

HTML+CSS&Medias

Menu

Introdução	2
Ferramentas	3
Extensões	4
Iframe	5
< >iframe001.html	5
< >iframe002.html	6
< >iframe003.html	7
< >iframe004.html	9
< >iframe005.html	12
< >iframe006.html	13
Projeto-Social	16
Form	24
< >form001.html	24
< >form002.html	28
< >form003.html	29
< >form004.html	30
< >form005.html	31
< >form006.html	33
< >form007.html	34
< >form008.html	36
< >form009.html	38
< >form010.html	39
Media Query	40
> mq001	41
> mq002	43
> mq003	44
Mobile First	45
> mq004	45
> mq005	48
Projeto Login	54

Introdução

Para obter mais informações sobre o conteúdo, recomendo aqui as fontes de pesquisa utilizadas.

- [GitHub](#)

Acesse o conteúdo do repositório público, lá o Guanabara disponibiliza todo material usado em aula.

- [Módulo 04 - Curso em Vídeo](#)

Assista a playlist completa do curso de HTML5 e CSS3.

- [Curso de JavaScript e ECMAScript para iniciantes](#)

Confira este curso completo para aprender a programar.

- [W3scools](#)

W3Schools é um site educacional voltado ao aprendizado de tecnologias web. Seu conteúdo inclui tutoriais e referências relacionadas a HTML, CSS e JavaScript, que irão ser estudados ao longo do curso.

- [W3C](#)

É a organização responsável pelas normas técnicas da World Wide Web. As tags obsoletas e outras infor-

- [MDN Web Docs](#)

MDN Web Docs é uma fonte de documentação para desenvolvedores, mantida com o apoio de uma comunidade de desenvolvedores e escritores técnicos e hospedando muitos documentos sobre uma grande variedade de assuntos como: HTML5, JavaScript, CSS, Web APIs, Node.js, WebExtensions e MathML.



- [developer-roadmap](#)

Roteiros interativos, guias e outros conteúdos educacionais para ajudar os desenvolvedores a crescer em suas carreiras.



Ferramentas

- [VSCode](#)

O Visual Studio Code é um editor de código-fonte desenvolvido pela Microsoft para Windows, Linux e macOS.



- [Gimp](#)

GIMP (GNU Image Manipulation Program) é um programa de código aberto voltado principalmente para criação e edição de imagens, e em menor escala também para desenho vetorial.



- [HandBrake](#)

HandBrake é um programa multiplataforma e multitarefa em código aberto (licenciado em GPL) de conversão de arquivos de vídeo e DVD e Blu-ray para MPEG-4, disponível para Mac OS X, Linux e Windows.



- [Adobe Color](#)

Crie paletas de cores com a roda de cores ou imagem e navegue por milhares de combinações de cores na comunidade do Adobe Color.



- [QR Code Generator](#)

Com o QR Code Generator você pode criar QR Codes personalizados com logotipos, molduras e suas cores preferidas. Crie seu QR Code como quiser!



- [Google Fonts](#)

Google Fonts é uma biblioteca com mais de 800 fontes livres licenciadas, um diretório web interativo para navegar na biblioteca, e APIs para usar convenientemente as fontes através de CSS e Android.

A blue rectangular button with the text "Google Fonts" in white.

- [DevTools](#)

Chrome DevTools é um conjunto de ferramentas de desenvolvedor da web integradas diretamente no navegador Google Chrome.



- [Mockflow](#)

É a ferramenta líder para projetar planos de interface do usuário para sites e aplicativos.



Extensões

- [Web Developer](#)

Web Developer é uma extensão para navegadores da web baseados em Mozilla que adiciona ferramentas de edição e depuração para desenvolvedores da web.



- [ColorZilla](#)

O ColorZilla permite obter uma leitura de cores em qualquer ponto do navegador, ajustando rapidamente essa cor e colando-a em outro programa, como o Photoshop.



- [Fonts Ninja](#)

O Fonts Ninja permite capturar fontes utilizadas em sites e também oferece os detalhes como cor, tamanho, entre outros.



- [Window Resizer](#)

O Window Resizer redimensiona a janela do navegador para emular várias resoluções.



- [GoFullPage - Full Page Screen Capture](#)

Faça uma captura de tela da sua página atual de forma completa e confiável, sem solicitar permissões extras!



Iframe¹

A tag `<iframe>` especifica um quadro em linha. Um quadro em linha é usado para incorporar outro documento ao documento HTML atual.

Esse elemento é como uma porta para outros sites, portanto é necessário utilizá-lo de maneira cuidadosa para evitar sites maliciosos.

iframe001

Em [iframe001.html](#), fizemos o primeiro teste com `<iframe>`s. Ele é especialmente útil para adicionar conteúdos atraentes, como os vídeos.

Testando o uso de iframe

Acessando o site do CursoemVideo



para aprender a

programar.

1. `<body>`
2. `<h1>Testando o uso do iframe</h1>`
3. `<p>`
4. Acessando o site do CursoemVideo
 `<iframe src="https://www.cursoemvideo.com" frameborder="0"></iframe>`
 para aprender a programar.
5. `</p>`
6. `</body>`

¹ [O que é iFrame](#) | [Teste suas habilidades!](#) | [Play safely in sandboxed IFrames](#)

iframe002

Em [iframe002.html](#), fizemos um teste simples com `<iframe>`, linkando-o à uma página local:

Testando com iframe

Lorem ipsum dolor sit amet consectetur adipisicing elit.

Isso é um teste!

Lorem ipsum dolor sit amet consectetur adipisicing elit. Id consequatur ab natus.

Criando a página local (pag001.html)

```
1 <head>
2     <title>Site de Teste</title>
3     <style>
4         body {
5             font-family: Arial, Helvetica, sans-
6                 serif;
7             background-color: rgb(240, 255, 255);
8             color: rgb(95, 158, 160);
9         }
10    </style>
11 </head>
12 <body>
13     <h1>Isso é um teste!</h1>
14 </body>
```

Teste com `<iframe>`

```
1 <body>
2     <h1>Teste com iframe</h1>
3     <p>Lorem...</p>
4     <iframe src="pag001.html" frameborder="0">
5         <p>Infelizmente, o seu navegador não
6             é compatível com isso!</p></iframe>
7     <p>Lorem...</p>
8 </body>
```

Em caso de o navegador não ser compatível ao `<iframe>`, o elemento `<p>` exibirá uma mensagem de erro.

iframe003

Em [iframe003.html](#), aprimoramos a maneira de exibir os conteúdos, utilizando o elemento `<a>`. O usuário ao clicar em um dos links estará dizendo para o navegador exibir o endereço correspondente em uma interface semelhante a essa:

- [Primeira página](#)
- [Segunda página](#)
- [Terceira página](#)

Escolha uma das opções acima

Lorem ipsum dolor sit, amet consectetur adipisicing elit. Quod incidunt officiis a vel laborum nobis esse et aliquam reprehenderit rerum consequuntur ex.



Criando as páginas extras

Diferentemente do exemplo anterior, iremos criar uma nova pasta, chamada paginas-extras, que irá manter os arquivos locais.

Assim, em [paginas-extras/pag001.html](#), modelo para os outros dois arquivos deste exemplo, temos:

```
1 <head>
2   <title>Site de teste</title>
3   <style>
4       body {
5           font-family: Arial, Helvetica, sans-
6               serif;
7           background-color: rgb(240, 255, 255);
8           color: rgb(95, 158, 160);
9       }
10  </style>
11 </head>
12 <body>
13   <h1>Primeira página de exemplo</h1>
14   <p>Lorem...</p>
15   <p>Lorem...</p>
16 </body>
```

De volta ao exercício, o primeiro passo é construir a estrutura inicial do site:

```
14 <body>
15     <h1>Faça uma escolha</h1>
16     <p>Lorem...</p>
17     <ul><li></li><li></li><li></li></ul>
23     <p>Lorem...</p>
24     <iframe><p>Infelizmente, o seu navegador não é com
    patível!</p></iframe>
28     <p>Lorem...</p>
29 </body>
```

Então, devemos aplicar um estilo:

```
1  <head>
2      <style>
3          body {
4              font-family: Arial, Helvetica, sans-se
                rif;
5          }
6
7          iframe#tela {
8              width: 95vw;
9              height: 400px;
10             border: 2px solid black;
11         }
12     </style>
13 </head>
```

O passo seguinte é criar uma lista que contenha as <a>s, apontando-as para às páginas locais.

```
17 <ul>
18     <li><a href="paginas-extras/pag001.html"
    target="frame">Primeira página</a></li>
19     <li><a href="paginas-extras/pag002.html"
    target="frame">Segunda página</a></li>
20     <li><a href="paginas-extras/pag003.html"
    target="frame">Terceira página</a></li>
21 <!-- O atributo target especifica onde abrir o
    documento vinculado -->
    <!-- Neste caso ele refere-se ao name="frame" -->
22 </ul>
```


Por fim, o `<iframe>`:

```
24 <iframe id="tela" name="frame" srcdoc="<h1>Escolha
    uma das opções acima</h1><p>Lorem...</p><img
    src='cursoemvideo-logo.png'>;"
25     <p>Infelizmente, o seu navegador não é
        compatível!</p>
26 <!-- O atributo srcdoc especifica o conteúdo HTML da
    página para ser exibida no <iframe> -->
27 </iframe>
```

Este `<iframe>` possui um conteúdo padrão, especificado através do atributo `srcdoc`; um `id` e um `name` como forma de endereço para ser buscado pelo `target`.

iframe004

Em [iframe004.html](#), tornamos os `<iframe>`s mais seguros através da utilização de atributos importantes, como o `sandbox` e `referrerpolicy`. Então, foi apresentado o conceito de formulário web, que coleta dados e direciona-os a um servidor:

Meu site principal

Lorem ipsum dolor sit amet consectetur, adipisicing elit. Neque blanditiis distinctio amet quia suscipit, earum sunt eveniet obcaecati pariat! Fugiat optio hic possimus alias distinctio odit veritatis consectetur ex dolore.

Cadastre-se aqui

Nome:

Lorem ipsum dolor sit, amet consectetur adipisicing elit. Quia ducimus sit rem saepe nihil autem eligendi sint tempore facilis. Porro, odio aspernatur odit iusto reiciendis deleniti officia hic consequuntur id?

O atributo **sandbox** permite um conjunto extra de restrições para o conteúdo em um **<iframe>**.

Valor	Descrição
sandbox	Aplica todas as restrições.
allow-forms	Permite submissão de formulários.
allow-same-origin	Permite que o conteúdo do <iframe> seja tratado como sendo da mesma origem.
allow-scripts	Permite executar scripts.

O atributo **referrerpolicy** especifica quais informações do referenciador devem ser usadas ao buscar o recurso.

Valor	Descrição
no-referrer	Previne que as informações do referenciador sejam passadas para o site de destino, removendo-as do cabeçalho HTTP.
no-referrer-when-downgrade	Predefinição. O cabeçalho do referenciador não será enviado para origens sem HTTPS.
origin	Envia apenas esquema, host e port para o cliente solicitante.
origin-when-cross-origin	Apenas a origem é enviada no cabeçalho do referenciador das solicitações de origem cruzada.
unsafe-url	Este valor é considerado inseguro.

Criando a página de cadastro

Em [paginas-extras/pag004.html](#), o formulário deve ter um nome de entrada para ser enviado a uma página PHP. Então, o dado obtido deve ser submetido como forma de cadastro.

```
1 <body>
2     <h1>Cadastre-se aqui</h1>
3     <form action="cadastro.php" method="get">
4 <!-- action especifica para onde enviar os dados -->
  <!-- GET é usado para solicitar dados de um recurso
    especificado -->
5         <p>
6             Nome:
7             <input type="text" name="nome" id="nome">
8             <input type="submit" value="cadastrar">
9         </p>
10    </form>
11 </body>
```

Este formulário ajuda a exemplificar o tratamento de dados entre servidores, se tentarmos cadastrar um nome o navegador exibira uma mensagem de erro. Porém, os novos conceitos vistos aqui serão discutidos mais adiante.

Agora com tudo pronto, vamos iniciar o exercício!

```
1 <body>
2     <h1>Meu site principal</h1>
3     <p>Lorem...</p>
4     <iframe src="paginas-extras/pag004.html"
5         frameborder="1" sandbox="allow-same-origin
6         allow-forms allow-scripts" referrerpolicy=
7         "no-referrer">
8     <p>Infelizmente, o seu navegador não
9         é compatível com isso!</p>
10    </iframe>
11 </body>
```

Neste exemplo, permitimos que o `<iframe>` carregue apenas conteúdos de mesma origem, formulários e scripts; porém, não iremos passar informações do usuário para o cabeçalho HTTP.

iframe005

Em [iframe005.html](#), testamos o uso de scripts nos `<iframe>`s:

Meu site principal

Lorem ipsum dolor sit, amet consectetur adipisicing elit. Aliquam quam eveniet temporibus nulla sequi? Nemo dignissimos possimus quod facilis corrupti in similique ab expedita quisquam, fuga temporibus eius corporis necessitatibus!

[Clique em mim](#)

Você clicou!

Você clicou!

Você clicou!

Lorem ipsum dolor sit, amet consectetur adipisicing elit. Quia ducimus sit rem saepe nihil autem eligendi sint tempore facilis. Porro, odio aspernatur odit iusto reiciendis deleniti officia hic consequuntur id?

Criando os scripts

Em [paginas-extras/pag005.html](#), desenvolvemos a aplicação de JavaScript na prática! Vamos discutir passo a passo o seu funcionamento:

```
1 <body>
2 <!-- 1. ao clicar no elemento <a> um resultado será
   exibido na página -->
3     <a href="#" onclick="clique()">Clique em mim</a>
4     <div id="saida"></div>
5
6 <!-- 2. o resultado vai sair daqui -->
7     <script>
8         function clique() {
9             s = window.document.getElementById("saida");
10            s.innerHTML += "<p>Você clicou!</p>";
11        } // 3. a função deste código é passar valores
12            // para a <div>, e repetir a cada click
13    </script>
14 </body>
```

Não há necessidade de saber tudo neste momento, o foco deve ser HTML e CSS. Caso queira saber mais, [aproveite o curso de JavaScript do curso em vídeo](#) para iniciar os estudos nesta parte do front-end.

Com tudo funcionando, iremos partir para o exercício e testar as novas funcionalidades obtidas:

```
1 </head>
2   <style>
3     iframe {
4       background-color: hotpink;
5     }
6   </style>
7 </head>
8 <body>
9   <h1>Minha página</h1>
10  <p>Lorem...</p>
11  <iframe src="paginas-extras/pag005.html"
12    referrerpolicy="no-referrer" sandbox="allow-
13    scripts">
14  <p>Infelizmente, o seu navegador não
    é compatível com isso!</p></iframe>
15  <p>Lorem...</p>
16 </body>
```

Dessa forma, permitimos que o `<iframe>` carregue as mensagens exibidas pela página local, resultando em uma interação divertida entre o usuário e o site.

iframe006

Em [iframe006.html](#), implementamos diferentes conteúdos à página HTML.

Para incorporar elementos externos ao `<iframe>`, é necessário que o site disponibilize uma opção de compartilhamento, como esta:

Compartilhar este vídeo

Link

<https://vimeo.com/revelco/inert>

Redes Sociais

Enviar email

Incorporação

+ Mostrar opções

```
<iframe src="https://player.vimeo.com/video/255804519?h=e145a36566"
width="640" height="338" frameborder="0" allow="autoplay; fullscreen;
picture-in-picture" allowfullscreen></iframe>
<p><a href="https://vimeo.com/255804519">Inertia ft. Brandon Semenuk</a>
from <a href="https://vimeo.com/revelco">Revel Co.</a> on <a
href="https://vimeo.com">Vimeo</a>.</p>
```

Este vídeo será incorporado com 640 pixels de largura.

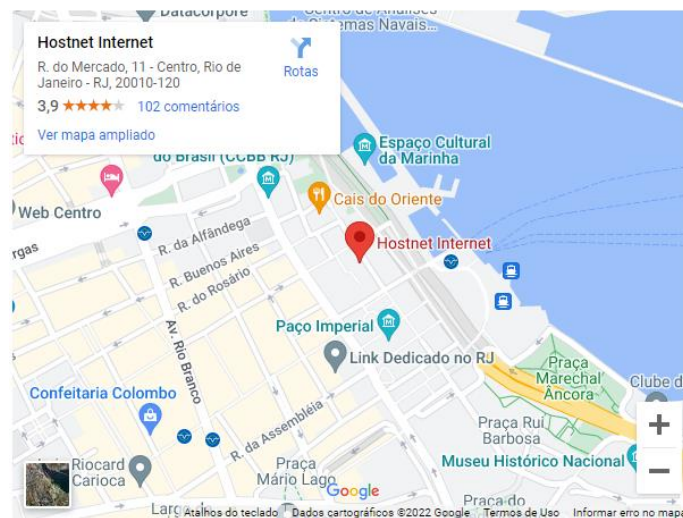
Este vídeo incorporado incluirá um link de texto.

Videos



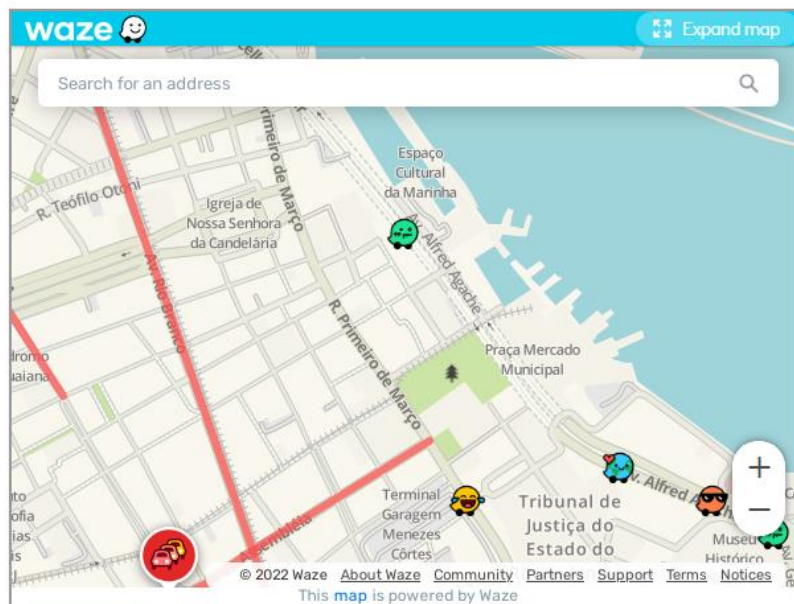
- 1 <h2>Vídeos</h2>
- 2 <iframe src="https://player.vimeo.com/video/255804519?h=e145a36566" width="640" height="338" frameborder="0" allow="autoplay; fullscreen; picture-in-picture" allowfullscreen></iframe>

Mapas do Google



- 3 <h2>Mapas do Google</h2>
- 4 <iframe src="https://www.google.com/maps/embed?pb=!1m18!1m12!1m3!1d3675.3108013811343!2d-43.177107535034395!3d-22.901904435014195!2m3!1f0!2f0!3f0!3m2!1i1024!2i768!4f13.1!3m3!1m2!1s0x997f58a6a00a9d%3A0x3f251d85272f76f7!2sHostnet%20Internet!5e0!3m2!1spt-BR!2sbr!4v1655508716354!5m2!1spt-BR!2sbr" width="600" height="450" style="border:0;" allowfullscreen="" loading="lazy" referrerpolicy="no-referrer-when-downgrade"></iframe>

Mapas do Waze



5 <h2>Mapas do Waze</h2>

6 <iframe src="https://embed.waze.com/iframe?zoom=16&lat=-22.902249&lon=-43.174909&ct=livemap" width="600" height="450" allowfullscreen></iframe>

Google Documents

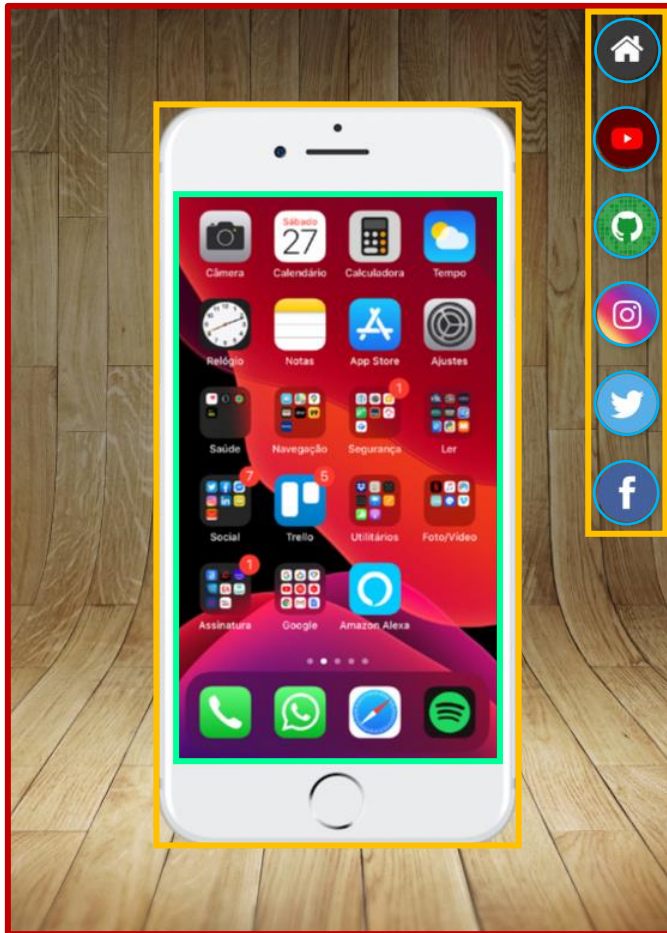


7 <h2>Google Documents</h2>

8 <iframe src="https://docs.google.com/presentation/d/e/2PACX-1vSW6WgirW1BUTC87yhFB1k7Ls79Iw6jsUYZBMzXnH6gGtxcluizFGSsXe88z8syjxouDHf18hmzLMDK/embed?start=false&loop=false&delayms=3000" frameborder="0" width="640" height="389" allowfullscreen="true" mozallowfullscreen="true" webkitallowfullscreen="true"></iframe>

Projeto Social

Neste projeto, construímos uma interface interativa, utilizando um sistema de links que exibe uma imagem na tela do smartphone de acordo com o elemento selecionado.



```
<html>
  <body>
    <main>
```

```
  <section id="
telefone">
```

```
    <iframe id="
tela">
```

```
  </section>
```

```
  <section id=redes-
sociais>
```

```
    <a target="tela">
      <img>
    </a>
```

```
  </section>
```

```
    </main>
  </body>
</html>
```

O pacote de imagens pode ser encontrado no perfil GitHub do [gustavogua-nabara](#).

Definindo a estrutura HTML

```
1 <body>
2     <main>
3         <section id="telefone">
4             <iframe src="home.html" name="tela"
5                 id="tela" frameborder="0">
6                 Seu navegador não é compatível com
7                 isso.
8             </iframe>
9         </section>
10        <section id="redes-sociais">
11            <a href="home.html" target="tela">
12                
14            </a><br>
15            <a href="youtube.html" target="tela">
16                
18            </a><br>
19            <a href="github.html" target="tela">
20                
22            </a><br>
23            <a href="instagram.html" target="tela">
24                
26            </a><br>
27            <a href="twitter.html" target="tela">
28                
30            </a><br>
31            <a href="facebook.html" target="tela">
32                
34            </a><br>
35        </section>
36    </main>
37</body>
```

Home

```
1 <style>
2     * {
3         margin: 0px;
4         padding: 0px;
5     }
6
7     body {
8         width: 100vw;
9         height: 100vh;
10        background: url("imagens/tela-home.jpg")
11            no-repeat;
12        background-size: cover;
13    }
14 </style>
```

Youtube

```
1 <body>
2     
4     <a href="https://www.youtube.com/user/
5         cursoemvideo/playlists" target="_blank">ACESSE</
6     a>
7 </body>
```

GitHub

```
1 <body>
2     
3     <a href="https://github.com/gustavoguanabara/"
4         target="_blank">ACESSE</a>
5 </body>
```

Instagram

```
1 <body>
2     
3     <a href="https://www.instagram.com/gustavoguanabara" target="_blank">ACESSE</a>
4 </body>
```

Twitter

```
1 <body>
2     
3     <a href="https://twitter.com/guanabara"
      target="_blank">ACESSE</a>
4 </body>
```

Facebook

```
1 <body>
2     
3     <a href="https://m.facebook.com/profile.php?id=1160402746" target="_blank">ACESSE</a>
4 </body>
```

Adicionando as cores

No começo da estilização, focamos nas primeiras camadas do corpo HTML. E depois, caímos para os elementos subjacentes, como uma cascata:

```
1  /* preparando a folha */
2  * {
3      margin: 0px;
4      padding: 0px;
5      font-family: Arial, Helvetica, sans-serif;
6  }
7
8  html, body {
9      height: 100vh;
10     width: 100vw;
11 }
12
13 body {
14     background: url("../imagens/fundo-madeira.jpg")
15         no-repeat top center;
16     background-size: cover;
17     background-attachment: fixed;
18 }
19 main {
20     position: relative;
21     height: 100vh;
22 }
```

Zeramos o **margin** e o **padding** para agrupar os elementos e definimos uma fonte.

Precisamos de toda a tela, então especificamos 100vh e 100vw para o **<html>** e **<body>**, ou seja, o ancestral expande-se, e o corpo ganha as mesmas proporções.

O **<body>** possui uma imagem centralizada de fundo que cobre todo seu tamanho e um **attachment** fixo para o conteúdo se ajustar junto a tela.

O **<main>** ocupa a altura total e possui uma posição relativa para posicionar, livremente os elementos que ele carrega.

Com a folha pronta, é hora de passar a ideia para o papel.

O telefone deve ser a parte principal, então centralizamos sua posição:

```
23 /* posicionando o telefone */
24 section#telefone {
25     position: absolute;
26     top: 50%;
27     left: 50%;
28     transform: translate(-50%, -50%);
29     height: 627px;
30     width: 311px;
31     background: url("../imagens/frame-iphone.png")
        no-repeat;
32 }
```

O `<iframe>` está posicionado por `top` e `left`, tente alterar seus valores, usando o developer tool.

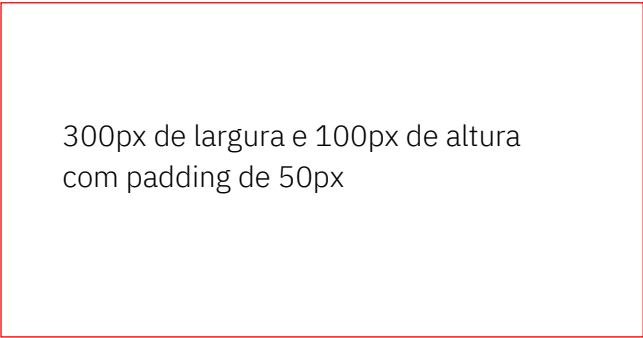
```
33 /* posicionando o iframe */
34 iframe#tela {
35     position: relative;
36     top: 80px;
37     left: 22px;
38     width: 267px;
39     height: 471px;
40 }
```

Na `section#redes-sociais`, os elementos ficam posicionados a direita como botões. Este formato circular deriva de `border-radius: 50%`, e seu conteúdo está contido pelo `box-sizing`, assim toda alteração nas dimensões ficará delimitada pelo tamanho do elemento. Por exemplo, sem `box-sizing`:

300px de largura e 100px de altura



300px de largura e 100px de altura
com padding de 50px



Com **box-sizing** deixamos as duas **<div>**s com o mesmo tamanho:

300px de largura e 100px de altura

300px de largura e 100px de altura com padding de 50px

```
41 /* posicionando as redes sociais */
42 section#redes-sociais {
43     text-align: right;
44 }
45
46 section#redes-sociais > img {
47     width: 50px;
48     margin: 10px;
49     border-radius: 50%;
50     box-shadow: 2px 2px 5px rgba(0, 0, 0, 0.400);
51     box-sizing: border-box;
52 }
```

Para finalizar adicionamos transição às imagens para que se mexam quando o mouse passar por cima do elemento:

```
53 /* transição */
54 section#redes-sociais > img:hover {
55     border: 2px solid white;
56     transform: translate(-3px, -3px);
57     transition: transform .3s, border .4s;
58     box-shadow: 5px 5px 10px rgba(0, 0, 0, 0.400);
59 }
```

O **:hover** é utilizado para iniciar o efeito de transição nas imagens.

O **transform** define um **translate()** que cresce o elemento para esquerda/cima.

O **transition** define o tempo para que os valores dos atributos **transform** e **border** mudem suavemente.

Por fim, utilizamos sombras para deixar a transição ainda mais sutil.

Estilizando as redes sociais

Afim de deixar a imagem contida no telefone bem posicionada, fizemos as mesmas configurações universais e implementamos `::-webkit-scrollbar` para retirar a barra de rolagem:

```
1  * {
2      margin: 0px;
3      padding: 0px;
4      font-family: Arial, Helvetica, sans-serif;
5  }
6
7  ::-webkit-scrollbar {
8      width: 0px;
9      height: 0px;
10 }
```

Depois, a imagem deverá ter 100% de largura:

```
11 img {
12     width: 100%;
13 }
```

Para finalizar, o botão “ACESSE” precisa ter um destaque, então:

```
14 a {
15     display: block;
16     background-color: darkred;
17     color: white;
18     padding: 20px;
19     text-align: center;
20     font-weight: bolder;
21     text-decoration: none;
22 }
23
24 a:hover {
25     background-color: red;
26 }
```

form

Os formulários são utilizados para coletar os dados de um visitante que solicita um cadastro em alguma parte do site. A entrada (ou input) destes dados é frequentemente enviada a um servidor para processamento.

form001

Em [form001.html](#), preparamos nosso primeiro formulário:

Meu primeiro formulário

Nome:

Sobrenome:

```
1 <body>
2   <h1>Meu primeiro Formulário</h1>
3   <form action="cadastro.php" autocomplete="off">
4     <p>Nome: <input type="text" name="nome"
5       id="nome"></p>
6     <p>Sobrenome: <input type="text" name="
7       sobrenome" id="sobrenome"></p>
8     <p><input type="submit" value="Enviar"></p>
9   </form>
10 </body>
```

O atributo **action** especifica o caminho para enviar os dados do formulário quando ele é submetido.

Você também pode desabilitar as sugestões de preenchimento automático do navegador definindo **autocomplete="off"**.

O **<input>** é basicamente um campo para preenchimento de dados, contendo várias finalidades que vamos discutir durante este capítulo.

Etiquetando os <input>s

O formulário acima ainda é ineficiente, isto pois não possui um elemento essencial para questões de usabilidade. Portanto, vamos dar prosseguimento aos estudos:

```
1 <body>
2     <h1>Meu primeiro Formulário</h1>
3     <form action="cadastro.php" autocomplete="off">
4         <p><label for="nome">Nome:</label>
5             <input type="text" name="nome"
6                 id="nome"></p>
7         <p><label for="sobrenome">Sobrenome:</label>
8             <input type="text" name="sobrenome" id="
9                 sobrenome"></p>
10    <!-- for sempre terá um id como valor -->
11    <p><input type="submit" value="Enviar"></p>
12 </form>
13 </body>
```

A tag `<label>` é necessária para dar nome aos dados de entrada, isso significa que o navegador não consegue identificar a ligação entre nome/sobrenome e os dados coletados.

Assim, este elemento permite que os leitores de tela leiam corretamente o conteúdo da página para os usuários.

Além do mais, o elemento é útil para podermos selecionar pequenas regiões de uma tela pequena, porque ao clicar na `<label>` o campo será automaticamente selecionado.

Meu primeiro formulário

Nome:

Sobrenome:

Visualmente, não muda nada, porém agora o código está escrito corretamente.

Métodos de segurança

O formulário que criamos passa os dados coletado para a URL da página, isto ocorre porque a requisição que enviamos para o servidor é processada e estes dados são enviados novamente para o lado do cliente, tornando a resposta visível. Então, por exemplo:

Meu primeiro formulário

Nome:

Sobrenome:

`../cadastro.php?nome=José&sobrenome=Pereira`

Esta é uma característica comum na internet, pois muitas vezes compartilhamos resultados de busca, por exemplo, e a URL precisa incluir todos os dados para que a página possa ser encontrada.

GET

Utilize o GET quando os dados:

- podem ser armazenados em cache;
- podem permanecer no histórico do navegador;
- podem ser visíveis para todo mundo na URL;
- não são dados confidenciais (sensíveis).

OBS.: As requisições feitas com GET têm restrição de 3.000mb, ou 3.000 letras.

POST

Utilize o POST quando os dados:

- não devem ser armazenados em cache;
- não devem permanecer no histórico do navegador;
- não devem ser exibidas na URL.

As requisições feitas com POST enviam dados a um servidor, que cria/atualiza um recurso:

```
1 <body>
2   <h1>Meu primeiro Formulário</h1>
3   <form action="cadastro.php" method="post"
4     autocomplete="off">
5     <p><label for="nome">Nome:</label>
6       <input type="text" name="nome"
7         id="nome"></p>
8     <p><label for="sobrenome">Sobrenome:</label>
9       <input type="text" name="sobrenome" id="
10        sobrenome"></p>
11     <p><input type="submit" value="Enviar"></p>
12   </form>
13 </body>
```

Os dados enviados ao servidor são armazenados no corpo da requisição HTTP, e podem ser acessados pelo developer tools:

Network				
Name	Headers	Payload	Preview	Response Initiator >>
cadastro.php	Form Data	view source	view URL-encoded	
	nome: José			
	sobrenome: Pereira			

form002

Em [form002.html](#), testamos uma nova funcionalidade do `<form>` que nos permite exigir o preenchimento mínimo e máximo dos campos requeridos:

Teste de Formulário

Nome:

Sobrenome:

```
1 <body>
2   <h1>Teste de Formulário</h1>
3   <form action="cadastro.php" method="post"
4     autocomplete="on">
5     <p><label for="iusu">Usuário: </label>
6       <input type="text" name="usu" id="
7         iusu" minlength="3" maxlength="15"
8         size="10" placeholder="nome do usuário"
9         autocomplete="username" required></p>
10    <p><label for="isob">Senha: </label>
11      <input type="password" name="sen" id="
        isen" minlength="8" maxlength="20"
        size="8" placeholder="mínimo 8 letras"
        autocomplete="current-password"
        required></p>
12    <p><input type="submit" value="Enviar">
13      <input type="reset" value="Limpar"></p>
14  </form>
15</body>
```

Além disso, estamos especificando um tamanho para o campo de preenchimento, uma pequena descrição do dado de entrada e um preenchimento automático. Respectivamente, com os atributos `size`, `placeholder` e `autocomplete`.

form003

Em [form003.html](#), testamos novos tipos de `<input>`s para construir um formulário estudantil:

Teste de Formulário

Nome:

Média:

Período letivo:

Dia da prova:

Horário da prova:

```
1 <body>
2   <h1>Teste de Formulário</h1>
3   <form action="cadastro.php" method="get"
4     autocomplete="on">
5     <p><label for="iname">Nome:</label>
6       <input type="text" name="nome"
7         id="nome" maxlength="30" placeholder="
8         Nome completo" autocomplete="name"
9         required></p>
10    <p><label for="imedia">Média:</label>
11      <input type="number" name="media" id="
12        imedia" min="0" max="10" placeholder="0 a
13        10" step="0.5" value="5" required></p>
14    <p><label for="imes">Período letivo:</label>
15      <input type="month" name="mes" id="imes"
16        value="2022-09"></p>
17    <p><label for="idia">Dia da prova:</label>
18      <input type="date" name="dia" id="idia"
19        value="2022-9-30"></p>
20    <p><label for="ihora">Horário da prova:</
21      label>
22      <input type="time" name="hora" id="ihora"
23        ></p>
24    <p><input type="submit" value="Enviar">
25    <input type="reset" value="Limpar"></p>
26  </form>
27 </body>
```

Este código é bastante dinâmico, tente alterar os valores e testar novos resultados em outros navegadores ou dispositivos móveis.

form004

Em [form004.html](#), trabalhamos com dados pessoais em um formulário com `<fieldset>`, que especifica um grupo de conteúdos relacionados:

Exemplo de Formulário

Dados pessoais

Nome:

E-mail:

Telefone:

```
1 <body>
2   <h1>Exemplo de Formulário</h1>
3   <form action="cadastro.php" method="post"
4     autocomplete="on">
5     <fieldset>
6       <legend>Dados pessoais</legend>
7       <p><label for="iname">Nome: </label>
8         <input type="text" name="nome" id="
9           inome" minlength="3" maxlength="20"
10            required></p>
11       <p><label for="iemail">E-mail: </label>
12         <input type="email" name="email" id="
13           iemail" autocomplete="email" required
14            ></p>
15       <p><label for="itel">Telefone: </label>
16         <input type="tel" name="tel" id="itel
17           " autocomplete="tel" pattern="^\(\d{2
18           }\)\d{4,5}-\d{4}$" placeholder="(xx)x
19           xxx-xxxx" required></p>
20     </fieldset>
21     <p><input type="submit" value="Enviar">
22       <input type="reset" value="Limpar"></p>
23   </form>
24 </body>
```

O atributo **pattern** especifica uma expressão regular (RegEx), padrão de caracteres que devem ser respeitados no campo de preenchimento. No nosso exemplo do telefone, o número deve ter: os dois dígitos do DDD e de 8 a 9 dígitos após o DDD.

O **pattern** pode associar sequências de caracteres em **<input>**s do tipo: **text**, **data**, **search**, **url**, **tel**, **email** e **password**.

form005

Em [form005.html](#), testamos os **<input>**s **radio** e **checkbox**:

Exemplo de Formulário

Sexo

☒ Masculino
☐ Feminino

Esportes favoritos

☐ Basquete
☒ Futebol
☐ Natação
☐ Tênis

Enviar

Limpar

```

1  <body>
2      <h1>Exemplo de Formulário</h1>
3      <form action="cadastro.php" method="get"
4          autocomplete="on">
5          <fieldset>
6              <legend>Sexo</legend>
7              <label for="isxmas">
8                  <input type="radio" name="sexo" id="
9                      isxmas" value="Masculino" checked>
10                     Masculino</label><br>
11              <label for="isxfem">
12                  <input type="radio" name="sexo" id="
13                      isxfem" value="Feminino" checked>
14                     Feminino</label>
15          </fieldset>
16          <fieldset>
17              <legend>Esportes favoritos</legend>
18              <label for="iesbas">
19                  <input type="checkbox" name="esbas"
20                      id="iesbas">Basquete</label><br>
21              <label for="iesfut">
22                  <input type="checkbox" name="fut"
23                      id="iesfut" checked>Futebol</
24                      label><br>
25              <label for="iesnat">
26                  <input type="checkbox" name="esnat"
27                      id="iesnat">Natação</label><br>
28              <label for="iesten">
29                  <input type="checkbox" name="esten"
30                      id="iesten">Tênis</label>
31          </fieldset>
32      </form>
33 </body>

```

Um grupo `<input>` de tipo `radio` deve compartilhar o mesmo nome para funcionar corretamente, assim apenas um botão pode ser selecionado ao mesmo tempo.

Contrariamente, um `<input>` de tipo `checkbox` pode ser usado para selecionar múltiplos campos.

form006

Em [form006.html](#), testamos os `<input>`s `color`, `range` e `file`:

Exemplo de Formulário

Qual é sua cor favorita?:

Nível de Satisfação:

Foto de Perfil: No file chosen

```
1 <body>
2     <h1>Exemplo de Formulário</h1>
3     <form action="cadastro.php" method="get"
4         autocomplete="on">
5         <p><label for="icor">Qual é sua cor favorita?
6             : </label>
7             <input type="color" name="cor" id="icor"
8                 value="#00ff00"></p>
9         <p><label for="inivel">Nível de Satisfação:
10             </label>
11             <input type="range" name="nivel" id="
12                 nivel" min="1" max="5" value="2">
13             <label for="ifoto">Foto de Perfil: </label>
14             <input type="file" name="foto"
15                 id="ifoto">
16             <p><input type="submit" value="Enviar">
17                 <input type="reset" value="Limpar"></p>
18         </form>
19 </body>
```

`color` define um seletor de cores, seu valor deve estar em hexadecimal de sete caracteres.

`range` define um controle para inserir um número cujo valor exato não é importante. Possui um intervalo padrão de 0 a 100, porém, podemos definir restrições do tipo `max`, `min`, `step` e `value` para os números que devem ser aceitos.

`file` define um campo de seleção de arquivos e um botão de “procurar” para uploads de arquivo.

form007

Em [form007.html](#), testamos uma nova finalidade do `<form>`, que utiliza as tags `<select>`, `<datalist>` e `<textarea>` como formas alternativas do tradicional `<input>` para coletar informações do usuário.

Exemplo de Formulário

Estado:

Profissão:

Mensagem:

O elemento `<select>` é usado para criar uma lista suspensa. Para cada item da lista devemos especificar a tag `<option>` para dar opções de escolha. Em nosso exemplo, temos:

```
1 <form action="cadastro.php" method="post"
  autocomplete="on">
2 <p><label for="iset">Estado: </label>
3   <select name="estado" id="iest">
4 <!-- Opção padrão (pré-seleção) -->
5       <option value="" selected>--- Escolha ---
        </option>
6       <optgroup label="Região Sudeste">
7           <option value="MG">Minas Gerais</option>
8           <option value="RG">Rio de Janeiro</
              option>
9           <option value="SP">São Paulo</option>
10          </optgroup>
11      </select></p>
```

Note que a tag `<optgroup>` é usada para agrupar opções relacionadas; e também possui um atributo `label` para rotulá-los.

`<select>` precisa do atributo `name` para fazer referência aos dados da lista e envia-los corretamente após submetidos, sem ele nenhum dado poderá ser enviado. O atributo `id` faz a associação da lista com a `<label>`.

Na parte seguinte, temos a `<datalist>`, que especifica uma lista de opções predefinidas para um elemento `<input>`:

```
12 <p><label for="iprof">Profissão: </label>
13   <input type="text" name="prof" id="iprof"
    list="lstprof">
14   <datalist id="lstprof">
15       <option value="ADM">Administrador</option>
16       <option value="CONT">Contabilista</option>
17       <option value="DEV">Desenvolvedor</option>
18       <option value="PROF">Professor</option>
19   </datalist></p>
```

Observe a ordem de relação que utilizamos aqui, a `<label>` associa-se ao `<input>` que se associa a `<datalist>`. Dessa forma, quando o atributo `list` está presente, conseguimos fornecer um recurso “autocomplete” para elementos `<input>`, mostrando a lista de opções predefinidas.

Por fim e não menos importante, `<textarea>` é utilizado para coletar informações, como comentários ou revisões:

```
20 <p><label for="img">Mensagem: </label><br>
21   <textarea name="msg" id="img" cols="30"
    rows="10"></textarea></p>
```

O tamanho de `<textarea>` é especificado pelos atributos `cols` e `rows` (ou com CSS). Tente alterá-los para ver seu funcionamento.

Porém podemos redimensioná-lo livremente pelo canto inferior direito.

```
22 <p><input type="submit" value="Enviar">
23 <input type="reset" value="Limpar">
24 </form>
```

form008

Em [form008.html](#), testamos o valor de saída do formulário, aquele que é exibido na tela.

A ideia desse exemplo é mostrar o resultado da soma dos valores obtidos dos `<input>`s `in1` e `in2` através da tag `<output>`:

Exemplo de Formulário

Número 1:

Número 2:

Soma: 0

```
1 <body>
2   <h1>Exemplo de Formulário</h1>
3   <form action="cadastro.php" method="get"
4   autocomplete="on">
5       <p><label for="in1">Número 1: </label>
6           <input type="number" name="n1" id="in1"
7               min="0" max="50" required>
8       <p><label for="in2">Número 2: </label>
9           <input type="number" name="n2" id="in2"
10              min="0" max="50" required>
11       <p><label for="isoma">Soma: </label>
12           <output name="soma" id="isoma">0</output>
13       <p><input type="submit" value="Enviar">
14           <input type="reset" value="Limpar"></p>
15   </form>
16 </body>
```

Somando valores (oninput)

O atributo **oninput** é acionado quando obtemos a entrada do usuário, assim toda vez que o valor contido em **<input>** é alterado, conseguimos utilizá-lo para fazer a equação entre **in1** e **in2**. Por exemplo:

Exemplo de Formulário

Número 1:

Número 2:

Soma: 2

```
1 <body>
2   <h1>Exemplo de Formulário</h1>
3   <form action="cadastro.php" method="get"
4     autocomplete="on">
5     <p><label for="in1">Número 1: </label>
6       <input type="number" name="n1" id="in1"
7         min="0" max="50" oninput="isoma.innerHTML
8         = Number(in1.value) + Number(in2.value)"
9         required>
10    <p><label for="in2">Número 2: </label>
11      <input type="number" name="n2" id="in2"
12        min="0" max="50" oninput="isoma.innerHTML
13        = Number(in1.value) + Number(in2.value)"
14        required>
15    <p><label for="isoma">Soma: </label>
16      <output name="soma" id="isoma">2</output>
17    <p><input type="submit" value="Enviar">
18      <input type="reset" value="Limpar"></p>
19  </form>
20 </body>
```

Em **oninput**, basicamente estamos dizendo que **isoma** recebe a soma dos dois valores.

form009

Em [form009.html](#), utilizamos o `<input>` de tipo `range` para fazer um novo teste com o `<output>`:

Agora, precisamos atualizar o valor do lado do “range” toda vez que ele é alterado.

Portanto, precisamos definir um `id="ival"` para o `<output>` e através de `oninput` dizemos que `ival` recebe o valor de `inum`.

Exemplo de Formulário

Número:  5

Enviar

Limpar

```
1 <body>
2   <h1>Exemplo de Formulário</h1>
3   <form action="cadastro.php" method="get"
4   autocomplete="on">
5       <p><label for="inum">Número: </label>
6           <input type="range" name="num" id="inum"
              min="0" max="10" value="5" oninput="
              ival.innerHTML = Number(inum.value
              )">
7           <output id="ival">5</output></p>
8       <p><input type="submit" value="Enviar">
9       <input type="reset" value="Limpar"></p>
10   </form>
11 </body>
```

Tente mexer no controle para ver o valor sendo alterado.

form010

Em [form010.html](#), similarmente aos dois últimos exercícios, temos um valor de entrada e este valor deve ser exibido na tela por meio da tag `<output>`.

Exemplo de Formulário

Em que ano você nasceu?

Sua idade é: 22

```
1 <body>
2   <h1>Exemplo de Formulário</h1>
3   <form action="cadastro.php" method="get"
4     autocomplete="on">
5     <p><label for="iano">Em que ano você nasceu?
6       <input type="number" name="ano" id="iano"
7         min="1900" max="2022" oninput="
8         calcIdade()"></p>
9     <p><label for="iidade">Sua idade é: </label>
10      <output id="iidade">22</output></p>
11    <p><input type="submit" value="Enviar">
12      <input type="reset" value="Limpar"></p>
13  </form>
```

Pois bem. Devemos então descobrir a idade do usuário com base no valor (ano) que ele nos dá, fazendo a equação ano atual menos o valor de entrada:

```
12   <script>
13     function calcIdade() {
14       let atual = new Date().getFullYear();
15       iidade.innerHTML = Number(atual) -
16         Number(iano.value);
17   // sempre que o valor de entrada é alterado
18     calcIdade() aciona a equação e passa este resultado
19   } // para oninput, que atualiza <output>
20   </script>
21 </body>
```

Media Query²

A media query é um recurso da CSS3 que permite adaptar o conteúdo de um site à diferentes mídias, como uma impressora, televisão, celular, notebook, tablet, entre outros.

Podemos expressa-la como sendo o resultado da soma entre media types (tipos) e media features (características).

CSS media attribute

Esse atributo é usado para especificar estilos diferentes para diferentes tipos de mídia e suas características (features).

Operador lógico (value)	Descrição
and	Operador E
not	Operador NÃO
,	Operador OU

Valor (type)	Descrição
all	Padrão. Usado para todos os dispositivos de tipo de mídia.
print	Usado para o modo de visualização de impressão
screen	Usado para telas de computador, tablets, smartphones etc.
speech	Usado para leitores de tela que "lê" a página em voz alta.

Valor (feature)	Descrição
orientation	Especifica a orientação (portrait/landscape) do visor.
resolution	Especifica a densidade de pixels (DPI ou DPCM) do visor.
height/width	A altura/largura da janela de visualização.

² [CSS Media Queries - Examples](#) | [HTML <link> media Attribute](#)

mq001

Em <mq001/index.html>, criamos um exemplo para adaptar o conteúdo da página HTML ao modo de visualização da página impressa.

Em um site há muitas coisas que não ficam legais na versão impressa, como os links de navegação o tamanho/alinhamento das letras, fonte, etc.:



Portanto, definimos duas folhas de estilo para separar a configuração de cada mídia:

```
1 <head>
2   <link rel="stylesheet" href="estilos/tela.css"
3     media="screen">
4   <link rel="stylesheet" href="estilos/
5     impressora.css" media="print">
6 </head>
```

Estilos para telas

Este site contém um menu logo abaixo da **<header>** para carregar os links de navegação. Depois um **<article>** com o conteúdo centralizado para que o usuário não tenha dificuldade de lê-lo em telas maiores.

Confira em <mq001/estilos/tela.css>.

Estilos para impressora

Nesta versão, em mq001/estilos/impressora.css, os parágrafos estão ocupando uma largura maior (100%), a fonte mudou e há um comentário após o artigo para indicar o autor da impressão.

Notícias

Estou aprendendo a criar Media Queries

Lorem ipsum dolor sit amet, consectetur adipisicing elit. Culpa, qui consectetur maxime non natus modi, facilis voluptatibus ratione, quisquam officia sit doloribus quo ut corporis voluptas dolore quis recusandae ullam? Lorem ipsum dolor sit amet, consectetur adipisicing elit. Culpa ad tempora ipsam, in sunt illo, velit doloremque voluptatibus obcaecati nesciunt esse voluptates mollitia exercitationem, consectetur ab enim quis officia nostrum.

Lorem ipsum dolor sit amet, consectetur adipisicing elit. Culpa, qui consectetur maxime non natus modi, facilis voluptatibus ratione, quisquam officia sit doloribus quo ut corporis voluptas dolore quis recusandae ullam? Lorem ipsum dolor sit amet, consectetur adipisicing elit. Culpa ad tempora ipsam, in sunt illo, velit doloremque voluptatibus obcaecati nesciunt esse voluptates mollitia exercitationem, consectetur ab enim quis officia nostrum.

Lorem ipsum dolor sit amet, consectetur adipisicing elit. Culpa, qui consectetur maxime non natus modi, facilis voluptatibus ratione, quisquam officia sit doloribus quo ut corporis voluptas dolore quis recusandae ullam? Lorem ipsum dolor sit amet, consectetur adipisicing elit. Culpa ad tempora ipsam, in sunt illo, velit doloremque voluptatibus obcaecati nesciunt esse voluptates mollitia exercitationem, consectetur ab enim quis officia nostrum.

Esse artigo foi impresso através do site www.cursoemvideo.com

O que fizemos foi adaptar todo o conteúdo para oferecer uma leitura agradável no papel.

mq002

Em mq002/index.html, fizemos o exercício das media features para aplicar estilos diferentes de acordo com a orientação da tela.



```
1 <head>
2   <link rel="stylesheet" href="estilos/style.css"
   media="all">
3   <link rel="stylesheet" href="estilos/retrato.css"
   media="screen and (orientation: portrait)">
4   <link rel="stylesheet" href="estilos/
   paisagem.css" media="screen and
   (orientation: landscape)">
5 </head>
```

Com três folhas de estilo podemos dividir a CSS em estilos gerais, retrato e paisagem. Esta é uma forma eficiente quando há muitas configurações a fazer, pois deixa tudo menos confuso.

Note a presença do operador **and**, estamos adicionando uma característica ao tipo de mídia **screen**.

Veja o css completo em mq002/estilos.

mq003

Em <mq003/index.html>, reunimos todas as mídias do exercício anterior em um único estilo [<mq003/estilos/style.css>].

```
1  /* declarações gerais */
2  @media all {
3
4  }
5
6  /* declarações retrato */
7  @media screen and (orientation: portrait) {
8
9  }
10
11 /* declarações paisagem */
12 @media screen and (orientation: landscape) {
13
14 }
```

Podemos utilizar esta estilização centralizada de duas formas:

CSS Internal style

Dentro do elemento `<style>`, a aplicação CSS é feita junto ao HTML.

CSS External style

Dentro do elemento `<link>`, incluímos uma referência a folha de estilo externo.

Mobile First³

O conceito mobile first traz a criação de projetos feitos para dispositivos móveis em primeiro lugar, para depois fazer adaptações para o desktop. Foi desenvolvido entre 2009 e 2010 por Luke Wroblewski quando trabalhava como Diretor Sênior de ideação de Produto & Design no Yahoo.

Hoje ele é Diretor de Produto do Google e recomenda o Mobile First pela melhoria na experiência do usuário, e também pois, sites construídos desta forma serão mais valorizados pelo algoritmo do Google.

mq004

Em mq004/index.html, aprofundamos todo conteúdo visto até aqui para incluir os breakpoints.

O pacote de imagens está disponível no perfil GitHub do [gustavoguanabara](#).



Esta é uma forma de criar pontos na CSS para o conteúdo se ajustar ao tamanho da tela dos diferentes dispositivos. No entanto, não são pontos exatos, porque há uma grande variedade de tamanhos entre os dispositivos.

Então, temos os típicos breakpoints para simplificar um pouco:

- Pequenas telas: até 600px de largura
- Celular: de 600px até 768px de largura
- Tablet: 768px até 992px de largura
- Desktop: 992px até 1200px de largura
- Grandes telas: acima de 1200px de largura

³ [Entenda o que é mobile first](#)

Versão Mobile First

A CSS para este caso é um estilo geral, pois iniciamos com a versão para pequenas telas e celulares.

Em [mq004/estilos/style.css](#), temos um seletor universal para ajustar os elementos:

```
1 * {  
2     font-family: Arial, Helvetica, sans-serif;  
3     font-size: 1.3em;  
4     margin: 0px;  
5     padding: 0px;  
6     box-sizing: border-box;  
7 }
```

Especificamos uma fonte padrão, uma proporção maior de tamanho, depois agrupamos todo o conteúdo, zerando o **margin/padding** e por fim adicionamos **box-sizing**.

```
8 html {  
9     height: 100vh;  
10 }
```

É importante que a altura do elemento ancestral esteja ocupando toda a tela do dispositivo, assim exibimos o máximo da imagem.

```
11 body {  
12     background: black url("../imagens/back-phone  
13         .jpg") no-repeat;  
14     background-size: cover;  
15     background-position: center center;  
16 }
```

A imagem de fundo cobre o espaço inteiro do **<body>** e por conta do **<html>** ela ganha a proporção de toda a tela do dispositivo; sua posição é totalmente centralizada.

```

16 main {
17     background-color: rgba(255, 255, 255, 0.842);
18     width: 400px;
19     margin: auto;
20     margin-top: 20px;
21     border-radius: 10px;
22     padding: 10px;
23 }

```

No `<main>` é onde exibiremos a parte principal do HTML, as imagens que mudam conforme a tela.

Testando Media Queries



```

24 h1 {
25     text-align: center;
26     color: white;
27     font-size: 2em;
28     text-shadow: 2px 2px 2px rgba(0, 0, 0, 0.493);
29 }
30
31 img {
32     display: block;
33     margin: auto;
34 }

```

O `<h1>` e `` são blocos centralizados. O `<h1>` tem sombra e também possui um `font-size` para sobrepor-se ao seletor universal.

```

35 img#phone { display: block; }
36 img#tablet { display: none; }
37 img#print { display: none; }
38 img#pc { display: none; }
39 img#tv { display: none; }

```

No final do código definimos a imagem que deve aparecer para este tamanho de tela (block), e as que não devem (none). Faremos a mesma coisa para as demais mídias.

Demais Medias Queries

Nesta parte, agrupamos toda CSS em external style para especificar cada breakpoint: tablet, desktop e tv; mais uma mídia de impressão.

Em [mq004/estilos/media-query.css](#), começamos com **@media print**:



- Alteramos a fonte, o `<main>` ganhou uma borda e uma largura de 90vw.
- O conteúdo após o `<main>` possui um `text-decoration: underline;`.
- O `<h1>` tem cor preta e perdeu a sombra.

Depois, para as demais mídias, simplesmente, atualizamos a imagem que deve ser exibida em cada uma.

mq005

Em [mq005/index.html](#), criamos dois tipos de menu para se adaptar de acordo com o tamanho da tela.



Criando o menu hambúrguer

Este e outros ícones podem ser encontrados no [Google Fonts](https://fonts.google.com/), sua implementação é feita por uma referência ao elemento `<link>`:

```
1 <head>
2   <title>Media Query</title>
3   <link rel="stylesheet" href="https://fonts.
  googleapis.com/css2?family=Material+Symbols+
  Outlined:opsz,wght,FILL,GRAD@20..48,100..700,
  0..1,-50..200">
4   <link rel="stylesheet" href="estilos/style.css">
5   <link rel="stylesheet" href="estilos/media-
  query.css">
6 </head>
```

Ao `<body>` devemos definir `onresize` para o navegador identificar quando o tamanho da tela é reajustado e chamar uma ação.

```
7 <body onresize="mudouTamanho()">
```

No `<header>` implementamos o ícone do menu ao código para obtermos o efeito desejado ao clicar no elemento ``.

```
8   <header>
9     <h1>Meu site</h1>
10    <span id="burger" class="material-symbols-
  outlined" onclick="clickMenu()">menu</span>
11    <menu id="itens">
12      <ul>
13        <li><a href="#">Opção 1</a></li>
14        <li><a href="#">Opção 2</a></li>
15        <li><a href="#">Opção 3</a></li>
16        <li><a href="#">Opção 4</a></li>
17        <li><a href="#">Opção 5</a></li>
18      </ul>
19    </menu>
20  </header>
```

Dentro de `<script>` adicionamos a primeira função `mudouTamanho()` para rodar o código que exhibe ou esconde o menu em telas maiores ou iguais a 768 de largura.

```
21     <script>
22         function mudouTamanho() {
23             if (window.innerWidth >= 768) {
24                 itens.style.display = "block";
25             } else {
26                 itens.style.display = "none";
27             }
28         }
```

Se a largura da tela for maior o igual a 768 o menu é exibido (block), por outro lado é escondido (none) se a largura da tela for menor que 768.

Para telas menores que 768px de largura precisaremos alterar o menu para um tipo de lista suspensa que cria e desfaz a exibição dos itens de acordo com o click do usuário.

A segunda função `clickMenu()` deve exhibir o menu hambúrguer ou escondê-lo pelo click do usuário.

```
29         function clickMenu() {
30             if (itens.style.display == "block") {
31                 itens.style.display = "none";
32             } else {
33                 itens.style.display = "block";
34             }
35         }
```

Se o menu está sendo exibido (block), ao clicar deve ser escondido (none). Porém, se estiver escondido, ao clicar deve ser exibido.

```
36     </script>
37 </body>
```

Versão Mobile First

Em [mq005/estilos/style.css](#), por padrão, fizemos algumas declarações gerais para o site:

```
1  * {
2      font-family: Arial, Helvetica, sans-serif;
3      margin: 0px;
4      padding: 0px;
5  }
6
7  html, body {
8      background-color: lightgray;
9  }
10
11 header {
12     background-color: rgb(148, 148, 148);
13 }
14
15 header > h1 {
16     padding: 10px;
17     text-align: center;
18 }
```

Após esta parte, temos:

```
19 span#burger {
20     background-color: rgb(48, 48, 48);
21     color: white;
22     display: block;
23     text-align: center;
24     padding: 10px;
25     cursor: pointer;
26 }
```



Adicionamos um `:hover`:

```
27 span#burger:hover {
28     background-color: white;
29     color: black;
30 }
```



Para o `<menu>`, especificamos `none` para que a experiência comece com a lista escondida.

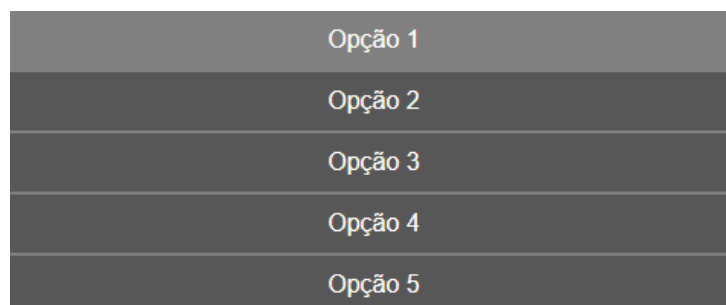
```
31 menu {  
32     display: none;  
33 }  
34  
35 menu > ul {  
36     list-style-type: none;  
37 }
```

Depois, definimos o estilo do `<a>`:

```
38 menu > ul > li > a {  
39     display: block;  
40     padding: 10px;  
41     text-decoration: none;  
42     text-align: center;  
43     background-color: rgb(87, 87, 87);  
44     color: white;  
45     border-top: 2px solid gray;  
46 }
```



```
47 menu > ul > li > a:hover {  
48     background-color: gray;  
49 }
```



Por fim, fizemos as configurações do conteúdo principal:

```
50 main {
51     width: 90vw;
52     background-color: white;
53     margin: auto;
54     margin-top: 10px;
55     padding: 10px;
56     border-radius: 10px;
57 }
58 article > h2 {
59     padding-bottom: 20px;
60 }
61
62 article > p {
63     text-align: justify;
64     margin-bottom: 20px;
65     text-indent: 20px;
66 }
```

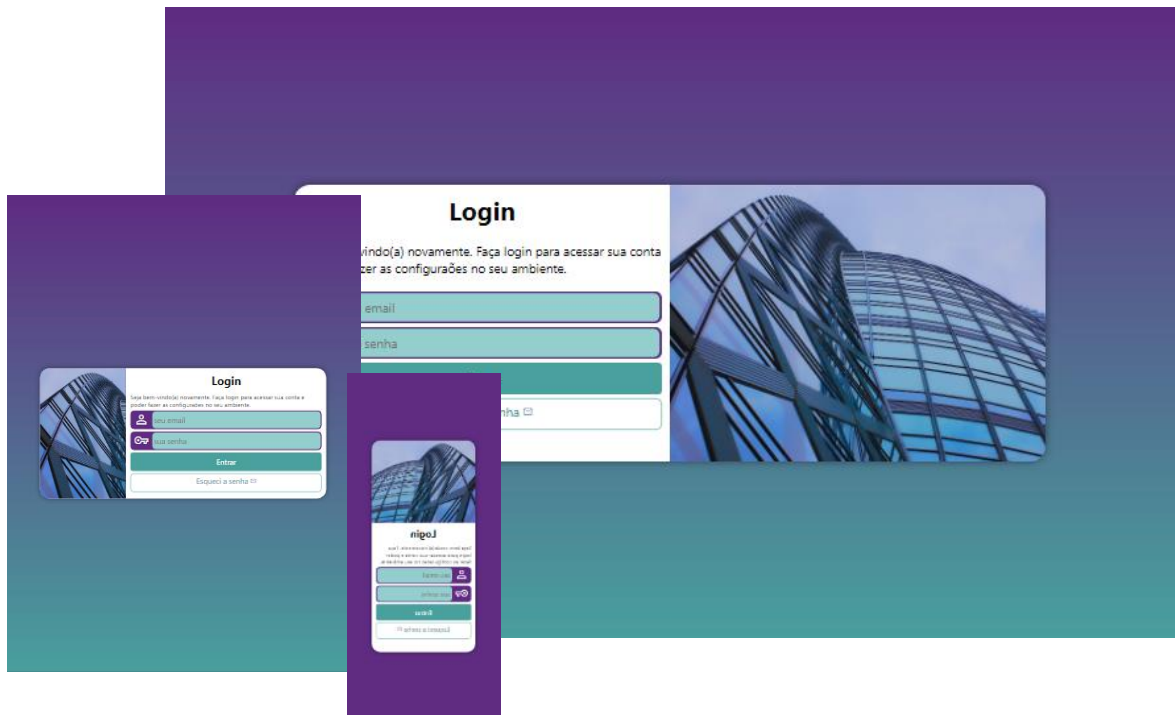
Estilo para telas

Em [mq005/estilos/media-query.css](#), para telas maiores ou iguais a 768px escodemos o menu hambúrguer e exibimos seu equivalente para telas maiores:

```
1 @media screen and (min-width: 768px) {
2     span#burger {
3         display: none;
4     }
5
6     menu {
7         display: block;
8     }
9
10    menu > ul > li {
11        display: inline-block;
12    }
13
14    main {
15        width: 768px;
16    }
17 }
```

Projeto Login

Neste novo projeto, construímos um modelo de interface para usuários acessarem nosso site através de um login. Este exercício abrange grande parte dos conceitos vistos até aqui, como formulário, media query e mobile first, sendo um ótimo desafio para testar suas habilidades.



Você pode encontrar esta e outras imagens no [pexels](https://www.pexels.com).

Definindo a estrutura HTML

Para começar crie o HTML básico e depois faça alguns ajustes:

Login

Seja bem-vindo(a) novamente. Faça login para acessar sua conta e poder fazer as configurações no seu ambiente.

person Login

vpn_key Senha

[Esqueci a senha mail](#)

```
7 <body>
8     <main>
9         <section id="login">
10             <div id="imagem">
11
12             </div>
13             <div id="formulario">
14                 <h1>Login</h1>
15                 <p>Seja bem-vindo(a) novamente.
16                 Faça login para acessar sua conta e
17                 poder fazer as configurações no seu
18                 ambiente.</p>
19                 <form action="login.php" method="
20                 post"></form>
21
22             </div>
23         </section>
24     </main>
25 </body>
```

Em seguida vamos adicionar o formulário:

```
16 <form action="login.php" method="post">
17     <div class="campo">
18         <span class="material-symbols-outlined">
19             person</span>
20         <input type="email" name="login" id="ilogin"
21             placeholder="seu email" autocomplete="email"
22             maxlength="30" required>
23         <label for="ilogin">Login</label>
24     </div>
```

```

23     <div class="campo">
24         <span class="material-symbols-
outlined">vpn_key</span>
25         <input type="password" name="senha" id="
isenha" placeholder="sua senha"
autocomplete="current-password"
minlength="8" maxlength="20" required>
26         <label for="isenha">Senha</label>
27     </div>
28     <input type="submit" value="Entrar">
29     <a href="esqueci.html" class="botao">
30         Esqueci a senha <span class="material-
symbols-outlined">mail</span>
31     </a>
32 </form>

```

Precisamos definir os `<link>`s entre o documento e as CSS:

```

1 <head>
2     <link rel="stylesheet" href="https://fonts.
3     googleapis.com/css2?family=Material+Symbols+
Outlined:opsz,wght,FILL,GRAD@48,400,0,0"/>
4     <link rel="stylesheet" href="estilos/style.css">
5     <link rel="stylesheet" href="estilos/media-
query.css">
6 </head>

```

O primeiro `<link>` é responsável por exibir os ícones que estão disponíveis no [Google Fonts](https://fonts.google.com/).

Os outros dois serviram para fazer referência a nossa CSS.

Adicionando as cores

No seletor universal, configuramos a parte base do site, especificando a padronização do estilo do conteúdo.

```

1 * {
2     font-family: system-ui, -apple-system, Blink
MacSystemFont, 'Segoe UI', Roboto, Oxygen,
Ubuntu, Cantarell, 'Open Sans', 'Helvetica Neue',
sans-serif;
3     padding: 0px;
4     margin: 0px;
5     box-sizing: border-box;
6 }

```


Na parte `<html>` e `<body>`, fizemos de modo que preencham toda a tela do dispositivo em largura e altura.

Por semântica, utilizamos o `<main>` para depois centralizar a `<section>` na página.

```
7 html, body {
8     background-color: #5f2c82;
9     height: 100vh;
10    width: 100vw;
11 }
12
13 main {
14     position: relative;
15     height: 100vh;
16     width: 100vw;
17 }
```

O `<main>` deverá ter sua posição relativa e ocupar toda a tela. Dessa forma podemos posicionar a `<section>` no meio, fazendo algumas especificações:

```
18 section#login {
19     position: absolute;
20     top: 50%;
21     left: 50%;
22     overflow: hidden;
23     background-color: white;
24     height: 515px;
25     width: 250px;
26     border-radius: 20px;
27     box-shadow: 0px 0px 10px rgba(0, 0, 0, 0.450);
28     transition: width .3s, height .3s;
29     transition-timing-function: ease;
30     transform: translate(-50%, -50%);
31 }
```

Especificamos `overflow: hidden;` (tente remove-lo) para esconder a imagem dentro da `<section>` de bordas arredondadas.

Na parte de efeitos, adicionamos sombra e uma transição suave entre as variações de tamanho.

Para exibirmos a imagem dentro da `<div>`, utilizamos `background`:

```
32 section#login > div#imagem {
33     display: block;
34     background: #5f2c82 url("../imagens/metal.jpg")
35     no-repeat;
36     background-size: cover;
37     background-position: left bottom;
38     height: 200px;
39 }
```

Para a parte do formulário, fizemos as configurações gerais para o conteúdo dentro da `<div>`:

```
40 section#login > div#formulario {
41     display: block;
42     padding: 10px;
43 }
44 div#formulario > h1 {
45     text-align: center;
46     margin-bottom: 10px;
47 }
48
49 div#formulario > p {
50     font-size: 0.8em;
51 }
```

Para os campos de entrada, fizemos com que ocupassem toda largura e sua altura de 40px, também arredondamos as bordas e definimos um `border`, assim o conteúdo que está dentro não ficará totalmente sobreposto.

E por fim, utilizamos `margin` para separar cada campo.

```
52 form > div.campo {
53     background-color: #5f2c82;
54     border: 2px solid #5f2c82;
55     display: block;
56     height: 40px;
57     width: 100%;
58     border-radius: 8px;
59     margin: 5px 0px;
60 }
```

Apesar de usarmos `<label>`s, estamos escondendo sua exibição, porém sua importância é semântica:

```
61 div.campo > label {  
62     display: none;  
63 }
```

No lugar, utilizamos os ícones para fazer referência com o tipo de entrada.

Sua largura é de 40px, mesmo com o `padding` de 5px. Isto pois, anteriormente especificamos o valor `box-sizing`, dando um tamanho constante para todos os elementos.

Eu utilizei um `transform` para buscar uma posição mais centralizada dentro da `<div>`.

```
64 div.campo > span {  
65     color: white;  
66     font-size: 2em;  
67     width: 40px;  
68     padding: 5px;  
69     transform: translateY(-5px);  
70 }
```

O `<input>` ocupa 100% do espaço que sobrou da `<div>`, para fazer isto com a largura, utilizamos `calc()` para calcular o espaço total menos o ícone.

Zeramos a borda para cancelar o padrão do `<input>`.

Depois, com o `padding` deixamos o texto mais afastado da borda.

Por fim, adicionamos um `transform` para posicionar melhor o `<input>` dentro da `<div>`.

```
71 div.campo > input {  
72     background-color: #94cfcd;  
73     font-size: 1em;  
74     height: 100%;  
75     width: calc(100% - 45px);  
76     border: 0px;  
77     border-radius: 8px;  
78     padding: 4px;  
79     transform: translate(0px, -13px);  
80 }
```

Adicionamos também um foco dentro do `<input>`, isto é, quando a caixa estiver selecionada a cor de fundo ficará branca e cancelaremos a borda.

```
81 div.campo > input:focus-within {  
82     background-color: white;  
83     border: none;  
84 }
```

Na submissão do formulário (Entrar), mantemos as mesmas configurações de tamanho e borda, apenas adicionamos outra cor e especificamos um `pointer` para alterar o cursor do mouse.

```
85 form > input[type=submit] {  
86     display: block;  
87     font-size: 1em;  
88     height: 40px;  
89     width: 100%;  
90     background-color: #49a09d;  
91     color: white;  
92     border: none;  
93     border-radius: 5px;  
94     cursor: pointer;  
95 }
```

Também, adicionamos `:hover` para alterar a cor de fundo.

```
96 form > input[type=submit]:hover {  
97     background-color: #2d6462;  
98 }
```

Na última parte do login, implementamos um “botão” de “esqueci a senha”, que na verdade é um elemento `<a>`, simplificando o processo para não precisar utilizar Java Script.

Dessa forma, o botão ganhou o mesmo tamanho, **border**, **padding**, **margin** dos demais elementos.

Então apenas, personalizamos o posicionamento do texto com o alinhamento centralizado, bordas mais arredondadas, um **padding-top** e eliminamos a decoração padrão.

```
99   form > a.botao {
100     display: block;
101     text-align: center;
102     font-size: 1em;
103     height: 40px;
104     width: 100%;
105     padding-top: 5px;
106     margin-top: 5px;
107     background-color: white;
108     color: #49a09d;
109     border: 1px solid #49a09d;
110     border-radius: 7px;
111     text-decoration: none;
112 }
```

Também, adicionamos um **:hover** para alterar a cor.

```
113 form > a.botao:hover {
114     background-color: #6cd3cf;
115 }
```

Para o detalhe final, implementamos o estilo do ícone:

```
116 form > a.botao > span {
117     font-size: 0.8em;
118 }
```

Adicionando a media-query

```
1 @media screen and (min-width: 768px) and (max-width: 992px) {
2   /* Configuração para Tablet */
3   body {
4     background-image: linear-gradient(to top,
5       #49a09d, #5f2c82);
6   }
7   section#login {
8     height: 280px;
9     width: 80vw;
10  }
11  section#login > div#imagem {
12    float: left;
13    height: 100%;
14    width: 30%;
15  }
16  section#login > div#formulario {
17    float: right;
18    width: 70%;
19  }
```

Definimos um gradiente para a nova abordagem de mídia. Então, alteramos o tamanho da **<section>**.

A imagem está para a esquerda, pois mudamos seu posicionamento com **float**. Sua distribuição dentro da **<section>** está em 100% de altura e apenas 30% de largura.

O formulário ao contrário está para a direita e ocupa o resto da largura da **<section>**.

```

20 @media screen and (min-width: 992px) {
21     /* Configuração para desktop */
22     body {
23         background-image: linear-gradient(to top,
24             #49a09d, #5f2c82);
25     }
26     section#login {
27         height: 350px;
28         width: 950px;
29     }
30
31     section#login > div#imagem {
32         float: right;
33         height: 100%;
34         width: 50%;
35     }
36
37     section#login > div#formulario {
38         float: left;
39         width: 50%;
40     }
41
42     div#formulario > h1 {
43         font-size: 2em;
44     }
45
46     div#formulario > p {
47         font-size: 1em;
48         margin: 20px 0px;
49     }
50 }

```

Atribuímos uma nova abordagem para o desktop, a `<section>` ocupa um espaço maior e alteramos os valores de tamanho da imagem e do formulário.

A imagem está para a direita e ocupa a metade da `<section>`.

O formulário está para esquerda e ocupa a outra metade.