MC0037 – Programação para Web

Aula 6: CSS – Cascading Style Sheet JavaScript

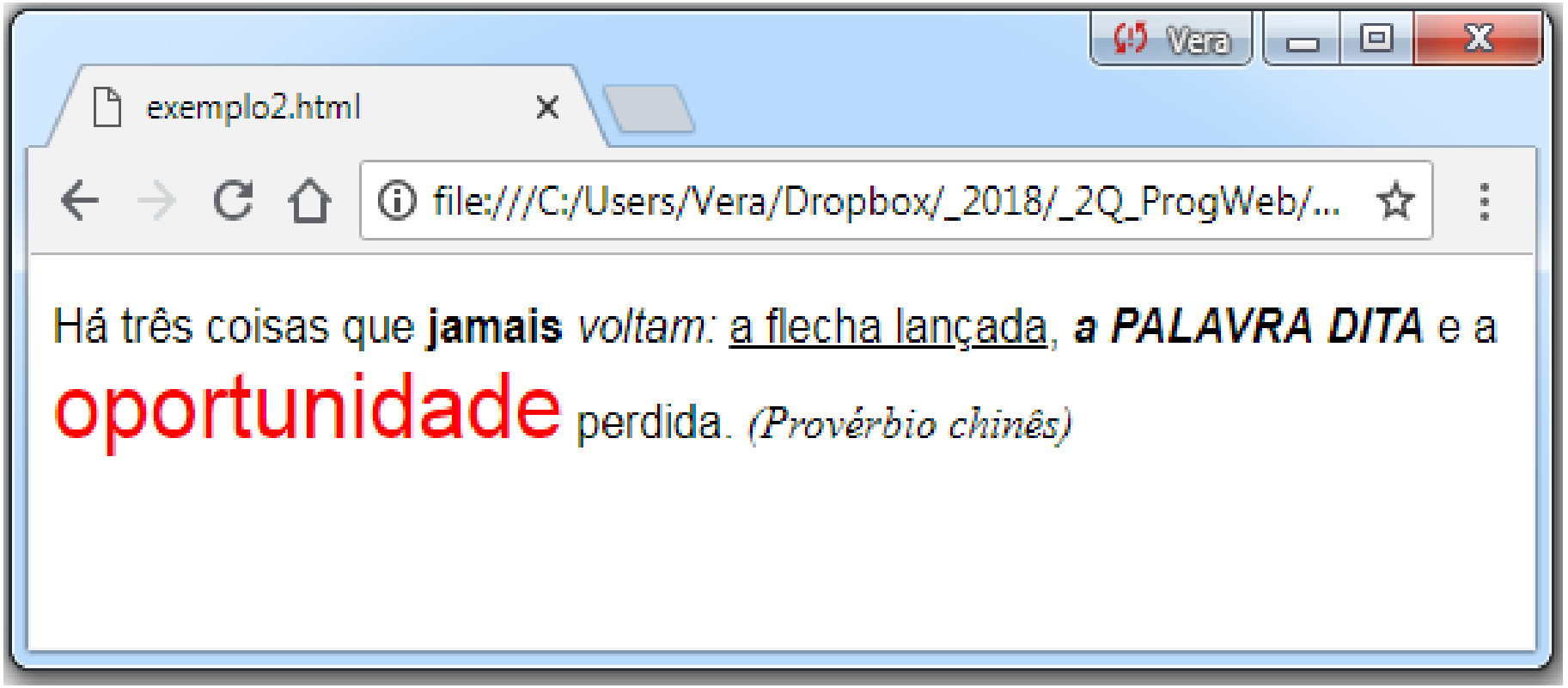
2 Q / 2018

CSS: Cascading Style Sheets

- CSS: Cascading Style Sheets (Folhas de Estilo em Cascata)
- O CSS descreve a aparência, *layout* e apresentação da informação em uma página web (em oposição ao HTML que descreve o conteúdo e a estrutura da página).
- Descreve *como* a informação é apresentada e não *o que* será apresentado.
- É possível definir propriedades de estilo nas próprias tags HTML para modificar a apresentação (aparência) dos elementos e do documento.
  - Por exemplo, estilos de texto, fontes, cores de fundo, alinhamento de conteúdo, formatação de listas e tabelas, etc.

- Principal problema: a apresentação (aparência) e conteúdo + estrutura do documento ficam misturados, dificultando o desenvolvimento e manuntenção.

Exemplo de estilos (tags HTML)



Exemplo de estilos (tags HTML)

- Utilizando tags de estilo do HTML:

```
<p>
   <font face="Arial">Há três coisas que <b>jamais</b>
   <i>voltam:</i> <u>a flecha lançada</u>, <b><i>a
   PALAVRA DITA</b></i> e a <font size="+3"
   color="red"> oportunidade</font> perdida.
   </font> <i>(Provérbio chinês)</i>
</p>
```

Separação de conteúdo e estilo

- Separar o conteúdo / estrutura da apresentação do estilo é importante pelos seguintes aspectos:

  - Facilita o desenvolvimento e a manuntenção das páginas web uma vez que é possível aplicar ou alterar estilo aos elementos de uma ou mais páginas sem ter que acessar todas as páginas e elementos.

  - Permite a reutilização (ou compartilhamento) do estilo em todas as páginas da aplicação, permitindo que tenham uma aparência comum.

CSS

- Há três formas para acrescentar estilo usando o CSS:
- (1) Embutir no próprio elemento HTML (estilo *in-line*);
- (2) Escrever dentro de um documento HTML (folha de estilo interna);
- (3) Criar um ou mais arquivos separados (folha de estilo externa).

- Dessa forma, estilos podem ser empregados em diferentes níveis de granularidade:
  - Em um elemento específico (1)
  - Em uma página específica (2)
  - Em todas as páginas da aplicação (3) → arquivo css

- A principal vantagem de utilizar um arquivo css é que este pode ser compartilhado por muitos documentos HTML da aplicação.

Estilo *in-line*

- Os estilos são aplicados em cada elemento HTML individualmente.
- Utiliza-se o atributo *style*.

- Sintaxe:

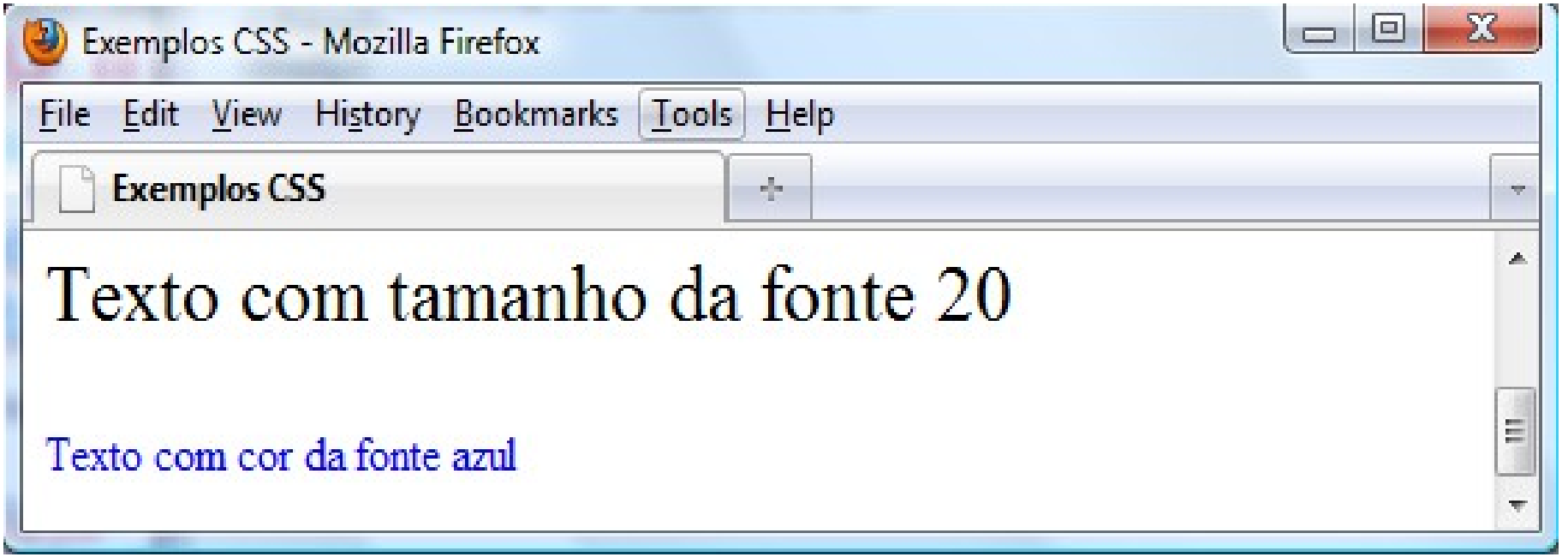
```
<elemento style="propriedade: valor">Texto</elemento>
```

- Obs.: O estilo in-line não faz a separação de conteúdo / estilo, é semelhante às propriedades de estilo do próprio HTML. Apropriado somente para aplicações de estilo pontuais, específicos.

Estilo *in-line*

- Exemplo:

```
<p style="font-size: 20pt">Texto com tamanho da fonte 20</p>
<p style="color: blue">Texto com cor da fonte azul</p>
```



Folha de estilo interna

- Estilos são definidos dentro do documento HTML.
- Os estilos definidos são aplicados internamente, para o documento inteiro.
  - É possível aplicar um estilo padrão para elementos HTML específicos que aparecem várias vezes no documento com apenas uma declaração.
  - Estilos são incluídos na tag <head> do documento HTML usando a tag <style>, coforme abaixo:

```
<head>
<style type="text/css">

   /* definição dos estilos */

</style>
</head>
```

Folha de estilo interna

- Os estilos são definidos da seguinte forma:

```
selector {
   propriedade: valor;
   propriedade: valor;
   ...
   propriedade: valor;
}
```

- O selector ou seletor identifica os elementos HTML que deverão receber o estilo, pondedo ser o nome do elemento, id, classe, atributo, etc.

- O seletor define uma ou mais propriedades e o valor que cada uma destas devem ter.

Folha de estilo interna

- Exemplo:

```
<style type="text/css">
p {
    color: blue;
    font-size: 20pt;
}
</style>
```

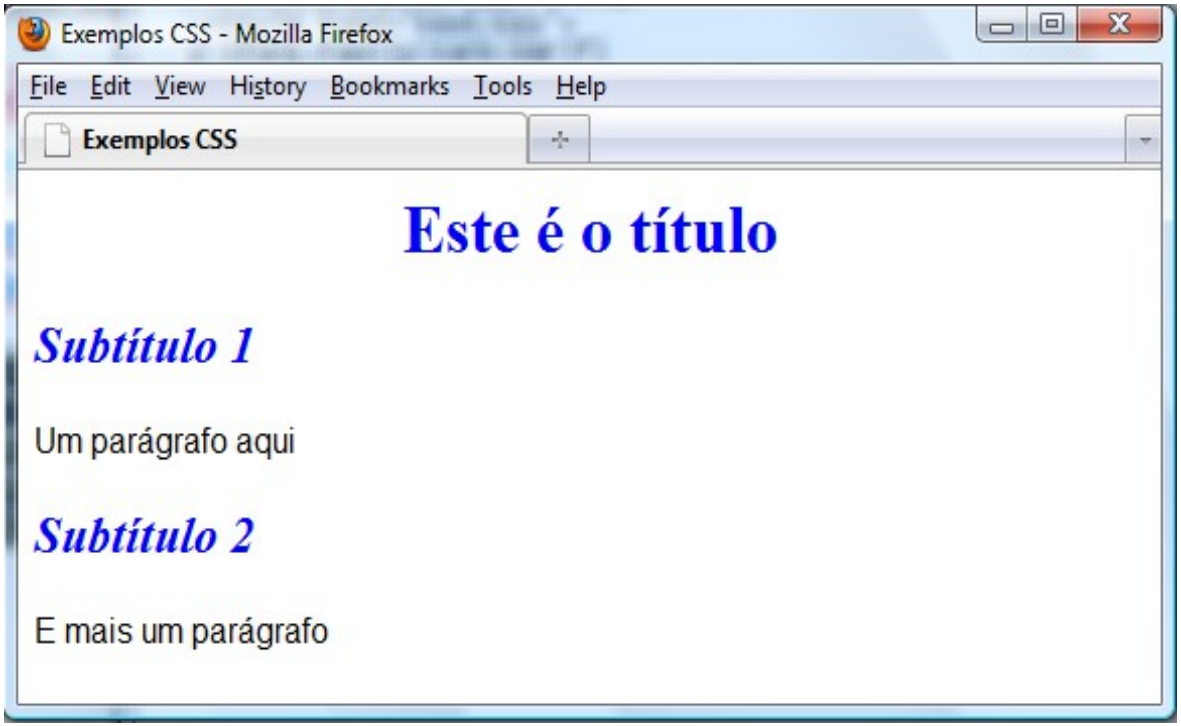
- Neste exemplo, todos os parágrafos receberão os estilos definidos: cor da fonte azul e tamanho da fonte igual a 20
- É possível também aplicar o mesmo estilo para mais de um tipo de elemento, por exemplo:

```
<style type="text/css">
h1, h2 {
    color: blue;
    text-align: center;
}
</style>
```

Folha de estilo interna

- Podemos incluir vários estilos para diferentes elementos no corpo de <style>:

```
<style type="text/css">
p {
    font-family: sans-serif
}

h1 {
    color: blue;
    text-align: center
}

h2 {
    color: blue;
    font-style: italic
}
</style>
```



Folha de estilo externa

- A folha de estilos é especificada em um documento separado.
- A principal vantagem é que é possível aplicar os mesmos estilos por toda a aplicação, fazendo com tenha uma aparência consistente.
  - Ex.: todos os títulos, botões, etc. de todas as páginas terão a mesma aparência.
  - Facilita a manuntenção, é preciso alterar apenas o arquivo.

- Os estilos devem ser salvos em um arquivo com a extensão .css (por exemplo: estilo.css).

Folha de estilo externa

- Exemplo de arquivo css:

```
p {
font-family: sans-serif
}

h1 {
color: blue;
text-align: center
}

h2 {
color: blue;
font-style: italic
}
```

Folha de estilo externa

- Para que a folha de estilos seja aplicada a um documento, é necessário especificar o arquivo CSS para que este possa ser referenciado.
- Para isso, basta adicionar a tag <link> no cabeçalho do documento:

```
<head>
<link href="estilo.css" rel="stylesheet" type="text/css">
<title>Título da Página</title>
<head>
```

- E o arquivo estilo.css deve estar dentro da pasta src/main/webapp/WEB-INF do projeto.

Aplicando estilos com o atributo *class*

- Quando é preciso empregar os mesmos estilos a vários tipos diferentes de elementos HTML, é possível utilizar o atributo *class.*
  - O estilo que será utilizado em vários elementos deve ser nomeado.
  - Todo elemento onde será empregado o estilo deve declarar o atributo *class* associando o nome do estilo definido.
  - Exemplo: (declarações de estilo no próprio documento ou em um arquivo externo).

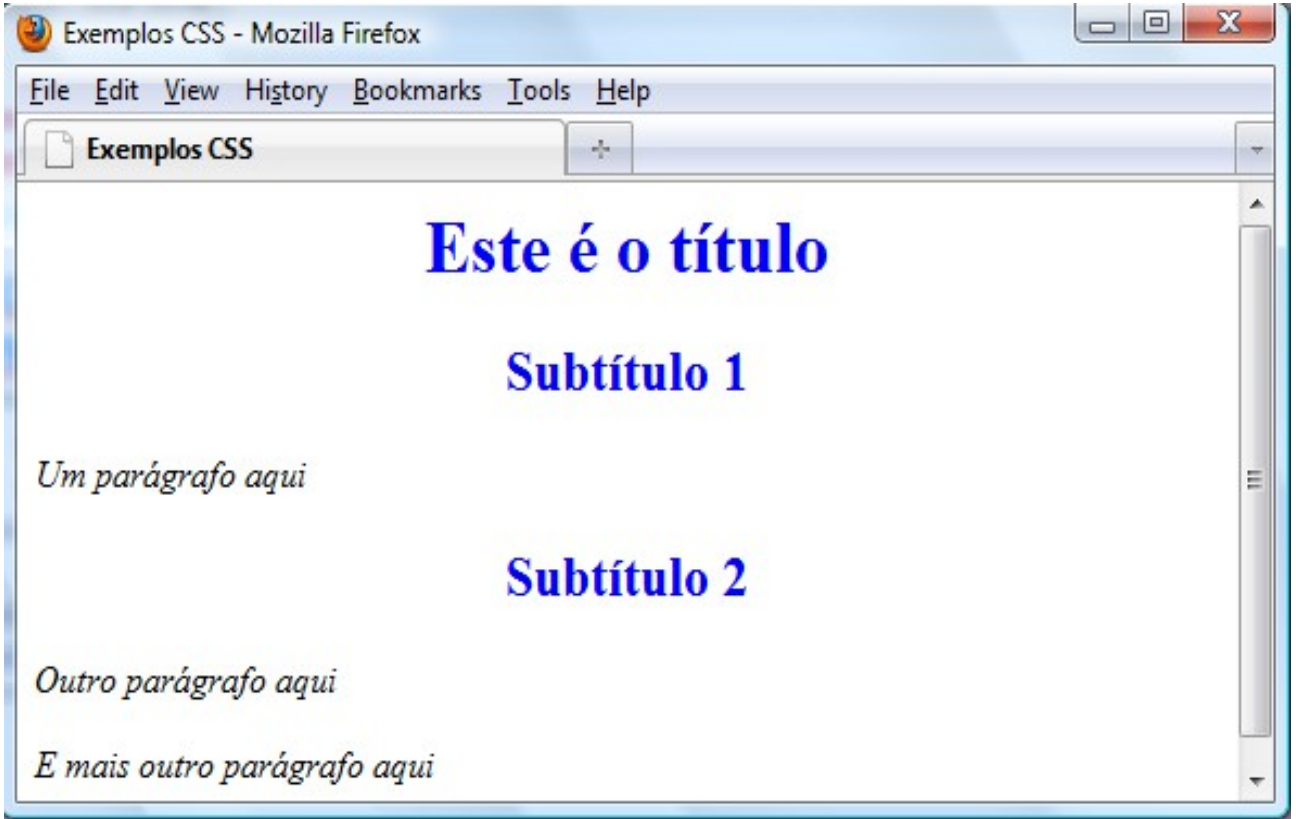
```
.titulo {
    color: blue;
    text-align: center
}
p {
    font-style: italic
}
```

Definição do estilo **titulo.** Observe o ponto no início: **.titulo**

Aplicando estilos com o atributo *class*

- Exemplo:

```
<h1 class="titulo">Este é o título</h1>
<h2 class="titulo">Subtítulo 1</h1>
<p>Um parágrafo aqui</p>
<h2 class="titulo">Subtítulo 2</h2>
<p>Outro parágrafo aqui</p>
<p>E mais outro parágrafo aqui</p>
```



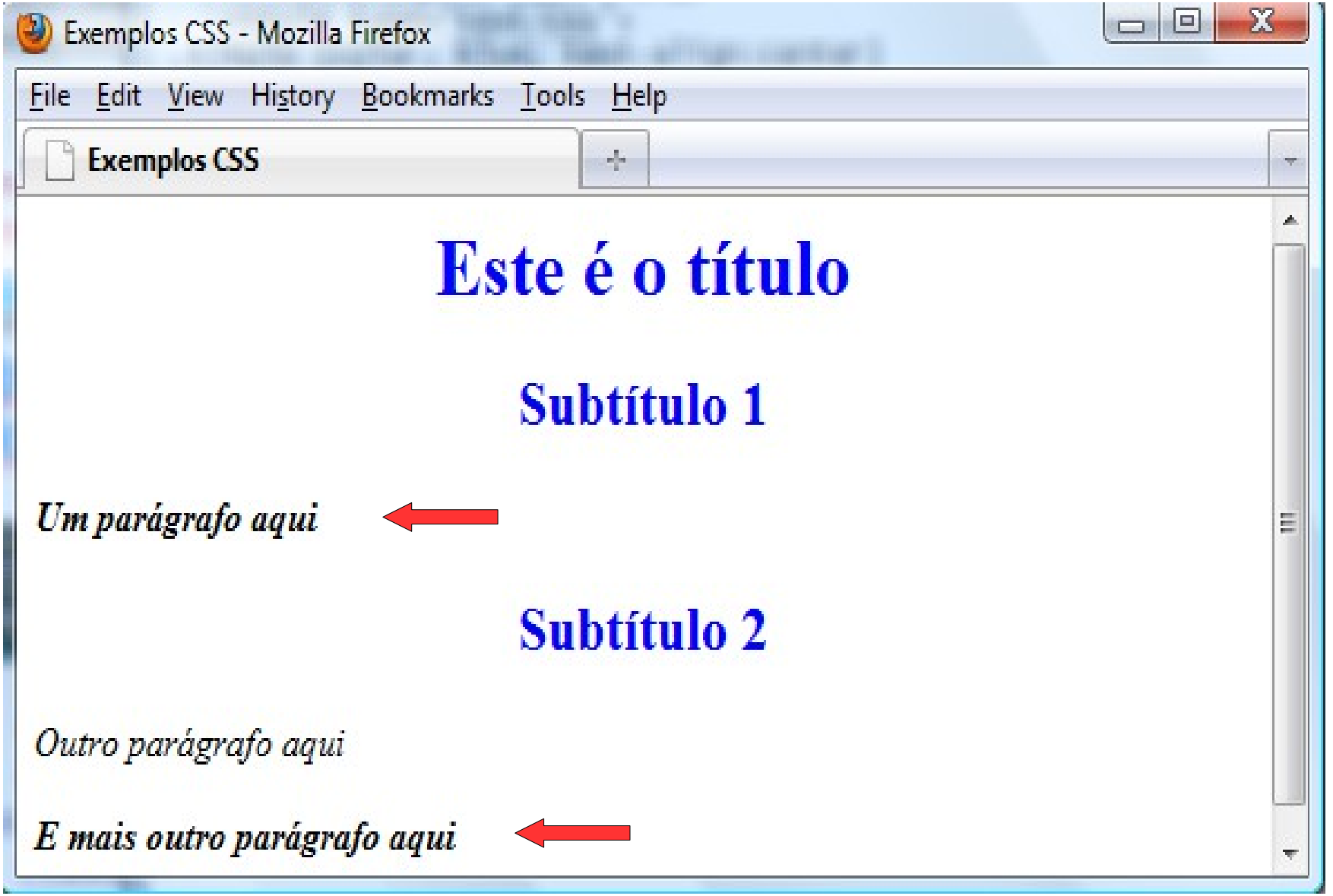
Aplicando estilos com o atributo *class*

- O atributo *class* pode ser utilizado também para atribuir estilo a um subconjunto de elementos de um mesmo tipo:

```
.titulo {
    color: blue;
    text-align: center
}
p {
    font-style: italic
}
p.enfase {
    font-weight: bold
}
```

```
<h1 class="titulo">Este é o título</h1>
<h2 class="titulo">Subtítulo 1</h2>
<p class="enfase">Um parágrafo aqui</p>
<h2 class="titulo">Subtítulo 2</h2>
<p>Outro parágrafo aqui</p>
<p class="enfase">E mais outro parágrafo aqui</p>
```

Aplicando estilos com o atributo *class*



Aplicando estilos com o atributo *id*

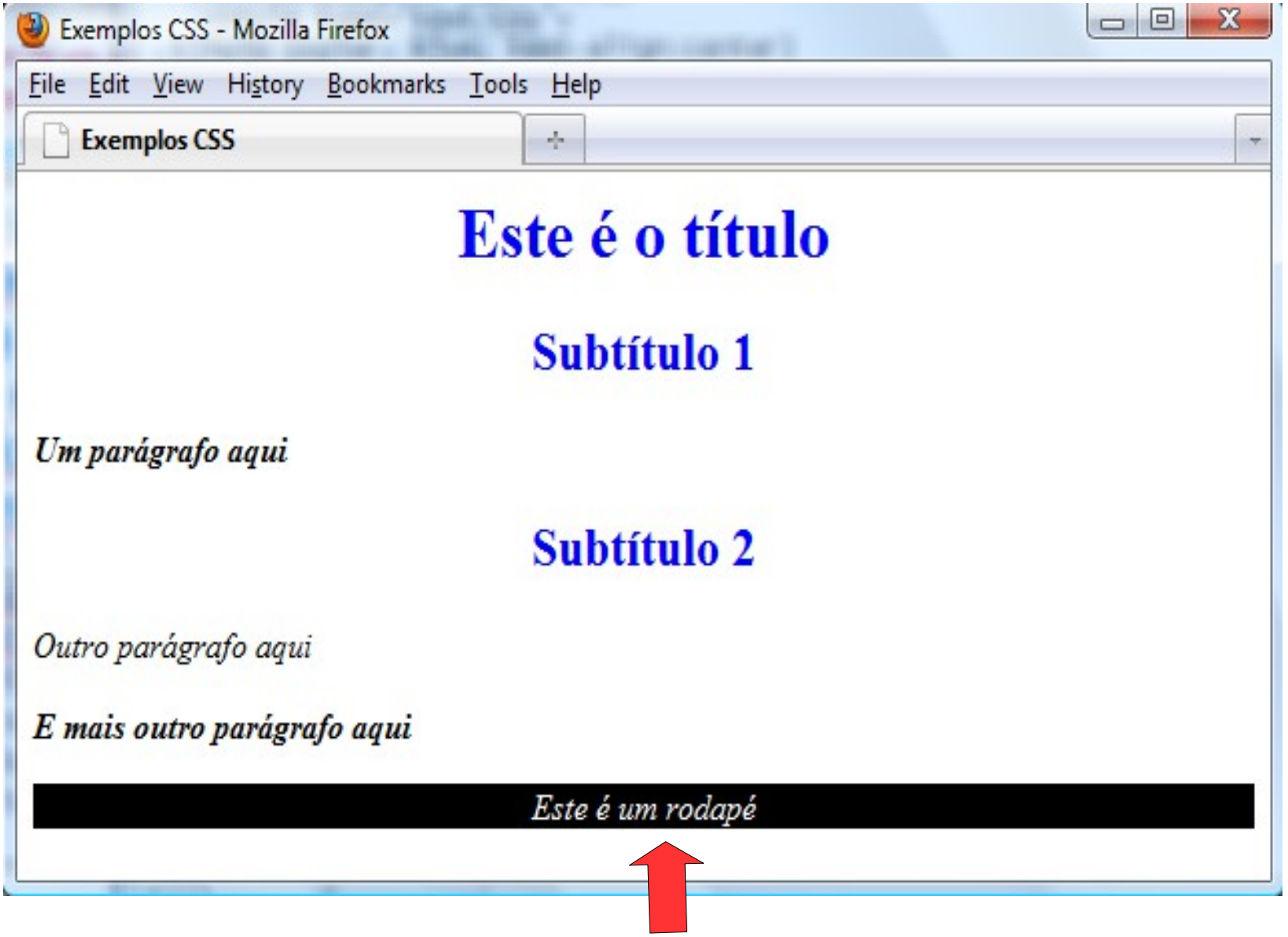
- Há situações em que é necessário aplicar estilos específicos em apenas um dado elemento.
- Para isso, deve ser especificado o atributo *id* do elemento, que é um identificador único do elemento no documento e associar um estilo a esse identificador.
- Exemplo:

```
#rodape {
    font-weigth: bold;
    font-style: italic;
    color: white;
    background-color: black;
    text-align: center
}
```

Definição de estilo para o id **rodape.** Observe a # no início: **#rodape**

```
<p id="rodape">Este é um rodapé</p>
```

Aplicando estilos com o atributo *id*



Estilos conflitantes

```
body { font-family: sans-serif; background-color:
yellow; }
p { color: red; background-color: aqua; }
a { text-decoration: overline underline; }
h2 { font-weight: bold; text-align: center; }
p.especial { color: blue; background-color: black; }
```

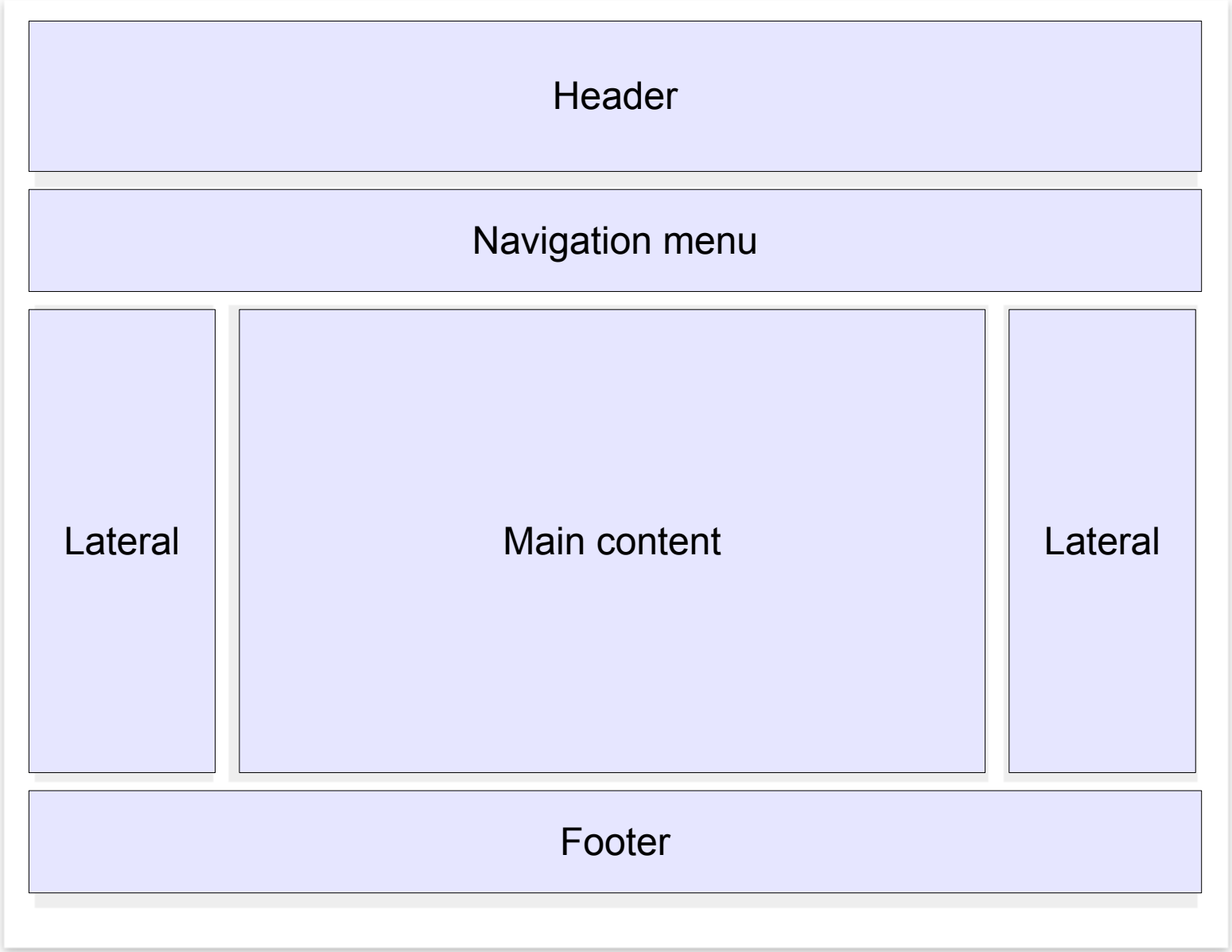
- Quando há múltiplos estilos a serem aplicados a um elemento, eles são herdados.
- Estilos mais específicos sobrepõem os mais gerais.
- Nem todas as propriedades são herdadas (cor do link).

Propriedades do CSS

- Há inúmeros estilos, propriedades para fontes, textos, tabelas, listas, *links*, *backgrounds,* etc., além de recusos avançados, animação, etc., uma sugestão é o site:

  http://www.w3schools.com/css/default.asp

Exemplo de layout comum de websites



Fonte:https://www.w3schools.com/css/css_website_layout.asp

Exemplo com CSS

Exemplo com CSS - header

```
<style>
body {
  margin: 0;
  background-color: lightblue;
}

/* header ou cabeçalho */
.header {
    color: brown;
    background-color: #6495ED;
    background-image: url("img/logo.jpg");
    background-repeat:no-repeat;
    background-position: left;
    border: solid gray;
    text-align: center;
    padding: 20px;
}
...
```

Exemplo com CSS - header

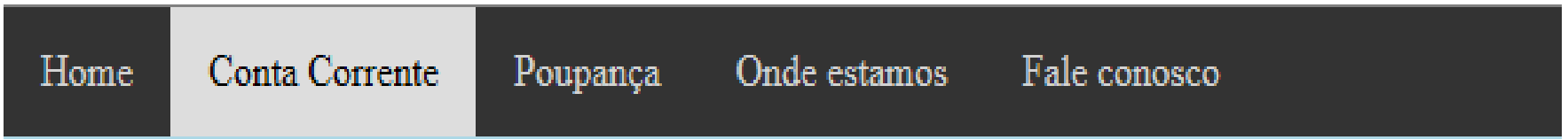
```
<body>

<div class="header">
  <h1>Sistema Bancário - UFABC</h1>
</div>
  ...
```

Exemplo com CSS – menu horizontal

```css
.menu {
    overflow: hidden;
    background-color: #333;
}
/* links do menu*/
.menu a {
    float: left;
    display: block;
    color: lightgray;
    text-align: center;
    padding: 14px 16px;
    text-decoration: none;
}
/* muda a cor do link qdo passa o mouse */
.menu a:hover {
    background-color: #ddd;
    color: black;
}
```

Exemplo com CSS – menu horizontal

```
<div class="menu">
  <a href="#">Home</a>
  <a href="#">Conta Corrente</a>
  <a href="#">Poupança</a>
  <a href="#">Onde estamos</a>
  <a href="#">Fale conosco</a>
</div>
```



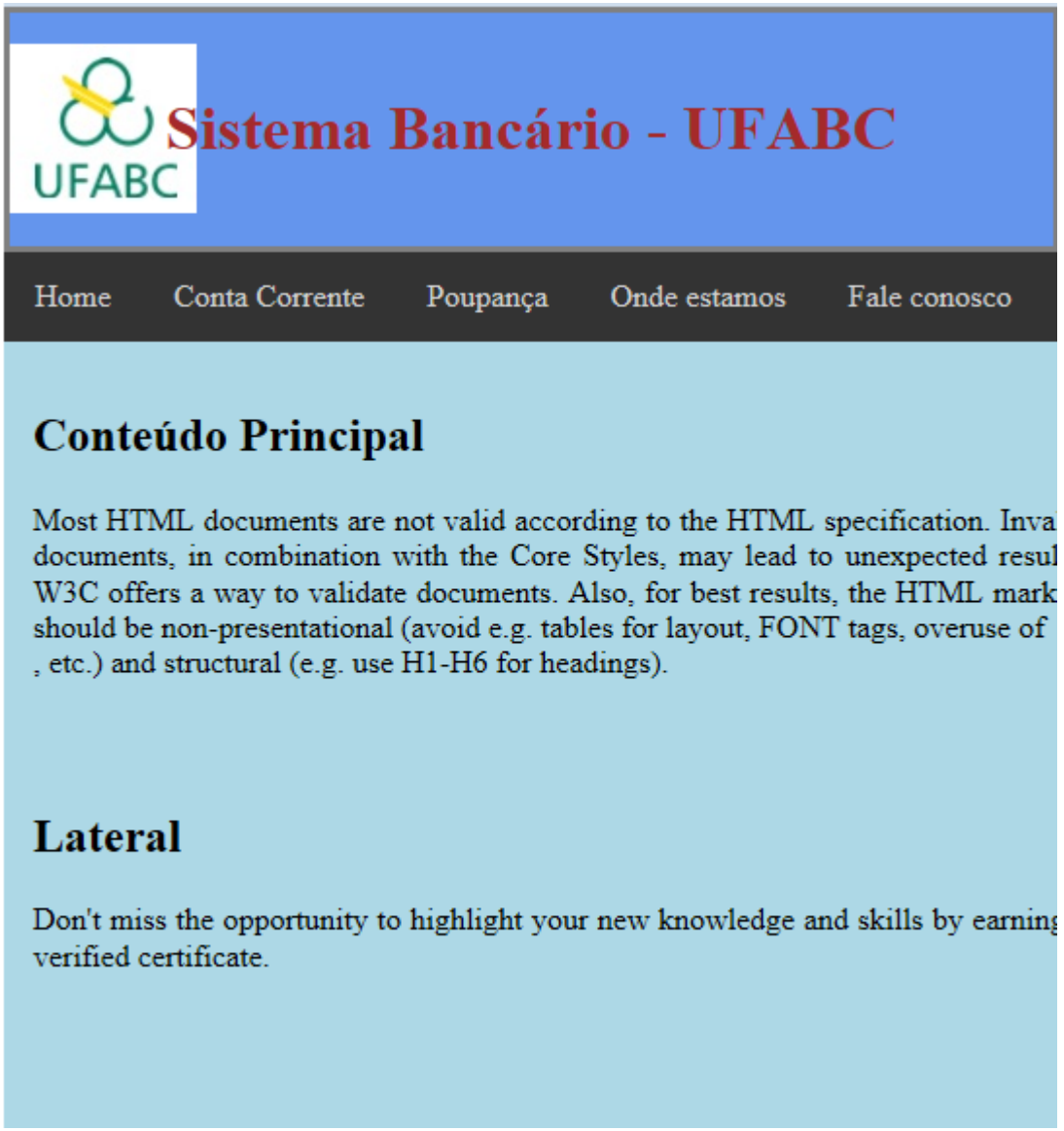
Exemplo com CSS - conteúdo

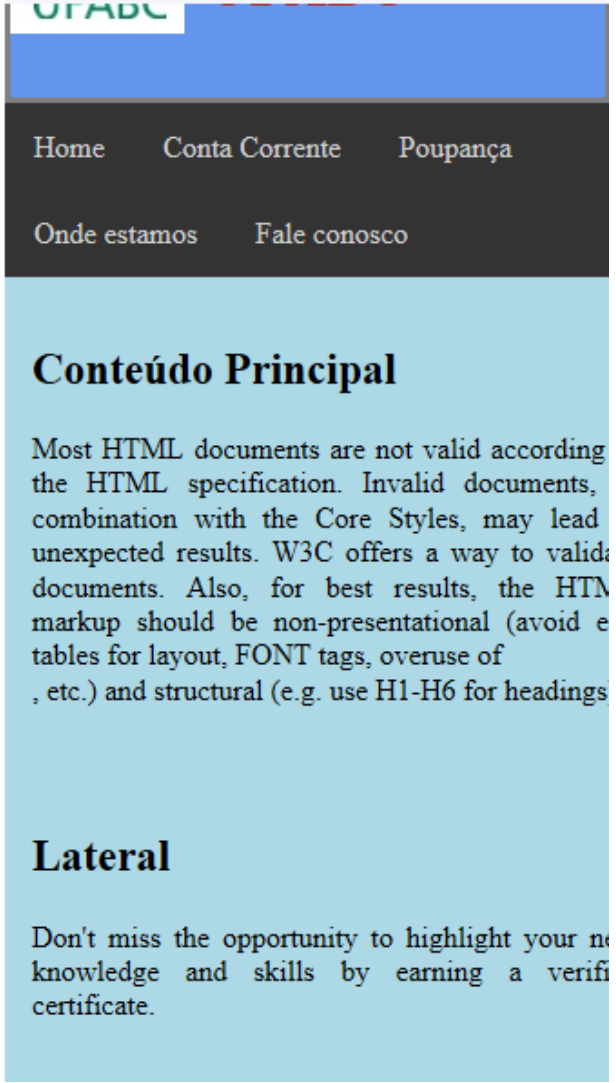
```
/* Definindo colunas */
.column {
    float: left;
    padding: 15px;
}
/* Coluna principal */
.column.middle {
    width: 70%;
    text-align: justify;
}
/* coluna lateral */
.column.side {
    width: 20%;
    text-align: justify;
}
/* permite layout responsivo: se a tela for menor que 600px
apresenta a coluna lateral abaixo da principal*/
@media screen and (max-width: 600px) {
    .column.side, .column.middle {
        width: 100%;
    }
}
```

Exemplo com CSS - conteúdo

```
<div class="column middle">
    <h2>Conteúdo Principal</h2>
    <p>
        Most HTML documents are not valid according to the HTML
        specification. Invalid documents, in combination with the Core
        Styles, may lead to unexpected results. W3C offers a way to validate
        documents. Also, for best results, the HTML markup should be
        non-presentational (avoid e.g. tables for layout, FONT tags, overuse
        of <BR>, etc.) and structural (e.g. use H1-H6 for headings).
    </p>
</div>
<div class="column side">
    <h2>Lateral</h2>
    <p>Don't miss the opportunity to highlight your new knowledge and
        skills by earning a verified certificate.</p>
</div>
```

Exemplo com CSS – conteúdo





Exemplo com CSS – formulário

Exemplo com CSS – formulário

```css
input[type=text], select {
    width: 100%;
    padding: 12px 20px;
    margin: 8px 0;
    display: inline-block;
    border: 1px solid #ccc;
    border-radius: 4px;
    box-sizing: border-box;
}
input[type=submit] {
    width: 100%;
    background-color: #4CAF50;
    color: white;
    padding: 14px 20px;
    margin: 8px 0;
    border: none;
    border-radius: 4px;
    cursor: pointer;
}
input[type=submit]:hover {
    background-color: #45a049;
}
```

Exemplo com CSS - formulário

```
<div class="form">
  <form action="insere">
    <label for="numero">Número da conta:</label>
    <input type="text" id="numero" name="numero" placeholder="Insira o
número da conta corrente.">

    <label for="agencia">Agência:</label>
    <input type="text" id="agencia" name="agencia" placeholder="Insira a
agência.">
    <label for="descricao">Descrição:</label>
    <input type="text" id="descricao" name="descricao"
placeholder="Descrição da conta.">
    <label for="variacao">Variação:</label>
    <select id="variacao" name="variacao">
      <option value="0"> 0 </option>
      <option value="1"> 1 </option>
      <option value="2"> 2 </option>
    </select>

    <input type="submit" value="Enviar">
  </form>
```

MC0037 – Programação para Web

JavaScript

2 Q / 2018

JavaScript

- JavaScript é uma linguagem de *script* (linguagem de programação "leve", mas poderosa)
  - Possui uma sintaxe mais simples
  - É uma linguagem interpretada (os scripts executam sem compilação preliminar)
  - É a mais utilizada dentre as linguagens de script dos navegadores web
  - É usualmente embutida diretamente nas páginas HTML ou referenciada em arquivos externos para gerar conteúdo da página dinamicamente
  - É principalmente utilizada para tornar as páginas HTML mais funcionais, eficientes e interativas para o usuário
    - Inserir textos/imagens dinamicamente na página
    - Reagir a eventos do usuário (ex.: clique de mouse em um botão)
    - Executar validação de campos de formulário no cliente (reduz a carga no servidor)

Características do JavaScript

- Em parte, o JavaScript é similar a outras linguagens como C, C++ ou Java, porém há importantes diferenças
  - JavaScript é fracamente tipada: quando declaramos uma variável, não precisamos declarar o seu tipo
  - O tipo da variável é definido quando for atribuído um valor - na inicialização da variável ou em tempo de execução (*dynamic variable typing*)
  - Uma mesma variável pode referenciar diferentes tipos de dados em diferentes momentos (isso não é possível em linguagens fortemente tipadas como Java)
  - Javascript ≠ Java

Adicionando scripts

- O código JavaScript pode ser adicionado em uma página web de três formas:
  - Dentro do elemento <body> (executa quando a página é carregada)
  - Dentro do elemento <head> (executa na ocorrência de um evento)
  - Através de um link para um arquivo externo (arquivo .js)

- Scripts podem ser adicionados a uma página web usando a tag <script>
- Sintaxe:

```
<script type="text/javascript">
    // codigo JavaScript
</script>
```

Exemplo de JavaScript

- Inserir dinamicamente um texto na página HTML
  - *document*: objeto que representa o documento HTML

```
<script type="text/javascript">
    document.write("<h2>Primeiro programa JavaScript!</h2>");
    document.write("Escreve uma mensagem");
</script>
```

Usando um arquivo externo

- Sintaxe:

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

- Exemplo:

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

- Esta tag é usualmente inserida no elemento <head>, embora seja possível no elemento <body>
- Arquivos que contém código JavaScript devem ter a extensão .js
- Utilizar arquivo externo é preferido se o código JavaScript for reutilizado em várias páginas, facilitando a manutenção do código

Variáveis e tipos

- Declaração de variáveis:

```
var nome;
var nome = valor;
```

- Exemplos:

```
var nomeCliente = "John";
var idade = 33;
var altura = 1.80;
altura = "1.80";  // atribui uma string
```

- Uma variável é declarada com a palavra-chave **var** (Javascript é case-sensitive: faz distinção entre maiúsculas e minúsculas)
- O tipo da variável não é declarado, a inicialização da variável define o tipo de dado

Tipos de dados

Tipo	Descrição	Exemplos
Number	Número inteiro ou real	42, -16, 3.14, 2.4e-6
String	Cadeia de caracteres	“Bom dia”, 'Boa noite'
Boolean	Valor lógico	true, false
Array	Lista indexada (array)	[12, 17, -7, 22.5]
Object	Objeto	{nome: John, age: 23}

Operadores aritméticos

Operador	Descrição
+	Soma
-	Subtração
*	Multiplicação
/	Divisão
%	Módulo (resto da divisão)

- Operadores unários (semelhantes ao Java):

  ++, - -, +=, -=, *=, /=, %=

- Ordem de precedência dos operadores é similar ao Java
- Muitos operadores fazem conversão automática de tipo:
  - “2” * 3 (resulta em 6)

Operadores relacionais e lógicos

- Operadores relacionais: >, <, >=, <=, ==, !=, ===, !==
- Operadores lógicos: &&, ||, !
- A maioria dos operadores lógicos convertem os tipos automaticamente:

```
5 < "7"  // verdadeiro
42 == 42.0  // verdadeiro
"5.0" == 5  // verdadeiro
"5.0" === 5 // falso
```

- === e !== são testes de igualdade estrita, checa ambos, o tipo e o valor

Strings e valores numéricos

```
var s = "Nome Sobrenome";
var primeiroNome = s.substring(0,s.indexOf(" ")); // Nome
var comprimento = primeiroNome.length;   // 4

var count = 10;           // 10
var s1 = "" + count;      // "10"
var s2 = count + " bananas";   // "10 bananas"
var n1 = parseInt("42");    // 42
var n2 = parseInt("x");     // NaN
var texto;                  // undefined
```

- parseInt: converte uma string em um inteiro
- parseFloat: converte uma string em um real
- NaN: Not a Number (é retornado apenas pela função isNaN())
- undefined: quando a variável não foi declarada ou foi declarada e não foi inicializada

Objeto Date

- O objeto Date representa ambos, a data e hora e precisa ser criado quando for utilizá-lo (usando comando *new*)

```
var hoje = new Date(); // cria um objeto com a data atual
var umaData = new Date(2016, 4, 4); // 4 de maio de 2016

var ano = hoje.getFullYear();   // 2016
var dia = hoje.getDay(); // 5 (dia da semana de 0
                         // (domingo) a 6 (sábado)
var diaMes = hoje.getDate();//dia do mês(1 a #dias do mês)
var mes = hoje.getMonth(); // mês de 0-11 (e não 1-12)
```

Arrays

- Um array é uma coleção de valores que tem o mesmo nome e são identificados por um índice (em colchetes)
- Um array é criado usando o operador *new*
- Exemplo:

```
var array1 = new Array(3); // cria um array com 3 posições
array1[0] = "uma string";
array1[1] = 7;
var array2 = [1, 2, 3]; // declaração e inicialização
```

- O array não possui um tipo específico, os elementos do array podem conter valores de diferentes tipos

- Veja os métodos e propriedades de array em:
  http://www.w3schools.com/jsref/jsref_obj_array.asp

Alguns comandos

- Comando if-else

Sintaxe:

```
if (condição) {
    // comandos
} else if (condição) {
    // comandos
} else {
    // comandos
}
```

Exemplo:

```
<script type="text/javascript">
    var d = new Date()
    var time = d.getHours()
    if (time < 10) {
        document.write("<b>Bom dia</b>");
    } else if (time > 10 && time < 18) {
        document.write("<b>Boa tarde</b>");
    } else {
        document.write("<b>Boa noite!
</b>");
    }
</script>
```

Laço for

- Sintaxe:

```
for (inicialização; condição; incremento) {
    // comandos
}
```

- Exemplo:

```
<script type="text/javascript">
    for (var i = 0; i < 10; i++) {
        document.write("<p>" + i + " ao quadrado = " + (i * i) +
"</p>");
    }
</script>
```

Laço while

- Sintaxe:

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

Ou:

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

Funções

- Uma função em JavaScript é declarada usando a palavra *function*, seguida do nome da função e lista de parâmetros:

```
<script type="text/javascript">
    function nomeFuncao (parametro1, parametro2, ...) {
        //  ...
        return valor;
    }
</script>
```

- A função pode retornar algum valor na última linha através do comando *return* (senão, é devolvido o valor *undefined*)
- Os tipos de parâmetros e de retorno não são declarados
- A declaração *var* não é escrita nos parâmetros
- As funções podem ser escritas nas seções <head> ou <body> de um documento HTML

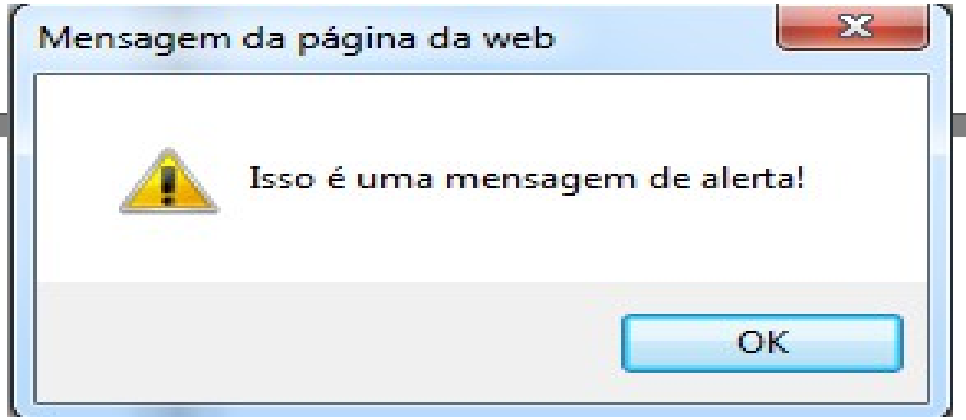
Chamada de função

```
<head>
<title>Teste funcao</title>
<script type="text/javascript">
    function quadratica(a, b, c) {
        return (-b + Math.sqrt(b * b - 4 * a * c) / (2 * a));
    }
</script>
</head>
<body>
    <script type="text/javascript">
        document.write("x1 = " + quadratica(3, 4, 1));
    </script>
</body>
```

Janelas Pop-up

- Janela de Alerta

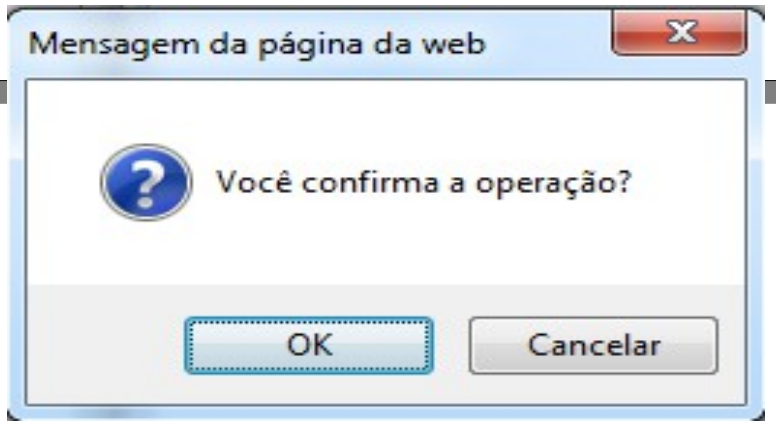
```
<script type="text/javascript">
    alert("Isso é uma mensagem de alerta!");
</script>
```

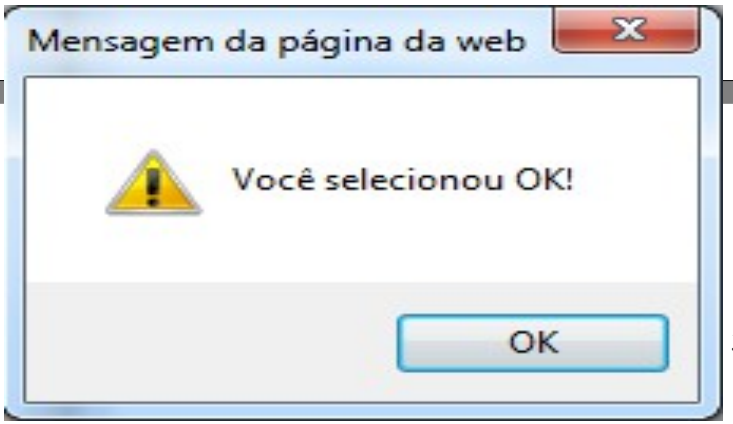


Janelas Pop-up

Janela de Confirmação

```
<script type="text/javascript">
    var resp = confirm("Você confirma a operação?");
    if (resp == true) {
        alert("Você selecionou OK!");
    } else {
        alert("Você selecionou Cancel!");
    }
</script>
```

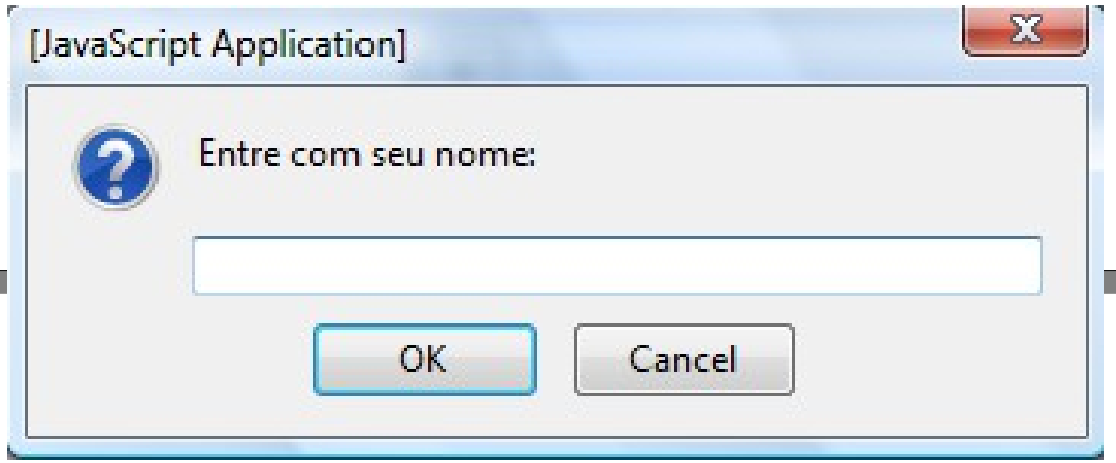




Janelas Pop-up

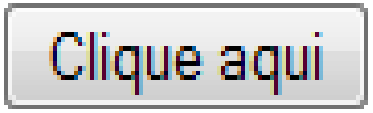
- Janela de Entrada de dados (prompt)

```
<script type="text/javascript">
    var nome = prompt("Entre com seu nome:");
    if (nome != "") {
        document.write("Olá " + nome + "! Bem-vindo!");
    }
</script>
```



Eventos do usuário

(1) Usuário interage com a página



(2) Gera um evento

(3) Uma função é excutada em resposta

```
function minhaFuncao() {

                …
 }
```

Tratador do evento

(4) Um resultado é apresentado ou uma parte da página é atualizada / modificada

O programa aguarda por ações do usuário (eventos)

Função para tratar um evento

- Chamando uma função JavaScript para tratar um evento de botão (*onclick*)

```
<head>
<title>Teste</title>
<script type="text/javascript">
    function mostra_alerta() {
        alert("Isso é uma mensagem de alerta!");
    }
</script>
</head>
<body>
    <input type="button" onclick="mostra_alerta()" value="alerta" />
</body>
```

Função para tratar um evento

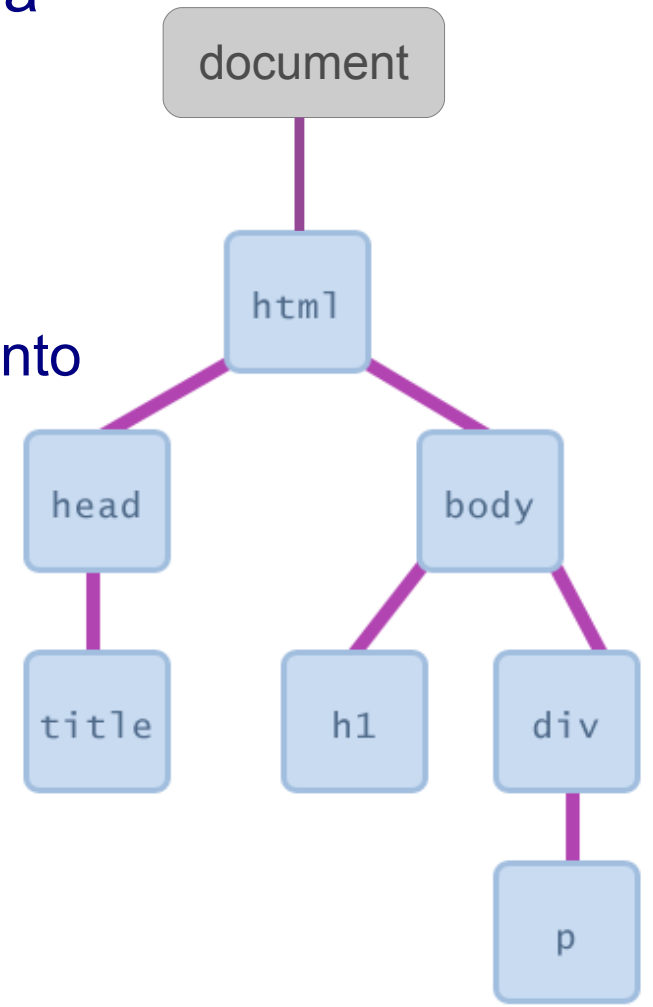
- Os elementos HTML podem ser associados a eventos para a interação com o usuário (ex.: *onclick*)
- As funções JavaScript podem ser codificadas como tratadoras de eventos
  - Quando o usuário interage com o elemento, a função será invocada (ex.: um clique em um botão)
  - A função é executada *somente* na ocorrência de um evento

- Vários tipos de eventos podem ser associados a elementos HTML: *onclick, onKeyPress, onMouseUp, onMouseOver,* etc., dependendo do elemento

- Veja uma lista de tipos de eventos em:
  http://www.w3schools.com/jsref/default.asp

Document Object Model (DOM)

- Representa o conteúdo da página HTML em uma estrutura hierárquica de nós (árvore que representa o documento HTML)
- O JavaScript manipula elementos em uma página HTML através desses objetos
  - Inserir ou remover elementos da página
  - Pode alterar um elemento
    - Ex.: Trocar um texto de um parágrafo
  - Examinar e mudar o estado de um elemento
    - Ex.:Mudar um *checkbox* de não selecionado para selecionado
  - Mudar o estilo dos elementos
    - Ex.: Mudar a cor de um título, parágrafo, etc.
  - Permite maior interatividade com o usuário, fazer validação de campos durante o preenchimento

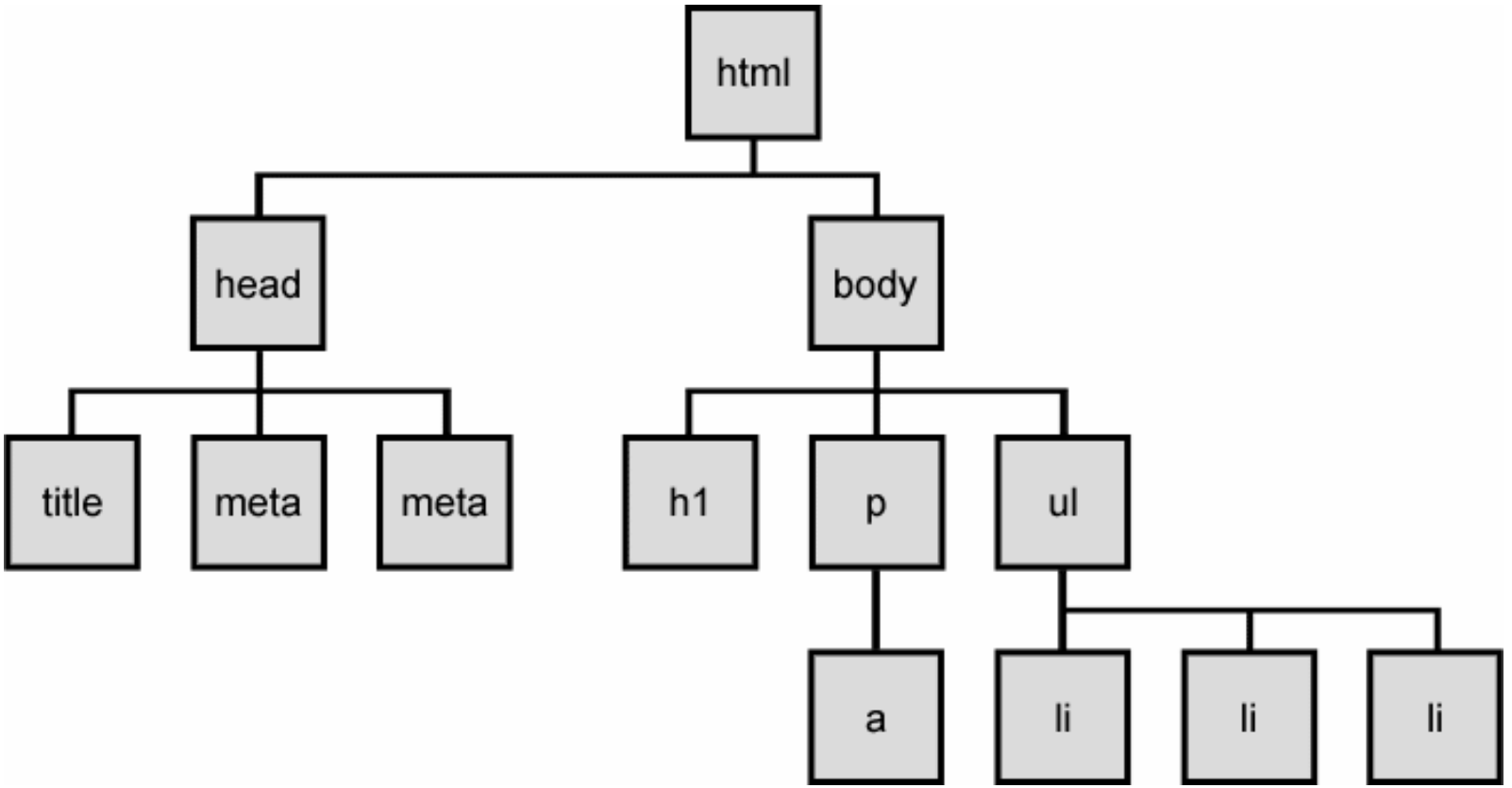


Exemplo: árvore DOM

```
<!DOCTYPE  html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html xmlns="http://www.w3.org/1999/XHTML">
<head>
<title>Titulo</title>
<meta http-equiv="content-type" content="text/html; charset=utf-8"/>
<meta http-equiv="Content-Language" content="pt-br" />
</head>
<body>
<h1>Um título</h1>
<p>Um parágrafo com um <ahref="http://www.google.com/">link</a>.</p>
<ul>
<li>um item</li>
<li>outro item</li>
<li>e mais item</li>
</ul>
</body>
</html>
```

Exemplo: árvore DOM

- Cada um dos elementos da página corresponde a um nó da árvore DOM



- Os navegadores geram a árvore de objetos DOM em tempo de execução

Acessando elementos

- Cada elemento HTML corresponde a um nó da árvore
- Um nó pode ser acessado através do método *getElementById*:

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

- O método *getElementById* devolve o nó correspondente ao elemento HTML dado o id (devolve null se não existir)

- Exemplo: Suponha o seguinte elemento na página HTML:

```
<img src="praia.gif" id="praia" />
```

- Alterando a imagem do elemento no JavaScript:

```
<script type="text/javascript">
    var imagem = document.getElementById("praia");
    imagem.src = "montanha.gif";
</script>
```

Acessando elementos

- Exemplo: Trocar a imagem ao clicar no botão

```
<head>
<title>Exemplo</title>
<script type="text/javascript">
    function trocarImagem() {
        var imagem = document.getElementById("praia");
        imagem.src = "montanha.gif";
    }
</script>
</head>
<body>
    <input type="button" onclick="trocarImagem()" value="Clique
aqui" />
    <img src="praia.gif" alt="Praia" id="praia" />
</body>
```

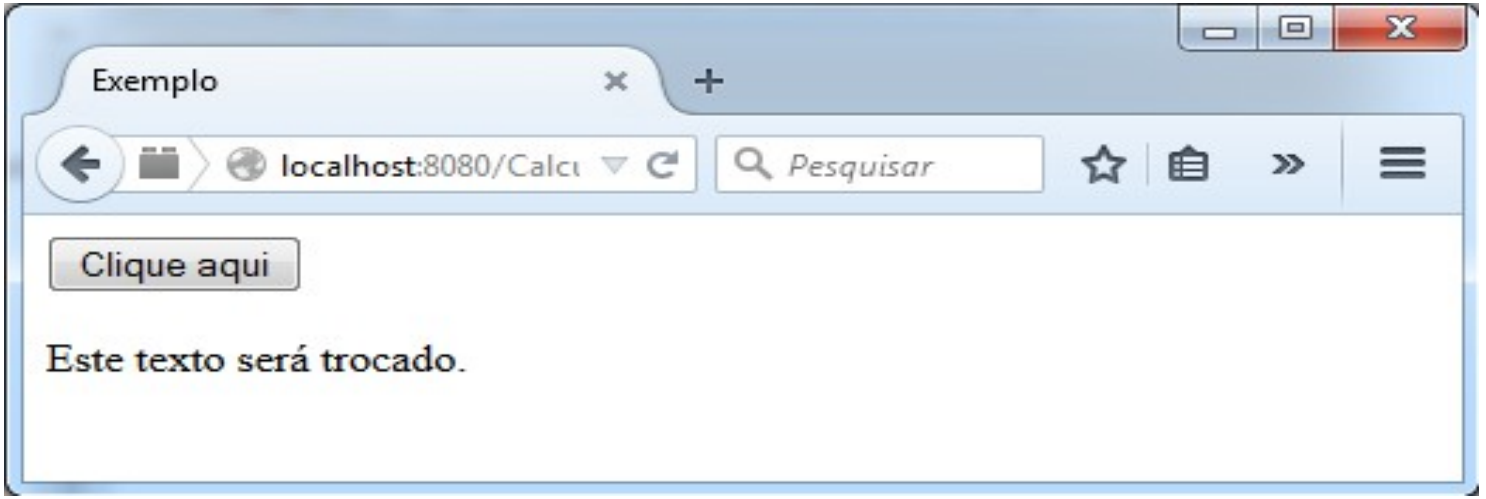
Trocando o texto de um parágrafo

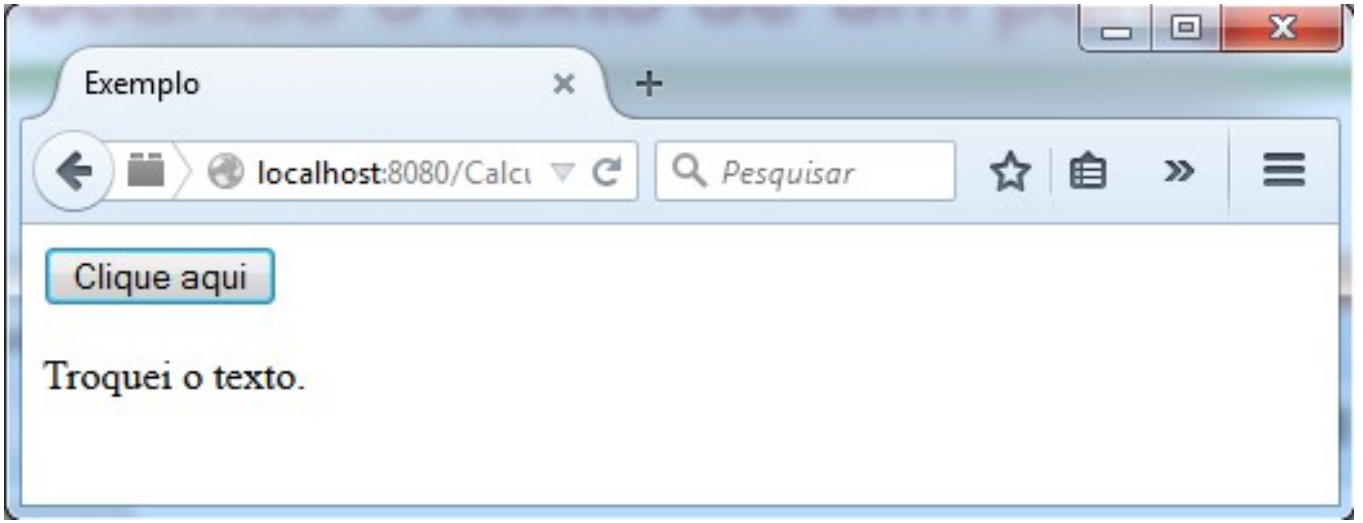
- Exemplo: Trocar um texto de parágrafo

```
<head>
<title>Exemplo</title>
<script type="text/javascript">
     function trocarTexto() {
        var p = document.getElementById("paragrafo");
        p.innerHTML = "Troquei o texto.";
     }
</script>
</head>
<body>
  <input type="button" onclick="trocarTexto()" value="Clique aqui" />
  <p id="paragrafo">Este texto será trocado.</p>
</body>
```

- *innerHTML* se refere ao texto HTML em um elemento:
  Este é o innerHTML da tag p

Trocando o texto de um parágrafo





Acessando elementos

- Para obter todos os objetos de um mesmo tipo, podemos utilizar:

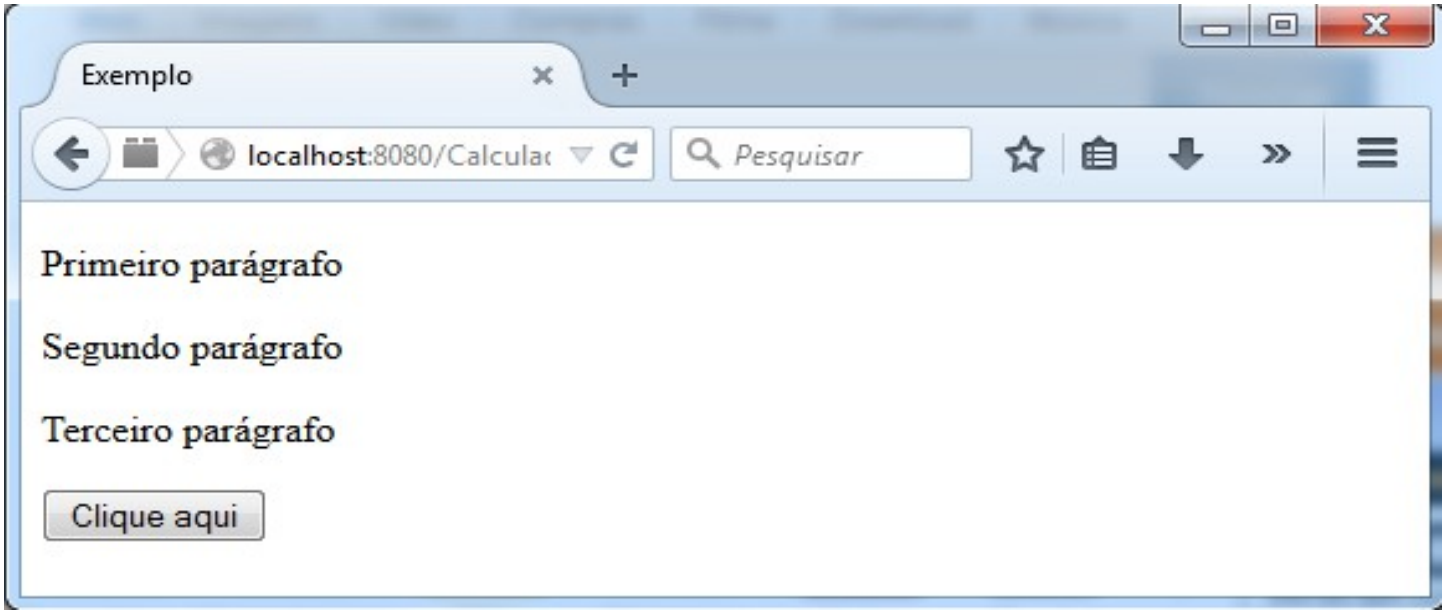
```
var objetos = document.getElementsByTagName("tag");
```

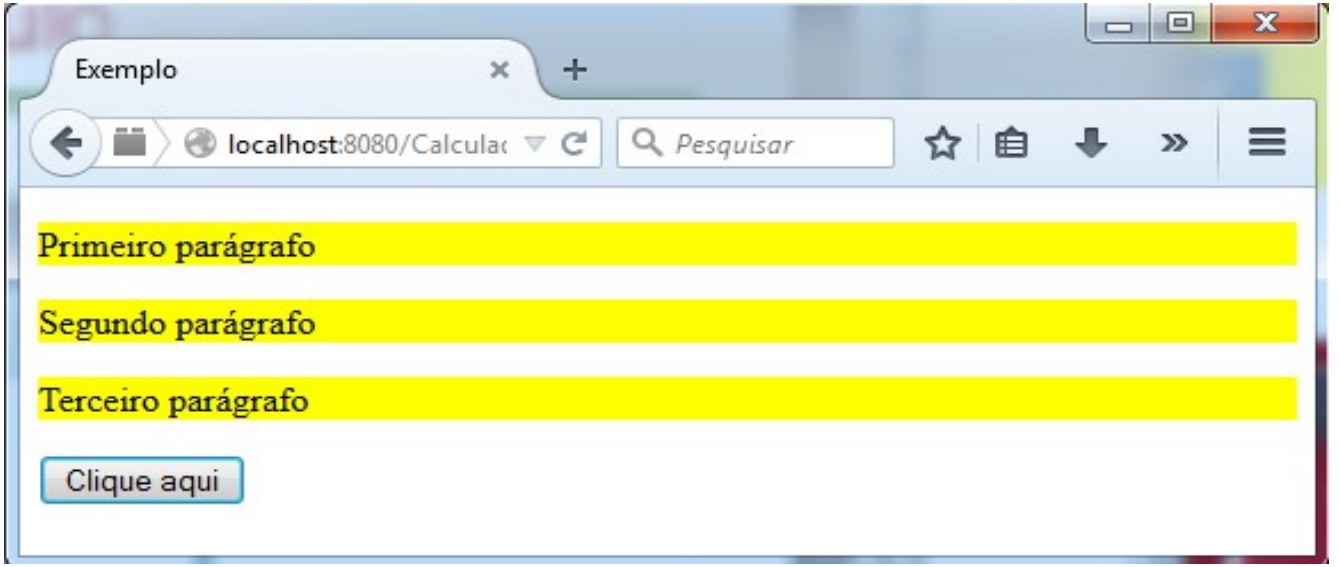
  - Este método retorna um array com todos os nós do mesmo tipo (por exemplo, tag “p”)
  - Pode ser chamado através do objeto *document* ou por um nó específico

Outras formas para acessar nós

- Exemplo: mudar a cor de todos os parágrafos no documento

```
<head>
<title>Exemplo</title>
<script type="text/javascript">
    function mudaCor() {
        var paragrafos = document.getElementsByTagName("p");
        for (var i = 0; i < paragrafos.length; i++) {
            paragrafos[i].style.backgroundColor = "yellow";
        }
    }
</script>
</head>
<body>
    <p>Primeiro parágrafo</p>
    <p>Segundo parágrafo</p>
    <p>Terceiro parágrafo</p>
    <input type="button" onclick="mudaCor()" value="Clique aqui" />
</body>
```


Exemplo
localhost:8080/Calcula
Pesquisar
Primeiro parágrafo
Segundo parágrafo
Terceiro parágrafo
Clique aqui


Exemplo
localhost:8080/Calcula
Pesquisar
Primeiro parágrafo
Segundo parágrafo
Terceiro parágrafo
Clique aqui

Exemplo: validação de campos

```
<head>
<script type="text/javascript">
        function validaCampos() {
            var numero = document.getElementById('numero').value;
            if(numero == "" || null) {
                alert("O campo Número não pode ser vazio!");
                return false;
            }
        }
</script>
</head>
<body>
    <h3>Adicionar conta corrente:</h3>
    <form action="controller" method="post">
        Número: <input type="text" name="numero" id="numero" /><br>
        Agência: <input type="text" name="agencia" id="agencia" /><br>
        Descrição: <input type="text" name="desc" id="desc" /><br><br>
        <input type="submit" value="Enviar" onclick="return validaCampos()"/>
    </form>
</body>
```


- Obs.: Por precaução as validações devem ser feitas também do lado do servidor.

Referências

- CSS (W3C):
  https://www.w3.org/Style/CSS/

- Tutorial de CSS:
  http://www.w3schools.com/css/

- Tutorial de JavaScript:
  http://www.w3schools.com/js/default.asp

- Tutoriais do Java EE (site da Oracle):
  http://www.oracle.com/technetwork/java/javaee/documentation/tutorials-137605.html