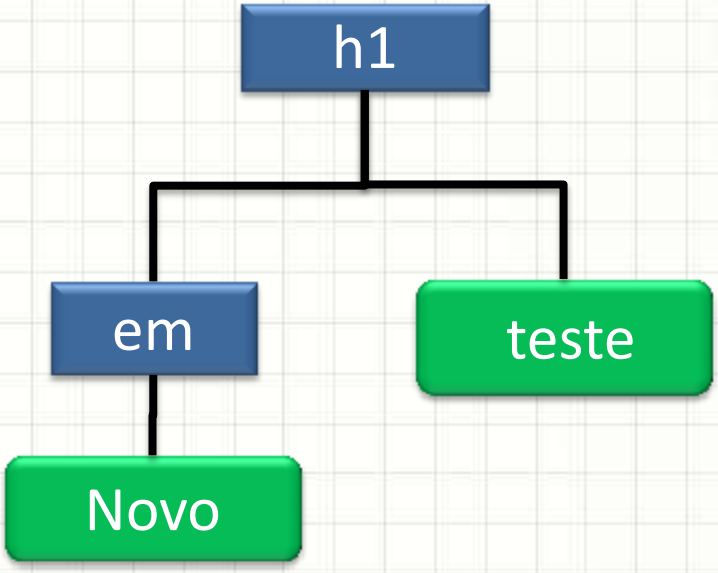
```
<h1 id="a"><em>Novo</em> teste</h1>
```

# Manipulando o DOM

- innerHTML

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

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

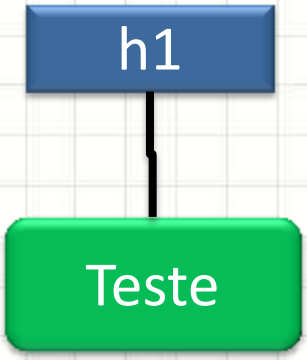


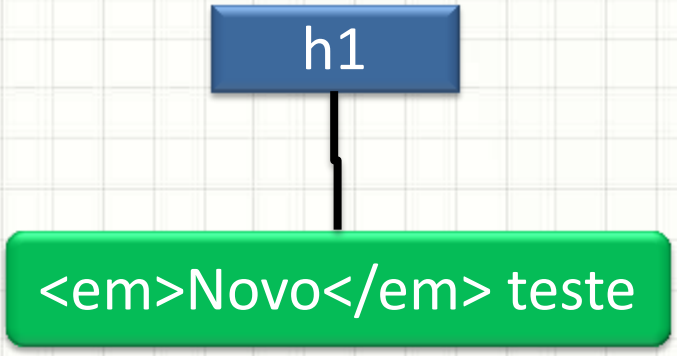



# Manipulando o DOM

- innerHTML **é diferente de nodeValue**!

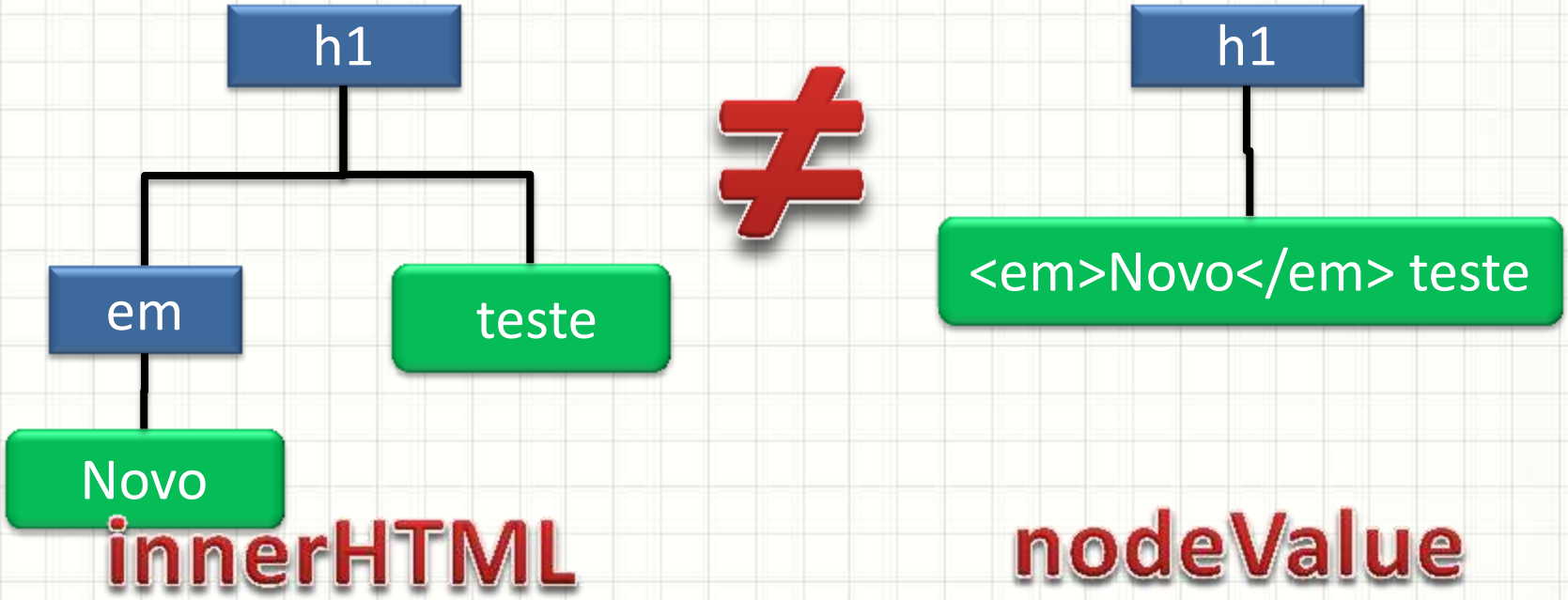
```
<h1 id="a">Teste</h1>

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



# Manipulando o DOM

- innerHTML x nodeValue
  - h1.innerHTML = “Novo teste”;
  - h1.nodeValue = “Novo teste”;



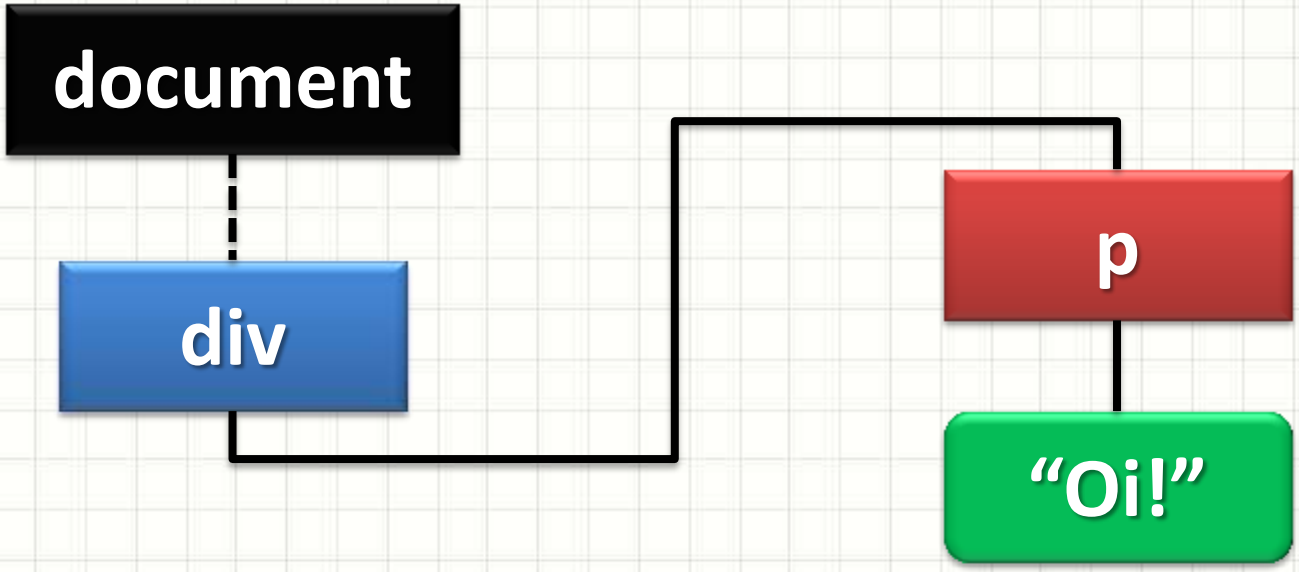
# Criando Elementos DOM

# Manipulando o DOM

- E para apenas **acrescentar** algo?
- document.**createElement**(“tag”)
  - Cria um elemento
  - Função do objeto **document**
- 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

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



# 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 Visual pelo DOM

# Manipulando o Visual pelo DOM

- Duas formas de mudar propriedades visuais
- 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

- Duas formas de mudar propriedades visuais
- Mudando a classe de um elemento
  - Modificar o atributo “class” do elemento

```
var tmp = document.getElementById(“p1”);
tmp.className = “umaClasse”;
```

# Manipulando o CSS pelo DOM

- Para **adicionar** uma classe:

```
var tmp = document.getElementById(“p1”);
tmp.className = “umaClasse”;
(...)
tmp.className += “ outraClasse”;
```

- Atributo classes: “umaClasse outraClasse”

# Manipulando o CSS pelo DOM

- Para **remover** todas as classes:

```
var tmp = document.getElementById(“p1”);
tmp.className = “”;
```

- Para remover **uma** classe: complicado!
- className: “umaClasse outraClasse”

# Como Ler o CSS?

# Lendo o CSS com JavaScript

- Além dos estilos, 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 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?
- Não existe uma forma padrão (unificada) de fazer isso...

# 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");
```

# TUTORIAL

# Jogo da Velha

- A) Região de Debug
- B) Janela de mensagens em HTML
- C) Placar (HTML5)
- D) Áudio (HTML5)

# Conclusões

# Resumo

- DOM:  acesso diretamo a elem. do XHTML
- O que fazer com DOM e seus elementos?
  - Adicionar
  - Criar
  - Modificar
  - Remover
- Acesso ao CSS: não padronizado, mas possível

- **TAREFA**
  - **AV1**

# Próxima Aula


- Complicado!
  - getElement... pouco prático!
  - Buscas: Jquery!

# PERGUNTAS?

# Bom Descanso a Todos!