4 - Manipulando flex-itens

Objetivos da Aula

- Entender o que são os flex-items na prática
- Entender as funcionalidades dos flex items
- Entender as principais manipulações dos flex-items de forma independente.

Introdução

Fala programador, tudo bem com você? Nessa aula nós vamos fazer um exercício de fixação, fazendo um header básico. Eu desafio você a fazer primeiro com tudo que aprendeu até agora, e depois faremos juntos.

Modificando o html

1. Vamos colocar uma classe extra em todos os itens, pois nós iremos em alguns momentos manipular apenas 1 item.

```
<body>
  <div class="container">
    <div class="item item1">
      <p>Item 1</p>
    </div>
    <div class="item item2">
      <p>Item 2</p>
    </div>
    <div class="item item3">
      <p>Item 3</p>
    </div>
  </div>
</body>
```

Entendendo o flex-basis

1. O flex basis é o tamanho do item. Nós vamos tirar o width e colocar o flex-basis. A diferença entre eles é mínima, nós vamos utilizar o basis por ser algo mais

“correto”, acaba sendo uma boa prática utilizar o elemento do flex, já que estamos usando o flex.

```
.item {
  height: 200px;
  border: 2px solid black;
  background-color: white;
  font-size: 20px;
  display: flex;
  justify-content: center;
  align-items: center;
  flex-basis: 200px;
}
```

2. Podemos ver que ficou igual, normalmente o nosso flex-basis é usado junto as outras duas funcionalidades que vamos ver logo logo. Podemos também usar % no basis, mas vamos deixar como esta para prosseguir.

Entendendo o flex-grow

1. O flex-grow é uma forma que temos de aumentar o tamanho dos itens sem fazer com que eles quebrem de forma imediata, nós vamos colocar através de números, colocaremos 1.

```
.item {
  height: 200px;
  border: 2px solid black;
  background-color: white;
  font-size: 20px;
  display: flex;
  justify-content: center;
  align-items: center;
  flex-basis: 200px;
  flex-grow: 1;
}
```

2. Podemos ver que todos os itens ficaram com o mesmo tamanho, preenchendo todo o espaço disponível que eles tem. A junção dele e do flex-basis é importante para ativar o wrap. Nós colocando aqui em inspecionar e diminuindo a tela, vemos que quando chegar o item em 200px, eles vão quebrar e vai descer, sem o basis, isso não acontece.

3. Nós também podemos fazer o aumento de um item de forma individual usando o flex-grow.

```
.item1{
  flex-grow: 2;
}
```

4. Conseguimos ver que o item 1 agora está muito maior que os outros, fizemos com, que de forma individual, ele crescece. Nós vamos deixar assim para prosseguir.

Entendendo o flex-shrink

1. O flex-shrink está ligado agora a diminuição dos nosso itens. Nós vamos usar no item 1 esse shrink, o valor padrão dele é 1, vamos colocar 2, isso vai fazer com que ele diminuia primeiro em relação aos outros 2 quando chegar no limite do basis. Vamos desativar o “wrap” para conseguir ver isso acontecendo de fato.

```
.container {
  background-color: darkgray;
  min-height: 100vh;
  display: flex;
  gap: 15px;
  justify-content: center;
  align-items: center;
}
.item1 {
  flex-shrink: 2;
}
```

2. Podemos ver que quando diminuímos, em um certo momento, o item 1 começa a diminuir mais do que os outros e em “364px” ele fica bem menor que os outros.
3. Agora, acontece o inverso caso a gente coloque 0 nesse nosso shrink. Isso vai fazer que ele não diminua quando chegar em 200px, que é o nosso basis.

```
.item1 {
  flex-shrink: 0;
}
```

4. Antes de prosseguir, vamos colocar a direção como coluna, colocaremos também o width dos itens como 100%, apagaremos a altura dos itens e por último, iremos apagar os itens que aprendemos agora. (basis, grow e shrink)

```
.container {
  background-color: darkgray;
  min-height: 100vh;
  display: flex;
  gap: 15px;
  justify-content: center;
  align-items: center;
  flex-direction: column;
}
.item {
  width: 100%;
  border: 2px solid black;
  background-color: white;
  font-size: 20px;
  display: flex;
  justify-content: center;
  align-items: center;
}
```

5. Nós precisamos usar o width aqui pois nós vamos user a direção como coluna, e o basis não funciona como queremos usando como coluna.

Entendendo o flex

1. Nós agora iremos ver sobre o flex. O flex puro é quase que um atalho para o basis, grow e shrink, nós podemos usar os 3 juntos aqui.
2. Nós vamos colocar aqui no item 2 esse flex, para testar e também entender a ordem dele. A ordem que nós temos nesse flex é: Grow, shrink, basis.

```
.item2 {
  flex: 1 0 200px;
}
```

3. Podemos ver que o nosso item 2 ficou com 200px de forma de altura, pois é como ele fica por estar em coluna. Nós conseguimos usar uma técnica muito interessante com isso, nós conseguimos fazer com que o item 3 fique sempre colado no final da

página. Imaginando uma página real, o nosso item 1 seria o header, o item 2 é o conteúdo principal e o 3 é o footer, o ideal é o footer colado sempre no final da nossa página.

```
.item2 {
  flex: 1 0 auto;
}
```

4. Com isso, nós conseguimos ver que o item 3 está colado no final da página. Isso faz com que a gente tenha um footer sempre colado, não importando quando conteúdo nós temos no item 2. No caso de uma página real, esse item 2 ele acaba sendo invisível, não será um bloco que nem temos agora.

Entendendo order

1. Agora nós vamos entender algo simples, que é a ordem dos nossos itens. Nós temos a alteração da ordem dos itens vindos do reverse, seja ele column-reverse ou row-reverse. Nós temos que colocar esse order individualmente.

```
.item1 {
  order: 2;
}
.item2 {
  flex: 1 0 auto;
  order: 3;
}
.item3 {
  order: 1;
}
```

2. Com isso, podemos ver que o item 2 ficou por último, o 3º em primeiro e o 1º em 2º, coisa que não da para se fazer apenas com o reverse.

Entendendo o align-self

1. Para finalizarmos o nosso estudo dos flex-items, nós iremos ver sobre o align-self. Nós iremos então remover o “flex” do item 2 e também deixar a posição dos itens como normal, não mais como coluna.

```
.container {
  background-color: darkgray;
  min-height: 100vh;
  display: flex;
  gap: 15px;
  justify-content: center;
  align-items: center;
}
.item1 {
  order: 2;
}
.item2 {
  order: 3;
}
.item3 {
  order: 1;
}
```

2. Agora podemos fazer o align-self, como o nome mesmo diz, ele vai fazer o alinhamento único de um dos itens. Nós vamos colocar o item 1 para ter esse alinhamento.

```
.item1 {
  order: 2;
  align-self: end;
}
```

3. Podemos ver que item 2 agora foi para o final da página. Esse alinhamento é igual ao “align-items”. Não temos esse alinhamento de 1 item para horizontal, apenas para o vertical.

Código final

Aqui vai ficar o código final que temos na aula, para você conferir se bate com o seu.

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

<head>
  <meta charset="UTF-8">
  <meta http-equiv="X-UA-Compatible" content="IE=edge">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <link rel="stylesheet" href="./styles.css">
```

```
  <title>Flexbox layout</title>
</head>

<body>
  <div class="container">
    <div class="item item1">
      <p>Item 1</p>
    </div>
    <div class="item item2">
      <p>Item 2</p>
    </div>
    <div class="item item3">
      <p>Item 3</p>
    </div>
  </div>
</body>

</html>
```

```
* {
  margin: 0;
  padding: 0;
  box-sizing: border-box;
}
.container {
  background-color: darkgray;
  min-height: 100vh;
  display: flex;
  gap: 0px 15px;
  justify-content: center;
  align-items: center;
  flex-wrap: wrap;
}
.item {
  width: 200px;
  height: 200px;
  border: 2px solid black;
  background-color: white;
  font-size: 20px;
  display: flex;
  justify-content: center;
  align-items: center;
}
```

Conclusão

Finalizamos os tópicos realcionados aos itens do flexbox, com isso, você já vai estar 100% apto a realizar o exercício que iremos fazer de conclusão de módulo, vamos lá?