Curso Superior de Tecnologia em Sistemas para Internet
Desenvolvimento front-end I (DFE_1)
Rafael Cunha Cardoso
rafaelcardoso@ifsul.edu.br

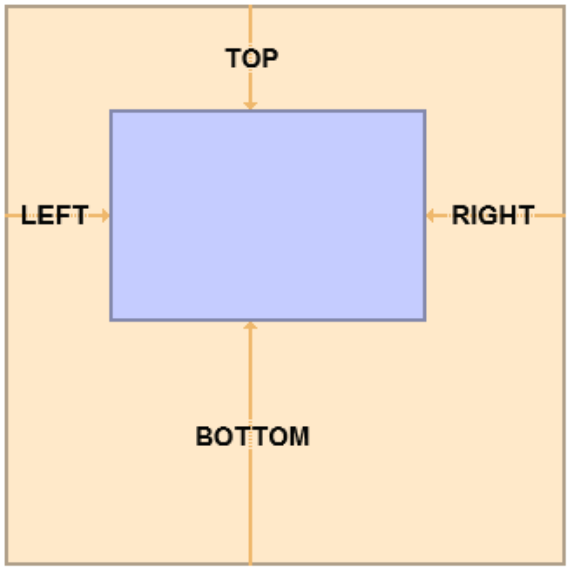
Posicionamento de elementos

INSTITUTO FEDERAL
Sul-rio-grandense
Câmpus Pelotas

Mecanismos de posicionamento

Permitem ao desenvolvedor **alterar o comportamento padrão dos elementos**, não só modificando a ordem como também **posicionando elementos de nível de bloco um ao lado do outro**.

Isso é feito por meio de **aplicação regras CSS que definem valores para as propriedades destinadas a posicionar os elementos** na página.



Layout

centralizado

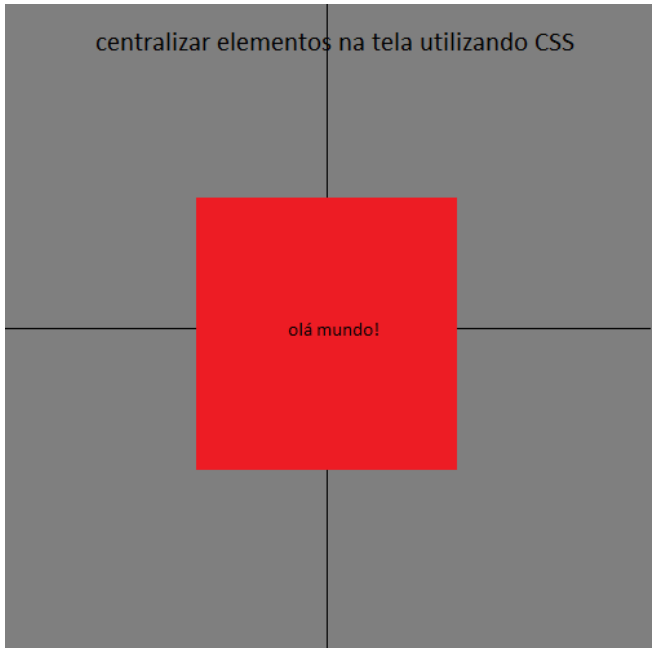
Como centralizar elementos?

Uma atividade bastante constante em desenvolvimento de interfaces é realizar a **centralização adequada de elementos na página**.

É possível utilizar uma guia simples para orientar a realização desta tarefa.

1) Qual tipo de elemento se quer centralizar?
Elemento em linha ou elemento de bloco?

2) Qual sentido deseja-se centralizar?
Horizontal, vertical ou em ambos sentidos?



Centralizando elementos de bloco

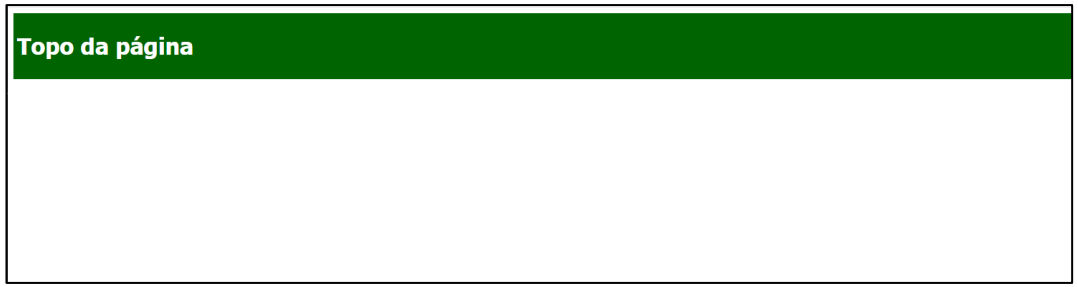
Um **elemento em nível de bloco ocupa todo o espaço de seu elemento pai** (container), criando assim um “bloco”.

```
<body>
    <header>
        <h1>Topo da página</h1>
    </header>
</body>
```

H1 vai ocupar todo o espaço, de seu pai, o **header**.

```
<style>
    header {

        padding: 5px;
        background: darkgreen;
        font-family: Tahoma;
        color: #fff;
    }
</style>
```



Centralizando elementos de bloco

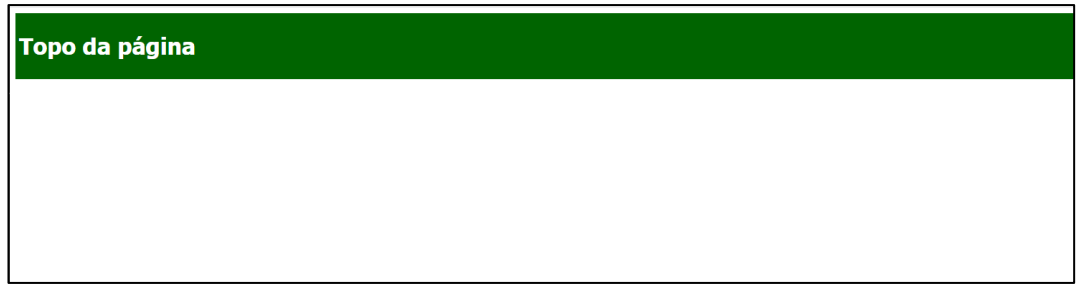
Elementos nível de bloco podem ser centralizados **horizontalmente** por meio da propriedade **`margin: auto`**.

```
<body>
    <header>
        <h1>Topo da página</h1>
    </header>
</body>
```

```
<style>
    header {

        margin: auto;
        padding: 5px;
        background: darkgreen;
        font-family: Tahoma;
        color: #fff;
    }
</style>
```

Alinha centralizado em relação ao **contêiner** (elemento pai)



O alinhamento centralizado **não funciona** sem a definição da propriedade **`width`** ou se ela for configurada em 100%.

Nestes casos, o elemento ocupará totalmente o espaço reservado a ele

Centralizando elementos de bloco

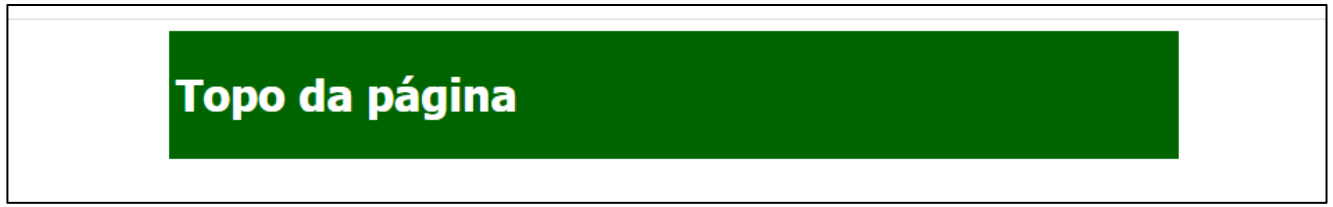
Para **centralizar horizontalmente** então, é necessário definir uma **largura específica** para o elemento por meio da propriedade **`width`**.

```
<body>
    <header>
        <h1>Topo da página</h1>
    </header>
</body>
```

```
<style>
    header {
        width: 75%;
        margin: auto;
        padding: 5px;
        background: darkgreen;
        font-family: Tahoma;
        color: #fff;
    }
</style>
```

Especifica uma largura para o elemento selecionado.

Exemplo



Assim, o espaço restante na tela é **automaticamente distribuído igualmente em ambos os lados**.

Centralizando elementos *inline*

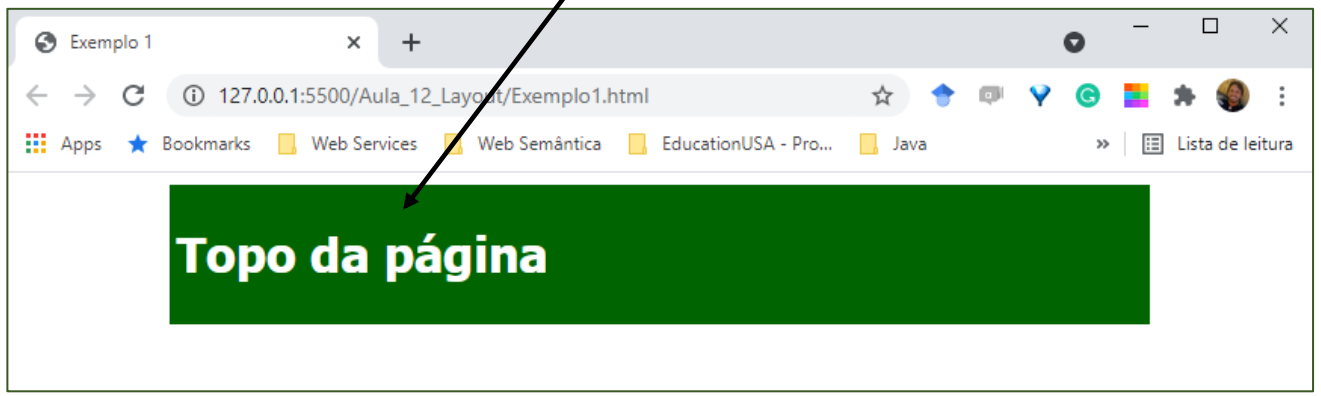
```html
<header>
    <h1>Topo da página</h1>
</header>
```

HTML

```css
header{
    width: 75%;
    margin: auto;
    padding: 5px;
    background: darkgreen;
    font-family: Tahoma;
    color: #fff;
}
```

CSS

2) Mas o texto ainda permanece com sua configuração padrão.



1) Com as alterações em `margin` e `width`, o elemento de bloco foi centralizado em relação a página.

`width:75%` - Largura ocupada pelo elemento

`margin: auto` – o navegador calcula a margem

Centralizando horizontalmente elementos *inline*

```html
<header>
    <h1>Topo da página</h1>
</header>
```

HTML

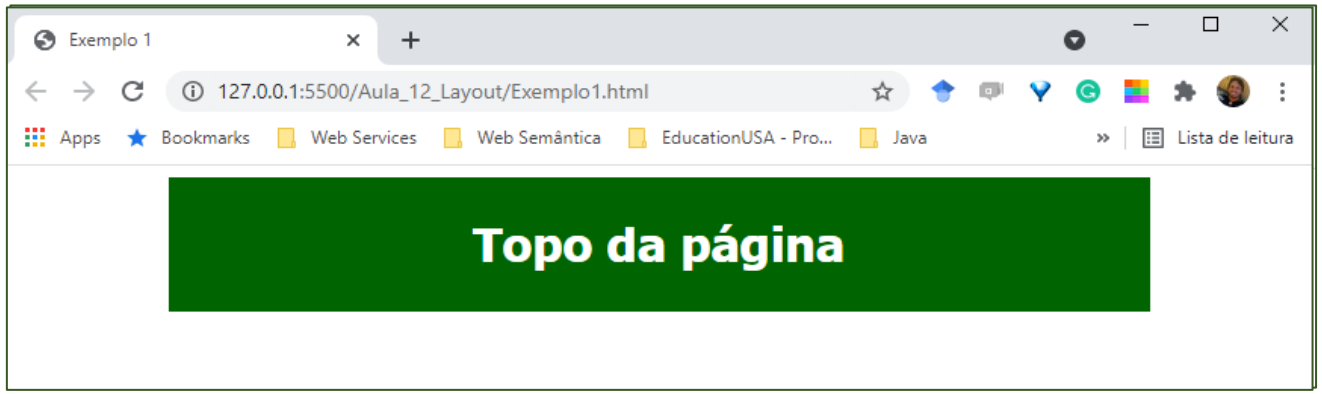
```css
header{
    width: 75%;
    margin: auto;
    padding: 5px;
    background: #036;
    font-family: Tahoma;
    color: #fff;
    text-align: center;
}
```

CSS

text-align
Alinha o conteúdo interno em relação ao próprio elemento.

Como centralizar o texto?

Exemplo



Elementos ***inline*** admitem apenas valores para as margens **`left`** e **`right`**. Margens de **`top`** e **`bottom`** são ignoradas para esse tipo de elemento.

Alinhamento vertical de elementos *inline*

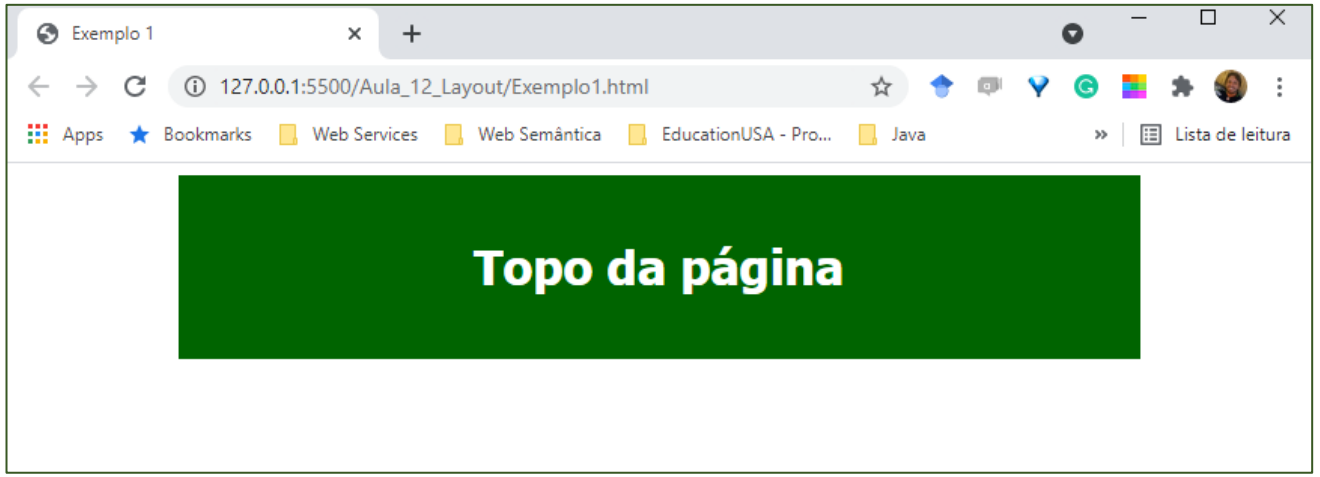
```html
<header>
    <h1>Topo da página</h1>
</header>
```

HTML

```css
header{
    width: 75%;
    margin: auto;
    padding: 20px 0;
    background: #036;
    font-family: Tahoma;
    color: #fff;
    text-align: center;
}
```

CSS

Exemplo



O alinhamento **vertical** pode ser feito por meio da propriedade **padding**.

Propriedade

float

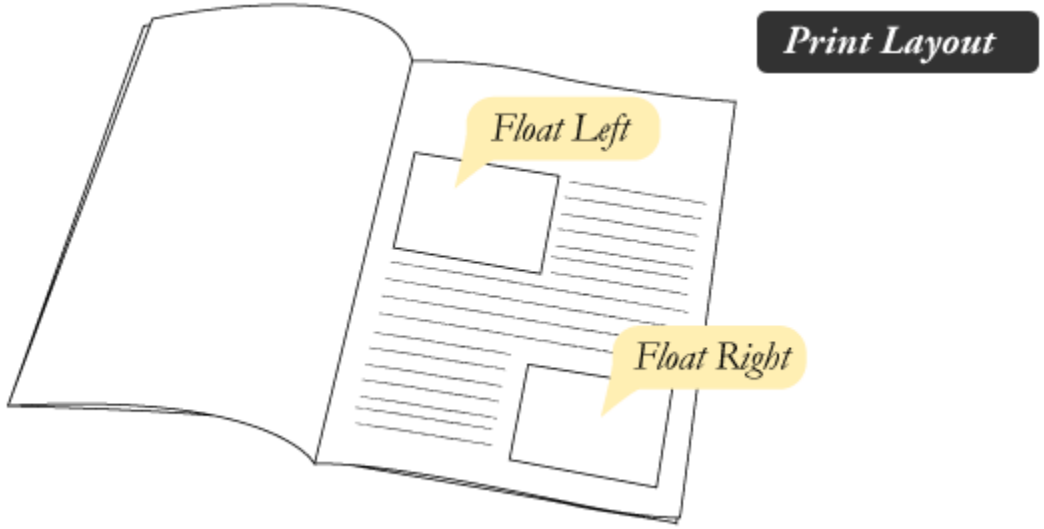
float

Propriedade utilizada para realizar posicionamento.

Para entender seu **propósito e origem**, podemos olhar para o design de impressão.

Em um **layout de impressão**, as imagens podem ser definidas na página de forma **que o texto as envolva** conforme necessário.

Isso é comumente chamado de "*text wrap*".



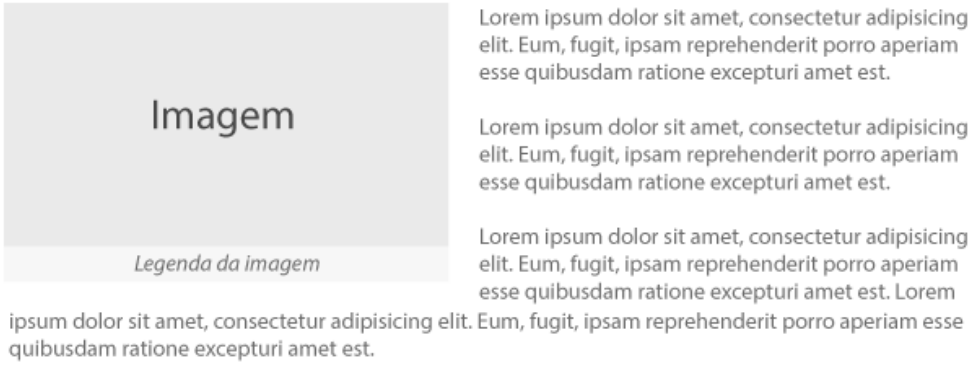
Comumente usada para criar **layouts de estilo de jornal ou revista** antigamente, onde os elementos eram dispostos em várias colunas ao longo da página.

`float`

Por meio de regras CSS que utilizam **`float`** é possível compor *layouts* **sem o uso de tabelas** (*tableless*).

Isso traz benefícios como economia no **tempo de carregamento, adequação às normas W3C e, um aumento de performance**.

Embora o elemento flutuante não siga o fluxo normal da página, **ele ainda faz parte do fluxo**. Assim, o elemento se deslocará para a **direita ou para a esquerda até tocar em outro elemento flutuante ou na borda do contêiner**.



Atualmente, as principais tecnologias usadas para layouts em desenvolvimento web são **Flexbox** e **CSS Grid**.

Mas para registro histórico (e caso se encontre essas soluções em algumas aplicações), é importante conhecer como funciona a estratégia do **`float`**.

float

Determina que um elemento deve ser retirado do seu fluxo normal e colocado ao longo do lado direito ou esquerdo do seu contêiner.

O espaço original de posicionamento ocupado pela caixa, será ocupado pelo elemento que se segue no fluxo do documento.

Especifica como um elemento deve **flutuar** no layout.

Remoção do fluxo normal:

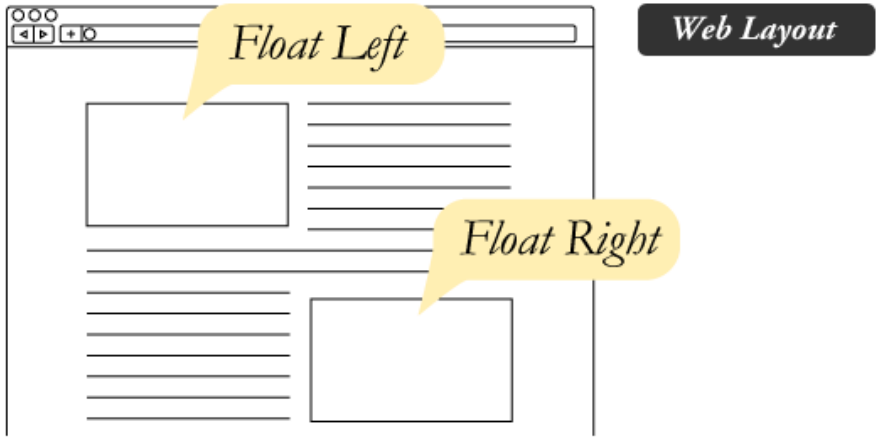
Quando um elemento é flutuado, **ele é removido do fluxo normal** do documento. Isso significa que outros elementos fluirão ao redor do elemento flutuado.

float

Propriedade é utilizada para **posicionamento e formatação de conteúdo**.

Podem ser utilizados os seguintes valores:

- **`left`** elemento flutua à **esquerda** do container;
- **`right`** elemento flutua à **direita** do container;
- **`none`** valor padrão que **não flutua** elementos; e
- **`inherit`** elemento **herde** o valor float do elemento pai.



Propriedade `float`

O comportamento de elementos de bloco, (**`display:block`**) faz que um **elemento seja disposto embaixo de outro**.

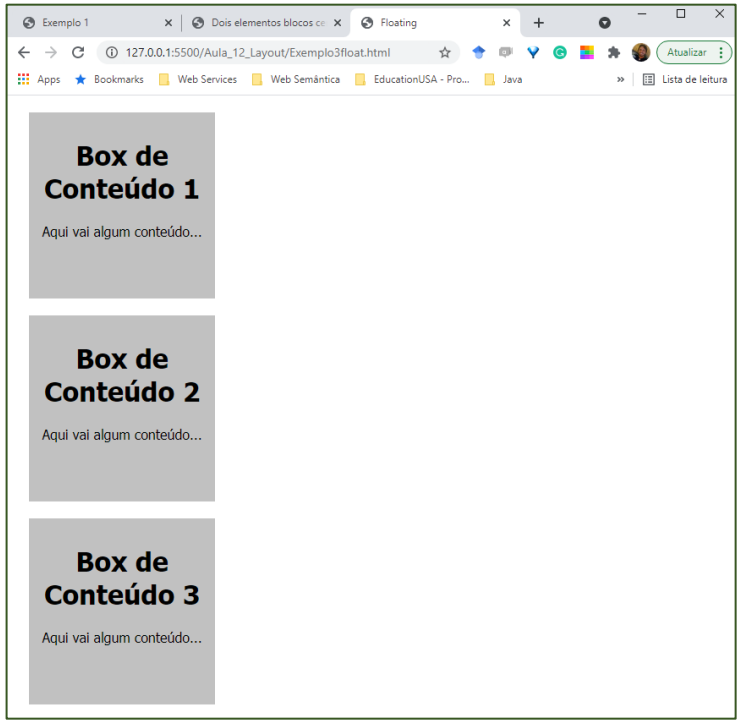
HTML

```
<div class="bloco">
    <h1>Box de Conteúdo 1</h1>
    <p>Aqui vai algum conteúdo... </p>
</div>
<div class="bloco">
    <h1>Box de Conteúdo 2</h1>
    <p>Aqui vai algum conteúdo...</p>
</div>
<div class="bloco">
    <h1>Box de Conteúdo 3</h1>
    <p>Aqui vai algum conteúdo...</p>
</div>
```

CSS

```
.bloco{
    width: 200px;
    height: 200px;
    padding: 10px;
    background: #c1c1c1;
    margin: 20px;
    font-family: Tahoma;
    text-align: center;

}
```



Assim, não importa o tamanho (**`width`**) do elemento, já que elementos de bloco sempre preenchem o restante do espaço com **`margin`**, jogando o próximo elemento para a outra linha.

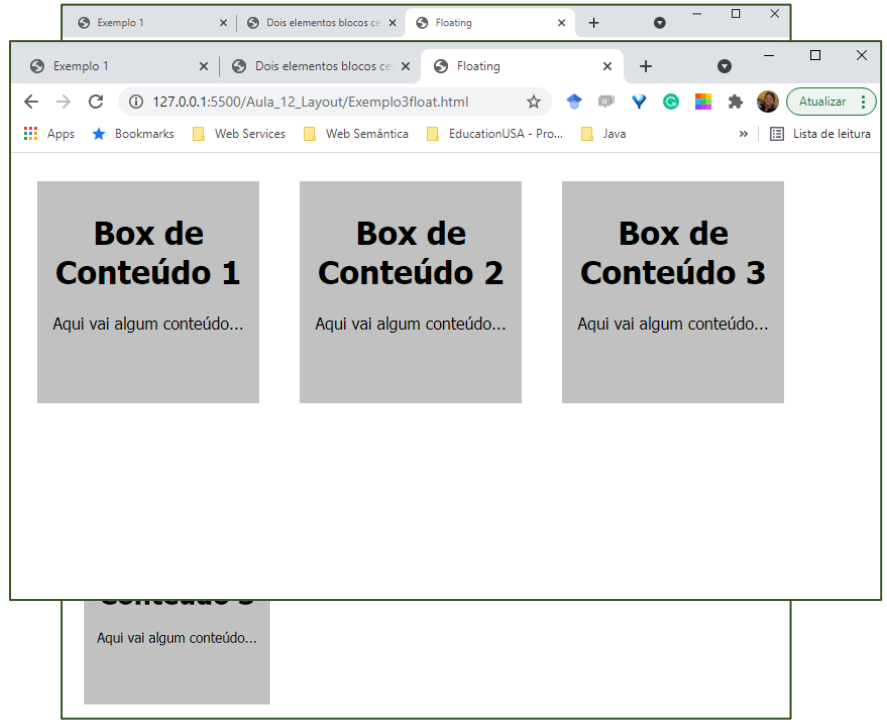
Propriedade `float`

HTML

```
<div class="bloco">
    <h1>Box de Conteúdo 1</h1>
    <p>Aqui vai algum conteúdo... </p>
</div>
<div class="bloco">
    <h1>Box de Conteúdo 2</h1>
    <p>Aqui vai algum conteúdo...</p>
</div>
<div class="bloco">
    <h1>Box de Conteúdo 3</h1>
    <p>Aqui vai algum conteúdo...</p>
</div>
```

CSS

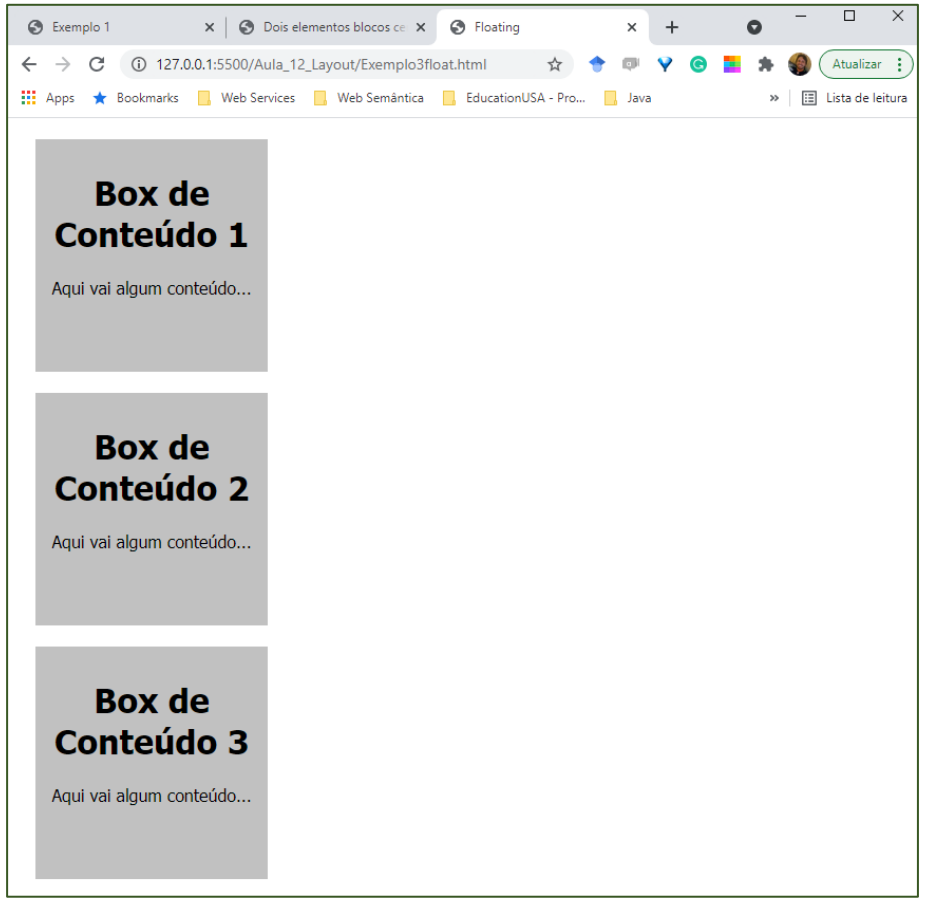
```
.bloco{
    width: 200px;
    height: 200px;
    padding: 10px;
    background: #c1c1c1;
    margin: 20px;
    font-family: Tahoma;
    text-align: center;
    float: left;
}
```



Ao utilizar essa configuração, os **elementos se deslocam para a esquerda, como se estivessem flutuando**.

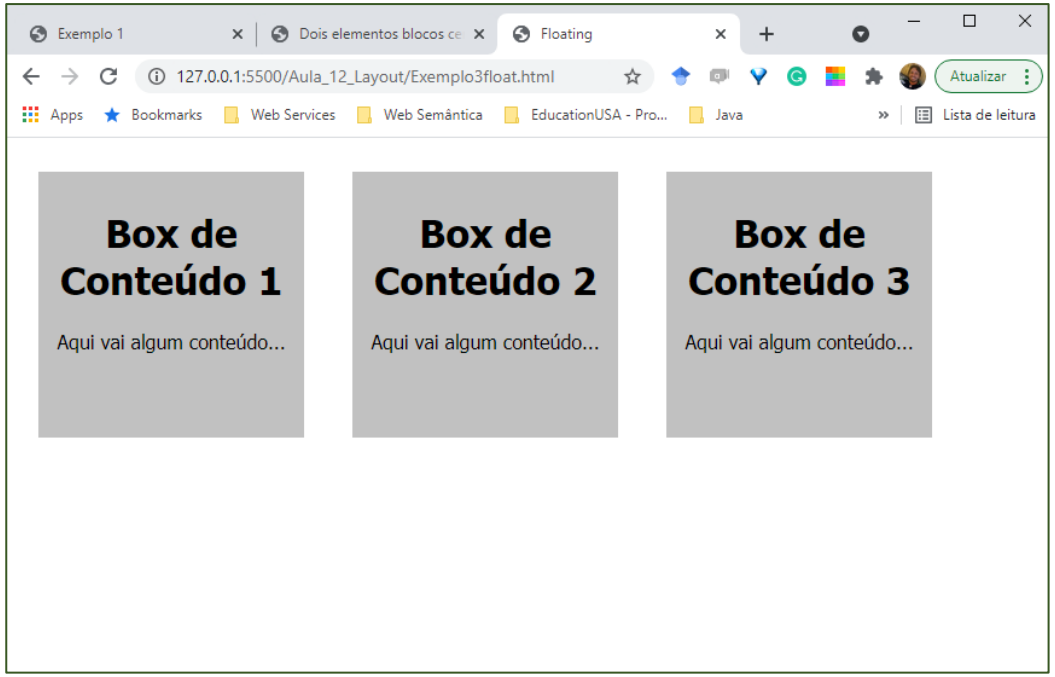
Enquanto houver espaço serão colocados a esquerda um do outro.

Comparação



Sem `float`

Exemplo



`float: left`

Propriedade

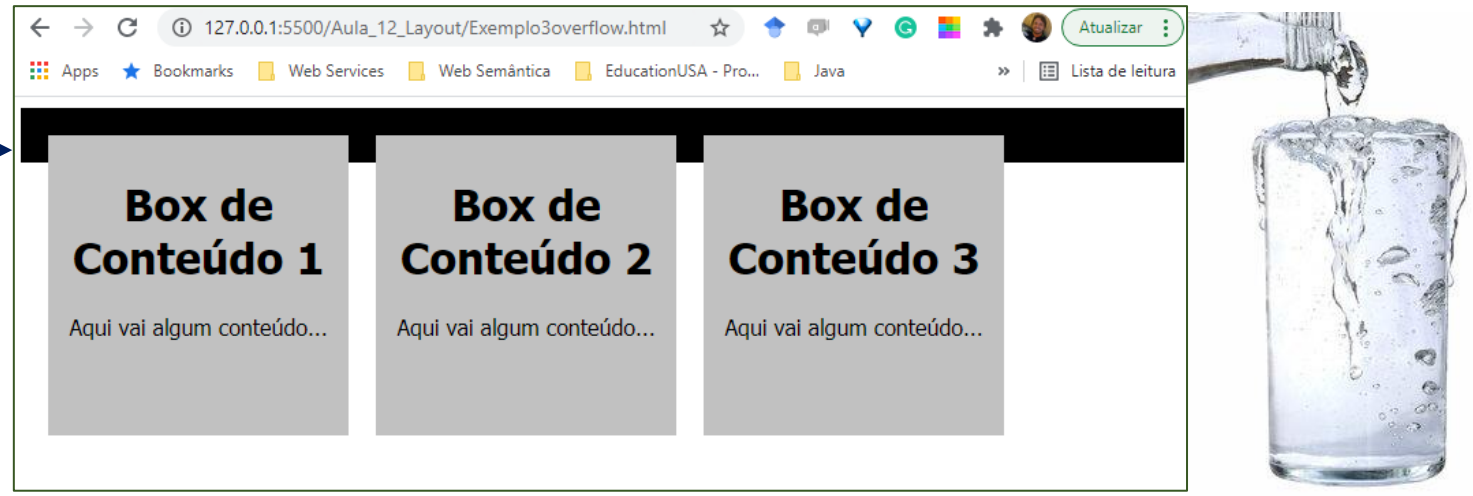
overflow

`overflow` (transbordar)

Um *box* descendente **não precisa necessariamente ter as suas dimensões confinadas aos limites de seu *box container***, ou seja, **pode ultrapassá-lo**.

Esse comportamento é denominado *overflow*.

A **`divs`** de conteúdo (quando configuradas com **`float`**), transbordam os limites da sua **`div`** pai, causando o ***overflow***.



overflow

Propriedade usada para controlar o que acontece **quando o conteúdo de um elemento excede o tamanho do contêiner** que o envolve.

Especifica se um elemento deve exibir barras de rolagem, cortar o conteúdo ou simplesmente omitir o conteúdo excedente.

Pode assumir os seguintes valores:

1) `visible`: Esse é o valor padrão. O conteúdo que excede o tamanho do contêiner será visível fora do contêiner, sem barras de rolagem.

2) `hidden`: Qualquer conteúdo que exceda o tamanho do contêiner será cortado (oculto) e não será exibido. Não haverá barras de rolagem para visualizar o conteúdo oculto.

3) `scroll`: Barras de rolagem serão exibidas para permitir que o usuário role para ver o conteúdo excedente. As barras de rolagem são sempre visíveis, independentemente de haver conteúdo excedente ou não.

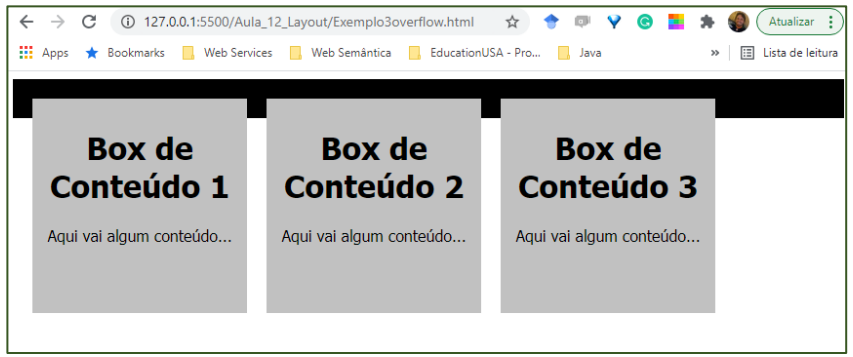
4) `auto`: Barras de rolagem serão exibidas apenas quando o conteúdo exceder o tamanho do contêiner. Se não houver conteúdo excedente, nenhuma barra de rolagem será exibida.

Modificando o código para ver o overflow

1) adicionar uma div pai

```html
<div class="principal">
    <div class="bloco">
        <h1>Box de Conteúdo 1</h1>
        <p>Aqui vai algum conteúdo...</p>
    </div>
    <div class="bloco">
        <h1>Box de Conteúdo 2</h1>
        <p>Aqui vai algum conteúdo...</p>
    </div>
</div>
```

HTML




```css
.bloco{
    width: 200px;
    height: 200px;
    padding: 10px;
    background: #c1c1c1;
    margin-right: 20px;
    font-family: Tahoma;
    text-align: center;
    float: left;
}

.principal{
    padding: 20px;
    background-color: #000;
}
```

CSS

2) Configurar a classe para visualizar o overflow

Exemplo

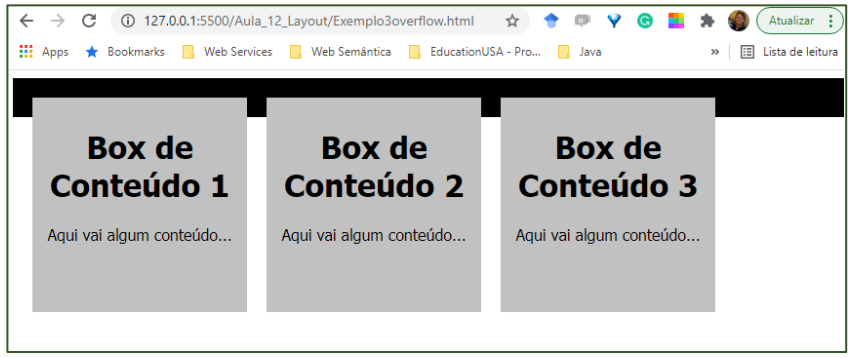
Analisando o código

Elementos **divs** da classe **bloco** estão inseridas dentro da **div** configurada com a classe **principal**.

```html
<div class="principal">
    <div class="bloco">
        <h1>Box de Conteúdo 1</h1>
        <p>Aqui vai algum conteúdo...</p>
    </div>
    <div class="bloco">
        <h1>Box de Conteúdo 2</h1>
        <p>Aqui vai algum conteúdo...</p>
    </div>
</div>
```

HTML

Estes elementos **flutuam** lado a lado com a propriedade **float: left**.



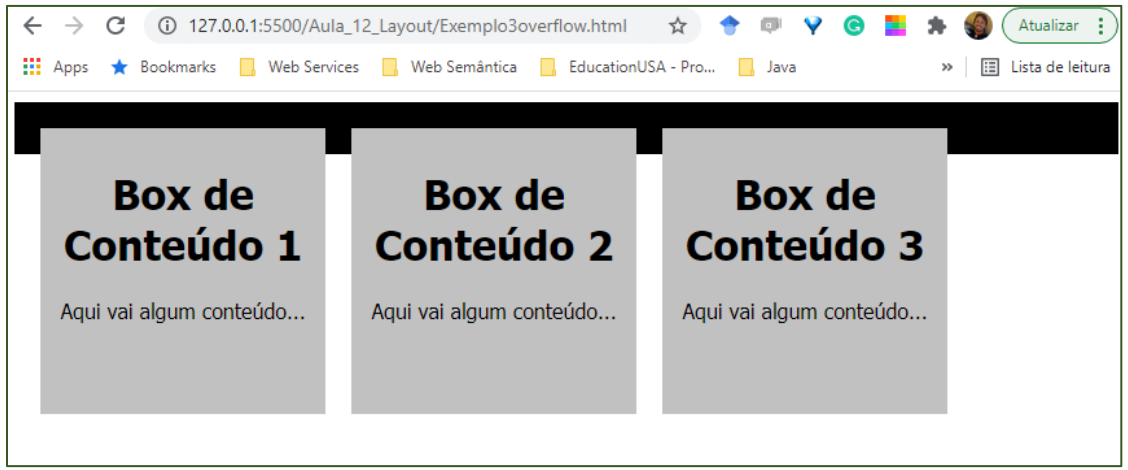
CSS

```css
.bloco{
    width: 200px;
    height: 200px;
    padding: 10px;
    background: #c1c1c1;
    margin-right: 20px;
    font-family: Tahoma;
    text-align: center;
    float: left;
}

.principal{
    padding: 20px;
    background-color: #000;
}
```

Problema com elemento pai (container). Ele não ocupa a área necessária para conter todos os filhos. Elementos se sobrepõe ao pai.

Modificando o código para contornar o problema.



Se os elementos internos (**.bloco**) forem maiores que o elemento container (**.principal**) e eles estiverem **flutuando**, eles irão **ultrapassar** o elemento container.

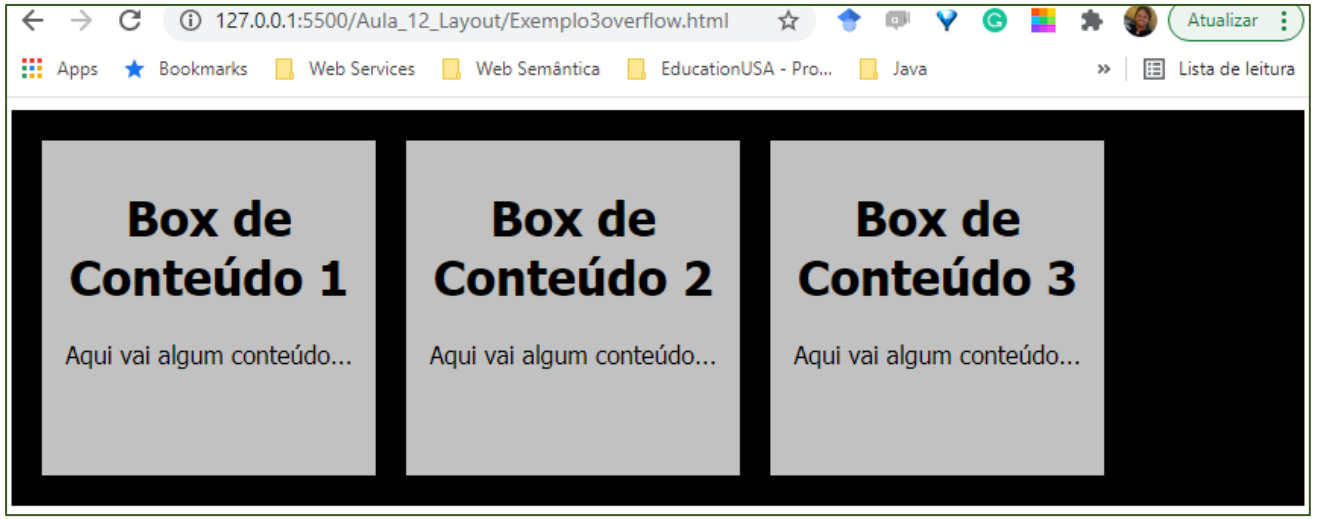
```css
.bloco{
    width: 200px;
    height: 200px;
    padding: 10px;
    background: #c1c1c1;
    margin-right: 20px;
    font-family: Tahoma;
    text-align: center;
    float: left;
}

.principal{
    padding: 20px;
    background-color: #000;
}
```

CSS

Propriedade overflow

Define o comportamento de um elemento **quando as suas dimensões são excedidas pelo conteúdo.**




```css
.bloco{
    width: 200px;
    height: 200px;
    padding: 10px;
    background: #c1c1c1;
    margin-right: 20px;
    font-family: Tahoma;
    text-align: center;
    float: left;
}

.principal{
    padding: 20px;
    background-color: #000;
    overflow: auto;
}
```

CSS

Este problema pode ser solucionado através da propriedade **overflow: auto**.

O elemento container **se ajusta ao tamanho** dos elementos internos.

Propriedade

clear

Propriedade `clear`

Usada para controlar como um elemento se comporta em relação a elementos flutuantes que estão antes dele no fluxo do documento.

Ela especifica se um elemento deve se mover abaixo (limpar) os elementos flutuantes, ou se ele deve se posicionar ao lado deles.

A propriedade `clear` pode ter os seguintes valores:

1) none: Este é o **valor padrão**. O elemento não será movido abaixo de quaisquer elementos flutuantes à esquerda ou à direita.

2) `left`: O elemento só será movido abaixo de elementos flutuantes à **esquerda**.

3) `right`: O elemento só será movido abaixo de elementos flutuantes à **direita**.

4) `both`: O elemento será movido **abaixo de ambos**, elementos flutuantes à esquerda e à direita. Isso garante que o elemento seja colocado abaixo de todos os elementos flutuantes anteriores.

Propriedade `clear`

Um elemento que tenha a propriedade **`clear`** definida não se moverá para cima e adjacente ao **`float`** (como o **`float`** deseja). Este elemento **se moverá para baixo, além do `float`.**

A barra lateral flutua para a direita, mas é mais curta que a área de conteúdo principal.

O rodapé então é obrigado a ir para o espaço disponível conforme exige o `float`.

Para corrigir esse problema, o rodapé pode ser "limpo" para garantir que permaneça abaixo de ambas as colunas flutuantes.

Propriedade `clear`

Utilizada para “limpar” o que está abaixo dos elementos flutuantes.

```html
<div class="principal">
    <div class="bloco">
        <h1>Box de Conteúdo 1</h1>
        <p>Aqui vai algum conteúdo...</p>
    </div>
    <div class="bloco">
        <h1>Box de Conteúdo 2</h1>
        <p>Aqui vai algum conteúdo...</p>
    </div>
    <footer>
        <h1>Box do rodapé com informações.</h1>
    </footer>
</div>
```

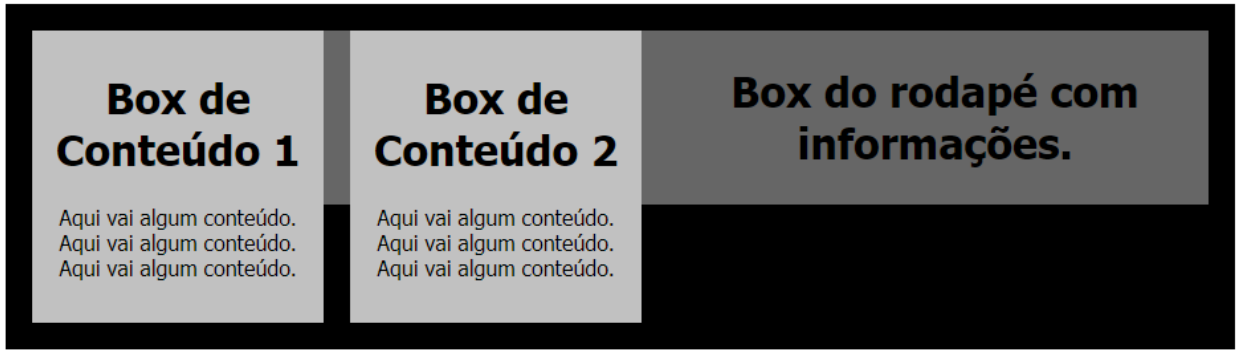
HTML

```css
/* ... Estilização da
classe .principal e .bloco */

footer{
    padding: 5px;
    background: #666;
    font-family: Tahoma;
    text-align: center;
}
```

CSS

Propriedade `clear`



Ao adicionar um novo elemento (**`footer`**) após os elementos flutuantes da classe **`.bloco`**, ocorre uma **sobreposição**.

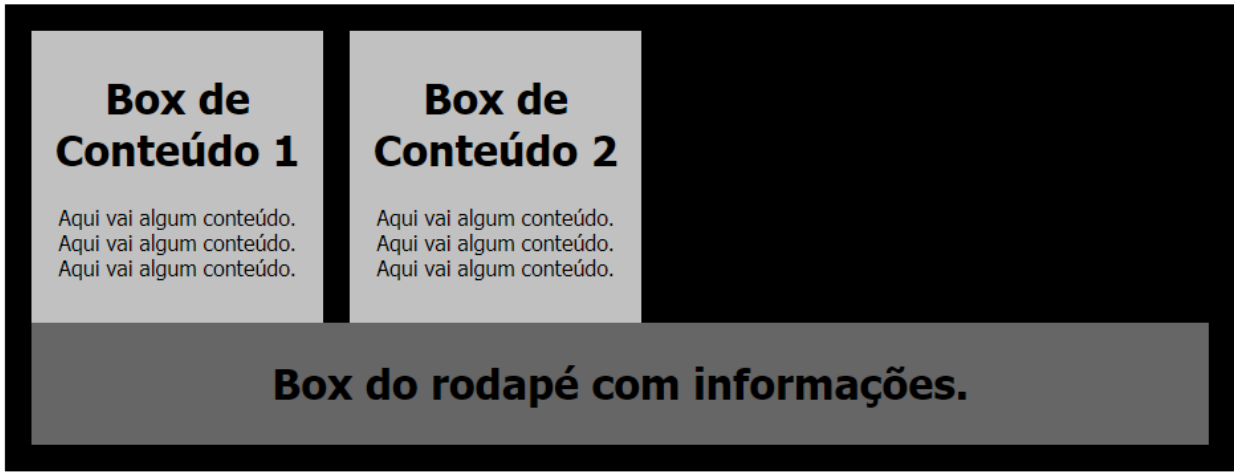
```css
/* ... Estilização da
classe .principal e .bloco */

footer{
    padding: 5px;
    background: #666;
    font-family: Tahoma;
    text-align: center;
}
```

CSS

Exemplo

Propriedade `clear`



Para solucionar este problema, deve-se usar a propriedade **`clear: left`** no elemento **`footer`**.

```css
/* ... Estilização da
classe .principal e .bloco */

footer{
    padding: 5px;
    background: #666;
    font-family: Tahoma;
    text-align: center;
    clear: left;
}
```

Trecho CSS

Exemplo

Propriedade `clear`

A propriedade **`clear`** especifica como elementos subsequentes aos elementos flutuantes devem se comportar.

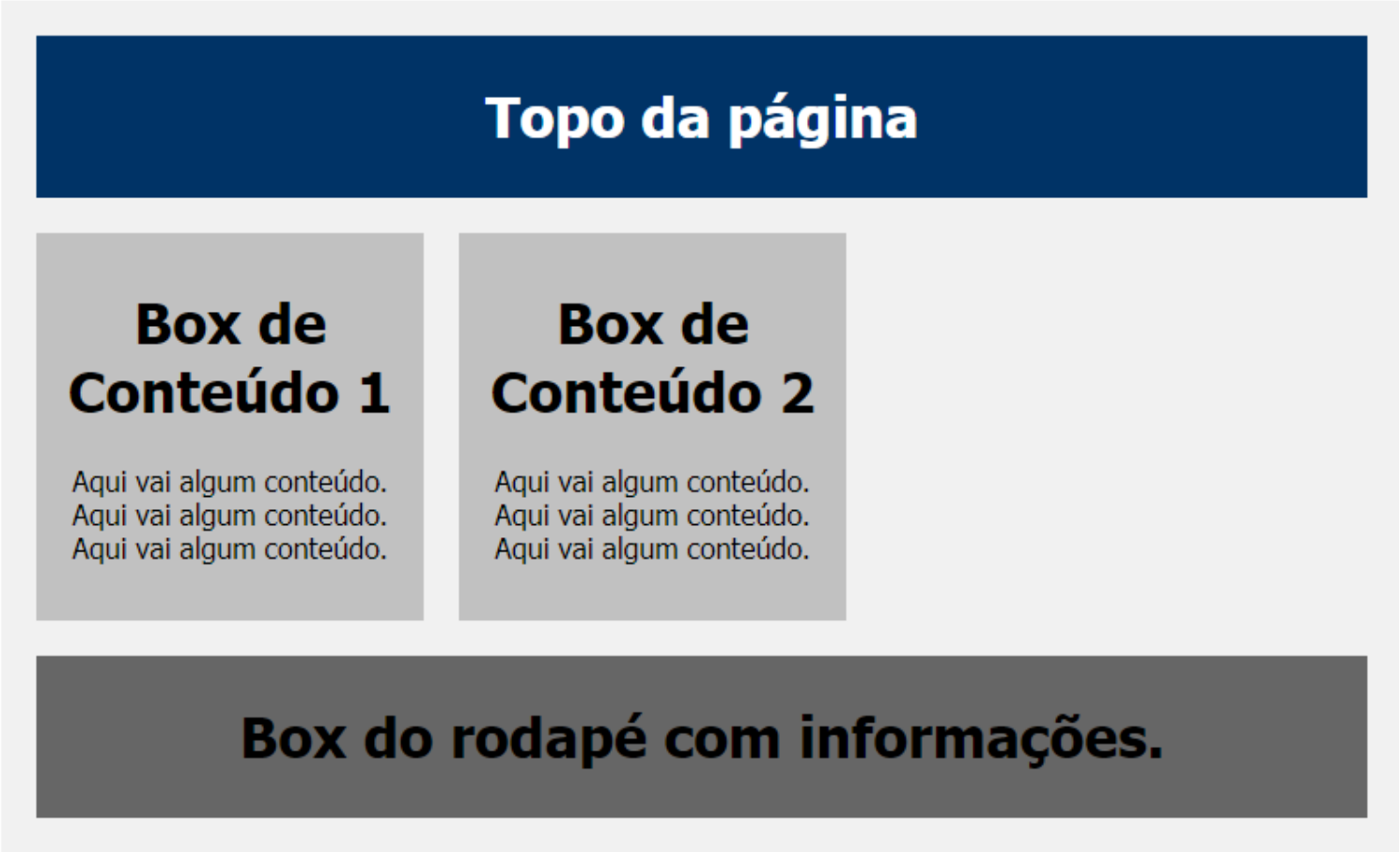
Podem ser utilizados os seguintes valores:

- **`left`**;
- **`right`**;
- **`none`**;
- **`both`** e
- **`inherit`**.

`clear` deve ser usada em concordância com o **`float`**.

Se **`float: left`**, então **`clear: left`**.

Grid Box


Topo da página
Box de Conteúdo 1
Aqui vai algum conteúdo.
Aqui vai algum conteúdo.
Aqui vai algum conteúdo.
Box de Conteúdo 2
Aqui vai algum conteúdo.
Aqui vai algum conteúdo.
Aqui vai algum conteúdo.
Box do rodapé com informações.

Curso Superior de Tecnologia em Sistemas para Internet
Desenvolvimento front-end I (DFE_1)
Rafael Cunha Cardoso
rafaelcardoso@ifsul.edu.br

Posicionamento de elementos

INSTITUTO FEDERAL
Sul-rio-grandense
Câmpus Pelotas