Curso Superior de Tecnologia em Sistemas para Internet
Desenvolvimento front-end I (front 1)
Rafael Cunha Cardoso
rafaelcardoso@ifsul.edu.br

display & position properties

INSTITUTO FEDERAL
Sul-rio-grandense
Câmpus Pelotas

`display` property

Propriedade CSS mais importante para **controlar o *layout***.

Especifica se/como um elemento é exibido.

Cada elemento HTML tem um valor de exibição (display) padrão, dependendo do tipo de elemento.

O valor de exibição padrão para a maioria dos elementos é `block` ou `inline`.

`Elementos nível de bloco`

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

`Elementos inline`

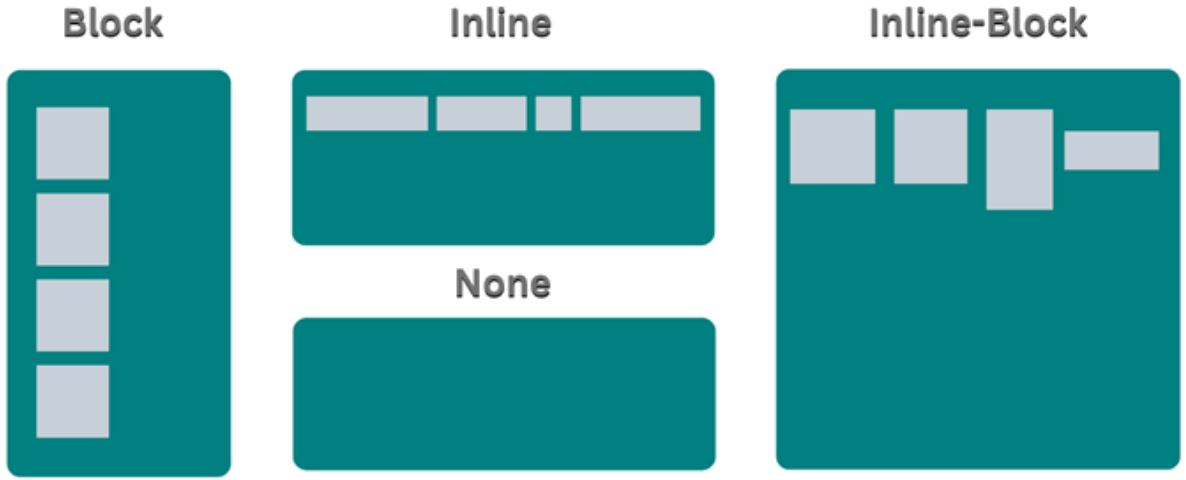
Um elemento **inline** não **começa em uma nova linha e ocupa apenas a largura necessária**.

Sobrescrevendo o valor de `display`

Como vimos, cada elemento tem **um valor de exibição padrão**.

No entanto, é possível alterar essas configuração.

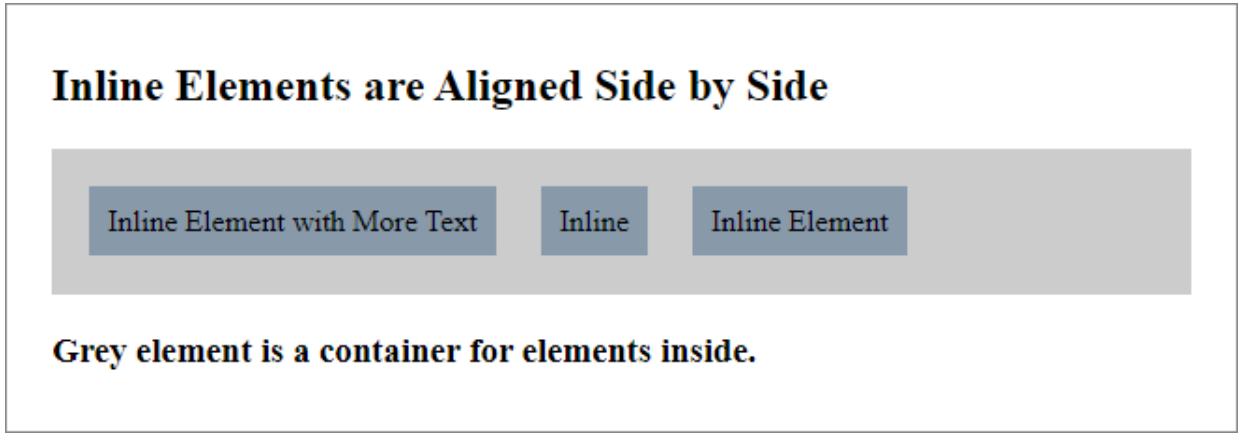
Transformar um elemento ***inline*** em um elemento de **bloco** (ou vice-versa), pode ser útil para fazer a página parecer de uma maneira específica e ainda seguir os padrões da web.



display: inline

Ao aplicar **display:inline** os elementos são **dispostos em linha**.

Geralmente, elementos **inline** são partes de conteúdo, incluindo textos, imagens, links, que não quebram a linha e não podem definir a dimensão arbitrariamente.



Um exemplo comum desse tipo de configuração é alterar o display de elementos <li> para *inline,* visando criar menus horizontais:

```
li {
  display: inline;
}
```

Exibindo <li> como elementos inline


```
<html lang="en">

    <style>
        li {
            display: inline;
        }
    </style>

    <title>Lista de links como menu horizontal</title>
</head>

<body>

    <p>Exibindo uma lista de links como menu horizontal:</p>

    <ul>
        <li><a href="home.asp" target="_blank">Início</a></li>
        <li><a href="alunos.asp" target="_blank">Alunos</a></li>
        <li><a href="docentes.asp" target="_blank">Professores</a></li>
    </ul>


</body>

</html>
```

Elementos <li> com configuração normal (block):

Exibindo uma lista de links como menu horizontal:

- Início
- Alunos
- Professores

Elementos <li> configurados como inline:

Exibindo uma lista de links como menu horizontal:

Início Alunos Professores

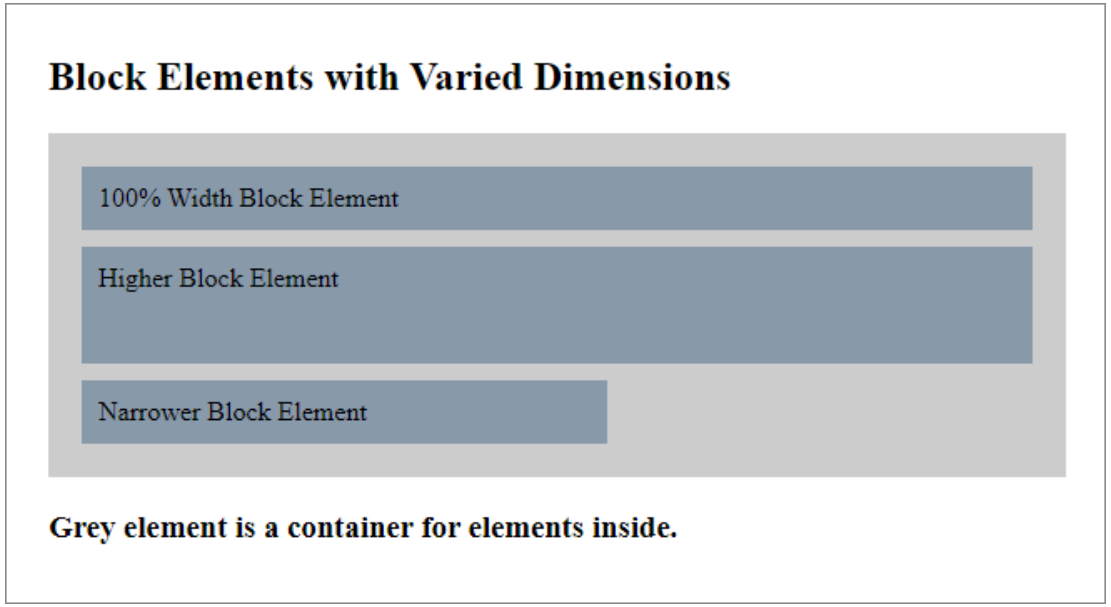
Exemplo

display: block

O layout da página da Web é estruturado principalmente por **elementos de bloco**.

Geralmente o elemento **`div`** (e elementos semânticos) é a escolha mais comum para construir a espinha dorsal do layout da web.

Ao contrário dos elementos *inline*, os elementos de bloco irão quebrar a linha e podem definir a **dimensão arbitrariamente**.

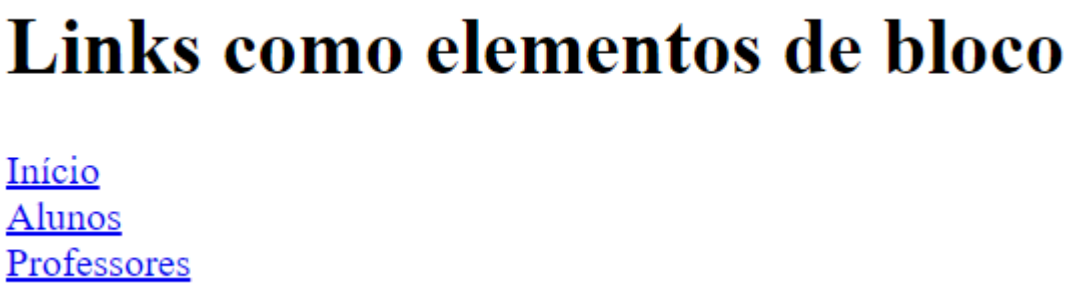


Exibindo <a> como elementos de bloco

```
<html lang="en">

<head>
   <style>
        a {
            display: block;
        }
    </style>
</head>

<body>

    <h1>Links como elementos de bloco</h1>

    <a href="home.html" target="_blank">Início</a>
    <a href="alunos.html" target="_blank">Alunos</a>
    <a href="docentes.html" target="_blank">Professores</a>

</body>

</html>
```

Elementos <a> com configuração normal (inline):

Elementos <li> configurados como inline:



Exemplo

Outros valores para display

Inline e *block* são os modos de exibição mais básicos para elementos HTML desde o início do desenvolvimento da web.

Há anos, **alinhar várias *boxes* na mesma linha** é algo difícil de realizar.

Elementos *inline* podem ser alinhados lado a lado, **mas não é possível definir suas dimensões**.

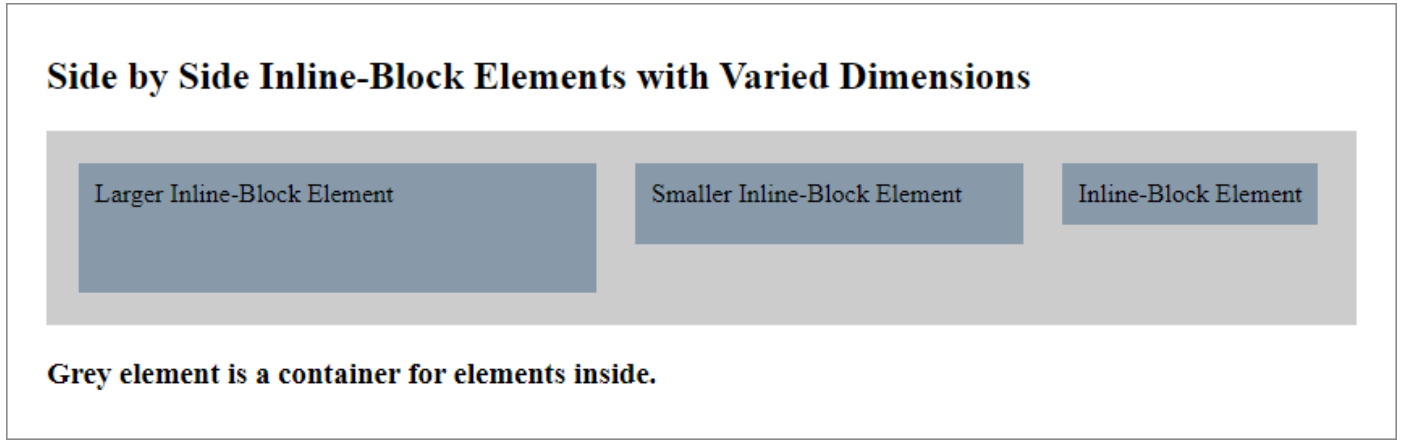
Já elementos do bloco podem ter dimensões definidas, **mas sempre quebram a linha sozinhos**.

display: inline-block

O valor display: **inline-block** é uma combinação das propriedades de display: block e display: inline.

inline-block

Elementos configurados com esse valor, podem ter dimensões configuradas sem quebra de linha.



São dispostos em linha, um ao lado do outro, como elementos **inline.**

Diferente dos elementos inline, esses elementos respeitam as propriedades do box model (width, height, margin e padding), como elementos block

Vantagens:

- **Alinhamento Horizontal**: Permite colocar elementos em linha e permite controlar suas dimensões (largura e altura).
- **Substituto para Floats**: Em muitos casos, é uma boa alternativa ao uso de floats para criar layouts de múltiplas colunas.

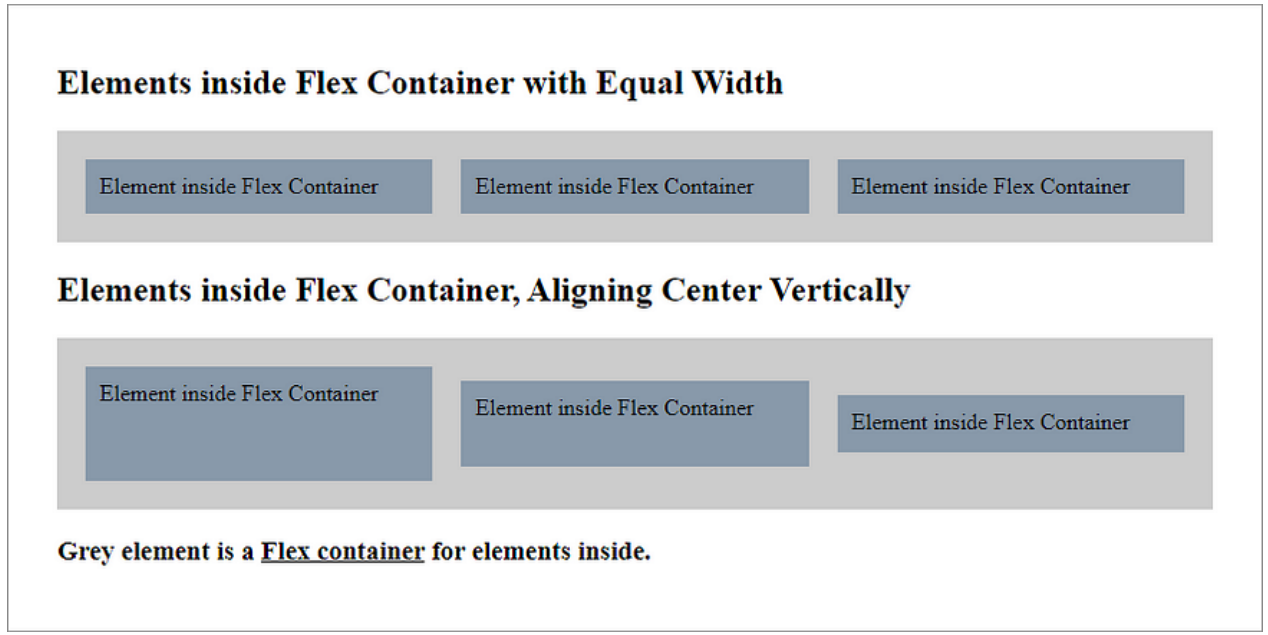
display: flex

Elementos inline-block são utilizados para alinhar várias caixas lado a lado, mas provém pouco controle sobre os detalhes do layout.

O Flexbox (ou *Flexible Box Layout*) é um **modelo de layout unidimensional** que fornece uma maneira eficiente de distribuir espaço entre itens dentro de um contêiner e de alinhar esses itens.

Antes do CSS Flexbox, não havia uma ferramenta poderosa para layout da web.

O elemento HTML com `display:flex` é um contêiner Flexbox para outros elementos internos.



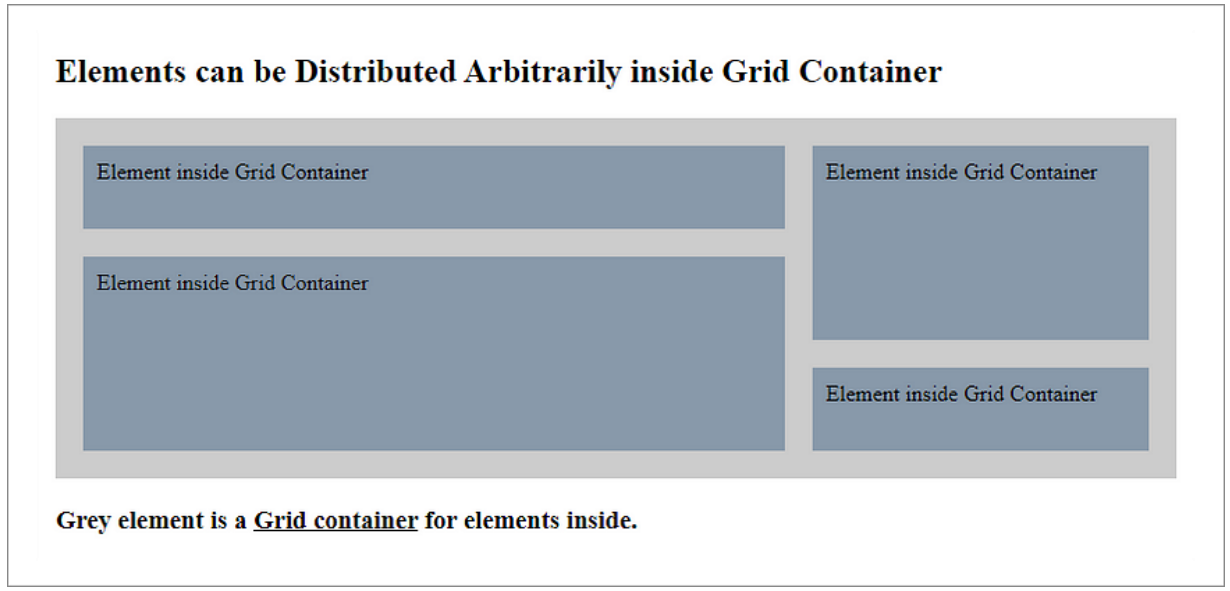
display: grid

Pode-se dizer que o sistema CSS Grid é uma evolução do sistema Flexbox.

O Flexbox ajuda a **alinhar caixas em apenas uma dimensão**, por linha ou coluna.

Por sua vez, o sistema em grade amplia as possibilidades, permitindo o **alinhamento das caixas em duas dimensões ao mesmo tempo**.

Assim como ocorre no Flexbox, o elemento HTML com `display:grid` torna-se um contêiner de grade (grid) para outros elementos internos. É o núcleo do sistema **CSS Grid**.



“Escondendo” elementos

`display:none`

Oculta um elemento fazendo com que a página seja exibida como se o **elemento não existisse** (elemento não afeta o layout).

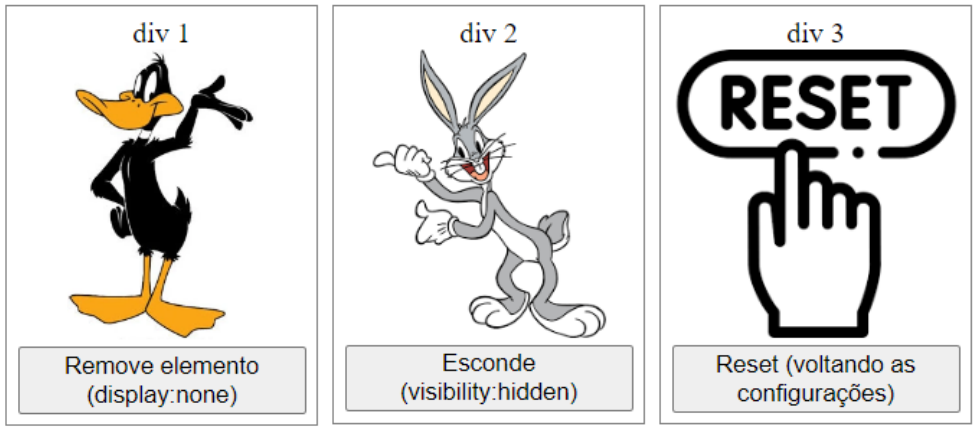
`visibility:hidden`

Também **esconde** um elemento, no entanto, **ele ainda ocupará o mesmo espaço na página** (o elemento oculto desta forma ainda afetará o layout).

display:none vs visibility:hidden

`visibility:hidden` Oculta o elemento, mas ainda ocupa espaço no layout.

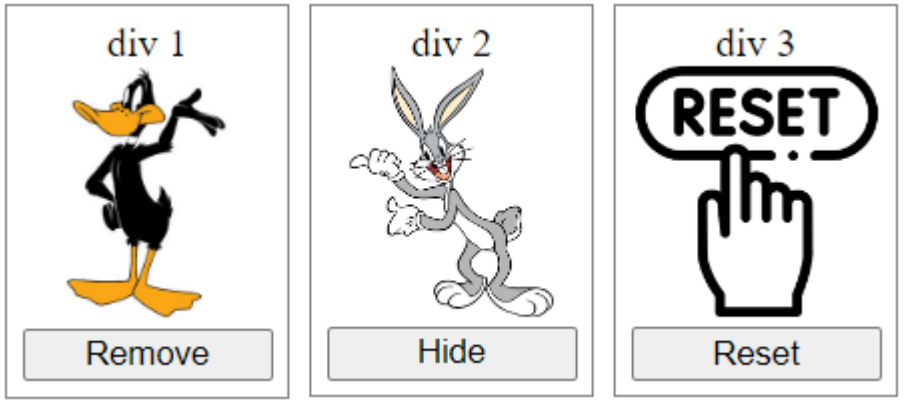
`display:none` Remove o elemento do documento. Não ocupa nenhum espaço.



Exemplo

```
<html>

<body>

    <div class="imgbox" id="imgdiv1">div 1<br>
        <img src="img/Patolino.webp" alt="Patolino">
        <button onclick="removeElement()">Remove</button>
    </div>

    <div class="imgbox" id="imgdiv2">div 2<br>
        <img src="img/pernalonga.png" alt="Pernalonga">
        <button onclick="changeVisibility()">Esconde</button>
    </div>

    <div class="imgbox">div 3<br>
        <img src="img/reset.png" alt="Reset" >
        <button onclick="reset()">Reset</button>
    </div>

</body>

</html>
```

```
<script>
    function removeElement() {
        document.getElementById("imgdiv1").style.display = "none";
    }

    function changeVisibility() {
        document.getElementById("imgdiv2").style.visibility = "hidden";
    }

    function reset() {
        document.getElementById("imgdiv1").style.display = "block";
        document.getElementById("imgdiv2").style.visibility = "visible";
    }
</script>
```



display:none vs visiblity: hidden

`visibility:hidden` Oculta o elemento, mas ainda ocupa espaço no layout.

`display:none` Remove o elemento do documento. Não ocupa nenhum espaço.

Exemplo

Valor	Descrição
inline	Exibe um elemento como um elemento inline (Exemplo:). Quaisquer propriedades de altura e largura não terão efeito.
block	Exibe um elemento como um elemento de bloco (exemplo <p>). Começa em uma nova linha e ocupa toda a largura disponível.
contents	Faz o container desaparecer, tornando os seus elementos filhos, em filhos do elemento do próximo nível no DOM
flex	Exibe um elemento como um contêiner flexível em nível de bloco
grid	Exibe um elemento como um contêiner de grid em nível de bloco
inline-block	Torna o elemento um contêiner de bloco de nível inline. O elemento é formatado como um elemento inline, mas é possível aplicar valores de altura e largura
inline-flex	Torna o elemento um contêiner flexível de nível inline
inline-grid	Torna o elemento um contêiner de grade de nível inline
inline-table	O elemento é exibido como uma tabela de nível inline
list-item	O elemento se comporta como um elemento
run-in	Exibe um elemento como bloco ou embutido, dependendo do contexto
table	Deixe o elemento se comportar como um elemento <table>

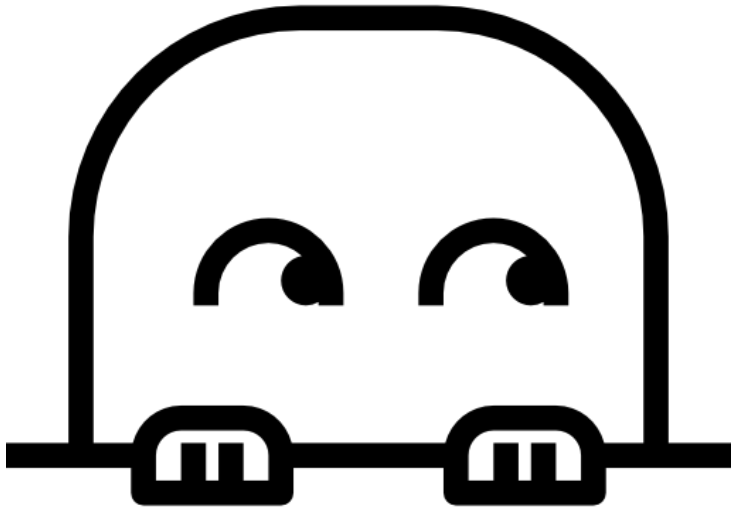
Valores aceitos por `display`

Valor	Descrição
`table-caption`	Faz o elemento se comportar como um elemento <caption>
`table-column-group`	Faz o elemento se comportar como um elemento <colgroup>
`table-header-group`	Faz o elemento se comportar como um elemento <thead>
`table-footer-group`	Faz o elemento se comportar como um elemento <tfoot>
`table-row-group`	Faz o elemento se comportar como um elemento <tbody>
`table-cell`	Deixe o elemento se comportar como um elemento <td>
`table-column`	Deixe o elemento se comportar como um elemento <col>
`table-row`	Deixe o elemento se comportar como um elemento <tr>
`none`	O elemento é completamente removido
`initial`	Define esta propriedade com seu valor padrão.
`inherit`	Herda esta propriedade de seu elemento pai.

Importante

Configurar a propriedade **`display`** de um elemento **altera apenas como o elemento é exibido**, **NÃO o tipo de elemento**.

Por exemplo, um elemento **`inline`** com **`display:block`**; não permite ter outros elementos de bloco dentro dele, porque ele ainda é um elemento *em linha*.



position

position

Propriedade que especifica o tipo de método de posicionamento usado para um elemento.

Existem 5 valores diferentes para esta propriedade:

1) **static;**
2) **relative;**
3) **fixed;**
4) **absolute;** OU
5) **sticky.**

A partir daí, os elementos são posicionados usando as propriedades **top**, **bottom**, **left** e **right**.

No entanto, essas propriedades não funcionarão a menos que a propriedade **position** seja definida primeiro.

Eles também funcionam de forma diferente, dependendo do valor de **position**.

position: static;

Os elementos HTML são posicionados estáticos **por padrão**.

Elementos posicionados estaticamente não são afetados pelas propriedades **top**, **bottom**, **left** e **right**.

Um elemento com **position:static** é posicionado de acordo com o **fluxo normal da página**.

```
<body>
    <h2><code>position: static;</code> </h2>

    <p>Um elemento configurado com <code>position: static;</code>
        sempre é posicionado de acordo com o fluxo normal da página</p>

    <div class="static">
        Elemento div com <code>position: static;</code>
    </div>
</body>
```

Exemplo

```
div.static {
    position: static;
    border: 3px solid #657e41;
}
```

position: static;

Um elemento configurado com `position: static;` sempre é posicionado de acordo com o fluxo normal da página

Elemento div com `position: static;`

position: relative;

Um elemento com **position:relative** está posicionado em relação à sua **posição normal**.

Definir as propriedades **top**, **bottom**, **left** e **right** de um elemento com **position:relative** fará com que ele seja posicionado de forma relativa a sua posição normal.

Outros conteúdos não serão ajustados para caber em qualquer lacuna deixada pelo elemento.

```
<body>
    <h2><code>position: relative;</code> </h2>

    <p>Um elemento configurado com <code>position: relative;</code>
        sempre é posicionado relativamente à sua posição original página</p>

    <div class="relative">
        Elemento div com <code>position: relative;</code>
    </div>
</body>
```

Exemplo 1

```
div.relative {
   position: relative;
   left: 60px;
   border: 3px solid #657e41;
}
```

position: relative;

Um elemento configurado com `position: relative;` sempre é posicionado relativamente à sua posição original página

Elemento div com `position: relative;`

position: fixed;

Um elemento com **position:fixed** é removido do fluxo normal e posicionado em relação à **viewport**.

Ou seja, ele permanece fixo no mesmo lugar, mesmo que a página seja rolada.

As propriedades **top**, **bottom**, **left** e **right** são utilizadas para posicionar o elemento.

```
<body>
    <h2><code>position: fixed;</code> </h2>

    <p>Um elemento configurado com position:fixed está posicionado em relação
à viewport, o que significa que sempre permanece no mesmo lugar, mesmo que a
página seja rolada</p>

    <div class="fixed">
        Elemento div com <code>position: fixed;</code>
    </div>
</body>
```

```
div.fixed {
    position: fixed;
    bottom: 0;
    right: 0;
    width: 300px;
    border: 3px solid #73AD21;
}
```

position: fixed;

Um elemento configurado com `position: fixed;` está posicionado em relação à *viewport*, o que significa que sempre permanece no mesmo lugar, mesmo que a página seja rolada

Elemento div com `position: fixed;`

Exemplo

position: absolute;

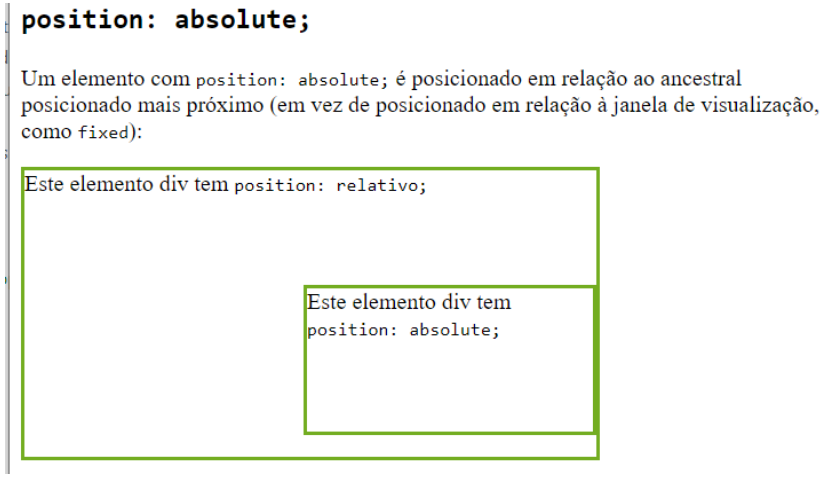
Um elemento com `position:absolute` é removido do fluxo normal e posicionado em relação a seu ancestral mais próximo (ao invés de ser posicionado em relação à **viewport**, como o **fixed**)

Se o elemento não tiver ancestrais posicionados, ele usará o **body** do documento e se moverá junto com a rolagem da página.

```
<body>
    <h2><code>position: absolute;</code> </h2>

      <div class="relative">Este elemento div tem position:relative
         <div class="absolute">Este elemento div tem position:absolute; </div>
      </div>
</body>
```

```
div.relative {
      position: relative;
      width: 400px;
      height: 200px;
      border: 3px solid #73AD21;
 }
```

```
div.absolute {
     position: absolute;
     top: 80px;
     right: 0;
     width: 200px;
     height: 100px;
     border: 3px solid #73AD21;
}
```



Exemplo

position: sticky;

Um elemento com **position:sticky** é posicionado com base na posição de rolagem do usuário.

Um elemento **sticky** alterna entre **relative** e **fixed**, dependendo da posição de rolagem.

Ele é posicionado como **relative** até que uma determinada posição de deslocamento seja encontrada na **viewport** - então ele "gruda" no lugar (como **position:fixed**).

```
<body>
    <p>Role para ver o funcionamento do <code>position:sticky</code>.</p>

    <div class="sticky">Olá <code>sticky</code>!</div>

    <div style="padding-bottom:2000px">
        <p>O elemento sticky fica no topo da página (top:0), quando o   usuário
            atinge a posição de rolagem.</p>
        <p>Scroll para cima para "desgrudar".</p>
        <p>Lorem ipsum dolor sit amet consectetur...</p>
        <p>Lorem ipsum dolor sit amet consectetur ...</p>
    </div>
</body>
```

```
div.sticky {
  position: sticky;
   top: 0;
   padding: 5px;
   background-color: #cae8ca;
   border: 2px solid #4CAF50;
}
```

Scroll para ver o funcionamento do `position:sticky`.

Olá `sticky`!

O elemento `sticky` fica no topo da página (`topo: 0`), quando o usuário atinge a posição de rolagem.

Scroll para cima para "desgrudar".

Lorem ipsum dolor sit amet consectetur, adipisicing elit. Molestiae maxime maiores doloribus, sequi, voluptatem id aperiam expedita a natus blanditiis accusamus corporis rem, doloremque impedit dolores nam asperiores nobis laboriosam.

Exemplo

Resumo `position`

- **`static`**: Útil quando não é preciso nenhum posicionamento especial (*default*).
- **`relative`**: Recomendada para ajustes finos e quando é preciso deslocar um elemento de sua posição normal sem afetar o *layout* dos outros elementos.
- **`absolute`**: Utilizada para criar layouts complexos e sobreposições onde o elemento precisa ser posicionado em relação ao seu contêiner ou à página.
- **`fixed`**: Útil para elementos que devem permanecer visíveis durante a rolagem, como cabeçalhos fixos ou barras de navegação.
- **`sticky`**: Interessante para cabeçalhos ou menus que devem se fixar ao topo da tela após serem rolados até certo ponto.

Essas propriedades permitem um controle preciso sobre o layout e a aparência dos elementos em uma página web, facilitando a criação de interfaces dinâmicas e responsivas.

Curso Superior de Tecnologia em Sistemas para Internet
Desenvolvimento front-end I (front 1)
Rafael Cunha Cardoso
rafaelcardoso@ifsul.edu.br

display & position properties

INSTITUTO FEDERAL
Sul-rio-grandense
Câmpus Pelotas