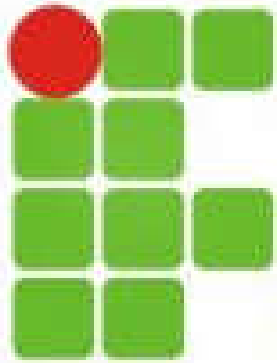
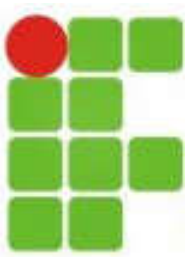
Fundamentos de Web Design 2

Professor Eng. Dr. Will Roger Pereira

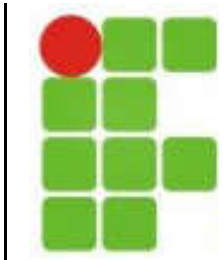


Conteúdo



- Manipular eventos dos elementos do HTML DOM;
- Manipular eventos temporais.

Introdução

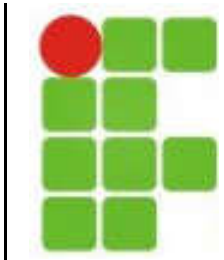


- Todas as ações do visitante que uma página Web pode responder/reagir, são chamadas eventos;

- No momento que um evento acontece, abre-se a possibilidade de executar **ações**;

- Através da programação Javascript, é possível:
  - Controlar quais elementos possuirão eventos;
  - Qual a natureza destes eventos;
  - Quais ações que serão executadas quando tais eventos forem disparados.

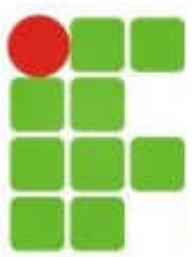
Introdução



- Um evento representa o momento preciso quando algo acontece:
  - Quando o usuário clica com o mouse sobre um elemento;
  - Quando uma imagem termina de ser carregada;
  - Quando o mouse é posicionado encima de um elemento;
  - Quando um campo de entrada é modificado;
  - Quando um formulário HTML é submetido;
  - Quando o usuário aperta uma tecla;
  - Quando um intervalo de tempo termina...

- **É o principal mecanismo de controle e interação com as ações do usuário!!!**

Manipuladores de Eventos



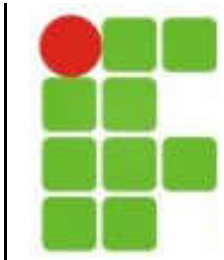
- O termo "dispara" é utilizado muito em eventos. Vem do inglês *trigger*;

- Os eventos disparados são capturados por meio de manipuladores de eventos:

Evento	Manipulador	Quando ocorre?
click	onclick	Quando o usuário clicar com o mouse sobre o elemento.

- Referência para manipuladores de eventos do HTML DOM:
  - https://www.w3schools.com/jsref/dom_obj_event.asp

Disparo de Evento



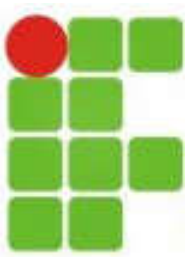
- Um evento dispara uma ação. Uma ação é representada por qual estrutura?

- **R: Função;**

- **Esta função é denominada função de callback;**

- Pode ser uma função já existente ou uma função anônima definida para aquele comportamento.

Manipulação de eventos



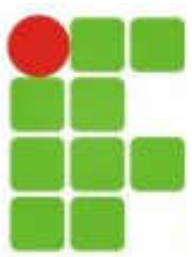
- Inline, no HTML:

```
<elemento manipulador="AÇÕES">
```

- Exemplo:

```
<p onclick="alert('clicou');">
<p onclick="função();">
```

Manipulação de eventos



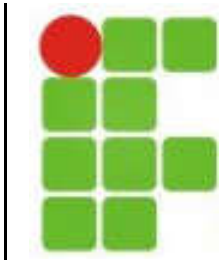
- No script, a partir do elemento selecionado:

```
elemento.manipulador = função
```

- Exemplo:

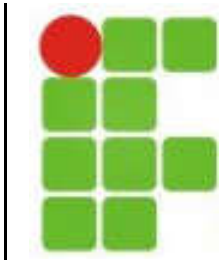
```
document.getElementById("jovem").onclick = função;
document.getElementById("jovem").onclick = function(){
        alert("clicou");
        alert(soma(1, 2));
};
```

Problema!!!



- Dentro da função de callback, pode-se utilizar todos os seletores que foram aprendidos;
- Quando somente um elemento é contemplado pelo seletor, fica muito fácil saber quem disparou o EVENTO;
- Porém quando um EVENTO é anexado a diversos elementos de uma só vez, como manipular o elemento que disparou o evento????
- R: O elemento **this**.

O elemento this

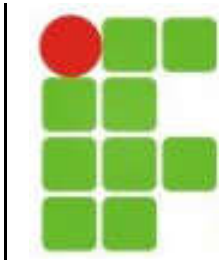


- Dentro da função de call-back, **this** representa o elemento que disparou o evento:

- Exemplo:

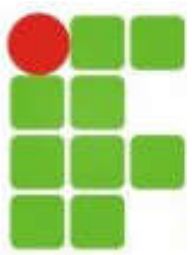
```
document.getElementById("jovem").onclick = function(){
        this.style.color = "red";
};
```

Observações importantes



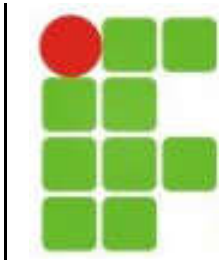
- Cada elemento pode possuir quantos eventos forem desejados;
- Caso um evento seja anexado novamente ao elemento, o evento anterior será sobrescrito;
- Um elemento pode ter um evento anexado só, e somente se ele puder ser selecionado, i. e., já estiver criado;
- Para remover o evento de um elemento, atribua **null** à seu manipulador.

Evento click



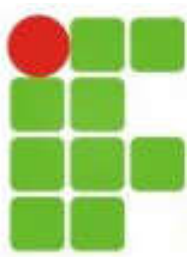
- Dispara quando o usuário realiza um clique sobre o elemento;
- Elementos alvo:
  - Todos os elementos visíveis. Normalmente não utilizado para elementos de entrada, i. e., input.
- Manipulador: **onclick**
- Exemplos em **1-click.html.**

Evento double click



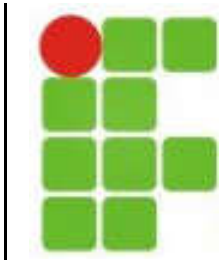
- Dispara quando o usuário realiza um clique duplo sobre o elemento;
- Elementos alvo:
  - Todos os elementos visíveis. Normalmente não utilizado para elementos de entrada, i. e., input.
- Manipulador: **ondblclick**
- Exemplos em **2-dblclick.html.**

Evento mouseenter



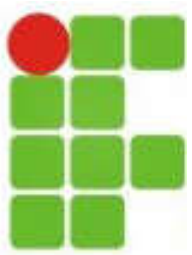
- Dispara quando o usuário entra com o mouse sobre o elemento;
- Elementos alvo:
  - Todos os elementos visíveis.
- Manipulador: **onmouseenter**

Evento mouseleave



- Dispara quando o usuário retira o mouse de sobre o elemento;
- Elementos alvo:
  - Todos os elementos visíveis.
- Manipulador: **onmouseleave**
- Exemplos em **3-onmousenter-onmouseleave.html.**

Evento input

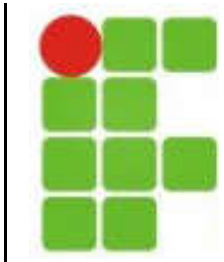


- Dispara quando o usuário modifica o texto de um elemento de entrada;

- Elementos alvo:
  - Elementos de entrada textuais, i. e., input type: text, color, email, number, password, etc.

- Manipulador: **oninput**

Evento change



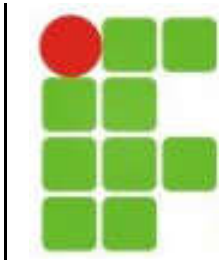
- Dispara quando o usuário modifica o valor de um elemento de entrada. Para elementos textuais, ele dispara após a perda do foco;

- Elementos alvo:
  - Elementos de entrada em geral: input, select, textarea.

- Manipulador: **onchange**

- Exemplos em **4-input-change.html.**

Evento focus

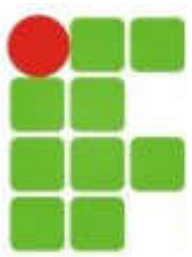


- Dispara quando o elemento recebe o foco, e. g., quando um elemento textual torna-se foco do teclado;

- Elementos alvo:
  - Todos os elementos visíveis. Mais utilizado para elementos de entrada.

- Manipulador: **onfocus**

Evento blur



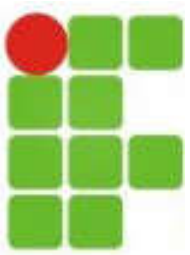
- Dispara quando o elemento perde o foco, e. g., quando um elemento textual perde o foco do teclado.

- Elementos alvo:
  - Todos os elementos visíveis. Mais utilizado para elementos de entrada.

- Manipulador: **onblur**

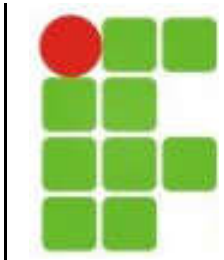
- Exemplos em **5-focus-blur.html.**

Evento submit



- Dispara quando o usuário submete um formulário;
- Elementos alvo:
  - Somente formulários.
- Manipulador: **onsubmit**
- Exemplos em **6-submit.html.**

Eventos temporais

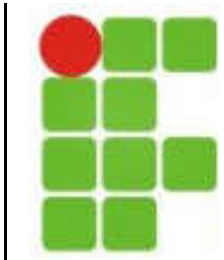


- Há também a possibilidade de colocar eventos para acontecerem depois de passado algum tempo;

- Para isto, temos as seguintes modalidades:
  - Disparo uma única vez após um tempo determinado;
  - Disparo a cada vez que uma quantidade de tempo determinada se passa.

- Também há a possibilidade de cancelamento destes eventos.

setTimeout



- Para executar uma função, uma única vez, após passado determinado tempo. Utilize a seguinte função:

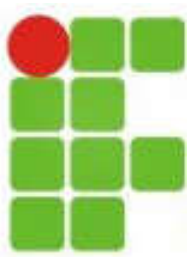
```
setTimeout(função, tempo)
```

  - **Função:** Função a ser executada quando o evento disparar;
  - **Tempo:** Tempo, em milissegundos, até o evento disparar;
  - **Retorno:** Uma referência do evento temporal.

- Para cancelar o evento temporal, utilize sua referência **ref** na seguinte função:

```
clearTimeout(ref)
```

setInterval



- Para executar uma função, uma vez a cada intervalo de tempo. Utilize a seguinte função:

```
setTimeout(função, tempo)
```

  - **Função:** Função a ser executada quando o evento disparar;
  - **Tempo:** Duração do intervalo, em milissegundos, até o evento disparar;
  - **Retorno:** Uma referência do evento temporal.

- Para cancelar os eventos temporais intervalados, utilize sua referência **ref** na seguinte função:

```
clearInterval(ref)
```