



otterwise



# FORMULÁRIO

# Formulário



- Uma das interações mais comuns de um usuário em uma página é o envio de formulários.
- Quando nos cadastramos em um site, realizamos login ou enviamos um feedback, por exemplo, estamos utilizando um formulário.



## Elementos (tags HTML) de formulário

---

- **<form** [atributo=valor]>[conteudo]</form>
  - Utilizado para criar formulários dentro do body da página.
- **<input** [atributo=valor] />
  - Elemento para controle dos dados informados pelo usuário no formulário, pode ser um checkbox, caixa de texto, radio button, etc.
- **<label** [atributo=valor]>[texto\_do\_rotulo]</label>
  - Elemento utilizado para prover uma legenda para o campo de input.
  - Vínculo é dado pelo atributo *for* informando o *id* do input.

# Button



- É um elemento HTML comum em formulários mas não exclusivo, podendo ser encontrado e utilizado fora da tag *form*.
- Com o atributo **type="submit"** envia o formulário quando clicado.
- Exemplo:
  - `<button [atributo=valor]>[texto_botao]</button>`

# Exemplo



```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta http-equiv="X-UA-Compatible" content="IE=edge">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Web Developer</title>
</head>
<body>
  <form>
    <label for="name">Nome:</label>
    <input type="text" name="name" />
    <button type="submit">Enviar</button>
  </form>
</body>
</html>
```



# EXERCÍCIO

## Exercício



Crie uma página HTML com um formulário para cadastro de pessoas. Essa página terá duas seções: uma primeira explicativa, com o título e a descrição da página (textos de sua escolha) e uma segunda com o formulário. O título da página na aba do navegador deve ser o mesmo título da página.



## Exercício



No formulário tente sempre utilizar o tipo do elemento que mais se encaixa no valor que ele irá receber, utilize também outros atributos como placeholder para um resultado final mais robusto. O formulário deve conter os seguintes campos:

- Nome
- Email
- Data de Nascimento
- Linguagem Favorita com as opções JS e Basic
- Botão de enviar



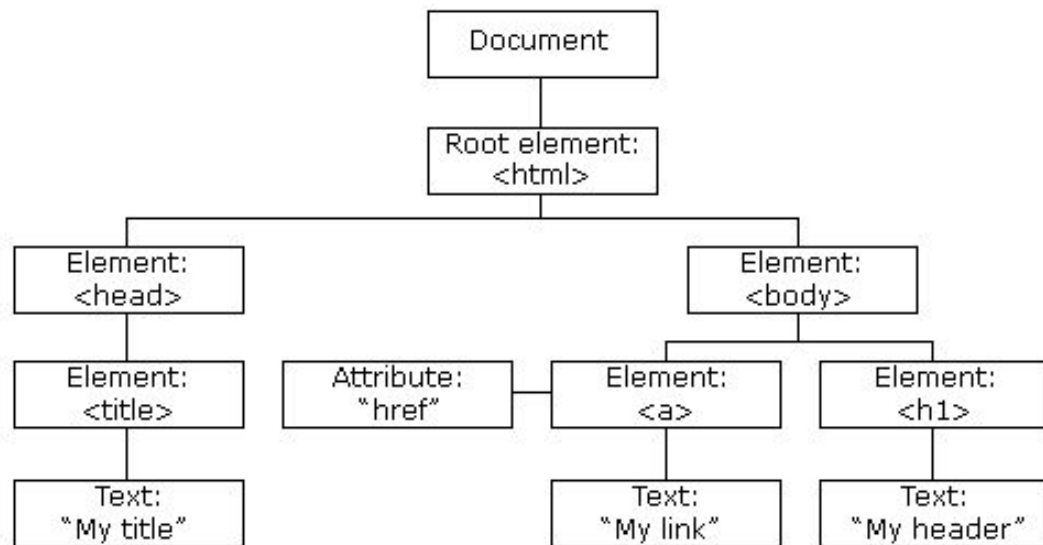
# DOM

# DOM



- **Document Object Model** - Modelo de Objeto de Documento.
- Representação da estrutura e dos elementos de um documento Web (HTML, XML, etc...).
- Através do DOM pode-se acessar os elementos da estrutura (árvore lógica).
- É possível acessar o DOM e modificar conteúdo, estilo e até a própria estrutura da página.
- Eventos podem ser associados ao elementos.
  - Ex.: **Quando** clicar em um botão, uma mensagem deve aparecer na tela.

# DOM



Fonte: [https://www.w3schools.com/js/js\\_htmlDOM.asp](https://www.w3schools.com/js/js_htmlDOM.asp)



# JAVASCRIPT NO HTML

# JavaScript no HTML



- Utilizando a tag *script* pode-se conectar um script JS em um documento HTML.
- Dessa forma, pode-se executar algoritmos que manipulam o DOM para criar e disparar eventos, adicionar, modificar e remover elementos, atributos e estilos, por exemplo.



# MANIPULAÇÃO DO DOM

# Manipulação do DOM




- **window**
  - Instância que representa a janela do navegador, possui vários métodos e atributos como largura da página, altura da página, histórico de navegação, documento exibido, etc...
- **document**
  - Instância que representa a página, possui uma série de atributos e métodos que permitem sua manipulação.



# Atributos de window



- **window.onload**
  - Função que será executada assim que o documento for carregado.
  - Utilizado para sincronizarmos nossos scripts e não precisar colocar os elementos *script* somente no final do documento HTML.
- **window.document**
  - Referência ao documentos carregado na janela do navegador.



```
window.onload = () => {  
  console.log(document)  
}
```

# Atributos de window



- **window.innerHeight**
  - Valor da altura da página carregada no navegador em pixels.
- **window.innerWidth**
  - Valor da largura da página carregada no navegador em pixels.



# Métodos de document

- `document.createElement(element)`
  - Cria e retorna um elemento HTML do tipo passado como argumento.

```
const paragrafo = document.createElement("p")
```

- `document.getElementById(id)`
  - Retorna o elemento com o *id* passado como argumento caso encontre, caso contrário retorna *null*.

```
const imagem = document.getElementById("minha-imagem")
```



# Métodos de document

- `document.getElementsByClassName(name)`
  - Retorna um “array” de elementos que possuem a classe *name*.

```
const containers = document.getElementsByClassName("container")
```

- `document.getElementsByTagName(name)`
  - Retorna um “array” de elementos do tipo *name*.

```
const paragrafos = document.getElementsByTagName("p")
```

# Métodos de element



- *parentElement.removeChild(element)*
  - Remove o elemento passado como argumento que seja filho do elemento *parentElement*.
- *parentElement.appendChild(element)*
  - Adiciona o elemento passado como argumento como último filho do elemento *parentElement*.
- *element.remove()*
  - Remove o elemento.



## Métodos de element

- `element.getAttribute(attributename)`
  - Retorna o valor do atributo *attributename* do elemento passado como parâmetro.
- `element.hasAttribute(attributename)`
  - Retorna **true** se o elemento tem o atributo *attributename* e **false** caso contrário.
- `element.hasAttributes()`
  - Retorna **true** se o elemento tem atributos e **false** caso contrário.

# Métodos de element



- *element.setAttribute(attributename, attributevalue)*
  - Adiciona o atributo no elemento.
- *element.removeAttribute(attributename)*
  - Remove o atributo no elemento.

# Atributos de element

- ***element.innerText***

- Texto do elemento (tanto para acessar quanto para modificar).

```
const paragrafo = document.getElementById("meu-texto")
paragrafo.innerText = "Novo texto"
```

- ***element.innerHTML***

- HTML do elemento (tanto para acessar quanto para modificar).

```
const paragrafo = document.getElementById("meu-texto")
paragrafo.innerHTML = "<p>oi</p>"
```



# Métodos de Seleção de Elementos




- *element.querySelector(selector)*
  - Retorna o **primeiro** elemento que satisfaz o seletor especificado.
  - Utilizado o mesmo tipo de seletor do CSS.
  - Pode ser utilizado com **document** também.
  
- *element.querySelectorAll(selector)*
  - Retorna um “array” com elementos que satisfaçam o seletor especificado.
  - Utilizado o mesmo tipo de seletor do CSS.
  - Pode ser utilizado com **document** também.

# Métodos de Seleção de Elementos



```
// Pega o primeiro elemento com o id "minha-imagem"  
const minhaImagem = document.querySelector("#minha-imagem")
```



```
// Pega todos os elementos com a classe "container"  
const containers = document.querySelector(".container")
```

# Fontes



- [https://www.w3schools.com/js/js\\_htmldom\\_document.asp](https://www.w3schools.com/js/js_htmldom_document.asp)
- [https://www.w3schools.com/js/js\\_htmldom\\_elements.asp](https://www.w3schools.com/js/js_htmldom_elements.asp)
- [https://www.w3schools.com/js/js\\_htmldom\\_html.asp](https://www.w3schools.com/js/js_htmldom_html.asp)



# EXERCÍCIOS

## Exercícios

A thick blue line starts from the top right corner and curves downwards and to the left, ending near the top right of the slide.

- Crie dois parágrafos na tela e coloque um texto qualquer. Via javascript mude o texto apenas do segundo parágrafo para outro texto qualquer e exclua o primeiro parágrafo.



# EVENTOS

# Eventos



- Podemos executar JS quando um evento ocorrer no HTML.
- Esse evento pode ser um clique em um elemento, interação com o mouse, quando o valor do elemento mudar, etc...
- Podemos atribuir uma função JS à um evento como um atributo ou utilizando o método de elemento **addEventListener**.

# Métodos



- *element.addEventListener(event, function)*
  - Adiciona uma função para ser executada quando o evento especificado ocorrer.
  - *event*: o nome do evento que queremos que nosso listener observe
  - *function*: função de call com o resultado do evento. Recebe um objeto event como parâmetro
- *element.click()*
  - Simula um clique no elemento.



# Métodos



```
const button = document.querySelector('#meu-botao')  
  
button.addEventListener("click", function (event) {  
    console.log("clicou")  
});
```

## Fontes



- [https://www.w3schools.com/js/js\\_htmlDOM\\_events.asp](https://www.w3schools.com/js/js_htmlDOM_events.asp)
- [https://www.w3schools.com/js/js\\_htmlDOM\\_eventlistener.asp](https://www.w3schools.com/js/js_htmlDOM_eventlistener.asp)

## Exemplo



Criar um formulário que cadastra um post na fake API [JSONPlaceholder - Free Fake REST API](#).



# EXERCÍCIOS

## Exercício 1



Crie uma página que calcula a operação selecionada entre dois números e mostre na página o resultado.

No início da página deve haver um título e uma descrição da página.

O formulário possui três campos e um botão de enviar. Os dois primeiros campos recebem números e o terceiro campo é um select com as opções: Dividir, Somar, Subtrair e Multiplicar.

Após o formulário deve haver um elemento mostrando o resultado.

## Exercício 2



Faça uma página com uma calculadora funcional com as operações de multiplicação, soma, subtração e divisão e funções de limpar e resultado.