

Laboratório 02

Praticando o desenvolvimento de páginas HTML.

Visando exercitar habilidades importantes para o desenvolvimento de uma aplicação web, vamos exercitar o envio de uma página simples em HTML personalizada para o Github e simular um processo de exibição para o público através de uma screencast.

HTML é uma linguagem de marcação amplamente utilizada para confecção de páginas web. Além dela, temos o CSS, que é uma linguagem de folha de estilos para apresentação de documentos escritos em uma linguagem de marcação, que no caso é HTML. Nesse sentido, poderíamos dizer que CSS aumenta as possibilidades do que podemos fazer com relação à personalização. Por fim, temos frameworks que colabora para construção de páginas ainda mais elaboradas. **Para essa atividade, seguem algumas sugestões de ferramentas e frameworks:**

AngularJS 1: <https://angularjs.org/>

Bootstrap: <http://getbootstrap.com/>

Mdlite: <https://getmdl.io/index.html>

Javascript (<https://www.javascript.com/>)

Sugestão de site para estudar: <http://w3schools.bootcss.com/default.html>

Observação: caso ache conveniente utilizar alguma outra ferramenta não listada aqui, sinta-se livre para fazê-lo.

IMPORTANTE: os links para importação no cabeçalho estão disponíveis nos respectivos sites das ferramentas.

Introdução básica ao uso das ferramentas e tecnologias recomendadas



Aqui daremos uma noção básica de como fazer funcionar uma aplicação. Com relação à HTML, basta abrir um editor de texto e salvá-lo com a extensão **‘.html’**. Além disso, para que esse arquivo funcione de fato como se deve e seja reconhecido como uma página, é preciso colocar as tags básicas. Veja na imagem a seguir um exemplo, que foi chamado de **“helloWorld.html”**.

```
1 <html>
2
3   <title> Título que aparece na aba </title>
4
5   <head>
6     Aqui ficam os cabeçalhos.
7
8     <meta charset="UTF-8"/>
9
10    <script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.4.8/angular.min.js"></script>
11
12    <script src="http://maxcdn.bootstrapcdn.com/bootstrap/3.3.6/js/bootstrap.min.js"></script>
13
14    <script src='controller.js'> </script>
15
16    <link rel="stylesheet" href="http://maxcdn.bootstrapcdn.com/bootstrap/3.3.6/css/bootstrap.min.css">
17
18    <link rel="stylesheet" type="text/css" href="component.css">
19
20  </head>
21
22  <body>
23    <h1> Aqui fica o corpo do documento.</h1>
24  </body>
25
26 </html>
```



Todo o código html estará entre as tags **<html>** e **</html>**. Dentro delas podemos ter diversos elementos, sendo os mais básicos:

- O **<title>** e **</title>**, que representa o título que a página terá na aba;
- O **<head>** e **</head>**, que representa toda a parte de cabeçalhos e é aqui que colocaremos nossos imports, os quais são úteis para chamarmos:
 - O suporte ao sistema de encoding utf-8 (**linha 7**) que permite inserir letras acentuadas e caracteres do gênero;
 - Nossos arquivos Javascript locais (**linha 14**) e externos referentes a alguma ferramenta (**linhas 10 – angularjs - e 12 – bootstrap**);
 - Nossos CSS locais (**linha 18**) e externos referentes a alguma ferramenta (**linha 16 - bootstrap**);
- O **<body>** **</body>** que é o corpo no qual será preenchido com os elementos de interesse;

Observação: note que chamamos nosso arquivo Javascript local diretamente através de **src="controller.js"**, pelo fato dele está no mesmo diretório que o nosso arquivo html.

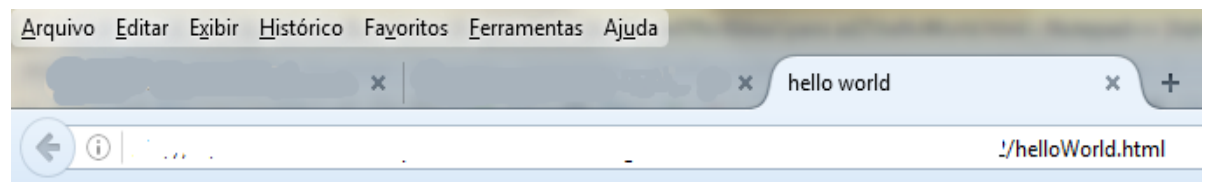
 controller	04/11/2016 21:12	Arquivo de script do JScript	1 KB
 helloWorld	11/11/2016 16:05	Firefox HTML Document	1 KB

Porém, podemos coloca-lo em uma pasta de arquivos específica para esse tipo de arquivo, bastando agora especificar o diretório dele antes do nome. Por exemplo, digamos que os arquivos estejam na pasta **js**, então bastaria fazer **src="js/arquivo.js"** e teríamos o mesmo funcionamento anterior, porém agora mais organizado. Veja a imagem a seguir:

 helloWorld	11/11/2016 16:05	Firefox HTML Document	1 KB
 js	11/11/2016 16:38	Pasta de arquivos	

O mesmo comentário poderia ser aplicado ao CSS. Além disso, é válido ressaltar que esse procedimento de colocar os arquivos de mesmo tipo em uma mesma pasta é vital para organização, facilitando o processo de desenvolvimento.

A seguir, a demonstração de uma página simples em html.

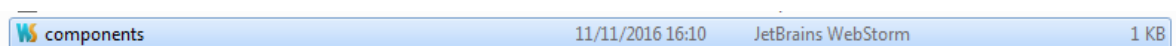


Primeira página em HTML.

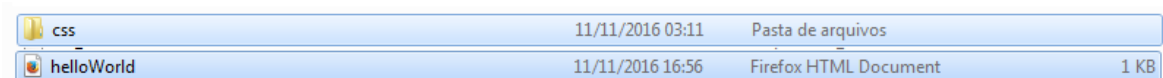
Aqui está o código da respectiva página:

```
1 <html>
2
3     <title> hello world </title>
4
5     <head>
6         <meta charset="UTF-8"/>
7     </head>
8
9     <body>
10         <h1> Primeira página em HTML.</h1>
11     </body>
12
13 </html>
14
15
16
```

Vamos dar continuidade e abordar agora a personalização com CSS. Porém, iremos organizar no código separando a folha de estilos do código html, bem como do código javascript. Para isso, precisamos criar um arquivo com a extensão '.css' e chama-lo dentro da nossa página, como foi mostrado anteriormente. Para cria-lo, basta abrir um editor de texto e salvar o seu arquivo com o nome que desejar e a extensão '.css'. Por exemplo, digamos que você escolhe como nome para seu arquivo 'components'. Então, na hora da criação digite "*components.css*". Uma vez criado seu arquivo CSS, vamos fazer um pequeno teste. Observe a imagem abaixo que mostra o arquivo criado:



Como fizemos com os arquivos Javascript, é adequado que coloquemos os de mesmo tipo em uma mesma pasta. Assim, teremos o seguinte:



Agora, vamos chamar nosso arquivo “components.css” dentro do html . Segue uma imagem de como deve ficar:

```
<html>

  <title> hello world </title>

  <head>
    <meta charset="UTF-8"/>
    <link rel="stylesheet" type="text/css" href="css/components.css" />
  </head>

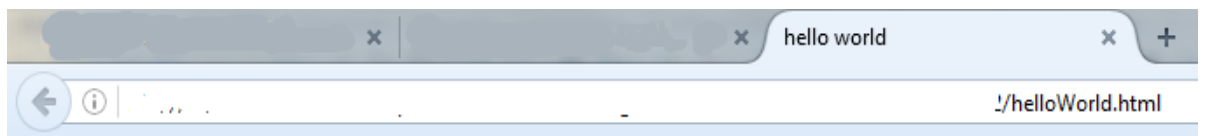
  <body>
    <h1> Primeira página em HTML.</h1>
  </body>

</html>
```

Como o nosso arquivo “components.css” está em branco, nossa página html não sofrerá qualquer alteração. Porém, vamos inserir o seguinte trecho de código para alterar a cor do background: `body { background-color: rgb(0,121,192); }`

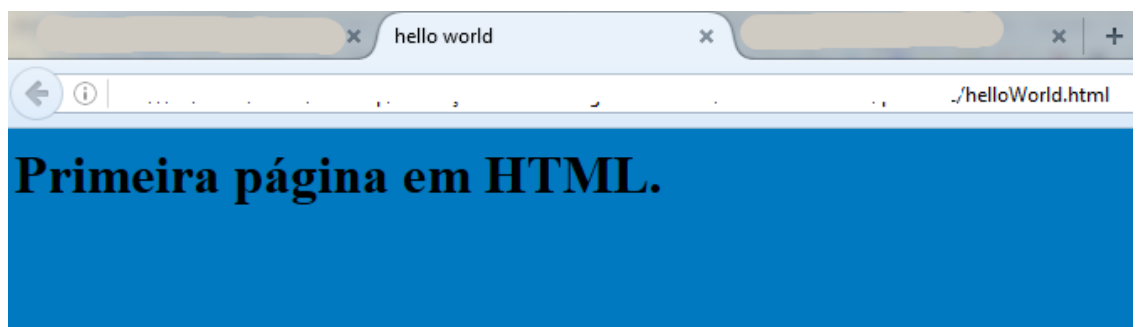
Vejamos o resultado do antes e depois desse trecho.

Imagem mostrando o antes de o trecho ser inserido no arquivo “components.css”



Primeira página em HTML.

Imagem mostrando o depois de o trecho ser inserido no arquivo “components.css”

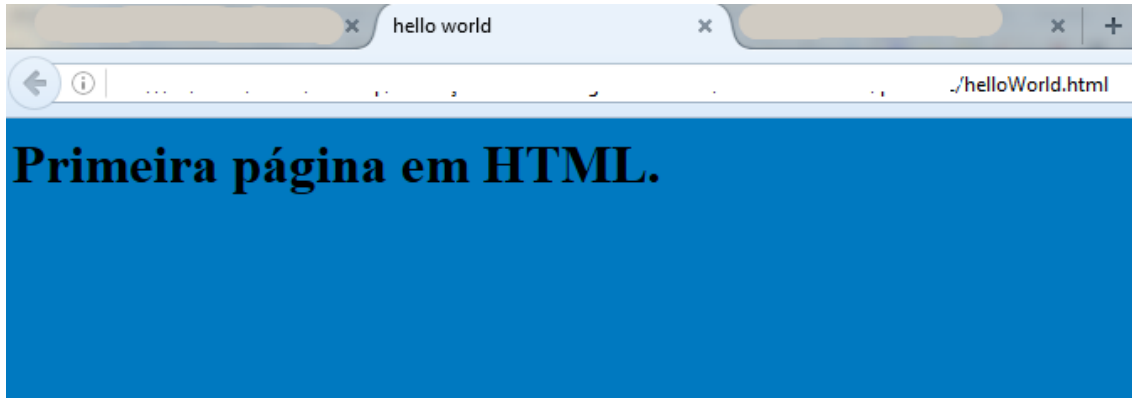


Javascript

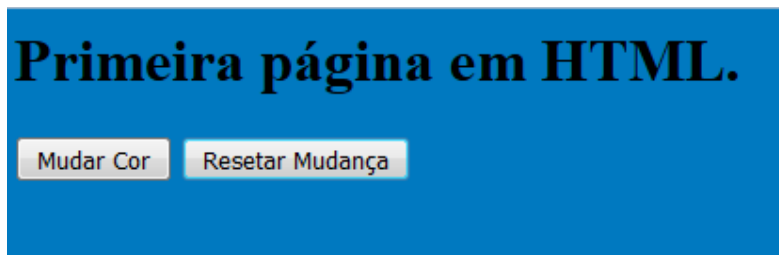
Neste seguimento, abordaremos algo parecido com o processo para um arquivo CSS, mas agora iremos tratar de um arquivo do tipo Javascript (cuja extensão é '.js').

Iremos simular o seguinte evento ao clicar em um botão: alterar a cor do background da página.

Inicialmente, a nossa página está da seguinte forma:



Agora vamos adicionar os nossos dois botões à página, mas eles ainda não funcionam. Vejamos:



Devemos agora adicionar uma funcionalidade aos botões; antes, porém, é adequado criar o nosso arquivo com a extensão '.js' que estará com a nossa função de mudança de cor e de desfazer essa mudança. Ele receberá o nome de **'funcoes.js'**.

O arquivo em questão foi criado na pasta específica para esses arquivos, que pode ser vista a seguir:



O código da função que precisamos é o seguinte:

```
function changeColor(color) {  
  
    document.body.style.background = color;  
  
}
```

Agora, precisamos chamar o nosso arquivo dentro da nossa página html e faremos isso inserindo `<script src="js/funcoes.js"></script>` dentro do cabeçalho da página da seguinte forma:

```
<head>  
  <meta charset="UTF-8"/>  
  <script src="js/funcoes.js"></script>  
  <link rel="stylesheet" type="text/css" href="css/components.css" />  
</head>
```

Além disso, também adicionaremos o evento que desejamos ao nosso botão.

```
<body>
  <h1> Primeira página em HTML.</h1>

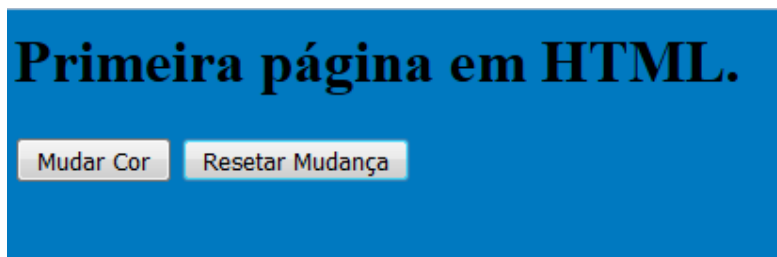
  <input type="button" onClick="changeColor('yellow')" value="Mudar Cor" />
  <input type="button" onClick="changeColor('')" value="Resetar Mudança" />
</body>
```

Note que tanto para a mudança quanto para desfazê-la usamos a mesma função. O que muda é apenas o parâmetro passado para ela. Vejamos agora o resultado ao clicar nos botões.

Clicando em 'Mudar Cor':



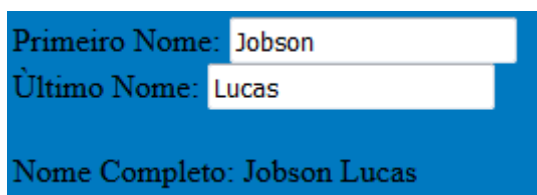
Clicando em 'Resetar Mudança':



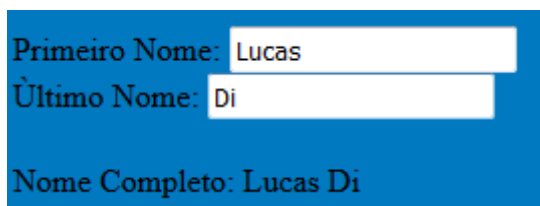
AngularJS

Aqui vamos usar elementos que são cruciais quando pretendemos desenvolver alguma aplicação utilizando o framework AngularJS. O primeiro conceito que temos é o de **módulo** (ele, por definição, define um aplicativo), sendo que este é uma espécie de container para os vários componentes da aplicação, como por exemplo, os **controllers** (**estes sempre pertencentes a algum módulo**), que controlam os dados em uma aplicação AngularJS.

Agora, vamos a um exemplo de como seria uma aplicação simples em AngularJS que recebe dois nomes em dois inputs diferentes e os mostra juntos. Por default, você pode colocar um nome para sempre aparecer que a página for aberta. Vejamos como seria:



A vantagem de usar AngularJS nesse aspecto, por exemplo, é de mudar dinamicamente a medida que o form é preenchido com outro input. Vejamos a seguir um exemplo:



Primeiro Nome: Lucas
Último Nome: Di
Nome Completo: Lucas Di

Pois bem, agora vejamos a seguir o código por trás disso e o que fundamenta esse funcionamento. Antes disso, porém, devemos levar em consideração que devemos criar um arquivo para conter o nosso código Angularjs. Devemos colocar ele na pasta 'js' criada anteriormente por questões de organização. No caso do exemplo em tela, chamei-o de 'app.js'.

Primeiro, vejamos o código do nosso arquivo 'app.js'.

```
1 var app = angular.module('myApp', []);  
2 app.controller('controller', function($scope) {  
3     $scope.firstName = "Jobson";  
4     $scope.lastName = "Lucas";  
5 });
```

Agora, vejamos fica nossa página compatível com o framework AngularJS.

```
1 <html>  
2  
3     <title> hello world </title>  
4  
5     <head>  
6         <meta charset="UTF-8"/>  
7         <script src="js/funcoes.js"></script>  
8         <script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.4.8/angular.min.js"></script>  
9         <script src="js/app.js"></script>  
10        <link rel="stylesheet" type="text/css" href="css/components.css" />  
11    </head>  
12  
13    <body>  
14        <div ng-app="myApp" ng-controller="controller">  
15  
16            Primeiro Nome: <input type="text" ng-model="firstName"><br>  
17            Último Nome: <input type="text" ng-model="lastName"><br>  
18            <br>  
19            Nome Completo: {{firstName + " " + lastName}}  
20  
21        </div>  
22    </body>  
23</html>
```

Perceba que na linha 14 nós especificamos que o controller de interesse é aquele que está em nosso arquivo 'app.js'. Desse modo, podemos ter acesso a todos os dados que ele controla, como o Primeiro Nome e o Último. Assim, quando você digita um valor na caixa de input, ele atualiza automaticamente o output depois de "Nome Completo:".

Bootstrap

Inicialmente, como nos exemplos anteriores, a primeira coisa a ser feita é dar suporte ao uso do framework Bootstrap. Para fazermos isso, olhemos o exemplo abaixo que de um form de email e senha, bem como seu respectivo código.

Formulário de email e senha básico

Email:

Por favor, digite um endereço de e-mail.

☐ Lembrar me

Enviar

O respectivo código está a seguir:

```
1 <html>
2   <title> hello world </title>
3
4   <head>
5     <meta charset="UTF-8"/>
6     <link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css">
7     <script src="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/js/bootstrap.min.js"></script>
8   </head>
9
10  <body>
11    <div class="container">
12      <h2>Formulário de email e senha básico</h2>
13      <form>
14        <div class="form-group">
15          <label for="email">Email:</label>
16          <input type="email" class="form-control" placeholder="Enter email">
17        </div>
18        <div class="form-group">
19          <label for="pwd">Password:</label>
20          <input type="password" class="form-control" placeholder="Enter password">
21        </div>
22        <div class="checkbox">
23          <label><input type="checkbox"> Lembrar me</label>
24        </div>
25        <button type="submit" class="btn btn-default">Enviar</button>
26      </form>
27    </div>
28  </body>
29 </html>
```

Perceba que as **linhas 6 e 7** são as responsáveis por incluírem os arquivos que dão suporte ao uso do framework que precisamos.

Como podemos observar, ao especificar que o tipo do input é um email, ele já impõe que devemos digitar um válido para podermos prosseguir com nossas atividades. O mesmo poderia ser dito da senha.

Importante: não se limite a aprender HTML, CSS, Javascript e outras ferramentas apenas para fazer a atividade em questão. Tenha em mente que ela faz parte de uma preparação para o projeto ao final da disciplina. Portanto, é recomendado que os estudos sobre essas linguagens e ferramentas de interesse comecem imediatamente, e que este estudo seja constante e aprofundado.

Nesse sentido, o que é requerido do aluno é a seguinte atividade:

1. Construa uma agenda de tarefas (to do list) em que seja possível adicionar e remover tarefas, bem como, à medida que for realizando as tarefas, verificar o progresso, expressas em termos de porcentagem, ou de quantas tarefas foram cumpridas e quantas ainda faltam, ou mesmo excluir a tarefa da lista ao marcar. Além disso, caso desmarque uma tarefa, ela deverá voltar ao grupo das que ainda não foram completadas.
2. Você deve personalizar a página: trocar cores, família das fontes, tamanho das fontes, colocar menu, efeitos no hover, mudança quando a tarefa for marcada como concluída etc.
3. A página deve iniciar com algumas tarefas pré-definidas (pelo menos três), isto é, que já constam na página sem precisar adicionar (sugestão: caso use angularjs, é possível fazer isso usando o controller).
4. Tente ao máximo evitar a duplicação de código (sugestão: para o caso de usar o angularjs, opte pelo ng-repeat quando for repetir um determinado padrão para um conjunto de elementos). Além disso, atente para a qualidade da nomenclatura que utiliza (**use nomes significativos!!!**).
5. Faça a separação dos códigos: página HTML em um arquivo (já que neste lab é apenas uma página); Javascript em outro e o(s) CSS's em outro(s).
6. **Lembre-se do suporte ao UTF-8.**
7. **PARA ENTREGA:** Faça uma screencast (vídeo) demonstrando a aplicação e as funcionalidades solicitadas (50% da nota será baseada nessa demonstração) e coloque a atividade no github em um repositório público, tendo em vista que o restante da nota será com base na análise do código, sendo observados, principalmente, os aspectos qualitativos e organizacionais, tais como a separação dos arquivos HTML, CSS e JS, além da utilização de nomes significativos.
8. **Tanto para o vídeo quanto para o repositório, disponibilize os respectivos links de acesso no envio da atividade.**
9. Aqui está um vídeo para inspiração: <http://bit.ly/2foudbS>