|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Natureza da Avaliação | Valores | | Disciplina: Programação Web | Curso | Identificação da Turma | | |
| Valor | Nota | Turno | Período | Turma |
| Passo a Passo |  | - | Professor(a): Priscila Nunes | SIS INF | Noturno |  |  |

# **Aluno(a):** \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Data: \_\_\_/\_\_\_/\_\_\_\_

1. **Hotsite com Bootstrap e jQuery**

Neste passo a passo vamos criar um layout de um hotsite no modelo One Page (uma única página) utilizando HTML, CSS, Bootstrap e jQuery, além do iframe do Google Maps.

O layout One Page é um conceito de web design, que visa a exibição de todo o conteúdo do site em uma única página, removendo a necessidade de navegar entre múltiplas páginas, o que é excelente para sites de pequenos ou médio porte.

Esse modelo de layout difere do modelo Single Page, que também trabalha com uma única página, mas tem o seu conteúdo carregado dinamicamente, de acordo com a interação do usuário. No modelo One Page o conteúdo completo do site já está disponível desde o carregamento da página.

Na criação do layout utilizaremos os seguintes componentes do Bootstrap:

* Navbar;
* Tabs;
* List Group;
* Thumbnail;
* Input Group;
* Modal.
* Além dos plugins nativos do framework:
* Affix;
* ScrollSpy;
* Tooltip.

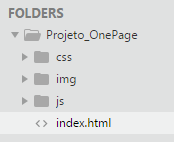
O resultado final do nosso exemplo pode ser visto a seguir: [Link](site_pronto.html)

Links úteis:

* Download do [Bootstrap](http://getbootstrap.com/getting-started/#download) - <https://getbootstrap.com/docs/4.3/getting-started/introduction/>
* Download do [jQuery](https://jquery.com/download/) - <https://jquery.com/download/>

1. **Estrutura do Projeto**

A estrutura do projeto, que é composta por **index.html** e três pastas: **css**, **img** e **js**, como mostra a **Figura 1.**

****  
**Figura 1**. Estrutura do projeto

O index.html é o arquivo principal do projeto, onde será montado o HTML;

A pasta **CSS** conterá os arquivos **CSS** que utilizaremos no projeto: ***bootstrap.min.css*** e ***style.css***

A pasta **IMG** conterá as imagens que utilizaremos no projeto;

A pasta **JS** conterá os arquivos **JavaScript** que utilizaremos no projeto: ***bootstrap.min.js***, ***jquery-3.2.1.min.js*** e ***script.js***.

A estrutura do layout será inicialmente dividida em três partes: topo, corpo e rodapé.

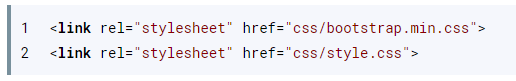
Veja que o topo do layout que é dividido em duas partes: o banner e a barra de navegação. Além disso, o corpo que é composto de cinco seções, cada uma focada em área do layout.

|  |
| --- |
|  |

Veja que o topo do layout que é dividido em duas partes: o banner e a barra de navegação. Além disso, o corpo que é composto de cinco seções, cada uma focada em área do layout.

1. **Estrutura do HTML**

No cabeçalho da página referenciamos as folhas de estilos que iremos utilizar no nosso exemplo:

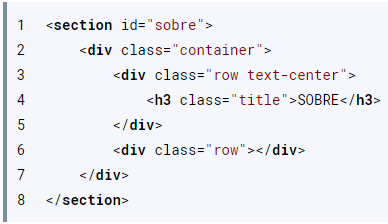


* **Linha 1**: inserimos a folha de estilo do Bootstrap;
* **Linha 2**: inserimos a nossa folha de estilo.

Dividimos o conteúdo da página em três partes utilizando as tags <header>,  <main>  e  <footer>:

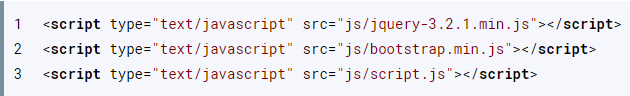


* **Linhas 3 a 7**: Dividimos o conteúdo do main criando cinco <section>, uma para cada seção do layout com seu respectivo id: sobre, agenda, palestrantes, local e assine.



* **Linha 1**: definimos o id da seção;
* **Linha 2**: adicionamos uma <div> com a classe container do Bootstrap, que define uma largura máxima ao conteúdo de acordo com a resolução da tela;
* **Linha 3**: com uma <div> utilizando a classe row criamos uma linha no sistema de grid do Bootstrap, ocupando toda a largura do elemento. E com a classe text-center, centralizamos o alinhamento dos textos no elemento;
* **Linha 4**: utilizamos um <h3> com a classe title para definir o título da seção;
* **Linha 6**: adicionamos uma outra <div> com a classe row que conterá o conteúdo da seção.

No final do <body>, incluímos os scripts que utilizaremos no exemplo:



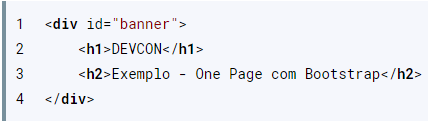
* **Linha 1**: inserimos o arquivo minificado do jQuery;
* **Linha 2**: inserimos o arquivo minificado do Bootstrap;
* **Linha 3**: inserimos o nosso arquivo JavaScript;

Os arquivos possuem dependência um do outro, então devem ser carregados nesta exata ordem.

1. **Criação do Baner**

No cabeçalho da página referenciamos as folhas de estilos que iremos utilizar no nosso exemplo:

Dentro do <header> adicionamos o seguinte bloco de código:



* **Linha 1:** adicionamos uma <div> com o id banner, que iremos utilizar como o banner da página;
* **Linha 2:** definimos o título da página utilizando um <h1>;
* **Linha 3:** utilizamos um <h3> para criar um subtítulo da página.

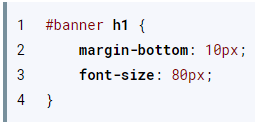
O banner ocupa toda a largura e altura da tela e possui uma imagem de fundo que se ajusta ao seu tamanho, além de um título e um subtítulo alinhados ao centro do elemento. Para definir essa aparência precisamos modificar o arquivo style.css.

No bloco abaixo estamos definindo a aparência do banner:



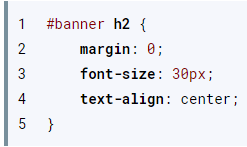
* Linha 2: com a propriedade min-height: 100vh definimos uma altura mínima do banner para 100vh (view height), que significa que o banner terá 100% da altura da tela;
* Linha 3: adicionamos a propriedade box-sizing: border-box, fazendo com o que no cálculo da altura do banner leve em conta o tamanho da borda e do padding;
* Linha 4: adicionamos um padding de 20px para afastar o conteúdo interno da borda;
* Linha 5: definimos uma cor para a fonte;
* Linha 6: com a propriedade background-image definimos a imagem de fundo passando o valor url(“”) com o caminho da imagem;
* Linha 7: utilizamos a propriedade background-size: cover para que a imagem de fundo cubra toda a área do banner proporcionalmente;
* Linha 8: posicionamos a imagem do fundo ao centro do banner;
* Linhas 9 a 10: definimos uma cor para o fundo e utilizamos a propriedade background-blend-mode com o valor multiply para misturar a cor com a imagem de fundo;
* Linha 11: dividimos área do banner pela quantidade de elementos dentro dele utilizando a propriedade display: flex;
* Linha 12: mudamos a orientação da divisão do espaço de horizontal para vertical utilizando a propriedade flex-direction: column;
* Linhas 13 e 14: alinhamos o conteúdo ao centro do banner com as propriedades justify-content e align-items.

Agora modificamos a aparência do título do banner com o bloco abaixo:



* Linha 2: adicionamos uma margem inferior de 10px;
* Linha 3: definimos uma fonte de 80px.

E no bloco abaixo definimos uma aparência para o subtítulo do banner:



* Linha 1: zeramos a margem do subtítulo do banner;
* Linha 2: definimos uma fonte de 30px;
* Linha 3: alinhamos o texto do subtítulo ao centro.

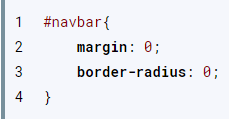
1. **Adicionando a Navbar**

Criaremos a barra de navegação do exemplo utilizando o componente navbar do Bootstrap:

|  |
| --- |
|  |

* **Linha 1**: criamos o componente navbar utilizando uma <nav> com a classe navbar. Adicionamos a classe navbar-inverse para mudar o tema e definimos um id para a navbar;
* **Linha 3**: adicionamos uma <div> com a classe navbar-header, que serve para agrupar o conteúdo da navbar que ficará sempre visível.
* **Linha 4**: criamos um botão com a classe navbar-toggle que define sua aparência. A classe collapsed indica que o botão será iniciado oculto. Com o atributo data-toggle foi definida a ação que o botão irá fazer quando clicado, e com o atributo data-target o elemento sofrerá a ação.
* **Linha 6 a 8**: definimos a aparência do ícone do botão com três tags <span> com a classe icon-bar, formando o ícone do menu hambúrguer.
* **Linha 10**: definimos o link principal da página com a classe navbar-brand e criamos uma âncora para o banner.
* **Linha 12**: adicionamos uma <div> com a classe navbar-collapse, que serve para agrupar todo o conteúdo da navbar que será ocultado em resoluções mobile, formando um menu hambúrguer. Com a classe collapse ocultamos essa div em resoluções mobile. E definimos um id para esse grupo de elementos.
* **Linha 13**: criamos uma lista de links. A classe nav serve como base de uma lista de links em uma barra de navegação e a classe navbar-nav define a aparência da lista na navbar. E por último, a classe navbar-right coloca o elemento na ponta direita da navbar;
* **Linhas 14 a 18**: criamos uma lista de links ancorando para cada seção do layout.

No arquivo style.css, iremos adaptar a aparência da navbar ao layout do exemplo, adicionando o bloco abaixo:

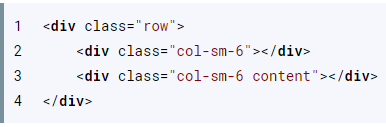


* **Linhas 2 e 3**: removemos a margem padrão da navbar e as bordas arredondadas.

1. **A seção Sobre**

Essa seção é dividida em duas partes: uma parte com o componente carousel do Bootstrap e outra com uma descrição do site.

Dentro da segunda row da seção “sobre” dividimos o conteúdo em duas colunas utilizando o código abaixo:



* **Linha 1**: definimos uma linha no sistema de grid do Bootstrap utilizando a classe row;
* **Linha 2**: utilizando a classe col-sm-6, definimos uma coluna que ocupa o espaço de seis de 12 colunas na grid enquanto a largura da tela for maior ou igual a 769px;
* **Linha 3**: definimos a classe content para a coluna que conterá a descrição da seção.

Dentro da primeira coluna implementamos o carousel com o código abaixo:



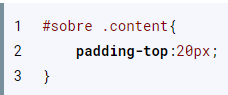
* **Linha 1**: criamos o carousel utilizando a classe carousel, que é a base do componente carousel, e a classe slide que serve para colocar uma animação durante a transição de cada item do carousel. O atributo data-ride diz para o carousel executar a animação após a página terminar de ser carregada. Além disso, adicionamos um id com o valor carousel-sobrepara identificar o carousel;.
* **Linha 2**: criamos a <div> que conterá cada item do carousel, utilizando a classe carousel-inner;
* **Linha 3 a 11**: criamos cada item do carousel, compostos por uma <div> com a classe item, contendo uma imagem, além da classe active para definir o item exibido ao carregar o carousel;
* **Linha 13**: utilizando um link com a classe carousel control, criamos um botão para transitar entre os itens do carousel, e com a classe left definimos a posição dele no lado esquerdo do carousel. O atributo href ponta para qual carousel que ele irá manipular e com o atributo data-slide com o valor prev, fará com que o carousel exiba o item anterior;
* **Linha 14**: definimos o ícone do link utilizando um <span> com a classe icon-prev, criando uma seta para esquerda;
* **Linha 16**: com a classe **right** definimos a posição link na direta do carousel, e com o atributo data-slide com o valor next, definimos que o carousel deve avançar para o próximo item;
* **Linha 17**: definimos o ícone do link utilizando um <span> com a classe icon-next, criando uma seta para direita.

No arquivo style.css definiremos a aparência das seções e do conteúdo da seção sobre.

Abaixo iremos definir um estilo padrão para todas as seções do layout:



* **Linhas 1 a 4**: definimos a aparência de cada seção utilizando a propriedade box-sizing, além de um padding para afastar o conteúdo interno da bordas ;
* **Linhas 5 a 7**: utilizando o seletor section:nth-child(odd) selecionamos todas as seções ímpares, e com a propriedade background-color definimos uma cor de fundo, assim alternando a cor de cada seção;
* **Linhas 8 a 13**: criamos a aparência do título de cada seção, definindo o display para inline-blockborder-bottom definindo um risco abaixo do título.
* Com o bloco abaixo modificamos a aparência do conteúdo da descrição:



1. **A seção Agenda**

Na criação da seção utilizamos:

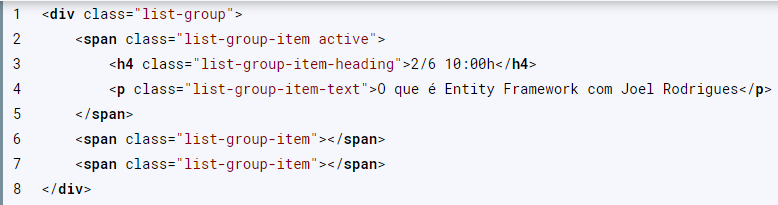
* componente tabs do Bootstrap: separa a agenda em abas, cada uma com um tema;
* componente list group: monta uma lista para cada tema com o horário e a descrição de cada palestra.

Dentro da segunda linha da seção agenda, adicionamos o bloco abaixo:



* **Linha 1**: adicionamos uma <div> com a classe col-sm-12, para definir uma coluna que ocupa o espaço de 12 das 12 colunas do sistema de grid do Bootstrap.
* **Linha 2**: definimos uma lista de links com a classe nav e nav-tabs agrupando todas as abas;
* **Linhas 3 a 6**: criamos cada aba, e utilizando o href do link, associamos a aba ao seu conteúdo. Com o atributo data-toggle, iremos exibir o conteúdo da aba quando clicada;
* **Linha 8**: utilizamos uma <div> com a classe tab-content definimos o elemento que agrupa o conteúdo de todas as tabelas.
* **Linhas 9 a 12**: definimos o conteúdo de cada aba utilizando uma <div> com a classe tab-pane e para identificar o conteúdo adicionamos um atributo id.

No bloco abaixo criamos a estrutura das listas com os horários de cada palestra dentro de cada tab-panel:

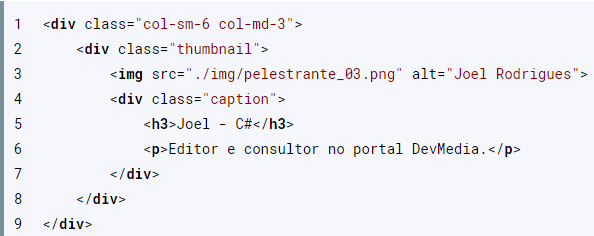


* **Linha 1**: iniciamos o componente list-group utilizando uma <div> com a classe list-group;
* **Linha 2**: definimos um item da lista utilizando a classe list-group-item, e a destacamos utilizando a classe active;
* **Linha 3**: utilizando um <h4> com a classe list-group-item-heading definimos o cabeçalho do item da lista, onde exibimos a data com o horário;
* **Linha 4**: utilizando um <p> com a classe list-group-item-text definimos a descrição do item;
* **Linhas 6 e 7**: adicionamos mais dois itens que representam as demais palestras do tema, e que contêm a mesma estrutura da anterior.

1. **A seção Palestrantes**

Utilizaremos o componente thumbnail do Bootstrap para definir a aparência do bloco de cada palestrante.

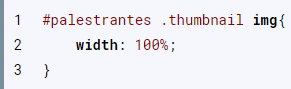
Dividimos o conteúdo da seção palestrantes em quatro colunas, e cada uma delas é composta do código abaixo:



* **Linha 1**: utilizando a classe col-md-3, definimos uma coluna que ocupa o espaço de três de 12 colunas do sistema de grid do Bootstrap, enquanto a largura da tela for maior ou igual a 992px. Com a classe col-sm-6 modifica-se a coluna para ocupar o espaço de seis de 12 colunas da grid quando a largura da tela for maior ou igual a 768px.
* **Linha 2**: adicionamos o componente thumbnail utilizando uma <div> com a classe thumbnail;
* **Linha 3**: adicionamos a imagem do palestrante ao thumbnail;
* **Linhas 4 a 7**: criamos a descrição do thumbnail utilizando uma <div> com a classe caption

O componente thumbnail ajusta o tamanho da imagem a partir da altura, mas para o nosso layout iremos modificar para que o tamanho da imagem seja baseado a partir da largura do thumbnail.

No arquivo style.css adicionaremos o seguinte bloco:



* **Linha 2**: definimos a largura da imagem para 100%.

1. **A seção Como Chegar**

Nesta seção incorporamos o iframe do Google Maps, que utilizaremos para exibir a nossa localização no site.

Para implementá-lo precisamos gerar o código do iframe no site do Google Map, acessando o link <https://www.google.com.br/maps>.

Após acessar o site iremos digitar o endereço que queremos exibir na barra de busca, como mostra a **Figura 1**:

**Figura 1**. Barra de busca

Após obter a localização, basta clicar no botão compartilhar, como mostra a **Figura 2**:

**Figura 2**. Botão compartilhar

Clique no botão incorporar mapa para copiar o código do iframe disponibilizado, como mostra a **Figura 3**:

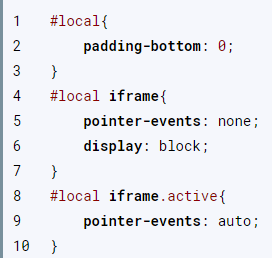
**Figura 3**. Código do iframe

Após obter o código adicionamos o iframe dentro da seção local:



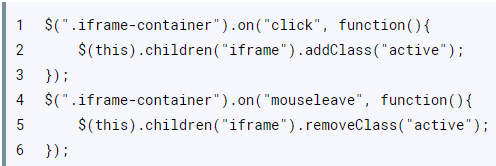
* **Linha 1**: utilizamos a classe container-fluid no lugar da classe container para que o conteúdo utilize toda largura disponível;
* **Linha 5**: adicionamos a classe iframe-container para identificar qual elemento está envolvendo o mapa;
* **Linha 6**: incorporamos o iframe do Google Maps retirando os atributos que definem a altura, a largura e a borda padrão do iframe.

No arquivo style.css modificamos a aparência da seção local e do iframe:



* **Linha 2**: zeramos o padding inferior que é definido como padrão para todas as seções, na seção local;
* **Linha 5**: com a propriedade pointer-events: none removemos as interações do mouse no mapa para que ele não seja modificado por acidente durante a navegação;
* **Linha 6**: definimos um display: block;
* **Linha 9**: com a propriedade pointer-events: auto retornamos ao padrão as interações do mouse no mapa quando o iframe tiver a classe active.

No arquivo script.js adicionamos o bloco abaixo para habilitar e desabilitar a interação do mouse sobre o mapa. Com ele habilitamos as interações com o mapa após clicarmos no iframe-container e desabilitamos após retirarmos o mouse de cima do mapa:

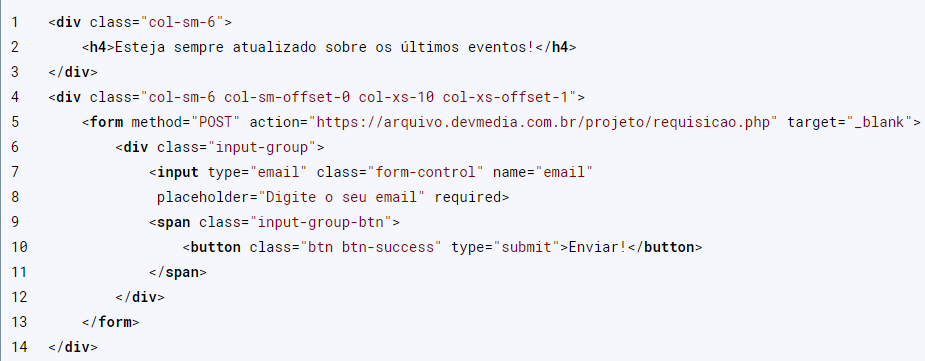


* **Linha 1**: utilizamos a função on do jQuery para escutar o evento de click no elemento com a classe .iframe-container;
* **Linha 2**: utilizando a função children buscamos pelo elemento filho com a tag <iframe> do elemento que disparou o evento de clique, representado pela palavra **this**, e então adicionamos a classe active utilizando a função addClass;
* **Linha 4:** utilizando a função on escutamos pelo evento mouseleave, que é disparado quando o mouse deixa a área do elemento com a classe iframe-container;
* **Linha 5**: buscamos pelo <iframe> dentro do elemento que disparou o evento mouseleave utilizando a função children, e então removemos a classe active com a função removeClass.

1. **A seção Fique por dentro**

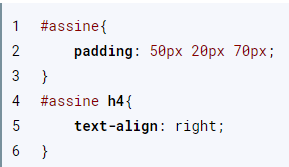
O conteúdo dela foi dividido em duas partes: uma mensagem e um formulário de cadastro de e-mail onde utilizamos o input group do Bootstrap.

No código abaixo dividimos a área em duas colunas:



* **Linha 1**: criamos a primeira coluna utilizando uma <div> com a classe col-sm-6, fazendo com que essa coluna ocupe o espaço de seis de 12 colunas do sistema de grid do Bootstrap;
* **Linha 4**: criamos a segunda coluna utilizando a classe col-sm-6, e a classe col-xs-10 modifica a quantidade de colunas da grid que esse elemento irá ocupar para 10 em conjunto com a classe col-xs-offset-1, que define uma margem esquerda com o tamanho de uma coluna do sistema de grid enquanto a largura da tela for menor que 768px. E por último, a classe col-sm-offset-0 retorna a margem ao tamanho padrão quando o tamanho da largura da tela for maior ou igual a 768px.
* **Linha 5**: adicionamos um formulário com o método POST como target \_blank, abrindo o resultado da requisição em outra janela ou aba;
* **Linha 6**: criamos uma <div> com a classe input-group, que definirá a aparência do <input> junto com o botão;
* **Linha 7**: utilizando a classe form-control no input definimos que a largura do <input> será do tamanho de todo o espaço disponível, e com a propriedade required indicamos que esse campo é obrigatório para o envio do formulário;
* **Linha 9**: definimos um grupo de botões que terão a aparência mesclada com o <input>, utilizando um <span> com a classe input-group-btn;
* **Linha 10**: inserimos o botão que enviará o formulário. Para isso utilizamos as classes btn e btn-success para definir a aparência do botão de acordo com o Bootstrap.

No arquivo style.css definimos a aparência da seção assine como da mensagem adicionar o bloco abaixo:



* **Linha 2**: definimos um padding: 50px 20px 70px para afastar o conteúdo das bordas;
* **Linha 5**: alinhamos a direita o texto da mensagem utilizando a propriedade **text-align: right**.

1. **Criando o rodapé**

Neste rodapé temos três links sociais utilizando o plugin tooltip do Bootstrap, que exibe um balão informativo próximo ao elemento quando passamos o mouse por cima.

Dentro do <footer> adicionamos o bloco abaixo:



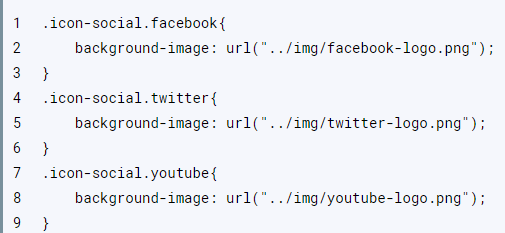
* **Linha 1**: criamos uma <div> com a classe footer-links, que utilizaremos para organizar os links sociais;
* **Linhas 2 a 7**: adicionamos três links sociais, cada um com a classe icon-social que utilizaremos para definir uma aparência padrão e uma classe relativa à rede social: YouTube, Twitter e Facebook. Além disso temos o atributo data-toggle com o valor tooltip e o data-placement indicando a posição em que o tooltip irá parecer: top, right, bottom, left. Por último temos o atributo title onde definimos a mensagem de cada tooltip;
* **Linhas 9 a 11**: incluímos um <p> com texto no final do footer.

Para definir a aparência do footer e seus elementos adicionamos o seguinte bloco ao arquivo style.css:



* **Linhas 1 a 5**: criamos a aparência do footer, definindo a cor de fundo com o background-color, uma margem interna utilizando o padding e a cor do texto;
* **Linhas 7 a 8**: utilizando a propriedade display:flex fazemos com que os links dentro desse elemento utilizem todo o espaço disponível, e com o justify-content: centeralinhamos os mesmos ao centro do elemento;
* **Linha 9**: com a propriedade flex-wrap: wrap indicamos que se algum link tem alguma propriedade que altere o seu tamanho definido, ela será respeitada criando uma quebra de linha caso algum elemento não se encaixe no tamanho disponível.
* **Linha 10**: adicionamos uma margem superior e inferior de 15px e automática nas laterais, centralizando o elemento;
* **Linhas 13 e 14**: definimos a altura e largura padrão dos links;
* **Linha 15**: modificamos o display de inline para block;
* **Linha 16**: definimos o background-size para contain, fazendo com o que a imagem de fundo se adapte a largura e altura do link.
* **Linha 17**: por último temos uma margem para afastar os links uns dos outros.

E com o código abaixo definimos a aparência de cada link social:



* **Linhas 1 a 9**: adicionamos uma imagem de fundo para cada link a partir das respectivas classes: facebook, twitter e youtube.

O plugin tooltip precisa ser inicializado manualmente, e para isso colocaremos o bloco abaixo no arquivo script.js:



* **Linhas 1**: utilizando o seletor do jQuery selecionamos os elementos com a classe icon-social e executamos a função tooltip.

1. **Formulário de Contato**

Após o <footer> adicionaremos o HTML do modal com o bloco abaixo:



* **Linha 1**: definimos o modal utilizando uma <div> com a classe modal, e a classe fade inicializa o modal fechado, além de definirmos um id para podermos identificá-lo;
* **Linha 4**: definimos o cabeçalho do modal com a classe modal-header;
* **Linha 5**: adicionamos um botão para fechar o modal, definindo a aparência com a classe close e definindo a ação ocorrida quando clicado, com o atributo data-dismiss;
* **Linha 6**: definimos o título do modal com um <h4> utilizando a classe modal-title;
* **Linha 8**: adicionamos uma <div> com a classe modal-body para conter o corpo do modal.

Dentro do modal-body adicionamos o formulário de contato com o block abaixo:



* **Linha 1**: adicionamos um formulário com o método POST como target \_blank, abrindo o resultado da requisição em outra janela ou aba;
* **Linhas 2 a 15**: criamos três campos utilizando a classe **form-control** envoltos de uma <div> com a classe form-group;
* **Linha 17**: adicionamos um botão para submeter o formulário de contato.

Para disparar a abertura do modal adicionaremos no link da barra de navegação Contato os seguintes atributos:

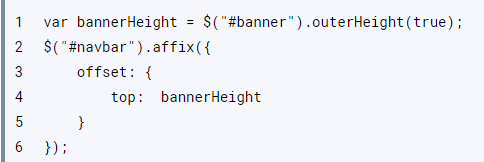


* **Linha 1**: adicionando o atributo data-toggle com o valor modal e definimos que ao ser clicado o mesmo será aberto, e o atributo data-target indica qual modal será aberto.

1. **Fixando a navbar após o scroll**

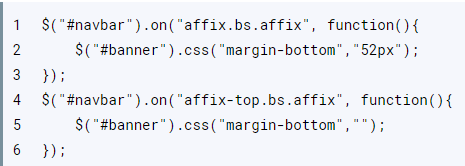
Para fazer isso utilizaremos o plugin affix do Bootstrap.

Esse plugin verifica a posição do scroll da página e a distância em relação ao início ou ao final do documento, colocando a classe affix no elemento, mudando assim a propriedade position para fixed:



* **Linha 1**: criamos a variável que irá guardar o tamanho total do banner, utilizando a função outerHeight, passando o parâmetro true, indicando que o tamanho das margens é adicionado ao cálculo do tamanho do elemento.
* **Linha 2 a 6**: utilizamos a função affix na navbar passando como parâmetro um objeto com a propriedade offset que contém a propriedade top recebendo o valor de bannerHeight.

E com o bloco abaixo iremos observar os eventos disparados antes do plugin affix fixar ou desafixar a navbar, e então adicionar ou remover uma margem para compensar o espaço da navbar no layout:



* **Linha 1**: observamos o evento affix.bs.affix que é disparado antes do plugin fixar a navbar;
* **Linha 2**: adicionamos um margin-bottom de 52px ao banner para compensar o espaço deixado pela navbar quando ela muda a posição dela para fixed.
* **Linha 4**: observamos o evento affix-top.bs.affix que é disparado antes do plugin mudar a classe affix para affix-top, retirando o position:fixed da navbar;
* **Linha 5**: retiramos o margin-bottom do banner.

Agora precisamos modificar a classe affix para ajustar a aparência da navbar quando ela estiver fixa:



* **Linha 2**: definimos a propriedade top: 0 para posicionar o navbar no topo da tela;
* **Linha 3**: definimos a largura para 100%;
* **Linha 4**: adicionamos um z-index: 20 para que nenhum outro elemento além do modal fique na frente do banner.

1. **Adicionando scrollSpy**

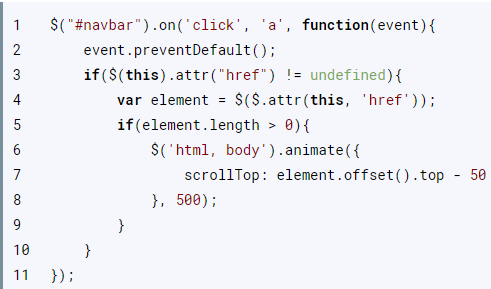
Para implementá-lo precisamos adicionar os seguintes atributos ao documento, normalmente ao <body>:



* **Linha 1**: no <body> adicionamos a classe data-spy com o valor scroll, indicando que utilizaremos o ScrollSpy. O atributo data-target define o elemento que envolve um elemento com a classe nav e que possui as âncoras para cada seção do site. Por último, o atributo data-offset delimita uma margem para o ScrollSpy calcular a posição do scroll relativa ao documento.

1. **Adicionando um scroll suave**

Uma âncora é um link apontando para alguma seção do HTML, então ao clicarmos nele somos levados à seção do site apontada, instantaneamente. Para criar um scroll suave até a seção apontada adicionamos o bloco abaixo no arquivo script.js:

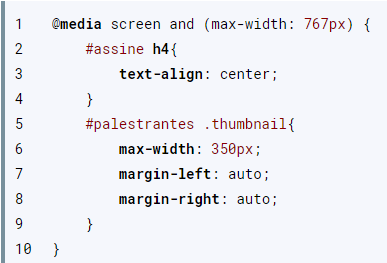


* **Linha 1**: utilizando a função on do jQuery e escutamos o evento de click no elemento <a> dentro da navbar;
* **Linha 2**: com a função event.preventDefault() evitamos que o comportamento padrão do clique no elemento ocorra;
* **Linha 3**: verificamos se o elemento clicado possui o atributo href definido;
* **Linha 4**: guardamos na variável element o elemento referenciado no atributo href;
* **Linha 5**: utilizando a comparação element.lenght > 0 verificamos se foi encontrado algum elemento;
* **Linhas 6 a 8**: executamos a função animate no <html> e no <body>, passando como parâmetro um objeto com a propriedade scrollTop com o valor element.offset().top - 50, onde element.offset().top é a posição do topo do elemento relativa ao documento e 50 é uma margem para que a seção apareça por inteira.

1. **Adicionando media query**

Para isso utilizaremos as media queries a fim de criar regras para modificar a aparência de certos elementos de acordo com o tamanho da tela.

No bloco abaixo definimos a media query que entrará em vigor enquanto a largura da tela for de até 767px:



* **Linha 1**: utilizando o media type screen definimos que essa media query será considerada para qualquer tipo de tela, e dentro dos parênteses especificamos um filtro para a media query, indicando que os blocos de estilo dentro dela entrarão em vigor enquanto a tela tiver até 767px de largura;
* **Linhas 2 a 4**: alinhamos o texto do h4 na seção com o id assine ao centro do elemento;
* **Linhas 5 a 9**: limitamos a largura máxima do thumbnail para 350px e definimos o valor automático nas margens laterais, centralizando-o.

1. **Adicionando media query**

Aqui solucionamos os problemas que aparecem quando exibimos o site em resoluções de dispositivos mobile, como o tamanho da fonte dos títulos, das imagens nos thumbnails o alinhamento do texto na seção assine.

No bloco abaixo definimos a media query que entrará em vigor enquanto a largura da tela for de até 425px:



* **Linhas 2 a 4**: diminuímos a fonte do título no banner para 60px;
* **Linhas 5 a 7**: diminuímos a fonte do subtítulo no banner para 20px;
* **Linhas 8 a 10**: diminuímos a fonte do título de cada seção para 15px;
* **Linhas 11 a 13**: diminuímos a fonte do nome do palestrante de thumbnail para 18px .