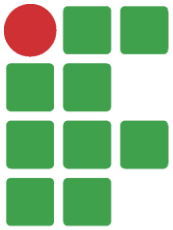
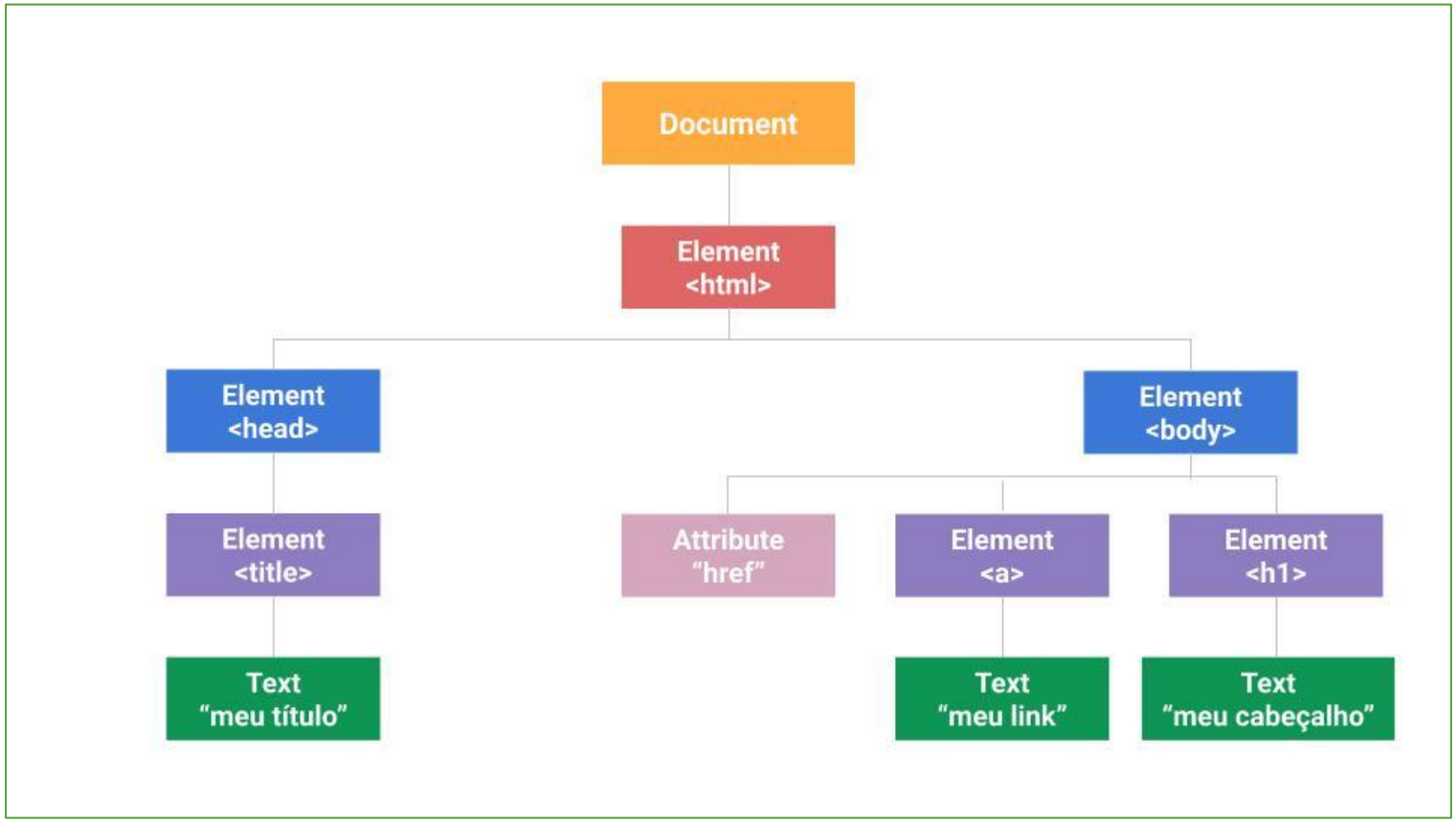


INSTITUTO FEDERAL
Triângulo Mineiro | Campus Patrocínio

LABORATÓRIO DE PROGRAMAÇÃO II

GILBERTO VIANA DE OLIVEIRA

HTML DOM (Document Object Model)



O que é Modelo de Objeto de Documento (DOM)?

- O Modelo de Objeto de Documento (DOM) é uma interface de programação para documentos HTML, XML e SVG.
- O DOM foi criado pela W3C com o objetivo de desenvolver um padrão para linguagens de script para os navegadores.
- Ele fornece uma representação estruturada do documento como uma árvore.
- O DOM define métodos que permitem acesso à árvore, para que eles possam alterar a estrutura, estilo e conteúdo do documento.
- Embora o DOM seja frequentemente acessado usando JavaScript, não é uma parte da linguagem JavaScript.

HTML DOM (Document Object Model)

- Javascript tem o poder para criar conteúdo dinâmico no HTML.
  - JS pode mudar qualquer elemento HTML de uma página.
  - JS pode alterar atributos, CSS estilos, remover HTML existente.
  - JS pode também criar novos elementos HTML em uma página através de eventos.
- Para isso, é necessário enxergar o DOM como um modelo de objeto padrão. Onde:
  - Os elementos HTML serão objetos com propriedades, métodos e eventos.

Olá mundo! Mudando o conteúdo HTML

- Um dos vários métodos Javascript é o *getElementById*().
- Esse método é usado para encontrar um elemento HTML com um **id** especificado.
- A mudança mais simples desse tipo de objeto é fazer diretamente no código executado.
- Ex:

```
No arquivo HTML, insira o trecho de código abaixo.
 <p id=“teste >Hello world</p>
   <script>
        document.getElementById("teste").innerHTML = "Olá
mundo";
   </script>
```

Outra forma de fazer

- Dentro do body no arquivo index.html

```
<body>

    <p id="ola"></p>
    <script src="main.js"></script>

</body>
```

- No arquivo main.js

```
const elementoP = document.getElementById("ola")

elementoP.innerHTML = "Olá mundo!"
```

Criando um elemento HTML

- Em um documento HTML, o método *document.createElement()* cria o elemento HTML especificado por *tagName.*
- Já o método *appendChild()* adiciona um nó ao final da lista de filhos de um nó pai especificado.
- *Sintaxe:*
  - `createElement(tagName)`
  - `var filho = elemento.appendChild(filho);`
- *Exemplo:*

```
const elementoCriado = document.createElement("p")
elementoCriado.innerHTML = `Esse parágrafo foi criado no JS`
document.body.appendChild(elementoCriado)
```

neste exemplo, criamos o elemento com a tag p, adicionamos conteúdo a ele e depois inserimos ele no body do HTML, como filho da tag body

Criando um elemento HTML

- Exemplo 2: Criando elementos filhos de outros elementos

```
const personagens = ["Luke", "Leia", "Han Solo"]
const conteudo = document.createElement("div")
document.body.appendChild(conteudo)
for(let p of personagens){
    const atual = document.createElement("p")
    atual.innerHTML = p
    conteudo.appendChild(atual)
}
```

Nesse código, criamos um array com 3 nomes. Em seguida, criamos um elemento div e adicionamos ele no body. Por fim, percorremos os nomes e adicionamos eles como filhos da div criada. Veja no inspecionar do navegador o resultado.

Remover um elemento do HTML

- A função removeChild() serve para remover um nó filho do DOM. Ela devolve o nó removido.
- Sintaxe:

```
const filhoRemovido = elemento.removeChild(filho);

elemento.removeChild(filho);
```

- O nó filho removido **ainda existe** em memória, mas não é mais parte do DOM. Você pode reutilizar o nó removido mais tarde no seu código por meio da referência filhoRemovido.

Remover um elemento do HTML

- Exemplo:

```
 Este código deve estar dentro do body do arquivo index.html
<div id="conteudo">
        <p>Este P está dentro de um elemento DIV com id conteudo.</p>
   </div>


Este código deve estar no arquivo main.js
const divConteudo = document.getElementById("conteudo")
document.body.removeChild(divConteudo)
Neste código, buscamos o elemento da div com ID conteúdo e armazenamos como um elemento filho. Posteriormente, buscamos no body um filho que seja este nó e o removemos.
```

Outro métodos para manipulação de elementos

- Métodos para selecionar elementos no HTML
  - document.getElementByID(id) - Recupera um elemento pelo ID.
  - document.getElementsByTagName(name) - Recupera um elemento pelo nome.
  - document.getElementsByClassName(name) - Recupera um elemento pelo nome da classe.

- Propriedades e métodos para alterar elementos no HTML
  - element.innerHTML - Esta propriedade obtém ou altera qualquer elemento no HTML, inclusive tags.
  - element.innerText - Esta propriedade permite inserir textos no HTML.
  - element.attribute - Esta propriedade altera o valor de um elemento HTML
  - element.setAttribute(atributo, valor) - Este método altera o valor de um atributo de um elemento HTML.

- Adicionando e excluindo elementos
  - document.write() - Escreve no fluxo de saída do HTML.
  - document.appendChild() - Adiciona um elemento HTML.
  - document.removeChild() - Remove um elemento HTML.
  - document.replaceChild() - Substitui um elemento HTML.
  - document.createElement() - Cria um elemento HTML.

Atividades para fixação

- *Crie um arquivo HTML com a div de id "principal" no arquivo HTML.*
- *Crie e importe um arquivo Javascript chamado main.js. Adicione o CSS do próximo slide a um arquivo CSS externo e link com seu HTML.*
- *Com Javascript:*
- *Adicione 3 elementos filhos do tipo h1, h2 e p, respectivamente à div principal.*
- *Adicione o texto "Bem vindo ao JS" ao elemento h1*
- *Adicione o texto "Criando elementos de forma dinâmica" ao elemento h2*
- *Adicione o texto "Esse é um exemplo clássico de geração de HTML com Javascript" ao p*
- *Adicione a classe “tituloPrincipal” para o h1.*
- *Adicione a classe “tituloSec” para o h2.*
- *Adicione a classe “texto” para a tag p.*

Escreva no caderno o que aconteceu após a execução de cada etapa.

Arquivo CSS e Arquivo HTML devem ter esses conteúdos

```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Atividade 1</title>
    <link rel="stylesheet" href="styles.css">
</head>
<body>

    <div id="principal">

    </div>

  <script src="main.js"></script>
</body>
</html>
```

```css
*{
    margin:0;
}
body{
    margin: auto;
    text-align: center;
}
.tituloPrincipal, .tituloSec, .texto{
    font-family: 'Arial';
}
.tituloPrincipal{
    padding: 10px;
    color: #227326;
}
.tituloSec{
    padding: 15px;
}
```

Atividade 2

- Crie 10 objetos do tipo aluno que contenham nome, email e matéria favorita (leia com prompt).
- Adicione esses objetos de forma dinâmica dentro da seção **main** do arquivo HTML a seguir (próximo slide). Essa adição deve ser feita após a leitura dos dados desses alunos. Você deve criar um card para cada aluno, conforme exemplo abaixo.

HTML e CSS base do exercício anterior

```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Atividade 2</title>
    <link rel="stylesheet" href="styles.css">
</head>
<body>


    <main id="principal">

    </main>

    <script src="main.js"></script>
</body>
</html>
```

```css
#principal{
    display: flex;
    flex-wrap: wrap;
}
div{
    font-family: 'Arial';
    border: 1px blue solid;
    border-radius: 8px;
    text-align: center;
    margin: 4px;
    justify-content: center;
    align-items: center;
    padding: 10px;
}
h1{
    font-size: 24px;
}
h2{
    font-size: 12px;
}
p{
    font-size: 14px;
}
```

Métodos DOM

- Outros métodos comuns de manipulação de elementos:
- *querySelector()*
  - Retorna o primeiro elemento dentro do documento que corresponde ao grupo especificado de seletores.
  - Exemplo – pegar a primeira div no código HTML.

```
var el = document.querySelector("div");
```

- *querySelectorAll()*
  - Exemplo 2 – pegar todos os h1 e armazenar em um objeto, posteriormente no for, imprimir cada um deles

```
var el = document.querySelectorAll("h1");
for(let i of el){
    console.log(i);}
```

Métodos DOM

Exemplos

- `document.querySelector("#demo").innerHTML = "Hello World!"; //muda o texto do elemento com id “demo”.`
- `document.querySelector("div > p");//Seleciona o primeiro elemento p que o pai é uma div`
- `document.querySelector("h3, h4").style.backgroundColor = "red"; //seleciona o primeiro h3 ou h4 que aparecer`

Outros métodos

`querySelector`	CSS-selector
`querySelectorAll`	CSS-selector
`getElementById`	`id`
`getElementsByName`	`name`
`getElementsByTagName`	tag or `'*'`
`getElementsByClassName`	class

Exemplo com *querySelector()*

Assuma esse código dentro do body do arquivo index.html

```
<div id="conteudo-filmes">
        <h2>Trilogia Clássica</h2>
        <p>Star Wars: Episódio IV – Uma Nova Esperança</p>
        <p>Star Wars: Episódio V – O Império Contra-Ataca</p>
        <p>Star Wars: Episorio VI – O retorno de Jedi </p>
    </div>
```

Assuma esse código no arquivo main.js

```
const elemento = document.querySelector("p")
const elementos = document.querySelectorAll("p")
```

Qual a diferença prática dessas duas linhas de código? Como manipular esses elementos?

- Observação: também podemos fazer uma busca por Ids ou classes.
  - Exemplo

```
const elemento = document.querySelector("#conteudo-filmes") //busca o elemento que tem o id conteúdo-filmes
const filmes = document.querySelectorAll(".filme-nome") //busca os elementos que tem a classe filme-nome
```

Atividade – O que esse código faz?

Esse código deve estar no seu index.hml

```
<div id="conteudo-filmes">
        <h2>Trilogia Clássica</h2>
        <p>Star Wars: Episódio IV – Uma Nova Esperança</p>
        <p>Star Wars: Episódio V – O Império Contra-Ataca</p>
        <p>Star Wars: Episódio VI – O retorno de Jedi </p>

 </div>
```

Esse código deve estar no seu main.js

```
const elemento = document.querySelector("#conteudo-filmes")
const paragrafos = elemento.querySelectorAll("p")

for(let p of paragrafos){
    p.classList.add("filme-nome")
}

const nTrilogia = document.createElement("h3")
nTrilogia.innerHTML = "Nota: 9.3"

const pP = document.querySelector("p")
elemento.insertBefore(nTrilogia, pP)
```

Eventos em Javascript

- Eventos são ações ou ocorrências que acontecem no sistema que estamos desenvolvendo, no qual este te alerta sobre essas ações para que você possa responder de alguma forma, se desejado.
- Existem diferentes maneiras de se aplicar eventos aos elementos HTML, são elas:
  - *Inline*
  - Em um arquivo externo, usando um manipulador de eventos

Evento *inline* vs externos

- Podemos adicionar eventos diretamente em um elemento HTML.
- Exemplo:

```
<button onclick="this.innerHTML='Alguém clicou aqui!!'">Clique aqui!</button>
```

- Outros eventos que podem ser usados inline:
  - *onLoad()*
  - *onMouseOver()*
  - Dentre outros: (https://www.w3schools.com/js/js_events.asp)

Manipulador de evento (*event handler*)

- O método *addEventListener()* do JavaScript permite que você configure funções a serem chamadas quando um evento específico acontece, como, por exemplo, quando um usuário clica em um botão.
- Sintaxe:

```
target.addEventListener(event, function, useCapture)
```

- target: o elemento HTML ao qual você deseja adicionar seu manipulador de eventos (event handler).
- event: uma string que especifica o nome do evento.
- function: especifica a função a ser executada quando o evento for detectado.
- useCapture: um valor booleano opcional (verdadeiro ou falso) que especifica se o evento deve ser executado na fase de captura ou de envio aos eventos pais (texto em inglês).

Manipulador de evento (*event handler*)

- No HTML

```
<h1 id="externo">Não clique aqui! 😁</h1>
```

- NO JS

```
const elemento = document.getElementById("externo")

elemento.addEventListener("click", function(){
    elemento.innerHTML = 'Alguém clicou aqui! 😠'

})
```

Mudando o conteúdo HTML

- Adicionando uma função a um botão

```
<p id="p1">Você é diamante!</p>
<button onclick="mudaTexto()">Saiba a verdade</button>
<script>
  function mudaTexto() {
    document.getElementById("p1").innerHTML = "É mentira!";
  }
</script>
```

Métodos DOM

- Além de obter o elemento pelo ID, que é único, é possível obter o elemento que deseja alterar de outras formas.
- Exemplo:

```
let divs = document.getElementsByTagName('div');
console.log(divs.length);
```

- Esse trecho de código busca as divs presentes em uma página, as saídas nos consoles representam, o número de divs que foram resgatas pela função *getElementByTagName();*

Eventos em Javascript

Evento	É disparado…
click	quando é pressionado e liberado o botão primário do mouse, trackpad, etc.
mousemove	sempre que o cursor do mouse se move.
mouseover	quando o cursor do mouse é movido para sobre algum elemento.
mouseout	quando o cursor do mouse se move para fora dos limites de um elemento.
dblclick	quando acontece um clique duplo com o mouse, trackpad, etc.
load	quando todo o documento (DOM e recursos externos como imagens, scripts, etc) está totalmente carregado.
keydown	quando uma tecla no teclado é pressionada.
keyup	quando uma tecla no teclado é liberada (depois de pressionada).
scroll	quando há “rolagem” (scroll) num elemento.

Lista completa de eventos: http://www.w3schools.com/js/js_events.asp

*DOM – *Document Object Model* (Quando uma página é carregada, o navegador cria um DOM.

Lendo dados do HTML

- Podemos construir programas usando valores digitados pelo usuário no HTML.
- Exemplo:
  - Uma página que seja capaz de calcular o IMC de uma pessoa. A leitura deve ser feita usando campos de formulários.

Atividades

- Pesquise, dê a definição e a utilização, com um exemplo, das seguintes funções em JS:
  - removeChild
  - replaceChild
  - insertBefore

Exercício - Crie uma página que:

- Gere o conteúdo “Hoje é dia X/Y de XXXX (pesquise e use a função appendChild)
- Crie um elemento do tipo h1 no body do HTML com a mensagem inicial “msg base”.
- Em Javascript, acesse esse elemento pelo ID e modifique seu conteúdo com base no horário local, sendo:
  - Entre 6 e 11 horas: “Bom dia!”
  - 12 horas: “Hora do Almoço!”
  - 13 as 17: “Boa tarde!”
  - 18 as 23: “Boa noite!”
  - 0 horas: “Não disse que ia dormir mais cedo hoje?”
  - Entre 1 e 5 horas: “Não está na netflix, está?”
- Escreva separadamente o arquivo.html do arquivo.js;
- Para realizar essa atividade, pesquise sobre como fazer para obter a data atual do seu computador em javascript.

Referências

- Mozilla Developer Network: https://developer.mozilla.org/pt-BR/docs/Web/API/Document_Object_Model
- W3Schools: https://www.w3schools.com/js/js_htmldom.asp
- O que é DOM - A árvore de elementos do HTML: https://tableless.github.io/iniciantes/manual/obasico/oquedom.html
- O método addEventListener() - https://www.freecodecamp.org/portuguese/news/o-metodo-addeventlistener-exemplo-de-codigo-com-listener-de-eventos-em-javascript/

Dúvidas?

Para atendimento envie e-mail

gilbertooliveira@iftm.edu.br

MAY THE FORCE BE WITH YOU