DESENVOLVIMENTO DE SOFTWARE PARA WEB

JavaScript na web

Prof. Bruno Góis Mateus

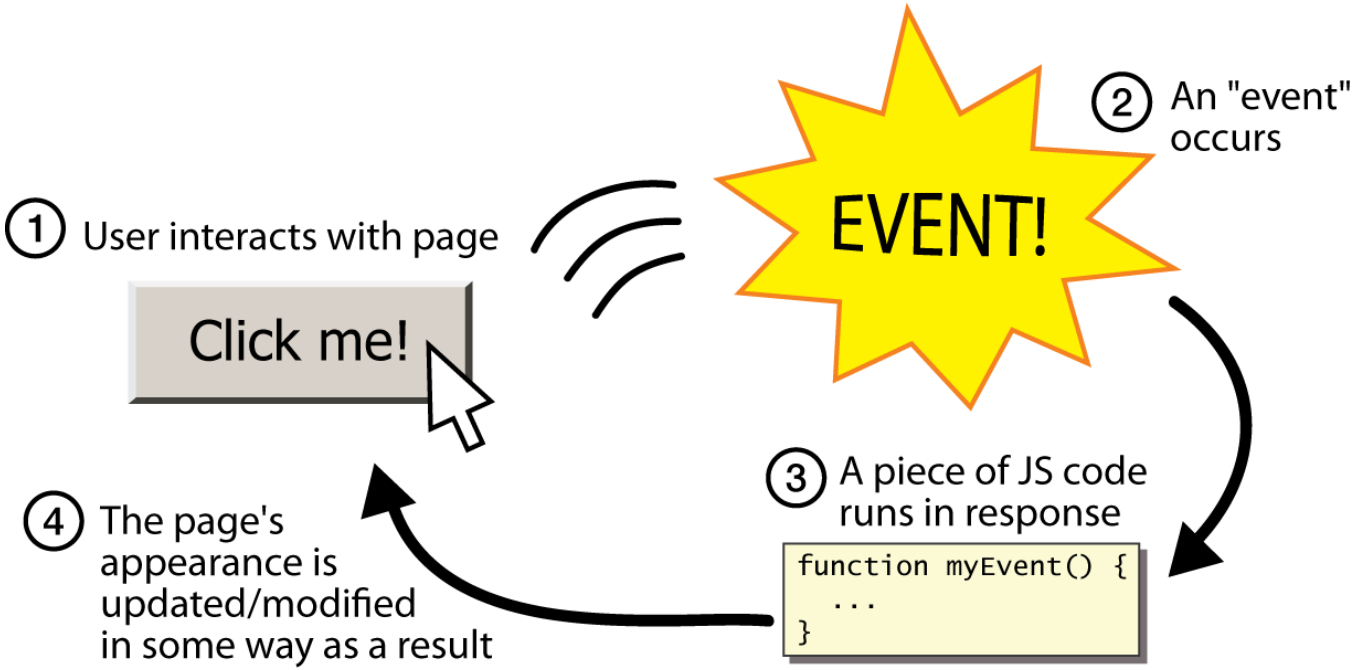
(brunomateus@ufc.br)

Índice

INTRODUÇÃO

Programação orientada a eventos

- Programas em JavaScript não possuem uma função principal
- No contexto web eles apenas respondem a ações do usuário que são chamados de eventos



Incluindo um javascript

- Sintaxe

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

- Exemplo

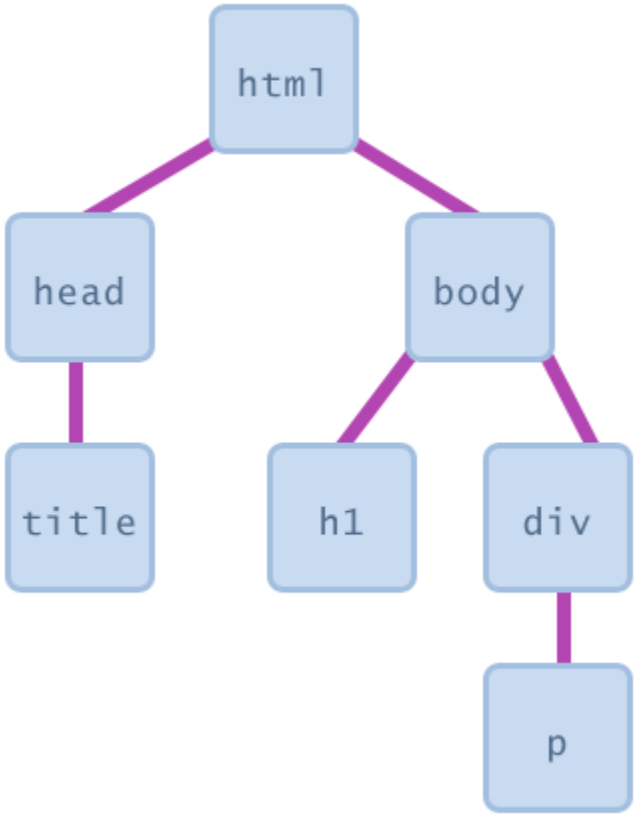
```
<script src="example.js" type="text/javascript"></script>
```

- A tag script deve ser colocada dentro da tag head
- O código do script é armazenado em um arquivo separado com extensão js
- O código JS pode ser colocado diretamente no arquivo HTML

DOM

Document Object Model

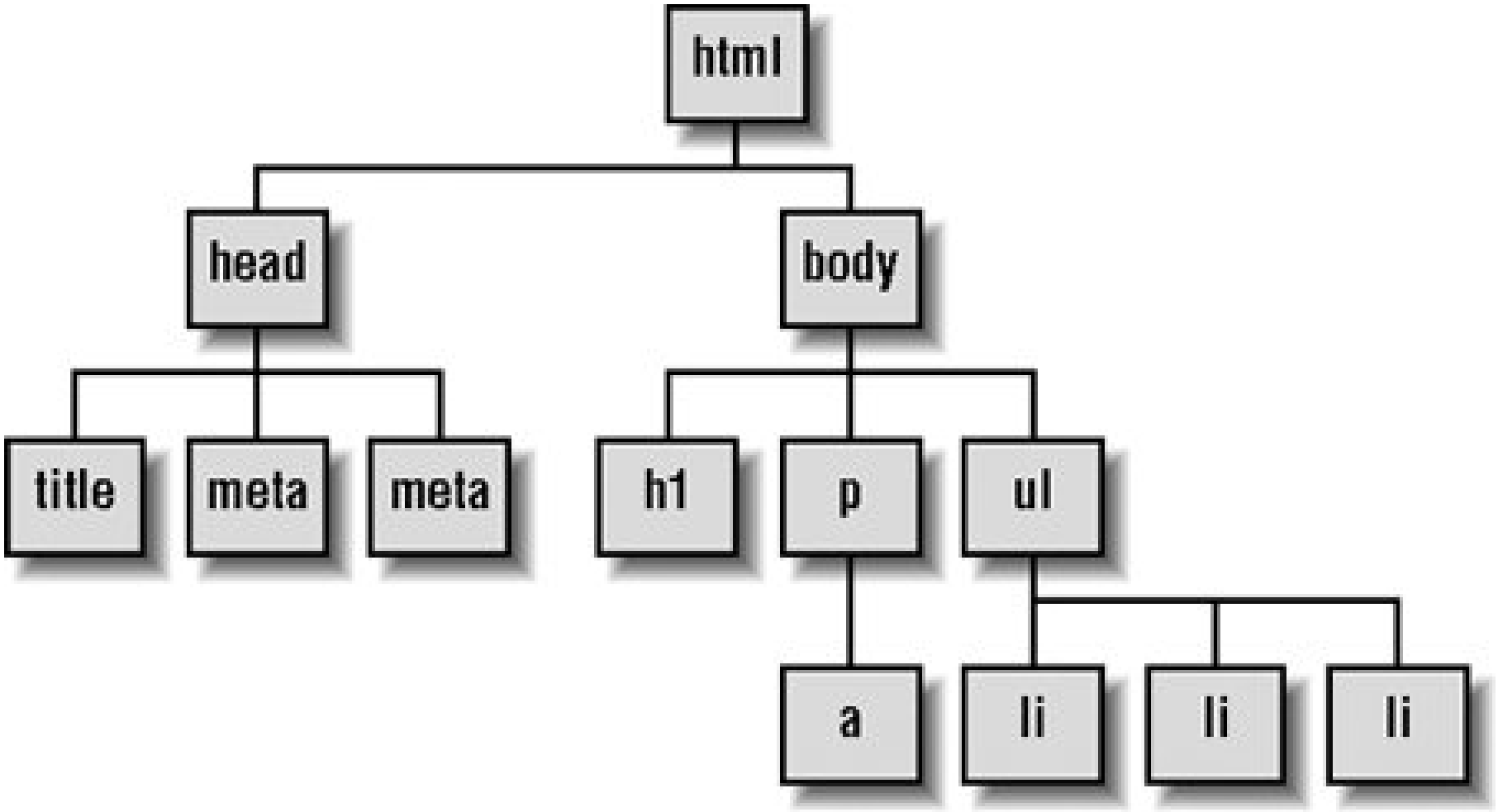
- A maioria dos códigos JS manipulam uma página HTML
  - Nós podemos verificar o estado de um elemento
    - Ex: Se um checkbox está checado
  - Nós podemos mudar o estado
    - Ex: Inserir um novo texto dentro de uma div
  - Nós podemos simplesmente mudar o estilo
    - Fazer um parágrafo ficar vermelho
- Todo elemento na página tem um objeto DOM correspondente



Document Object Model

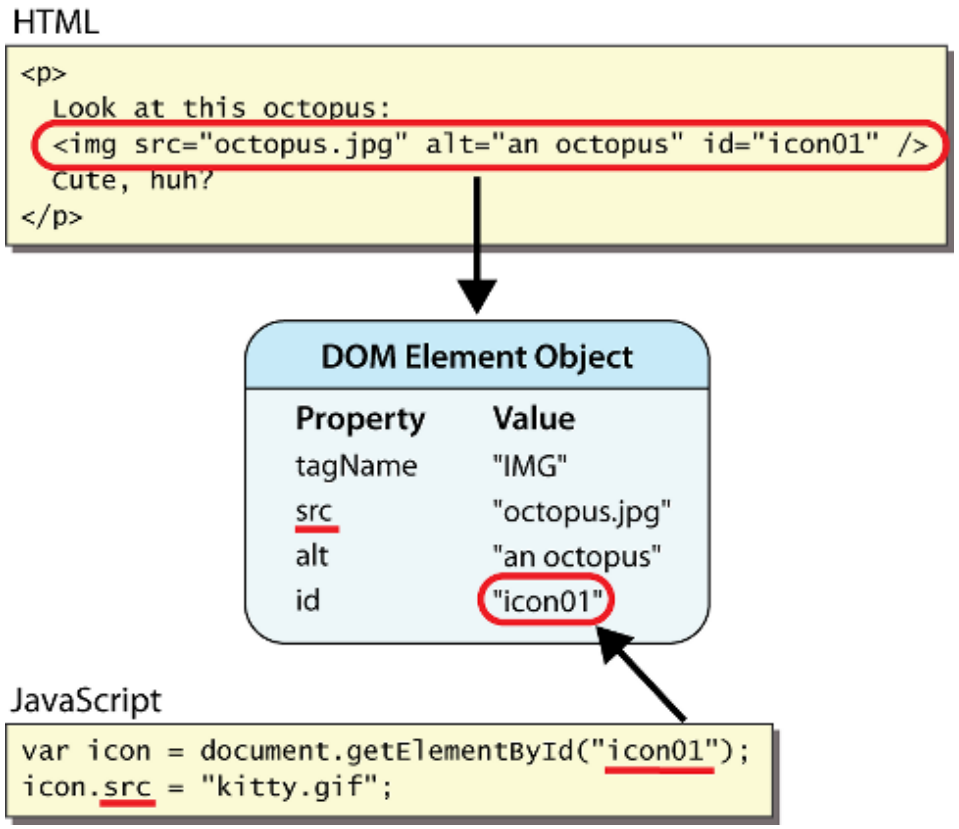
```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">
<head>
  <title>Sample XHTML</title>
  <meta http-equiv="content-type"
    content="text/html; charset=iso-8859-1" />
  <meta http-equiv="Content-Language" content="en-us" />
</head>
<body>
  <h1>This is a heading, level 1</h1>
  <p>This is a paragraph of text with a
    <a href="/path/to/another/page.html">link</a>.</p>
  <ul>
    <li>This is a list item</li>
    <li>This is another</li>
    <li>And another</li>
  </ul>
</body>
</html>
```

Árvore do documento



Elemento DOM

- Acessar e modificar os atributos de um objeto DOM é feito da seguinte forma:
  - objectName.attributeName



Propriedades de um objeto DOM

```
<div id="main" class="foo">
    <p>Olá, estou<em>muito</em> feliz de ver você!</p>
    <img id="icon" src="images/borat.jpg" alt="Borat" />
</div>
```

Propriedade	Descrição	Exemplo
tagName	Nome do elemento HTML	$(“main”).tagName = DIV
className	Class CSS do elemento HTML	$(“main”).className = foo
innerHTML	Conteúdo dentro do elemento	$(“main”).innerHTML = “<p> …
src	Caminho da imagem	$(“img”).src = images/borat.png

Acessando elementos

- document.getElementById retorna o objeto DOM elemento cujo id foi informado
  - É possível alterar o texto da maioria de elementos através da mudança da propriedade value

```
var name = document.getElementById("id");
function changeText() {
    var textbox = document.getElementById("output");
    textbox.value = "Hello, world!";
}
changeText();

<input id="output" type="text" value="replace me" />
```

Acessando elementos

```
function swapText() {
var span = document.getElementById("output");
var textBox = document.getElementById("textbox");
var temp = span.innerHTML;
span.innerHTML = textBox.value;
textBox.value = temp;
}
swapText();

<span id="output">Hello</span>
<input id="textbox" type="text" value="Goodbye" />
```

- Pode mudar o texto dentro da maioria dos elementos definindo a propriedade innerHTML

DOM

```
<div id="content">
  <h1>This is a heading</h1>
  <p>This is a paragraph.</p>
  <h2>This is another heading</h2>
  <p>This is another paragraph.</p>
  <p>Yet another paragraph.</p>
</div>

var the_div = document.getElementById( 'content' );
var h2s     = the_div.getElementsByTagName( 'h2' )
var the_h2  = h2s[0];
var everything = document.getElementsByTagName( '*' );
```

Value vs InnerHTML

- Existem duas maneiras de definir o texto de um elemento, dependendo do seu tipo:
  - innerHTML : texto conjunto entre a abertura e fechamento de tags (elementos regulares)
  - value : texto conjunto dentro de uma caixa de texto (controle de formulário)
    - Para elementos de formulário em geral, define o valor que será submetido para esse elemento
    - Em textarea (apesar de possuírem tags de abrir e fechar), apenas value funciona como o esperado em todos os navegadores

Value vs InnerHTML

```
<p id="welcome">Go away, you're not welcome here.</p>
var paragraph = document.getElementById("welcome");
// change text inside an opening and closing tag
paragraph.innerHTML = "Welcome to our site!";

<input type="text" id="username" value="stepp" />
var textbox = document.getElementById("username");
// change text inside text box
textbox.value = "moocy";
```

Má prática: Uso abusivo de innerHTML

```
// bad style!
var paragraph = document.getElementById("welcome");
paragraph.innerHTML = "<p>text and <a href="page.html">link</a>"
```

- innerHTML Pode ser utilizado para injetar o conteúdo HTML arbitrário na página
- No entanto, tal prática é muito é propensa a erros
  - Torna o código ilegível
- Procure injetar apenas texto simples

Get/Set

```
var href = document.getElementById( 'easy' ).getAttribute( 'href' );
var link = document.getElementById( 'easy' );
link.setAttribute( 'href', 'http://www.easy-designs.net/index.php' );
link.setAttribute( 'title', 'The Easy Designs, LLC homepage' );
```

Métodos

```
var new_div = document.createElement('div');

var text = document.createTextNode('This  is a new div');

new_div.appendChild(text);
new_div.appendChild(  document.createTextNode('This  is a new div'));

var body = document.getElementsByTagName('body')[0];
body.insertBefore(new_div,  body.firstChild);
body.replaceChild(new_div,  body.firstChild);
div.removeChild(div.firstChild);
```

Ajustando estilos

```
window.onload = function() {
    document.getElementById("clickme").onclick = changeColor;
};
function changeColor() {
    var clickMe = document.getElementById("clickme");
    clickme.style.color = "red";
}
<button id="clickme">Color Me</button>
```

Propriedade	Descrição
style	Permite você define qualquer propriedade CSS de um elemento

Erros comuns ao estilizar usando DOM

```
var clickMe = document.getElementById("clickme");
clickMe.color = "red"; // incorreto
clickMe.style.color = "red";
```

- Muitos desenvolvedores esquece de escrever o .style ao definir estilos

```
clickMe.style.font-size = "14pt"; // Incorreto
clickMe.style.fontSize = "14pt";
```

- Propriedades não utilizam o hífen como separador

```
clickMe.style.width = 200; // Incorreto
clickMe.style.width = "200px";
```

- Escreve exatamente o mesmo valor que você escreveria no CSS, apenas entre aspas

Erros comuns ao estilizar usando DOM

```
document.getElementById("main").style.top =
    document.getElementById("main").style.top + 100 + "px";
```

- O exemplo acima calcula, por exemplo 200px + 100 + px
  - Resultado: 200px100px
- Uma versão corrigida:

```
document.getElementById("main").style.top =
    parseInt(document.getElementById("main").style.top) + 100 +
"px";
```

Boas práticas ao aplicar estilo

```
function okayClick() {
this.style.color = "red"; // incorreto
this.className = "highlighted";
}
.highlighted { color: red; }
```

- Um código JavaScript bem escrito contém o mínimo de código CSS possível
  - Use JavaScript para atribuir classes e Id de elementos
  - Defina os estilos dessas classes e ids no arquivos CSS

OBJETOS GLOBAIS DOM

Objetos Globais DOM

Objeto	Descrição
document	Página HTML atual e seu conteúdo
history	Lista de páginas que o usuário visitou
location	URL da página atual
navigator	Informações sobre o navegador
screen	Informações sobre a área da tela utilizada pelo navegador
window	A janela do navegador

O objeto Window

- A janela do browser, o objeto de nível superior na hierarquia DOM
  - Tecnicamente, todo o código global e variáveis são parte do objeto window
- Propriedades:
  - document
  - history
  - location
  - name
- Métodos:
  - alert
  - confirm
  - prompt (popup)
  - setInterval
  - setTimeout
  - clearInterval
  - clearTimeout (temporizadores)
  - open
  - close (aparecendo novas janelas do navegador)
  - blur , focus , moveBy , moveTo , print , resizeBy , resizeTo , scrollBy ,
  - scrollTo

Janelas Popup usando Window

```
window.open ("http://foo.com/bar.html",
"My Window Foo", "width = 900, altura = 600, scrollbars = 1");
```

- window.open abre uma nova janela do navegador
  - Este método é a causa de todos os popups terríveis na web!
- Algum software bloqueador de pop-up irá impedir a execução desse método

O objeto document

- A página da web atual e os elementos dentro dela
- Propriedades:
  - anchors
  - body
  - cookie
  - domain
  - forms
  - imagens
  - links
  - referrer
  - title
  - URL
- Métodos:
  - getElementById
  - getElementsByName
  - getElementsByTagName
  - close
  - open
  - write
  - writeln

O objeto location

- A URL da página web atual
- Propriedades:
  - host
  - hostname
  - href
  - pathname
  - port
  - protocol
  - search
- Métodos:
  - assing
  - reload
  - replace

O objeto navigator

- Informações sobre o aplicativo do navegador web
- Propriedades:
  - appName
  - appVersion
  - browserLanguage
  - cookieEnabled
  - plataform
  - userAgent

O objeto screen

- Informações sobre a tela do cliente de exibição
- Propriedades:
  - availHeight
  - availWidth
  - colorDepth
  - Height
  - pixelDepth
  - width

O objeto History

- A lista de sites no navegador visitou nesta janela
- Propriedades:
  - lenght
- Métodos:
  - back
  - Forward
  - go
- Por motivos de segurança, às vezes o navegador não vai deixar de scripts acessar o histórico

Árvore de Objetos

Métodos

```
var ul = document.createElement(  'ul' );
var li = document.createElement(  'li' );
li.className  = 'check';
for( var i=0; i < 5; i++ ){
  var new_li = li.cloneNode(  true );
  new_li.appendChild(  document.createTextNode('list  item ' + (i + 1)));
  ul.appendChild(new_li);
}
```

TRATANDO EVENTOS

Eventos

- Interação do JavaScript com HTML é feita através de eventos que ocorrem quando o usuário ou o navegador manipula uma página
  - Quando a página é carregada, que é um evento
  - Quando o usuário clica em um botão
  - Pressionar qualquer tecla
  - A janela fechar

Eventos

- Os desenvolvedores podem usar esses eventos para executar respostas especificas para cada evento
  - Mensagens a serem exibidas aos usuários
  - Os dados a serem validados, e praticamente qualquer outro tipo de resposta que se possa imaginar para ocorrer.
- Os eventos são uma parte do Document Object Model (DOM) e todos os elementos HTML têm um certo conjunto de eventos que podem desencadear código JavaScript.

Tratando eventos

- Primeiro passo é registar um *hanlder*
- Existem 4 maneiras possíveis
  - Inline
  - Tradicional
  - W3C
  - Microsoft

Tratando eventos - Inline

- Maneira mais antiga
- Os tratadores de eventos são atribuídos aos atributos HTML

```
<a href="somewhere.html"
   onClick="alert(‘Você clicou em algum lugar!')">
<a href="somewhere.html"
   onClick=“minha_funcao()">
```

- Deve ser evitado, o ideal é manter o código javascript totalmente separado do código HTML

Tratando eventos - Tradicional

- Nova maneira de tratar eventos
  - Suportada incialmente pelo Netscape 3 e IE 4

```
element.onevent = function

<button id="ok">OK</button>
var okButton = document.getElementById("ok");
okButton.onclick = minha_funcao;
```

- É uma boa prática anexar os tratadores de eventos aos
- objetos dos elementos DOM em seu código JavaScript
  - Perceba que você não coloca parênteses após o nome da função

Tratando eventos – W3C

- Permite vários *handlers* para um mesmo event

```
element.addEventListener('click',startDragDrop,false);
element.addEventListener('click',spyOnUser,false);
```

- Ambos os *handlers* serão acionados
  - A ordem não é garantida
- Remoção de um handler

```
element.removeEventListener('click',startDragDrop,false);
```

Tratando evento

- This
  - Palavra chave que determina o dono da função
  - Muito útil para o tratamento de eventos

```
element.onclick = doSomething;
another_element.onclick = doSomething;

function doSomething() {
    this.style.backgroundColor = '#cc0000';
}
```

Eventos Comuns

Evento	Descrição
onchange	Elemento HTML é modificado
onclick	Usuário clicou em um elemento HTML
onmouseover	Quando o usuário colocar o ponteiro do mouse sobre o elemento HTML
onmouseout	Quando o usuário retira o ponteiro do mouse de “cima” elemento HTML
onkeydown	Quando usuário pressiona uma tecla do teclado
onload	Quando o navegador terminar de carregar a página
onsubmit	Quando o usuário clica no botão de envio de um formulário

EVENTOS DE TEMPO

Eventos de tempo

- O Javascript prover dois mecanismos de tratar eventos relacionados ao tempo

Método	Descrição
setTimeout	Faz com que uma função seja chamada após o tempo de atraso definido
setInterval	Faz com que uma função seja chamada repetidas vezes a período de tempo. O período é passado como parâmetro
clearTimout	Remove o cronômetro especificado

- setTimeout e setInterval retornam um ID que representa o cronômetro
- O ID é utilizado pela função clearTimout

Exemplo - setTimout

```
<button onclick="delayMsg();">Click me!</button>
<span id="output"></span>

function delayMsg() {
    setTimeout(booyah, 5000);
    document.getElementById("output").innerHTML = "Wait for
it...";
}

function booyah() {
    document.getElementById("output").innerHTML = "BOOYAH!";
}
```

Exemplo - setInterval

```
var timer = null;
 // stores ID of interval timer
function delayMsg2() {
    if (timer == null) {
        timer = setInterval(rudy, 1000);
    } else {
        clearInterval(timer);
        timer = null;
    }
}

function rudy() {
     // called each time the timer goes off
    document.getElementById("output").innerHTML += " Rudy!";
}
```

Passando parâmetros para os cronômetros

- Quaisquer parâmetros após o atraso são passado para a função de temporizador
  - Não funciona no IE6, deve criar uma função de intermediário para passar os parâmetros

```
function delayedMultiply() {
    // 6 and 7 are passed to multiply when timer goes off
    setTimeout(multiply, 2000, 6, 7);
}

function multiply(a, b) {
    alert(a * b);
}
```

Erro comum ao usar cronômetros

```
setTimeout(booyah(), 2000); //errado
setTimeout(booyah, 2000);
setTimeout(multiply(num1 * num2), 2000); //errado
setTimeout(multiply, 2000, num1, num2):
```

BOAS PRÁTICAS

JavaScript Discreto

- HTML deve conter o mínimo JavaScript
  - Usa o DOM para anexar e executar todas as funçõoes JavaScript
- Permite a separação do site em 3 camadas principais:
  - Conteúdo (HTML) - O que é o site?
  - Apresentação (CSS) - Como ele deve aparecer?
  - Comportamento (JavaScript) - como ele responder a
  - interação do usu ario?

Tratando eventos

```
<button onclick="okayClick();">OK</button>
function okayClick() {
    alert("booyah");
}
```

- HTML misturado com JavaScript
- Melhor

```
<button id="ok">OK</button>
var okButton = document.getElementById("ok");
okButton.onclick = minha_funcao;
```

Código global

```
<html>
<head>
<script src="myfile.js" type="text/javascript"></script>
</head>
<body>
<div><button id="ok">OK</button></div>

// global code - myfile.js
document.getElementById("ok").onclick = okayClick;
```

- O código JS é executado no momento que o navegador carrega o script
  - Todas as variáveis e funções são declaradas imediatamente
  - Ao ser carregado, outra partes da página podem ainda estar indisponíveis(não carregadas)

Código global

```
<html>
<head>
<script src="myfile.js" type="text/javascript"></script>
</head>
<body>
<div><button id="ok">OK</button></div>

// global code – myfile.js
window.onload = function(){
document.getElementById("ok").onclick = okayClick;}
```

O que vem por aí

- Fundamentos de Sevlet