@tiacademybrasil

HTML/CSS III

Prof. Marcelo

Sumário

1. FLEXBOX

Um dos grandes avanços no desenvolvimento web foi a inserção da propriedade *Flexible Box Module* (FLEXBOX). Esta propriedade permite atribuir alinhamentos nos elementos HTML.

Para que o **flexbox** seja "entendido" pelo navegador, basta você referenciá-lo no atributo *display* da estrutura CSS: *display: flex*.

Por padrão quando o FLEX é atribuído em um elemento "*pai*" os elementos "*filhos*" serão alinhados em "**row**" em uma única linha.

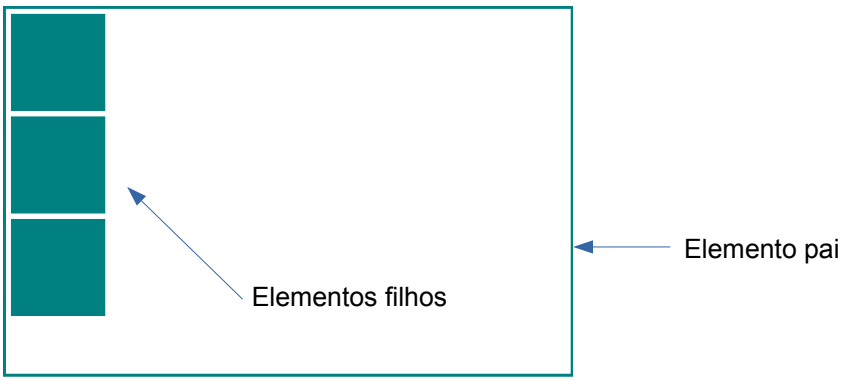

Figura 1: Elemento PAI em volvendo os elementos FILHO

Quando um box model é dimensionado com proporções de largura e altura seu "comportamento" será de acordo com seu **display**. Por exemplo, os elementos DIVs por padrão são alocados em blocos. Quando a propriedade FLEX é definida o comportamento dos elementos filhos irão alterar para "inline", mas propriamente dito em "row" quando no uso do **display:flex**. Veja o exemplo abaixo:

Figura 2: O display flex altera o comportamento dos elementos filhos em relação ao pai

Quando se atribui a propriedade flex-direction outros tipos de alinhamentos podem ser definidos.

- row
- row-reverse
- column
- column-reverse

Quando a propriedade flex é atribuída a um contêiner, este é denominado de FLEX CONTÊINER.

> Antes do recurso FLEX disponível no CSS3 o condicionamento e disposição dos elementos eram definidos por propriedades como o **FLOAT** e o **POSITION**, porém estes recursos são extremamente limitados.

Você poderá ver as propriedades e seus comportamentos do **FLEXBOX** no developer.mozilla.org

2. PROPRIEDADE BOX-SIZING

Alguns *box models* possuem estilos próprios que são renderizados pelo navegador automaticamente, como é o caso dos elementos:

Tags [elementos]	Referência HTML	Estilo Padrão
h1 até h6	Tagas de títulos	Padding e Margin
ul ("Unordered List Element) / li ("List item")	Tag para listas ordenadas	Padding e Margin
p	Parágrafos de textos	Margin-top e Margin-bottom
Body	Abarca todo o conteúdo de renderização	Margin

Estes estilos podem prejudicar a disposição de outros elementos quando ordenados em um mesmo contêiner. Dessa forma a renderização dos elementos não irá corresponder com o resultado esperado.

É o navegador que acrescenta estes estilos e isto também irá ser diferente de navegador para navegador.

Uma prática usada para que os elementos "*RESETEM*" ou "zerem"suas propriedades de estilo é fazer uma lista de *normalize* ou *reset* (redefinir).

No CSS uma das formas de definir um reset seria listar os atributos separando por "," vírgula para redefinir as propriedades, em geral padding e margin:

```
body, h1, h2, ul, li, p { margin:0; padding:0; }
```

Uma outra forma para "resetar" seus elementos é usando o *coringa* "*" asterisco, este irá atribuir as mesmas propriedades para todos os elementos:

```
* { margin:0; padding:0; }
```

O mesmo conceito também serve para que eu possa definir alguma outra propriedade para diversos elementos:

```
p, div {display: flex;}
```

> A diferença básica entre **MARGIN** e **PADDING** é que o atributo margin irá adicionar uma margem por fora do elemento. Já o padding irá acrescentar um espaço ou preenchimento dentro do elemento.

2.1 Mas o que é o Box Sizing

Como visto anteriormente alguns elemento HTML possuem especificações de estilo que os navegadores renderizam quando estão presentes em um código html.

No entanto, na formatação de estilos é comum observarmos que mesmo "resetando" estas propriedades, quando um box model é "encaixado" ou adicionado a outro box model seu comportamento de dimensão pode sofrer alterações.

Estas alterações ocorrem nas propriedades width (altura) e height (largura) principalmente quando há a estilização de **bordas**, **padding** e **margin** no elemento filho.

Por padrão os navegadores renderizam a propriedade BOX-SIZING como **CONTENT-BOX**, na prática isso significa que quando você atribui um width de 200px, um boder de 2px o resultado total de proporção do seu elemento será de aproximadamente 212pixels (a conta não fecha, mas é mais ou menos isso mesmo – coisa de *navegadores*).

Para que a renderização seja fiel ao que está sendo especificado no CSS a propriedade **BOX-SIZING** é "setada" para **BORDER-BOX**.

```
#elemento { box-sizing: border-box; }
```

3. UNIDADES DE MEDIDAS

As unidades de medidas mais comuns para dimensionar elementos html são o *pixel* (px) e o *percentual* (%).

No entanto existem outras unidades de medidas como cm, pt, em, rem, vw e o hw.

Aqui será dado uma atenção para as unidades **vw** e **vh**, e neste link você pode acessar o site do w3c.org e saber mais sobre as unidades de medidas.

3.1 O viewport

O *viewport* representa a área de visualização que está disposto um layout. Esta área irá depender do dispositivo que o usuária acessa o conteúdo.

3.2. O vw e o vh

O vw (*viewport width*) e vh (*viewport height*) são medidas que estão condicionadas a largura e a altura do viewport.

Por exemplo se um *viewport* possui 800px de largura isso significa que 1vw será igual a 8px. Consequentemente se a disposição de altura (height) de um viewpor é de 600px 1vh será de 6px.

4. FORMULÁRIOS

Formulários são utilizado basicamente para entrada de dados do usuário, como cadastro, atualização, remoção e atualização de informações. O elemento <form> é utilizado para definir os elementos HTML responsáveis pela entrada de dados, e deve ser definido como <form action="" method="método">.

Os elementos HTML que possibilitam criar campos para entrada do usuário são INPUT, SELECT e TEXTAREA.

Alguns elementos TYPE:

• <input type="text" name="nome"> : insere uma caixa de texto para entrada de dados;

• <input type="password" name="nome"> : insere uma caixa de texto para entrada de dados do tipo senha, inserindo círculos preenchidos ao invés dos valores que o usuário digita;

• <input type=”checkbox” name=”nome” value=”valor”>: insere um botão para seleção ou não do usuário. Normalmente, são utilizados vários desses juntos, dando ao usuário a opção de selecionar um ou várias opções;

• <input type=”radio” name=”nome” value=”valor”> : insere um botão para seleção ou não do usuário. Normalmente, são utilizados vários desses juntos, dando ao usuário a opção de selecionar um ou várias opções. A diferença aqui é que os botões são mutualmente exclusivos.

Os outros “radio” da mesma categoria devem ter o mesmo nome para que a exclusão possa funcionar corretamente.

• <input type=”submit” value=”valor”>: apresenta um botão que gera o envio dos dados para o servidor.

• <input type=”reset” value=”valor”>: apresenta um botão que anula os valores entrados pelo usuário.

O elemento **SELECT** apresenta uma lista de dados por meio do campo **OPTION**. Exemplo:

<**select** name=”nome”>

<option> Dado 1

<option> Dado 2

<option> Dado 3

</**select**>

O elemento **TEXTAREA** fornece uma área com mais uma linha para entrada de texto. Exemplo: <textarea cols=”20” rows=”5” name=”observacao”> texto ... </textarea>

No Developer Mozilla você encontrará outras especificações, uso e explicações sobre formulários.

4. 1 Atribuição CSS para um elemento TYPE

Além de aplicar um seletor CSS em um elemento HTML por sua respectiva tag, ou mesmo por um id ou class, os elementos de formulários podem ser manipulados por seu tipo (type). Exemplo de estilização de um ***input type text***.

```
input[type=text] { border: 1px solid #000 }
```

5. RESPONSIVIDADE

Quando se diz que um site ou aplicação se adapta ao viewport de um dispositivo dize-se que esta aplicação ou site é **responsiva**.

Existem diferentes tipos de dispositivos logo haverá áreas distintas para a apresentação de conteúdo.

Para que um site ou aplicação seja responsiva o desenvolvedor deve se preocupar em "adaptar" seu layout para os diferentes tipos de dispositivos.

Esta adaptação pode ocorrer com a aplicação de um **media query**. Um media query é definido em uma área do css. Neste media query será especificado em ponto (**break point**) deverá haver uma alteração no layout para se adaptar ao viewport.

Exemplo de uma instrução css para um media query:

```
@media screen and (min-width: 700px) { ... }
```

A instrução acima quer dizer que, quando o viewport atingir um tamanho menor que 700px, o navegador deverá executar a nova renderização CSS entre os { }.

```
@media screen and (min-width: 700px) {

    .elemento{
        display:none; /*ocultar um elemento*/
```

```
}
```

REFERÊNCIAS

[Flexbox]

https://developer.mozilla.org/pt-BR/docs/Web/CSS/CSS_Flexible_Box_Layout/Basic_Concepts_of_Flexbox

[Box-sizing**]**

https://developer.mozilla.org/pt-BR/docs/Web/CSS/box-sizing

[Unidades de Medidas]

https://desenvolvimentoparaweb.com/css/unidades-css-rem-vh-vw-vmin-vmax-ex-ch/

https://www.w3.org/Style/Examples/007/units.pt_BR.html

[Viewport]

https://developer.mozilla.org/pt-BR/docs/Glossary/Viewport