DISPLAY

Display é a propriedade CSS mais importante para controlar o layout, ela especifica se/como um elemento é exibido. Cada elemento HTML tem um valor de exibição padrão dependendo do tipo de elemento que é. O valor de exibição padrão para a maioria dos elementos é block ou inline.

Elementos block

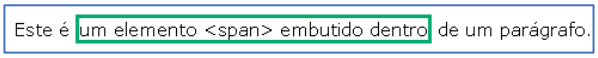
Um elemento de nível de block (bloco) sempre começa em uma nova linha e ocupa toda a largura disponível (se estende para a esquerda e para a direita o máximo possível).

Exemplos de elementos em nível de bloco:

-
- <h1> - <h6>
- <p>
- <formulário>
- <cabeçalho>
- <rodapé>
- <seção>

Elementos inline

Um elemento inline (embutido) não inicia em uma nova linha e só ocupa a largura necessária.



Exemplos de elementos embutidos:

-
- <a>
- <img>

Elementos inline-block

Um elemento display: inline-block; mantém as características de um elemento inline porém, propriedades de um display do tipo block. Ou seja, o ele

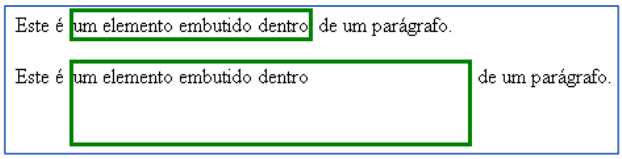
mento pode ser posicionado dentro de um bloco e como no exemplo anterior se manter dentro do texto na mesma linha, acrescentando propriedades de um elemento com display do tipo block, como por exemplo alterar valores de altura e largura do elemento, acrescentar margin e padding, entre outras propriedades que um elemento inline não permite alterar.

No próximo exemplo é demostrado a alteração do display de um elemento inline para um block-inline:

```
<p>Este é <span class="inline">um elemento embutido dentro</span> de um parágrafo.</p>

<p>Este é <span class="bloco">um elemento embutido dentro</span> de um parágrafo.</p>
```

```
.inline {
  border:solid 3px green;
  width: 200px;
  height: 50px;
}
.bloco {
  display: inline-block;
  border:solid 3px green;
  width: 300px;
  height: 60px;
}
```



É possível observar que o elemento <p>, por padrão, é definido como um elemento do tipo block, portanto, ele só aceita elementos do tipo inline dentro dele, assim como o elemento . Porém é possível alterar como o elemento é exibido (display), mostrando-o “como se fosse” um elemento block. No exemplo acima houve uma tentativa de acrescentar estilos e alterar valores de altura e largura do elemento , mas por ele se tratar de um elemento inline, essa alteração não será possível como demonstra o primeiro exemplo span. Porém como é feita a alteração do modo de exibição para block-inline no segundo elemento span, o elemento passa a ser exibido como um bloco, porém embutido dentro de outro bloco aceitando então tais alterações.

Display none

O valor de display: none; é comumente usado com JavaScript para ocultar e mostrar elementos sem excluí-los e recriá-los. O elemento <script> usa display: none; como padrão.

Alterar o Valor de Display Padrão

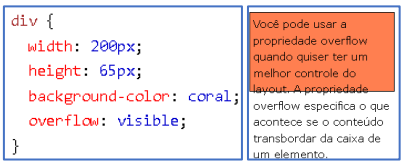
Como mencionado, cada elemento tem um valor de exibição padrão. No entanto, você pode substituir isso. Alterar um elemento inline para um elemento block, ou vice-versa, pode ser útil para fazer a página parecer de uma maneira específica e ainda seguir os padrões da web. Definir a propriedade display de um elemento altera apenas a forma como o elemento é exibido (display), não o tipo de elemento que ele é. Portanto, um elemento inline com display: block; não pode ter outros elementos de block dentro dele.

OVERFLOW

A propriedade overflow controla o que acontece com o conteúdo grande demais para caber em uma área. A propriedade especifica se o conteúdo deve ser recortado ou adicionado barras de rolagem quando o conteúdo de um elemento for muito grande para caber na área especificada. A propriedade overflow só funciona para elementos de bloco com uma altura especificada.

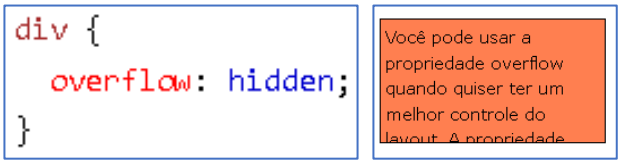
Overflow: visible;

Predefinição. O estouro não é cortado. O conteúdo é renderizado fora da caixa do elemento



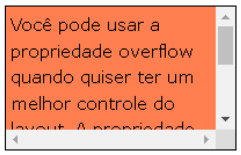
Overflow: Hidden;

O estouro é cortado e o restante do conteúdo ficará invisível



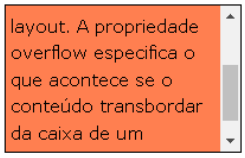
Overflow: scroll;

O estouro é cortado e uma barra de rolagem é adicionada para ver o restante do conteúdo. Esse valor adicionará uma barra de rolagem horizontal e verticalmente (mesmo que você não precise). Ainda podemos editar a barra; com overflow-x apenas a barra horizontal é exibida; com overflow-y apenas a parte vertical é exibida.



Overflow: auto;

Semelhante a scroll, mas adiciona barras de rolagem apenas quando necessário



ALINHAMENTO HORIZONTAL

elementos block

Para centralizar horizontalmente um elemento do tipo block (como), use **margin: auto**; definir a largura do elemento impedirá que ele se estenda até as bordas de seu contêiner. O elemento então ocupará a largura especificada e o espaço restante será dividido igualmente entre as duas margens:

```
.center {
  margin: auto;
  width: 60%;
  border: 3px solid #73AD21;
  padding: 10px;
}
```

Este elemento div é centralizado.

O alinhamento ao centro com uso de margin:auto não tem efeito se a propriedade width não estiver definida.

elementos inline ou inline-block

Elementos inline podem ser considerados como um “texto” dentro de um elemento block, seguindo por esse princípio, é possível então alinhar os elementos inline e inline-block com a propriedade **text-align: center;**. Segue um exemplo:

```
<div>
<img src="paris.jpg" alt="Paris" style="width:40%">
</div>
```

```
div{
    text-align:center;
}
```

No exemplo acima temos um elemento <img>, que por padrão é um elemento inline, ele foi alinhado através da criação de um bloco e alteração de estilo do mesmo com o uso da propriedade text-align, porém, como já abordado em um tópico anterior, elementos inline, inline-block, podem ser modificados para serem exibidos como elementos block e vice-versa com o uso da propriedade display. Através então da alteração da propriedade display é possível acrescentar a propriedade margin em elementos inline/inline-block para centralizá-los de outra maneira. Segue um exemplo:

```
img {
  display: block;
  margin-left: auto;
  margin-right: auto;
}
```

POSITION

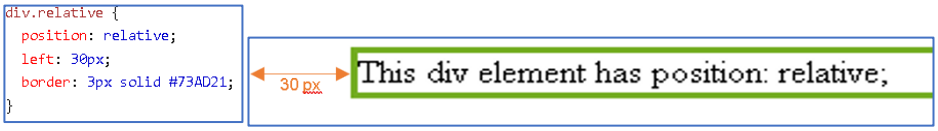
A propriedade position especifica o tipo de método de posicionamento usado para um elemento static, relative, fixed, absolute or sticky (estático, relativo, fixo, absoluto ou grudento). Os elementos são então posicionados usando as propriedades top, bottom, left e right. No entanto, essas propriedades não funcionarão a menos que a position propriedade seja definida primeiro. Eles também funcionam de forma diferente dependendo do valor da posição.

position: static;

A propriedade position define por default o valor static para todos os elementos. Elementos posicionados em static não são afetados pelas propriedades top, bottom, left e right. Um elemento com position: static;não está posicionado de forma especial; é sempre posicionado de acordo com o fluxo normal da página.

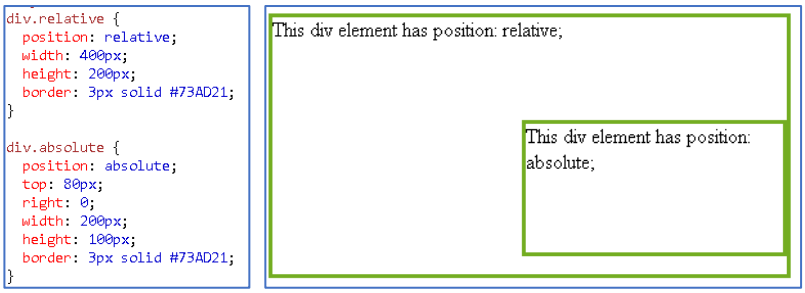
position: relative;

Um elemento com position: relative; é posicionado em relação à sua posição normal. Definir as propriedades top, bottom, left e right de um elemento relativamente posicionado fará com que ele seja ajustado para fora de sua posição normal. Outros conteúdos não serão ajustados para caber em qualquer lacuna deixada pelo elemento.

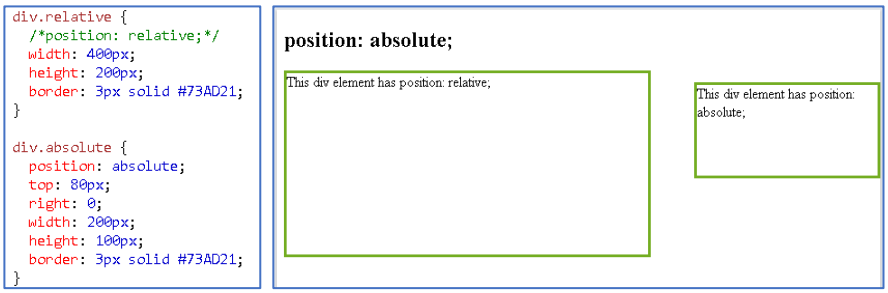


position: absolute;

Um elemento com position: absolute; é posicionado em relação ao ancestral posicionado mais próximo (em vez de posicionado em relação à janela de visualização, como fixo). No entanto; se um elemento posicionado absoluto não tiver parents posicionados, ele usará o corpo do documento e se moverá junto com a rolagem da página. Os elementos posicionados absolutos são removidos do fluxo normal e podem sobrepor elementos.



No exemplo decima foram determinadas as posições para o elemento “div.absolute” através das propriedades right 0px e top 80px, mas isso só foi possível devido a propriedade position deste elemento ter o valor absolute e o elemento parent “pai” deste elemento possuí valor atribuído como relative.



Nesse exemplo o elemento pai do elemento com position absolute, não possui valor de position atribuído (ou position static). Por isso ele vai buscar em outro elemento com relação superior o posicionamento correto. Logo as prorpiedades right 0px e top 80px foram aplicadas ao corpo da página(elemento body).

position: fixed;

Um elemento com position: fixed;é posicionado em relação à janela de visualização, o que significa que ele sempre permanece no mesmo lugar, mesmo que a página seja rolada(scroll). As propriedades top, right, bottom e left são usadas para posicionar o elemento. Um elemento fixo não deixa uma lacuna na página onde normalmente estaria localizado.

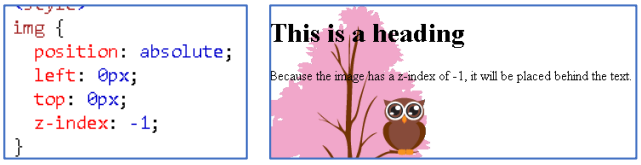
position: sticky;

Um elemento com position: sticky; é posicionado com base na posição de rolagem do usuário. Um elemento sticky se comporta de maneira parecida com elementos relative e fixed, dependendo da posição de rolagem. Ele tem uma

posição relativa até que uma determinada posição de deslocamento seja encontrada na viewport, então ele "gruda" no lugar (como position:fixed).

SOBREPOSIÇÃO Z-INDEX

Quando os elementos são posicionados, eles podem se sobrepor a outros elementos. A propriedade z-index especifica a ordem de pilha de um elemento (qual elemento deve ser colocado na frente ou atrás dos outros) com valores numéricos. Um elemento pode ter uma ordem de pilha positiva ou negativa. A propriedade z-index só funciona em elementos propriedade position definida e flex items (flexbox).



FLOAT e CLEAR

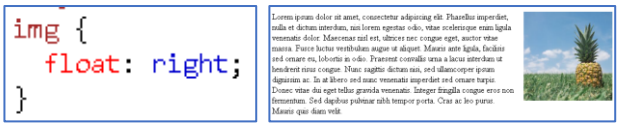
A propriedade float especifica como um elemento deve flutuar. A propriedade clear especifica quais elementos podem flutuar ao lado do elemento limpo e de que lado.

float

A propriedade float é usada para posicionar e formatar o conteúdo, por exemplo, deixar uma imagem flutuar à esquerda do texto em um contêiner. A float pode ter um dos seguintes valores:

- left- O elemento flutua à esquerda de seu contêiner
- right- O elemento flutua à direita de seu contêiner
- none- O elemento não flutua (será exibido exatamente onde ocorre no texto). Isso é padrão
- inherit- O elemento herda o valor float de seu pai

Em seu uso mais simples, a propriedade float pode ser usada para quebrar o texto em torno das imagens



Clear

Quando usamos a propriedade float, e queremos o próximo elemento abaixo (não à direita ou à esquerda), teremos que usar a propriedade clear. A propriedade clear especifica o que deve acontecer com o elemento que está próximo a um elemento flutuante. Ela pode ter um dos seguintes valores:

- none- O elemento não é empurrado abaixo dos elementos flutuantes à esquerda ou à direita. Isso é padrão
- left- O elemento é empurrado abaixo dos elementos flutuantes à esquerda
- right- O elemento é empurrado abaixo dos elementos flutuantes à direita
- both- O elemento é empurrado abaixo dos elementos flutuantes esquerdo e direito
- inherit- O elemento herda o valor clear de seu pai

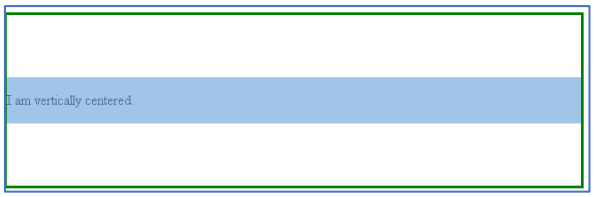
Ao limpar floats, você deve combinar o clear com o float: Se um elemento flutuar para a esquerda, então você deve limpar para a esquerda. Seu elemento flutuante continuará flutuando, mas o elemento desmarcado aparecerá abaixo dele na página da web.

ALINHAMENTO VERTICAL

Usando Padding.

Existem muitas maneiras de centralizar um elemento verticalmente em CSS. Uma solução simples é usar **top e bottom padding:**

```
div {
  padding: 70px 0;
  border: 3px solid green;
}
```



Usando Line-Height.

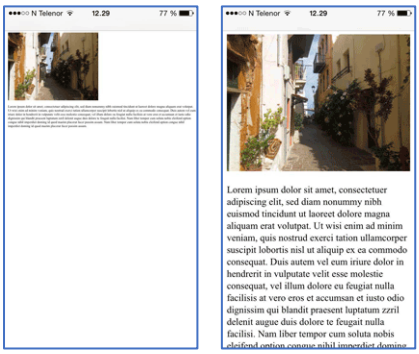
Outro truque é usar a propriedade line-height com um valor igual à propriedade height do elemento.

Usando Transform E Position

Se paddinge line-height não forem opções, outra solução é usar o posicionamento e a propriedade transform.

VIEWPORT

Viewport ou janela de visualização é a área visível do usuário de uma página da web. Ela varia de acordo com o dispositivo e será menor em um celular do que em uma tela de computador. Antes dos tablets e celulares, as páginas da web eram projetadas apenas para telas de computador, e era comum que as páginas da web tivessem um design estático e um tamanho fixo. Então, quando começamos a navegar na internet usando tablets e telefones celulares, as páginas da web de tamanho fixo eram muito grandes para caber na janela de visualização. Para corrigir isso, os navegadores desses dispositivos reduziram toda a página da Web para caber na tela.



Configurando a Viewport

O HTML5 introduziu um método para permitir que os web designers assumam o controle da viewport, por meio da <meta>tag. Você deve incluir o seguinte <meta>elemento de janela de visualização em todas as suas páginas da web:

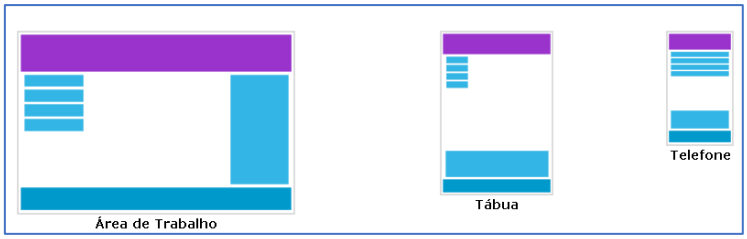
```
<meta name="viewport" content="width=device-width, initial-scale=1.0">
```

Isso fornece ao navegador instruções sobre como controlar as dimensões e o dimensionamento da página.

- A parte **width=device-width** define a largura da página para seguir a largura da tela do dispositivo (que varia de acordo com o dispositivo).
- A parte **initial-scale=1.0** define o nível de zoom inicial quando a página é carregada pela primeira vez pelo navegador.

RESPONSIVIDADE

O web design responsivo faz com que sua página da web fique bem em todos os dispositivos. Web design responsivo usa apenas HTML e CSS. Web design responsivo não é um programa ou JavaScript.



É possível fazer essa responsividade alterando valores de atributos como width e height para corresponder a um valor em porcentagem de elementos parente.

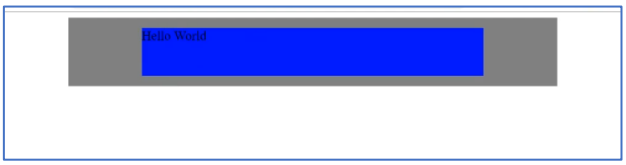
```css
.header{
    position: relative;
    margin:auto;
    width:  80%;
    height: 100px;
    background-color: gray;
}
```

```css
.content{
    position: absolute;

    width: 70%;
    height: 70%;
    background-color: blue;

    top:50%;
    left: 50%;


    transform: translate(-50%,-50%);
```



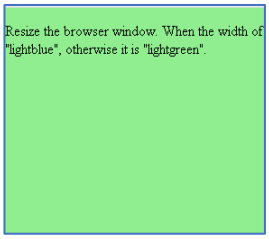
No exemplo acima um elemento parágrafo representado pela cor azul está dentro de um elemento div representado pela cor cinza, o elemento de cor azul sempre vai ficar com width proporcional a 80% da width do elemento cinza. Já o elemento cinza está com propriedades width e heigth diretamente relacionadas ao tamanho da viewport já que foram atribuídos valores em porcentagem para essas propriedades.

MEDIA QUERY

Media query é uma técnica CSS introduzida no CSS3. Ele usa a regra @media para incluir um bloco de propriedades CSS somente se uma determinada condição for validada. Ou seja, ela cria uma condição para certos atributos de estilos serem atribuídos.

Nesse exemplo a condição observa se o tamanho da janela do navegador corresponde a 600px ou menor, caso isso aconteça a cor de fundo será azul claro:

```
@media only screen and (max-width: 600px) {
  body {
    background-color: lightblue;
  }
}
```



Media query é útil quando você deseja modificar seu site ou aplicativo dependendo do tipo de dispositivo (como tela de impressão x tela) ou características e parâmetros específicos (como resolução de tela ou largura da janela de visualização do navegador).

Sintaxe

Uma media query pode ser composta por **media types** (opcional) e expressões **media feature**, que podem ser combinadas de acordo com a necessidade e de varias maneiras através do uso de **operadores lógicos**. Expressões media query não diferenciam letras minúsculas e letras maiúsculas.

Media types – Definem qual a tipo de dispositivo a media query esta sendo aplicada: all (Adequado para todos os dispositivos), print (Destinado a material paginado e documentos visualizados em uma tela no modo de visualização de impressão), screen (Destinado para telas). Se não for especificado o media type, será então atribuído o valor “all” (exceto quando os operadores not e only são utilizados, pois os mesmos exigem os media types).

```
@media print { ... }
```

Media features – Descrevem e especificam características do agente de usuário (página web, browser, viewport, etc): any-hover, any-pointer, aspect-ratio, color, color-gamut, color-index, device-aspect-ratio, device-height, device-width, display-mode, forced-colors, grid, height, hover, inverted-colors, monochrome, orientation, overflow-block, overflow-inline, pointer, prefers-color-scheme, prefers-contrast, prefers-reduced-motion, resolution, scripting, update, width. As expressões de media features devem ser especificadas entre parentes.

```
@media (hover: hover) { ... }
```

Por exemplo, a feature (recurso) **hover** permite que a media query teste se o usuário está passando o mouse em cima do elemento e essa condição for atendia estilos podem ser atribuídos, etc.

Operadores Lógicos – Podem ser usadas para compor media querys mais complexas. São eles: “not”, “Only”, “and” e “,(vírgula)”.

Exemplos:

Este exemplo combina dois media features para restringir estilos a dispositivos orientados em paisagem e com uma largura de pelo menos 30 em:

```
@media (min-width: 30em) and (orientation: landscape) {
... }
```

A regra a seguir aplicará seus estilos se o dispositivo do usuário tiver uma altura mínima de 680px ou for um dispositivo de tela no modo retrato:

```
@media (min-height: 680px), screen and (orientation:
portrait) { ... }
```

REFERENCIAS

https://alistapart.com/article/responsive-web-design/ MEDIA QUERY

https://developer.mozilla.org/en-US/docs/Web/CSS/Media_Queries/Using_media_queries#Media_features – MEDIA QUERY (MEDIA FEATURES E MEDIA TYPES)