COM222 DESENVOLVIMENTO DE SISTEMAS WEB

Aula 07: Eventos DOM

Conteúdo

- Document Object Model
  - Eventos
  - Listeners
  - Exercícios

Eventos

- Código Javascript é “event-driven”
  - Código, em geral, só executa quanto um evento é disparado

Exemplo:
Elemento de interface com o qual o usuário pode interagir

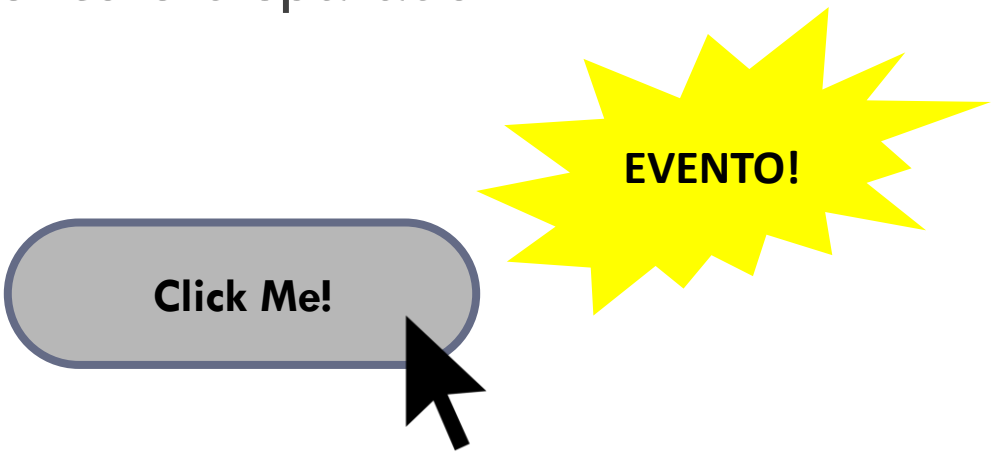
Eventos

- Código Javascript é “event-driven”
  - Código, em geral, só executa quanto um evento é disparado

Quando o usuário clica no botão...

Eventos

- Código Javascript é “event-driven”
  - Código, em geral, só executa quanto ocorre um evento é disparado



… botão emite um “evento”, que é como um anúncio de que algo aconteceu.

Eventos

- Código Javascript é “event-driven”
  - Código, em geral, só executa quanto ocorre um evento é disparado


```
function onClick() {
  ...
}
```

Uma função que esteja “escutando” este evento é executada. Esta função é chamada de "**event handler.**"

Event listener

- Todo elemento DOM tem o seguinte método definido:
  - addEventListener(nome_evento, nome_função)
    - nome_evento: é o nome do evento Javascript que se deseja capturar
      - click, focus, blur, etc.
    - nome_função: é o nome da função Javascript que deve ser executada quando o evento for disparado

Event listener

- É possível também remover um event listener
  - removeEventListener(nome_evento, nome_função)

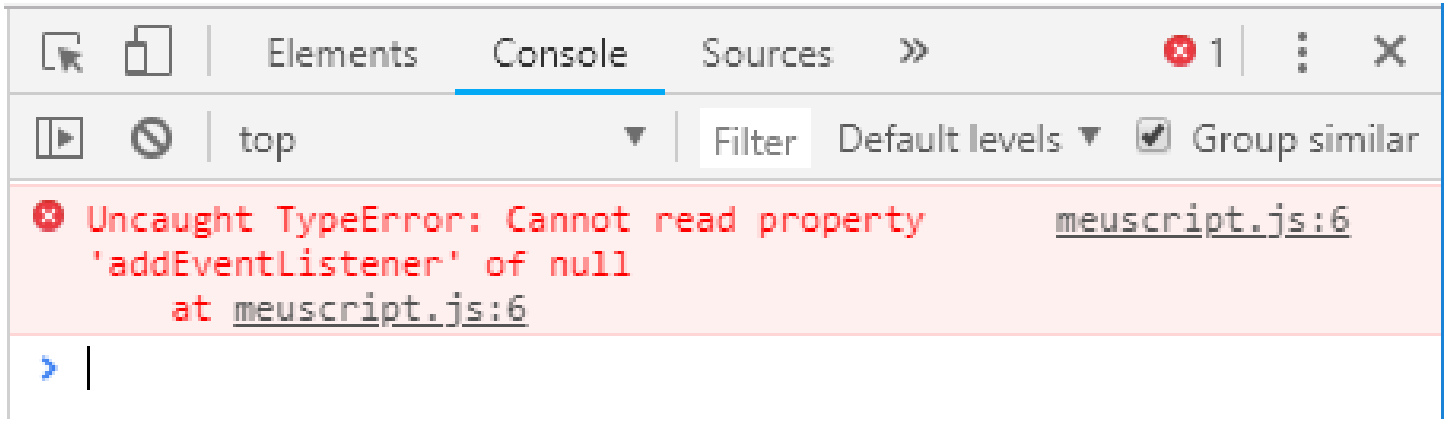
Event listener

- O uso correto de event listeners é importante, pois, segundo as boas práticas de programação web:
  - HTML deve conter apenas tags de estruturação
    - Estilização fica no arquivo CSS
    - Programação de eventos fica no arquivo Javascript
  - Veremos um exemplo a seguir

```html
<!DOCTYPE html>
<html>
   <head>
      <meta charset="UTF-8">
      <title>Exemplo eventos JS</title>
      <script src="meuscript.js"></script>
   </head>
   <body>
      <button>Click Me!</button>
   </body>
</html>
```

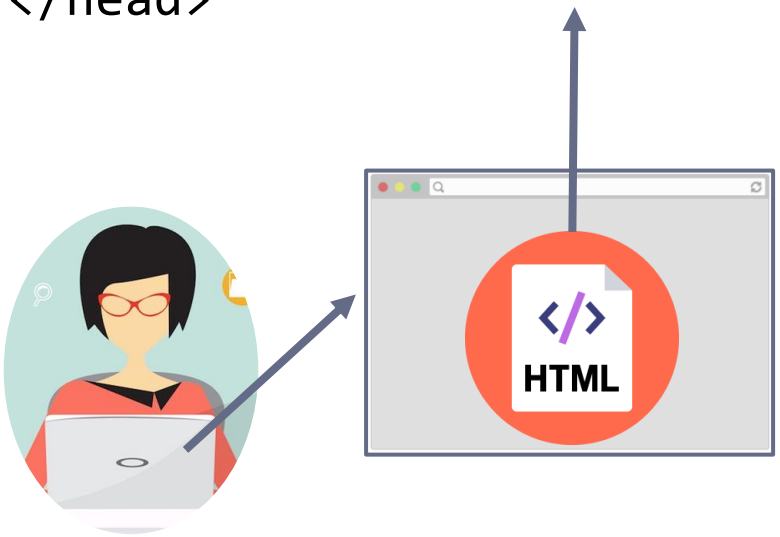
```javascript
function onClick(){
   alert("Fui clicado!");
}

const button = document.querySelector('button');
button.addEventListener('click', onClick);
```

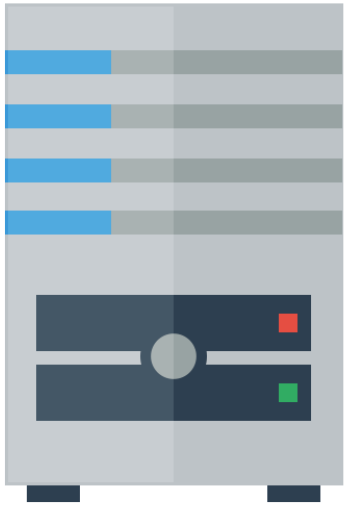
```
function onClick(){
    alert("Fui clicado!");
}

const button = document.querySelector('button');
button.addEventListener('click', onClick);
```



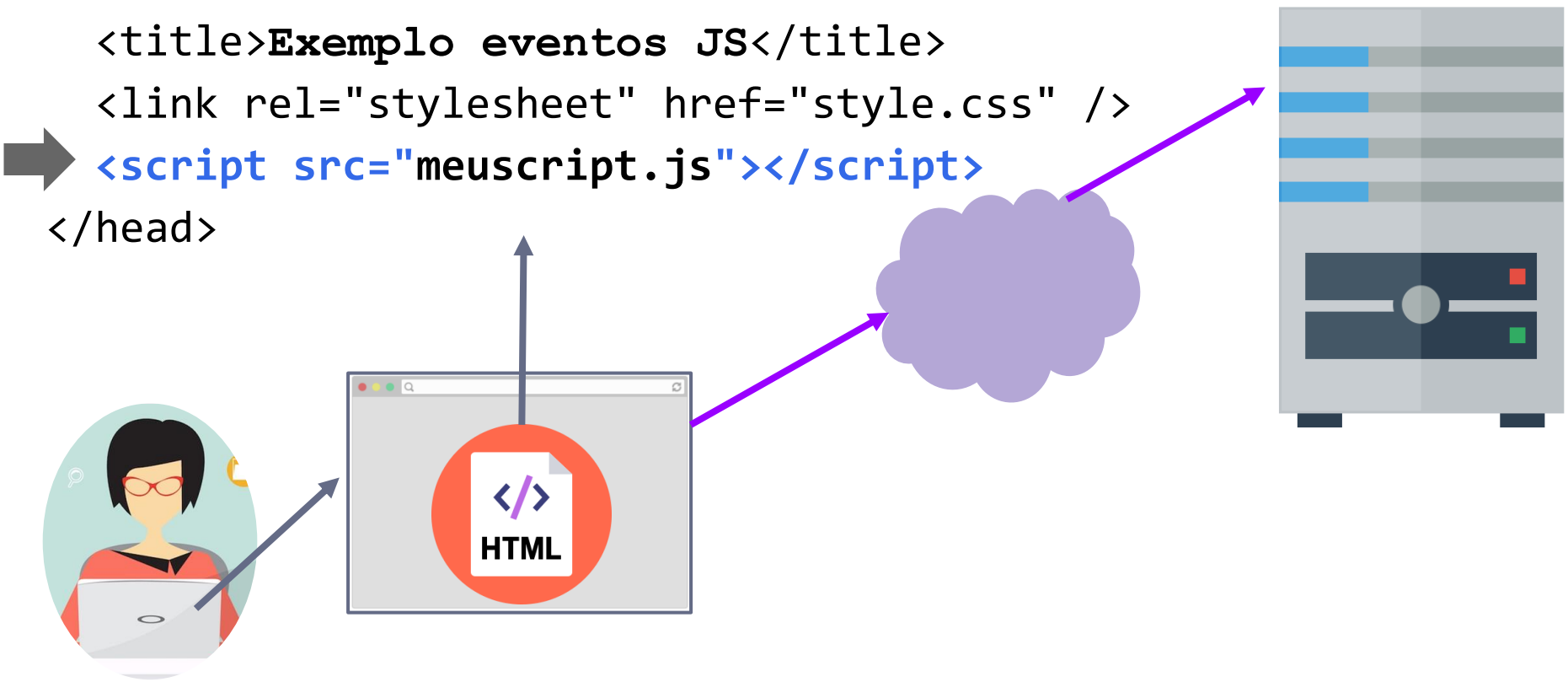
Por quê?

```
<head>
  <title>Exemplo eventos JS</title>
  <link rel="stylesheet" href="style.css" />
  <script src="meuscript.js"></script>
</head>
```

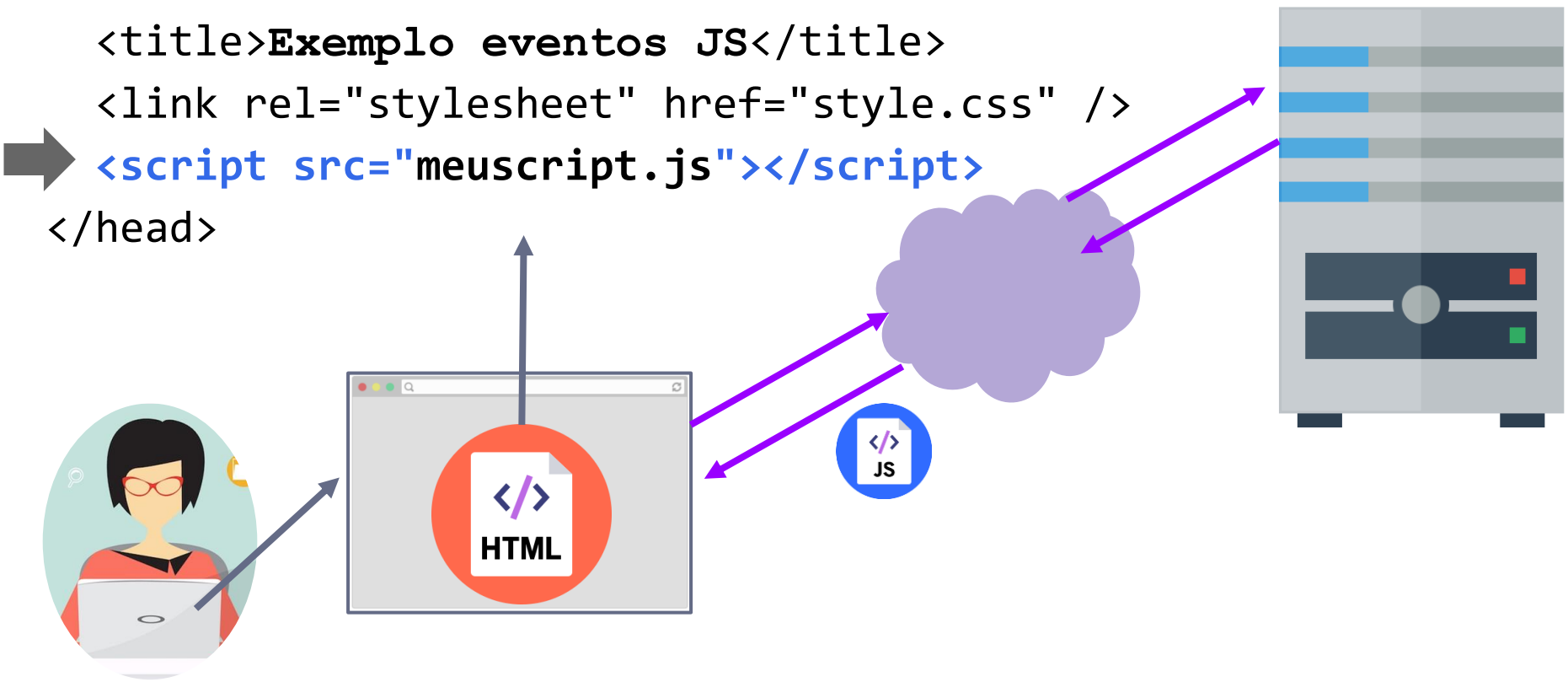




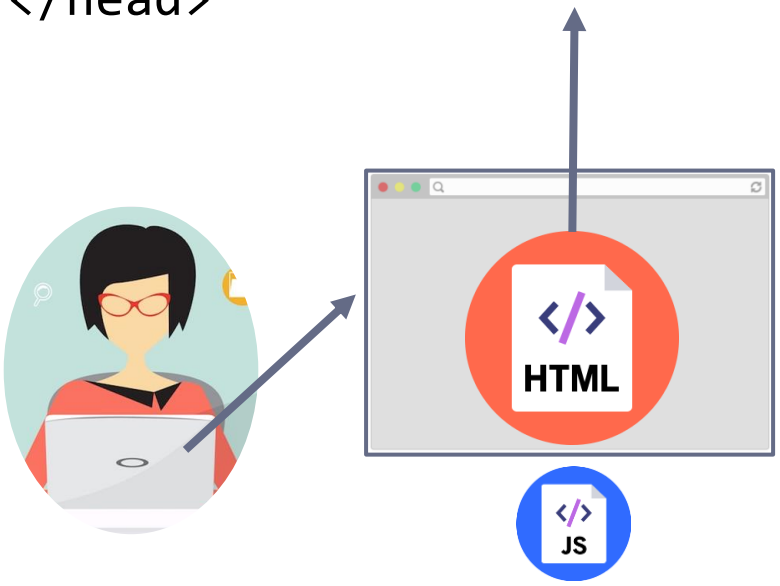
```
<head>
  <title>Exemplo eventos JS</title>
  <link rel="stylesheet" href="style.css" />
  <script src="meuscript.js"></script>
</head>
```



```
<head>
  <title>Exemplo eventos JS</title>
  <link rel="stylesheet" href="style.css" />
  <script src="meuscript.js"></script>
</head>
```

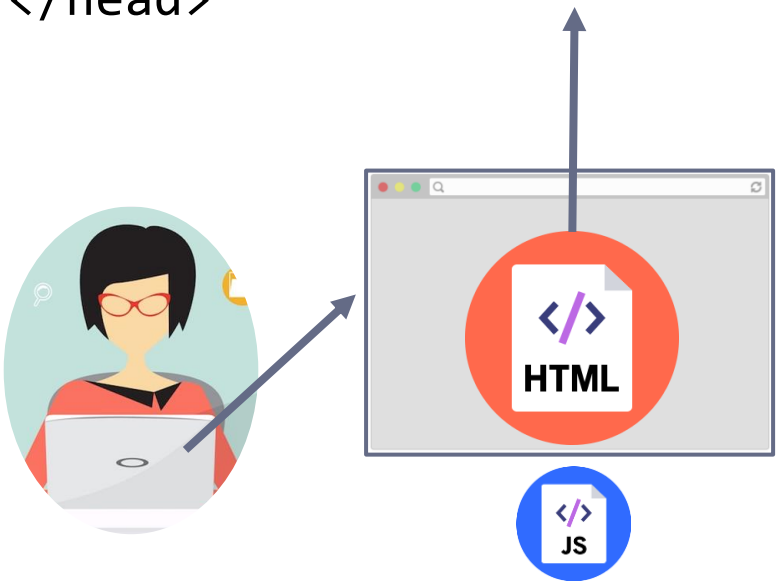


```
<head>
  <title>Exemplo eventos JS</title>
  <link rel="stylesheet" href="style.css" />
  <script src="meuscript.js"></script>
</head>
```



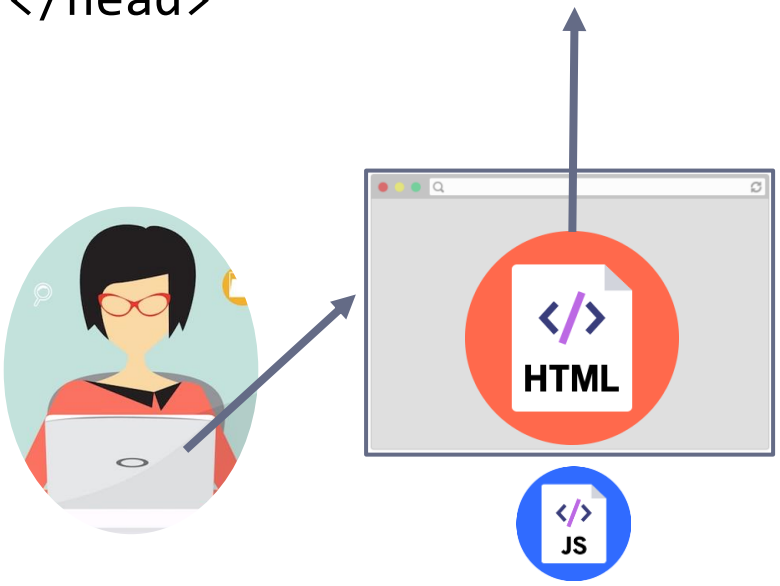

```
function onClick(){
    alert("Fui clicado!");
}

const button =
document.querySelector('button');
button.addEventListener('click',
onClick);
```

```
<head>
  <title>Exemplo eventos JS</title>
  <link rel="stylesheet" href="style.css" />
  <script src="meuscript.js"></script>
</head>
```




```
function onClick(){
    alert("Fui clicado!");
}

const button =
document.querySelector('button');
button.addEventListener('click',
onClick);
```

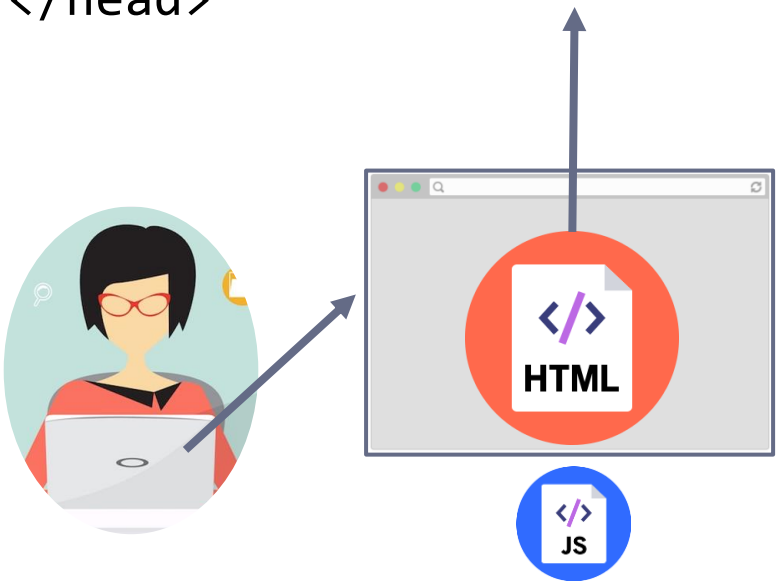
```
<head>
  <title>Exemplo eventos JS</title>
  <link rel="stylesheet" href="style.css" />
  <script src="meuscript.js"></script>
</head>
```




```
function onClick(){
    alert("Fui clicado!");
}

const button =
document.querySelector('button');
button.addEventListener('click',
onClick);
```

Estamos ainda na tag <script>, que fica no topo do documento HTML. Então <button> ainda não está disponível.

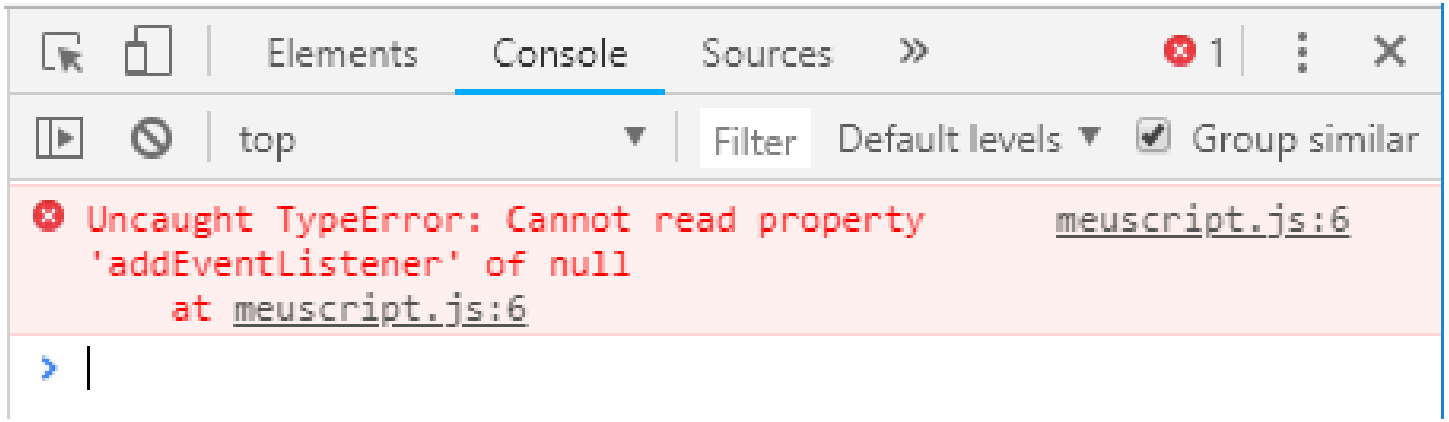
```
<head>
  <title>Exemplo eventos JS</title>
  <link rel="stylesheet" href="style.css" />
  <script src="meuscript.js"></script>
</head>
```




```
function onClick(){
    alert("Fui clicado!");
}

const button =
document.querySelector('button');
button.addEventListener('click',
onClick);
```

Assim, querySelector retorna null, e não podemos chamar addEventListener em null.

```
function onClick(){
    alert("Fui clicado!");
}

const button = document.querySelector('button');
button.addEventListener('click', onClick);
```



Solução: defer

- O atributo “defer” da tag script faz com que o código Javascript só execute após o carregamento do DOM

  <script src=“meuscript.js” defer>...</script>

- Existem outras soluções para este problema, mas são antigas (antes do defer ser suportado)
  - Colocar a tag <script> no final da página
  - Esperar o evento de load da página para depois disparar eventos Javascript

```html
<!DOCTYPE html>
<html>
   <head>
      <meta charset="UTF-8">
      <title>Exemplo eventos JS</title>
      <script src="meuscript.js" defer></script>
   </head>
   <body>
      <button>Click Me!</button>
   </body>
</html>
```

```javascript
function onClick(){
   alert("Fui clicado!");
}

const button = document.querySelector('button');
button.addEventListener('click', onClick);
```

Exemplo: Presente

Click for a present:

Veja o código no CodePen

```
function openPresent() {
  const image = document.querySelector('img');
  image.src = 'https://media.giphy.com/media/27ppQUOxe7KlG/giphy.gif';
}

const image = document.querySelector('img');
image.addEventListener('click', openPresent);
```

Encontrando elemento 2 vezes

```
function openPresent() {
  const image = document.querySelector('img');
  image.src = 'https://media.giphy.com/media/27ppQUOxe7KlG/giphy.gif';
}

const image = document.querySelector('img');
image.addEventListener('click', openPresent);
```

Essa redundância é estranha!

Há uma forma de resolver isso?

Encontrando elemento 2 vezes

```
function openPresent() {
  const image = document.querySelector('img');
  image.src = 'https://media.giphy.com/media/27ppQUOxe7KlG/giphy.gif';
}

const image = document.querySelector('img');
image.addEventListener('click', openPresent);
```

Essa redundância é estranha!

Há uma forma de resolver isso?

CodePen

event.currentTarget

Um elemento event é passado para o listener como parâmentro:

```
function openPresent(event) {
  const image = event.currentTarget;
  image.src = 'https://media.giphy.com/media/27ppQUOxe7KlG/giphy.gif';
  image.removeEventListener('click', openPresent);
}

const image = document.querySelector('img');
image.addEventListener('click', openPresent);
```

A propriedade currentTarget é uma referência ao objeto associado ao evento, neste caso o elemento <img> ao qual associamos o listener.

Exemplo: Presente

Click for a present:

Seria interessante mudar o texto após o presente ser revelado.

Algumas propriedades dos elementos

Propriedade	Descrição
`id`	O valor do atributo id do elemento, como uma string
`innerHTML`	O HTML “puro” entre a tag de início e a tag de fim do elemento, como uma string
`textContent`	O conteúdo de texto de um nó e seus descendentes. (Essa propriedade é herdada de Node)
`classList`	Um objeto contendo as classes aplicadas ao elemento

Alternativa 1: ajustar o
`textContent`!
CodePen

```
function openPresent(event) {
  const image = event.currentTarget;
  image.src = 'https://media.giphy.com/media/27ppQUOxe7KlG/giphy.gif';

  const title = document.querySelector('h1');
  title.textContent = 'Hooray!';

  image.removeEventListener('click', openPresent);
}

const image = document.querySelector('img');
image.addEventListener('click', openPresent);
```

Neste exemplo, o elemento h1 é selecionado e seu textContent é alterado. (CodePen)

Outra alternvativa: Mudar os elementos

Adicionar elementos via DOM

Podemos criar elementos dinamicamente e adicioná-los à página web utilizando `createElement` and `appendChild`:

```
document.createElement(tag string)
element.appendChild(element);
```

É possível também adicionar elementos à página utilizando innerHTML, mas isso gera um security risk.

```
// Tente não usar innerHTML dessa forma:
element.innerHTML = '<h1>Hooray!</h1>';
```

Remover elementos via DOM

Para remover elementos do DOM:

```
element.remove();
```

Outra possibilidade de remoção é ajustar o `innerHTML` de um elemento para uma **string vazia**. Essa é uma forma eficaz de remover todos os nós filho de um nó pai:

```
element.innerHTML = '';
```

```
function openPresent(event) {
  const newHeader = document.createElement('h1');
  newHeader.textContent = 'Hooray!';
  const newImage = document.createElement('img');
  newImage.src = 'https://media.giphy.com/media/27ppQUOxe7KlG/giphy.gif';

  const container = document.querySelector('#container');
  container.innerHTML = '';
  container.appendChild(newHeader);
  container.appendChild(newImage);
}

const image = document.querySelector('img');
image.addEventListener('click', openPresent);
```

CodePen

O efeito resultante não foi dos melhores: o texto muda mais rápido e aparece muito antes da figura

Como resolver isso?

display: none;

Para solucionar o problema vamos usar display:

```
display: block;
display: inline;
display: inline-block;
display: flex;
display: none;
```

display: none;

Não exibe o elemento e seus filhos, mas o carregamento do conteúdo associado ao elemento e seus filhos ocorre normalmente.

```html
<div id="gift-outside">
  <h1>Click for a present:</h1>
  <img src="https://s3-us-west-2.amazonaws.com/s.cdpn.io/1083533/gift-icon.png" />
</div>
<div id="gift-inside" class="hidden">
  <h1>Hooray!</h1>
  <img src="https://media.giphy.com/media/27ppQUOxe7KlG/giphy.gif" />
</div>
```

```css
.hidden {
  display: none;
}
```

A solução consiste em criar duas view, uma delas escondida (hidden) (CodePen)

```
function openPresent(event) {
  const image = event.currentTarget;
  image.removeEventListener('click', openPresent);

  const giftOutside = document.querySelector('#gift-outside');
  const giftInside = document.querySelector('#gift-inside');
  giftOutside.classList.add('hidden');
  giftInside.classList.remove('hidden');
}

const image = document.querySelector('#gift-outside img');
image.addEventListener('click', openPresent);
```

No momento certo, trocamos o estado do display,
esondendo uma div e exibindo outra.
(CodePen)

Recapitulando

Existem várias estratégias para atualizar elementos HTML:

1. Mudar o conteúdo dos elementos HTML existentes

- Bom para fazer updates de texto

2. Acrescentar elementos via CreateElement e AppendChild

- Abordagem usada quando é necessário acrescentar um número variável de elementos

3. Trabalhar com diferentes "visões" do HTML

- Bom quando se conhece bem o compartamento da aplicação
- Pode ser usado em conjunto com as soluções 1 e 2

Exercícios

- Considere um conjunto de 100 estrelas, conforme mostrado ao lado
  - O código que gera essas estrelas é exibido a seguir


```
<table id="star_table">
  <tr>
    <td><img src="star_off.gif"><td>
    <td><img src="star_off.gif"><td>
    <td><img src="star_off.gif"><td>
    ...
  <tr>
  </tr>
    <td><img src="star_off.gif"><td>
    <td><img src="star_off.gif"><td>
    <td><img src="star_off.gif"><td>
    ...
  <tr>
  ...
</table>
```

Exercício 1

- Escreva um script para acender uma estrela quando clicada
- Exibir um contador que mostra o número de estrelas acesas
  - https://baldochi.unifei.edu.br/COM222/stars/star_off.gif
  - https://baldochi.unifei.edu.br/COM222/stars/star_on.gif


```
<table id="star_table">
  <tr>
    <td><img src="star_off.gif"><td>
    <td><img src="star_off.gif"><td>
    <td><img src="star_off.gif"><td>
    ...
  <tr>
  </tr>
    <td><img src="star_off.gif"><td>
    <td><img src="star_off.gif"><td>
    <td><img src="star_off.gif"><td>
    ...
  <tr>
  ...
</table>
```

Exercício 2

- Desenvolva uma página que mostre um botão (ou imagem) com o dizer “Acender”. Ao ser clicado, todas as estrela devem ser acesas e o dizer “Acender” deve ser trocado por “Aceso”


```
<table id="star_table">
  <tr>
    <td><img src="star_off.gif"><td>
    <td><img src="star_off.gif"><td>
    <td><img src="star_off.gif"><td>
    ...
  <tr>
  </tr>
    <td><img src="star_off.gif"><td>
    <td><img src="star_off.gif"><td>
    <td><img src="star_off.gif"><td>
    ...
  <tr>
  ...
</table>
```

Exercício 3

- Desenvolva uma página que mostre um botão (ou imagem) com o dizer “Acender”. Ao ser clicado, todas as estrelas devem ser acesas e o dizer “Acender” deve ser trocado por “Apagar”. Ao clicar o botão “Apagar”, todas as estrelas devem ser apagadas e o dizer deve ser trocado para “Acender”.


```
<table id="star_table">
  <tr>
    <td><img src="star_off.gif"><td>
    <td><img src="star_off.gif"><td>
    <td><img src="star_off.gif"><td>
    ...
  <tr>
  </tr>
    <td><img src="star_off.gif"><td>
    <td><img src="star_off.gif"><td>
    <td><img src="star_off.gif"><td>
    ...
  <tr>
  ...
</table>
```