

CSS

Prof. Daniel Di Domenico

daniel.domenico@ifpr.edu.br



Layout

Revisão...

- O que já sabemos:
 - Utilizar CSS de modo inline, incorporado ou em arquivo externo
 - Seletores
 - Pseudo-classes e pseudo-elementos
 - Modelo de caixa



Dimensionar e posicionar os elementos

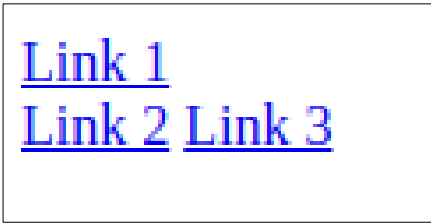
- Como podemos fazer para posicionar e dimensionar os elementos:
 - O CSS possui funcionalidades para realizar tais configurações em uma página HTML
 - Desta maneira, é possível utilizar melhor o espaço disponível para exibir o conteúdos



Elementos HTML **div** e **span**

- **<div>** e **** são utilizados para estruturar páginas HTML
 - Agrupam outros elementos, servindo como *containers*
 - Não adicionam modificações aparentes à página HTML
 - Diferença:
 - **div**: é um elemento de bloco (elementos após ele serão exibidos abaixo dele)
 - **span**: é um elemento de linha (elementos após ele serão exibidos ao lado dele)

```
<div>  
  <a href="#">Link 1</a>  
</div>  
<span>  
  <a href="#">Link 2</a>  
</span>  
<span>  
  <a href="#">Link 3</a>  
</span>
```



Link 1
Link 2 Link 3

Propriedades de tamanho

- **width**: define a largura do componente
 - **min-width** e **max-width**: largura mínima e máxima
- **height**: define a altura do componente
 - **min-height** e **max-height**: altura mínima e máxima

```
<div class="tamanho">  
  <a href="#">Link 1</a>  
  <a href="#">Link 2</a>  
</div>
```

```
.tamanho {  
  width: 200px;  
  min-height: 80px;  
  background-color: lightblue;  
}
```



[Link 1](#) [Link 2](#)

Propriedades de posicionamento

- **position**: permite o posicionamento de duas formas:
 - absolute: estático, através das coordenadas **top**, **right**, **bottom** e **left** em pixels (px) considerando a área da página
 - relative: relativo, também através das coordenadas **top**, **right**, **bottom** e **left** em pixels (px), porém considerando a posição que o elemento seria originalmente desenhado
- Utilizando da propriedade **position**, os elementos podem se sobrepor
- **Teste os estilos abaixo em uma página HTML:**

```
p.fixo {  
    position: absolute;  
    top: 400px;  
    left: 100px;  
}
```

```
p.relATIVO {  
    position: relative;  
    top: 80px;  
    left: 150px;  
}
```

Propriedades de posicionamento

- **float**: permite posicionar (flutuar) um elemento, possibilitando que outros elementos fiquem em torno dele
 - A flutuação é apenas horizontal (direita ou esquerda)
 - left: flutua o elemento para a esquerda
 - right: flutua o elemento para a direita

Sem usar
float



Texto de parágrafo com bla blá blá blá blá blá blá
Texto de parágrafo com bla blá blá blá blá blá blá
Texto de parágrafo com bla blá blá blá blá blá blá
Texto de parágrafo com bla blá blá blá blá blá blá

```
.imagemFlutuante {  
    float: left;  
}
```

Utilizando
float



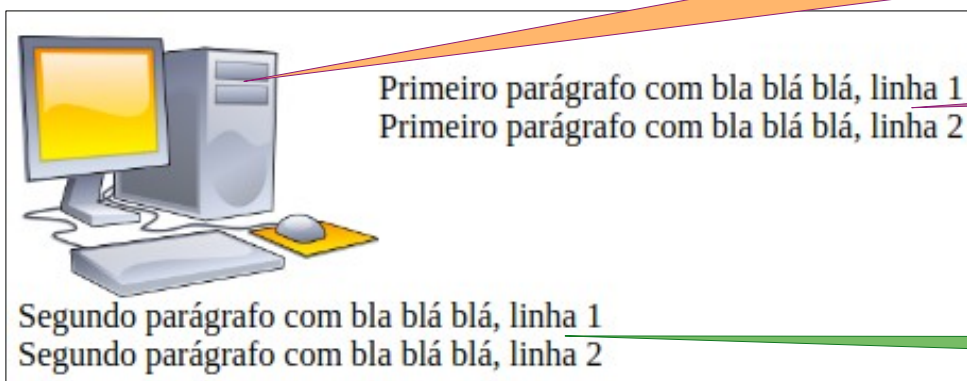
Texto de parágrafo com bla blá blá blá blá blá blá
Texto de parágrafo com bla blá blá blá blá blá blá
Texto de parágrafo com bla blá blá blá blá blá blá
Texto de parágrafo com bla blá blá blá blá blá blá

Propriedades de posicionamento

- **clear**: impede que um elemento tenha outros elementos flutuantes ao seu lado (direita ou esquerda)
- A propriedade **clear** pode impedir a exibição de elementos flutuantes:
 - *left*: à esquerda do elemento
 - *right*: à direita do elemento
 - *both*: em ambos os lados do elemento

Imagem com
float para a
esquerda (**left**)

Sem usar
clear



```
.parClear {  
  clear: left;  
}
```

Utilizando
clear

```
<p> Primeiro parágrafo com bla blá blá, linha 1 <br>  
  Primeiro parágrafo com bla blá blá, linha 2 </p>  
<p class="parClear"> Segundo parágrafo com bla blá blá, linha 1 <br>  
  Segundo parágrafo com bla blá blá, linha 2 </p>
```


Layout

- Pode-se criar layouts agrupando os elementos dentro de tags <div>
 - No arquivo CSS, é possível determinar o posicionamento destas DIVs
- Exemplo de layout:

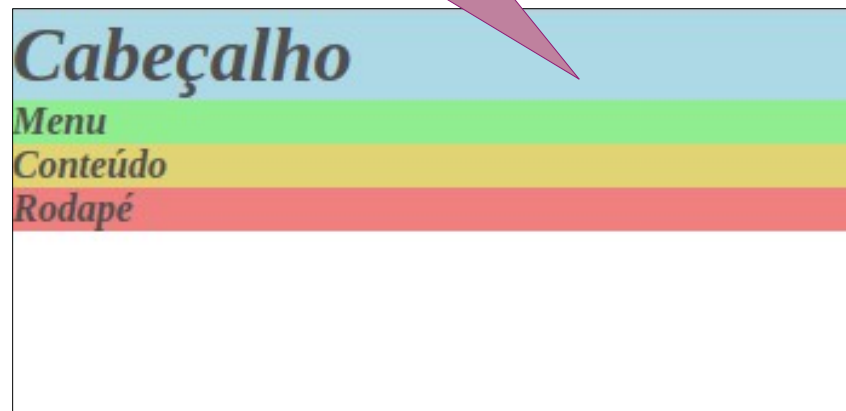


Layout

- Com a mesma página HTML, é possível definir um layout de diversas formas:

```
<div id="container">  
  <div id="cabecalho">  
    <h1>Cabeçalho</h1>  
  </div>  
  <div id="menu">  
    <h4>Menu</h4>  
  </div>  
  <div id="conteudo">  
    <h4>Conteúdo</h4>  
  </div>  
  <div id="rodape">  
    <h4>Rodapé</h4>  
  </div>  
</div>
```

CSS para alterar
apenas a cor de fundo
das DIVs



Layout

- Posicionamento fixo:

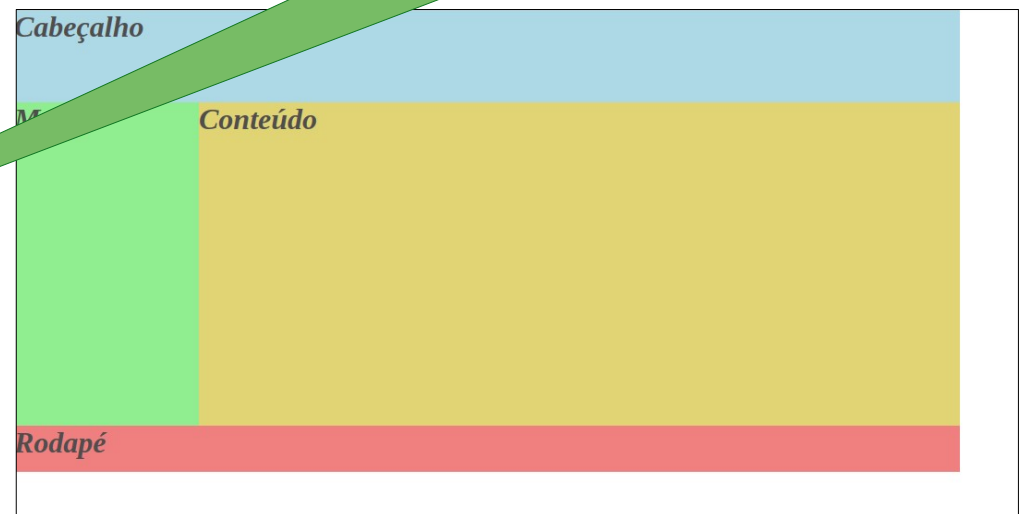
```
#cabecalho {  
  height: 100px;  
  width: 1024px;  
}
```

```
#menu {  
  height: 350px;  
  width: 200px;  
}
```

```
#conteudo {  
  position: absolute;  
  left: 200px;  
  top: 100px;  
  height: 350px;  
  width: 824px;  
} /*Continua...*/
```

```
#rodape {  
  height: 50px;  
  width: 1024px;  
}
```

Utilizado
position, left e top para
posicionar a
DIV #conteudo



Layout

- Posicionamento fixo com **float**:

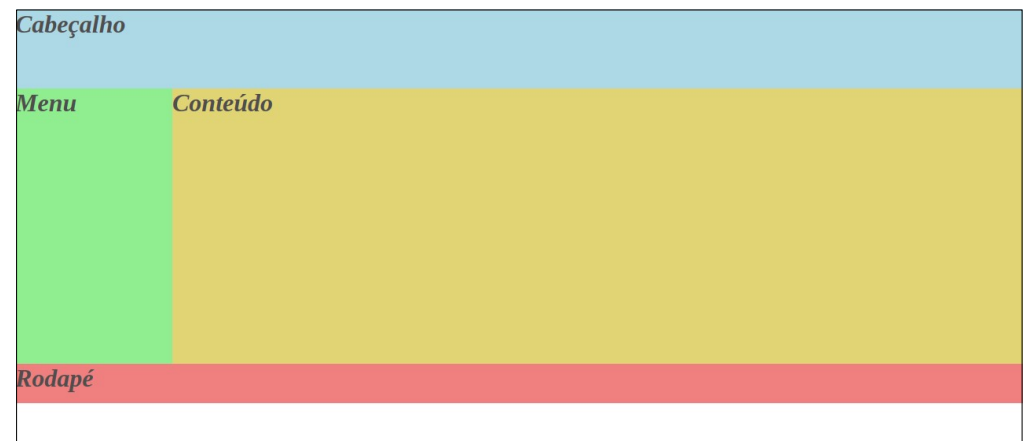
```
#cabecalho {  
  height: 100px;  
  width: 100%;  
}
```

```
#menu {  
  height: 350px;  
  width: 200px;  
  float: left;  
}
```

```
#conteudo {  
  height: 350px;  
  width: 100%;  
} /*Continua...*/
```

```
#rodape {  
  height: 50px;  
  width: 100%;  
}
```

Utilizado **float: left**
para posicionar o
menu a esquerda
da DIV #conteudo

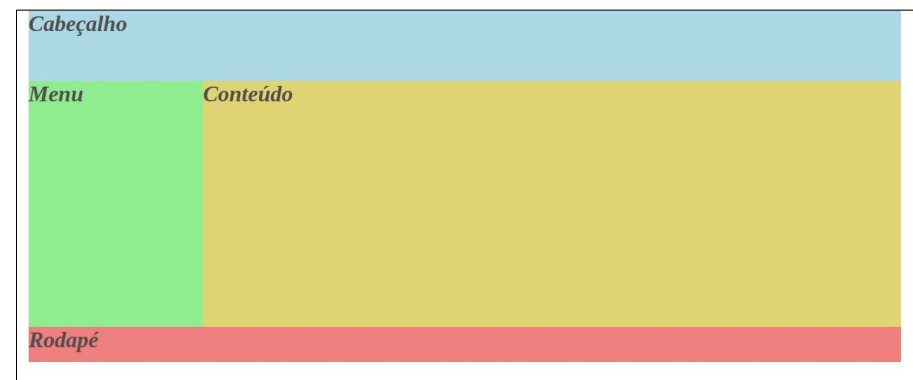


Layout

- Posicionamento com **CSS Grid**:

```
#container {  
  width: 95%;  
  margin: 0 auto;  
  display: grid;  
  grid-template-columns: 20% 80%;  
  grid-template-rows: 100px 350px 50px;  
  grid-template-areas: "divCabec divCabec"  
                      "divMenu divConteudo"  
                      "divRodape divRodape";  
}  
  
#cabecalho {  
  grid-area: divCabec;  
}  
  
#menu {  
  grid-area: divMenu;  
} /*Continua...*/
```

```
#conteudo {  
  grid-area: divConteudo;  
}  
  
#rodape {  
  grid-area: divRodape;  
}
```

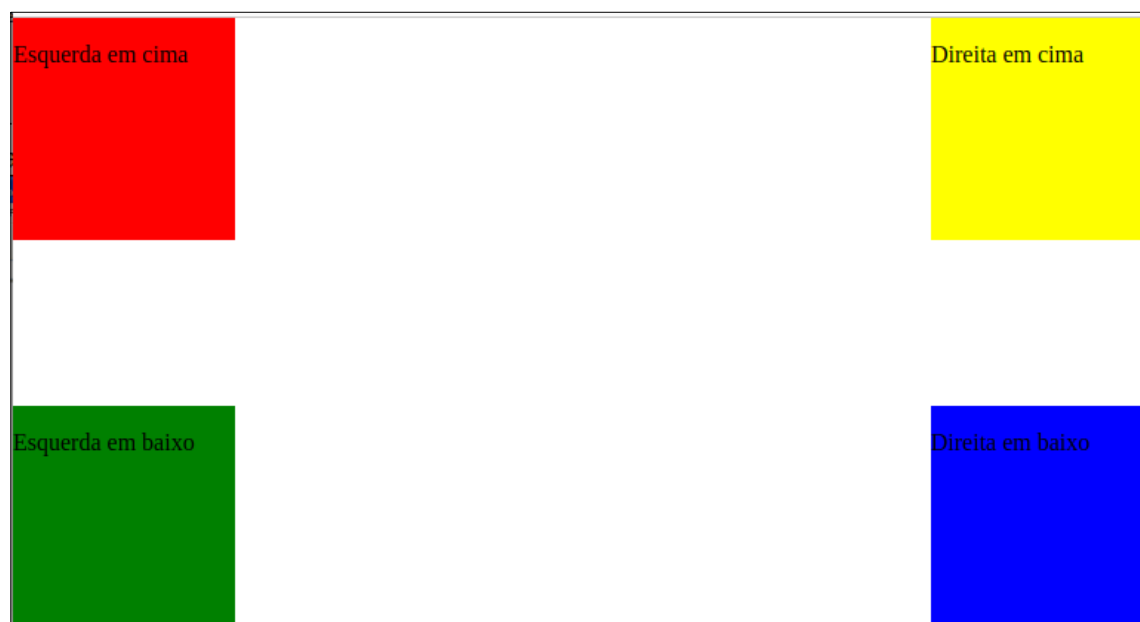


Layout

- Passo a passo para um layout com **CSS Grid**:
 - Na DIV que será o pai dos demais elementos, setar as propriedades:
 - **display: grid** (para ativar o CSS Grid)
 - **grid-template-columns**: define a quantidade e tamanho das colunas (px ou %)
 - **grid-template-rows**: define a quantidade e o tamanho de linhas (px)
 - **grid-template-areas**: define qual elemento será exibido em cada parte da grid
 - Cada valor referencia outra DIV onde a propriedade **grid-area** foi declarada
 - Nas demais DIVs que precisam ser posicionadas dentro da DIV pai, setar a propriedade **grid-area**
 - **Atenção:** o valor definido para **grid-area** deve referenciar um valor já definido na propriedade **grid-template-areas**
 - Mais detalhes: https://www.w3schools.com/css/css_grid.asp

Exercício

- Crie um layout que exibe 4 DIVs da seguinte maneira:
 - Dica: utilizar posicionamento fixo



Exercício

- Crie um layout com CSS Grid, exibindo as DIVs da seguinte forma:

