

UNIDADE III

7. JAVASCRIPT

7.1. O que é o JavaScript

A linguagem de programação JavaScript, como seu próprio nome diz, é uma linguagem scripting. Normalmente uma linguagem scripting se define por uma linguagem de programação que disponibiliza ao programador a possibilidade de controle de aplicações de terceiros (ERIC e ROBSON, 2016).

Em JavaScript, pode-se efetuar o controle de comportamentos dos navegadores por meio de pedaços de código que são enviados na página Html. Além disso, podemos dizer que o JavaScript é uma linguagem de script que incorporado nos tag's Html, permite adicionar efeitos de apresentação e interatividade em páginas (BALDUINO, 2013).

Outra característica comum nas linguagens de scripting é que normalmente elas são linguagens interpretadas, ou seja, não dependem de compilação para serem executadas. Estes scripts serão interpretados pelo próprio browser sem fazer apelo aos recursos de um servidor (ERIC e ROBSON, 2016).

O Javascript surgiu através da empresa Netscape com nome Livescript. No final de 1995 foi comprado pela empresa Sun (criadora do Java), devido a isso recebeu o nome ao qual o conhecemos até hoje. Javascript não é próprio do browser Netscape. Microsoft e outros fornecedores também adotaram o Javascript. Vale lembrar que Javascript não é Java (BALDUINO, 2013).

A linguagem de programação JavaScript armazena uma grande vantagem em relação a muitas outras utilizadas no mercado: Ser tolerante a erros. Para que o navegador diferencie um código Html de um JavaScript, utilizamos a tag <script> (BALDUINO, 2013).

A tag <script> pode ser declarada dentro da tag <head> ou mesmo na tag <body>, porém devemos nos atentar, pois devido a ser uma linguagem de programação interpretada o código é lido pelo browser de forma imediata (ERIC e ROBSON, 2016).

7.2. Iniciando com o JavaScript

Para testarmos nosso primeiro JavaScript, vamos criar um novo projeto, para isso crie uma pasta chamada projeto_js, conforme imagem a seguir:

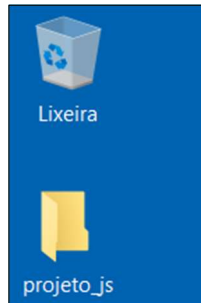


Figura 269. Pasta para armazenar o projeto JavaScript [Fonte: Próprio Autor]

<Lembrete início>

Os conhecimentos adquiridos nas Unidades I e II sobre Html e Css são fundamentais para o entendimento do JavaScript, pois o JavaScript é executado dentro de páginas Html.

<Lembrete fim>

Agora iremos criar um arquivo chamado index.html dentro da pasta projeto_js que acabamos de criar.

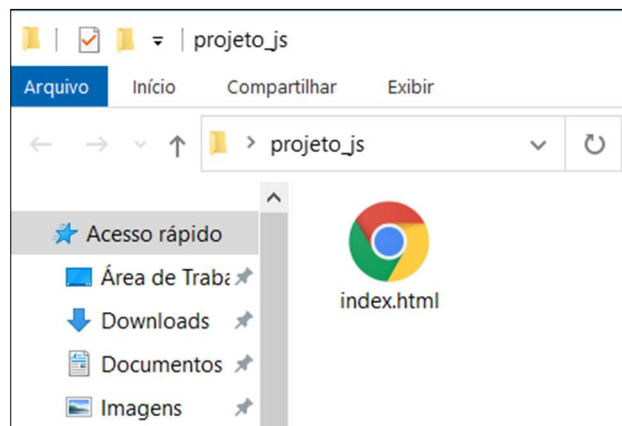


Figura 270. Arquivo index.html dentro da pasta do projeto [Fonte: Próprio Autor]

Vamos agora editar o arquivo index.html, inserindo nele a estrutura padrão de uma página html.

```
<> index.html X
C: > Users > prof- > Desktop > projeto_js > <> index.html >
1  <!DOCTYPE html>
2  <html lang="pt-br">
3      <head>
4          <meta charset="utf-8">
5          <title>JavaScript</title>
6      </head>
7      <body>
8          <h1>JavaScript</h1>
9          <h2>Meu primeiro exemplo</h2>
10     </body>
11 </html>
```

Figura 271. Arquivo index.html com estrutura básica criada [Fonte: Próprio Autor]

Agora, dentro do <head> iremos inserir nossa primeira tag <script> e dentro dela um evento do tipo alert em JavaScript:

```
6  <script>
7      alert("Olá Mundo!!!");
8  </script>
```

Figura 272. Script dentro da tag <head> [Fonte: Próprio Autor]

<Observação início>

O comando alert é responsável por emitir uma mensagem de alerta no navegador. No exemplo emitiremos um alerta informando Olá Mundo!!!.

<Observação fim>

Agora executando a página no navegador, iremos obter o seguinte resultado:

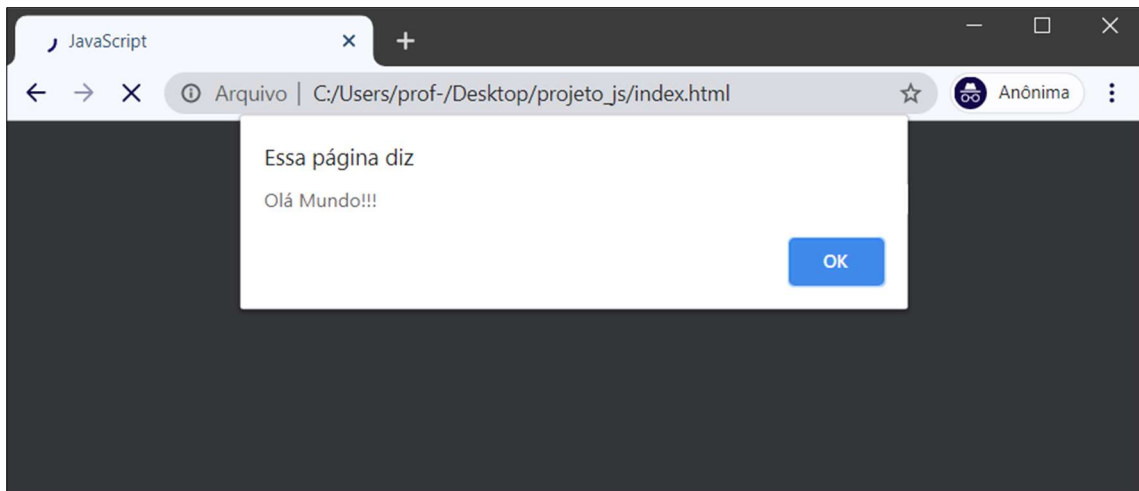


Figura 273. Pagina em execução no navegador. [Fonte: Próprio Autor]

Após clicar no botão OK.



Figura 274. Pagina em execução no navegador após clique no botão OK.
[Fonte: Próprio Autor]

Podemos notar que o Script executado faz a pausa no processamento da página, pois foi inserido no `<head>`, nesse caso seria mais interessante inserir o mesmo dentro da tag `<body>`.

Vamos remover o script do `<head>` e coloca-lo dentro do `<body>`.



```
<> index.html X
C: > Users > prof- > Desktop > projeto_js > <> index.html >
1  <!DOCTYPE html>
2  <html lang="pt-br">
3      <head>
4          <meta charset="utf-8">
5          <title>JavaScript</title>
6      </head>
7      <body>
8          <h1>JavaScript</h1>
9          <h2>Meu primeiro exemplo</h2>
10         <script>
11             alert("Olá Mundo!!!");
12         </script>
13     </body>
14 </html>
```

Figura 275. Página index.html com tag <script> dentro do <body> [Fonte: Próprio Autor]

7.3. JavaScript em arquivo externo

Da mesma forma que aprendemos com o CSS, o JavaScript pode ser executado também em um arquivo externo. Imagine uma situação em que um mesmo script precise ser executado em outras páginas.

Inicialmente, crie uma pasta chamada js dentro da pasta inicial do nosso projeto, lembrando que js é uma abreviação de JavaScript e esse nome foi dado para organizar melhor o projeto, portanto, podendo a seu critério ser atribuído outro nome a essa pasta.

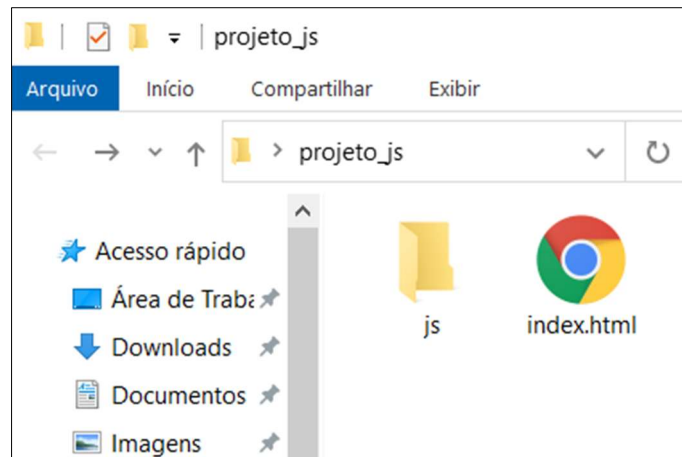


Figura 276. Pasta js criada dentro da pasta do projeto [Fonte: Próprio Autor]

Agora, vamos criar um arquivo de JavaScript. Para isso abra o Microsoft Visual Studio Code, clique em File / New File:

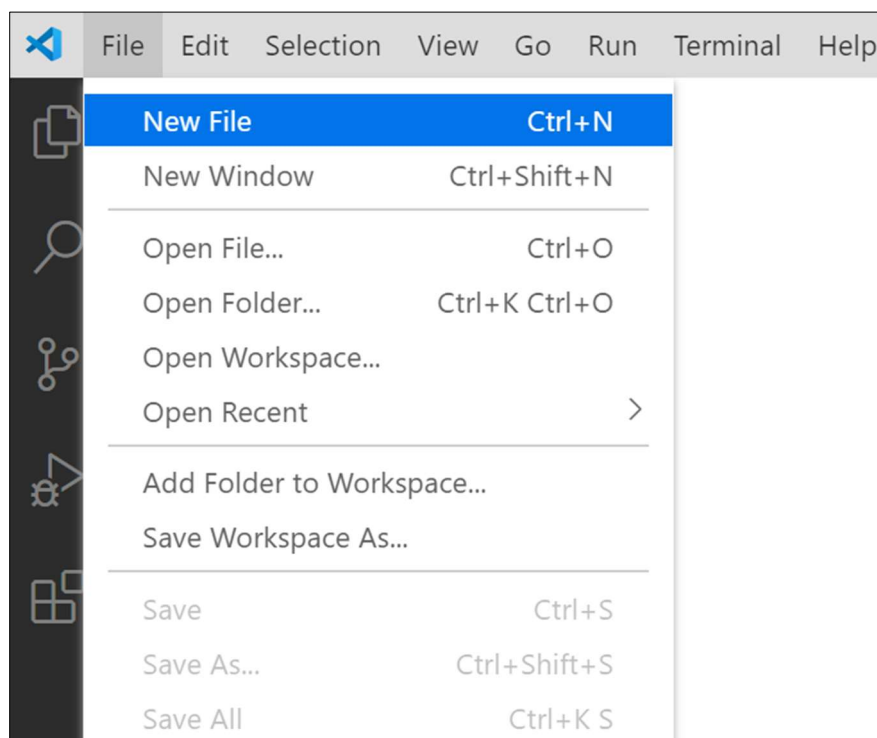


Figura 277. Criação de Novo Arquivo no Microsoft Visual Studio Code [Fonte: Próprio Autor]

Após a abertura do arquivo que foi criado, clique novamente na opção File e selecione a opção Save as:

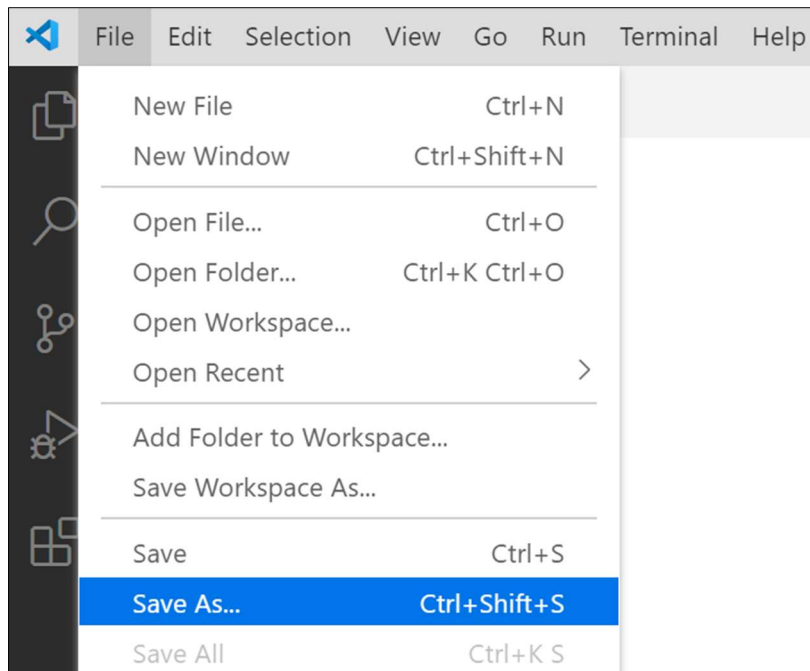


Figura 278. Salvamento de Novo Arquivo no Microsoft Visual Studio Code
[Fonte: Próprio Autor]

Agora navegue até a pasta js do projeto. Na opção Nome escreva scripts.js e na opção Tipo selecione JavaScript, conforme imagem a seguir. Ao concluir clique em Salvar.

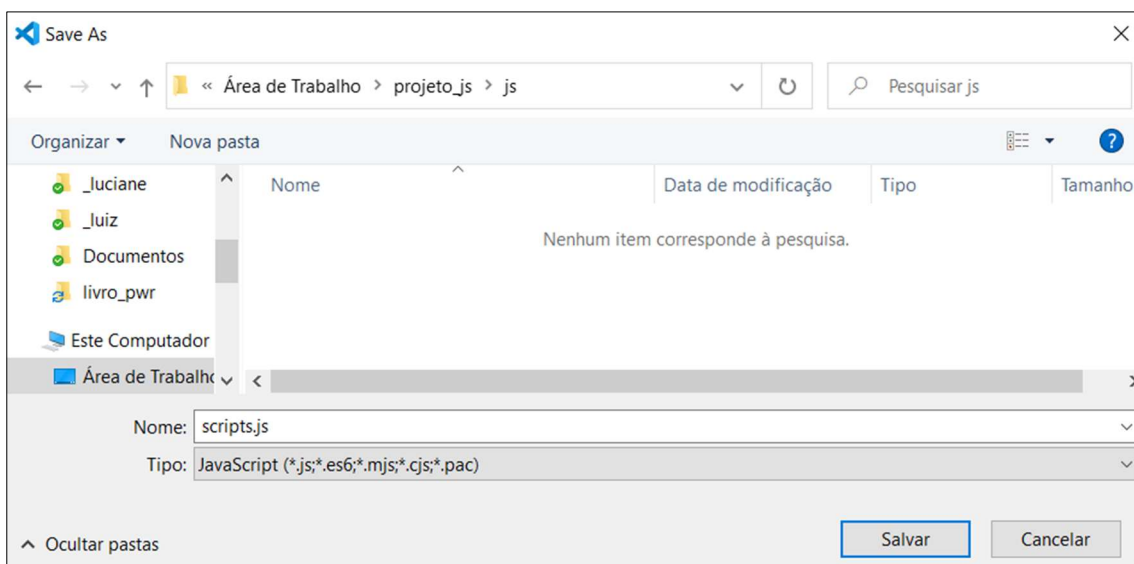


Figura 279. Opções de salvamento do arquivo no Microsoft Visual Studio Code
[Fonte: Próprio Autor]

Vamos agora editar o arquivo scripts.js que acabamos de criar, inserindo o seguinte código no mesmo:

```
JS scripts.js  X
C: > Users > prof- > Desktop > projeto_js > js > JS scripts.js
1  alert("Olá Mundo!!! Agora, utilizando um arquivo externo");
2
```

Figura 280. Arquivo scripts.js com primeiro código em JavaScript [Fonte: Próprio Autor]

Já criamos o arquivo com o código JavaScript, agora iremos criar uma página Html. Primeiramente crie uma pasta páginas sem acento dentro do projeto. Logo em seguida crie uma página chamada exemplo1.html dentro da pasta páginas.

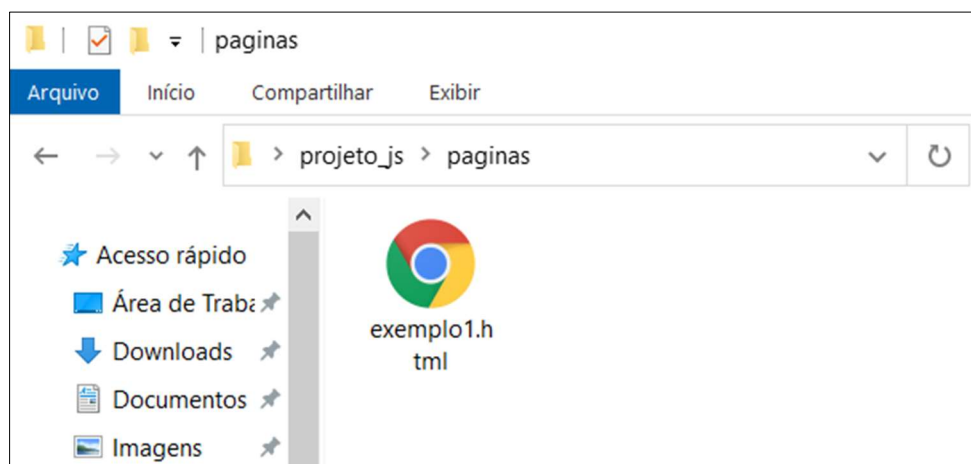


Figura 281. Arquivo exemplo1.html criado dentro da pasta paginas do projeto [Fonte: Próprio Autor]

Agora edita o arquivo exemplo1.html conforme código abaixo. Repare que iremos incorporar/vincular nossa página Html ao arquivo de scripts (js/scripts.js) que criamos, da mesma forma que fizemos nas unidades anteriores do nosso livro o vínculo com o CSS.



```
<> exemplo1.html X
C: > Users > prof- > Desktop > projeto_js > paginas > <> exemplo1.html > ...
1  <!DOCTYPE html>
2  <html lang="pt-br">
3      <head>
4          <meta charset="utf-8">
5          <title>Pagina exemplo1</title>
6          <script type="text/javascript" src="../js/scripts.js"></script>
7      </head>
8      <body>
9          <h1>JavaScript</h1>
10         <h2>Utilizando um arquivo js externo</h2>
11     </body>
12 </html>
```

Figura 282. Arquivo exemplo1.html editado no Microsoft Visual Studio Code
[Fonte: Próprio Autor]

Devido a colocarmos o script em arquivo externo podemos fazer o reaproveitamento de funcionalidades em diversas páginas. Ao inicializar a página, será exibido a seguinte mensagem:

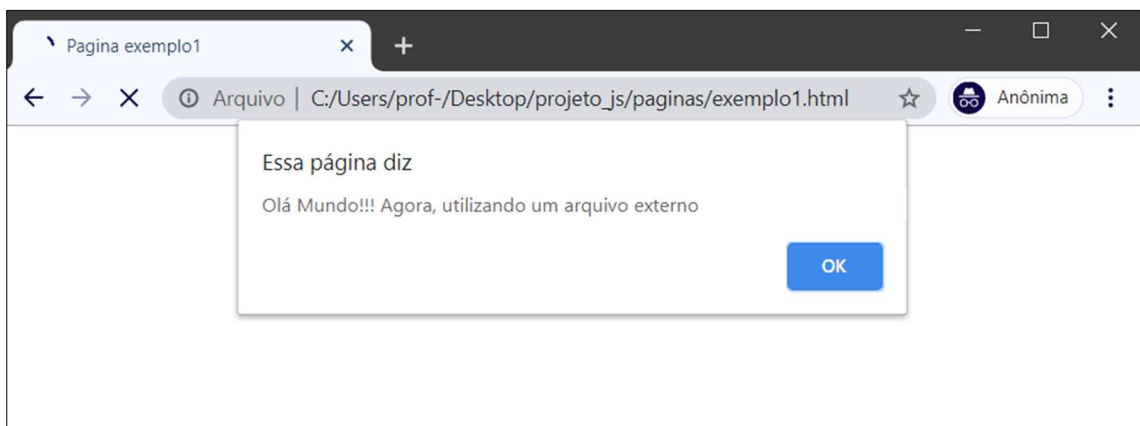


Figura 283. Arquivo exemplo1.html executado no navegador [Fonte: Próprio Autor]

E em seguida, a página será carregada:



Figura 284. Arquivo exemplo1.html executado no navegador após clique no botão OK [Fonte: Próprio Autor]

A seguir, aprenderemos três conceitos muito importantes para o desenvolvimento de códigos em JavaScript. São os Eventos, as Funções e o DOM.

7.4. Eventos

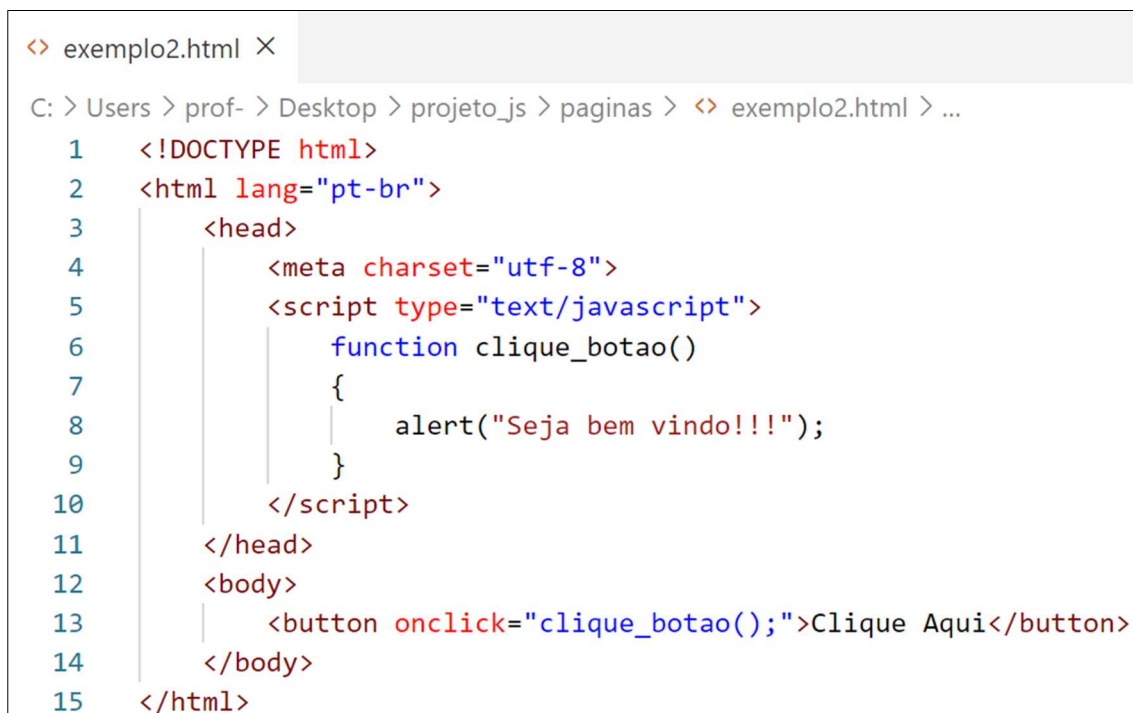
No JavaScript podemos utilizar vários eventos em diversos elementos para interação com o usuário, abaixo listamos os principais:

- `oninput`: ocorre quando um input ocorreu alteração.
- `onclick`: evento de clique no mouse.
- `ondblclick`: evento de dois cliques no mouse.
- `onmousemove`: evento quando o mouse é mexido.
- `onmousedown`: evento quando é apertado algum botão do mouse.
- `onmouseup`: quando o botão do mouse é solto.
- `onkeypress`: evento quando uma tecla é pressionada e solta.
- `onkeydown`: evento quando uma tecla é pressionada.
- `onkeyup`: evento quando solta uma tecla.
- `onblur`: evento quando um elemento deixa de ter o foco.
- `onfocus`: evento quando um elemtno passa a ter o foco.
- `onchange`: evento de quando um campo texto tem seu valor modificado.
- `onload`: evento de quando uma pagina é atualizada.
- `onunload`: evento de quando uma página é fechada.
- `onsubmit`: evento de disparo antes de envio do formulário.

7.5. Função

Uma função é funciona como um sub-programa dentro do nosso programa. Em estudos de algoritmos funções/procedimentos são um conjunto de instruções que executa uma ou várias tarefas. Essas tarefas podem ser desde um cálculo até uma específica, por exemplo, enviar uma mensagem de alerta ao usuário (ERIC e ROBSON, 2016).

No exemplo a seguir, criaremos dentro da pasta páginas do nosso projeto uma página Html chamada exemplo2.html e incluiremos um botão. Ao clicar neste botão (evento onclick) será emitido um alerta através de uma função no qual iremos criar (clique_botão) (BALDUINO, 2013).



```
<> exemplo2.html X
C: > Users > prof- > Desktop > projeto_js > paginas > <> exemplo2.html > ...
1  <!DOCTYPE html>
2  <html lang="pt-br">
3      <head>
4          <meta charset="utf-8">
5          <script type="text/javascript">
6              function clique_botao()
7              {
8                  alert("Seja bem vindo!!!");
9              }
10         </script>
11     </head>
12     <body>
13         <button onclick="clique_botao();">Clique Aqui</button>
14     </body>
15 </html>
```

Figura 285. Arquivo exemplo2.html [Fonte: Próprio Autor]

Ao executar a página teremos o seguinte resultado:

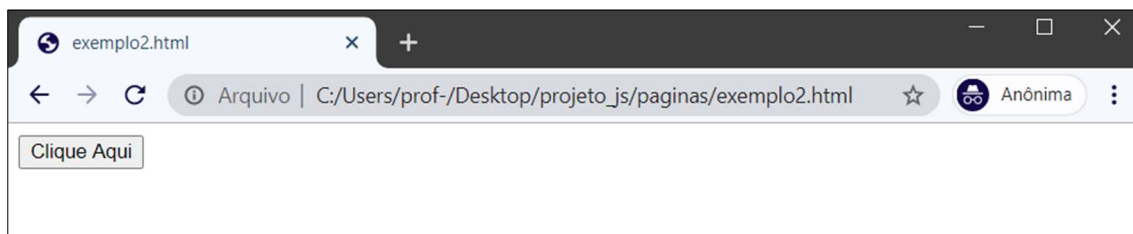


Figura 286. Pagina exemplo2.html em execução no navegador [Fonte: Próprio Autor]

Ao clicar no botão o seguinte resultado será apresentado:

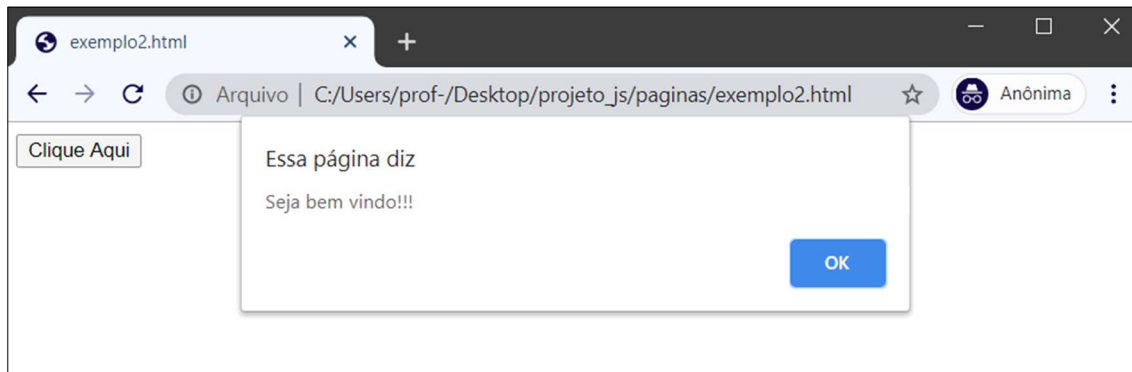


Figura 287. Pagina exemplo2.html executada após clique no botão [Fonte: Próprio Autor]

7.6. DOM

Já estudamos muito o HTML, criamos nossas primeiras tags, aprendemos CSS, fizemos formulários e páginas bonitas, porém chegou o momento de avançarmos um pouco mais. Precisamos inserir vida às nossas páginas, portanto, movimentos. Ao clicar em um botão o mesmo resultará em um evento, para isso usaremos o DOM, que foi criado pelo W3C para representar como os navegadores organizam as marcações HTML, XHTML e XML.

7.6.1. Métodos

O DOM é composto por diversos métodos. Esses métodos são responsáveis por fazer a ligação entre os elementos e os eventos. A seguir iremos demonstrar os métodos mais utilizados e pra que eles servem:

7.6.1.1. getElementById()

O método `getElementById`, como o seu próprio nome diz, retorna o elemento que contém o nome do ID informado por parâmetro. Devido aos ID's normalmente serem únicos, esse método irá pegar apenas um elemento.

7.6.1.2. InnerHTML

Para fazer alterações no HTML através do JavaScript, fazemos o uso do `.innerHTML`. Além de realizar essa troca o `.innerHTML` pode acessar só o conteúdo da tag selecionada.

Vamos agora criar uma nova página dentro da nossa pasta páginas com o nome exemplo3.html, conforme imagem a seguir:



```
<> exemplo3.html X
C: > Users > prof- > Desktop > projeto_js > paginas > <> exemplo3.html
1  <!DOCTYPE html>
2  <html lang="pt-br">
3      <body>
4          <p id="demo"></p>
5      </body>
6  </html>
```

Figura 288. Pagina exemplo3.html [Fonte: Próprio Autor]

Este parágrafo não terá nenhum conteúdo, portanto, ao executar nossa página, ela ficará toda em branco conforme imagem a seguir:

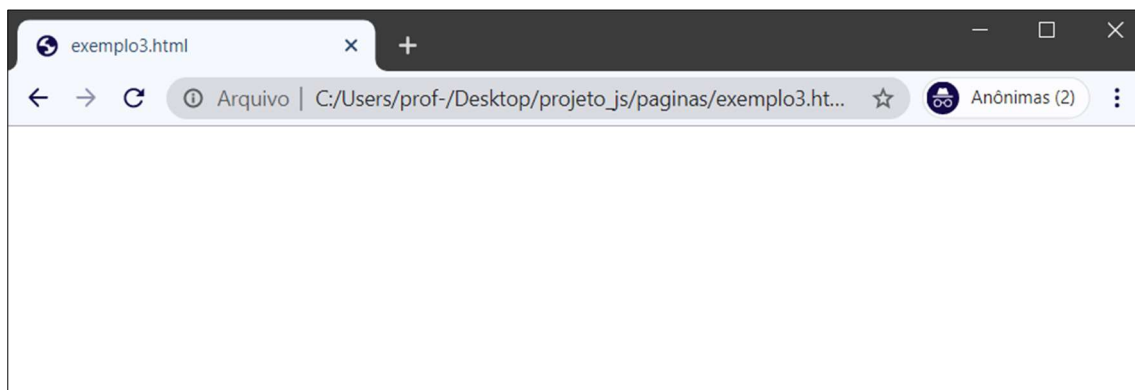
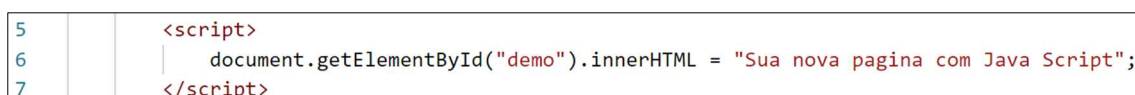


Figura 289. Pagina exemplo3.html em execução no navegador [Fonte: Próprio Autor]

Agora, definiremos um conteúdo para o parágrafo através do seguinte código Java Script.



```
5  <script>
6      document.getElementById("demo").innerHTML = "Sua nova pagina com Java Script";
7  </script>
```

Figura 290. Script dentro da pagina exemplo3.html[Fonte: Próprio Autor]

Ao executar a página no navegador teremos o seguinte resultado:

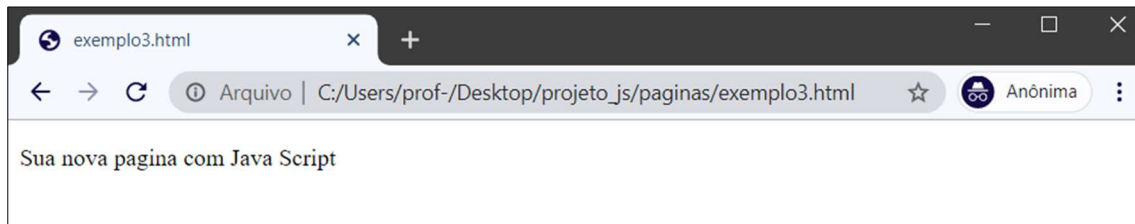


Figura 291. Pagina exemplo3.html em execução após a inserção do script
[Fonte: Próprio Autor]

No nosso próximo exemplo, criaremos uma página chamada exemplo4.html dentro da pasta páginas do nosso projeto. Iremos inserir dois parágrafos, no primeiro iremos inserir um evento onclick e definiremos um id para o segundo conforme imagem a seguir.



Figura 292. Pagina exemplo4.html [Fonte: Próprio Autor]

Como resultado, teremos a seguinte página. Faça um teste clicando duas vezes sobre o parágrafo. Repare que ao clicar duas vezes, nenhuma ação ou mudança ocorrerá em nossa página, pois ainda não implementamos nosso código Java Script.

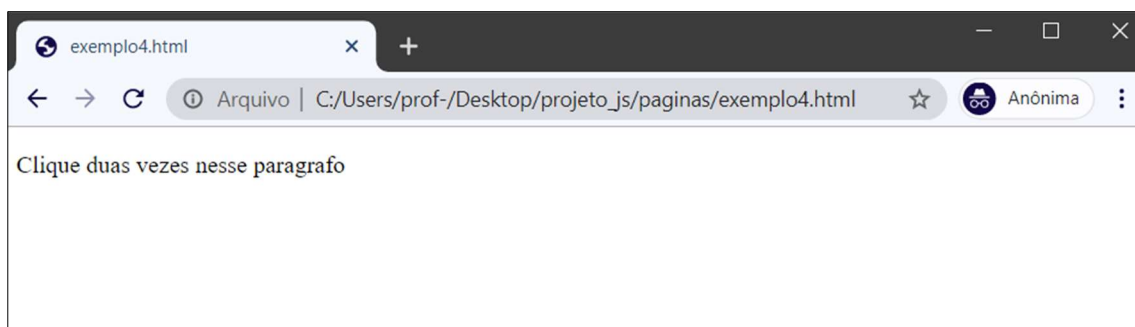


Figura 293. Pagina exemplo4.html em execução no navegador [Fonte: Próprio Autor]

Dando sequência, complemente nosso código, incluindo o seguinte código JavaScript conforme imagem a seguir. Repare que nesse caso, o comando JavaScript foi definido dentro do <body> da página. Isso foi proposital, pois como o código HTML é interpretado pelo browser, o conteúdo do parágrafo (id="demo") é modificado automaticamente após execução deste trecho da página.

```
7      <script>
8      |     function myFunction(){
9      |         document.getElementById("demo").innerHTML = "Olá Mundo!!!";
10     |     }
11     </script>
```

Figura 294. Pagina exemplo4.html utilizando getElementById. [Fonte: Próprio Autor]

Como resultado, teremos a frase Olá Mundo!!! preenchida automaticamente em nosso parágrafo com id="demo" após o evento de clique do mouse no parágrafo.

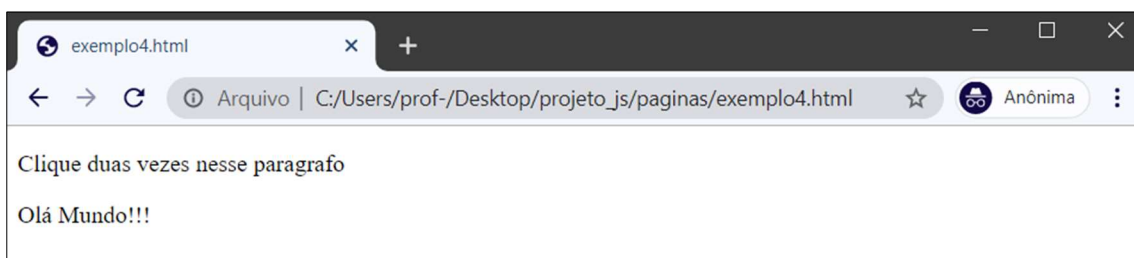


Figura 295. Pagina exemplo4.html em execução no navegador utilizando getElementById. [Fonte: Próprio Autor]

Agora, iremos fazer uma nova implementação utilizando o evento “Clique Duplo” (ondblclick). Para isso crie uma nova página chamada exemplo5.html e digite o código conforme imagem a seguir:

```

<> exemplo5.html X
C: > Users > prof- > Desktop > projeto_js > paginas > <> exemplo5.html > ...
1  <!DOCTYPE html>
2  <html lang="pt-br">
3      <body>
4          <p>Um novo exemplo de como implementar o "ondblclick".</p>
5          <p id="demo" ondblclick="myFunction()">Clique duas vezes aqui</p>
6          <script>
7              function myFunction(){
8                  document.getElementById("demo").innerHTML = "Voce clicou duas vezes aqui!!!";
9              }
10         </script>
11     </body>
12 </html>

```

Figura 296. Pagina exemplo5.html implementando ondblclick. [Fonte: Próprio Autor]

Ao inicializar nossa página, teremos o seguinte resultado:

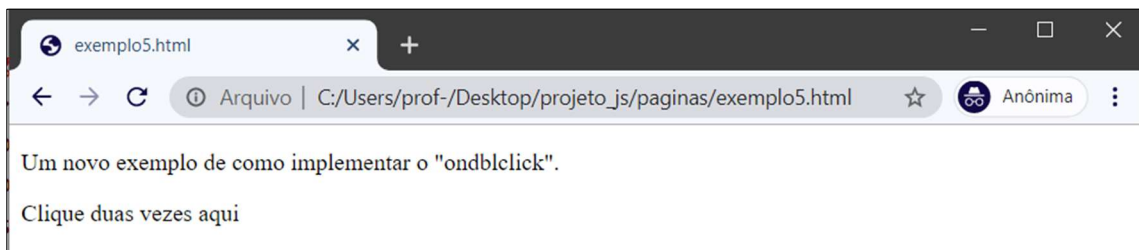


Figura 297. Pagina exemplo5.html em execução no navegador. [Fonte: Próprio Autor]

Ao clicar duas vezes sobre o parágrafo, teremos o seguinte resultado:

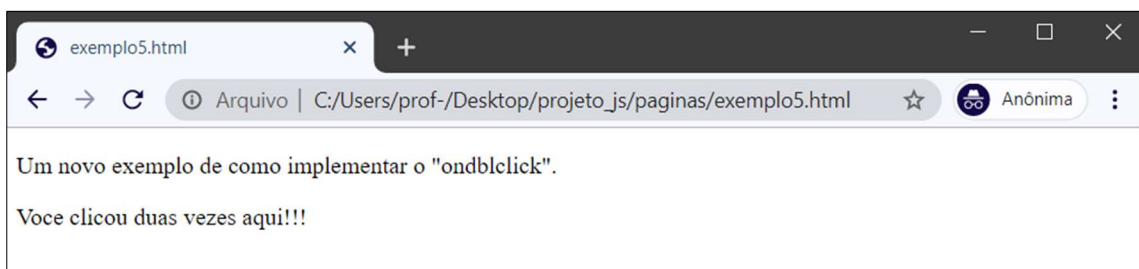


Figura 298. Pagina exemplo5.html em execução no navegador após clicar duas vezes no parágrafo. [Fonte: Próprio Autor]

Agora, implementaremos os eventos onmousedown e mouseUp. Vamos criar uma página dentro da pasta páginas chamada exemplo6.html. Em seguida digite o seguinte código:


```

<> exemplo6.html X
C: > Users > prof- > Desktop > projeto_js > paginas > <> exemplo6.html > ...
1  <!DOCTYPE html>
2  <html lang="pt-br">
3      <body>
4          <p id="myP" onmousedown="mouseDown()" onmouseup="mouseUp()">
5              Clique neste texto e mantenha o clique pressionado. Repare que o texto mudará de cor Preto
6              para Vermelho. Ao soltar o clique, o texto ficará na cor Verde.
7          </p>
8          <script>
9              function mouseDown(){
10                 document.getElementById("myP").style.color = "red";
11             }
12
13             function mouseUp(){
14                 document.getElementById("myP").style.color = "green";
15             }
16         </script>
17     </body>
18 </html>

```

Figura 299. Implementação dos eventos mouseDown e MouseUp na página exemplo6.html [Fonte: Próprio Autor]

Como resultado, teremos. Página inicial:



Figura 300. Página exemplo6.html em execução no navegador [Fonte: Próprio Autor]

Repare a mudança da cor da fonte ao manter o clique pressionado (ficará vermelho):

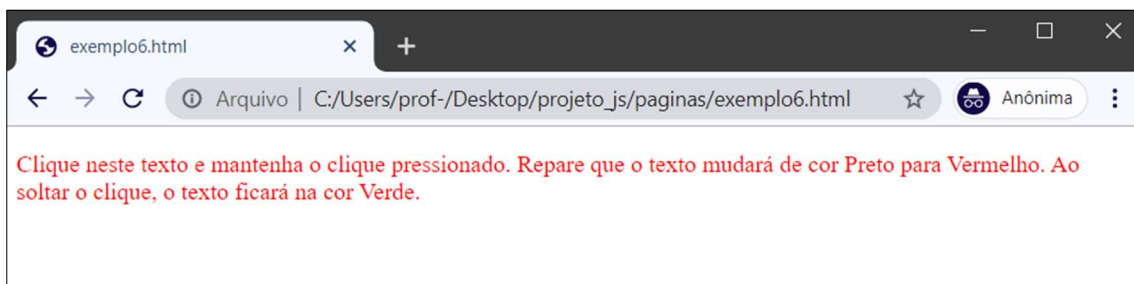


Figura 301. Página exemplo6.html em execução no navegador após clique longo com o mouse no parágrafo. [Fonte: Próprio Autor]

Por fim, ao soltar o clique do mouse, a cor do texto se modificará novamente, ficando verde conforme imagem a seguir:

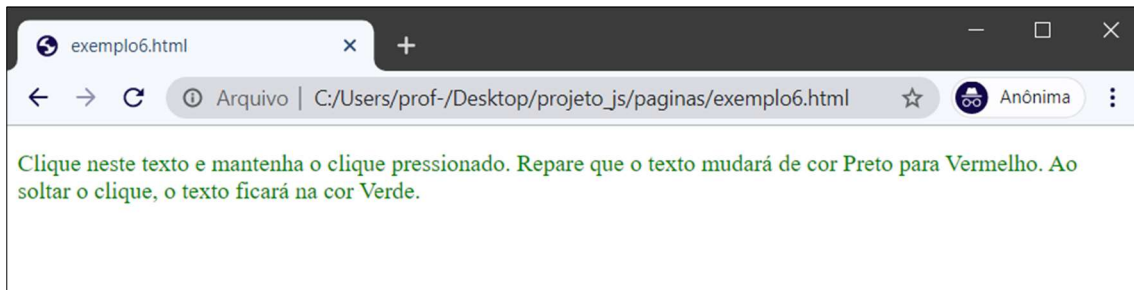


Figura 302. Página exemplo6.html em execução no navegador após soltar clique longo com o mouse no parágrafo. [Fonte: Próprio Autor]

No exemplo a seguir, iremos criar dentro da pasta páginas do projeto uma página chamada exemplo7.html. Dentro da página iremos implementar uma função para modificar as propriedades de um elemento, ou seja, criaremos uma função que irá modificar o tamanho, a cor da fonte e o background de um parágrafo. Implemente o código conforme imagem a seguir:

```
<> exemplo7.html X
C: > Users > prof- > Desktop > projeto_js > paginas > <> exemplo7.html > ...
1  <!DOCTYPE html>
2  <html lang="pt-br">
3      <body>
4          <h1>Meu novo JavaScript</h1>
5          <p id="demo">Mudando as propriedades de fonte via Java Script</p>
6
7          <script>
8              function myFunction(){
9                  document.getElementById("demo").style.fontSize = "25px";
10                 document.getElementById("demo").style.color = "red";
11                 document.getElementById("demo").style.backgroundColor = "yellow";
12             }
13         </script>
14
15         <button type="button" onclick="myFunction()">Clique Aqui!!!</button>
16     </body>
17 </html>
```

Figura 303. Customização da fonte através do JavaScript na página exemplo7.html [Fonte: Próprio Autor]

Ao executar nossa página, teremos o seguinte resultado:

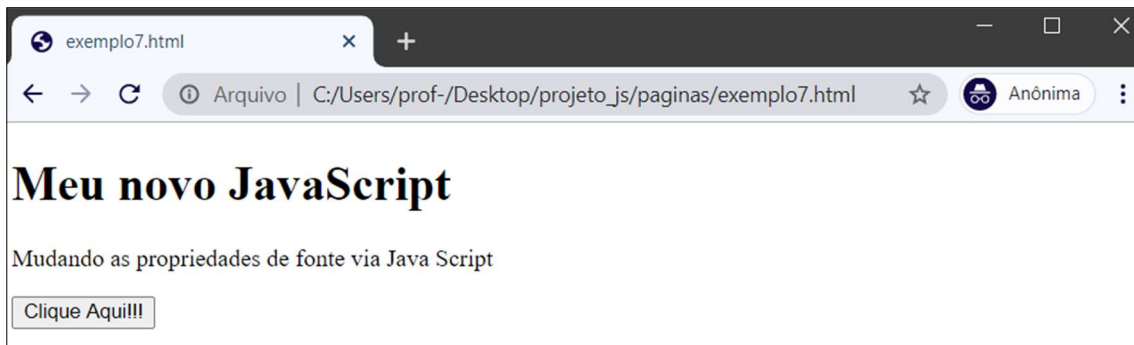


Figura 304. Página exemplo7.html em execução no navegador. [Fonte: Próprio Autor]

Após clicar no botão Clique Aqui! teremos o seguinte o resultado:



Figura 305. Página exemplo7.html em execução no navegador após clique no botão. [Fonte: Próprio Autor]

No próximo exemplo, iremos criar uma página chamada exemplo8.html na pasta páginas do nosso projeto. Essa página irá realizar a soma de dois valores. Para isso, criaremos um formulário no qual receberá o número 01 e o número 02 e apresentará o resultado no campo Resultado. Para isso, implemente o seguinte código:

```
<> exemplo8.html X
C: > Users > prof- > Desktop > projeto_js > paginas > <> exemplo8.html > ...
1  <!DOCTYPE html>
2  <html lang="pt-br">
3      <head>
4          <meta charset="utf-8">
5      </head>
6      <body>
7          <form name="meu_form">
8              <h1>Entre com os numeros a serem somados</h1>
9              <p>
10                 <label for="nome">Numero 1 : </label>
11                 <input type="text" id="numero1" placeholder="Número 1" name="numero1">
12             </p>
13
14             <p>
15                 <label for="nome">Numero 2 : </label>
16                 <input type="text" id="numero2" placeholder="Número 2" name="numero2">
17             </p>
18
19             <p>
20                 <label for="nome">Resultado : </label>
21                 <input type="text" id="resultado" placeholder="Resultado" name="resultado">
22             </p>
23             <p>
24                 <input type="button" value="Calcular" onclick="calculadora()">
25             </p>
26          </form>
27      </body>
28  </html>
```

Figura 306. Criação de formulário na página exemplo8.html [Fonte: Próprio Autor]

Agora, faça um teste e verifique que nenhum cálculo será realizado, pois ainda não implementamos nossa função para a realização do cálculo.



Figura 307. Página exemplo8.html em execução no navegador. [Fonte: Próprio Autor]

Agora, vamos implementar a nossa função para que seja realizado a soma dos dois valores e seja exibido o resultado no campo Resultado. Para isso, vamos implementar o seguinte código:

```
5      <script type="text/javascript">
6      function calculadora(){
7          document.getElementById('resultado').value = parseFloat(document.getElementById('numero1').value) +
8              parseFloat(document.getElementById('numero2').value);
9
10         document.getElementById("resultado").style.color = "red";
11         document.getElementById("resultado").style.backgroundColor = "yellow";
12     }
```

Figura 308. Criação da função para realizar a soma na página exemplo8.html
[Fonte: Próprio Autor]

Agora, vamos executar nossa página e realizar um teste para ver como será o resultado. Repare que o campo Resultado ficou com o fundo amarelo e com a cor de fonte vermelho.



exemplo8.html

Arquivo | C:/Users/prof-/Desktop/projeto_js/paginas/exemplo8.html

Anônima

Entre com os numeros a serem somados

Numero 1 : 4

Numero 2 : 3

Resultado : 7

Calcular

Figura 309. Página exemplo8.html em execução no navegador após realizar a soma de dois números. [Fonte: Próprio Autor]