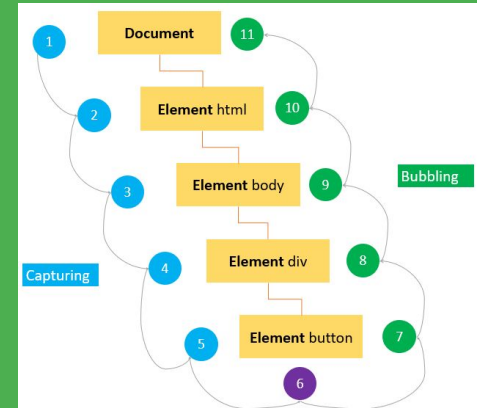


Desenvolvimento WEB II

Prof. Ernani Gottardo

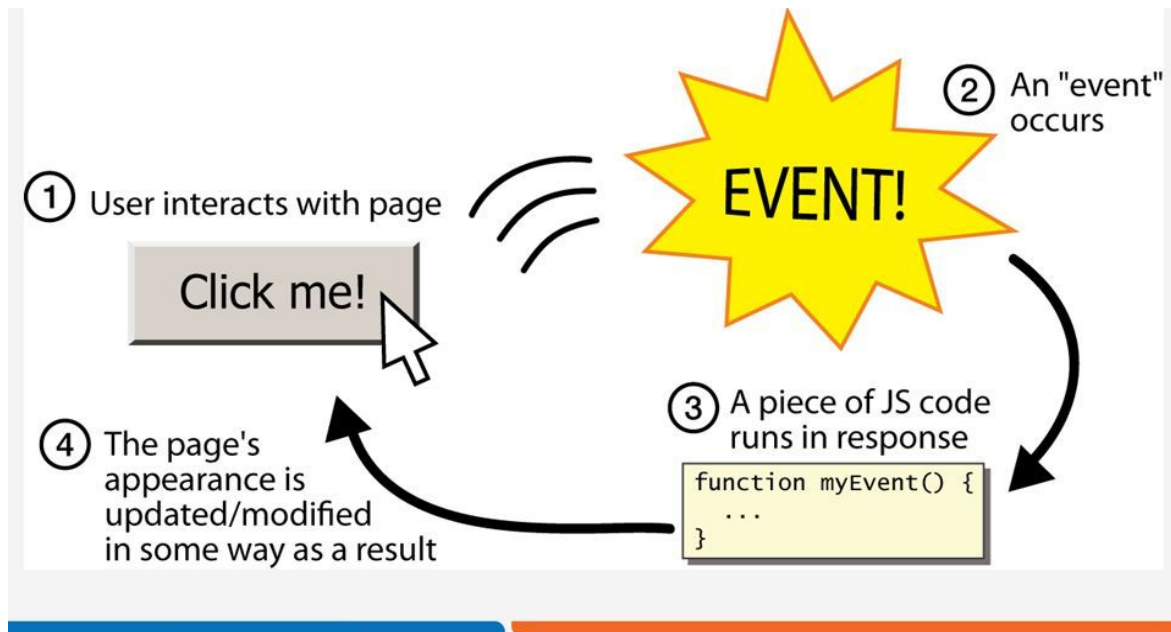
DOM - Document Object Model - eventos



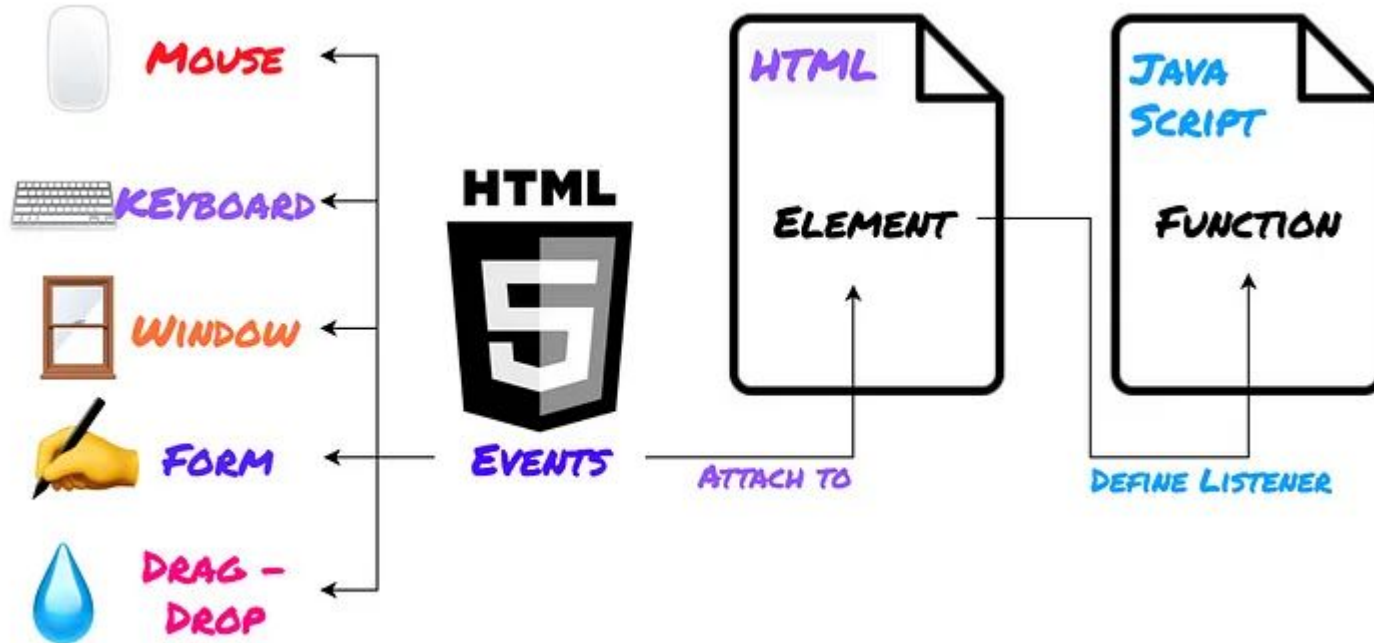
Eventos

- Os eventos DOM permitem que o JavaScript adicione ouvintes (***listeners***) de eventos ou manipuladores (***handling***) de eventos a elementos HTML.
- Eventos HTML são ações que acontecem com elementos HTML.
- Quando JavaScript é usado em páginas HTML, JavaScript pode "**reagir**" a esses eventos.
- Um evento HTML pode ser gerado pelo **navegador** ou **usuário**.
 - Uma página da web HTML **terminou de carregar**
 - Um campo de entrada HTML **foi alterado**
 - Um botão HTML foi **clicado**

Eventos - ciclo



Eventos - categorias



Exemplos de eventos

Evento	Descrição
abort (onabort)	Este evento acontece quando o usuário interrompe o carregamento de um recurso
blur (onblur)	Ocorre quando o elemento perde o foco , ou seja, quando o usuário clica fora do elemento, este não será mais selecionado como estando ativo.
change (onchange)	Ocorre quando o usuário altera o conteúdo de um campo de dados de um formulário.
click (onClick)	Ocorre quando o usuário clica no elemento associado ao evento.
dblclick (ondblclick)	Ocorre quando o usuário clica duas vezes no elemento associado ao evento (um hiperlink ou um elemento de formulário). Este evento só é suportado pelas versões do JavaScript 1.2 e superior
error (onerror)	É desencadeada quando ocorre um erro durante o carregamento da página.
focus (onfocus)	Ocorre quando o usuário dá foco a um elemento, isso significa que este elemento foi selecionado como estando ativo
mousemove (onmousemove)	Quando o cursor do mouse é movido

Exemplos de eventos

Evento	Descrição
keypress (onkeypress)	Ocorre quando o usuário mantém pressionada uma tecla do seu teclado. Este evento só é suportado pelas versões do JavaScript 1.2 e superior
load (onload)	Ocorre quando o navegador do usuário carrega a página em curso
mouseover (onmouseover)	Ocorre quando o usuário põe o cursor do mouse acima de um elemento
mouseout (onmouseout)	Ocorre quando o cursor do mouse deixa um elemento. Este evento faz parte do Javascript 1.1.
reset (onreset)	Ocorre quando o usuário apaga os dados de um formulário com o botão Reset.
submit (onsubmit)	Ocorre quando o usuário clica no botão de submissão de um formulário (o botão que permite enviar o formulário)
unload (onunload)	Ocorre quando o navegador do usuário sai da página em curso

https://www.w3schools.com/jsref/dom_obj_event.asp

Manipuladores de Eventos

- Em JavaScript pode-se definir a manipulação de eventos utilizando diversas estratégias.
- É possível inserir JavaScript diretamente dentro do atributo (**inline**), como no exemplo abaixo, porém, isso é uma **prática ruim**.

```
<input type="button" id="btnteste" name="btnteste1" value="Teste"
onclick="alert('Olá, forma mais antiga de manipulação de eventos');" />
```

Manipuladores de Eventos

- Outra maneira de utilizar eventos consiste em chamar a execução de uma função diretamente no código HTML, conforme o exemplo abaixo (esta função pode estar em um arquivo **javascript externo**)

HTML

```
<input type="button" id="btnteste" name="btnteste1" value="Teste"
onclick=bgChange() />
```

Javascript

```
function bgChange() {
    var rndCol = 'rgb(' + Math.random() * 255 + ',' + Math.random() * 255 + ',' +
Math.random() * 255 + ')';
    document.body.style.backgroundColor = rndCol;
}
```


Manipuladores de Eventos

- Mecanismos mais recentes, envolvem a definição dos manipuladores de evento **somente no código JavaScript**. Uma forma inicial de utilizar estes novos procedimentos, pode ser visto no exemplo abaixo:

HTML

```
<input type="button" id="btnteste" name="btnteste1" value="Teste"/>
```

Javascript

```
var btn = document.getElementById('btnteste'); //DOM  
btn.onclick = bgChange;
```

Manipuladores de Eventos

- O método mais atual, envolve a definição dos manipuladores de evento **somente no código JavaScript** usando o método **addEventListener**

HTML

```
<input type="button" id="btnteste" name="btnteste1" value="Teste"/>
```

Javascript

```
var btn = document.getElementById('btnteste'); //DOM  
btn.addEventListener( 'click', bgchange );
```

Objeto Alvo de Evento

- A propriedade '**target**' é uma referência ao objeto no qual o evento ocorreu.
- É possível acessar as informações do elemento alvo do evento, conforme exemplo abaixo

```
function bgChange(e) {  
    var rndCol = 'rgb(' + Math.random() * 255 + ',' + Math.random() * 255 + ',' +  
Math.random() * 255 + ')';  
    document.body.style.backgroundColor = rndCol;  
    alert('Objeto gerador do evento ' + e.target.getAttribute('name'))  
}
```

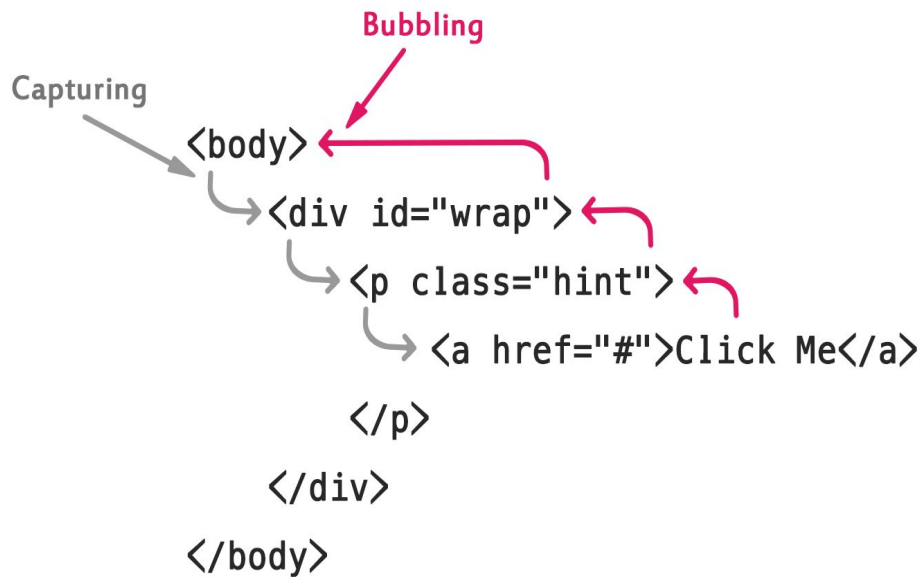
Eventos - passagem de parâmetros

- Para realizar a passagem de parâmetros em eventos no javascript faz-se necessário utilizar as chamada **funções anônimas**.

```
var nm = document.getElementById('nm1');  
nm.addEventListener('blur', function (e) {  
    checkUsername(5);  
});  
  
function checkUsername(tam) {  
    var nm1 = document.getElementById('nm1');  
    if (nm1.value.length < tam) {  
        p = document.getElementById('msg');  
        p.innerHTML = 'Tamanho do nome inválido';  
    } else {  
        p.innerHTML = '';  
    }  
}
```

Propagação Eventos

- Os eventos JavaScript “se propagam” para elementos HTML próximos.
- Bubbling(default)** : Fluxo de “baixo” para “cima” (padrão)
- Capturing**: Fluxo de “cima” para “baixo”



Eventos - evitar propagação

- O método **stopPropagation()** impede que a **propagação** do mesmo evento seja chamada.
- Exemplos de uso:

```
<div id="div1">  
  <p id="p1">Este é um parágrafo.</p>  
  <p id="p2">Este é outro parágrafo</p>  
</div>  
  
let p1=document.getElementById('p1');  
p1.addEventListener('click', function(e){  
  e.stopPropagation(); // Impede a propagação do evento para elementos pai  
  alert('Clicou no parágrafo 1');  
});  
  
let div1 = document.getElementById('div1');  
div1.addEventListener('click', function (e) {  
  alert('Clicou na div'); //não executa  
});
```

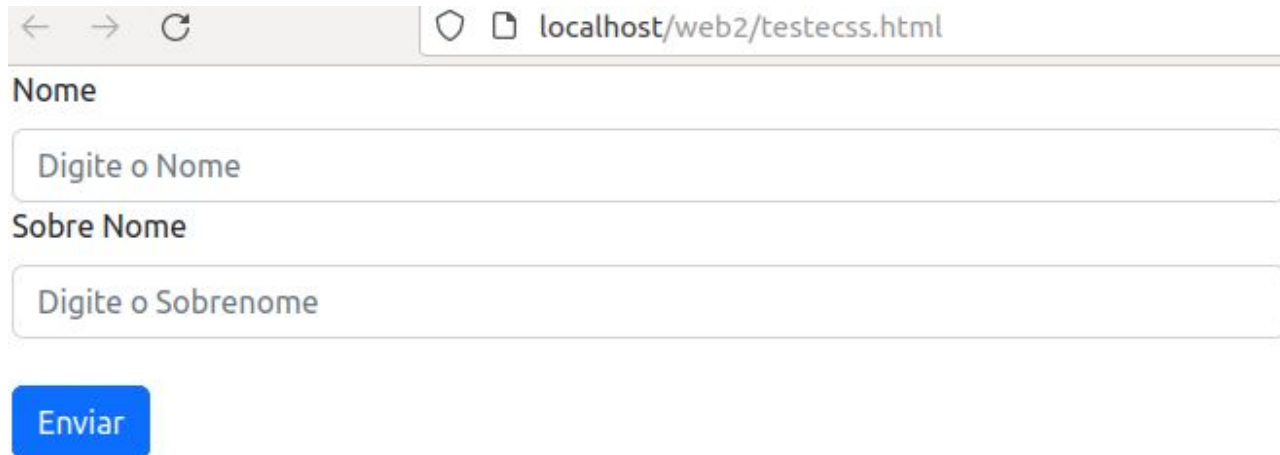
Eventos - evitar comportamento padrão

- O método **preventDefault()** cancela o evento, o que significa que a ação padrão que pertence ao evento não ocorrerá.
- Exemplos de uso:
 - Clicar no link não abre a url destino

```
<a id="ifrs" href="https://ifrs.edu.br" target="_blank">IFRS</a>  
let ifrs = document.getElementById('ifrs');  
ifrs.addEventListener('click', function (e) {  
    e.preventDefault(); // Impede o comportamento padrão do link  
    alert('Você clicou no link do IFRS!');  
});
```

Eventos - validação de formulário

- O método **preventDefault()** possibilita cancelar o envio do formulário em caso de problemas de validação



A screenshot of a web browser window. The address bar shows 'localhost/web2/testecss.html'. The page contains a form with two text input fields. The first field is labeled 'Nome' and contains the placeholder text 'Digite o Nome'. The second field is labeled 'Sobre Nome' and contains the placeholder text 'Digite o Sobrenome'. Below the fields is a blue button labeled 'Enviar'.

Eventos - validação de formulário

```
<div class="container-fluid">  
  <form action="teste.php" method="get" id="form1">  
    <label for="" class="form-label">Nome</label>  
    <input type="text" class="form-control" name="nome" id="nm1">  
    <label for="" class="form-label">Sobre Nome</label>  
    <input type="text" class="form-control" name="sobrenome" id="snome">  
    <br>  
    <button type="submit" class="btn btn-primary">Enviar</button>  
    <button type="button" class="btn btn-primary"  
id='verificar'>Verificar</button>  
  </form>  
</div>  
<p id='msg'></p>  
  
.erro{  
  border-color: red;  
  box-shadow: 2px 0px 5px red;  
}
```

Eventos - validação de formulário

```
var form1 = document.getElementById('form1');  
form1.addEventListener('submit', validar);
```

```
btnv= document.getElementById('verificar');  
btnv.addEventListener('click',validar);
```

```
function ver_nome(){  
    var nm = document.getElementById('nm1');  
    if (nm.value == ''){  
nm.classList.add('erro');  
        return false;  
    } else {  
        nm.classList.remove('erro');  
        return true;  
    }  
}
```

Eventos - validação de formulário

```
function ver_snome(){
    var snm = document.getElementById('snome');
    if (snm.value == ''){
        nm.classList.add('erro');
        return false;
    } else {
        nm.classList.remove('erro');
        return true;
    }
}

function validar(e) {
    let nome= ver_nome();
    let snome= ver_snome();
    if ((nome == false) || (snome==false)) {
        m = document.getElementById('msg');
        m.innerHTML = 'Informações Incompletas';
        e.preventDefault();
    }
}
```

End