



Sobre Este Curso

Público Alvo

Programadores que têm como objetivo atuar como Desenvolvedor Front-End e criar interfaces para produtos digitais.

Pré-Requisitos

Conhecimentos de Lógica de Programação.

Índice

SOBRE ESTE CURSO	I
PÚBLICO ALVO	I
PRÉ-REQUISITOS	I
ÍNDICE.....	I
COPYRIGHT	IV
EQUIPE.....	IV
HISTÓRICO DAS EDIÇÕES	IV
CAPÍTULO 01 – FUNDAMENTOS	5
O QUE É CSS	6
REGRA CSS	6
COMENTÁRIOS DE REGRAS CSS	7
CAPÍTULO 02 – SELETORES	8
O QUE SÃO SELETORES.....	9
SELETOR ELEMENTO	9
SELETOR ID.....	9
SELETOR CLASSE	9
SELETOR PSEUDO CLASSES	10
SELETOR PSEUDO ELEMENTOS	10
SELETOR UNIVERSAL.....	11
SELETOR ATRIBUTO	11
SELETOR COMPOSTO	13
CAPÍTULO 03 – FOLHAS DE ESTILO	15
TIPOS DE ESTILIZAÇÃO DE DOCUMENTOS	16
ESTILOS INLINE	16
ESTILOS INCORPORADOS	16
FOLHAS DE ESTILO LIGADAS.....	17
FOLHAS DE ESTILO IMPORTADAS	18
CAPÍTULO 04 – EFEITO CASCATA	20
EFEITO CASCATA	21
ORDEM DA CASCATA	21
ESPECIFICIDADE DE UM SELETOR	21
CÁLCULO DE ESPECIFICIDADE.....	22
CAPÍTULO 05 – BOX MODEL	23
O QUE É BOX MODEL.....	24

	Anotações



PROPRIEDADES DO BOX MODEL	24
BOX-SIZING	28
TIPO DE ELEMENTOS	28
CAPÍTULO 06 – TIPOGRAFIA.....	30
FONTES SEGURAS.....	31
FAMÍLIAS DE FONTES	31
ESPAÇAMENTO E ALINHAMENTO DE TEXTOS	32
EFEITOS DE TEXTOS	34
ESTILOS DE FONTE.....	36
CAPÍTULO 07 – CORES, BACKGROUNDS E LINKS.....	39
DEFINIÇÕES DE CORES DE PRIMEIRO PLANO	40
DEFINIÇÕES DE PLANO DE FUNDO - BACKGROUND	41
ESTILIZAÇÃO DE LINKS	42
CAPÍTULO 08 - LISTAS	44
PROPRIEDADE LIST-STYLE-TYPE	45
PROPRIEDADE LIST-STYLE-IMAGE	45
PROPRIEDADE LIST-STYLE-POSITION.....	45
CAPÍTULO 09 - POSICIONAMENTO	46
PROPRIEDADE FLOAT	47
PROPRIEDADE CLEAR	48
POSICIONAMENTOS.....	49
FLEXBOX	49
CAPÍTULO 10 – ANIMAÇÕES E TRANSIÇÕES NO CSS.....	53
ANIMAÇÕES E TRANSIÇÕES	54
TRANSITION.....	54
ANIMATION	60
APÊNDICE A	66
APÊNDICE B	69

Anotações	

	Anotações



Copyright

As informações contidas neste material se referem ao curso de **CSS3** e estão sujeitas as alterações sem comunicação prévia, não representando um compromisso por parte do autor em atualização automática de futuras versões.

A **Apex** não será responsável por quaisquer erros ou por danos acidentais ou consequenciais relacionados com o fornecimento, desempenho, ou uso desta apostila ou os exemplos contidos aqui.

Os exemplos de empresas, organizações, produtos, nomes de domínio, endereços de e-mail, logotipos, pessoas, lugares e eventos aqui representados são fictícios. Nenhuma associação a empresas, organizações, produtos, nomes de domínio, endereços de e-mail, logotipos, pessoas, lugares ou eventos reais é intencional ou deve ser inferida.

A reprodução, adaptação, ou tradução deste manual mesmo que parcial, para qualquer finalidade é proibida sem autorização prévia por escrito da **Apex**, exceto as permitidas sob as leis de direito autoral.

Equipe

Conteúdos

- Felipe de Oliveira e Ralf Lima

Diagramação

- Fernanda Pereira

Revisão

- Fernanda Pereira

Histórico das Edições

Edição	Idioma	Edição
1ª	Português	Julho de 2017
2ª	Português	Julho de 2019

© Copyright 2017 **Apex**. Desenvolvido por DJF Treinamentos e Consultoria e licenciado para Apex Ensino.

Anotações	

Capítulo 01 – Fundamentos



Objetivos:

Neste capítulo você irá aprender:

- O que é CSS
- Regra CSS
- Comentários CSS

	Anotações



O que é CSS

Folhas de estilo em cascata ou no inglês “*Cascading Style sheet*” (CSS), é uma linguagem de marcação utilizada para a estilização de documentos html, e tem como objetivo separar a estilização de apresentação do documento da informação contida no próprio documento.

Com o uso do CSS podemos adicionar fontes, cores, margens, bordas entre outros, a elementos contidos dentro de documentos web.

Regra CSS

Regra CSS é a unidade básica utilizada para a estilização de um documento através do CSS. Uma regra CSS é composta por um seletor e uma ou mais declarações de propriedades/valores.

Exemplo:

```
seletor {  
  propriedade:valor;  
}
```

Onde:

Seletor: É o alvo da regra css, o componente que se deseja estilizar;

Propriedade: É o atributo/característica do componente que desejamos configurar;

Valor: É a qualificação do atributo;

Exemplo:

```
h1{  
  color: red;  
  Font-family: “Times New Roman”;  
}
```

Anotações	

Comentários de regras CSS

Em folhas de estilo em cascata podemos adicionar comentários para melhor organização e manutenção.

Para adicionarmos um comentário devemos utilizar os caracteres `/*` para iniciar o bloco de comentários e em seguida os caracteres `*/` para fechar o bloco de comentários.

Exemplo:

```
/* Este é um comentário válido em uma folha de estilos */
```

	Anotações

Capítulo 02 – Seletores



Objetivos:

Neste capítulo você irá aprender:

- O que são seletores
- Seletor elemento
- Seletor id
- Seletor classe
- Seletor de pseudo-classes
- Seletor de pseudo-elementos
- Seletor Universal
- Seletor atributo
- Seletor composto

Anotações	

O que são seletores

Seletor CSS é o nome dado ao alvo da regra CSS que se deseja estilizar. Os seletores CSS são definidos conforme seu tipo, de acordo com as informações abaixo:

Seletor elemento

Este seletor recebe este nome porque utilizamos como seletor o próprio nome da tag html que queremos estilizar.

Exemplo:

```
p {
  color: green;
}
```

Seletor id

Seletor que contém o nome de um identificador único de uma tag HTML. Este tipo de seletor é definido colocando-se o carácter '#' antes do identificador.

Exemplo:

```
/* documento html */
<h2 id="subtitulo">Subtitulo aqui</h2>
```

```
/*Folha de estilo CSS*/
```

```
#subtitulo{
  font-size:26px;
  color: #cccccc;
}
```

Seletor classe

Seletor de classe é o tipo de seletor que utilizamos quando utilizamos o nome de uma classe definida como atributo de um componente html. Para referenciarmos a classe em uma folha de estilos devemos colocar o carácter '.' antes do nome da classe.

	Anotações



Exemplo:

```
/* documento html */
```

```
<tr class="impar">Linha impar de uma tabela</tr>
```

```
/*Folha de estilo CSS*/
```

```
.impar{  
  background-color: #cccccc;  
}
```

Seletor pseudo classes

Pseudo-classes são utilizadas para a definição/marcação de elementos quando estes possuem um estado específico.

A marcação de uma pseudo-classe é feita pela junção de um seletor (componente, id ou classe) mais adição de ':' seguido do nome da pseudo-classe que se deseja utilizar.

Por exemplo:

Desejamos aumentar a fonte de um parágrafo quando o cursor do mouse estiver sobre o respectivo parágrafo. Para realizarmos este tipo de estilização podemos utilizar a pseudo-classe 'hover'.

Exemplo:

```
p:hover{  
  font-size: 30px;  
}
```

Seletor pseudo elementos

Pseudo-elementos são utilizados para estilização de certas partes de um elemento. A marcação de um pseudo-elemento é feita pela junção de um seletor (componente, id ou classe) mais adição de ':' seguido do nome do pseudo-elemento que se deseja utilizar.

Anotações	

Por exemplo:

Desejamos alterar a cor de fundo apenas da primeira linha de um parágrafo. Para realizarmos este tipo de estilização podemos utilizar o pseudo-elemento 'first-line'.

Exemplo:

```
p:first-line{
background-color: #cccccc;
}
```

Seletor universal

O seletor universal é utilizado para aplicar uma determinada estilização a todos os elementos contidos no documento. O símbolo do seletor universal é '*'.

Exemplo:

```
*{
color: blue;
}
```

OBS. No exemplo acima todos os componentes terão sua cor definida para azul.

Seletor atributo

Este seletor utiliza a combinação de um determinado atributo/valor de um componente para realizar a estilização.

E[attr]	Um elemento 'E' que contenha o atributo 'attr'.
E[attr=val]	Um elemento 'E' que contenha o valor 'val' para o atributo 'attr'.
E[attr~=val]	Um elemento 'E' que contenha no atributo 'attr' uma lista de valores separados por espaço e que um dos valores seja 'val'.
E[attr =val]	Um elemento 'E' que contenha no atributo 'attr' uma lista de valores separados por hífen em que a primeira palavra é 'val'.

	Anotações



Exemplos:

```
/* documento html */
```

```
<p title="impar">meu parágrafo</p>
```

```
<input type="text">
```

```
<p class="impar colorido">meu parágrafo</p>
```

```
<p class="colorido-on">meu parágrafo</p>
```

```
/*Folha de estilo CSS*/
```

```
p[title]{  
  background-color: red;  
}
```

```
input[type=text]{  
  background-color: blue;  
}
```

```
p[class~=colorido]{  
  background-color: blue;  
}
```

```
p[class|=colorido]{  
  background-color: yellow;  
}
```

Anotações	

Seletor composto

Seletor composto é o seletor formado pela união de dois ou mais seletores simples. Podemos dividir os seletores compostos em duas subcategorias:

Seletor descendente

Utilizado para indicar elementos que estão dentro de uma ordem pai-filho dentro da estrutura do 'Data Object Model' (DOM) mas que no entanto não necessitam de uma ordem direta de ligação.

Exemplo:

```
.container p{
  background-color: blue;
}
```

A regra acima será aplicada a qualquer parágrafo que esteja contido dentro de '.container' mesmo que em sub níveis diferentes. Através da regra acima todos os parágrafos do documento abaixo serão atingidos:

```
/* documento html */
<div class="container">
  <p>Parágrafo filho direto</p>
  <div>
    <p>Parágrafo filho indireto também será atingido pelo alvo</p>
  </div>
</div>
```

Seletor filho

Utilizado quando desejamos atingir um elemento descendente direto na árvore do DOM. Para selecionarmos um filho direto devemos separar os seletores compostos por um sinal de '>'

Exemplo:

```
.container2 > p{
  background-color: green;
}
```

	Anotações



A regra acima será aplicada apenas ao parágrafo filho direto de `.container` , não afetando os parágrafos que estejam em sub níveis diferentes. Através da regra acima apenas o primeiro parágrafo do documento abaixo será atingido:

```
/* documento html */  
<div class="container2">  
<p>Parágrafo filho direto</p>  
<div>  
<p>Parágrafo filho indireto não será atingido pelo alvo</p>  
</div>  
</div>
```

Seletor irmão adjacente

Utilizado quando queremos selecionar um elemento que esteja no mesmo nível de hierarquia na árvore DOM. Para selecionarmos um irmão adjacente devemos separar os seletores compostos por um sinal de `'+'`

Exemplo:

```
.container3 p + h2{  
  color: green;  
}
```

A regra acima será aplicada apenas ao h2 irmão subjacente do parágrafo, não afetando os h2 que não possuam como irmãos um parágrafo. Através da regra acima apenas o primeiro h2 do documento abaixo será atingido:

```
/* documento html */  
<div class="container3">  
<p>Parágrafo filho direto</p>  
<h2>Titulo atingido</h2>  
<div>  
<p>Parágrafo filho indireto não será atingido</p>  
</div>  
<h2>Este título não será atingido</h2>  
</div>
```

Anotações	

Capítulo 03 – Folhas de Estilo



Objetivos:

Neste capítulo você irá aprender:

- Tipos de estilização de documentos
- Estilo inline
- Estilos incorporados
- Folhas de estilo ligadas
- Folhas de estilo importadas

	Anotações



Tipos de estilização de documentos

Existem 4 formas básicas de vincularmos estilos a um documento html:

Estilos inline

Estilos inline são vinculados através da propriedade style pertencente a todos os componentes html. Os valores atribuídos à propriedade style serão utilizados para a estilização do componente.

Exemplo:

```
/* documento html */  
<h2 style="color:blue; background-color:#ccc"> Meu título</h2>
```

Estilos incorporados

Outra forma de adicionarmos estilos a um documento html é configurando-os dentro da tag <style></style>. Esta tag deve ser definida dentro do componente head.

Exemplo:

```
/* documento html */  
<!DOCTYPE html>  
<html>  
<head>  
  <style>  
    body{  
      margin: 0 0 0 0;  
    }  
    h2{  
      color:blue;  
      background-color:#ccc;  
    }  
  
    .container{
```

Anotações	

```

        margin-left: 10px;
    }
</style>
</head>
<body>
</body>
</html>

```

Folhas de estilo ligadas

Folhas de estilo ligadas são criadas dentro de arquivos com a extensão '.css' e posteriormente ligadas ao documento onde serão utilizadas.

Exemplo:

```

/* documento meu-estilo.css */
body{
    margin: 0 0 0 0;
}
h2{
    color:blue;
    background-color:#ccc;
}

.container{
    margin-left: 10px;
}

```

```

/* documento html */
<!DOCTYPE html>
<html>
<head>
    <link rel="stylesheet" href="meu-estilo.css">
</head>

```

	Anotações



```
<body>
</body>
</html>
```

Folhas de estilo importadas

Folhas de estilo importadas são folhas de estilo criadas dentro de um arquivo com a extensão “.css” e posteriormente importados para o documento através da propriedade import dentro da tag style.

Exemplo:

```
/* documento meu-estilo.css */
```

```
body{
  margin: 0 0 0 0;
}
h2{
  color:blue;
  background-color:#ccc;
}
```

```
.container{
  margin-left: 10px;
}
```

```
/* documento html */
```

```
<!DOCTYPE html>
<html>
<head>
  <style>
    @import url('meu-estilo.css');
  </style>
```

Anotações	

</head>

<body>

</body>

</html>

	Anotações

Capítulo 04 – Efeito Cascata



Objetivos:

Neste capítulo você irá aprender:

- Efeito cascata
- Ordem da cascata
- Especificidade de um seletor
- Cálculo de especificidade

Anotações	

Efeito Cascata

As folhas de estilo podem ser originadas de 3 fontes diferentes:

- autor
- usuário
- agente de usuário (navegador).

O autor cria folhas de estilo e as disponibiliza em seus documentos.

O usuário pode criar folhas de estilo personalizadas e configurá-las no navegador.

Já os navegadores incorporam folhas de estilo interna utilizada para estilização padrão de um documento HTML pelo navegador.

Todas estas estilizações possuem uma ordem de carregamento, onde as últimas sobrescrevem as primeiras.

Ordem da cascata

1. Folha de estilo padrão do navegador
2. Folha de estilo do usuário
3. Folha de estilo do autor
 - a) Estilos ligados ou importados
 - b) Estilos definidos dentro da tag <style>
 - c) Estilos inline (definidos dentro da propriedade style)
4. Declarações do autor com !important;
5. Declarações do usuário com !important;

Especificidade de um seletor

A especificidade de um seletor determina a rapidez com que o navegador irá encontrar e estilizar o componente com o alvo definido pelo seletor. De modo geral, a ordem de especificidade segue a seguinte regra:

1. Estilos inline;
2. Ids;
3. Classes, pseudo-classes e atributos;
4. Elementos e pseudo elementos;

	Anotações



Cálculo de especificidade

Para realizarmos o cálculo de especificidade de um alvo específico devemos realizar a contagem de todos os elementos que formam o alvo, agrupando-os conforme a listagem de precedência definida anteriormente.

Exemplo:

Alvo = p

estilos inline = 0

ids = 0

classes/pseudo-classes/atributo s= 0

elementos/pseudo-elementos = 1

calculo = 0 0 0 1

Alvo = #nav .titulo p

estilos inline = 0

ids = 1

classes/pseudo-classes/atributo s= 1

elementos/pseudo-elementos = 1

calculo = 0 1 1 1

Com base nos exemplos acima os alvo *p* possui um valor hipotético de 0001 enquanto o alvo *#nav .titulo p* possui um valor hipotético de 0111 fazendo com se seja muito mais eficiente a busca pelo segundo alvo em comparação com o primeiro.

Anotações	

Capítulo 05 – Box Model



Objetivos:

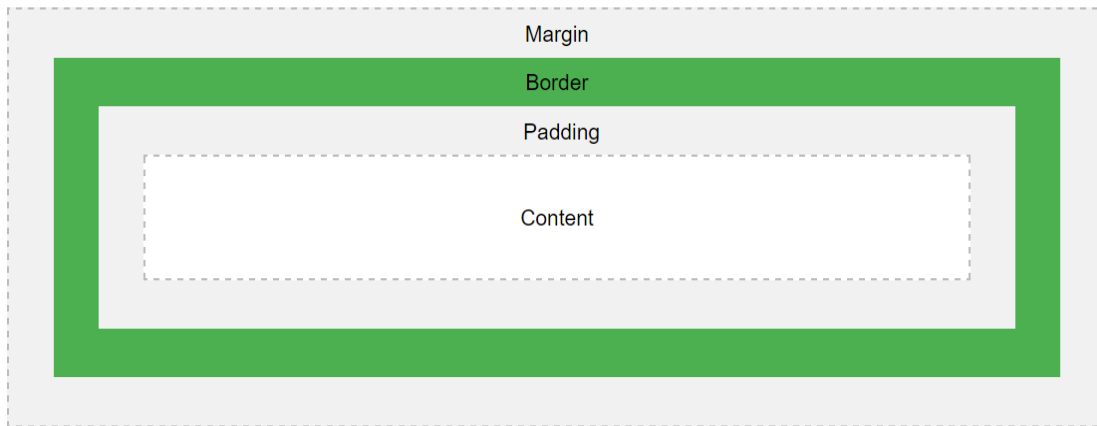
Neste capítulo você irá aprender:

- O que é Box Model
- Propriedades do Box Model
- Box-sizing
- Tipos de elementos

	Anotações

O que é Box Model

Todos os elementos HTML são colocados no DOM dentro de uma caixa que usualmente chamamos de 'Box Model'. Esta caixa tem as seguintes propriedades conforme a figura abaixo:



Propriedades do Box Model

Margin: Margem do element. Esta é a área fora da borda do elemento. Esta região é transparente.

Border: Borda do element. Esta área pode ser preenchida com uma cor específica.

Padding: Espaçamento entre a borda e o conteúdo do elemento.

Content: Área do conteúdo do elemento propriamente dito. Quando definimos um comprimento e uma largura para um determinado elemento estamos na realidade informando as dimensões de comprimento e largura do 'content'.

Através do CSS podemos utilizar as propriedades do box model para estilizarmos um componente.

Propriedade width

Utilizada para definirmos a largura de um elemento. Esta propriedade, como dito anteriormente afeta a largura do 'content'.

Propriedade height

Utilizada para definirmos a altura de um elemento. Esta propriedade afeta a altura do 'content'.

Anotações	

Propriedade margin

Utilizada para definirmos o tamanho da margem do element. Podemos utilizá-la indicando qual das margens que queremos configurar através das seguintes propriedades:

margin-top	utilizada para configurar a margem no topo do elemento;
margin-left	utilizada para configurar a margem na lateral esquerda do elemento;
margin-right	utilizada para configurar a margem na lateral direita do elemento;
margin-bottom	utilizada para configurar a margem inferior do elemento

Exemplo:

```
p{
  margin-top:20px;
  margin-left: 10px;
  margin-right: 10px;
  margin-bottom: 5px;
}
```

Também podemos passar um valor para todas as margens, utilizando a sintaxe mais curta. Nesta sintaxe as margens são configuradas utilizando-se a sequencia top, right, bottom, left, ou seja, através do sentido horário.

Exemplo:

```
p{
  margin: 20px 10px 5px 10px;
}
```

	Anotações



Propriedade border

Utilizada para definirmos a espessura, o estilo e a cor das bordas do elemento. Cada uma destas características do elemento podem ser configuradas separadamente.

border-width: configura a espessura da borda;

border-color: configura a cor da borda;

border-style: configura o estilo da borda;

Exemplo:

```
div{  
    border-top-width:20px;  
    border-left-width: 10px;  
    border-right-width: 10px;  
    border-bottom-width: 5px;  
    border-top-color:black;  
    border-left-color: red;  
    border-right-color: green;  
    border-bottom-color: yellow;  
    border-top-style:solid;  
    border-left-style: dashed;  
    border-right-style: double;  
    border-bottom-style: groove;  
}
```

Propriedade padding

Utilizada para determinar o espaçamento entre a borda e o conteúdo do elemento. Podemos utilizar as propriedades abaixo para configurar os espaçamentos de forma individual:

padding-top: Espaçamento a partir da borda superior;

padding-left: Espaçamento a partir da borda lateral esquerda;

padding-right: Espaçamento a partir da borda lateral direito;

Anotações	

padding-bottom: Espaçamento a partir da borda inferior;

Exemplo:

```
p{
  padding-top:20px;
  padding-left: 10px;
  padding-right: 10px;
  padding-bottom: 5px;
}
```

Propriedade border

Esta propriedade tem por finalidade indicar um valor de arredondamento para as bordas de um elemento.

Exemplo:

border-radius: 10px

Propriedade border

É utilizada para aplicar um efeito de sombra sobre uma borda. Esta propriedade é definida através de 4 valores que indicam respectivamente a distância horizontal em relação a borda, a distância vertical em relação a borda, o raio da borda e por fim a cor a ser utilizada.

Exemplo:

box-shadow: 5px 5px 15px rgba(255,0,0,0.8)

box-shadow: 5px -5px 20px #FF6F00

	Anotações



Box-Sizing

A propriedade CSS box-sizing é utilizada para alterar a propriedade padrão da box model, usada para calcular larguras (widths) e alturas (heights) dos elementos.

Exemplo:

```
div{  
  
    box-sizing:border-box;  
  
}
```

Os valores possíveis para a propriedade box-sizing são:

content-box

Essa é o estilo padrão. As propriedades width (largura) e height (altura) são medidas incluindo só o conteúdo, mas não o padding, border ou margin.

padding-box

As propriedades de largura (width) e de altura (height) incluem o tamanho padding size, mas não incluem a propriedade border ou margem.

border-box

As propriedades de largura (width) e de altura (height) incluem o tamanho padding size e a propriedade border, mas não incluem a propriedade margin.

Tipo de Elementos

Os elementos HTML são divididos basicamente em dois grandes grupos:

- Elementos de Bloco
- Elementos de Linha

Anotações	

Elementos de Bloco

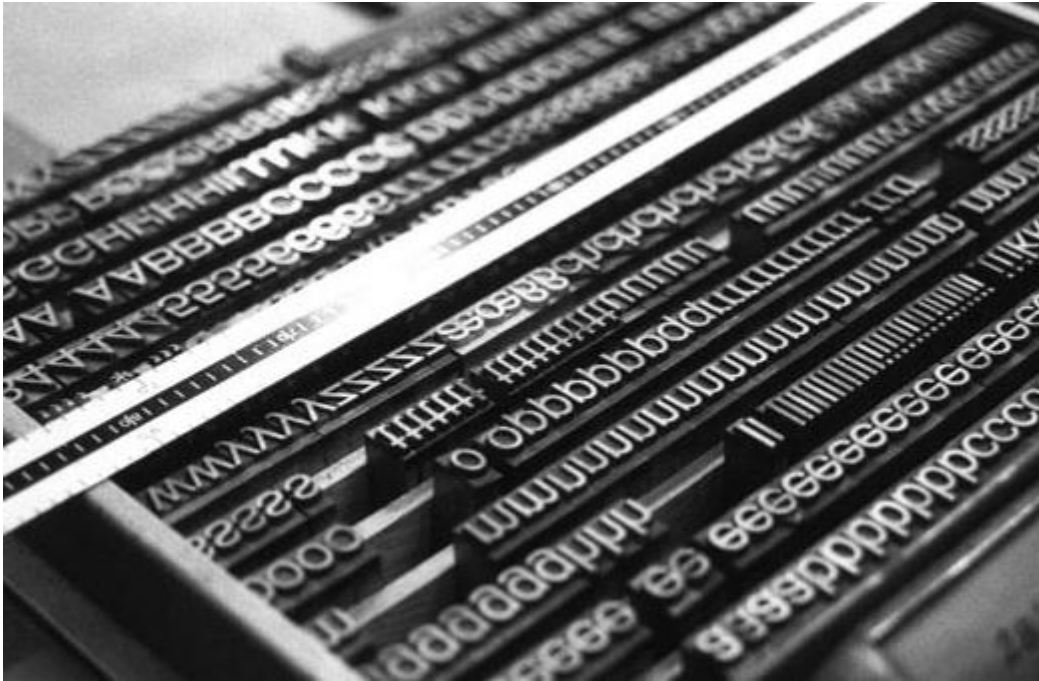
Elementos de bloco ou '*block elements*' são elementos que por padrão aparecem um abaixo do outro, possuindo uma quebra de linha antes e após a sua inclusão.

Elementos de Linha

Elementos de linha ou '*inline elements*' são elementos que por padrão aparecem um ao lado do outro, não possuindo quebras de linha antes ou após a sua inclusão.

	Anotações

Capítulo 06 – Tipografia



Objetivos:

Neste capítulo você irá aprender:

- Fontes Seguras
- Famílias de Fontes
- Espaçamento e alinhamento de texto
- Efeitos em textos
- Estilização de fontes

Anotações	

Fontes seguras

Fontes seguras é o nome dado ao conjunto de fontes que estão disponíveis na maioria dos sistemas operacionais. A listagem de fontes seguras é composta por um conjunto de 9 fontes:

1. Fonte Arial - 0123456789
2. **Fonte Arial Black - 0123456789**
3. Fonte Comic Sans MS - 0123456789
4. Fonte Courier New - 0123456789
5. Fonte Georgia - 0123456789
6. **Fonte Impact - 0123456789**
7. Fonte Times New Roman - 0123456789
8. Fonte Trebuchet MS - 0123456789
9. Fonte Verdana - 0123456789

Famílias de fontes

Família de fontes é a denominação dada ao conjunto de 5 fontes genéricas definidas pelo W3C. Estas fontes se destinam a servir como salvaguarda caso o sistema operacional não encontre a fonte definida pelo autor. As famílias de fontes definidas pelo W3C são:

Família Serif

Fontes serif, são caracterizadas por possuírem pequenos traços de acabamento em suas extremidades e suas letras serem espaçadas.

Exemplo: **Fonte Serif**

Família Sans-serif

Fontes sans-serif, são caracterizadas por não possuírem pequenos traços de acabamento em suas extremidades e suas letras serem espaçadas, também são chamadas de fontes sem serifas.

Exemplo: **Fonte Sans-serif**

	Anotações



Família Cursive

São as fontes que se simulam a escrita cursiva (a mão livre).

Exemplo: *Fonte Cursive*

Família Fantasy

São as fontes que apresentam características decorativas.

Exemplo: **Fonte Fantasy**

Família Monospace

Fontes mono espaçadas, tem como característica possuírem a mesma largura em todos os caracteres.

Exemplo: `Fonte Monospace`

Espaçamento e Alinhamento de Textos

Espaçamento de Texto

Para controlarmos o espaçamento de texto podemos utilizar as seguintes propriedades:

Propriedade **letter-spacing**

Controla o espaçamento entre as letras do texto.

Os valores para essa propriedade podem ser:

`letter-spacing:normal`

`letter-spacing: medita de comprimento css (Ex: 12px)`

`letter-spacing:inherit`

Anotações	

Propriedade word-spacing

Controla o espaçamento entre as palavras de um texto.

Os valores para essa propriedade podem ser:

```
word-spacing:normal
```

```
word-spacing: medida de comprimento css (Ex: 12px)
```

```
word-spacing:inherit
```

Propriedade line-height

Controla a altura dos elementos inline dentro dos elementos de bloco (altura da linha).

Os valores para essa propriedade podem ser:

```
line-height:normal
```

```
line-height: medida de comprimento exata ou percentual (Ex: 12px)
```

```
word-spacing:inherit
```

Obs. Line-height não suporta valores negativos.

Alinhamento de texto

Propriedade vertical-align

Controla o alinhamento vertical dos elementos inline, imagens e inputs.

Os valores para essa propriedade podem ser:

```
vertical-align:baseline
```

```
vertical-align:sub
```

```
vertical-align:sup
```

```
vertical-align:top
```

	Anotações



vertical-align:text-top

vertical-align:middle

vertical-align:bottom

vertical-align:text-bottom

vertical-align:percentual/medida de comprimento css

vertical-align:inherit

Propriedade text-align

Controla o alinhamento horizontal de um elemento de bloco.

Os valores possíveis são:

text-align:left

text-align:center

text-align:right

text-align:justify

Propriedade text-indent

Define uma indentificação para a primeira linha de um texto dentro de um elemento de bloco.

Os valores possíveis são:

text-indent:medida de comprimento css (Ex: 20px)

text-indent:percentual (Ex: 10%)

text-indent:inherit

Efeitos de textos

Propriedade text-decoration

Adiciona um efeito decorativo. Este efeito pode ser uma linha horizontal ou um efeito piscante.

Os valores possíveis são:

text-decoration: none

Anotações	

text-decoration:underline
text-decoration:overline
text-decoration:line-through
text-decoration:blink
text-decoration:inherit

Obs. Pode ser adicionado mais de um valor para a propriedade.

Propriedade text-transform

Aplica o efeito de capitalização do texto (caixa alta, caixa baixa).

Os valores possíveis são:

text-transform:none
text-transform:capitalize
text-transform:uppercase
text-transform:lowercase
text-transform:inherit

Dica:

Se desejar aplicar este efeito apenas na primeira letra de um parágrafo pode ser utilizado o pseudo elemento :first-letter.

Exemplo:

```
p:first-letter{
    text-transform:capitalize
}
```

	Anotações



Estilos de fonte

As principais propriedades para o controle de fonte são:

Propriedade font-family

Define um conjunto de fontes que podem ser utilizadas para a renderização do texto. O navegador tentará aplicar a fonte por ordem de prioridade, caso o sistema operacional não possua a primeira fonte definida, o navegador tentará utilizar a próxima da lista.

Exemplo:

```
p{  
    font-family:Verdana, "Times New Roman", Sans-Serif;  
}
```

Obs. O uso de letras maiúsculas ou minúsculas na nomeação das fontes é indiferente.

Propriedade font-style

Define a estilização da fonte no modo normal, itálico ou oblíquo. Na prática os navegadores aplicam o mesmo efeito para os valores itálico e oblíquo.

Os valores possíveis são:

font-style:normal

font-style:italic

font-style:oblique

Propriedade font-variant

Altera o tamanho das letras minúsculas para maiúsculas com a altura um pouco menor.

Anotações	

Os valores possíveis são:

font-variant:normal

font-variant:small-caps

font-variant:inherit

Propriedade font-weight

Aplica um peso para a cor da fonte.

Os valores possíveis são:

font-weight: normal

font-weight: bold

font-weight: bolder

font-weight: lighter

font-weight: um valor numérico entre 100 e 900 (Ex: 400)

font-weight: inherit

Propriedade font-size

A propriedade font-size é utilizada para definirmos o tamanho da fonte.

Os valores possíveis são:

font-size: uma medida de valor absoluto (Ex: 20mm)

font-size: uma medida de valor relativo (Ex: 1.2 rem)

font-size: uma valor percentual (Ex: 10%)

font-size: inherit

	Anotações



Propriedade font

Podemos utilizar a propriedade font para simplificarmos a configuração de várias propriedades em uma mesma linha. A sequência deve corresponder conforme o modelo abaixo:

font: font-style font-variant font-weight font-size/line-height font-family.

Não é necessário passar todos os valores.

Exemplos:

```
p{  
    font: italic bold 12px/14px verdana,sans-serif;  
}
```

```
p{  
    font: 12em verdana,sans-serif;  
}
```

Anotações	

Capítulo 07 – Cores, backgrounds e links



Objetivos:

Neste capítulo você irá aprender:

- Definições de Cores de primeiro plano
- Definições de plano de fundo (background)
- Estilização de links

	Anotações



Definições de cores de primeiro plano

Para definirmos a cor de primeiro plano, ou cor de texto, através do CSS devemos utilizar a propriedade **color**. Existem várias formas de configuração dessa propriedade, mas as 3 formas mais utilizadas são:

Cor hexadecimal

A definição de cor no formato hexadecimal é precedida pelo carácter '#' e mais 6 números no formato hexadecimal (valores de 0 a F), sendo que:

- Os dois primeiros valores representam a intensidade da cor vermelha;
- Os dois próximos valores representam a intensidade da cor verde;
- Os dois últimos valores representam a intensidade da cor Azul.

Podemos ainda utilizar um formato de 3 posições para simplificação de algumas cores quando os dois valores referentes a mesma indicação (Vermelho, Verde, Azul) possuem a mesma numeração.

Exemplo:

color:#f1f1f1

color:#008b8b

color:#ccddcc ou #cdc

color:#ffffff ou #fff

Cor RGB

A indicação de cor no formato RGB é definida pelo padrão RGB (R,G,B). Neste formato cada posição pode receber um valor entre 0 e 255.

Exemplo:

color:rgb(0,128,128)

color:rgb(255,0,0)

Podemos ainda utilizar o padrão RGBA. Este padrão acrescenta um valor dentre 0 e 1 para o canal alfa(A), este canal indica qual a intensidade da transparência a ser aplicada.

Exemplo:

color:rgba(255,255,255,0.7)

Anotações	

Cor nomeada

Podemos ainda utilizar a nomeação das cores utilizando o idioma inglês para a definição do valor a ser configurado.

Exemplo:

color:red

color:blue

Definições de plano de fundo - Background

Para a definição de plano de fundo podemos utilizar uma cor ou uma imagem.

Propriedade Background-color

A propriedade background-color é utilizada na configuração de uma cor para o plano de fundo.

Exemplos:

background-color:rgba(255,0,0,0.1)

background-color:#2F4F4F

Propriedade Background-image

A propriedade background-image é utilizada para a configuração de uma imagem como plano de fundo.

Exemplos:

background-image:url('fundo.png')

background-image:url("data:image/gif;base64,R0lGODlhUAAPAK...")

Propriedade Background-repeat

A propriedade background-repeat indica como a imagem deve se repetir no plano de fundo para ocupar todo o espaçamento.

Os valores possíveis são:

background-repeat:repeat

background-repeat:repeat-x

background-repeat:repeat-y

background-repeat:no-repeat

	Anotações



Propriedade Background-position

A propriedade background-position é utilizada para indicar em qual posição deve ser colocada a imagem de fundo. Caso não seja informada, a imagem irá ser definida a partir da posição (0,0) que é o canto superior esquerdo do elemento.

Os valores indicados nesta propriedade são referentes a posição x,y inicial da imagem.

Exemplos:

background-position:20px 20px

background-position:25% 25%

background-position:left top

background-position:center top

Propriedade Background-attachment

A propriedade background-attachment é utilizada para indicar se a imagem permanecerá fixa ou se deve rolar juntamente com a tela.

Os valores possíveis são:

background-attachment :scroll

background-attachment :fixed

background-attachment :inherit

Estilização de links

Para a estilização de um link (elemento <a>) devemos levar em consideração os 4 possíveis estados do elemento.

Estados dos links

Os estados do link são definidos através de pseudo-classes da seguinte forma:

:link -> anchor no seu estado inicial

:visited -> anchor visitado

:hover -> ponteiro do mouse está sobre o elemento anchor

:active -> quando o usuário pressiona o mouse em cima do link

Anotações	

Através destas pseudo-classes podemos aplicar valores de estilização de cor/plano de fundo, texto, bordas entre outros para cada link que desejarmos.

Exemplo:

```
a:link{
    color:black;
    text-decoration:none;
    font-size:16px;
}
```

```
a:visited{
    color:orange;
    text-decoration:none;
    font-size:16px;
}
```

```
a:hover{
    color:blue;
    text-style:italic;
    font-size:20px;
}
```

	Anotações

Capítulo 08 - Listas



Objetivos:

Neste capítulo você irá aprender:

- Propriedade list-style-type
- Propriedade list-style-image
- Propriedade list-style-position

Anotações	

Propriedade List-style-type

Esta propriedade indica o tipo de marcador a ser utilizado pela lista.

Os valores para esta propriedade são:

`list-style-type:disc`

`list-style-type:circle`

`list-style-type:square`

`list-style-type:decimal`

`list-style-type:decimal-leading-zero`

`list-style-type:lower-roman`

`list-style-type:upper-roman`

`list-style-type:georgian`

`list-style-type:armenian`

`list-style-type:lower-alpha/lower-latin`

`list-style-type:upper-alpha/upper-latin`

`list-style-type:greek`

`list-style-type:none`

Propriedade List-style-image

Esta propriedade define uma imagem como marcadora de um item da lista.

Propriedade List-style-position

Esta propriedade indica a posição do box que contém o marcador da lista em relação ao box principal.

Os valores possíveis são:

`list-style-position:outside`

`list-style-position:inside`

`list-style-position:inherit`

	Anotações

Capítulo 09 - Posicionamento



Objetivos:

Neste capítulo você irá aprender:

- Propriedade Float
- Posicionamentos
- Propriedade Clear
- Flexbox

Anotações	

Propriedade float

Por padrão os elementos de bloco, tais como containers div, são colocados um abaixo do outro em um documento html.

Entretanto, algumas vezes faz-se necessário que alguns elementos flutuem para a direita ou para esquerda, saindo do fluxo normal do documento.

Para conseguirmos este comportamento devemos utilizar a propriedade float.

A propriedade float aceita os seguintes valores:

none: não flutua, comportamento padrão.

left: flutua o componente para a esquerda;

right: flutua o componente para a direita;

inherit: utiliza o comportamento definido no elemento pai.

Exemplo:

```
<!doctype html>
<html>
<head>
  <style>
    div{
      border: 1px solid blue;
      width:100px;
      height: 100px;
    }
    .normal{
      float:none;
    }
    .esquerda{
      float:left;
    }
    .direita{
      float:right;
    }
  </style>
</head>
<body>
```




```
</style>
</head>
<body>
  <div class="normal">
    normal
  </div>
  <div class="esquerda">
    esquerda
  </div>
  <div class="direita">
    direita
  </div>
</body>
</html>
```

Propriedade clear

Qualquer elemento colocado após um elemento flutuante irá contorná-lo, pois o elemento flutuante não pertence ao fluxo normal de renderização.

Para solucionar alguns problemas que podem vir a ocorrer devido a esta característica, devemos alterar o componente subsequente aos componentes flutuantes, ajustando a sua propriedade clear.

Esta propriedade indica que o elemento não deve contornar o elemento flutuante.

Entre os valores disponíveis para a propriedade clear, os mais utilizados são:

left: deixa de contornar apenas elementos que estejam flutuando a esquerda.

right: deixa de contornar apenas elementos que estejam flutuando a direita.

both: deixa de contornar elementos que estejam flutuando a esquerda e a direita.

Anotações	

Posicionamentos

Para definirmos o posicionamento de um elemento podemos utilizar a propriedade position.

Os valores para a propriedade position podem ser:

position:relative

position:absolute

position:fixed

position:inherited

Posicionamento relativo

Este posicionamento não altera a forma de posicionamento dos elementos. Para mudarmos a posição do elemento devemos utilizar as propriedades top, left, right e bottom.

Posicionamento absoluto

O posicionamento absoluto retira o elemento do fluxo normal de posicionamento. Esta configuração indica que o elemento deve ser posicionado a partir da posição do elemento pai utilizando as propriedades top, left, right e bottom.

Posicionamento fixo

O posicionamento fixo é um caso especial do posicionamento absoluto. Neste tipo de posicionamento o elemento irá se posicionar de modo fixo em relação a área de renderização do documento.

Flexbox

Flexbox ou Modelo de layout em caixas flexíveis foi definido na especificação 3 do CSS e permite uma configuração mais dinâmica dos elementos.

Nesse tipo de modelo temos um elemento pai, também denominado de container, e um ou mais elementos filhos. As configurações serão normalmente definidas no elemento pai e refletirão para os elementos filhos.

	Anotações



Propriedades a serem definidas no elemento container

display: flex -> habilita o contexto flexível para o container.

flex-direction

Indica a direção em que os elementos filhos devem ser posicionados dentro do container.

Os valores possíveis são:

flex-direction: row

Todos os elementos filhos serão colocados um ao lado do outro da esquerda para a direita

flex-direction: row-reverse

Todos os elementos filhos serão colocados um ao lado do outro da direita para a esquerda

flex-direction: column

Todos os elementos filhos serão colocados um abaixo do outro do topo para o rodapé.

flex-direction: column-reverse

Todos os elementos filhos serão colocados um abaixo do outro do rodapé para o topo.

flex-wrap

Determina se os elementos devem realizar a quebra de linha.

Os valores possíveis são:

flex-wrap: wrap

Realiza a quebra de linha

flex-wrap: no-wrap

Não permite a quebra de linha

justify-content

Anotações	

Define o alinhamento em relação ao eixo horizontal

Os valores possíveis são:

`justify-content: flex-start`

`justify-content: flex-end`

`justify-content: center`

`justify-content: space-between`

`justify-content: space-around`

align-items

Define o alinhamento em relação ao eixo vertical.

Os valores possíveis são:

`align-items: flex-start`

`align-items: flex-end`

`align-items: center`

`align-items: stretch`

`align-items: baseline`

align-content

Define o alinhamento interno entre as linhas.

Os valores possíveis são:

`align-content: flex-start`

`align-content: flex-end`

`align-content: center`

`align-content: stretch`

`align-content: space-between`

`align-content: space-around`

	Anotações



Propriedades a serem definidas no elemento filho

order

Indica a ordem em que o elemento deve aparecer dentro do container.

flex-grow

Indica a habilidade do elemento crescer se necessário para ocupar o espaçamento do elemento pai.

flex-shrink

Indica a capacidade do elemento encolher se necessário.

flex-basis

Define o tamanho padrão de um elemento antes do espaço restante ser distribuído.

flex

Forma simplificada para a atribuição dos valores referente as propriedades flex-grow, flex-shrink e flex-base. O valor padrão é 0 1 auto;

align-self

Define o alinhamento do elemento em relação ao eixo y, sobrescrevendo o valor da propriedade align-items para o elemento corrente.

Anotações	

Capítulo 10 – Animações e Transições no CSS



Objetivos:

Neste capítulo você irá aprender:

- Desenvolver animações
- Desenvolver transições

	Anotações

Animações e Transições

Com a evolução dos browsers o CSS conseguiu trazer muitas novidades como bordas arredondadas, sombras, fontes customizadas, degradês, entre outras funcionalidades, porém uma que chama muito a atenção dos desenvolvedores é a animação.

Desde o início da web as animações eram desenvolvidas através de JavaScript, tanto que bibliotecas como o jQuery dominaram por muito tempo o mercado de animações, devido a facilidade de uso e as vastas funcionalidades que o desenvolvedor poderia aplicar em seus sites.

Atualmente é possível desenvolver animações muito interessantes no CSS utilizando as funções **transition** e **animation**.





Transition

O comando transition do CSS é capaz de criar animações simples trabalhando com as dimensões dos componentes, cores, sombras, margens, entre outras características.

Há cinco comandos que precisamos aprender para trabalhar com as transições do CSS que são:

1. transition
2. transition-delay
3. transition-duration
4. transition-property
5. transition-timing-function

Antes de iniciar nossos estudos é bom deixar claro a compatibilidade entre os browsers, abaixo segue a lista dos principais navegadores e as versões mínimas para que esse efeito CSS funcione.

Property				
transition	26.0	10.0	16.0	6.1
transition-delay	26.0	10.0	16.0	6.1
transition-duration	26.0	10.0	16.0	6.1
transition-property	26.0	10.0	16.0	6.1
transition-timing-function	26.0	10.0	16.0	6.1

Anotações	

Vamos fazer um exemplo simples utilizando apenas o comando **transition**, iremos implementar um efeito **hover**, que ocorre toda vez que o usuário passa o cursor sobre determinado elemento. Criaremos uma **div** com uma largura de 100px, ao passar o cursor sobre deixamos essa **div** com uma largura de 300px, abaixo segue a nossa estrutura:

```

1  <!DOCTYPE html>
2  <html>
3  <head>
4      <style>
5          div {
6              width: 100px;
7              height: 100px;
8              background: red;
9              transition: width 2s;
10         }
11
12         div:hover {
13             width: 300px;
14         }
15     </style>
16 </head>
17
18 <body>
19
20     <div></div>
21
22 </body>
23 </html>

```

Note que a propriedade **transition**, que está na linha 9 possui dois parâmetros, o primeiro é a propriedade CSS que você quer que seja alterada, já o segundo parâmetro é o tempo, por padrão medido em segundos, neste exemplo aplicamos uma transição de 2 segundos.

Vamos implementar esse exemplo adicionando duas transições, além da largura vamos alterar a altura, sendo assim basta separarmos nossas transições por vírgula, observe a implementação no exemplo abaixo:

	Anotações

```

1  <!DOCTYPE html>
2  <html>
3  <head>
4      <style>
5          div {
6              width: 100px;
7              height: 100px;
8              background: ■ red;
9              transition: width 2s, height 2s;
10         }
11
12         div:hover {
13             width: 300px;
14             height: 300px;
15         }
16     </style>
17 </head>
18
19 <body>
20
21     <div></div>
22
23 </body>
24 </html>

```

Na linha número 9 adicionamos os elementos que desejamos adicionar um evento de transição, separamos por vírgula cada uma das propriedades afetadas, já na linha 14 adicionamos uma altura diferente da altura padrão da div que é de 100px.

Toda vez que o usuário passar o cursor sobre essa **div** a largura e a altura serão alteradas, as transições basicamente precisam ter um ponto inicial e um ponto final, deixamos as características da **div** como pontos iniciais e na nossa **div:hover** as características finais.

Agora vamos implementar um tempo de espera para realizar a ação, isso chamamos de delay. Há uma propriedade chamada **transition-delay**, onde uma ação é executada após um determinado tempo. Vamos nos basear pelos exemplos anterior de transition, vamos adicionar um delay de dois segundos, para realizar a ação:

Anotações	

```

1  <!DOCTYPE html>
2  <html>
3  <head>
4      <style>
5          div {
6              width: 100px;
7              height: 100px;
8              background: ■ red;
9              transition: width 2s, height 2s;
10             transition-delay: 2s;
11         }
12
13         div:hover {
14             width: 300px;
15             height: 300px;
16         }
17     </style>
18 </head>
19
20 <body>
21
22     <div></div>
23
24 </body>
25 </html>

```

Observe a linha 10, estamos dizendo que as transições irão ocorrer apenas depois de dois segundos quando o cursor estiver sobre o nosso elemento **div**, quando sairmos do elemento levará mais dois segundos para voltar para a largura e altura de 100px;

Agora vamos utilizar o comando **transition-duration** e **transition-property**, você se lembra que utilizamos o comando **transition**, com dois parâmetros correto? O primeiro parâmetro é a característica que queremos que ocorra a transição e depois o tempo, mas imagine que você queira separar em característica que deseja alterar e o tempo de transição, como fazer isso? Nesse caso podemos utilizar esses dois comandos.

Para ficar mais fácil usaremos como base os exemplos anteriores para alterar a largura e a altura de uma **div**, veja na imagem abaixo:

	Anotações

```

1  <!DOCTYPE html>
2  <html>
3  <head>
4      <style>
5          div {
6              width: 100px;
7              height: 100px;
8              background: ■ red;
9              transition-property: width, height;
10             transition-duration: 2s;
11         }
12
13         div:hover {
14             width: 300px;
15             height: 300px;
16         }
17     </style>
18 </head>
19
20 <body>
21
22     <div></div>
23
24 </body>
25 </html>

```

Note que na linha 9 definimos apenas as características que serão alteradas através da transição, já na linha 10 informamos o tempo de transição, que será de dois segundos.

Agora imagine que você queira que a largura do objeto seja de dois segundos, mas a altura demore cinco segundos para executar, como fazer isso? Basta apenas adicionar mais um tempo de transição no **transition-duration**, vamos ao exemplo:

10	<code>transition-duration: 2s, 5s;</code>
----	---

Para finalizarmos nossos conteúdos sobre as transições, há um comando chamado **transition-timing-function**, que nos permite realizar alguns movimentos com uma escala de tempo. Imagine que você queira que sua transição de cinco segundos comece lenta e em seguida fica rápida, ou inicia de maneira rápida e em seguida fica mais lenta.

Anotações	

Há vários comandos para realizar esse tipo de ação, iremos abordar os principais na tabela abaixo:

Comando	Descrição
ease	Inicia de maneira lenta, depois rápido e por fim lenta novamente
linear	Mantêm a mesma velocidade do início ao fim
ease-in	Inicia de maneira lenta e finaliza de maneira rápida
ease-out	Inicia de maneira rápida e finaliza de maneira lenta
ease-in-out	Inicia de maneira lenta, aumenta a velocidade e por fim deixa lenta.

```

1  <!DOCTYPE html>
2  <html>
3  <head>
4      <style>
5          div {
6              width: 100px;
7              height: 100px;
8              background: red;
9              transition: width 10s;
10             transition-timing-function: ease;
11         }
12
13         div:hover {
14             width: 300px;
15         }
16     </style>
17 </head>
18
19 <body>
20
21     <div></div>
22
23 </body>
24 </html>

```

	Anotações

Na imagem anterior veja que na linha 9 adicionamos uma transição na largura do elemento com dez segundos de efeito, já na linha 10 adicionamos a expressão **ease**, para que inicie de maneira lenta, em seguida a velocidade aumenta e no final da transição volta a ficar lento novamente.

De maneira bem simples você pode criar botões, logos animados, efeitos de menu, imagens, entre outros elementos disponibilizados no HTML de maneira mais chamativa para os clientes, e o melhor de tudo sem utilizar JavaScript ou qualquer complemento.

Animation

O comando **Animation**, é muito familiar com as **transitions**, que tem como objetivo criar transições de largura, altura, margens, entre outras opções.

A grande vantagem das animações perante as transições é a possibilidade de manipular mais elementos e realizar mais de uma transição.

Anteriormente fizemos alguns exemplos com a largura e a altura de uma **div**, mas imagine que você queira que ela aumente as suas dimensões e depois diminua, em seguida gire em 360º e ainda mude de cor. Parece loucura, mas as animações do CSS conseguem fazer isso, diferente das **transitions**, que são mais limitadas.

Utilizando o comando **animation**, teremos diversas funcionalidades como:





1. @keyframes
2. animation-name
3. animation-duration
4. animation-delay
5. animation-iteration-count
6. animation-direction
7. animation-timing-function
8. animation-fill-mode
9. animation

Muitos sites utilizam JavaScript para criar essas animações, porém agora você pode criar animações de qualidade com códigos 100% CSS, uma vantagem interessante, pois você não precisa se preocupar em trabalhar com bibliotecas ou funcionalidades complexas do JavaScript, porém vale lembrar que mesmo a função **animation** existindo.

Anotações	

Há momentos que a complexidade do projeto não comporta o uso desse comando ou até mesmo a elaboração de animações muito longas pode ser considerada inviável, devido o código extenso e a dificuldade de encontrar pessoas para dar manutenção em determinados projetos.

Antes de iniciarmos nossas animações vale lembrar de sempre verificar a compatibilidade entre os navegadores, talvez seu público alvo não utilize uma versão compatível, abaixo segue a lista de navegadores e suas devidas versões compatíveis:

Property				
@keyframes	43.0	10.0	16.0	9.0
animation-name	43.0	10.0	16.0	9.0
animation-duration	43.0	10.0	16.0	9.0
animation-delay	43.0	10.0	16.0	9.0
animation-iteration-count	43.0	10.0	16.0	9.0
animation-direction	43.0	10.0	16.0	9.0
animation-timing-function	43.0	10.0	16.0	9.0
animation-fill-mode	43.0	10.0	16.0	9.0
animation	43.0	10.0	16.0	9.0

Agora que vimos o que é o comando **animation**, vamos dar um exemplo simples do seu uso, para isso serão utilizados alguns comandos que são:

1. animation-name
2. animation-duration
3. @keyframes

A ideia de utilizar esses três elementos é bem simples, o **animation-name** irá definir o nome dessa animação que será criada, o **animation-duration** será responsável por informar o tempo de animação, e por fim temos o **@keyframes**, que irá realizar a transição dependendo do **animation-name** informado.

Importante saber que o comando **@keyframes**, é um comando que ficará de maneira externa do elemento que receberá a animação, pois nele poderão ser adicionados comandos complexos, e como a ideia é deixar o código bem organizado o CSS cria essa divisão.

	Anotações

O **@keyframes**, possui dois comandos internos que são o **from** e o **to**, eles tem como objetivo informar o ponto inicial da animação e o ponto final.

Em nosso exemplo iremos criar uma animação onde haverá uma **div** com a cor de fundo azul, durante quatro segundos a cor será alterada de azul para roxo.

```

1  <!DOCTYPE html>
2  <html>
3  <head>
4      <style>
5          div {
6              width: 400px;
7              height: 400px;
8              animation-name: exemplo;
9              animation-duration: 4s;
10             background-color: blue;
11         }
12
13         @keyframes exemplo {
14             from {background-color: blue;}
15             to {background-color: purple;}
16         }
17     </style>
18 </head>
19
20 <body>
21
22     <div></div>
23
24 </body>
25 </html>

```

Note que a cor será alterada de azul para roxo em quatro segundos, graças a propriedade **@keyframe**.

Viu como é simples trabalhar com **animations**? Que tal trabalharmos com mais cores e com uma transição infinita? Vamos por partes, iremos implementar nosso **@keyframes** adicionando um percentual para alterar as cores.

Além das funções **to** e **from**, você pode trabalhar com um percentual para informar quando será realizada alguma ação. Vamos trabalhar com os percentuais de 0%, 20%, 40%, 60%, 80% e 100%, onde 0% significa o início da nossa animação e 100% quando for finalizada.

Anotações	

Iremos adicionar também o comando **animation-iteration-count**, para especificar quantas vezes será feita a troca de cores, por padrão quando não utilizamos essa função a animação ocorre apenas uma vez, podemos definir um número de vezes ou o comando **infinite**, para que ocorra de maneira contínua, vamos ao exemplo.

```

1  <!DOCTYPE html>
2  <html>
3  <head>
4      <style>
5          div {
6              width: 400px;
7              height: 400px;
8              animation-name: exemplo;
9              animation-duration: 4s;
10             background-color: blue;
11             animation-iteration-count: infinite;
12         }
13
14         @keyframes exemplo {
15             0% {background-color: blue;}
16             20% {background-color: purple;}
17             40% {background-color: green;}
18             60% {background-color: red;}
19             80% {background-color: purple;}
20             100% {background-color: blue;}
21         }
22     </style>
23 </head>
24
25 <body>
26
27     <div></div>
28
29 </body>
30 </html>

```

Como mencionado anteriormente utilizamos na linha 11 o comando **animation-iteration-count: infinite**, para uma animação contínua, já nosso **@keyframes** possui várias cores que serão alteradas dependendo do percentual atingido.

	Anotações

Talvez você se pergunte, da onde foi tirado esses percentuais, veja que na linha 9 possuímos nosso tempo de animação, quanto maior esse tempo maior o tempo de aparição das cores, logo 0% equivale a 0 segundo e 100% equivale a 4 segundos de animação.

Você pode utilizar qualquer valor em percentual para realizar alguma modificação, exemplo: 0%, 30%, 70%, 90% e 100%, isso fica sob responsabilidade do desenvolvedor.

Para finalizarmos nosso capítulo sobre animações com CSS, iremos implementar nosso exemplo de cores, adicionando movimentações, veja na imagem a seguir como deixar sua animação ainda mais interessante.

```

1  <!DOCTYPE html>
2  <html>
3  <head>
4      <style>
5          div {
6              width: 400px;
7              height: 400px;
8              animation-name: exemplo;
9              position: relative;
10             animation-duration: 4s;
11             background-color: blue;
12             animation-iteration-count: infinite;
13         }
14
15         @keyframes exemplo {
16             0% {background-color: blue; left:0px; top:0px;}
17             25% {background-color: purple; left:200px; top:0px;}
18             50% {background-color: green; left:200px; top:200px;}
19             75% {background-color: purple; left:0px; top:200px;}
20             100% {background-color: blue; left:0px; top:0px;}
21         }
22     </style>
23 </head>
24
25 <body>
26
27     <div></div>
28
29 </body>
30 </html>

```

Na linha 9 foi adicionado o comando **position:relative**, para funcionar os comandos de margem como **top** e **left**, foi removida a cor vermelha para deixar o efeito mais interessante.

Inicia à esquerda com a cor azul, depois será dada uma margem posicionando à direita com a cor roxa, em seguida uma margem de topo para descer e alterando a cor para verde, logo mais volta para a esquerda alterando a cor para o roxo, e por fim sobe voltando para a posição onde iniciou a animação com a cor azul.

Mantivemos o **animation-iteration-count: infinite**, para deixar a animação sendo executada de maneira contínua.

	Anotações



Apêndice A

Lista de Pseudo-classes:

:active
:checked
:default
:dir()
:disabled
:empty
:enabled
:first
:first-child
:first-of-type
:fullscreen
:focus
:hover
:indeterminate
:in-range
:invalid
:lang()
:last-child
:last-of-type
:left
:link
:not()
:nth-child()
:nth-last-child()
:nth-last-of-type()
:nth-of-type()

Anotações	

:only-child
:only-of-type
:optional
:out-of-range
:read-only
:read-write
:required
:right
:root
:scope
:target
:valid
:visited

	Anotações



Anotações	

Apêndice B

Lista de Pseudo-Elementos

::after

::before

::first-letter

::first-line

:selection

::backdrop

	Anotações