



# **PROGRAMAÇÃO PARA INTERNET RICA**

## **MANIPULANDO XHTML E CSS COM JAVASCRIPT**

Prof. Dr. Daniel Caetano

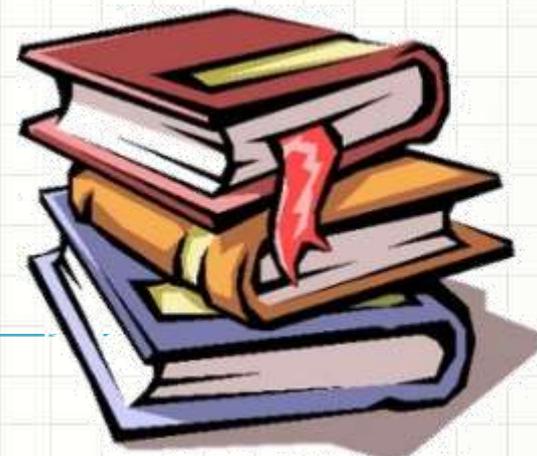
2012 - 2

# Objetivos

- Apresentar a manipulação direta de elementos do XHTML e CSS com o uso do JavaScript
- Apresentar e eliminar elementos (PopUp, por exemplo)
- Mudar funcionalidade de elementos
- **TRABALHO A ONLINE!**



# Material de Estudo



## Material

## Acesso ao Material

Notas de Aula

<http://www.caetano.eng.br/>  
(Aula 7)

Apresentação

<http://www.caetano.eng.br/>  
(Aula 7)

Material Didático

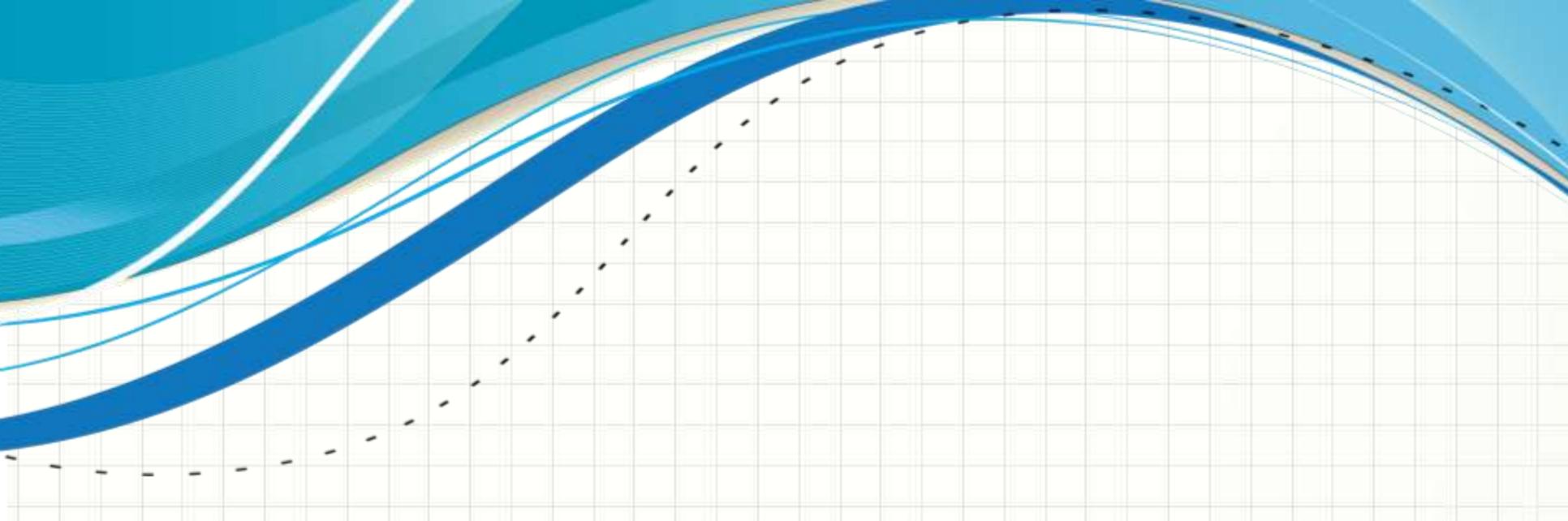
Aprenda a Criar Páginas Web c/ HTML, páginas 609 a 648

Google

+“JavaScript” +tutorial +DHTML

Web Sites

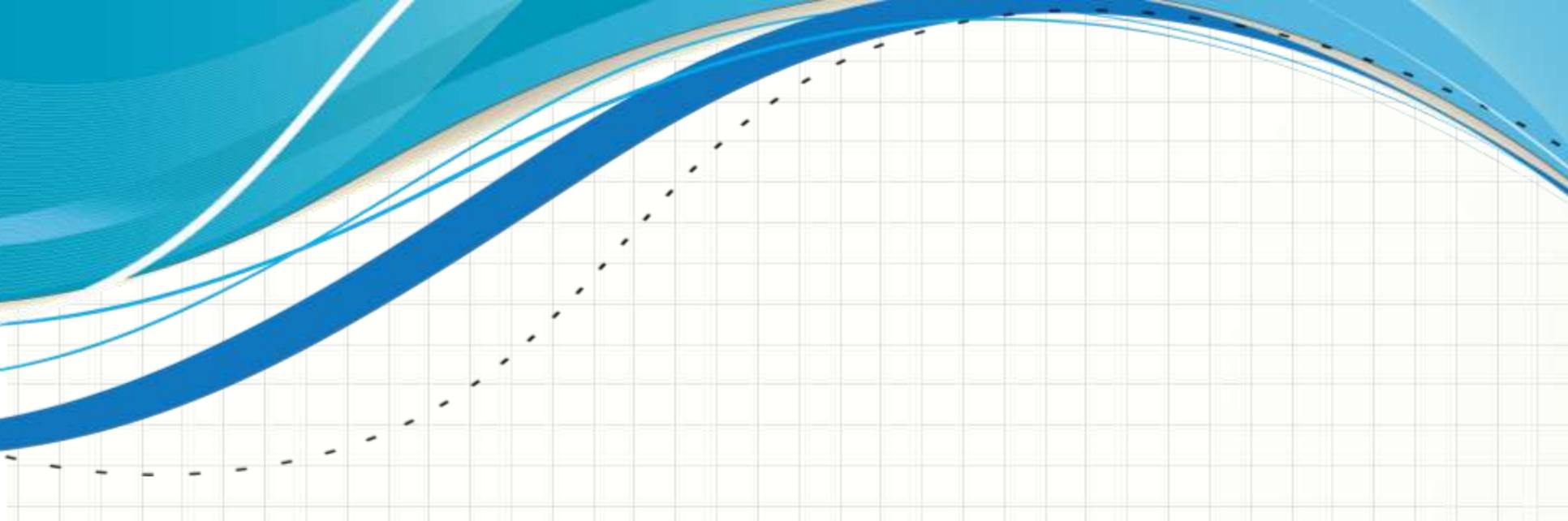
<http://www.w3.org/>



# **CODIFICAÇÃO “CLIENT-SIDE”**

# Codificação Client-Side

- Por que codificar no lado do cliente?
- Respostas rápidas
- Sensação de interação em tempo real
- Diminuir consumo de banda
- Diminuir processamento no servidor
  
- Manipular dados do XHTML
  - `document.getElementById("id")`



# **COMO “SUMIR” COM ELEMENTOS DA PÁGINA**

# Sumindo com Elementos

- Sumir = não desenhar
- Prático:
  - Deixar tudo pronto
  - Mostrar só quando necessário
- Duas formas de fazer o “sumiço”
  - A. Elemento continua ocupando seu espaço
  - B. Elemento some completamente

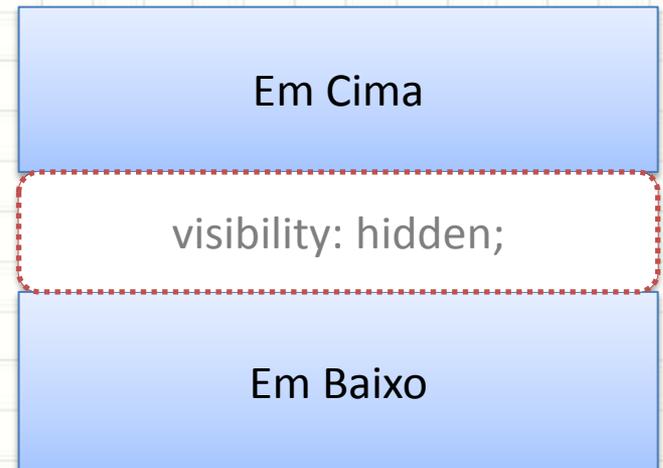
# Sumindo com Elementos

- Elemento ocupa espaço mas não é desenhado
- Propriedade: **visibility**
- **visibility: visible;**
  - Mostra elemento



# Sumindo com Elementos

- Elemento ocupa espaço mas não é desenhado
- Propriedade: **visibility**
- **visibility: visible;**
  - Mostra elemento
- **visibility: hidden;**
  - Esconde elemento



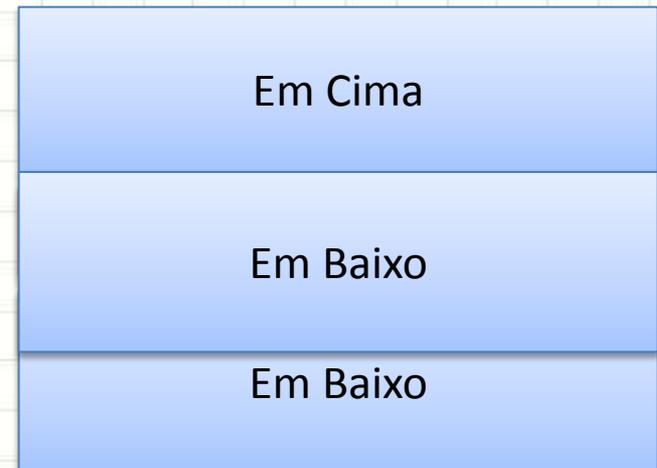
# Sumindo com Elementos

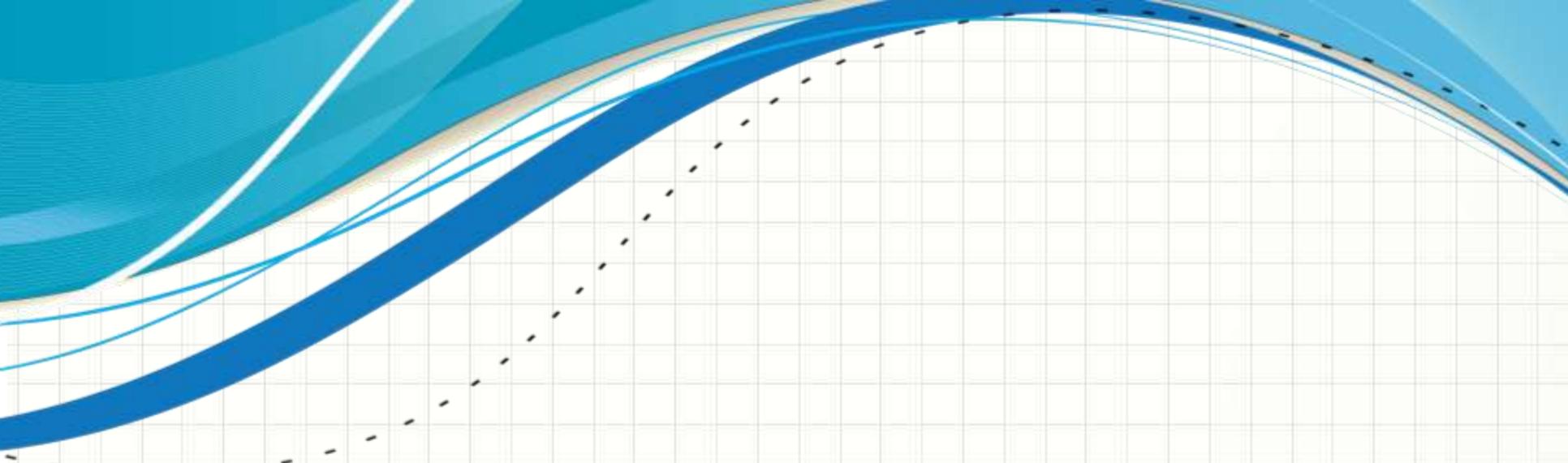
- Elemento nem ocupa espaço nem é desenhado
- Propriedade: **display**
- **display: block;**
  - Mostra elemento



# Sumindo com Elementos

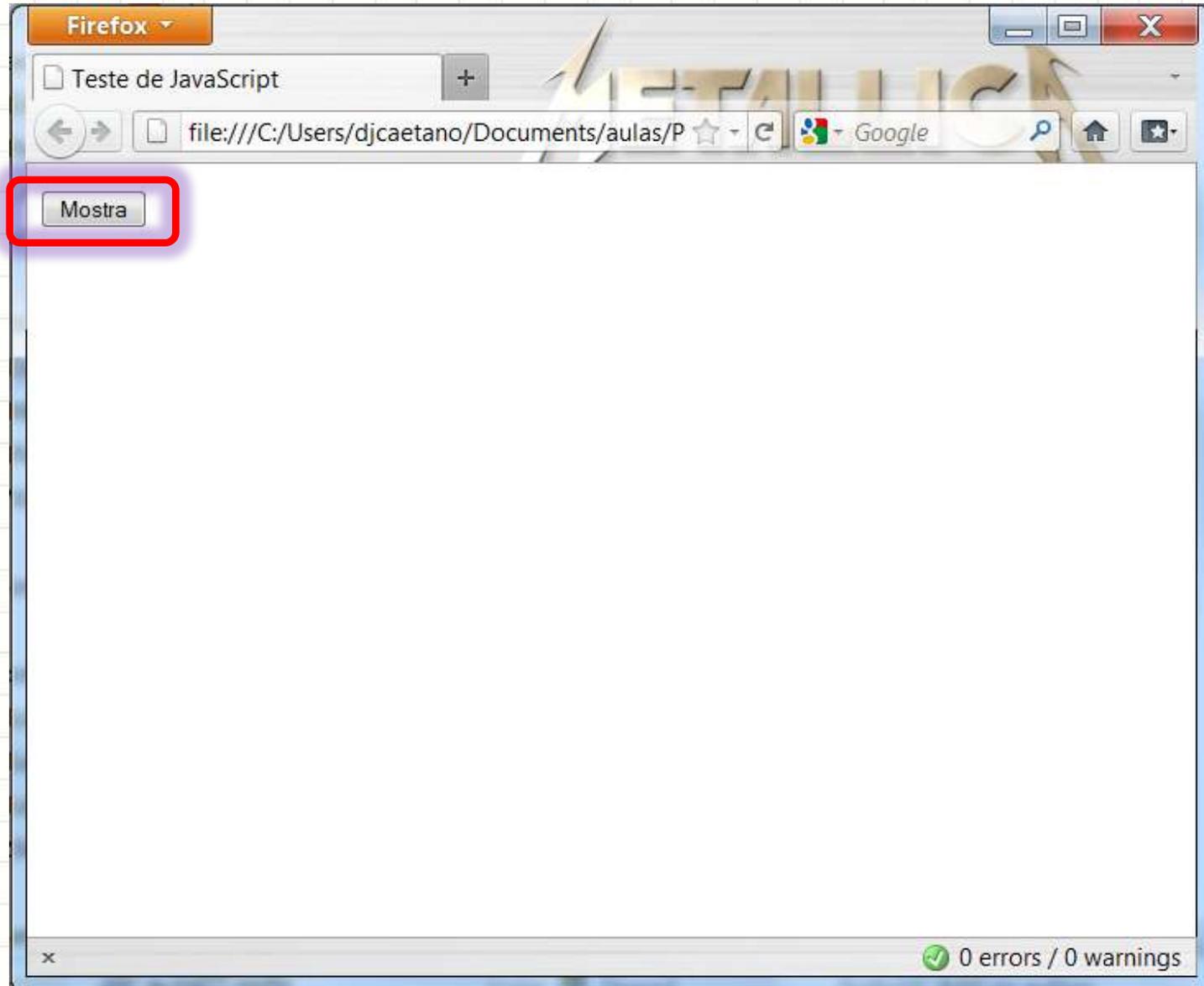
- Elemento nem ocupa espaço nem é desenhado
- Propriedade: **display**
- **display: block;**
  - Mostra elemento
- **display: none;**
  - Esconde elemento



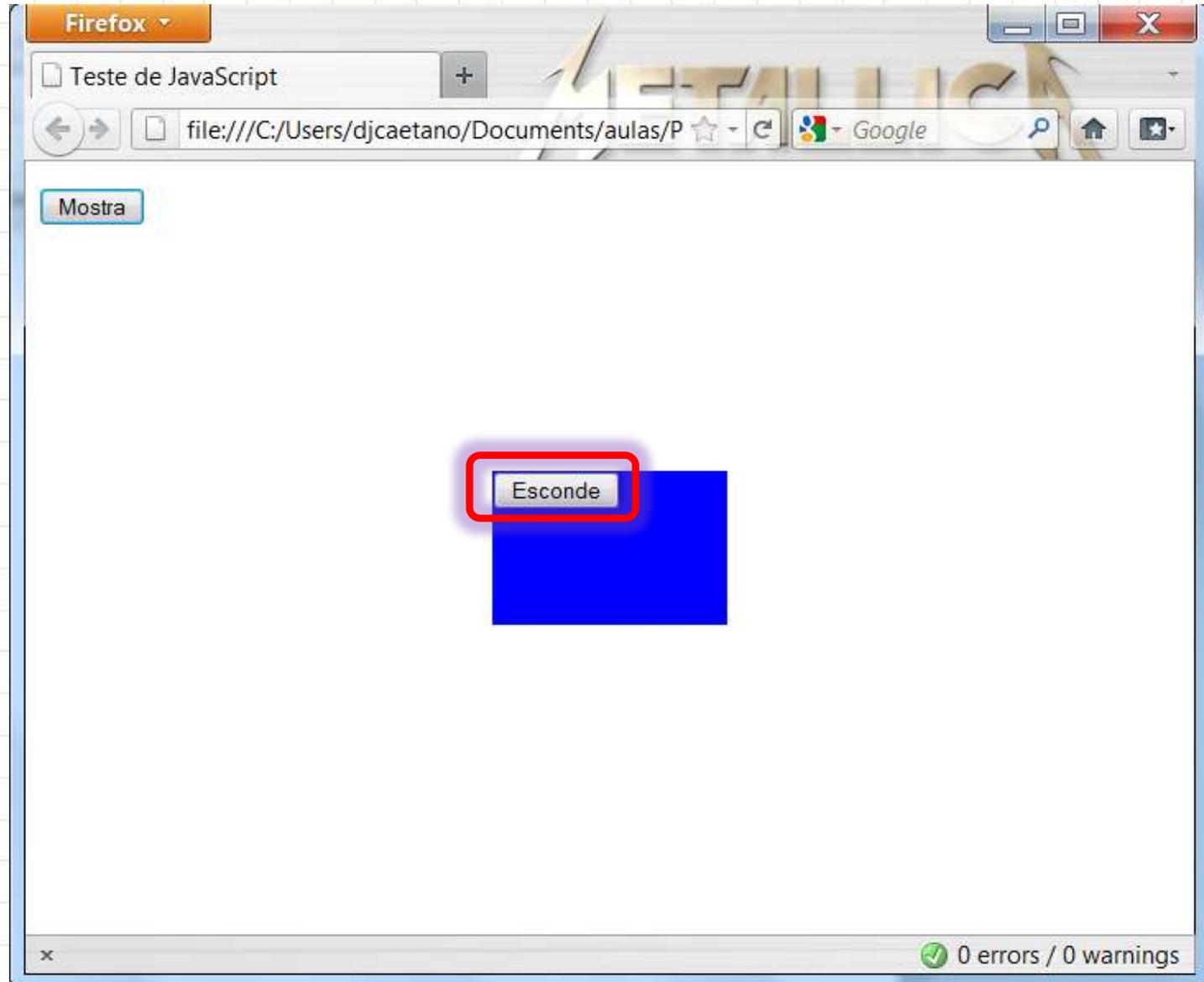


# TUTORIAL POPUP

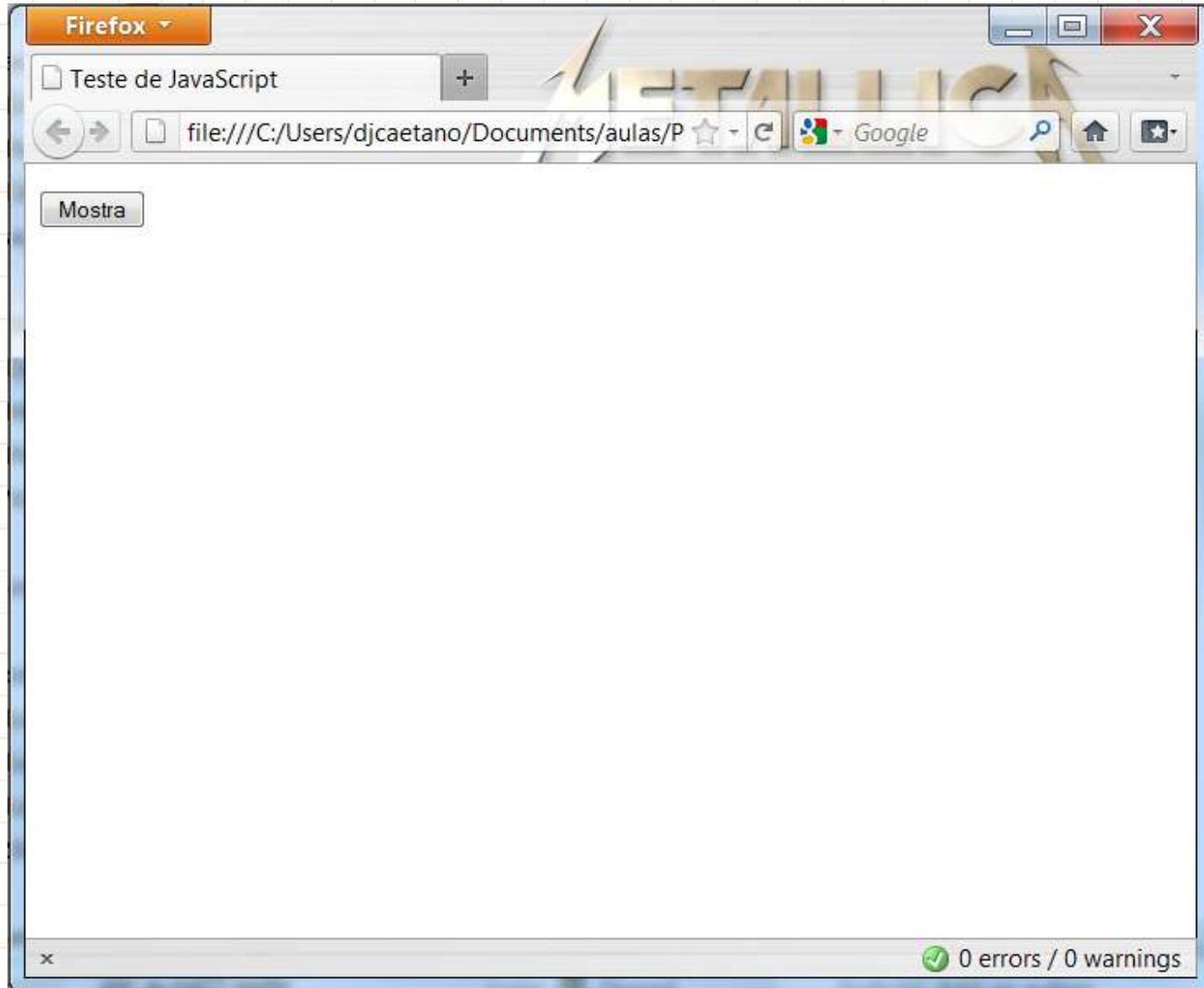
# Um PopUp em JavaScript+CSS

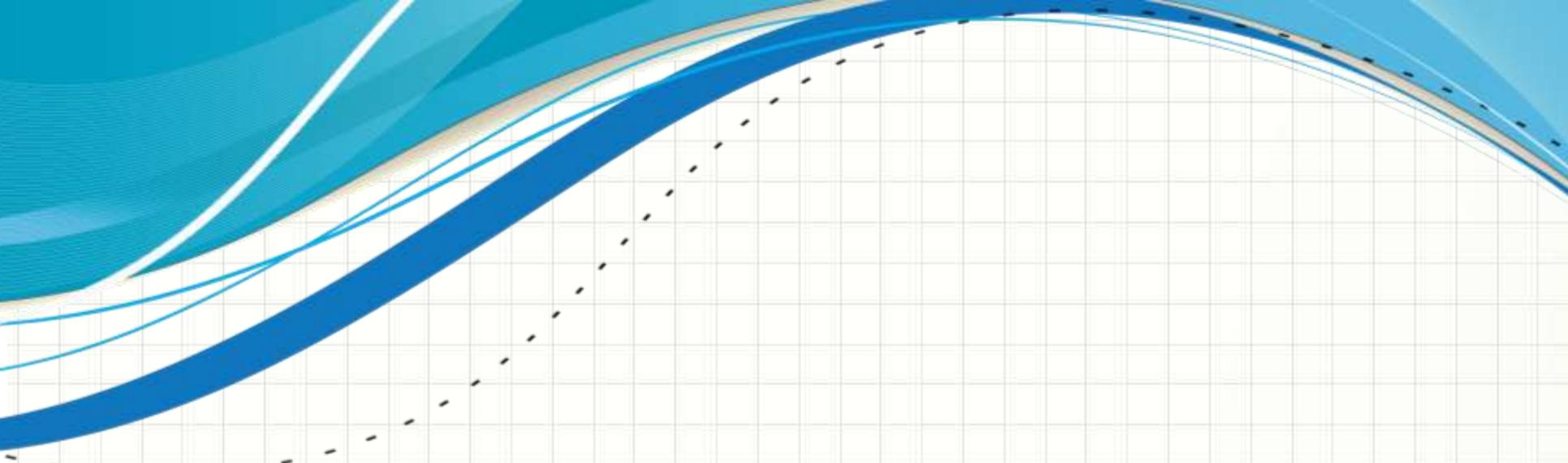


# Um PopUp em JavaScript+CSS



# Um PopUp em JavaScript+CSS

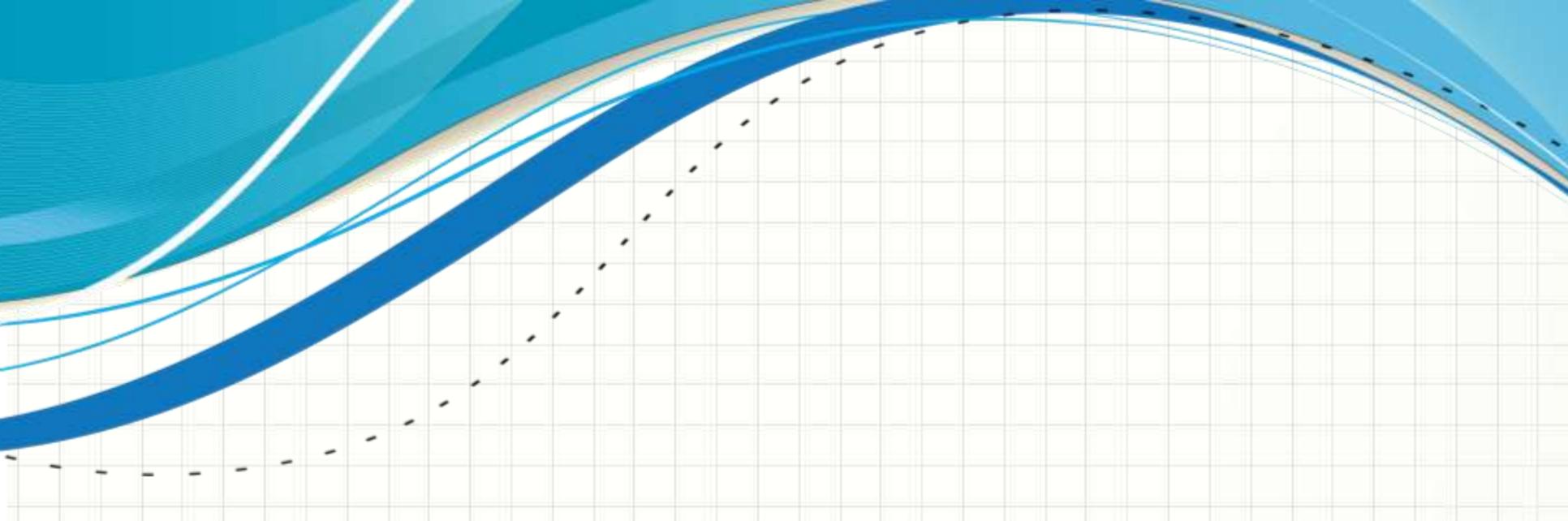




**ASSOCIANDO  
FUNCIONALIDADES A  
OUTROS ELEMENTOS**

# Funcionalidades em Elementos

- Quando associamos funções direto no HTML
  - Só podemos associar ações a links e botões...
- No JavaScript, entretanto, não há limitação!
- Podemos associar ações a um **div**, por exemplo!
  - A. Fechar o PopUp quando clicarmos duas vezes no mesmo;
  - B. Fechar o PopUp quando passarmos o mouse por cima do mesmo.



# **TROCANDO A FUNCIONALIDADE DE ELEMENTOS DA PÁGINA**

# Trocar a Funcionalidade

- Na função `init()`...
  - Associamos uma função ao botão “mostra”
- É possível mudar a função associada!
- Basta indicar um novo nome

```
botao.onclick = funcao1;
```

```
(...)
```

```
botao.onclick = funcao2;
```

# Trocar a Funcionalidade

```
botao.onclick = funcao1;
```



botao.onclick()

A red arrow points from the text 'botao.onclick()' in a purple rounded rectangle down to the 'function funcao1()' block in a blue rounded rectangle.

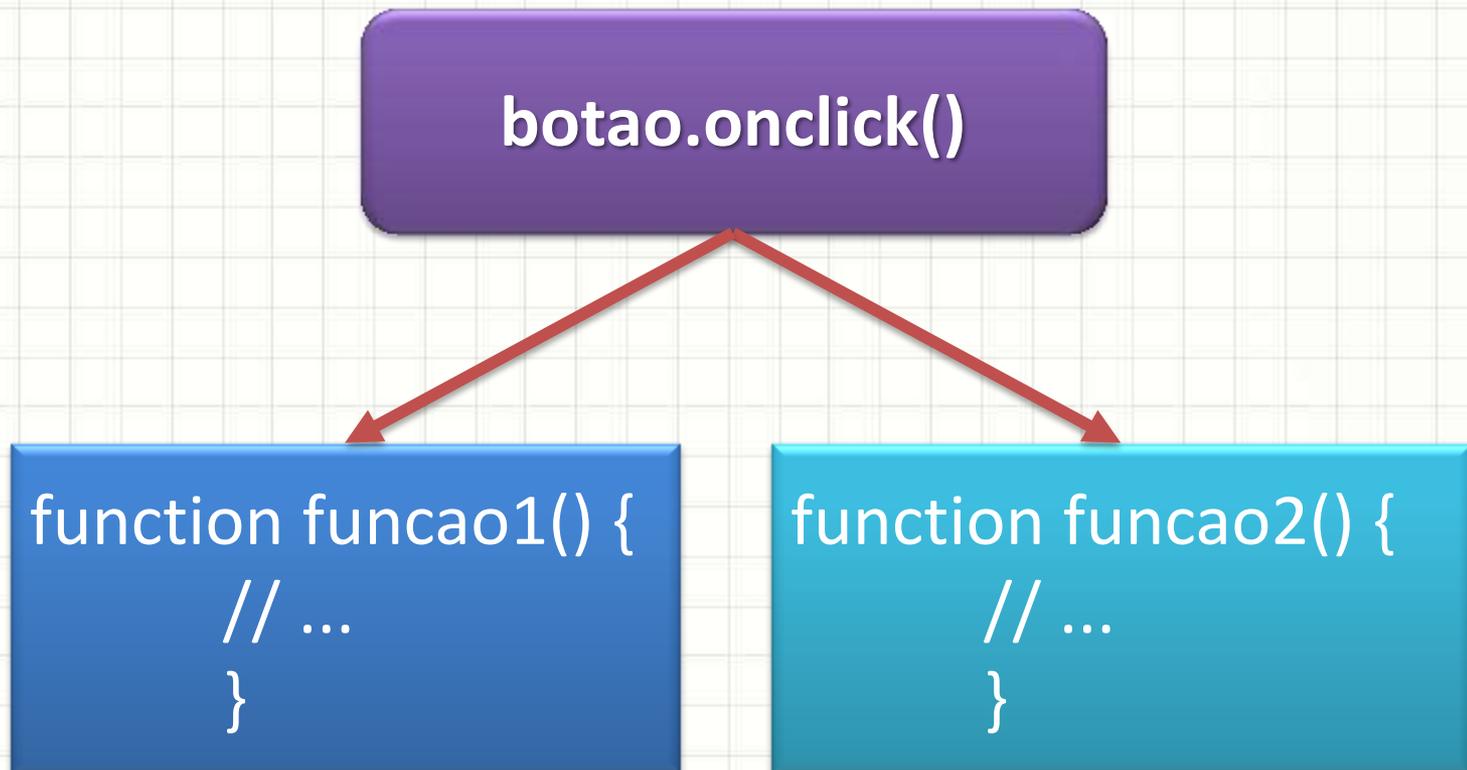
```
function funcao1() {  
  // ...  
}
```

```
function funcao2() {  
  // ...  
}
```

# Trocar a Funcionalidade

```
botao.onclick = funcao1;
```

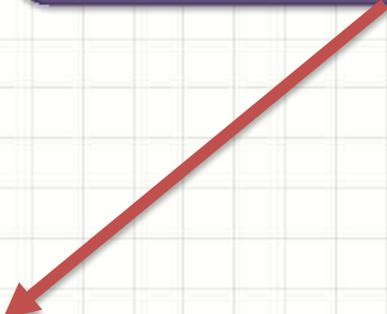
```
botao.onclick = funcao2;
```



# Trocar a Funcionalidade

- Exemplo:

**botao.onclick()**  
chamar: funcao1



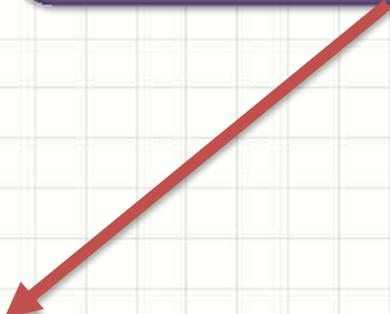
```
function funcao1() {  
    mostraPopUp();  
    botao.onclick = funcao2;  
}
```

```
function funcao2() {  
    escondePopUp();  
    botao.onclick = funcao1;  
}
```

# Trocar a Funcionalidade

- Exemplo:

**botao.onclick()**  
chamar: funcao1



```
function funcao1() {  
    mostraPopUp();  
    botao.onclick = funcao2;  
}
```

```
function funcao2() {  
    escondePopUp();  
    botao.onclick = funcao1;  
}
```

# Trocar a Funcionalidade

- Exemplo:

**botao.onclick()**  
chamar: funcao2



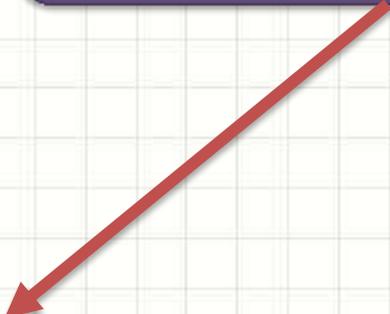
```
function funcao1() {  
    mostraPopUp();  
    botao.onclick = funcao2;  
}
```

```
function funcao2() {  
    escondePopUp();  
    botao.onclick = funcao1;  
}
```

# Trocar a Funcionalidade

- Exemplo:

**botao.onclick()**  
chamar: funcao1



```
function funcao1() {  
    mostraPopUp();  
    botao.onclick = funcao2;  
}
```

```
function funcao2() {  
    escondePopUp();  
    botao.onclick = funcao1;  
}
```

# Trocar a Funcionalidade

- Exemplo:

**botao.onclick()**

chamar: funcao1

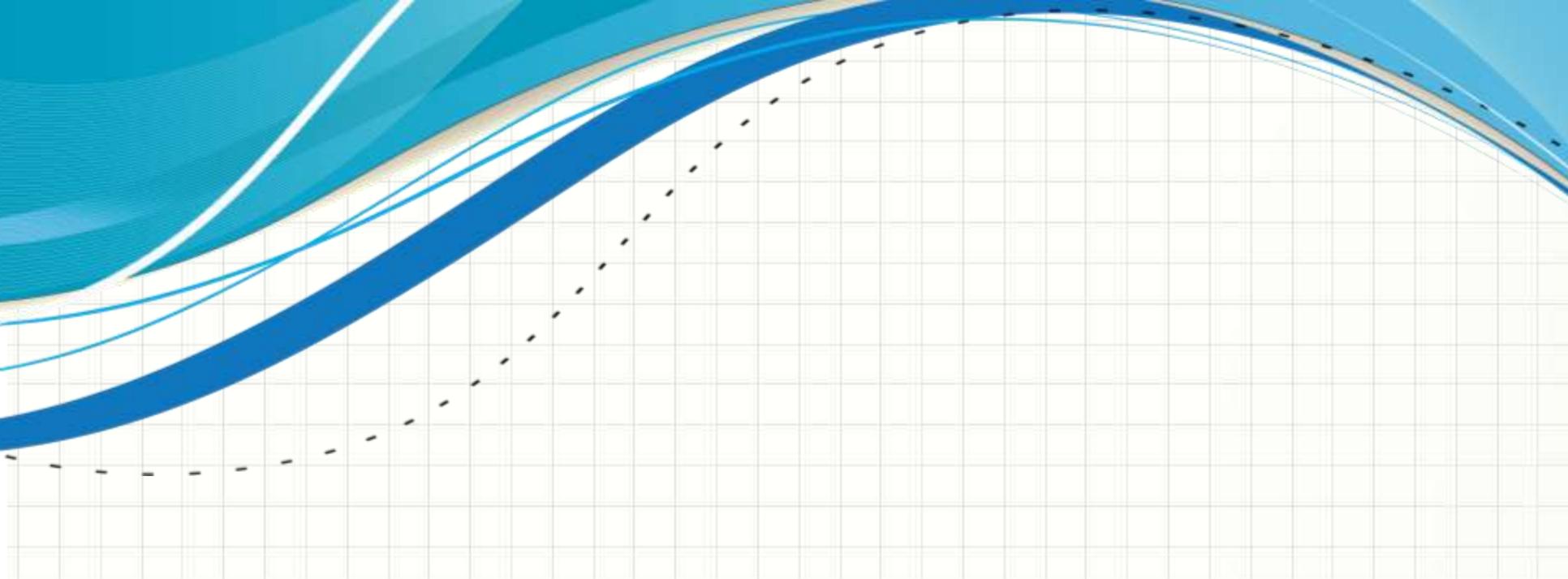
**E assim por diante...**

```
function funcao1() {  
    mostraPopUp();  
    botao.onclick = funcao2;  
}
```

```
function funcao2() {  
    escondePopUp();  
    botao.onclick = funcao1;  
}
```

# Trocar a Funcionalidade

- Vamos testar?
  - A. Modifique o botão que abre o PopUp para que, após abri-lo, sirva também para fechá-lo;
  - B. Incremente o código para que, quando a função do botão mudar, mude também seu texto.



# FUNÇÕES COM PARÂMETROS

# Parâmetros em Funções

- As funções JS pode receber parâmetros
- Basta indicar na declaração

```
function mudaCor(novaCor) {  
    document.body.style.color = novaCor;  
}
```

# Parâmetros em Funções

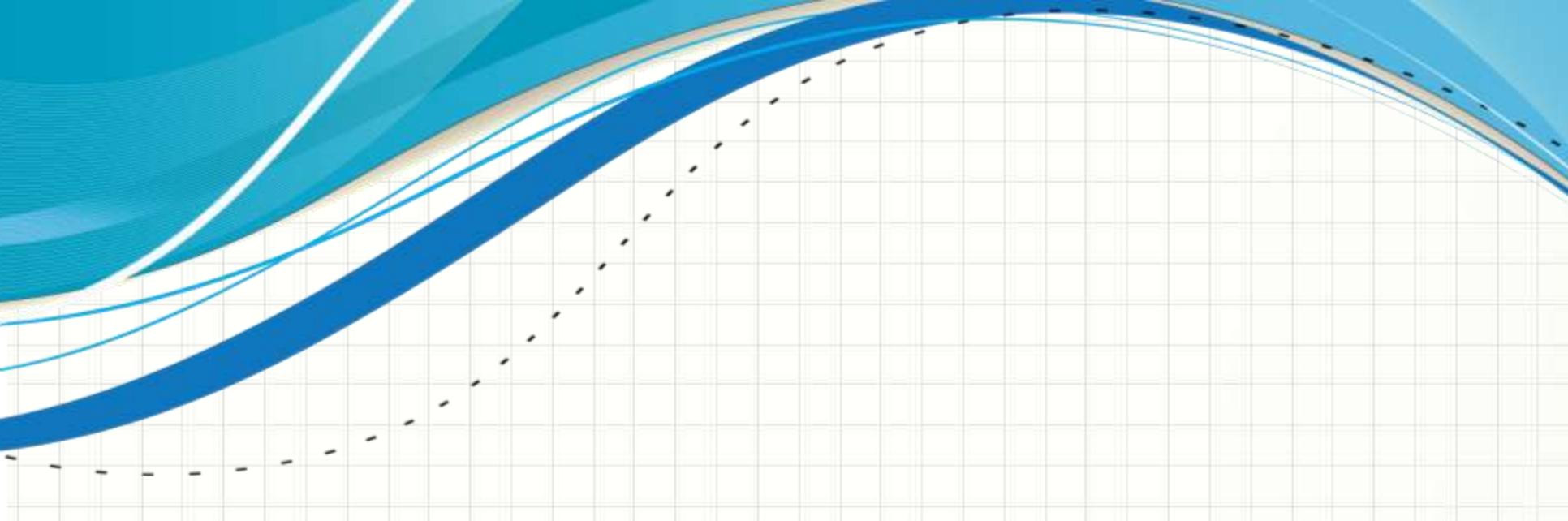
- As funções JS pode receber parâmetros
- Basta indicar na declaração

```
function mudaCor(novaCor) {  
    document.body.style.color = novaCor;  
}
```

- Chamando...

```
mudaCor("green");
```

- Qual a nova cor do texto?



**VÁRIOS EVENTOS, UMA  
ÚNICA FUNÇÃO**

# Função para Múltiplos Eventos

- Uma mesma função JS pode ser associada a vários eventos e elementos...

```
function mudaCor() {  
    document.body.style.color="green";  
}
```

```
button1.onclick = mudaCor;
```

```
div1.onclick = mudaCor;
```

```
td1.onclick = mudaCor;
```

# Função para Múltiplos Eventos

- Mas essa função muda sempre o texto global!

```
function mudaCor() {  
    document.body.style.color = "green";  
}
```

- E se quisermos mudar apenas a cor do texto do elemento clicado?

# Função para Múltiplos Eventos

- Exemplo:
  - Clicou em um botão, muda cor texto deste botão
  - Clicou em um div, muda cor deste div
- Dois jeitos: o “força bruta” e o “sei o que faço”
- Força Bruta?
  - Uma função diferente para cada elemento
- E o outro método?

# Função para Múltiplos Eventos

- Jeito “inteligente”:

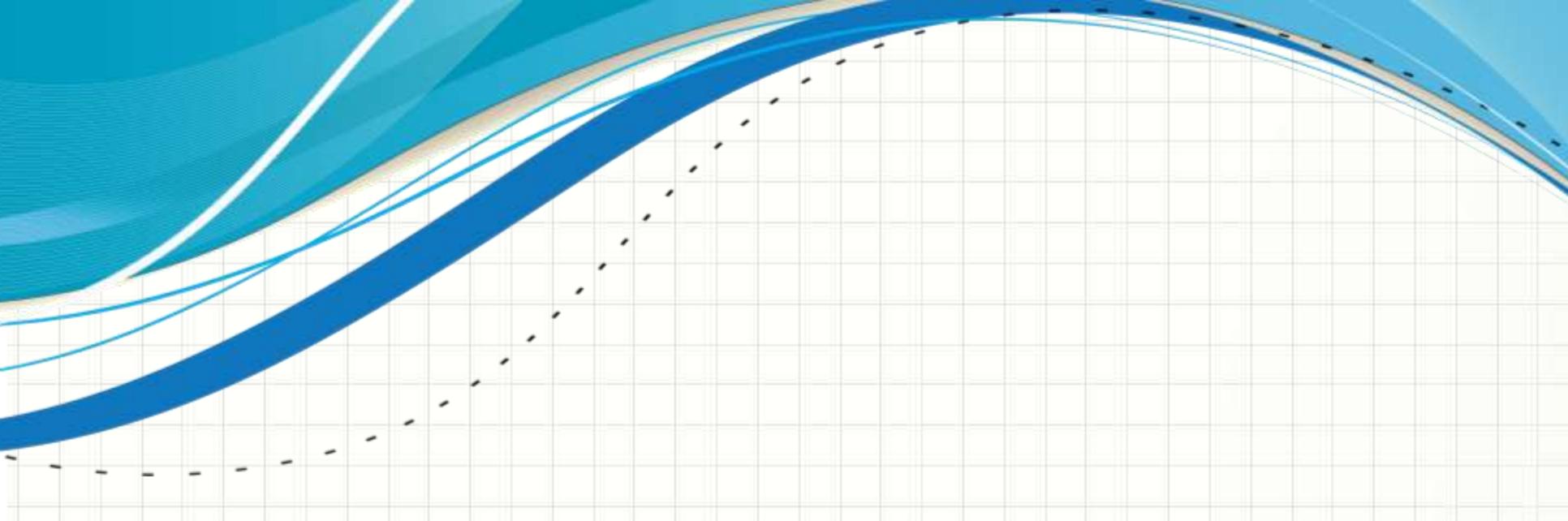
```
function mudaCor() {  
    this.style.color = “green”;  
}
```

- *this*?
- this: referencia ao objeto “atual” (clicado)

```
button1.onclick = mudaCor;
```

```
div1.onclick = mudaCor;
```

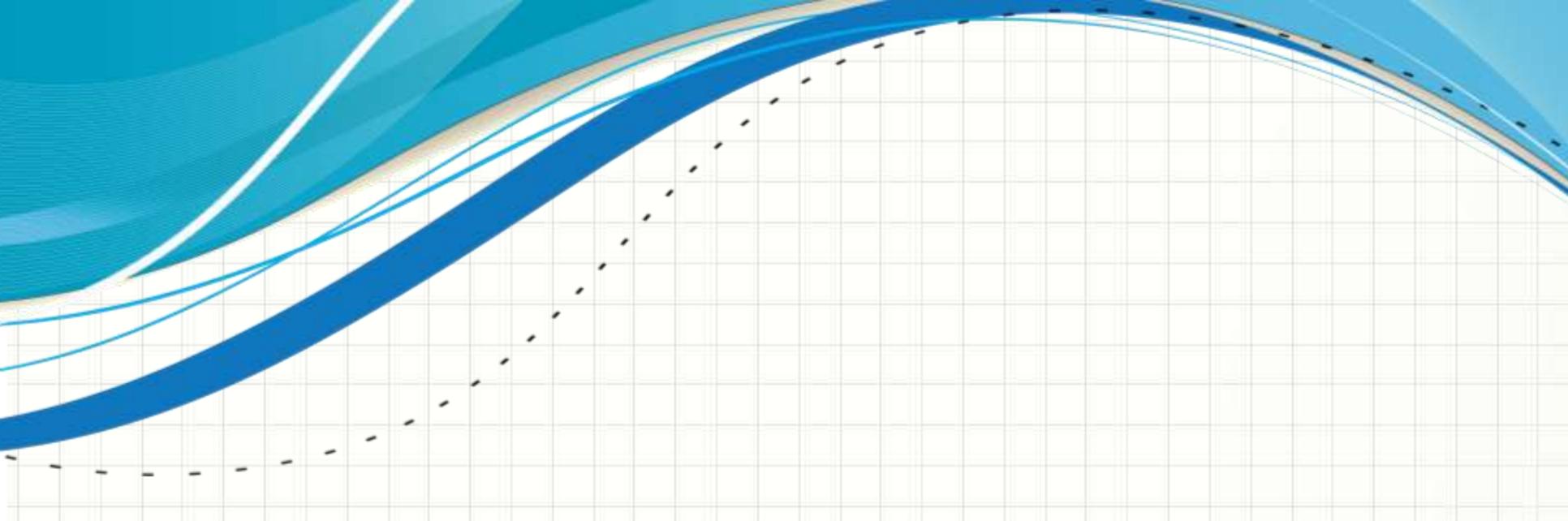
```
td1.onclick = mudaCor;
```



# **TUTORIAL: ADICIONANDO JAVASCRIPT NO JOGO DA VELHA**

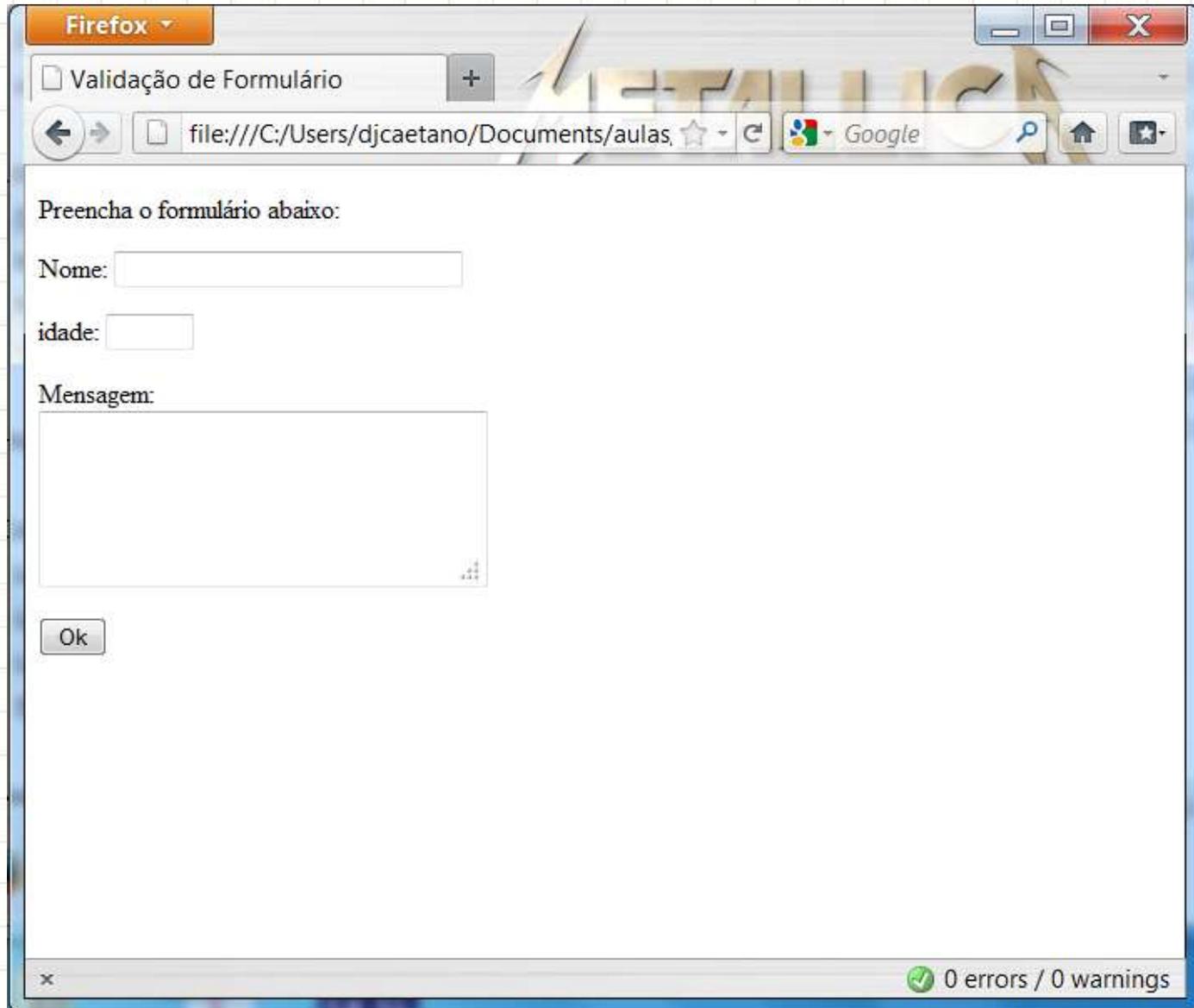
# Jogo da Velha

- Não esqueçamos de:
  - A. Sumir com fundo cinza das áreas de jogo
  - B. Tornar “invisível” elemento de seleção
  - C. Criar classes para diferentes marcações
  - D. Criar função `pintaTabuleiro()`
    - Pintar tabuleiro segundo texto
    - Ligar visibilidade da seleção só para campos vazios
  - E. Implementar função `click` (muda fundo)
    - Habilitar **click** só para campos vazios
  - F. `mostraJanela` com diferentes status de jogo
  - G. Chamar `pintaTabuleiro` no início do jogo

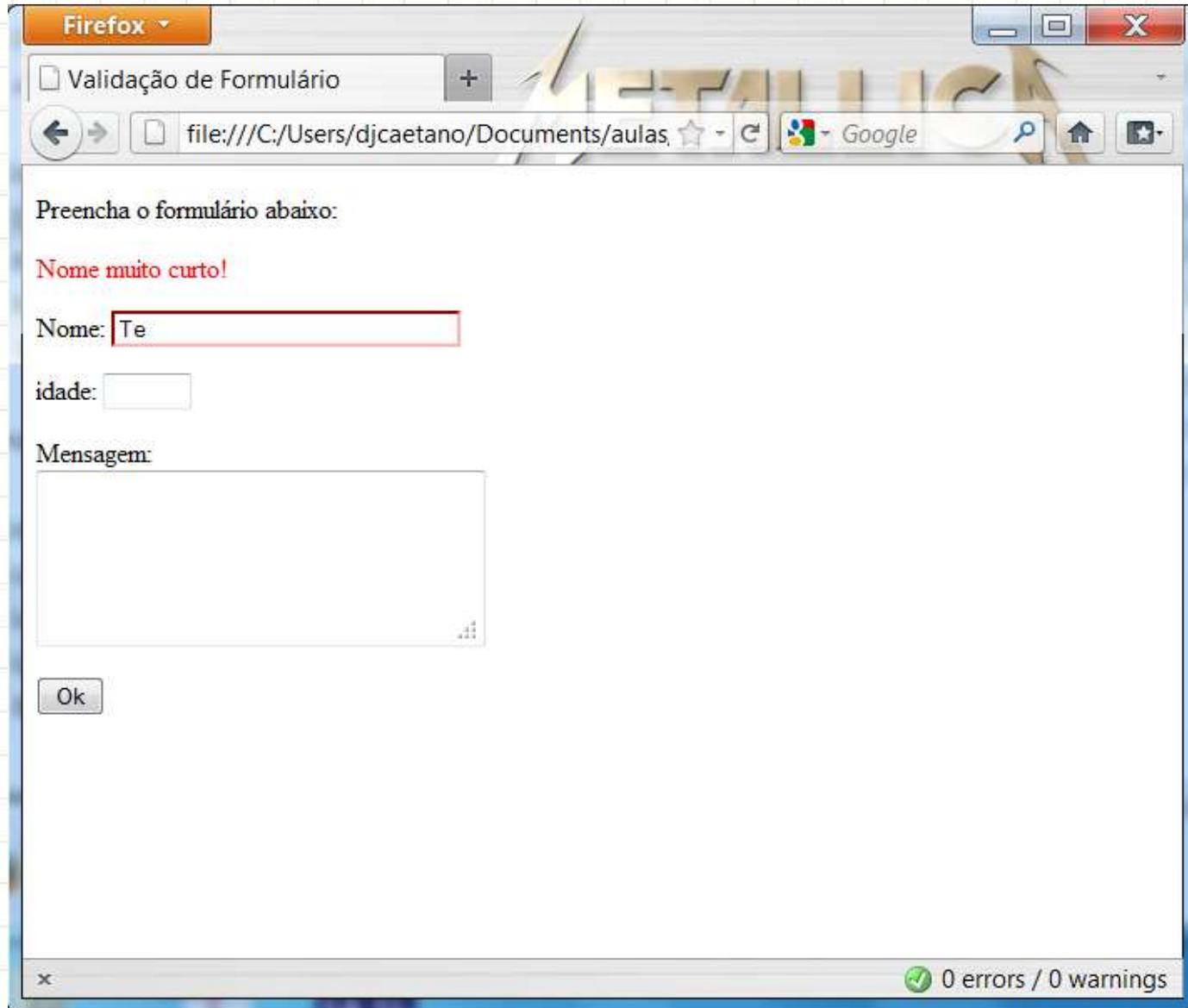


# **TUTORIAL: VALIDAÇÃO DE FORMULÁRIO**

# Validação com JavaScript + CSS



# Validação com JavaScript + CSS



Firefox

Validação de Formulário

file:///C:/Users/djcaetano/Documents/aulas

Google

Preencha o formulário abaixo:

Nome muito curto!

Nome:

idade:

Mensagem:

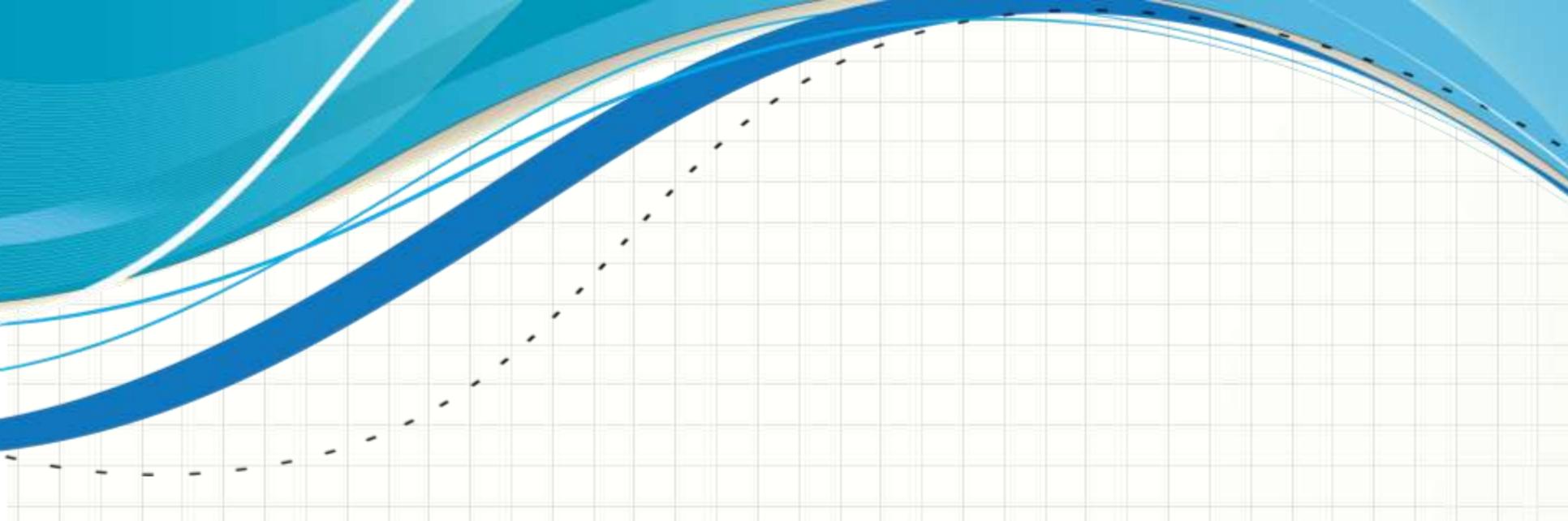
Ok

0 errors / 0 warnings

The image shows a Firefox browser window with a form titled "Validação de Formulário". The form contains three input fields: "Nome", "idade", and "Mensagem". The "Nome" field contains the text "Te" and is highlighted with a red border, indicating a validation error. Above the "Nome" field, the text "Nome muito curto!" is displayed in red. The "idade" field is empty. The "Mensagem" field is a large text area, also empty. Below the form is an "Ok" button. At the bottom right of the browser window, a status bar shows "0 errors / 0 warnings".

# Validação com JavaScript + CSS

- Não esqueçamos de:
  - A. Marcar o campo ativo com “:focus”
  - B. Marcar o campo “errado” com borda vermelha
  - C. Ativar o foco no campo “errado” com “.focus()”
  - D. Marcar texto do campo “errado” com “.select()”
  - E. Desligar o botão de envio quando for clicado
  - F. Mudar o texto do botão para “Enviando...”
  - G. Opcional: contador de caracteres do TextArea
    - Evento .onkeyup



**CONCLUSÕES**

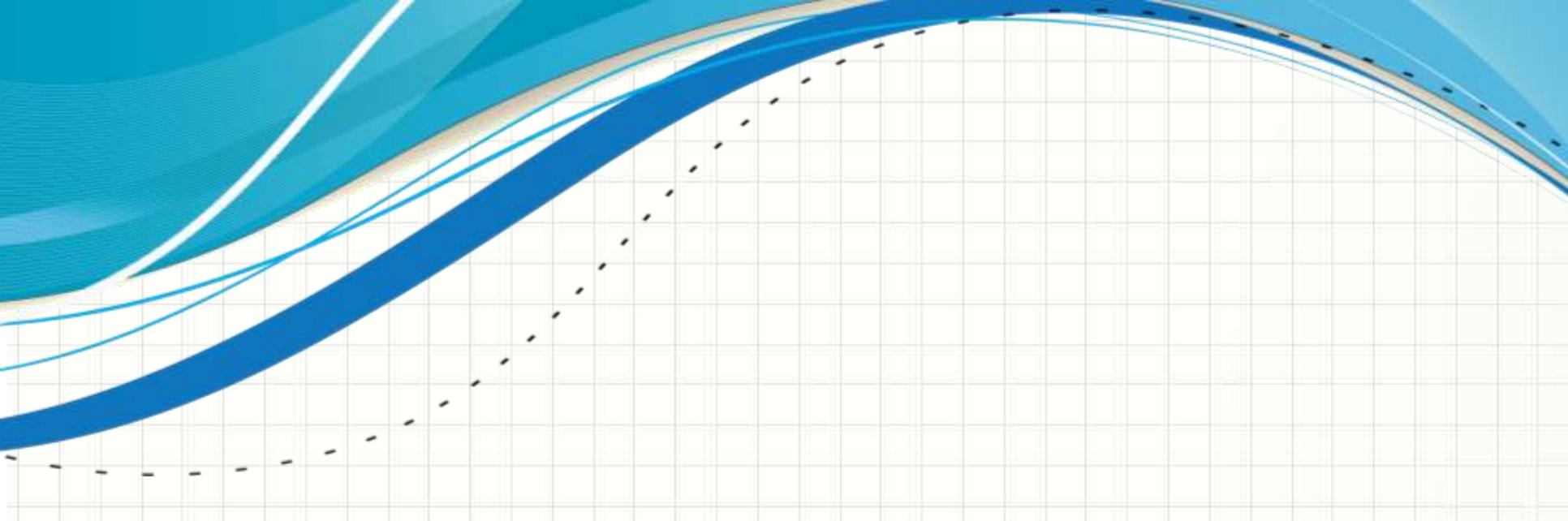
# Resumo

- JavaScript: processam. simples no lado do cliente
- JavaScript para modificar CSS:
  - Torna as páginas mais dinâmicas
  - “Sensação”: evita os tempos de espera da rede
- RIA **sempre** terrão código no lado do cliente
  - Objetivo: aumentar velocidade e dinamismo!
- **TAREFA**
  - Trabalho A!

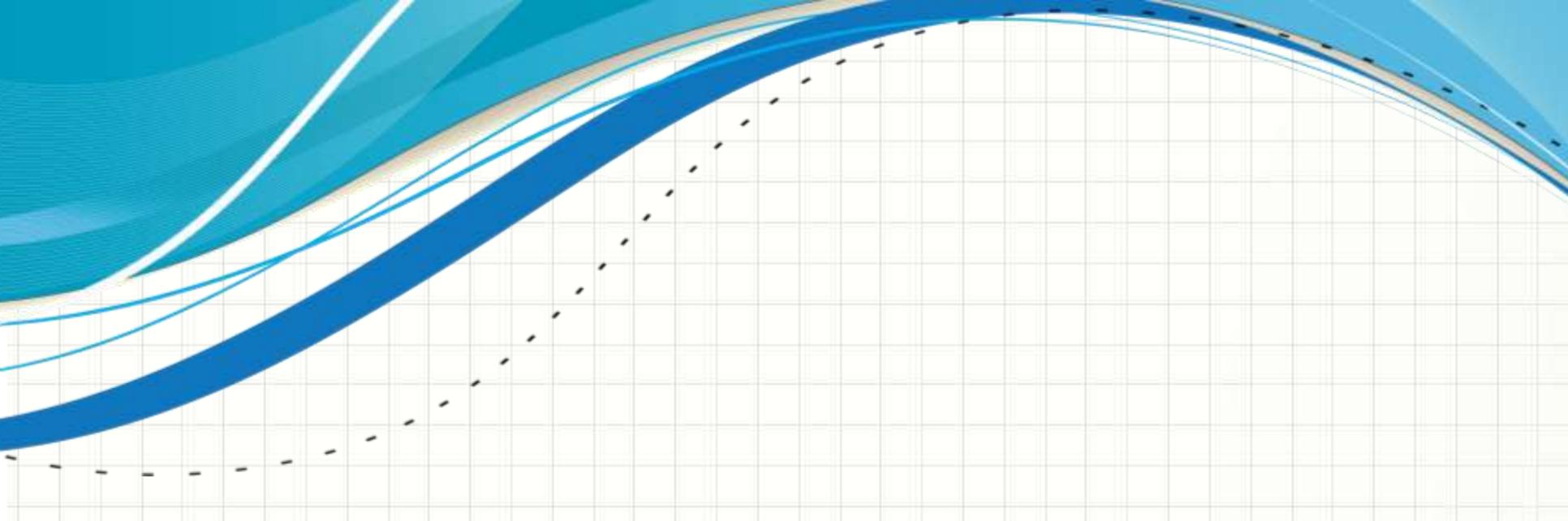
# Próxima Aula



- JavaScript e CSS só fazem essas coisas?
  - Sim e não!
  - DHTML!
  - Não vai ficar `<div>` sobre `<div>`!



**PERGUNTAS?**



**BOM DESCANSO  
A TODOS!**