Professor Rodrigo Bossini
www.rodrigobossini.com.br

Introdução a CSS

1 Introdução

Como vimos, a linguagem de marcação HTML5 é utilizada para descrever a estrutura de um documento, sem que sejam especificados detalhes sobre a sua aparência. Utilizamos CSS para estilizar elementos HTML. Neste guia, iremos aprender mais recursos que o CSS oferece.

2 Passo a passo

Utilizaremos trechos de HTML5 simples para ilustrar os principais recursos de CSS.

2.1 (Criando uma pasta) Crie uma nova pasta para abrigar os exemplos. De preferência ela não pode ter caracteres especiais e/ou espaços em seu nome. Cuidado também para não utilizar nenhuma pasta ou sub-pasta que tenha restrições de segurança impostas pelo seu sistema operacional.

2.2 (Abrindo o VS Code) Abra uma instância do VS Code vinculada a essa pasta. Isso pode ser feito abrindo-se o VS Code e clicando-se em File >> Open Folder. Também é possível abrir um terminal, navegar até a pasta criada e executar o comando

code .

As duas formas são equivalentes. Escolha a que for mais conveniente para você.

2.3 (Propriedade display) Nesta seção, iremos estudar sobre a propriedade display. Para isso, crie um arquivo html com nome **exemplo_display.html** em sua pasta. Seu conteúdo inicial é dado na Listagem 2.3.1.

Listagem 2.3.1

```
<!DOCTYPE html>
<html >
<head>
<meta charset="UTF-8">
<title>A propriedade display</title>
<style>
p {
background-color: #CCC;
}

a {
background-color: rgb(202, 184, 231);
}

footer{
background-color: beige;
}
button {
background-color: greenyellow;
}
</style>
</head>
<body>
<article id="cursos">
<p class="engenharia">Engenharia</p>
<p class="farmacia">Farmácia</p>
<a class="professores" href="#">Veja a lista de professores aqui</a>
<a class="Contato" href="#">Entre em contato</a>
<footer class="rodape">Universidade AprendaJa</footer>
<button class="botao">Clique aqui para falar conosco</button>
<p>Todos os direitos reservados</p>
</article>
</body>
</html>
```

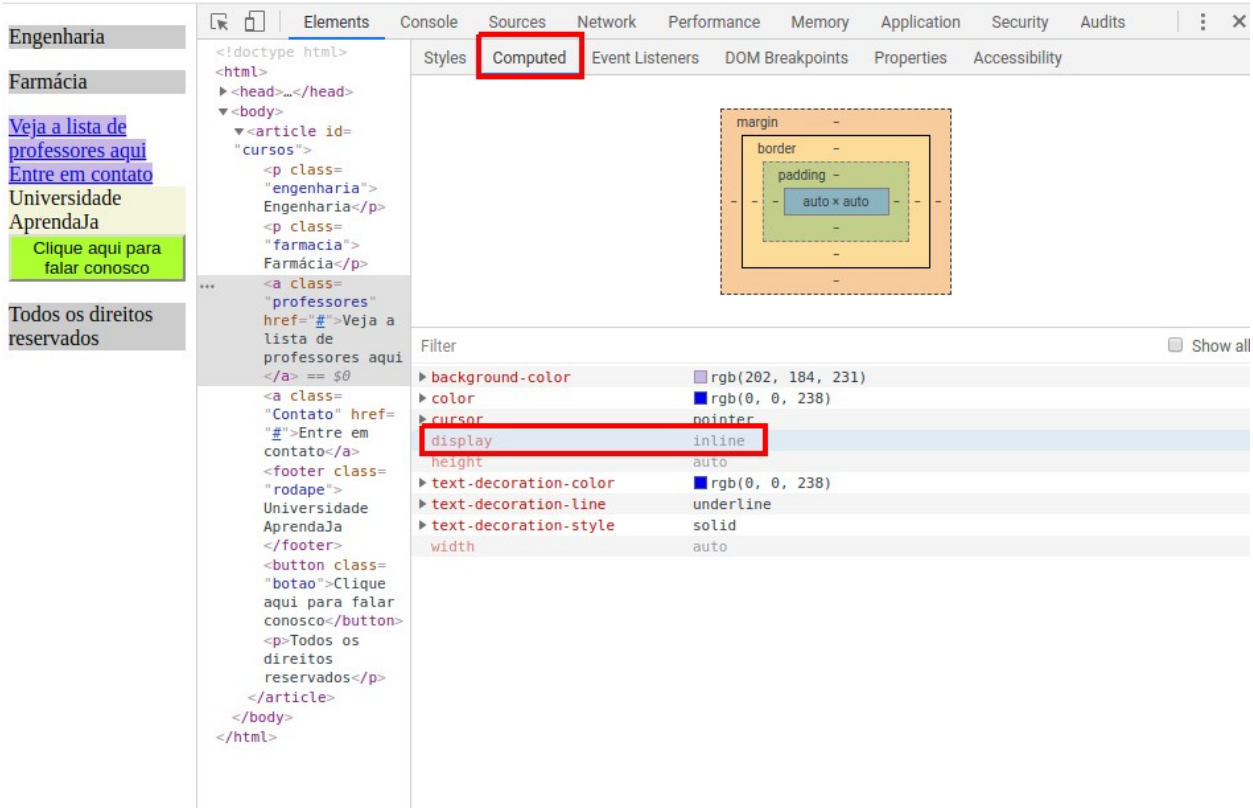
- Abra o arquivo no seu navegador e veja como se dá a exibição dos elementos. Note que alguns elementos aparecem um abaixo do outro enquanto outros estão lado a lado. A propriedade que determina essa característica é a propriedade **display**.

- Por enquanto, estamos lidando com dois valores associado à propriedade display: **block** e **inline**.

block: Quebras de linha são colocadas antes e depois do elemento.
inline: Sem quebras de linha, o próximo elemento será exibido na mesma linha se couber. Ocupa **somente o espaço necessário para o conteúdo**.

- Clique com o direito em um link da página de exemplo e abra o Chrome DevTools. Vá até a aba **Computed**, como mostra a Figura 2.3.1. Note que a propriedade display do link tem valor padrão de **inline**.

Figura 2.3.1



- Repita o processo para olhar os outros elementos, como **button**, **p** e **footer**. Verifique qual o valor padrão de display para cada um deles.

- Note que não podemos especificar **largura** e **altura** para elementos cujo valor de display é **inline**. Essas configurações não têm efeito já que, por definição, um elemento cujo display é inline ocupa somente o espaço necessário para seu conteúdo. Veja a Listagem 2.3.2.

Listagem 2.3.2

```
.professores{
width: 200px;
height: 200px;
}
```

Atualize a página e verifique que nada mudou.

- Contudo, alterando o display para **block**, as propriedades passam a ter efeito. Note também que uma **quebra de linha** é adicionada. Isso é padrão do valor block. Veja a Listagem 2.3.3.

Listagem 2.3.3

```
.professores{
display: block;
width: 200px;
height: 200px;
}
```

Faça também a alteração da Listagem 2.3.4, alterando o display de um **p** para **inline**.

Listagem 2.3.4

```
.engenharia {
display: inline;
}
```

- Agora suponha que desejamos especificar **tamanho** e **altura** para os dois cursos, porém, queremos que eles fiquem lado a lado. Como vimos, a tentativa da Listagem 2.3.5 não irá funcionar como desejamos.

Listagem 2.3.5

```
/*não funciona*/
.engenharia {
display: inline;
width: 300px;
height: 200px;
}
/*não funciona*/
.farmacia {
display: inline;
width: 300px;
height: 200px;
}
```

Como mostra a Figura 2.3.2, o próprio assistente do Visual Studio Code nos informa quais propriedades são ignoradas quando a propriedade display é utilizada.

Figura 2.3.2

- O valor **inline-block** associado à propriedade display permite que essa combinação seja realizada. Teremos os elementos lado a lado e eles poderão definir largura e altura. Veja a Listagem 2.3.6.

Listagem 2.3.6

```
.engenharia {
display: inline-block;
width: 300px;
height: 200px;
}

.farmacia {
display: inline-block;
width: 300px;
height: 200px;
}
```

- O valor **none** da propriedade display esconde o elemento, embora ele ainda exista na árvore DOM. Veja a Listagem 2.3.7.

Listagem 2.3.7

```
.rodape{
display: none;
}
```

2.4 (Propriedade border) Podemos especificar bordas para nossos elementos, dizendo qual a sua cor, espessura, forma entre outras coisas. Para este exemplo, crie um arquivo chamado **exemplo_bordas.html**. Seu conteúdo inicial é dado na Listagem 2.4.1.

Listagem 2.4.1

```
<!DOCTYPE html>
<html >
<head>
<meta charset="UTF-8">
<title>Bordas</title>
</head>
<style>

</style>
<body>

<main>
<div id="cursos">
<p class="engenharia">Engenharia</p>
<p class="farmacia">Farmácia</p>
<a href="#">Mais informações</a>
</div>
</main>

</body>
</html>
```

- Os nomes das propriedades referentes a bordas começam com **border**. Veja alguns exemplos na Listagem 2.4.2.

Listagem 2.4.2

```
a {
border-width: thin;
border-color: red;
border-style: solid;
}
```

Faça testes com outros valores para width e style, sugeridas pelo VS Code.

- Há também o atalho **border**, que pode ser utilizado para especificar todos os valores referentes à borda de uma única vez. Veja o exemplo da Listagem 2.4.3.

Listagem 2.4.3

```
a {
/*border-width: thin;
border-color: red;
border-style: solid;*/
border: thin black dashed;
}
```

- Note que, ao especificarmos a borda, ela apareceu em cima, embaixo e dos dois lados da caixa. Também é possível especificar que desejamos bordas somente em uma ou algumas dessas regiões, não necessariamente as quatro. Veja a Listagem 2.4.4.

Listagem 2.4.4

```
a {
/*border-width: thin;
border-color: red;
border-style: solid;*/
border: thick black dashed;
border-top: 10px solid blue;
}

.farmacia {
border-bottom: 5px solid #CCC;
}

.engenharia{
border-left: 1px solid green;
}
```

Perceba, neste exemplo, que a propriedade mais específica (**border-top**) tem **precedência** sobre a mais geral (**border**).

- A propriedade **border-radius** permite arredondar a borda, como na Listagem 2.4.5.

Listagem 2.4.5

```
.engenharia{
border: 5px solid green;
border-radius: 5px;
}
```

- Repare no que acontece com a cor de fundo e com o conteúdo conforme a borda fica mais arredondada, como na Listagem 2.4.6.

Listagem 2.4.6

```
.engenharia{
border: 5px solid green;
border-radius: 75px;
background-color: beige;
width: 200px;
height: 200px;
}
```

- Veja, na Listagem 2.4.7 como arredondar somente um dos campos, como o canto superior esquerdo.

Listagem 2.4.7

```
.engenharia{
border: 5px solid green;
border-radius: 75px;
background-color: beige;
width: 200px;
height: 200px;
border-top-left-radius: 120px;
}
```

- A própria propriedade **border-radius** pode servir de atalho para que sejam especificados diversos valores de uma única vez. Na Listagem 2.4.8 especificamos **dois** valores.

Listagem 2.4.8

```
engenharia{
border: 5px solid green;
border-radius: 20px 50px;
background-color: beige;
width: 200px;
height: 200px;
}
```

Repare que o primeiro valor se aplica às bordas superior esquerda e inferior direita e, o segundo às bordas superior direita e inferior esquerda.

- Se especificarmos **quatro** valores, eles se aplicam a cada uma das bordas, começando da borda superior esquerda em sentido horário, como na Listagem 2.4.9.

Listagem 2.4.9

```
.engenharia{
border: 5px solid green;
border-radius: 20px 50px 80px 100px;
background-color: beige;
width: 200px;
height: 200px;
}
```

2.5 (Propriedade padding) Podemos especificar medidas para que os conteúdos de nossas caixas não fiquem colados às suas bordas. Essa medida se chama **padding**. Para esse exemplo, crie um novo arquivo chamado **exemplo_padding.html**. Seu conteúdo inicial é dado na Listagem 2.5.1.

Listagem 2.5.1

```
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>Padding</title>
<style>
.primeiro, .segundo{
width: 200px;
display: inline-block;
border: 2px solid black;
background-color: #CCC;
}
</style>
</head>
<body>
<article>
<p class="primeiro">
Lorem ipsum dolor sit amet, consectetur adipisicing elit. Quaerat aspernatur culpa officiis odio quisquam, autem deserunt earum excepturi voluptas. Nisi voluptatem quas quis voluptate voluptates ipsam perspiciatis excepturi voluptatibus perferendis?
</p>
<p class="segundo">
Lorem ipsum dolor sit amet, consectetur adipisicing elit. Quaerat aspernatur culpa officiis odio quisquam, autem deserunt
```

```
earum excepturi voluptas. Nisi voluptatem quas quis voluptate voluptates ipsam perspiciatis excepturi voluptatibus perferendis?
</p></article></body></html>
```

- No primeiro exemplo, vamos colocar **padding** nas duas caixas, usando possibilidade diferentes. Veja a Listagem 2.5.2.

Listagem 2.5.2

```
.primeiro, .segundo{
width: 200px;
display: inline-block;
border: 2px solid black;
background-color: #CCC;
}

.primeiro{
padding: 20px;
}

.segundo {
padding: 20px 40px;
}
```

Note que também há os atributos **padding-left**, **padding-right** etc, de maneira análoga a atributos que já vimos.

- Veja o que ocorre quando adicionamos padding ao **article** da página. Adicionamos também uma borda para o resultado ficar bem claro. Veja a Listagem 2.5.3.

Listagem 2.5.3

```
article{
border: 2px solid blue;
padding: 30px;
}
```

- Também é possível ajustar o próprio **body** do documento, como na Listagem 2.5.4.

Listagem 2.5.4

```
body{
border: 2px solid black;
padding: 50px;
}
```

Observação importante: Note que as caixas têm **largura fixa em 200px**. Porém, quando adicionamos **padding**, o espaço que ocupam aumenta daquela quantidade, embora a caixa continue utilizando 500 pixels para seu conteúdo. A**umente o padding da segunda caixa para 400px para ver o resultado**. Essa forma de funcionamento é padrão no navegador e pode ser alterada, como veremos.

2.6 (Propriedade margin) A margem de um elemento é a medida que há entre suas bordas e elementos externos. Para este exemplo, crie um arquivo chamado **exemplo_margin.html**. Seu conteúdo inicial é dado na Listagemw 2.6.1.

Listagem 2.6.1

```
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>Margin</title>
<style>
</style>
</head>
<body>
<article>
<p>
Lorem ipsum dolor sit, amet consectetur adipisicing elit. Aperiam molestiae, asperiores ducimus esse quidem sequi quasi? Eius voluptate quo fuga quibusdam optio, aperiam eum dolorum est! Ex laboriosam ullam a.
</p>
<p>
Lorem ipsum dolor sit, amet consectetur adipisicing elit. Aperiam molestiae, asperiores ducimus esse quidem sequi quasi? Eius voluptate quo fuga quibusdam optio, aperiam eum dolorum est! Ex laboriosam ullam a.
</p>
</article>
</body>
</html>
```

- Comece mexendo na margem à esquerda da primeira caixa, como na Listagem 2.6.2.

Listagem 2.6.2

```
.primeira {
margin-left: 30px;
}
```

- Note que, zerando a margem da primeira caixa, ela não fica completamente colada na tela. Veja a Listagem 2.6.3.

Listagem 2.6.3

```
.primeira {
margin-left: 0px;
}
```

- Coloque um borda no elemento **body** para que isso fique mais claro, como na Listagem 2.6.4.

Listagem 2.6.4

```
body{
border: 1px solid blue;
}
```

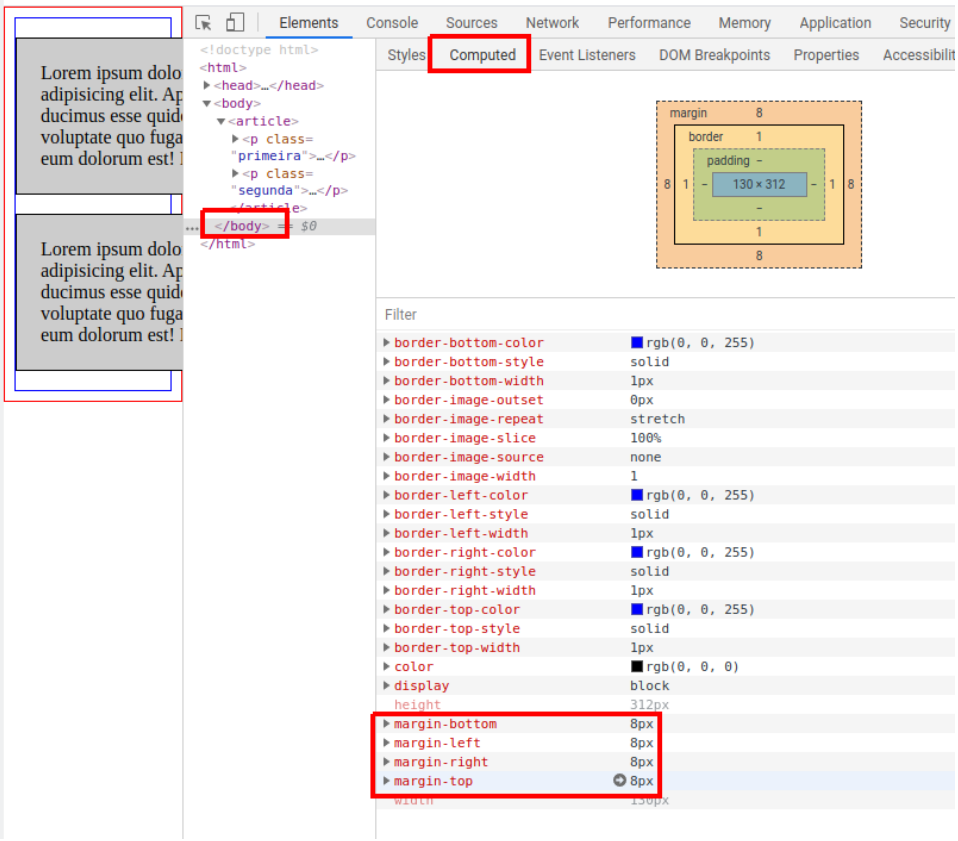
- Sendo assim, coloque uma borda no elemento **html** também, como na Listagem 2.6.5.

Listagem 2.6.5

```
html{
border: 1px solid red;
}
```

- Repare nos espaços existentes entre as bordas de **html** e **body**. Vamos abrir o Chrome DevTools para entender o que está acontendo. Aperte CTRL + SHIFT + I. Vá até a aba **Computed** e passe o mouse sobre o **body**. Verifique o que está ocorrendo com as suas margens. Por padrão, o navegador colocou 8 pixels de margem no body. Veja a Figura 2.6.1.

Figura 2.6.1



- Evidentemente podemos remover essa medida, caso desejado. Basta **zerar** a margem do elemento **body**, como na Listagem 2.6.6.

Listagem 2.6.6

```
body{
margin: 0px;
border: 1px solid blue;
}
```

- Veja que os elementos **p** também têm margem padrão, que podemos zerar, conforme a Listagem 2.6.7.

Listagem 2.6.7

```
.primeira, .segunda {
background-color: #CCC;
border: 1px solid black;
padding: 20px;
width: 300px;
margin: 0px;
}
```

- Também dá para especificar as margens usando **dois** ou **quatro** valores, como na Listagem 2.6.8. Caso **dois** valores sejam especificados, o primeiro será aplicado à margem superior e à margem inferior e o segundo será aplicado à margem esquerda e à margem direita. Caso **quatro** valores sejam especificados eles são usados pelas margens **superior**, **direita**, **inferior** e **direita**, nesta ordem.

Listagem 2.6.8

```
<p class="terceira">
Lorem ipsum dolor sit, amet consectetur adipisicing elit. Aperiam molestiae, asperiores ducimus esse quidem sequi quasi? Eius voluptate quo fuga quibusdam optio, aperiam eum dolorum est! Ex laboriosam ullam a.
</p>
<p class="quarta">
Lorem ipsum dolor sit, amet consectetur adipisicing elit. Aperiam molestiae, asperiores ducimus esse quidem sequi quasi? Eius voluptate quo fuga quibusdam optio, aperiam eum dolorum est! Ex laboriosam ullam a.
</p>

.terceira, .quarta{
border: 1px solid green;
}
.terceira{
margin: 20px 50px;
}

.quarta {
margin: 10px 50px 100px 200px;
}
```

Note, ainda no exemplo anterior, que **as margens positivas se sobrepõe.**

- As margens também podem ser negativas, como mostra a Listagem 2.6.9.

Listagem 2.6.9

```
<p class="quinta">
Lorem ipsum dolor sit, amet consectetur adipisicing elit. Aperiam molestiae, asperiores ducimus esse quidem sequi quasi? Eius voluptate quo fuga quibusdam optio, aperiam eum dolorum est! Ex laboriosam ullam a.
</p>
.quinta{
margin-top: -50px;
border: 1px solid yellow;
}
```

Note que a margem inferior do quarto elemento **p** tem dela subtraido 50 pixels, que é a margem superior do quinto elemento **p.**

- O valor auto costuma ser utilizado para centralizar uma caixa. Informalmente, ele instrui o navegador a escolher um valor conveniente, de acordo com o contexto. Veja a Listagem 2.6.10. Para mais informações sobre o valor auto, veja o Link 2.6.1.

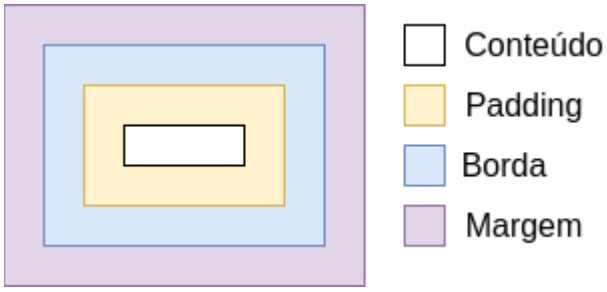
Link 2.6.1
https://drafts.csswg.org/css2/visudet.html#Computing_widths_and_margins

Listagem 2.6.10

```
.quinta{
margin-top: -50px;
border: 1px solid yellow;
margin: 20px auto;
width: 300px;
}
```

- Concluindo, a Figura 2.6.2 ilustra as medidas estudadas até então. Essa estrutura é conhecida como **Box Model** do CSS.

Figura 2.6.2



2.7 (Box sizing) Conforme especificamos valores de medidas para as propriedades de borda, padding etc, o navegador faz seus cálculos para renderizar a caixa da maneira mais adequada. Nesta seção, vamos estudar a propriedade **box-sizing** do CSS e entender os efeitos causados considerando o Box Model. Crie um arquivo chamado **exemplo_boxsizing.html**. Seu conteúdo inicial é dado na Listagem 2.7.1.

Listagem 2.7.1

```
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>Box Sizing</title>
<style>
body{
margin: 0px;
}
.caixa1, .caixa2{
width: 300px;
text-align: justify;
background-color: #CCC;
border: 1px solid black;
}

.rodape{
background-color: aliceblue;
padding: 12px;
```

```
border: 1px solid #EEE;
text-align: center;
}
</style>
</head>
<body>

<article>
<p class="caixa1">
Lorem ipsum dolor sit amet consectetur adipisicing elit. Est, nam! Velit corporis, facere sapiente ex vero accusantium minima. Excepturi consequatur fugiat incidunt ipsam molestiae soluta magnam quo neque magni adipisci.
</p>
<p class="caixa2">
Lorem ipsum dolor sit amet consectetur adipisicing elit. Est, nam! Velit corporis, facere sapiente ex vero accusantium minima. Excepturi consequatur fugiat incidunt ipsam molestiae soluta magnam quo neque magni adipisci.
</p>
<footer class="rodape">
Volte sempre
</footer>
</article>
</body>
</html>
```

- Vamos começar adicionando uma borda ao primeiro **p**, como na Listagem 2.7.2.

Listagem 2.7.2

```
.caixa1{
border: 5px solid black;
}
```

Note que, graças à borda, embora ambas as caixas possuam largura de 300 pixels, a primeira caixa parece ser mais larga.

Especificando um **padding**, como na Listagem 2.7.3, a caixa aumenta ainda mais.

Listagem 2.7.3

```
.caixa1{
border: 5px solid black;
padding: 50px;
}
```

É aí que entra a propriedade **box-sizing**. Ela tem alguns valores possíveis e **o valor padrão é content-box**. Ele indica que **a medida width se refere somente ao conteúdo**. As demais propriedades (border e padding) não entram na conta.

- Embora seja padrão, podemos deixar esse valor explícito, como mostra a Listagem 2.7.4.

Listagem 2.7.4

```
.caixa1{
border: 5px solid black;
padding: 50px;
box-sizing: content-box; /*padrão, não vai mudar nada*/
}
```

- Podemos fazer diferente. Podemos indicar ao navegador que a largura configurada na propriedade **width** é a desejada para o **somatório de conteúdo, padding e borda**. Para tal, basta trocar o valor de box-sizing para **border-box**, como na Listagem 2.7.5.

Listagem 2.7.5

```
.caixa2{
border: 5px solid black;
padding: 50px;
box-sizing: border-box; /*padrão, não vai mudar nada*/
}
```

- É interessante inspecionar os elementos com o Chrome DevTools. Na aba **Computed**, verifique os valores que compõe a Largura de cada um. Veja as figuras 2.7.1 e 2.7.2.

Figura 2.7.1 – (**caixa1**)

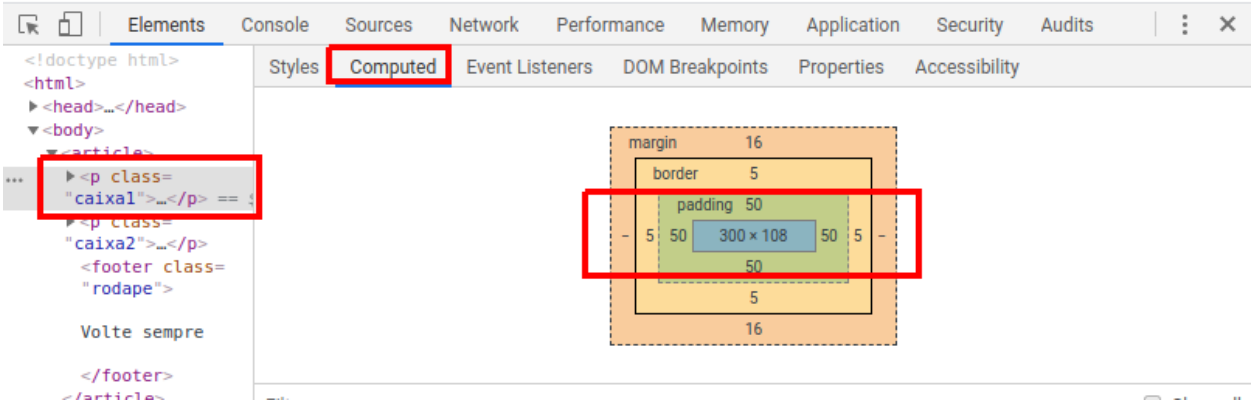
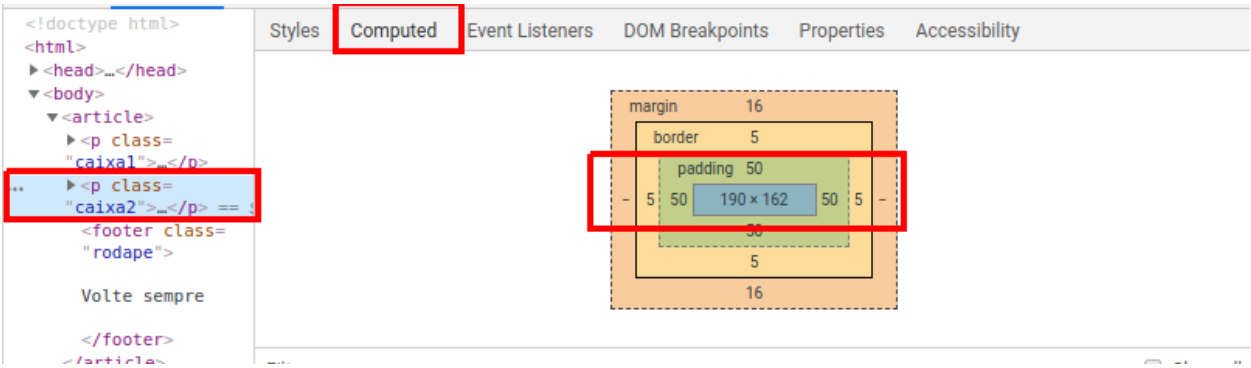


Figura 2.7.2 (**caixa 2**)



2.8 (Responsividade) Passamos agora a falar de um tópico muito importante no desenvolvimento web: a responsividade. Nos dias atuais, usuários acessam web sites utilizando dispositivos com as mais variadas características. Em particular, utilizando dispositivos que têm tamanho de tela diferente, resolução diferente entre outras coisas.

Quando fazemos nossas páginas Web, o ideal é que elas exibam conteúdo de maneira apropriada para o dispositivo que está em uso. Ou seja, que elas se adaptem de acordo com as necessidades do usuário. Se o dispostivo tem tela pequena (horizontalmente falando), por exemplo, é melhor exibir um pouco de conteúdo por vez e permitir que o usuário faça rolagem do que tentar exibir muito conteúdo que ficaria muito pequeno ou cortado. Com uma tela grande, o site pode exibir mais conteúdo com objetos maiores, de uma única vez.

Essa característica de adaptação quanto ao tamanho da tela do dispositivo é chamada de **responsividade**.

Há diversos frameworks que fornecem muitas funcionalidades voltadas para isso.

Nesta seção, veremos que é possível utilizar CSS puro para conseguir bom nível de responsividade.

Veja, no Link 2.8.1, um exemplo de site não responsivo. Reduza a largura de seu navegador e repare no menu superior.

Link 2.8.1
https://dequeuniversity.com/library/responsive/1-non-responsive

Por outro lado, o Link 2.8.2 mostra um exemplo de site responsivo.

Professor Rodrigo Bossini
www.rodrigobossini.com.br

Link 2.8.2
https://angular.io/docs

2.8.1 (Pixel) As telas dos dispositivos têm uma quantidade determinada de pixels. Por exemplo, se sua tela é **FULL HD**, ela tem **1920** pixels horizontalmente e **1080** pixels verticalmente. Se a tela é **HD** ela tem **1280** pixels horizontalmente e **720** pixels verticalmente. Dispositivos 4K têm diferentes quantidades de pixels. Uma bastante comum é **3840** pixels horizontalmente e **2160** pixels verticalmente. Aliás, **4K** vem da ideia de o dispositivo ter em torno de **4 mil (4K)** pixels na horizontal.

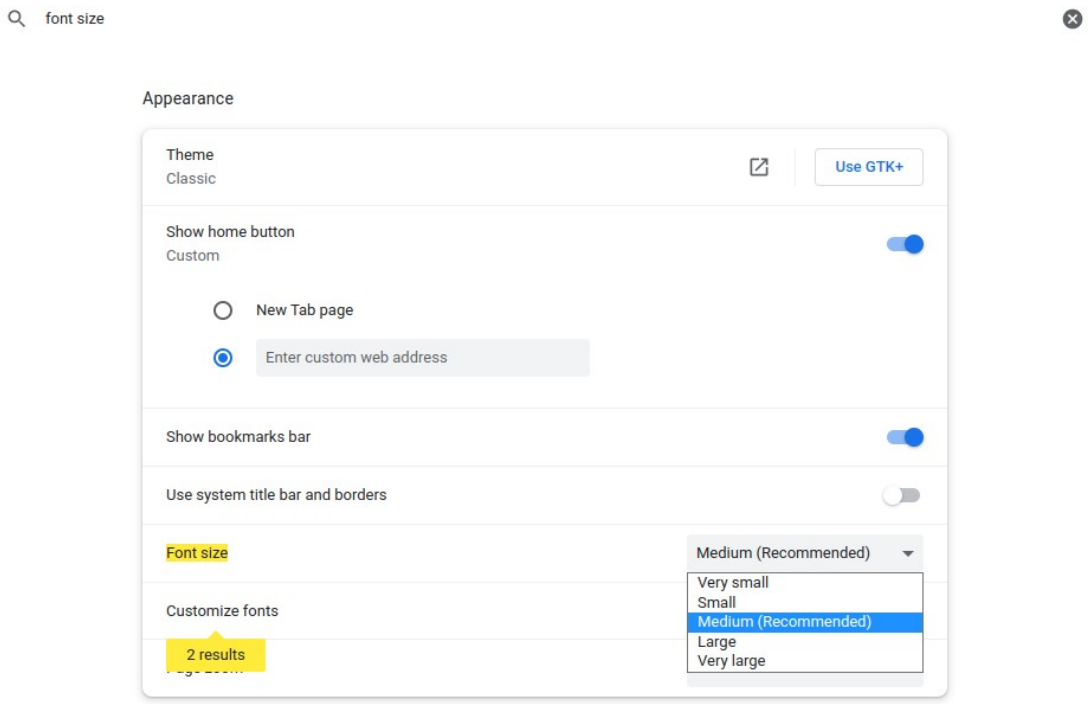
- Começamos a discussão considerando unidades de medida. A primeira será o **pixel**. Para esse exemplo, crie um arquivo chamado **exemplo_pixel.html**. Seu conteúdo é dado na Listagem 2.8.1.1

Listagem 2.8.1.1

```
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<style>

</style>
<title>Pixel</title>
</head>
<body>

<article>
<h1>Desenvolvimento de aplicações web</h1>

<p>
Lorem ipsum dolor sit amet consectetur adipisicing elit. Illo voluptatum rem, magnam ad minima tempore. Corrupti asperiores hic, amet nam magni impedit possimus, saepe vel blanditiis ea, nihil a reprehenderit.
</p>
</article>
</body>
</html>
```

- Note que, evidentemente, a fonte exibida pelo navegador tem um tamanho pré determinado. Esse tamanho é definido pelo próprio navegador, caso o desenvolvedor não especifique um tamanho. Isso pode ser alterado pelo usuário nas configurações do navegador. No Chrome, clique nas três bolinhas no canto superior direito e escolha **Settings (Configurações)**. Busque por **font size** na barra de pesquisa e note que ele oferece algumas opções para alteração de tamanho**,** de very small até very large. Claro, cada nome desse deve estar associado a uma quantidade real de pixels:

-



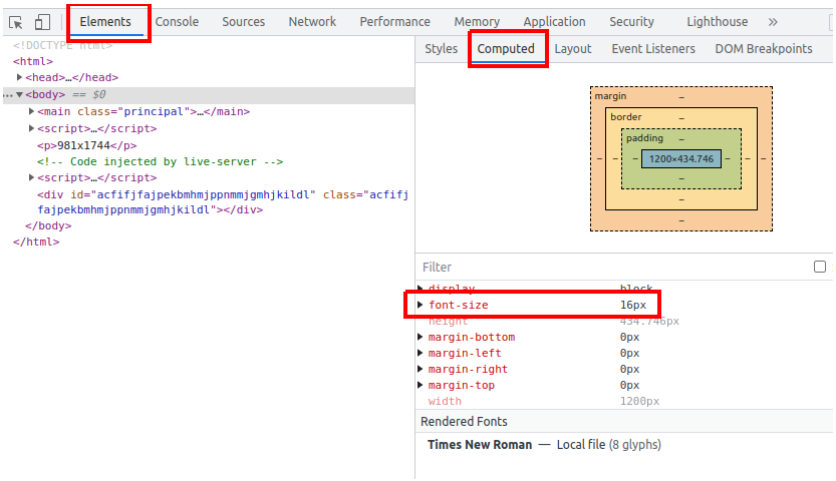
Em geral, o tamanho padrão da fonte (Medium) é 16px, valor escolhido arbitrariamente pelo navegador. Você pode checar isso no Google DevTools. Para isso, é preciso **especificar explicitamente o tamanho da fonte como 100%**. Não alteramos nada, só deixamos explicito para ser possível visualizar no Chrome:

Veja a Listagem 2.8.1.2.

Listagem 2.8.1.2

```
html{
font-size: 100%;
}
```

Agora verifique no Chrome Dev Tools (CTRL + SHIFT + I):



- Como vimos, podemos especificar o tamanho de fonte usando **font-size**. Faça a alteração da Listagem 2.8.1.3 e faça os testes no Chrome novamente, tentando alterar o tamanho de fonte.

Listagem 2.8.1.3

```
body{
font-size: 14px;
}
```

Note que, configurando o tamanho de fonte usando a medida **pixel**, acabamos tirando a chance de o usuário alterar o tamanho da fonte de acordo com suas necessidades. Para verificar isso, altere a fonte no Chrome para algumas opções diferentes como Very Small ou Very Large. Repare que o tamanho da fonte na página não muda. Está fixo em 14px.

- Agora vamos especificar a largura do elemento article. Digamos que ele tem 800 pixels. Veja a Listagem 2.8.1.4.

Listagem 2.8.1.4

```
article{
border: 2px solid #CCC;
background-color: aliceblue;
width: 800px;
}
```

Diminua a largura do seu navegador e veja que a caixa permanece fixa, com seus 800 pixels de largura.

- Caso o padding seja configurado, como na Listagem 2.8.1.5, o resultado é o mesmo. A aparência não muda conforme a largura do navegador muda.

Listagem 2.8.1.5

```
article{
border: 2px solid #CCC;
background-color: aliceblue;
width: 800px;
padding: 20px;
}
```

Perceba, portanto, que a unidade pixel, quando aplicada a determinados elementos, pode ser indesejável do ponto de vista da responsividade.

2.8.2 (Unidade percentual) Começamos agora a falar sobre a medida percentual, que pode ser mais apropriada em alguns cenários. Para esta seção, crie um arquivo chamado **exemplo_percentual.html**. Seu conteúdo inicial é dado na Listagem 2.8.2.1.

Listagem 2.8.2.1

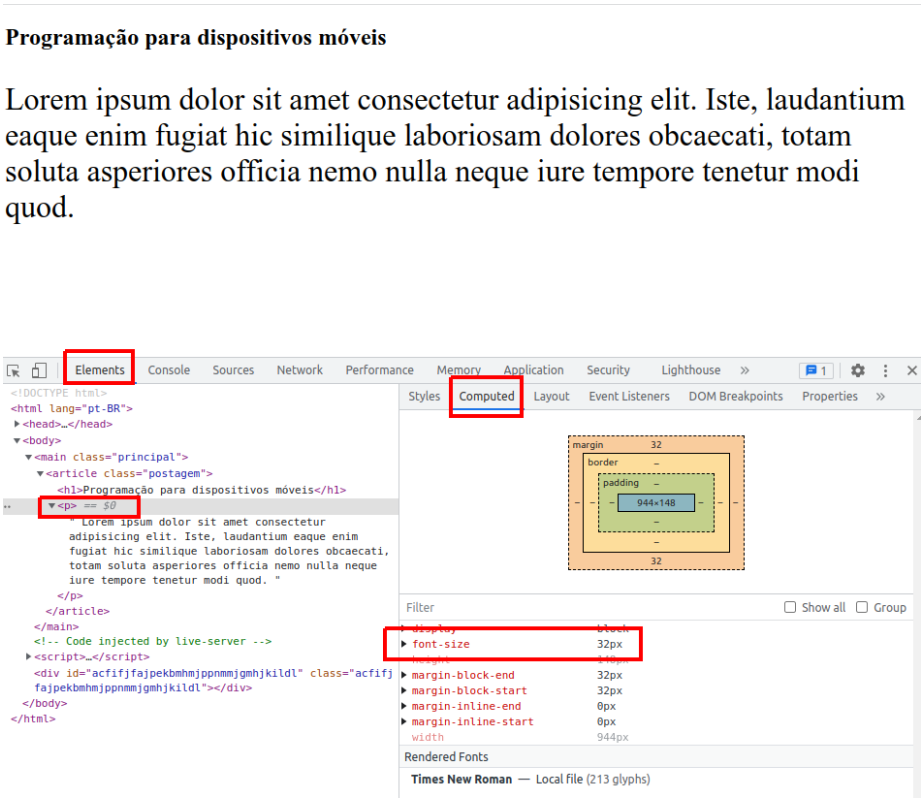
```
<!DOCTYPE html>
<html lang="pt-BR">

<head>
<meta charset="UTF-8">
<title>Percentual</title>
<style>

</style>
</head>

<body>
<main class="principal">
<article class="postagem">
<h1>Programação para dispositivos móveis</h1>
<p>
Lorem ipsum dolor sit amet consectetur adipisicing elit. Iste, laudantium eaque enim fugiat hic similique laboriosam dolores obcaecati, totam soluta asperiores officia nemo nulla neque iure tempore tenetur modi quod.
</p>
</article>
</main>
</body>
</html>
```

- Vamos começar alterando o tamanho da fonte, como na Listagem 2.8.2.2.

Listagem 2.8.2.2

```
<!DOCTYPE html>
<html lang="pt-BR">

<head>
<meta charset="UTF-8">
<title>Percentual</title>
<style>
p {
font-size: 200%;
}
</style>
</head>
<body>
...
```

Vá no Google Chrome Tools e inspecione o elemento **p** na aba computed. Veja o valor associado à propriedade **font-size:**



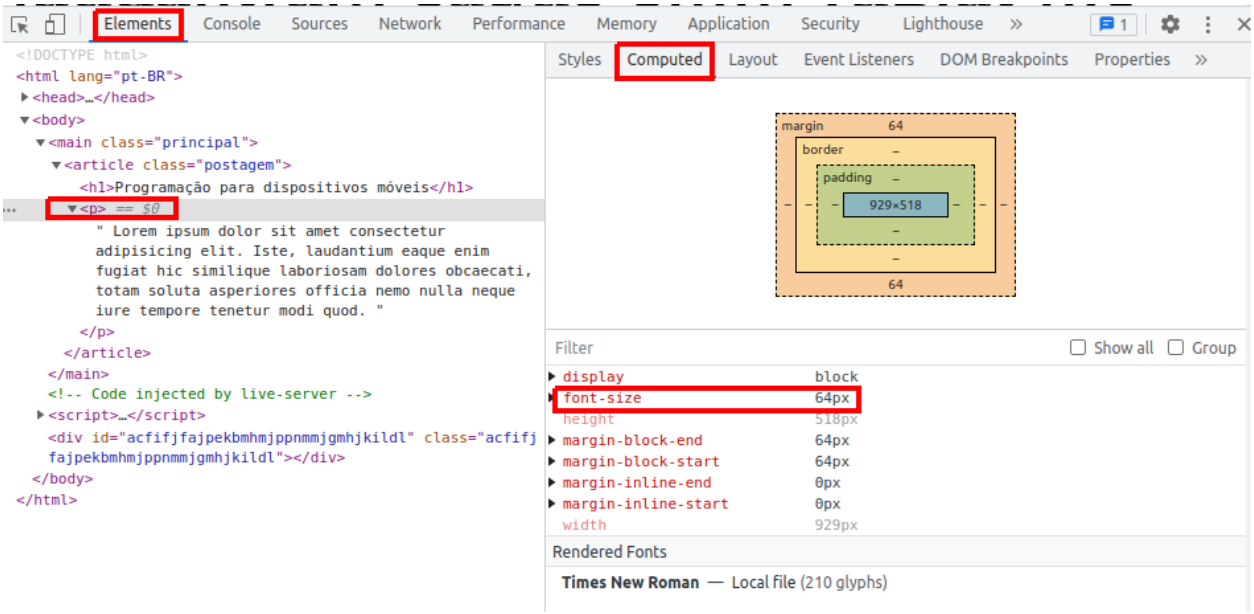
Tente também alterar o tamanho de fonte nas configurações do Chrome, escolhendo opções como **Very Small** e **Very Large**. Observe que, desta vez, a fonte muda de tamanho.

- A seguir, especifique o tamanho de fonte do elemento article, com na Listagem 2.8.3.3. Lembre-se que algumas propriedades são herdadas, como é o caso da font-size.

Listagem 2.8.3.3

```
<style>
p{
font-size: 200%;
}
article{
font-size: 200%;
}
</style>
```

Use o Dev Tools para encontrar o tamanho de fonte do elemento p agora. O percentual especificado no elemento p é relativo ao seu antecessor imediato, o article neste caso.



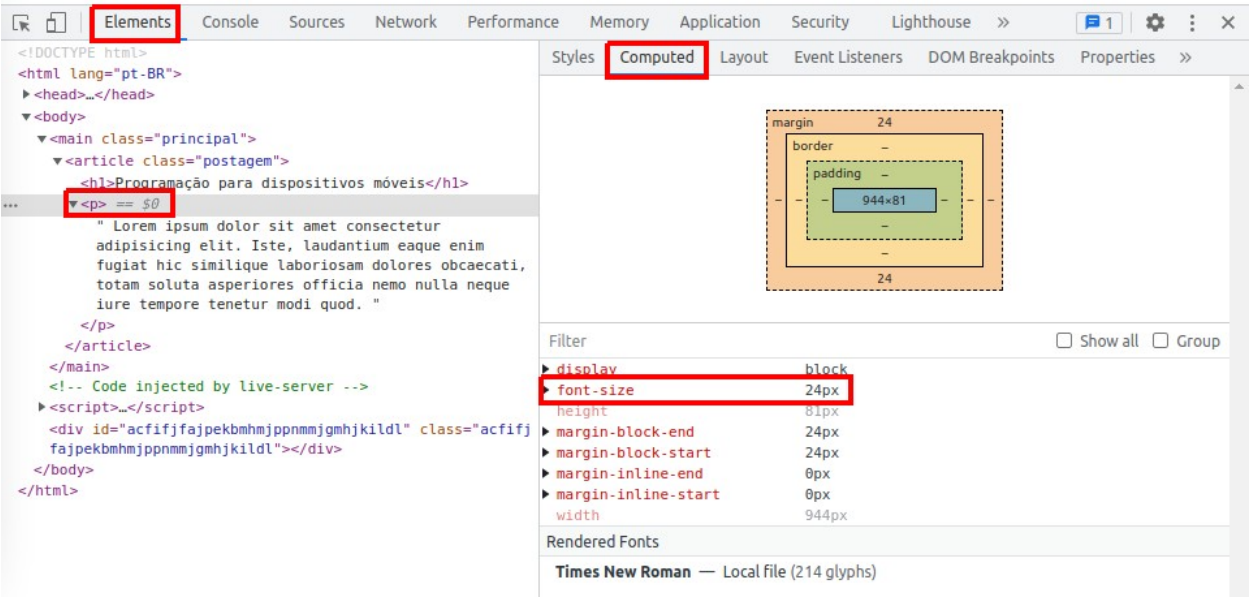
- Embora seja interessante utilizar a medida percentual, muitas vezes podemos desejar **especificar uma medida determinada, que depende da configuração do navegador do usuário**. Por exemplo, talvez queiramos especificar que o tamanho de fonte deve ser de **24 pixels** caso o usuário tenha configurado seu navegador para o tamanho de fonte padrão, que é 16px. Não queremos fazer isso usando a medida pixel para não tirar do usuário a possibilidade de alterar o tamanho da fonte. **Queremos especificar, portanto, o tamanho de nossa fonte, como um percentual em relação à fonte escolhida pelo usuário.**

Em uma conta básica: 24 / 16 = 1.5 encontramos o percentual que 24 representa de 16. Neste caso, 150%, ou seja, 50% maior. Assim, podemos fazer a especificação usando a medida percentual, como na Listagem 2.8.3.4.

Listagem 2.8.3.4

```
<style>
p{
font-size: 150%;
}
/*não vamos usar por enquanto
article{
font-size: 200%;
}*/
</style>
```

Use novamente o Chrome Tools para checar o tamanho da fonte computado para o elemento **p:**



Em outras palavras, quando o usuário escolher a opção **Medium (Recommended)** no Chrome, ele estará escolhendo a fonte de tamanho 24px e não 16px. E, ainda assim, terá a oportunidade de escolher as outras opções, pois a especificação que fizemos não ficou engessada com uma quantidade fixa de pixels. Ela é percentual.

- A conta pode ser tornar mais simples se tomarmos como base de comparação o valor 10. Para isso, vamos primeiro calcular o percentual que 10 representa de 16

$$10 / 16 = 0{,}625$$

Ou seja, 10 é igual a 62,5% de 16. Sendo assim, vamos dizer que a fonte inicial é essa, em percentual, como na Listagem 2.8.3.5.

Listagem 2.8.3.5

```
...
<style>
html {
font-size: 62.5%;
}

p {
font-size: 150%;
}

/*não vamos usar por enquanto
article{
font-size: 200%;
} */
</style>
...
```

Feito isso, para que o tamanho da fonte de **p** seja de 24 pixels caso o usuário tenha a configuração padrão em seu navegador, precisamos do ajuste da Listagem 2.8.3.6.

Listagem 2.8.3.6

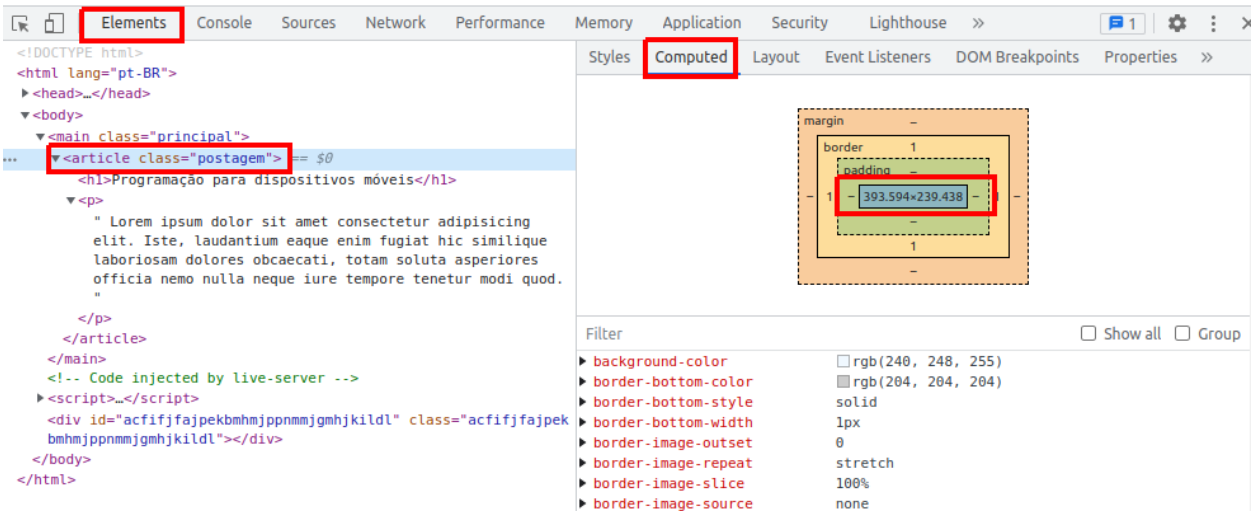
```
...
<style>
html {
font-size: 62.5%;
}

p {
font-size: 240%;
}

/*não vamos usar por enquanto
article{
font-size: 200%;
} */
</style>
...
```

- Agora vamos utilizar a medida percentual para ajustar a largura do elemento article. Quando o fazemos, **a medida é calculada em relação ao elemento de que ele é filho**. Veja o exemplo da Listagem 2.8.3.7.

Listagem 2.8.3.7

```
...
<style>
html {
font-size: 62.5%;
}

p {
font-size: 240%;
}

article {
background-color: aliceblue;
border: 1px solid #CCC;
width: 40%
}
</style>

...
```

Abra o Chrome Tools novamente e faça testes alterando a largura do navegador, olhando a largura dos componentes na aba **Computed**. Veja um exemplo:



- A medida para componentes é sempre relativa à largura do componente em que estiverem inseridos, **mesmo para medidas verticais**. Veja o exemplo da Listagem 2.8.3.8, em que alteramos a **margem superior** do **article**.

Listagem 2.8.3.8

```
...
<title>Percentual</title>
<style>
html {
font-size: 62.5%;
}

p {
font-size: 240%;
}

article {
background-color: aliceblue;
border: 1px solid #CCC;
width: 40%;
margin-top: 25%;
}
</style
...
```

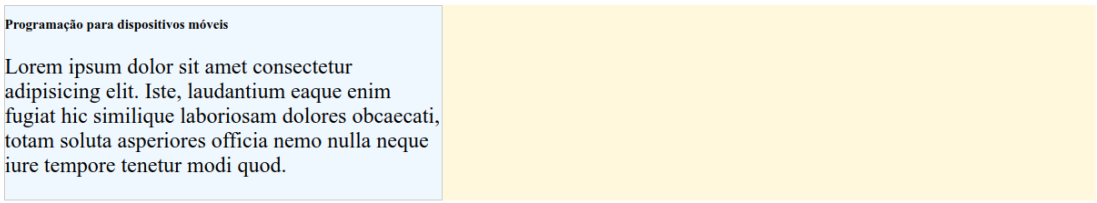
Novamente, faça testes alterando a largura do navegador e olhando os resultados na aba **Computed** do Chrome Dev Tools. Dê foco às medidas de margem superior:

- Na alteração da Listagem 2.8.3.9, verificaremos que o percentual aplicado sempre se refere ao pai do componente.

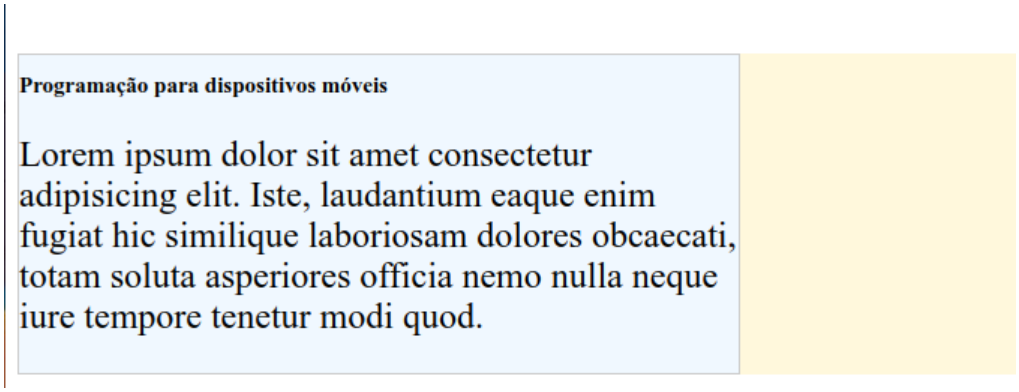
Listagem 2.8.3.10

```
.principal{
background-color: cornsilk;
width: 1200px;
margin: 0 auto;
}
```

- Testando no Dev Tools, veja que a largura do article não muda, já que ela está configurada para 40% de 1200 pixels, que é a medida fixa de seu pai. Com a tela grande, a caixa cabe inteira e ainda sobra espaço:

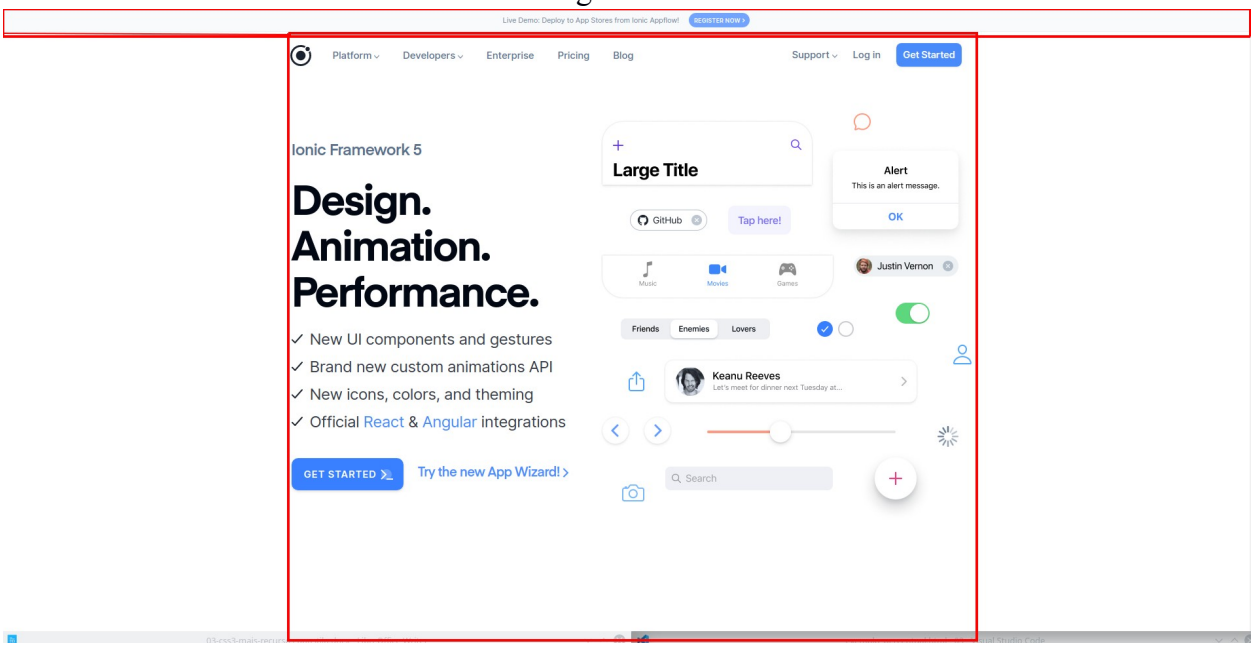


Com a tela pequena (horizontalmente), falta espaço e, como a largura do article está fixa em 40% de 1200px, a caixa fica cortada:



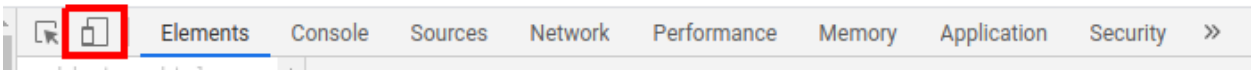
2.9 (A área visível da página: ViewPort) A expressão ViewPort se refere à area visível ou à área útil de um site. Nesta seção, veremos sua importância relacionada à responsividade. A Figura 2.9.1 destaca o ViewPort do site www.ionicframework.com.

Figura 2.9.1



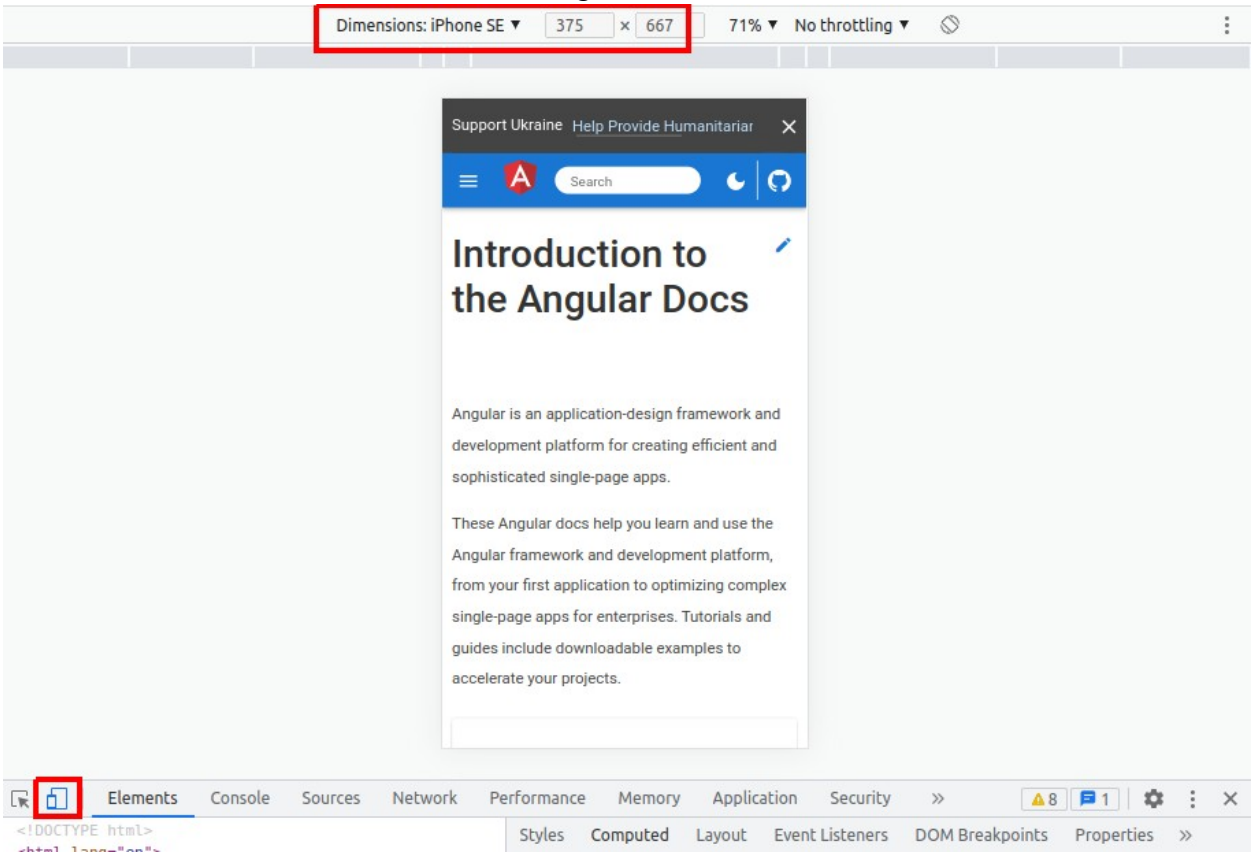
- Outro utilitário que o Dev Tools do Chrome oferece é a possibilidade de acessar um site como se estivessemos utilizando um dispositivo móvel, como a Figura 2.9.2 mostra.

Figura 2.9.2



- Note também, que uma vez escolhido esse modo, podemos escolher alguns dispositivos conhecidos, ou até fixar uma resolução qualquer. Veja a Figura 2.9.3.

Figura 2.9.3



Mantenha o **Iphone SE** selecionado (se estiver disponível na versão do Chrome que você estiver utilizando. Caso contrário, escolha outro de medidas parecidas) e visite os seguintes sites:

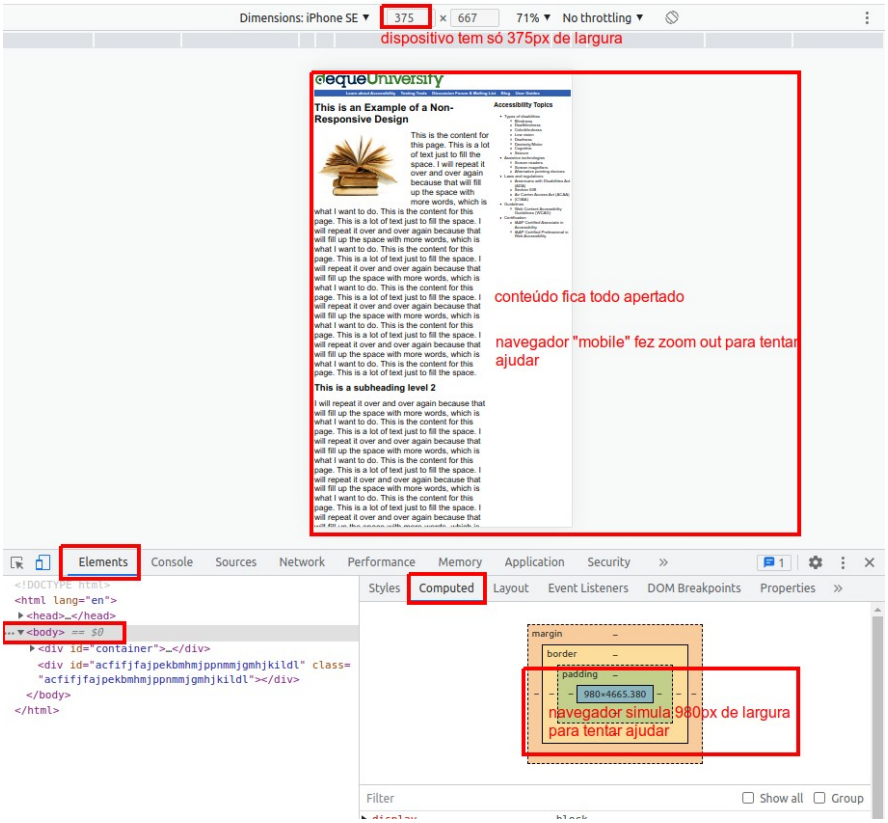
Esse é responsivo. Observe que o menu não é exibido e um botão hambúrguer aparece no lugar, por exemplo

https://angular.io/docs

Esse não é responsivo. Repare que todo o conteúdo é exibido. Para tentar ajudar, o navegador faz o seguinte:

- zoom out, simulando uma visualização “à distância”
- ainda que o dispositivo tenha apenas 375 pixels de largura, o navegador simula 980 pixels.

Observe:

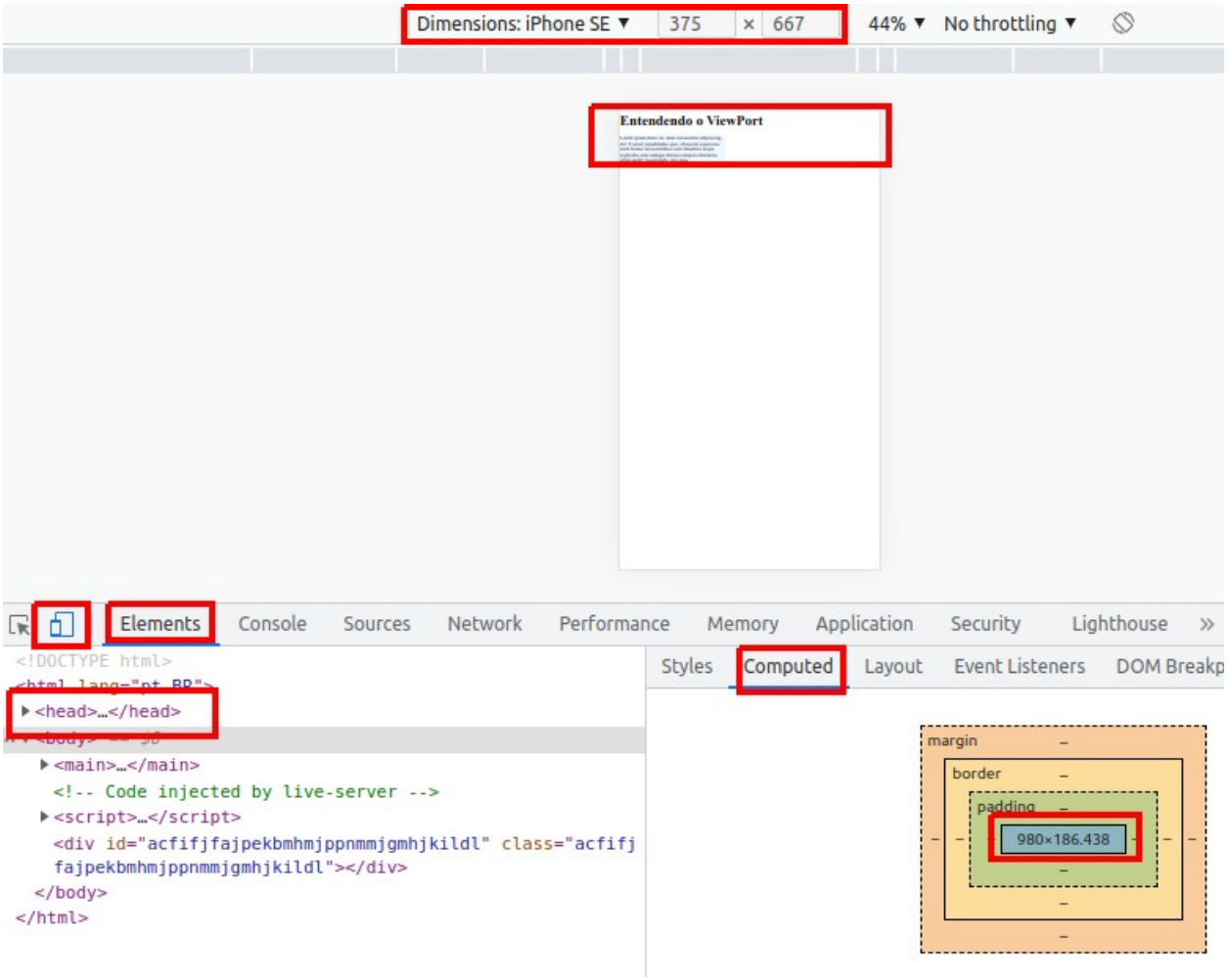


Nota. O valor de 980px pixels simulado tem razões históricas. No tempo em que grande parte dos dispositivos possuiam 1024px de largura, considerando que poderia ser de interesse reservar algum espaço para a margem, descolando o conteúdo das bordas do navegador, essa costumava ser uma boa medida.

Façamos um teste. Crie um arquivo chamado **exemplo_viewport.html** com o seguinte conteúdo:

```
<!DOCTYPE html>

<html lang="pt-BR">

<head>
<meta charset="UTF-8">
<style>
/* não esqueça isso */
body{
margin: 0;
}
p {
background-color: aliceblue;
width: 400px;
}
</style>
<title>ViewPort</title>
</head>

<body>
<main>
<h1>Entendendo o ViewPort</h1>
<p>
Lorem ipsum dolor sit, amet consectetur adipisicing elit. Eveniet repudiandae quis, obcaecati asperiores enim beatae necessitatibus iusto blanditiis neque explicabo cum cumque dolores tempore distinctio ullam unde? Assumenda, iure quos.
</p>
</main>
</body>
</html>
```

No Chrome Dev Tools (CTRL + SHIFT + I), clique para alterar a exibição simulada e escolha o Iphone SE (ou outro parecido que estiver disponível). O navegador continua fazendo zoom out e simulando a largura de 980px:

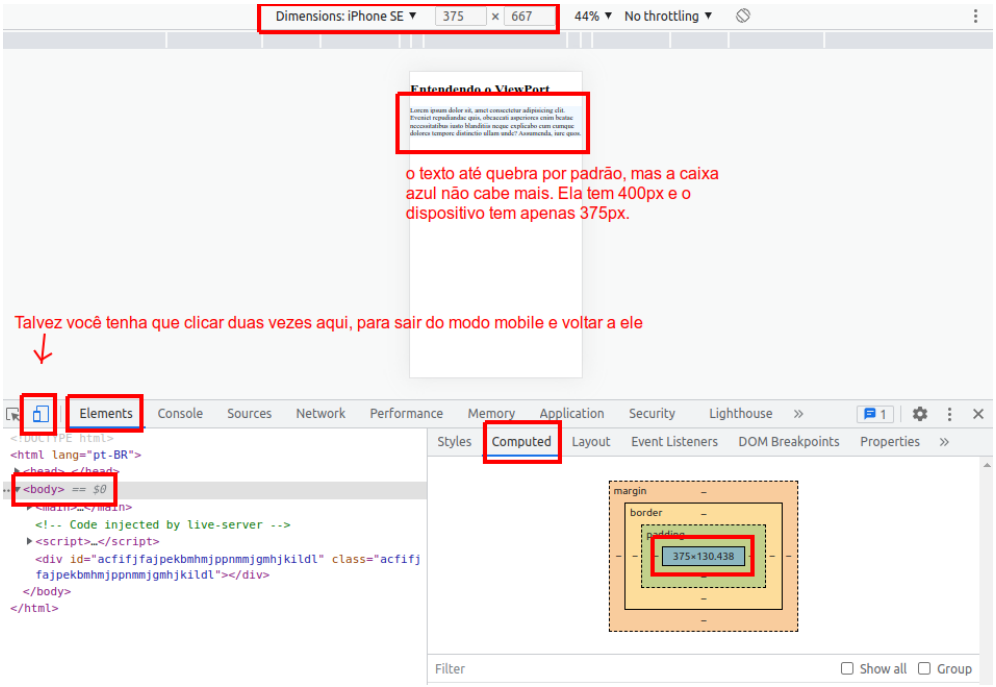


- Podemos explicar ao navegador que a nossa página está projetada para lidar com navegadores de dispositivos móveis (embora essa não esteja, estamos apenas testando) e que, portanto, ele não deve "tentar ajudar" fazendo zoom out e simulando a largura de 980px. Isso pode ser feito por meio da meta tag viewport:

```
<!DOCTYPE html>

<html lang="pt-BR">

<head>
<meta charset="UTF-8">
<!-- width=device-width significa: use a largura real do dispositivo, não simule os 980px -->
<!-- initial-scale=1 significa: não faça zoom out -->
<meta name="viewport" content="width=device-width, initial-scale=1">
<style>
/* não esqueça isso */
body{
margin: 0;
}
p {
background-color: aliceblue;
width: 400px;
}
</style>
<title>ViewPort</title>
</head>

<body>
<main>
<h1>Entendendo o ViewPort</h1>
<p>
Lorem ipsum dolor sit, amet consectetur adipisicing elit. Eveniet repudiandae quis, obcaecati asperiores enim beatae necessitatibus iusto blanditiis neque explicabo cum cumque dolores tempore distinctio ullam unde? Assumenda, iure quos.
</p>
</main>
</body>
</html>
```

Depois do ajuste, verifique o resultado com dispositivos diferentes no DevTools:

Nota. Pode ser necessário clicar no botão que alterna do modo mobile para normal para que a atualização de fato aconteça. Clique duas vezes: uma para voltar ao modo desktop e outra para voltar ao modo mobile. Sempre que remover ou adicionar a meta tag viewport, faça esse procedimento.



Resumindo, essa configuração serviu para instruirmos o navegador a não tentar fazer ajustes ao exibir a página, pois o nosso site está projetado para ser acessado por dispositivos móveis. Como a página na verdade não está preparada para a responsividade, a caixa não coube e o navegador não tentou ajudar.

2.10 (Layouts fixos e fluidos) Quando desenvolvemos um site responsivo, precisamos tomar o cuidado de ajustar a largura dos componentes conforme o tamanho da tela diminui. A unidade percentual pode nos ajudar nisso. Para esta seção, crie um arquivo chamado **exemplo_fixo_fluido.html.** Seu conteúdo inicial é dado na Listagem 2.10.1.

Listagem 2.10.1

```
<!DOCTYPE html>
<html lang="pt-BR">
<head>
<meta charset="UTF-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<title>Layout Fixo versus Layout Fluido</title>
<style>
body{
margin: 0;
font-size: 32px;
}
.engenharia, .farmacia, .matematica{
display: inline-block;
border: solid 1px black;
padding: 12px;
width: 300px;
height: 80px;
}
</style>
</head>
<body>
<main>

<article class="engenharia">
Engenharia
</article>
<article class="farmacia">
Farmácia
</article>

<article class="matematica">
Matemática
</article>

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

A tela fica assim (não centralizamos):

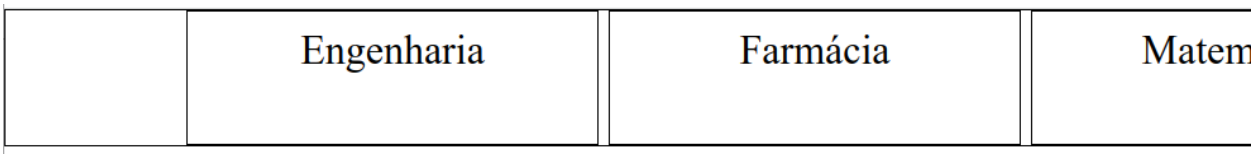
Engenharia	Farmácia	Matemática

- Começamos com um exemplo de layout **fixo**. Vamos determinar uma medida de largura para o elemento **main** e centralizá-lo. A seguir, vamos especificar a largura de cada **article**. Veja a Listagem 2.10.2.

Listagem 2.10.2

```
...
<head>
<meta charset="UTF-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<title>Layout Fixo versus Layout Fluido</title>
<style>
body {
margin: 0;
font-size: 32px;
}

main {
border: solid 1px black;
width: 1280px;
margin: 0 auto;
text-align: center;
}

.engenharia, .farmacia, .matematica {
display: inline-block;
border: solid 1px black;
padding: 12px;
width: 300px;
height: 80px;
}
</style>
</head>


...
```

Veja o que ocorre quando a largura do navegador é reduzida. Dado que a medida do container principal está fixa em pixels, em algum momento parte do conteúdo deixa de ser visível:

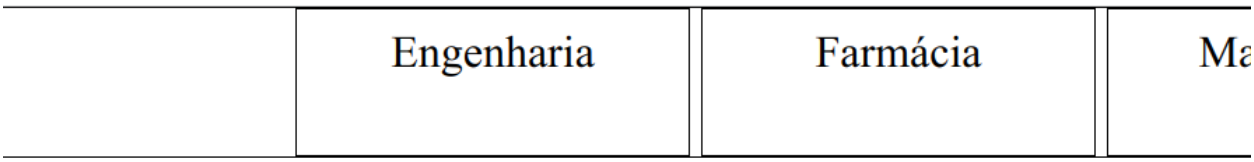


- Podemos tentar ajustar a largura dos **articles**, já que eles também estão definidos com pixels. Altere para, por exemplo, 20%, como na Listagem 2.10.3.

Listagem 2.10.3

```
...
.engenharia, .farmacia, .matematica{
display: inline-block;
border: solid 1px black;
padding: 12px;
width: 20%;
height: 50px;
}
...
```

Note que nada mudou. É só reduzir a largura da janela do navegador o suficiente para que o corte aconteça:



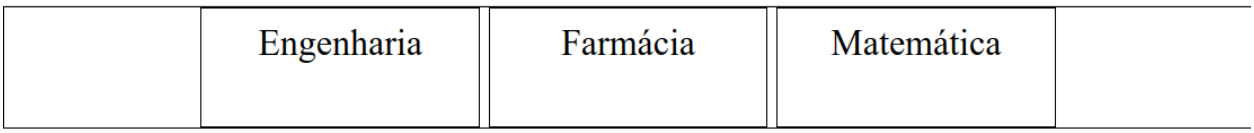
Isso ocorre pois o percentual dos elementos **article** se refere à medida de seu pai, que é o elemento **main**. Como **main** tem largura fixa em 1280 pixels, a largura de cada **article** está dada e é fixa: 1280 * 0.2 = 256 pixels.

- Vamos, portanto, definir a largura do componente **main** usando a medida percentual também. Para simplificar, digamos que **main** toma 100% de largura da tela. Conforme a largura da tela diminui, o valor computado de **main** diminui e, por consequência, o valor de todos os seus filhos também. Veja a Listagem 2.10.4.

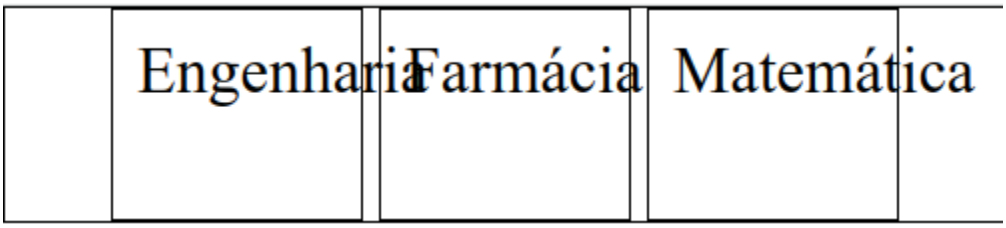
Listagem 2.10.4

```
...
main{
border: solid 1px black;
width: 100%;
margin: 0 auto;
text-align: center;
}
...
```

Esse é um exemplo de layout fluido. Conforme a tela diminui, a largura das caixas também diminui, mantendo sempre 20% dela:



Note, contudo, que quando a tela fica muito pequena horizontalmente, a apresentação se torna inadequada:



Precisamos de mais recursos.

2.11 (Media Queries) Esse recurso permite especificarmos valores para atributos que devem ser usados de acordo com a medida da tela sendo utilizada pelo usuário. Para esta seção, usaremos o mesmo código da seção anterior.

- Para começar, iremos especificar o que chamamos de **breakpoint**. Ele nos permite indicar, por exemplo, uma quantidade de pixels mínima necessária para que determinados atributos sejam aplicados.

Antes de mais nada, vamos fazer com que os cursos fiquem empilhados.

Veja o ajuste na Listagem 2.11.1.

Listagem 2.11.1

```
...
.engenharia, .farmacia, .matematica {
/* display: inline-block; */
border: solid 1px black;
padding: 12px;
/* width: 20%; */
height: 50px;
}
...
```

A seguir, vamos especificar nosso primeiro breakpoint, indicando que a exibição deve mudar a partir de uma medida mínima em pixels. Veja a Listagem 2.11.2.

Listagem 2.11.2

```
...
.engenharia, .farmacia, .matematica {
/* display: inline-block; */
border: solid 1px black;
padding: 12px;
/* width: 20%; */
height: 50px;
}

@media(min-width: 900px) {
.engenharia, .farmacia, .matematica {
display: inline-block;
width: 20%;
}
}
</style>
...
```

Faça testes alterando a largura do seu navegador. Janela com menos de 900px:

Engenharia
Farmácia
Matemática

Janela com pelo menos 900px:

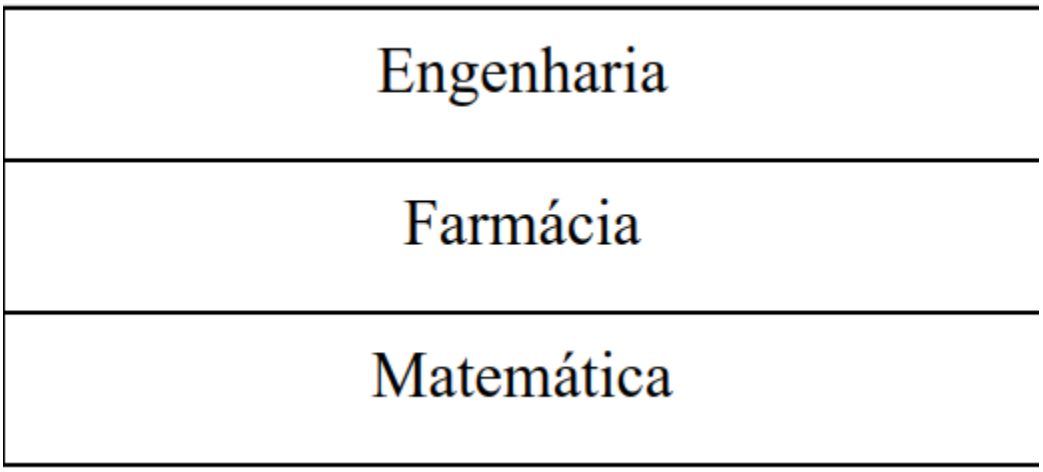
	Engenharia	Farmácia	Matemática	

- Veja que quaisquer propriedades podem ser alteradas em uma Media Query. Na Listagem 2.11.3, fazemos ajustes para alterar a cor de fundo. Se a tela tiver pelo menos 550px, a cor de fundo das caixas é alterada.

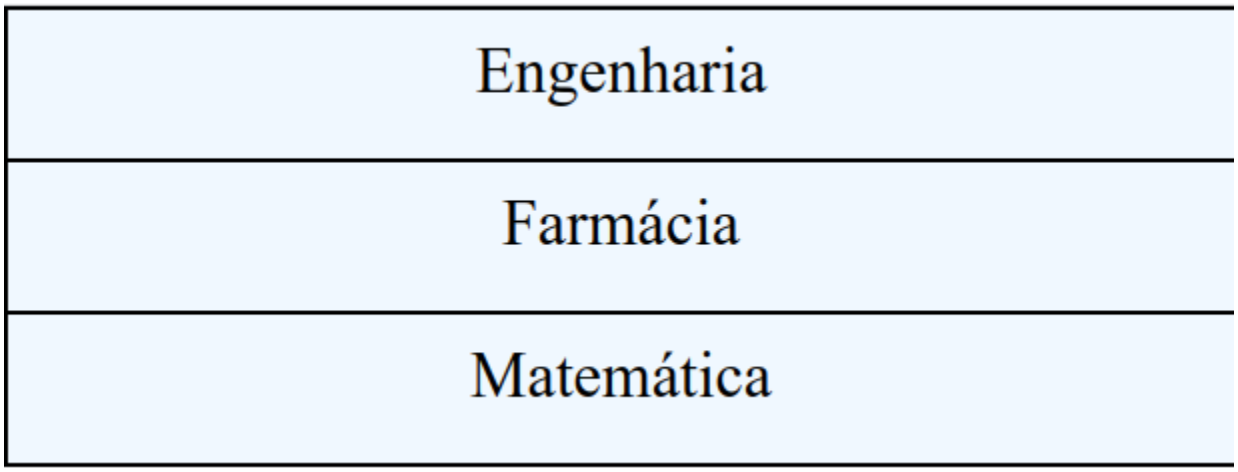
Listagem 2.11.3

```
...
@media(min-width: 900px) {
.engenharia, .farmacia, .matematica {
display: inline-block;
width: 20%;
}
}

@media (min-width: 550px){
.engenharia, .farmacia, .matematica{
background-color: aliceblue;
}
}
</style>
...
```

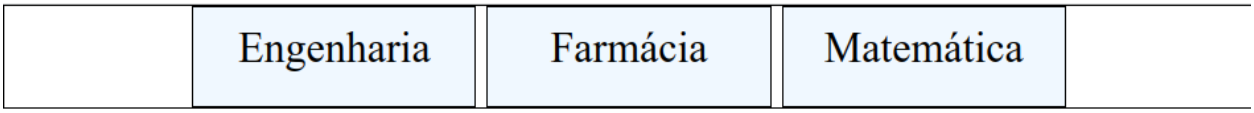
Tela com menos de 550px:



Tela com pelo menos 550px e menos do que 900px. A cor de fundo mudou:



Tela com pelo menos 900px. O display mudou, colocando as caixas lado a lado:



Observe, também, que a cor permanece. Ou seja, **as regras especificadas em uma media query não anulam regras especificadas em outra media query**. Claro, podemos ter regras conflitantes e aí elas são resolvidas pelo efeito em cascata e pela especificidade de cada regra. Repare, entretanto, que o uso de media queries pode ser um tanto trabalhoso e repetitivo, especialmente se formos nos preocupar com diferentes tamanhos de tela. E olha que temos muitos hoje em dia.

Referências

Web Hypertext Application Technology Working Group (WHATWG). 2020. Disponível em <https://whatwg.org/>. Acesso em março de 2020.

Web Hypertext Application Technology Working Group (WHATWG). 2020. Disponível em <https://developer.mozilla.org/en-US/>. Acesso em março de 2020.