SUMÁRIO

1. INTRODUÇÃO AO CSS 1

1.1. O que é CSS? 1

1.2. Formas de aplicação do CSS 1

2. TAGS DIV E SPAN 1

3. POSICIONAMENTO FLOAT 2

4. POSICIONAMENTO INLINE, BLOCK E INLINE-BLOCK 3

5. POSICIONAMENTO FLEX 5

6. Z INDEX 5

7. POSICIONAMENTO COM POSITION 6

8. UNIDADES DE MEDIDA EM CSS 7

9. FONTES DE PESQUISA COMPLEMENTARES 8

1. INTRODUÇÃO AO CSS

1.1. O que é CSS?

Cascading Style Sheets é um mecanismo para adicionar estilo a um documento web. O código CSS pode ser aplicado diretamente nas tags ou ficar contido dentro das tags <style>. Também é possível, em vez de colocar a formatação dentro do documento, criar um link para um arquivo CSS que contém os estilos.

1.2. Formas de aplicação do CSS

Inline Style: A estilização das tags HTML, e do documento em si, pode ser realizada utilizando o estilo como um atributo "<style>" de uma tag. Desta forma, se, por exemplo, o objetivo fosse estilizar um parágrafo, o atributo "style" seria implementado na própria tag . Teríamos, então: ** .**

Internal Style Sheet: A estilização do documento HTML pode ser implementada, também, dentro da tag "<style> </style>", na tag <head>. Desta forma, ao invés de ser pontual como a anterior, esta permite uma implementação em "cascata", utilizando seletores e chaves, de uma forma em que se pode estilizar vários elementos de uma só vez, de forma mais eficiente e eficaz. Se, por exemplo, o objetivo fosse o de estilizar todos os "input" do documento, faríamos:

<style>

input{

configurações de estilo

}

</style>

Desta mesma forma, todos os elementos do documento podem ser estilizados, desde o próprio <body> até um seletor artificial, como uma **classe** ou um **id**.

External Style Sheet: A estilização com seletores e chaves demonstrada pode, e deve, como boa prática, ser implementada em um arquivo.css externo, linkado ao documento HTML correspondente. Dentro da tag <head>, escreve-se, para isso, **<link rel="stylesheet" href="caminho_do_arquivo.css ">**.

2. TAGS DIV E SPAN

Tag Div: A tag geralmente é utilizada para criar uma divisão ou uma seção em um documento HTML. Neste caso, as divisões se sucedem verticalmente na página, estando abaixo ou acima uma das outras, a depender de sua implementação no documento HTML.

Tag Span: A etiqueta geralmente é usada para agrupar elementos em linha em um documento. Neste caso, os elementos se sucedem no sentido horizontal.

3. POSICIONAMENTO FLOAT

A propriedade **float** do CSS determina que um elemento deve ser retirado do seu fluxo normal e colocado ao longo do lado direito ou esquerdo do seu container, onde textos e elementos em linha irão se posicionar ao seu redor. Quando aplicado o float, os elementos são inseridos em um **novo contexto**, ou "plano", que não o plano default, podendo, por isso, inclusive, se localizarem abaixo de outros elementos.

Por conseguinte, **se dois elementos são posicionados com float left**, por exemplo, ao invés de um sobrepor o outro, ambos ficarão lado a lado, porque estarão, os dois, no plano "float", e não um plano acima ou abaixo do outro; elementos não podem ocupar um mesmo espaço, estando em um mesmo contexto.

Em alguns casos, o fato de os elementos flutuantes saírem do contexto do browser, faz com que alguns eventos inesperados ocorram, como a quebra de limites, em que uma div child flutuante, ultrapassa os limites da sua div parente não flutuante. Neste caso, aplicamos a funcionalidade "**overflow: hidden"** ao elemento parent, que faz com que este saiba que possui elementos child flutuantes, e faça o recalculo de suas dimensões considerando este fato.

Além disto, esta funcionalidade, **"overflow: hidden"**, permite esconder os elementos child, se a div parente for reduzida. Ou seja, se a div parent, de 100px, possuir uma caixa com height também de 100px, e for reduzida para 80px, a caixa child continuará com 100px, mas seus 20px restantes serão escondidos, porque a div parente possuirá, agora, 80px de height.

Outro aspecto que deve ser citado, é que um elemento flutuante não pode flutuar sobre um conteúdo; como um texto, por exemplo, de uma tag de parágrafo. Desta forma, o que ocorre, é que, dependendo da implementação no HTML, o conteúdo escrito poderá fluir entre os elementos flutuantes, como um fluído. Para resolver isto, usa-se a funcionalidade "clear", que "limpa" o contexto float ao redor do conteúdo, forçando-o a se posicionar ao lado, ou abaixo dos flutuantes. A propriedade aceita 4 valores:

- **left**: Elemento é empurrado para baixo de elementos com *float left*;
- **right**: Elemento é empurrado para baixo de elementos com *float right*;
- **both**: Elemento é empurrado para baixo de elementos com *float left ou right*;
- **none**: Elemento não é empurrado para baixo de elementos com *float*.

4. POSICIONAMENTO INLINE, BLOCK E INLINE-BLOCK

Todos os elementos HTML podem ser separados em dois grupos, em termos de comportamento visual CSS: elementos bloco (*block*) e elementos em linha (*inline*). Ser bloco ou em linha muda o comportamento visual do elemento, além de outras características.

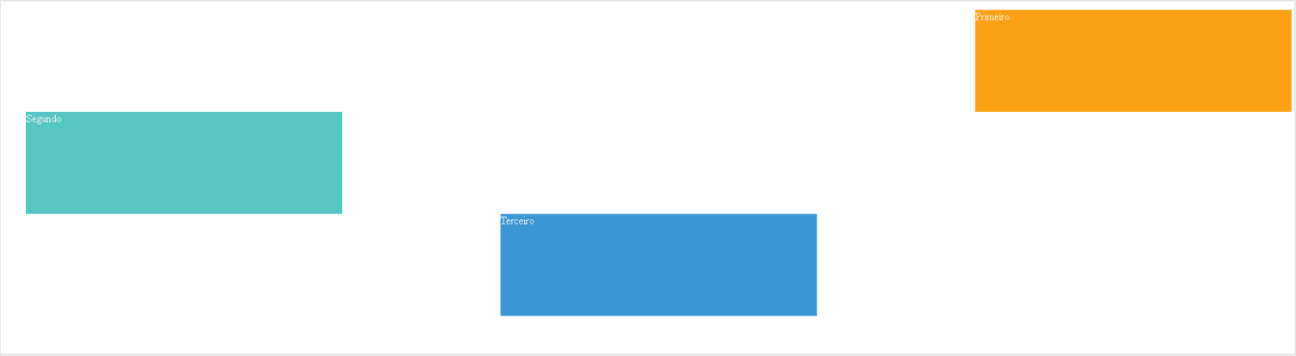
Elementos bloco ocupam todo o espaço horizontal disponível e iniciam uma nova linha no documento. Novos elementos irão começar na próxima linha livre. Exemplos são a , <h1>, <p>, <ul>, <ol>, <form>, entre outros.

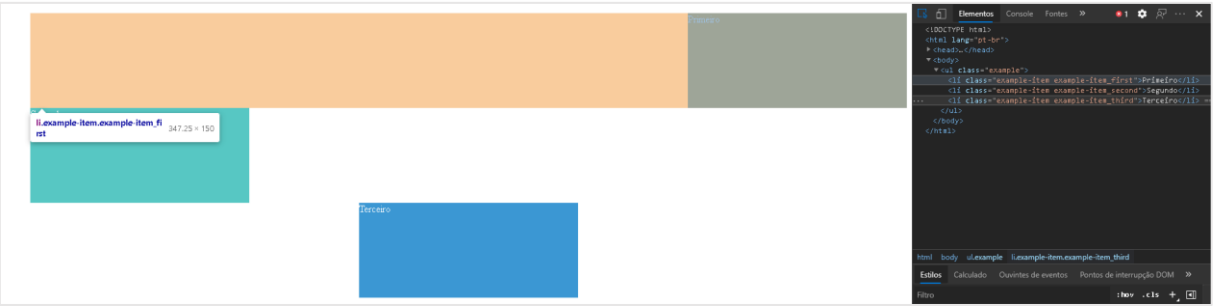
Elementos em linha ocupam apenas o espaço necessário e não iniciam uma nova linha. São chamados elementos em linha justamente por aparecer na mesma linha que outros elementos, caso seja possível. Exemplos são o , <img, <a, entre outros.

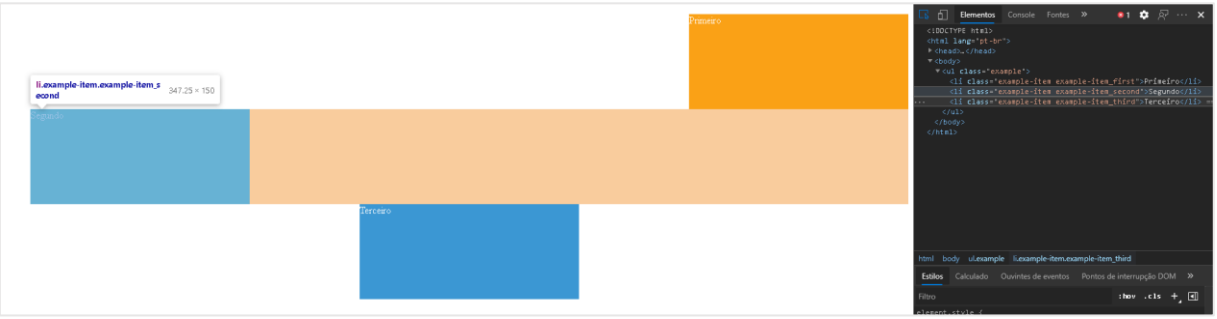
Display Inline: Esta funcionalidade faz com que os elementos passem a ter um comportamento de "palavras" inline; passam a se dispor horizontalmente, e a se suceder, igualmente, na horizontal. No entanto, não é esta a funcionalidade que deve ser usada para organizar divisões que possuem dimensões, porque quando utilizada faz com que os elementos percam seus atributos de largura e altura, passando a ter uma dimensão exatamente igual ao de seu conteúdo. Abaixo, a primeira imagem seguida de seu display inline.

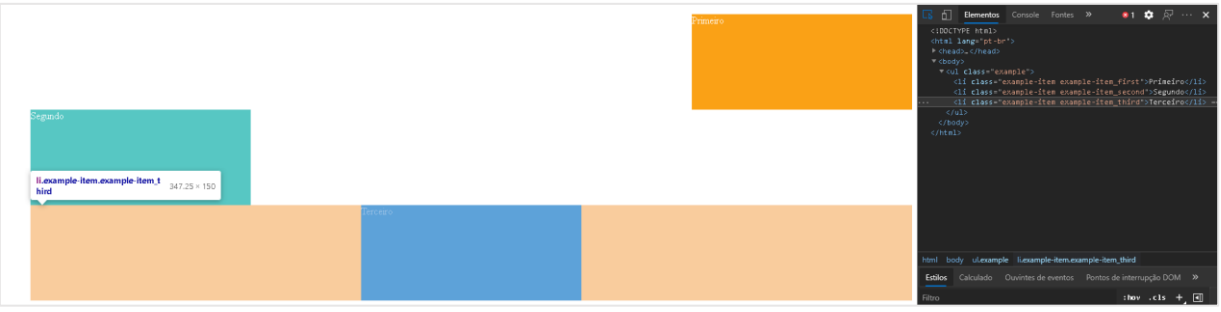
Display Block: Esta funcionalidade faz com os elementos passem a ter um comportamento de bloco, e isto inclui algumas regras; **(i)** o elemento irá ocupar a linha inteira, não admitindo um elemento ao seu lado (a não ser com gambiarras); **(ii)** irá admitir a configuração de suas dimensões, de largura e altura; **(iii)** permite que um único elemento, de uma linha, seja posicionado facilmente no sentido horizontal, por meio dos valores de margin, ou das funcionalidades margin auto.

Abaixo, ambas imagens são a mesma, onde os elementos estão com a funcionalidade display block. As três, abaixo da primeira, servem para demonstrar que apesar do elemento estar em uma posição especifica, está ocupando, na verdade, toda a linha; ele está em uma posição de toda a sua linha.









Display Inline-block: Em comparação com display: inline, a principal diferença é que display: inline-block permite definir uma largura e altura no elemento. E em comparação com display: block, a principal diferença é que display: inline-block não adiciona uma quebra de linha após o elemento, portanto, o elemento pode ficar próximo a outros elementos.

5. POSICIONAMENTO FLEX

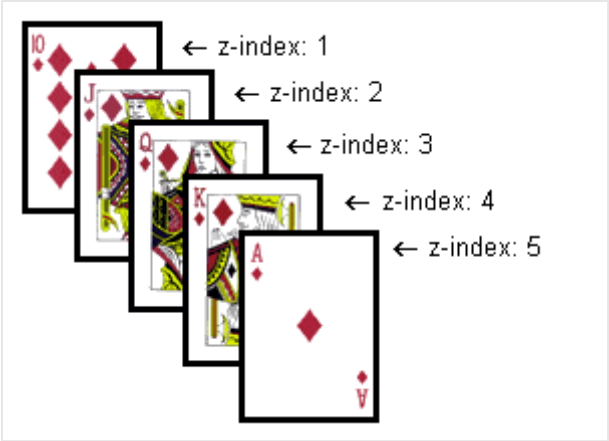
Assim como as funcionalidades vistas anteriormente, para posicionamento de elementos do documento HTML, o posicionamento flex, comumente chamado de FlexBox, seria uma das alternativas mais recentes, modernas e úteis para este mesmo fim.

Este, em especifico, apresenta configuração focadas na conversa entre elementos dentro de um elemento pai, como um container, que é uma divisão que contém mais de um elemento. Visa, assim, organizar os elementos de uma página HTML dentro de seus containers de forma dinâmica. Portanto, para utilizar esse recurso é necessário ter no HTML ao menos um elemento (container) contendo outros (itens), sendo necessariamente aplicado no elemento pai, na folha de estilo CSS.

Sua aplicação é, no geral, após assimilada, mais eficaz e eficiente que as outras vistas, mas possui alguns conceitos a mais, e por isso, foi separada uma matéria da Devmedia, especificamente sobre este tipo de recurso, e como utiliza-lo. O link está no tópico de fontes de pesquisa complementares.

6. Z INDEX

Quando pensamos em posicionamento de elementos em páginas web, a primeira coisa que vem em nossa cabeça são os eixos X e Y que são os eixos horizontais e verticais, respectivamente. A propriedade z-index trabalha com um eixo que não é muito conhecido e tampouco usado pela maioria dos desenvolvedores, o eixo Z. O eixo Z é o eixo responsável pelo cálculo e posicionamento da profundidade de algum elemento, ou seja, é aquele que irá determinar se o elemento estará mais próximo ou mais afastado da tela.



7. POSICIONAMENTO COM POSITION

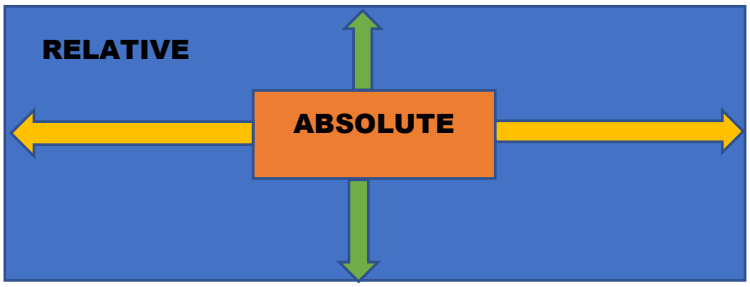
Para posicionar seus elementos, você precisa inserir uma coordenada. Essas coordenadas são comandadas pelas propriedades: top, left, right ou bottom. Todos os valores de positions só trabalham com essas coordenadas.

A propriedade position pode assumir 5 valores diferentes: Static, Relative, Absolute, Fixed e Sticky. Segue abaixo a explicação para de cada uma:

Position Static: Esse é o valor default de todo elemento HTML, ou seja, ele vai seguir o fluxo comum da sua página. Os elementos posicionados estáticos não são afetados pelas propriedades superior, inferior, esquerda e direita.

Position Relative: Com este position, o elemento passa a estar posicionado conforme implementado no HTML, tendo como referência o seu elemento pai – ou parent. Esta funcionalidade admite valores de coordenadas X e Y, ou seja, com top e bottom, podemos mover o elemento, dentro do seu parent, no eixo Y, e com left e right, podemos movê-lo no eixo X.

Position Absolute: Um elemento com position absolute é posicionado em relação ao ancestral posicionado mais próximo. Contudo, se um elemento posicionado de forma absoluta não tiver ancestrais posicionados, ele usa o corpo do documento como referencial – o body – podendo ignorar o fluxo de implementação do documento HTML.



Position Fixed: O position fixed irá fixar a posição do elemento na coordenada que você definir. À medida que a página é rolada, o elemento continua fixo na posição que você definiu e o conteúdo da página rola normalmente.

Position Sticky: Alterna entre relative e fixed, dependendo da posição de rolagem. Ele é posicionado em relação até que uma determinada posição de deslocamento seja encontrada na janela de exibição - então, ele "se fixa" no lugar (como a posição: fixa). O melhor exemplo são imagens que se mantém no mesmo local da página, mas que são desvendadas gradualmente, a partir da rolagem do mouse.

8. UNIDADES DE MEDIDA EM CSS

CSS oferece um número de unidades diferentes para a expressão de medida de tamanho. Mas, antes de prosseguirmos, precisamos entender qual a diferença entre medida absoluta e medida relativa.

Medida Absoluta: Essas são as mais comuns que vemos no dia a dia. São medidas que não estão referenciadas a qualquer outra unidade, ou seja, não dependem de um valor de referência. São unidades de medidas definidas pela física, como o píxel, centímetro e o metro.

Medidas Relativas: Essas medidas são calculadas tendo como base uma outra unidade de medida definida, como por exemplo o “em” e o “rem”.

As medidas que serão descritas, abaixo, serão as comumente mais utilizadas, apenas; o Píxel(PX), REM, EM, a porcentagem (%), o VW e o VH.

Unidade Píxel (px): O píxel é o menor elemento em um dispositivo de exibição, e é uma medida absoluta, isto é, que, em teoria, não varia de dispositivo para dispositivo, de resolução para resolução. Um ponto interessante de se comentar é que recentemente **o boostrap 4 deixou de utilizar PX e migrou para REM**., por motivos de adaptabilidade.

Unidade EM: Essa unidade muda para os elementos filhos de acordo com o tamanho da fonte (font-size) do elemento pai. Por exemplo, se o elemento pai possuir um font size de 16px, a fonte do elemento child, se definida em 1em, será, na verdade, a fonte do pai vezes 1. Assim, se o elemento child possuir um font size de 2em, este será equivalente a 32px.

Unidade REM: O REM vem como sucessor do EM e ambos compartilham a mesma lógica de funcionamento (font-size), porém a forma de implementação é diferente. Enquanto o em está diretamente relacionado ao tamanho da fonte do elemento pai, o rem está relacionado com o tamanho da fonte do elemento root (raiz), no caso, a tag. O fato de que o rem se relaciona com o elemento raiz resolve o problema de quando existem diversas divs aninhadas, que herdam sucessivamente seus referenciais de tamanho de fonte.

Medida porcentagem (%): A porcentagem costuma ser bastante utilizada quando falamos de layout responsivo e fluido. A porcentagem permite que criemos módulos que sempre vão se readaptar para ocupar a quantidade especificada. Por exemplo, se definirmos um elemento tendo um tamanho de 50%, independente do dispositivo em questão, esse módulo sempre ocupará metade do espaço que lhe cabe.

Unidade Viewport Width (VW): Essa medida faz parte das medidas mais atuais e do futuro do CSS. Viewport nada mais é que a área visível de uma página web para o seu usuário, essa viewport pode variar de acordo com o dispositivo, sendo menor em celulares e maior em desktops. Em suma, essa unidade, VW, se relaciona diretamente com a largura da viewport.

Unidade Viewport Height (VH): Essa unidade funciona da mesma forma que o vw, porém dessa vez, a referência será a altura e não a largura.

9. FONTES DE PESQUISA COMPLEMENTARES

https://developer.mozilla.org/en-US/docs/Web/CSS/display

https://www.devmedia.com.br/css3-flexbox-funcionamento-e-propriedades/29532

https://www.alura.com.br/artigos/guia-de-unidades-no-css