EAD

EDUCAÇÃO A DISTÂNCIA

UNIDADE III

Programação *Web* Responsiva

Prof. Me. Luiz Lozano

O que é JavaScript?

- A linguagem de programação JavaScript, como o 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 (FREEMAN; ROBSON, 2016).

O que é JavaScript?

- 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 nas *tags* Html, permite adicionar os efeitos de apresentação e interatividade em páginas (BALDUÍNO, 2013).

O que é JavaScript?

- 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 (FREEMAN; ROBSON, 2016).

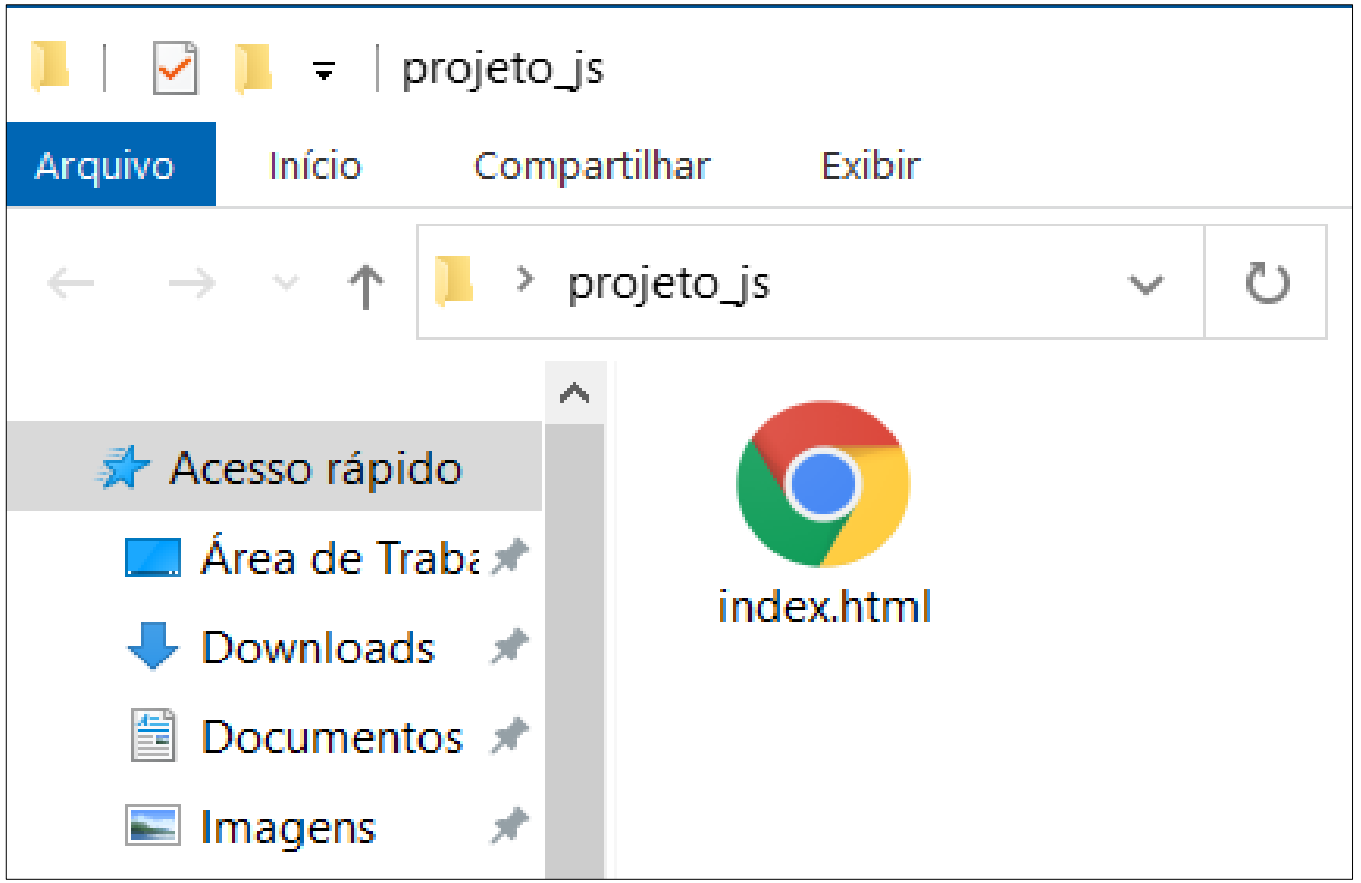
Iniciando com JavaScript

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

Fonte: autoria própria.

Iniciando com JavaScript

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



Fonte: autoria própria.

Iniciando com JavaScript

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

index.html

C: > Users > prof- > Desktop > projeto_js > index.html >

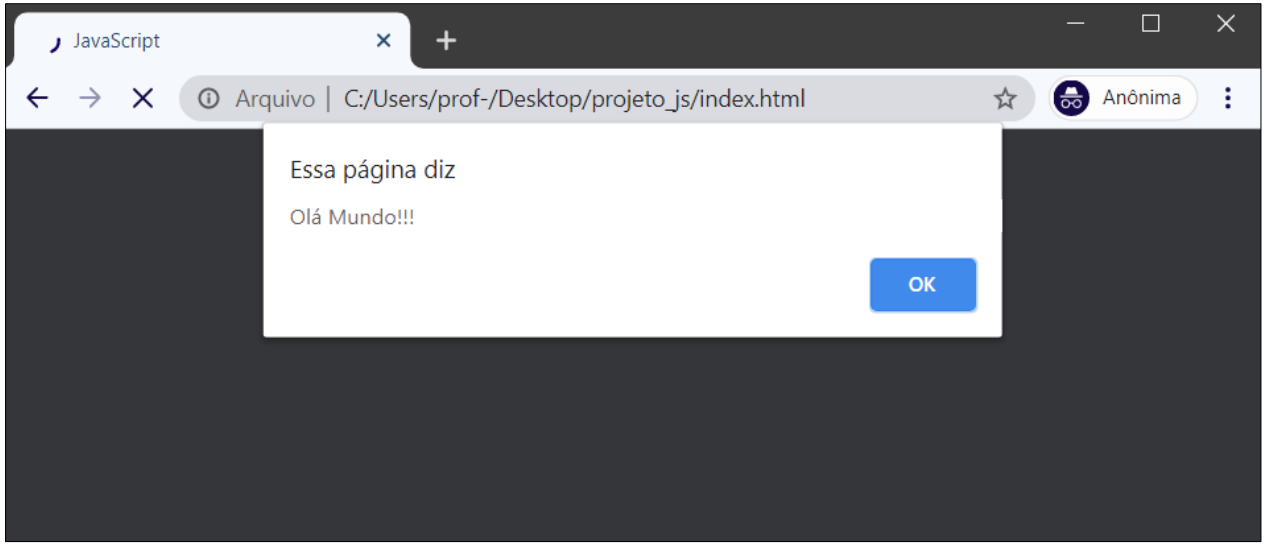
```
1   <!DOCTYPE html>
2   <html lang="pt-br">
3       <head>
4           <meta charset="utf-8">
5           <title>JavaScript</title>
6       </head>
7       <body>
8           <h1>JavaScript</h1>
9           <h2>Meu primeiro exemplo</h2>
10      </body>
11  </html>
```

Fonte: autoria própria.

Iniciando com JavaScript

- Agora, dentro do <head> iremos inserir a nossa primeira *tag* <script> e, dentro dela, um evento do tipo *alert* em JavaScript.
- O comando *alert* é responsável por emitir uma mensagem de alerta no navegador. No exemplo, emitiremos um alerta informando “Olá Mundo!!!”.

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



Fonte: autoria própria.

Iniciando com JavaScript

- Após, clicar no botão “OK”.

Fonte: autoria própria.

- 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>.

Iniciando com JavaScript

- Vamos remover o *script* do <head> e colocá-lo dentro do <body>.

index.html

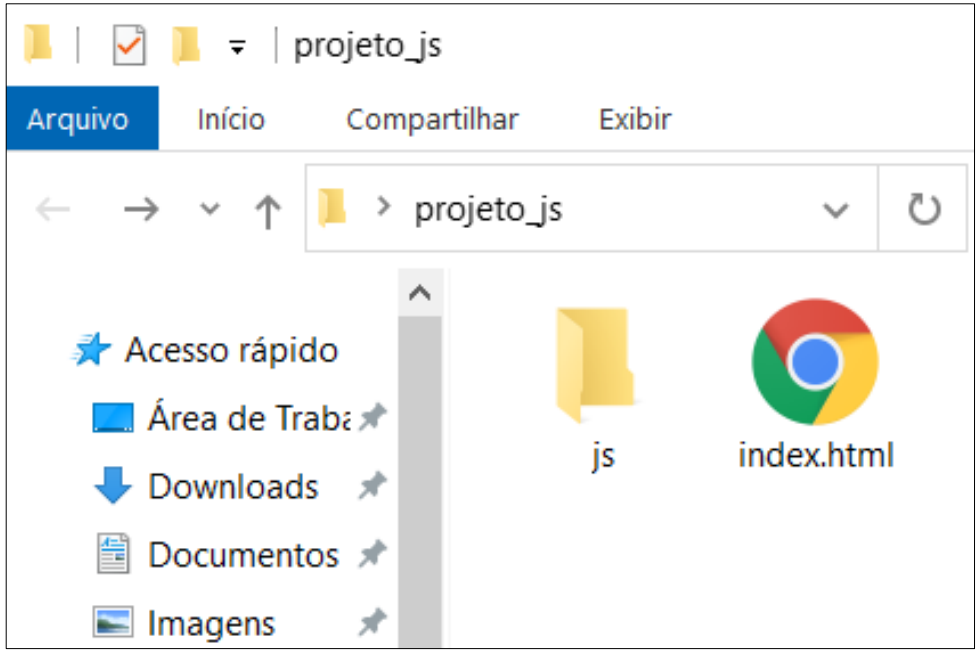
C: > Users > prof- > Desktop > projeto_js > index.html >

```
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>
```

Fonte: autoria própria.

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 JavaScriprit e esse nome foi dado para organizar melhor o projeto, portanto, podendo a seu critério ser atribuído outro nome a essa pasta.

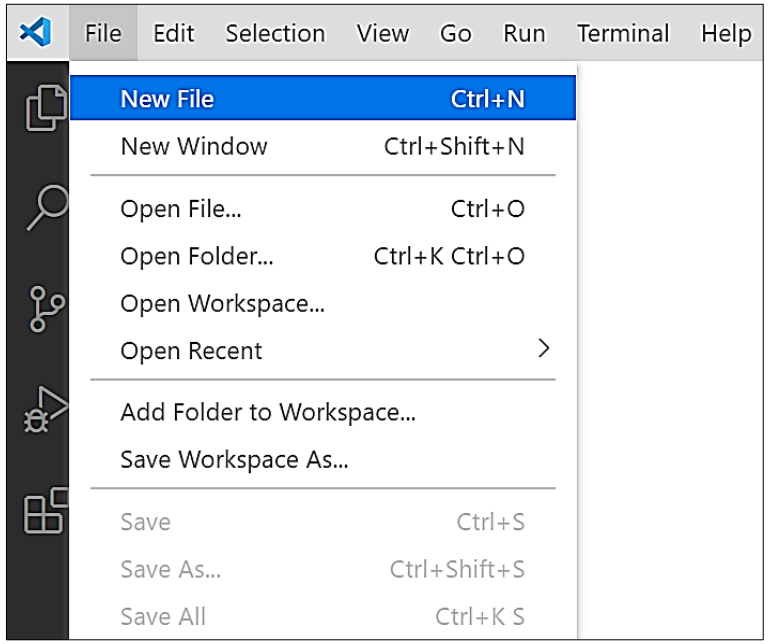


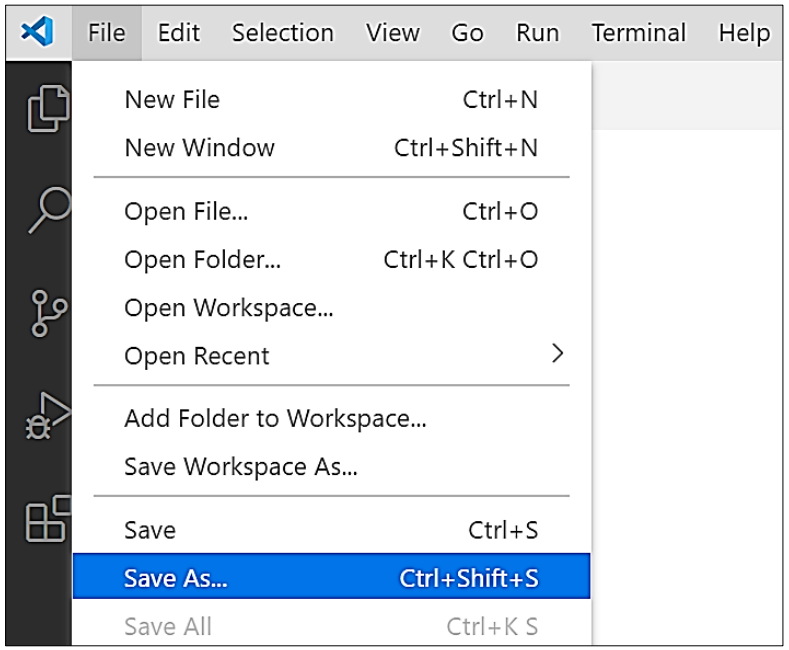
Fonte: autoria própria.

JavaScript em arquivo externo

- Agora, vamos criar um arquivo de JavaScript. Para isso, abra o Microsoft Visual Studio Code e clique em “*File/New File*”.

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

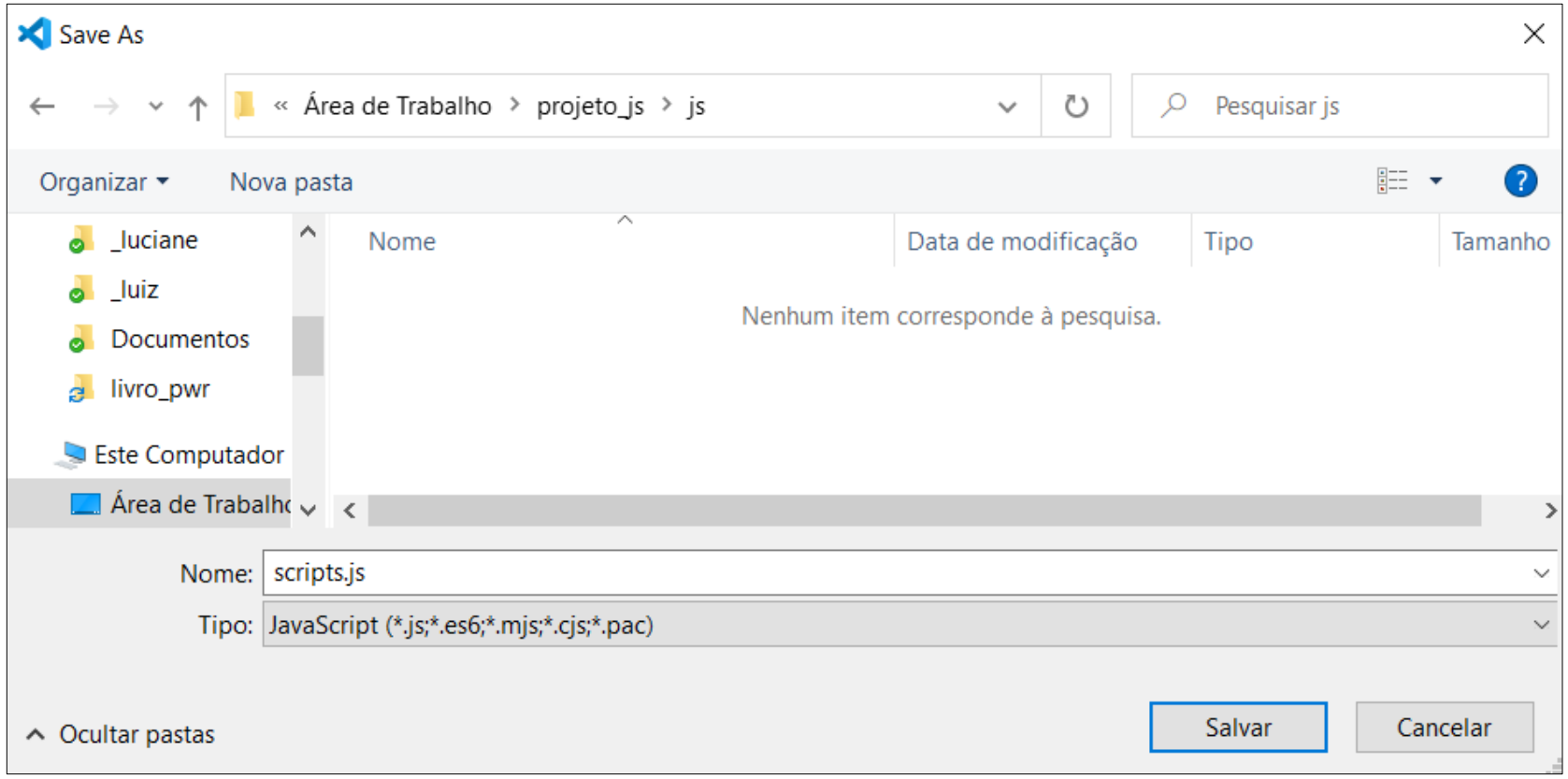




Fonte: autoria própria.

JavaScript em arquivo externo

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



Fonte: autoria própria.

JavaScript em arquivo externo

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

JS scripts.js ×

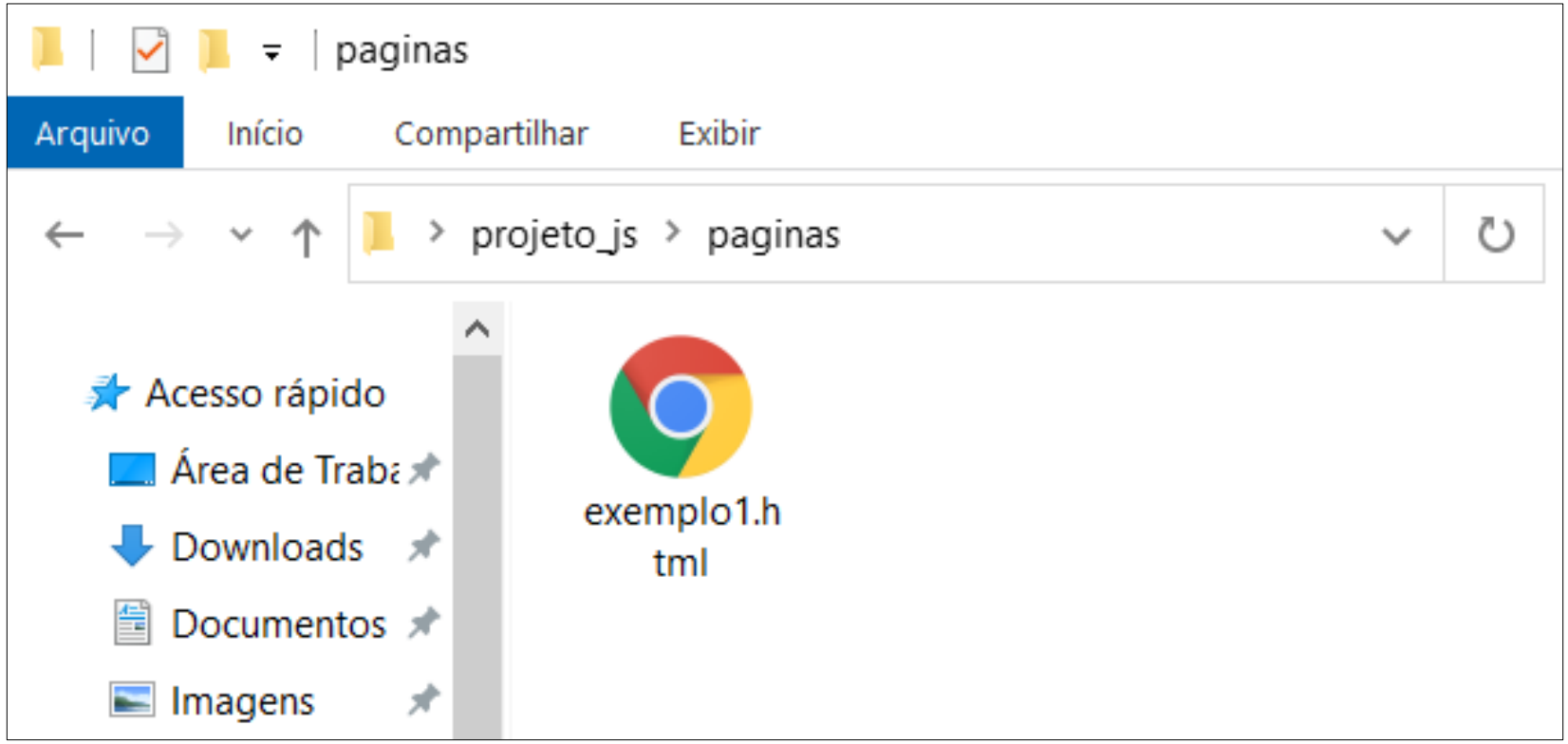
C: > Users > prof- > Desktop > projeto_js > js > JS scripts.js

```
1    alert("Olá Mundo!!! Agora, utilizando um arquivo externo");
2
```

Fonte: autoria própria.

JavaScript em arquivo externo

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



Fonte: autoria própria.

JavaScript em arquivo externo

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

exemplo1.html

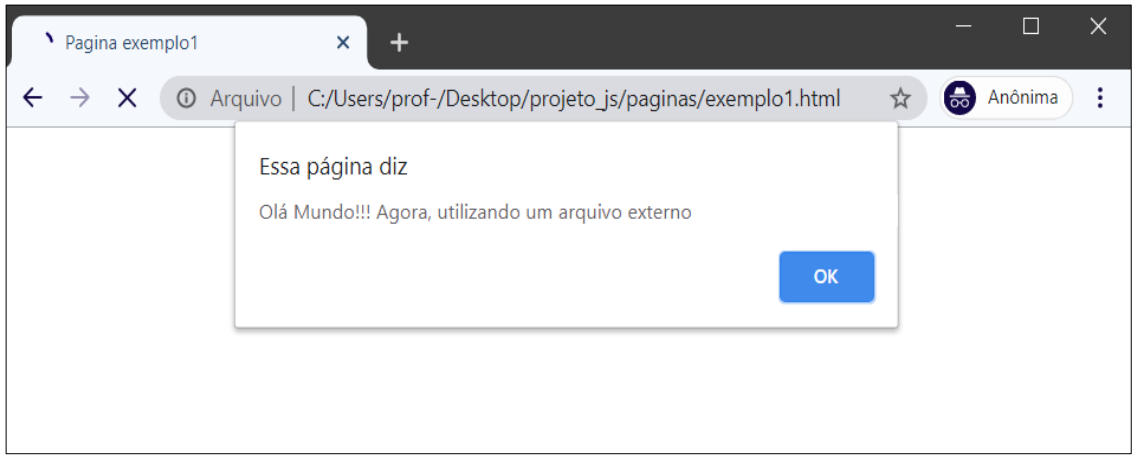
C: > Users > prof- > Desktop > projeto_js > paginas > exemplo1.html > ...

```
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>
```

Fonte: autoria própria.

JavaScript em arquivo externo

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



Fonte: autoria própria.

Interatividade

O JavaScript, como o próprio nome sugere, é uma linguagem de *scripting*. Uma linguagem de *scripting* é comumente definida como uma linguagem de programação, que permite ao programador controlar uma ou mais aplicações de terceiros; baseado nisso, temos duas asserções:

1 – Diferentemente do HTML, o Javascript é considerado uma linguagem de programação.

Porque

2 – Assim como o HTML, o Javascript é uma linguagem de marcação.

A respeito dessas asserções, assinale a alternativa correta:

a) As duas afirmações são verdadeiras e a segunda justifica a primeira.

b) As duas afirmações são verdadeiras e a segunda não justifica a primeira.

c) A primeira afirmação é falsa e a segunda é verdadeira.

d) A primeira afirmação é verdadeira e a segunda é falsa.

e) As duas afirmações são falsas.

Resposta

O JavaScript, como o próprio nome sugere, é uma linguagem de *scripting*. Uma linguagem de *scripting* é comumente definida como uma linguagem de programação, que permite ao programador controlar uma ou mais aplicações de terceiros; baseado nisso, temos duas asserções:

1 – Diferentemente do HTML, o Javascript é considerado uma linguagem de programação.

Porque

2 – Assim como o HTML, o Javascript é uma linguagem de marcação.

A respeito dessas asserções, assinale a alternativa correta:

a) As duas afirmações são verdadeiras e a segunda justifica a primeira.

b) As duas afirmações são verdadeiras e a segunda não justifica a primeira.

c) A primeira afirmação é falsa e a segunda é verdadeira.

d) A primeira afirmação é verdadeira e a segunda é falsa.

e) As duas afirmações são falsas.

Eventos

No JavaScript podemos utilizar vários eventos em diversos elementos para a interação com o usuário; a seguir listamos os principais:

- *oninput*: ocorre quando num *input* ocorre a 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 elemento passa a ter o foco;
    - *onchange*: evento de quando um campo texto tem o seu valor modificado;

Eventos

- *onload*: evento de quando uma página é atualizada;
- *onunload*: evento de quando uma página é fechada;
- *onsubmit*: evento de disparo antes de envio do formulário.

Função

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

Função

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 na qual iremos criar (clique_botão):

exemplo2.html

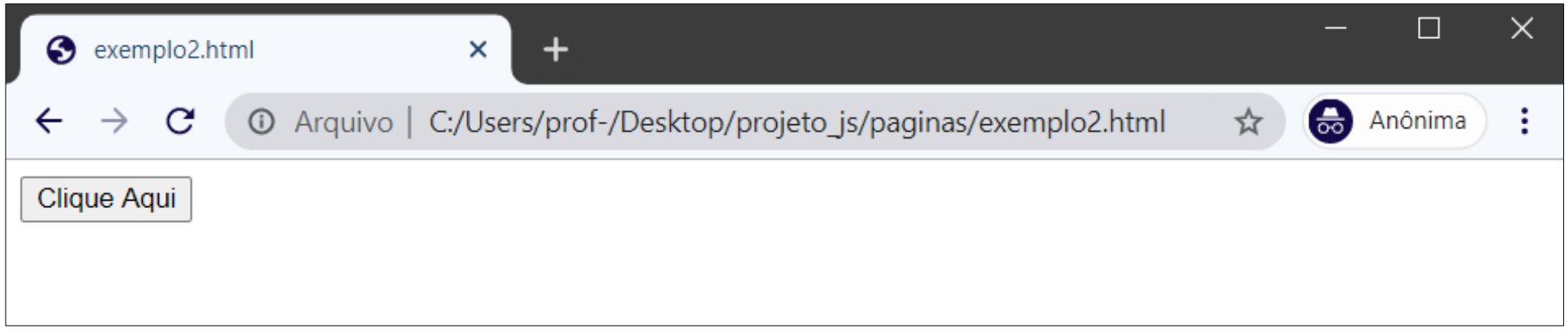
C: > Users > prof- > Desktop > projeto_js > paginas > exemplo2.html > ...

```
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>
```

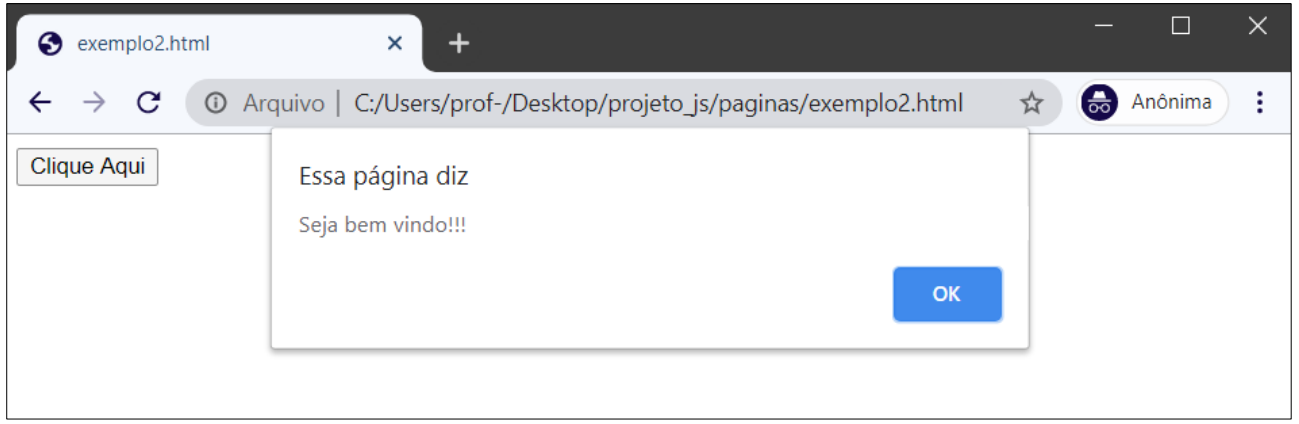
Fonte: autoria própria.

Função

Ao executar a página teremos o seguinte resultado:



Ao clicar no botão, o seguinte resultado será apresentado:



Fontes: autoria própria.

DOM

- Já estudamos muito o HTML, criamos as 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.

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 para que eles servem:

- getElementById();
- innerHTML.

.

Métodos

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

exemplo3.html

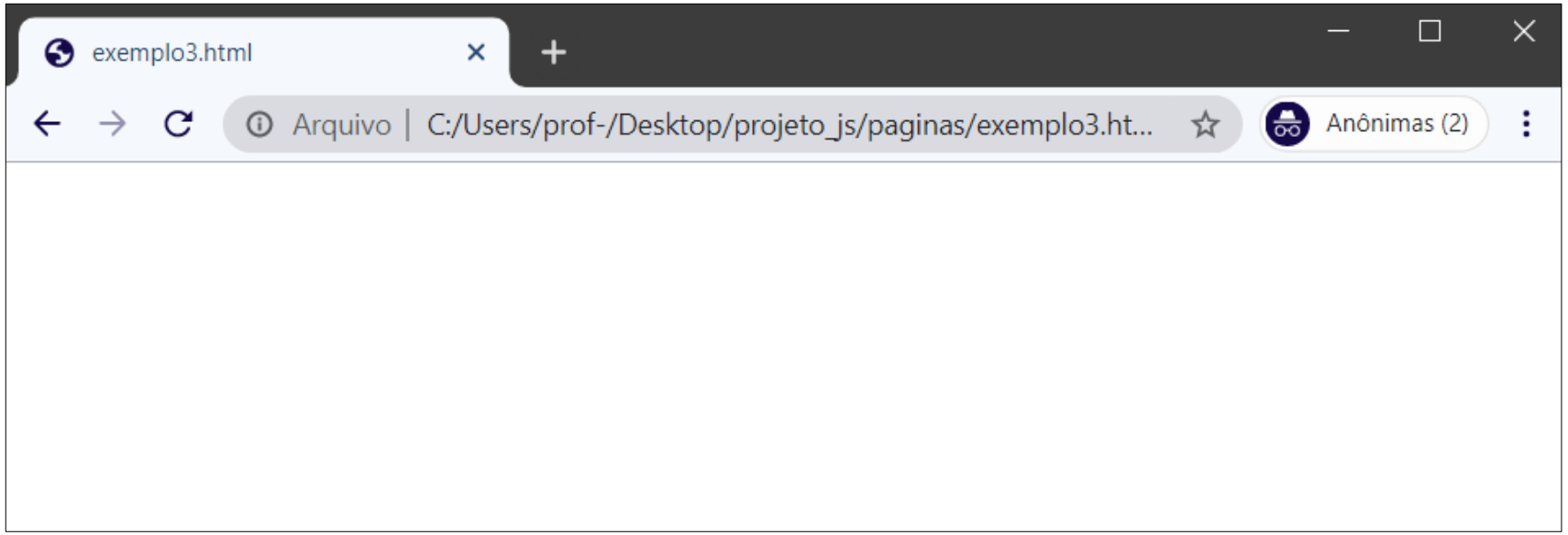
C: > Users > prof- > Desktop > projeto_js > paginas > exemplo3.html

```
1   <!DOCTYPE html>
2   <html lang="pt-br">
3       <body>
4           <p id="demo"></p>
5       </body>
6   </html>
```

Fonte: autoria própria.

Métodos

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



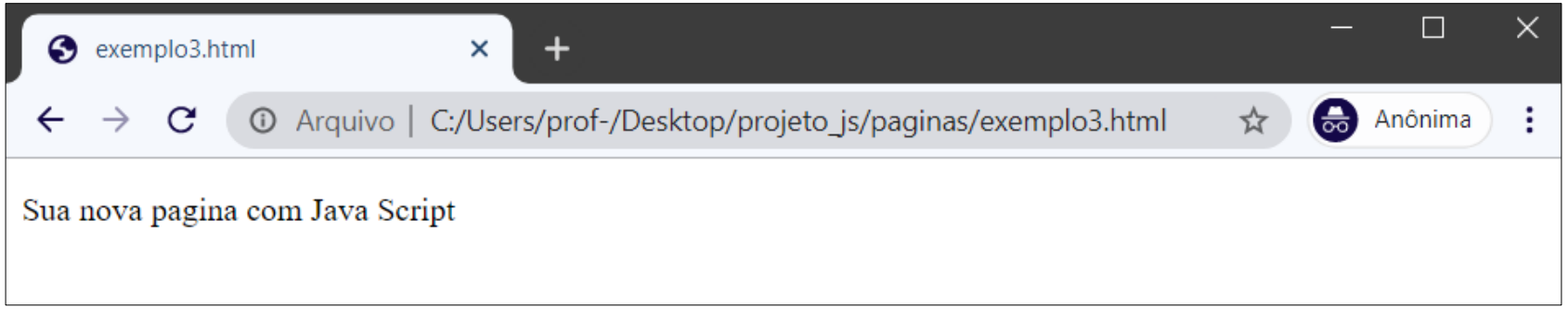
Fonte: autoria própria.

Métodos

Agora, definiremos um conteúdo para o parágrafo, através do seguinte código JavaScript:

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

Ao executar a página no navegador teremos o seguinte resultado:



Fontes: autoria própria.

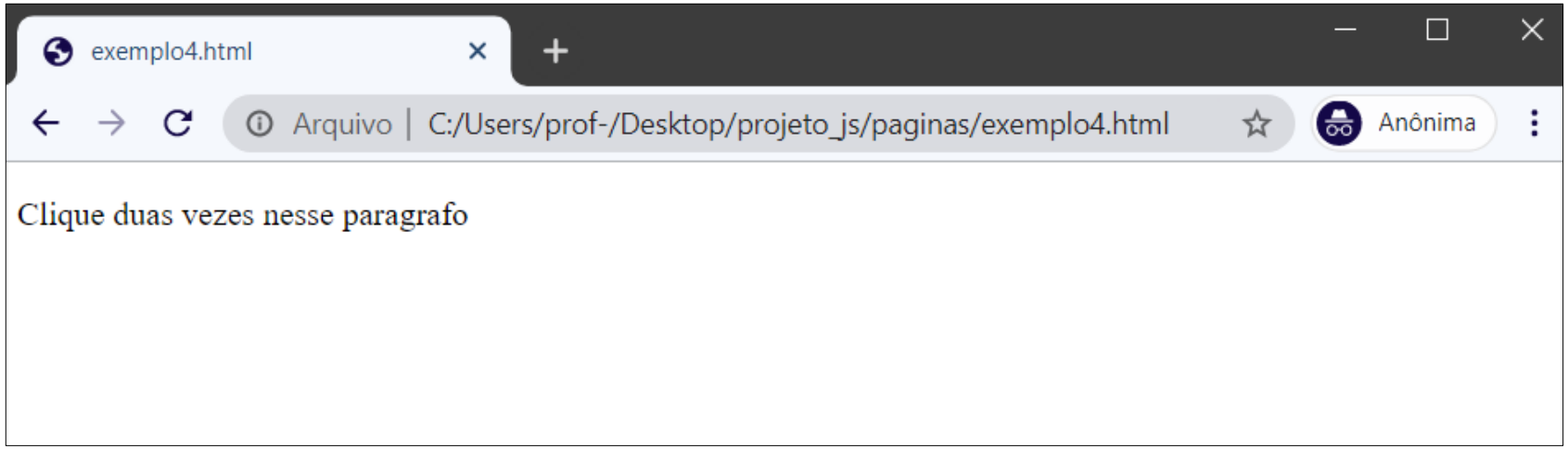
Métodos

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 *ondblclick* e definiremos um *id* para o segundo, conforme a imagem a seguir:

exemplo4.html

C: > Users > prof- > Desktop > projeto_js > paginas > exemplo4.html > ...

```
1   <!DOCTYPE html>
2   <html lang="pt-br">
3       <body>
4           <p ondblclick="myFunction()">Clique duas vezes nesse paragrafo</p>
5           <p id="demo"></p>
6       </body>
7   </html>
```

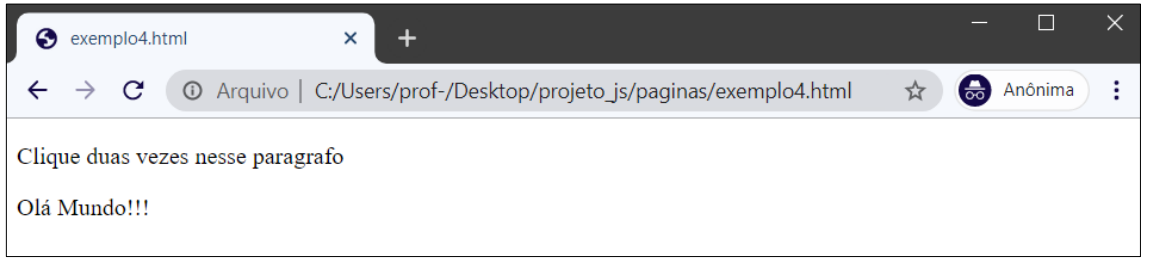


Fontes: autoria própria.

Métodos

Dando sequência, complemente o nosso código, incluindo o seguinte código JavaScript, conforme a 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 a execução deste trecho da página:

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



Fontes: autoria própria.

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

Interatividade

Criado pelo W3C, O DOM é uma multiplataforma que representa como as marcações em HTML, XHTML e XML são organizadas e lidas pelo navegador que você usa. Uma vez indexadas, estas marcações se transformam em elementos de uma árvore que você pode manipular via API, que é:

a) O que fazemos quando usamos programas ou *scripts* para excluir as funcionalidades de uma página.
b) O que fazemos quando deletamos programas ou *scripts* de uma página.
c) O que fazemos quando usamos programas ou *scripts* sem nenhuma pretensão.
d) O que fazemos quando usamos programas ou *scripts* para alterar as funcionalidades de uma página.
e) O que fazemos quando usamos páginas ou folhas para alterar as funcionalidades de um *script*.

Resposta

Criado pelo W3C, O DOM é uma multiplataforma que representa como as marcações em HTML, XHTML e XML são organizadas e lidas pelo navegador que você usa. Uma vez indexadas, estas marcações se transformam em elementos de uma árvore que você pode manipular via API, que é:

a) O que fazemos quando usamos programas ou *scripts* para excluir as funcionalidades de uma página.

b) O que fazemos quando deletamos programas ou *scripts* de uma página.

c) O que fazemos quando usamos programas ou *scripts* sem nenhuma pretensão.

d) O que fazemos quando usamos programas ou *scripts* para alterar as funcionalidades de uma página.

e) O que fazemos quando usamos páginas ou folhas para alterar as funcionalidades de um *script*.

Bootstrap

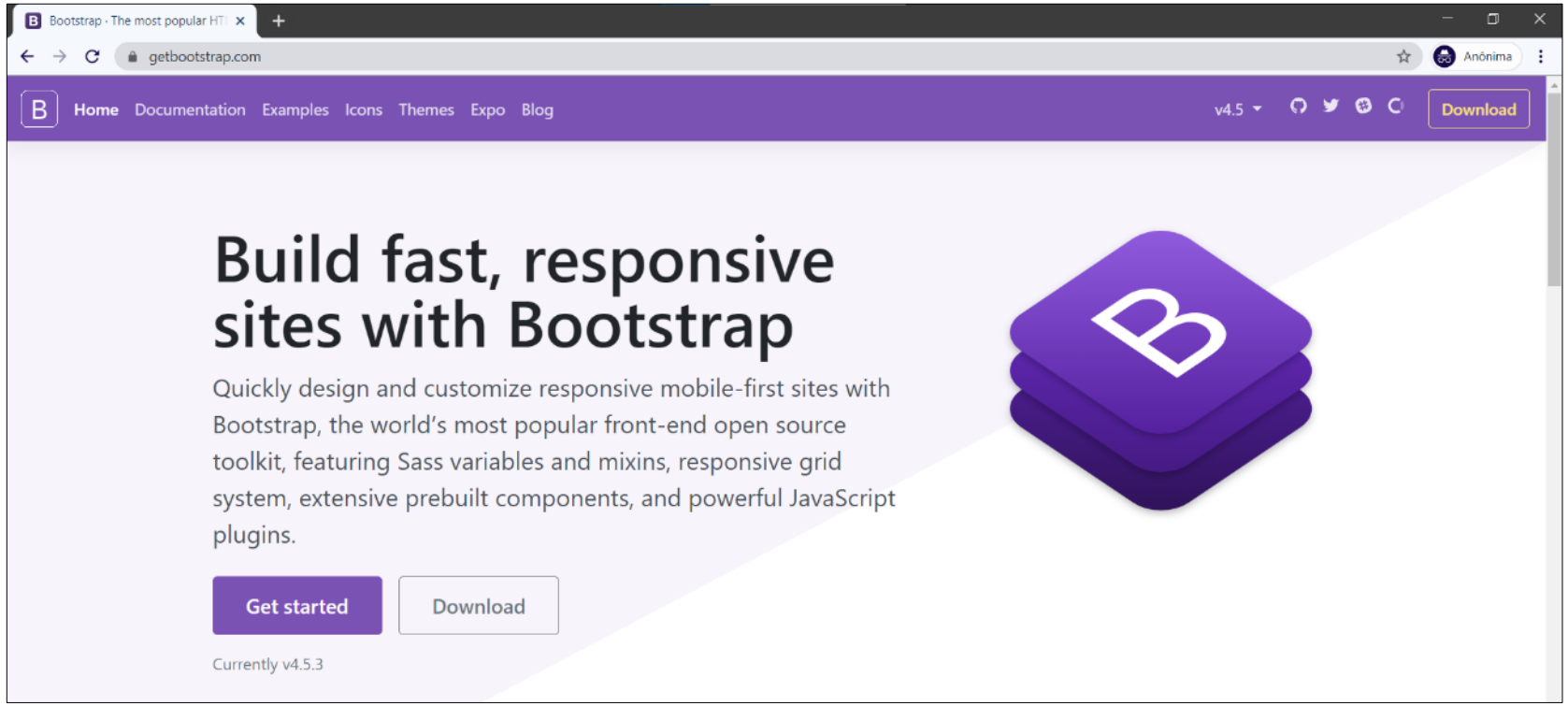
- O *framework* Bootstrap é um *framework* JavaScript, HTML e CSS, para o desenvolvimento de *sites* e aplicações *web* responsivas. Torna o desenvolvimento de páginas *web* muito mais dinâmico. Indicado para todos os níveis de desenvolvedores e funciona em dispositivos dos mais variados tipos.

Bootstrap

- Em meados de 2011, Mark Otto, um desenvolvedor trabalhando no Twitter, criador do Bootstrap, juntamente com Jacob Thornton, anunciou ao mundo o lançamento do Bootstrap, em um artigo publicado no *blog* do Twitter, relatando que, no início do Twitter, cada desenvolvedor usava a biblioteca que melhor conhecia, porém esse procedimento criou inconsistências, dificultando a integração, a escalabilidade e a manutenção das aplicações criadas por diferentes desenvolvedores da equipe.

Instalando o Bootstrap

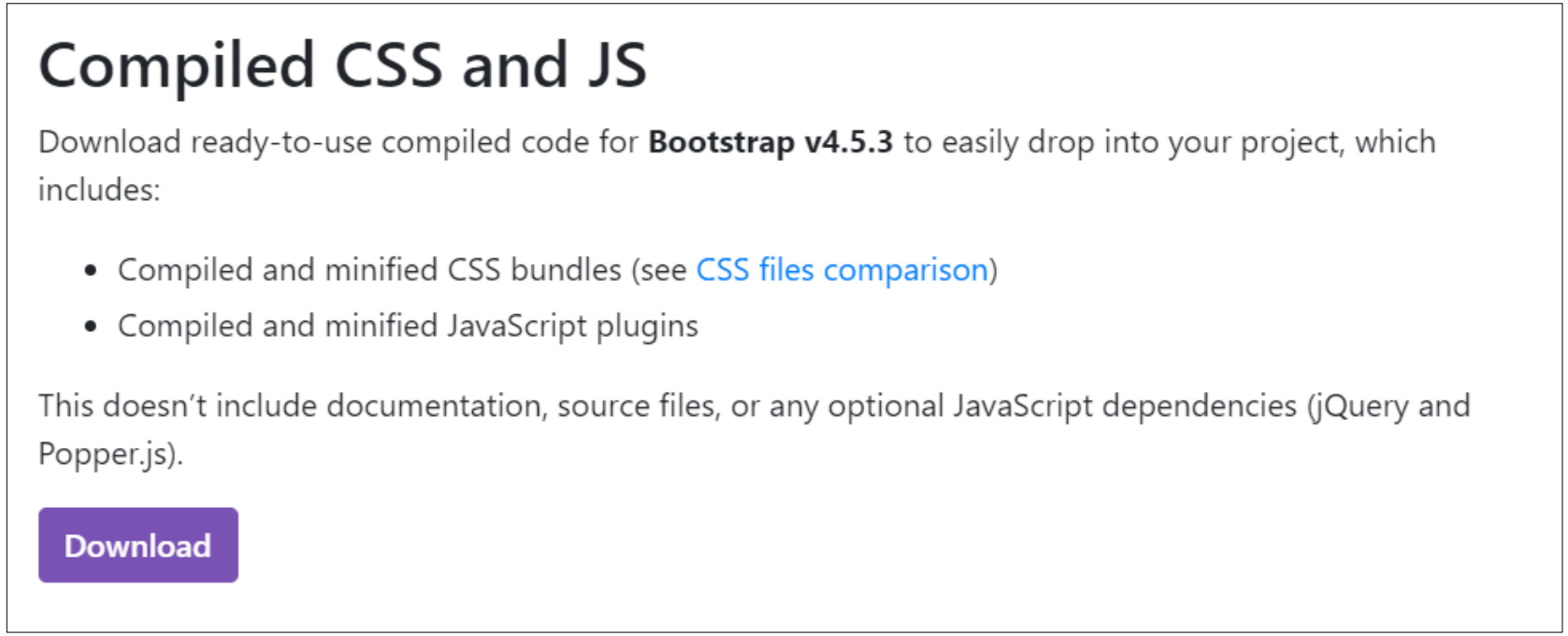
- Acesse o *site* do Bootstrap em: http://getbootstrap.com. Logo no início, você encontra um *link* para o *download* do Bootstrap. Clique em "*Download*" para iniciar o mesmo imediatamente.



Fonte: autoria própria.

Instalando o Bootstrap

Vá na opção “*Compiled* CSS *and* JS” e clique na opção “*Download*”:



Fonte: autoria própria.

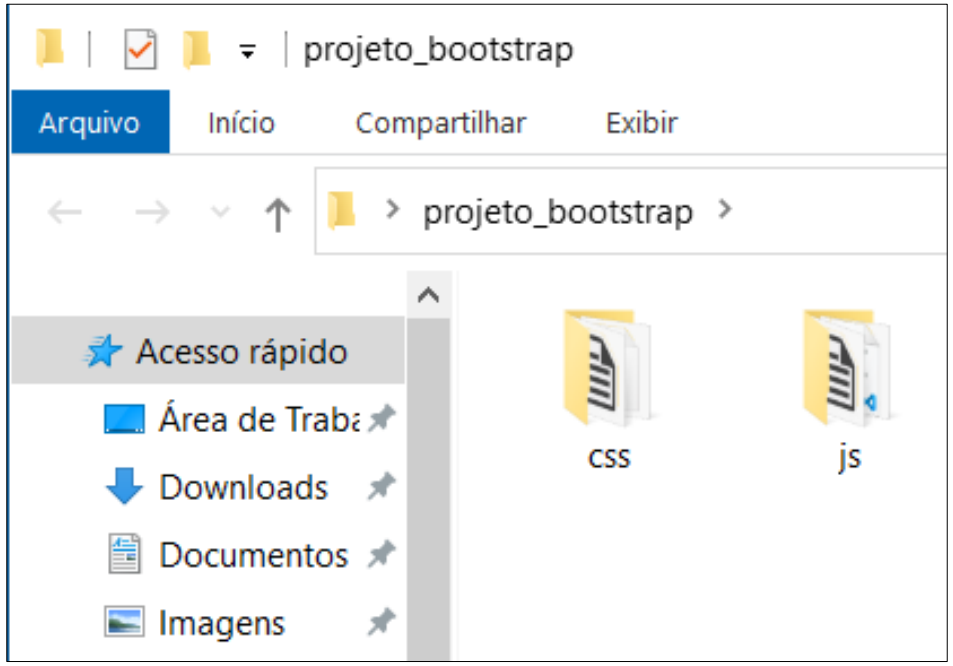
Criando um novo projeto com o Bootstrap

- Crie uma pasta em sua área de trabalho, onde iremos armazenar o nosso projeto utilizando o Bootstrap. No caso, a pasta foi criada como “projeto_bootstrap”.

Fonte: autoria própria.

Criando um novo projeto com o Bootstrap

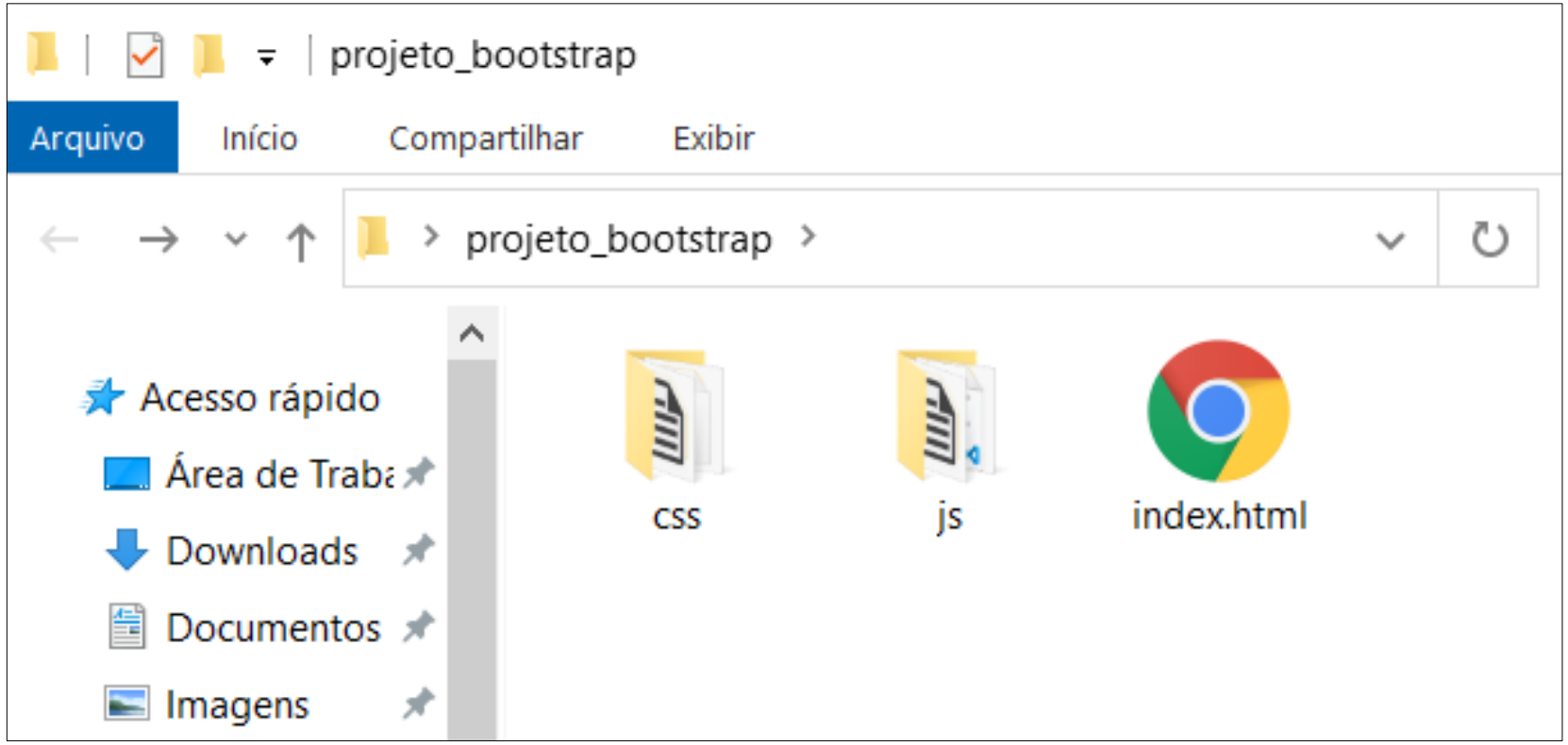
- Iremos descompactar o arquivo do “bootstrap.zip” que baixamos dentro da nossa pasta “projeto_bootstrap”.
- Teremos dentro da pasta “projeto_bootstrap” a seguinte hierarquia de arquivos, criada automaticamente. Feito isso, o Bootstrap está dentro da nossa pasta do projeto, porém, agora, ao trabalhar com o nosso projeto, teremos que importá-lo dentro de nosso código. Porém, isso veremos nos próximos passos do nosso estudo.



Fonte: autoria própria.

Criando um novo projeto com o Bootstrap

Dentro da pasta raiz do nosso projeto, iremos criar um arquivo chamado “index.html”, conforme já fizemos anteriormente.



Fonte: autoria própria.

Criando um novo projeto com o Bootstrap

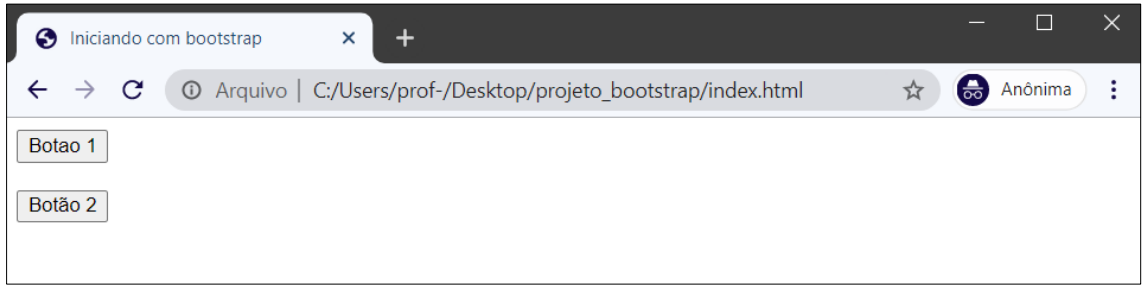
Esse arquivo deverá possuir a estrutura básica de uma página html. Dentro do <body> iremos criar dois botões, conforme a estrutura a seguir:

index.html

C: > Users > prof- > Desktop > projeto_bootstrap > index.html > ...

```
1   <!DOCTYPE html>
2   <html lang="pt-br">
3       <head>
4           <meta charset="utf-8">
5           <title>Iniciando com bootstrap</title>
6       </head>
7       <body>
8           <button>Botao 1</button>
9           <br>
10          <br>
11          <button>Botão 2</button>
12      </body>
13  </html>
```

Fontes: autoria própria.



Criando um novo projeto com o Bootstrap

- Podemos notar que ambos estão com a mesma aparência dos componentes que vimos nas aulas anteriores. Até o exato momento, inserimos as pastas do Bootstrap em nosso projeto, porém não utilizamos. Vamos, agora, chamar o Bootstrap em nossa página.

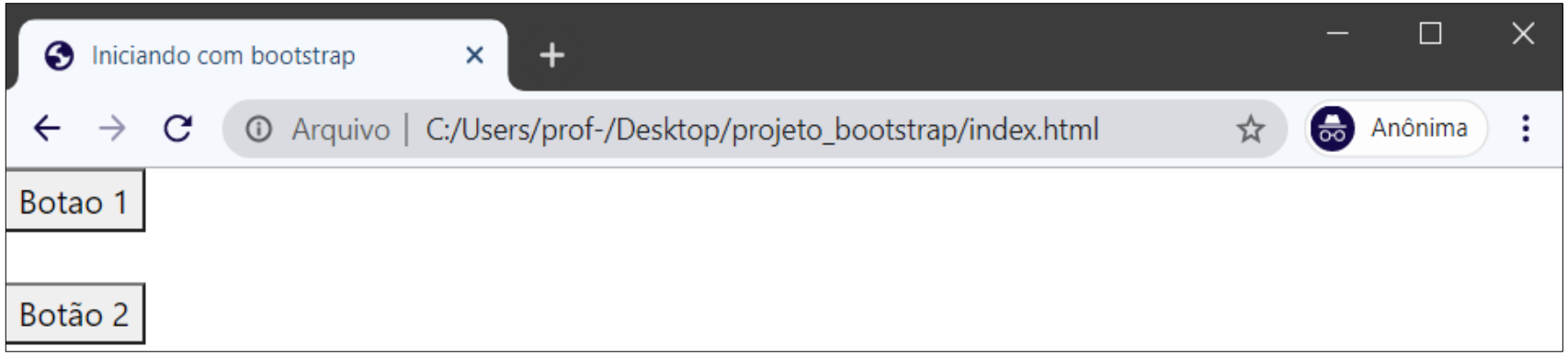
Iremos inserir dentro do nosso <head> o seguinte comando:

```
6        <link href="css/bootstrap.css" rel="stylesheet">
```

Fonte: autoria própria.

Criando um novo projeto com o Bootstrap

- Iremos notar que algumas configurações de nossa página já irão sofrer algumas alterações, visto que já estamos usando, com esse *link*, as configurações padrão de estilo do Bootstrap, como, por exemplo: a distância para a borda da esquerda e alguns polimentos de alguns componentes (dependendo do seu navegador, essas diferenças serão mínimas).



Fonte: autoria própria.

Interatividade

Bootstrap é o mais popular *framework* JavaScript, HTML e CSS, para o desenvolvimento de *sites* e aplicações *web* responsivas, e alinhadas com a filosofia *mobile first*. Torna o desenvolvimento *front-end* muito mais rápido e fácil. É indicado para:

a) Desenvolvedores de níveis avançados de conhecimento.
b) Gestores de elevados *feelings* de gestão.
c) Desenvolvedores com o Ensino Fundamental completo.
d) Gestores que utilizam as metodologias ágeis.
e) Desenvolvedores de todos os níveis de conhecimento.

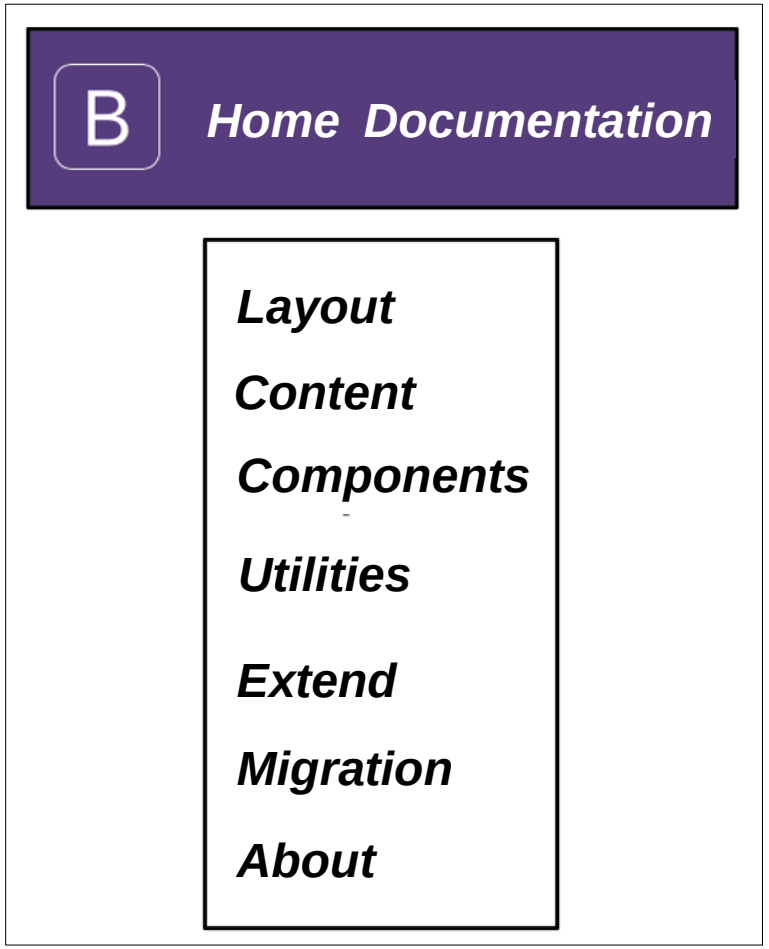
Resposta

Bootstrap é o mais popular *framework* JavaScript, HTML e CSS, para o desenvolvimento de *sites* e aplicações *web* responsivas, e alinhadas com a filosofia *mobile first*. Torna o desenvolvimento *front-end* muito mais rápido e fácil. É indicado para:

a) Desenvolvedores de níveis avançados de conhecimento.
b) Gestores de elevados *feelings* de gestão.
c) Desenvolvedores com o Ensino Fundamental completo.
d) Gestores que utilizam as metodologias ágeis.
e) Desenvolvedores de todos os níveis de conhecimento.

Explorando a documentação do Bootstrap

Agora, iremos à documentação do Bootstrap para buscar como inserir alguns componentes; para isso, vamos acessar o *site* getbootstrap.com e ir na seção de *documentation/components*, conforme a imagem a seguir:



Fonte: autoria própria.

Explorando a documentação do Bootstrap

- Na barra de componentes iremos encontrar diversos componentes; no caso do exemplo que estamos trabalhando, iremos clicar em “*Buttons*”.

Components

Alerts

Badge

Breadcrumb

Buttons

Button group

Card

Carousel

Collapse

Dropdowns

Forms

Examples

Bootstrap includes several predefined button styles, each serving its own semantic purpose, with a few extras thrown in for more control.

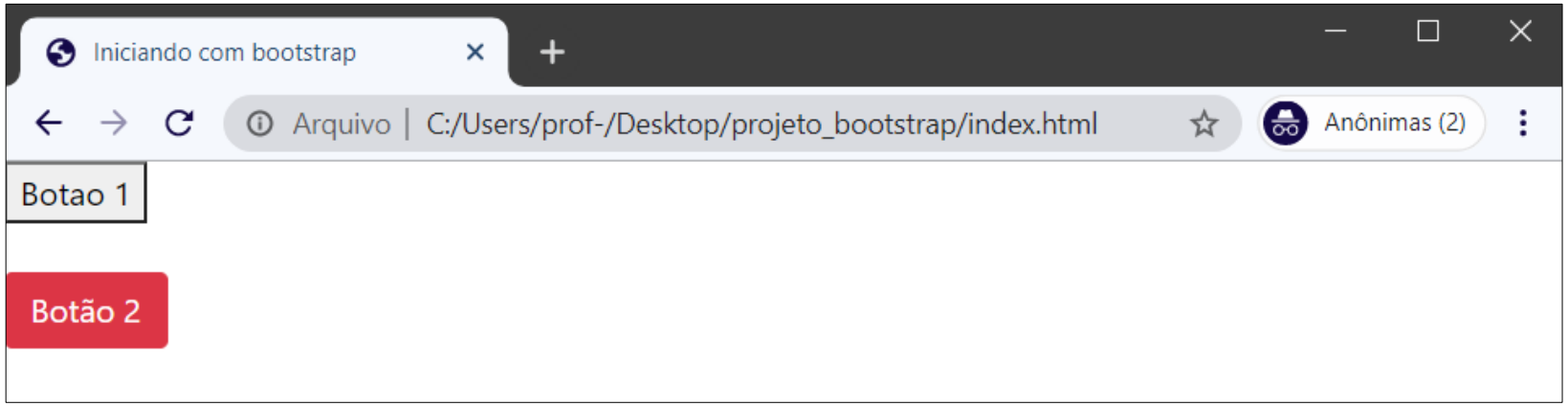

```
<button type="button" class="btn btn-primary">Primary</button>
<button type="button" class="btn btn-secondary">Secondary</button>
<button type="button" class="btn btn-success">Success</button>
<button type="button" class="btn btn-danger">Danger</button>
<button type="button" class="btn btn-warning">Warning</button>
<button type="button" class="btn btn-info">Info</button>
<button type="button" class="btn btn-light">Light</button>
<button type="button" class="btn btn-dark">Dark</button>

<button type="button" class="btn btn-link">Link</button>
```

Fonte: autoria própria.

Explorando a documentação do Bootstrap

Vamos utilizar o código do botão vermelho (quarta linha do exemplo anterior) no botão dois da página index.html. Com isso, o código do botão dois irá ficar da seguinte forma:

```
12          <button type="button" class="btn btn-danger">Botão 2</button>
```

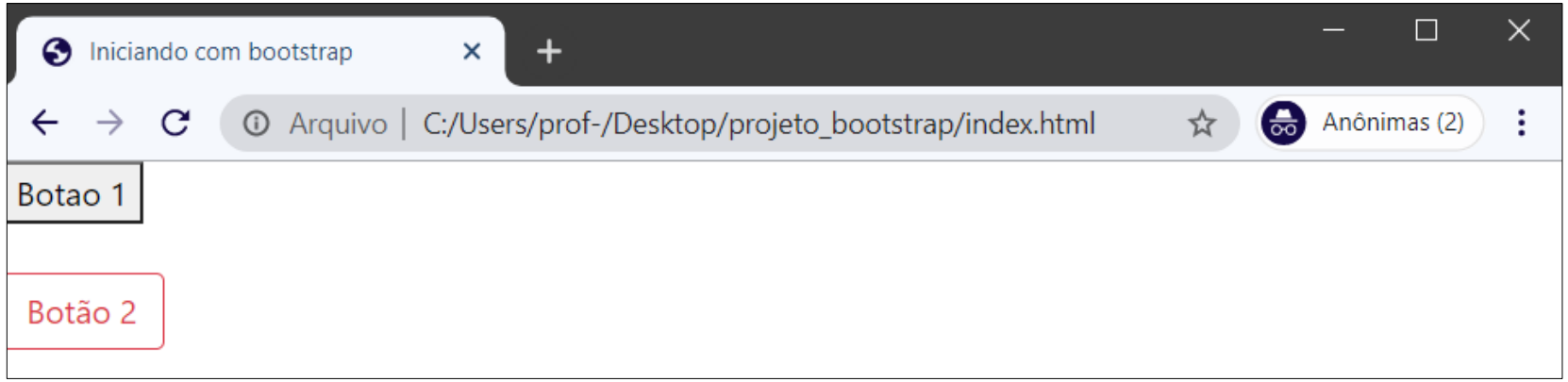


Fontes: autoria própria.

Explorando a documentação do Bootstrap

- O botão dois está utilizando as configurações do *framework* Bootstrap para ficar com um *layout* mais bonito. Podemos notar o quanto é rica a documentação do Bootstrap; podemos, assim, explorar os diversos componentes através do próprio *site* oficial e de forma gratuita.
- Se precisarmos de um botão, porém sem as cores de fundo, podemos substituir as classes modificadoras padrão por “.btn-outline-*” para remover todas as imagens e cores de fundo em qualquer botão.

```
12          <button type="button" class="btn btn-outline-danger">Botão 2</button>
```

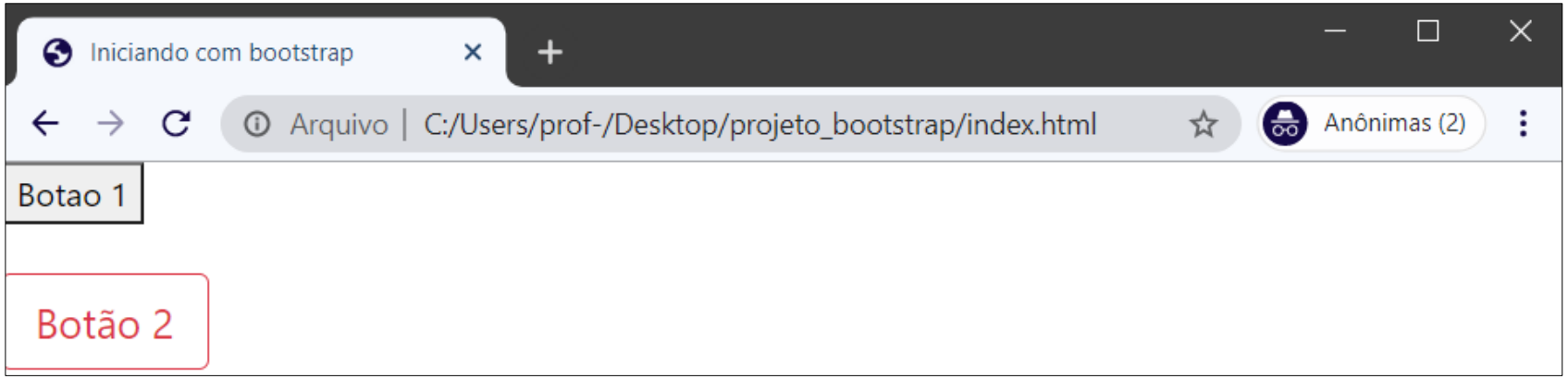


Fontes: autoria própria.

Explorando a documentação do Bootstrap

- Gosta de botões maiores ou menores? Adicione .btn-lg ou .btn-sm para tamanhos adicionais.

```
12        <button type="button" class="btn btn-outline-danger btn-lg">Botão 2</button>
```

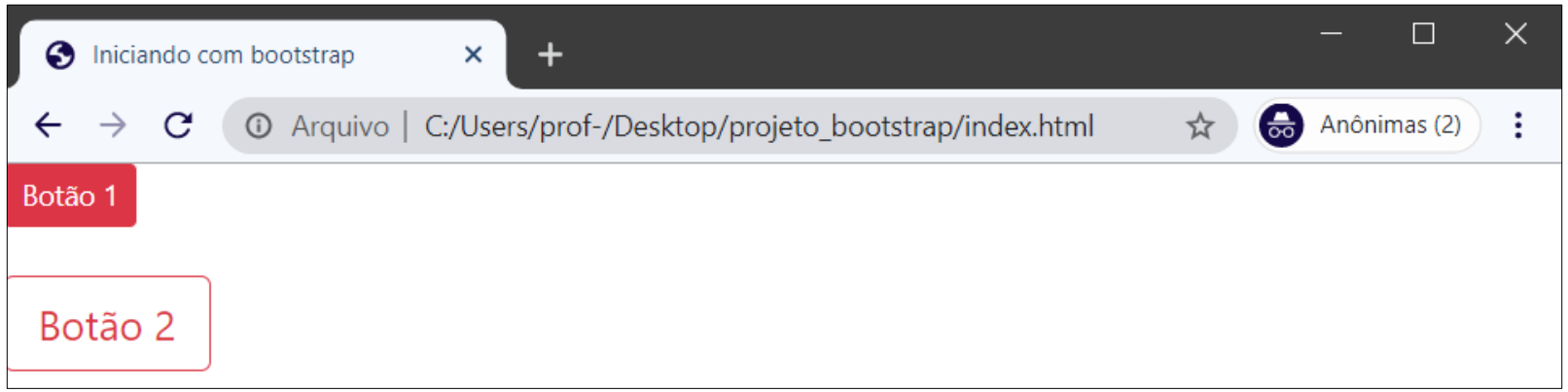


Fontes: autoria própria.

Explorando a documentação do Bootstrap

- Agora, vamos customizar o nosso botão um, porém utilizando btn-sm para deixá-lo menor e compará-lo com o botão dois.

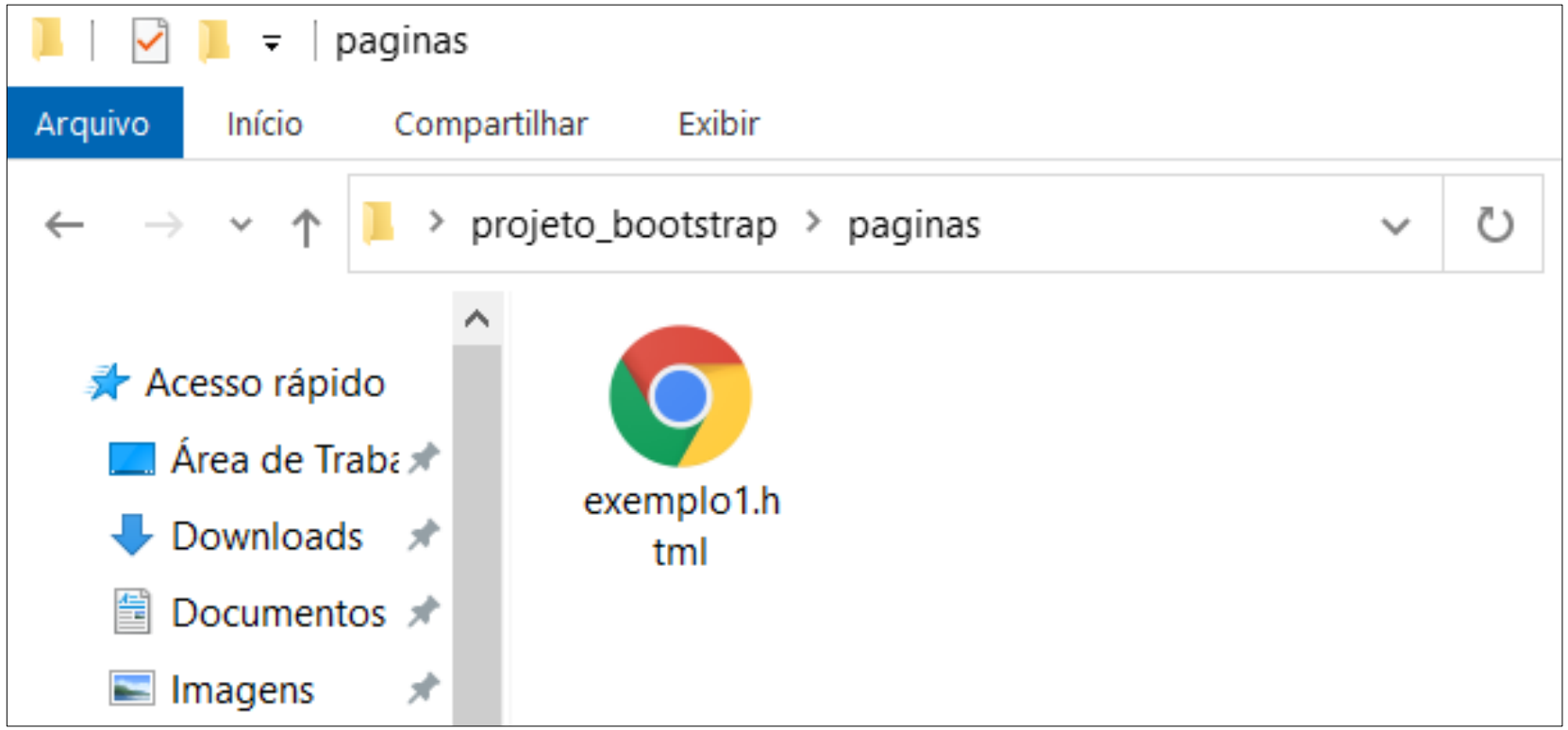
```
9        <button type="button" class="btn btn-danger btn-sm">Botão 1</button>
```



Fontes: autoria própria.

Criando um *menu* responsivo

Agora, iremos aprender a criar um *menu* responsivo utilizando o Bootstrap. Antes disso, iremos criar uma pasta chamada “paginas”, dentro do nosso projeto. Em seguida, criaremos um arquivo chamado “exemplo1.html”. Nossa pasta ficará com a seguinte estrutura:



Fonte: autoria própria.

Criando um *menu* responsivo

Criaremos, agora, uma estrutura básica html em nossa página, fazendo o *link* para o Bootstrap:

exemplo1.html

C: > Users > prof- > Desktop > projeto_bootstrap > paginas > exemplo1.html > ...

```
1   <!DOCTYPE html>
2   <html lang="pt-br">
3       <head>
4           <meta charset="utf-8">
5           <title>Exemplo 1</title>
6           <link href="../css/bootstrap.css" rel="stylesheet">
7       </head>
8       <body>
9
10      </body>
11  </html>
```

Fonte: autoria própria.

Criando um *menu* responsivo

Agora que o nosso arquivo “exemplo1.html” está configurado, entraremos no *site* do Bootstrap e iremos em *Documentation/Components/Navbar*.

Como funciona:

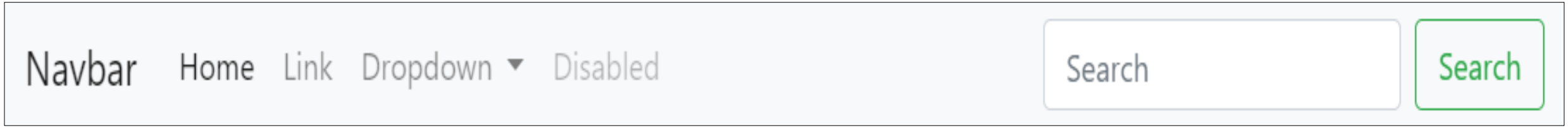
Aqui está oquê você precisa saber, antes de começar a usar o navbar:

- Navbars precisam de um `.navbar` com `.navbar-expand{-sm|-md|-lg|-xl}` (para responsividade) e classes do esquema de cores;
- Navbars e seus conteúdos são fluidos, por padrão;
  - Use containers alternativos para limitar suas larguras.
- Use nossas classes utilitárias de espaçamento e flex para controlar o espaço e alinhamento, dentro das navbars;
- Navbars são responsivos, por padrão, mas você pode modificar isso, facilmente;
  - Comportamento responsivo depende do nosso plugin JavaScript Collapse.
- Navbars são escondidos, por padrão, quando imprimindo. Force-os a serem imprimidos, usando a classe `.d-print` no `.navbar`;
  - Leia sobre a classe utilitária display.
- Garanta acessibilidade com o elemento `<nav>` ou, se estiver usando um elemento mais genérico (como `<div>`), coloque `role="navigation"` em cada navbar para identificá-lo como tal, em tecnologias assistivas.

Fonte: autoria própria.

Criando um *menu* responsivo

Veja que estamos retirando as informações dos componentes de um *menu* diretamente da documentação do Bootstrap. A seguir, o menu que iremos implementar:



Conteúdo suportado:

Navbars vem com suporte integrado para um punhado de sub-componentes. Escolha entre os seguintes, sempre que precisar:

- `.navbar-brand` para o nome de seu projeto, produto ou companhia;
- `.navbar-nav` para obter uma leve navegação com suporte a dropdowns;
- `.navbar-toggler` para usar com nosso plugin collapse e outros comportamentos;
- `.form-inline` para qualquer campo de formulário e ações;
- `.navbar-text` para adicionar texto centralizado verticalmente;
- `.collapse.navbar-collapse` para agrupar e esconder conteúdos navbar, de acordo com o breakpoint do pai.

Aqui está um exemplo de todos os sub-componentes incluídos em um navbar light responsivo, o qual colapsa no breakpoint `lg` (large, em português: grande), automaticamente.

Fontes: autoria própria.

Criando um *menu* responsivo

- Iremos clicar no botão “*copy*” (ao lado direito superior) e copiar o código do *navbar* inteiro para dentro do nosso *body*.

```
<nav class="navbar navbar-expand-lg navbar-light bg-light">
  <a class="navbar-brand" href="#">Navbar</a>
  <button class="navbar-toggler" type="button" data-toggle="collapse" data-target="#conteudoNavbarSup
    <span class="navbar-toggler-icon"></span>
  </button>

  <div class="collapse navbar-collapse" id="conteudoNavbarSuportado">
    <ul class="navbar-nav mr-auto">
      <li class="nav-item active">
        <a class="nav-link" href="#">Home <span class="sr-only">(página atual)</span></a>
      </li>
      <li class="nav-item">
        <a class="nav-link" href="#">Link</a>
      </li>
      <li class="nav-item dropdown">
        <a class="nav-link dropdown-toggle" href="#" id="navbarDropdown" role="button" data-toggle="d
          Dropdown
        </a>
        <div class="dropdown-menu" aria-labelledby="navbarDropdown">
          <a class="dropdown-item" href="#">Ação</a>
          <a class="dropdown-item" href="#">Outra ação</a>
          <div class="dropdown-divider"></div>
          <a class="dropdown-item" href="#">Algo mais aqui</a>
        </div>
      </li>
      <li class="nav-item">
        <a class="nav-link disabled" href="#">Desativado</a>
      </li>
    </ul>
    <form class="form-inline my-2 my-lg-0">
      <input class="form-control mr-sm-2" type="search" placeholder="Pesquisar" aria-label="Pesquisar
      <button class="btn btn-outline-success my-2 my-sm-0" type="submit">Pesquisar</button>
    </form>
  </div>
</nav>
```

Fonte: autoria própria.

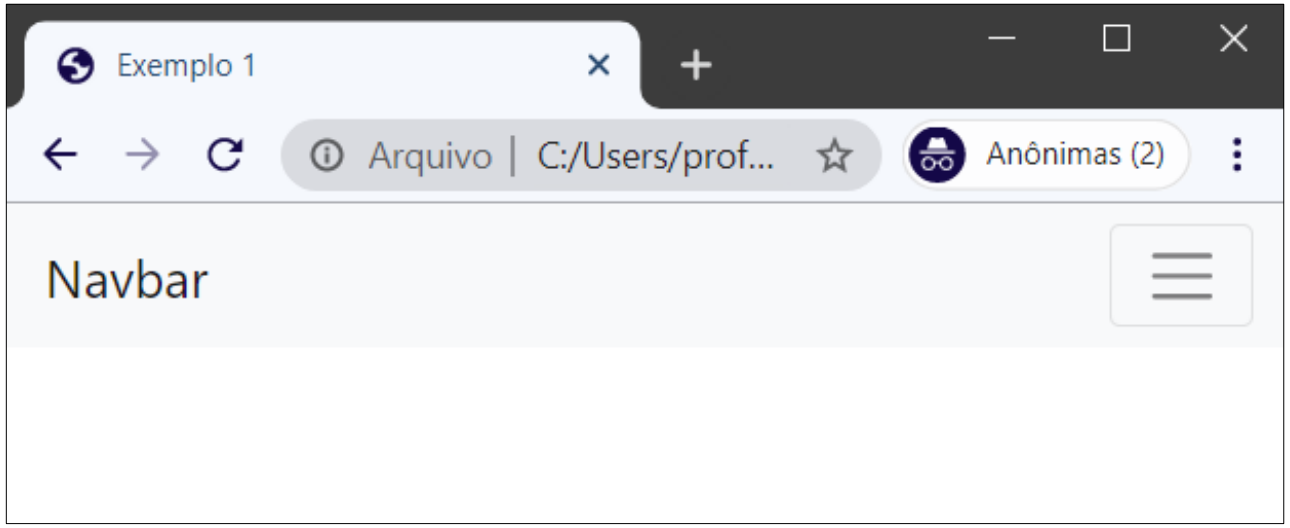
Criando um *menu* responsivo

Após salvarmos o arquivo “exemplo1.html”, com a alteração realizada anteriormente, podemos executá-la tendo, assim, o seguinte resultado em nossa página:

Fonte: autoria própria.

Criando um *menu* responsivo

Se diminuirmos (redimensionando manualmente) a nossa janela do navegador (com o intuito de simularmos uma janela do navegador em um *smartphone* ou *tablet*) teremos o seguinte resultado:



Fonte: autoria própria.

- Note que o comportamento de nosso *menu* mudou de acordo com o tamanho da tela onde ele está sendo executado. Portanto, criamos, de forma simples, um *menu* responsivo.

Interatividade

O *Responsive Web Design* é uma das soluções técnicas para esse problema: programar um *site* de forma que os elementos que o compõem se adaptem, automaticamente, à largura de tela do dispositivo no qual ele está sendo visualizado.

Com base no texto, assinale como verdadeiro (V) ou falso (F) as afirmações a seguir:

( ) A responsividade funciona, apenas, em *sites* acessados pelo PC.

( ) Um *site* responsivo não ajusta o seu *layout* quando acessado por um *smartphone*.

( ) O conceito de responsividade surgiu depois dos usuários acessarem a internet mais pelo *smartphone*.

( ) Responsividade está ligada, diretamente, com a UX e a UI.

Assinale a alternativa que apresenta a sequência correta:

a) V, V, V, F.

b) V, V, F, F.

c) V, V, V, V.

d) F, F, F, V.

e) F, F, V, V.

Resposta

O *Responsive Web Design* é uma das soluções técnicas para esse problema: programar um *site* de forma que os elementos que o compõem se adaptem, automaticamente, à largura de tela do dispositivo no qual ele está sendo visualizado.

Com base no texto, assinale como verdadeiro (V) ou falso (F) as afirmações a seguir:

(F) A responsividade funciona, apenas, em *sites* acessados pelo PC.

(F) Um *site* responsivo não ajusta o seu *layout* quando acessado por um *smartphone*.

(V) O conceito de responsividade surgiu depois dos usuários acessarem a internet mais pelo *smartphone*.

(V) Responsividade está ligada, diretamente, com a UX e a UI.

Assinale a alternativa que apresenta a sequência correta:

a) V, V, V, F.

b) V, V, F, F.

c) V, V, V, V.

d) F, F, F, V.

e) F, F, V, V.

Referências

- BALDUÍNO, P. *Dominando JavaScript com JQuery*. São Paulo: Casa do Código, 2013.
- FREEMAN, E. T.; ROBSON, E. *Use a cabeça!*: programação JavaScript. Rio de Janeiro: Alta Books, 2016.

ATÉ A PRÓXIMA!