

```
div class="container">
  <div class="row">
    <div class="col-md-6 col-lg-8"> <!-- _____ BEGIN NAVIGATION
      <nav id="nav" role="navigation">
        <ul>
          <li><a href="index.html">Home</a></li>
          <li><a href="home-events.html">Home Events</a></li>
          <li><a href="multi-col-menu.html">Multiple Column Men
          <li class="has-children"> <a href="#" class="current">
```

HTML E CSS

AULA 01

Apresentação

Combinados

Preparando ambiente de desenvolvimento - Introdução a editores de texto

Introdução HTML e CSS

O que é linguagem de marcação

HTML vs. HTML5

Elementos HTML: tags

Comentários no HTML

Adicionando links de arquivos externos `<link>`

Adicionando imagens `` e seus atributos `src` e `alt`)

Blocos `<div>`

Criando links `<a>` e seus atributos `href`

O que é CSS?

Diferentes formas de adicionar estilo à página

Importando fontes externas (ex.: Google Fonts)

Estilo padrão (default) da página

Uso de classes e identificadores nas tags

Propriedades de estilo no CSS:

Dimensões de elementos (largura, altura)

Margens e espaçamentos (padding)

Fontes e suas cores, famílias, tamanhos e ênfases

Cores de fundo, transparências e gradientes

Medidas usadas no CSS (px, %, em e rem)

Mãos no código

Exercício 01 - Landing Page Simples

Exercício 02 - Mini Currículo

HTML

- O que é linguagem de marcação

Linguagem de marcação é uma maneira de anotar algum conteúdo para que ele fique **sintaticamente** compreensível. Imagine um livro que foi comentado por um editor ou outro autor. Em cada parte deste livro teria algum tipo de marcação que indica o comentário ali inserido, deixando claro isso para você, leitora.

Para produzir páginas web também usamos a linguagem de marcação. O nome dela é HTML.

- HTML vs. HTML5

É importante compreender o papel do HTML como linguagem de marcação, já que pode ser confundido facilmente como linguagem de programação. Mas, como é usado, sua função é estruturar um conteúdo (que pode ser texto, figuras e tudo o mais) de maneira que um navegador web consiga interpretar e exibir para o usuário.

O significado da sigla **HTML** é Hyper Text Markup Language (em tradução livre: Linguagem de Marcação Hipertexto).

O HTML tomou suas primeiras conformações formais nos anos 90. A partir disso, foram muitas versões. Quem lembra dos layouts dos sites nos anos 2000?



Entendo isso, hoje usamos a versão 5, por isso HTML5. É uma versão que foi evoluindo ao longo do tempo, entendendo as necessidades das pessoas que usavam essa linguagem, assim como as transformações ocorridas na internet. Você pode ler mais sobre essa versão [aqui](#).

Algumas das suas características principais, são:

- **Semântica:** permite você descrever mais precisamente o seu conteúdo.
- **Conectividade:** permite uma comunicação com o servidor de formas modernas e inovadoras.
- **Offline e armazenamento:** Permite que páginas web armazenem dados localmente do lado do cliente e opere de forma offline mais eficientemente.
- **Multimídia:** Viabiliza a utilização de áudio e vídeo de forma primorosa na Web Aberta.
- **Gráficos e efeitos 2D/3D:** viabiliza um leque diversificado de opções de representação gráfica.
- **Performance e integração:** fornece grande otimização de velocidade e melhor utilização do hardware do computador.
- **Acesso ao dispositivo:** viabiliza a utilização de diversos métodos e dispositivos de entrada e saída.
- **Estilização:** permite aos autores a escrita de temas mais sofisticados.

- Elementos HTML: tags

Como entendemos, HTML é uma linguagem de marcação. Para que essa marcação ocorra, as tags são usadas. Elas nada mais são que palavras específicas, escritas entre os sinais < > e que indicam o conteúdo que estão dentro de cada uma delas.

Lembre-se: tags são como **caixas**!

Dentro do seu editor de texto, ao estruturar basicamente uma página HTML, pode-se ver:

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta http-equiv="X-UA-Compatible" content="ie=edge">
  <title>Document</title>
</head>
<body>

</body>
</html>
```

Vamos entender o que cada parte significa.

```
<!DOCTYPE html>
```

Ainda não é uma tag HTML. É apenas uma instrução ao navegador web sobre a versão HTML usada. É chamado de **declaração** `<!DOCTYPE>` e deve ser a **primeira** coisa no seu documento HTML. [Você pode descobrir mais.](#)

```
<html lang="en">
...
</html>
```

O atributo HTML lang pode ser usado para declarar o idioma de uma página Web ou uma parte de uma página Web. Este destina-se a ajudar os motores de busca e navegadores.

De acordo com a recomendação W3C você deve declarar o idioma principal para cada página Web com o atributo lang dentro da tag `<html>`. Neste caso, a abreviação do idioma é **en**, referente à língua inglesa.

```
<head>
...
</head>
```

A tag ``<head>`` é respectivo à “cabeça” do documento. Ali dentro geralmente contém o título do documento, os scripts (que são os arquivos que dão dinâmica ao site), os estilos da página e informações meta.

```
<meta charset="UTF-8">
```

Metadados são **dados** (informações) sobre o documento. A tag ``<meta>`` fornece metadados sobre o documento HTML. Eles não serão exibidos na página, mas os navegadores conseguem ler e compreender.

No caso do `charset="utf-8"`, é para a habilitação da leitura de caracteres acentuados (por exemplo á, à, ã, ç).

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

O **viewport** dá as instruções ao navegador sobre como controlar as dimensões da página. O **"width=device-width"** define a largura da página, para seguir a tela de largura do dispositivo (que irá variar dependendo do dispositivo), propiciando a responsividade do website.

Por fim, **"initial-scale=1.0"** define o nível de zoom inicial quando a página é carregada pela primeira vez pelo navegador. [Veja mais.](#)

```
<meta http-equiv="X-UA-Compatible" content="ie=edge">
```

Abrir sites nas versões obsoletas do Internet Explore era uma dificuldade. As novas versões do Microsoft Edge auxiliaram neste caso. Essa meta permite definir a compatibilidade do site que será programado em navegadores explorer. [Veja mais.](#)

```
<title>Document</title>
```

Segundo a [W3 Schools](#), a tag ``<title>`` é obrigatória em todos os documentos HTML e define o título do documento. Esse título é usado na barra de ferramentas do navegador, quando ele é adicionado aos favoritos e também exibe um título para a página nos resultados de motores de busca.

```
<body>
  ...
</body>
```

A tag ``<body>`` define o corpo do documento, que contém todo o conteúdo de um documento HTML, como texto, links, imagens, tabelas, listas, etc.

Ao pensar em uma estrutura básica preenchida, teríamos algo assim.

- Comentários no HTML

Um recurso muito bom para ser usado nas páginas HTML, especialmente quando se está começando a aprender a usar a linguagem ou quando se trabalha em equipe, são os comentários. Com eles é possível adicionar qualquer coisa (explicações, linhas de código que não serão usadas naquele momento, comentários pessoais) e não será exibido para o usuário.

Exemplo de um comentário:

```
<!-- Comentário no HTML com capa de invisibilidade do Harry -->
```

Eles são definidos por terem no início `<!--` e no final `-->`, além de ficarem com uma coloração diferenciada no editor de texto.

- Adicionando links de arquivos externos ``<link>``

A tag ``<link>`` define uma ligação entre um documento e um recurso externo. Pode ser usada, por exemplo, para importar arquivos de folhas de estilo. Seu uso é diverso e depende de cada projeto.

```
<link rel="stylesheet" href="css/style.css">
```

- Adicionando imagens ```` e seus atributos ``src` e `alt``

A tag ```` define uma imagem em uma página HTML. A imagem não é necessariamente importada no documento HTML, ela é apenas ligado ao documento através da tag. Tem dois atributos. O **src** irá descrever o caminho em que a imagem se encontra. E o **alt** é um ótimo atributo tanto de acessibilidade e de SEO, nele adiciona-se uma breve descrição do que a imagem é.

- Blocos ``<div>``

Define uma divisão ou uma seção em um documento HTML. O elemento `<div>` é frequentemente utilizado como um recipiente para outros elementos HTML para estilo deles com CSS ou para executar determinadas tarefas com JavaScript.

- Criando links `` e seus atributos `href` e `target`

Define um hiperlink, que é usado para ligar a partir de uma página para outra. O atributo mais importante do elemento `<a>` é o atributo **href**, o que indica o destino do link.

Por padrão, links aparecerão da seguinte forma em todos os navegadores:

Uma ligação não visitados é sublinhado e azul

Um link visitado é sublinhada e roxo

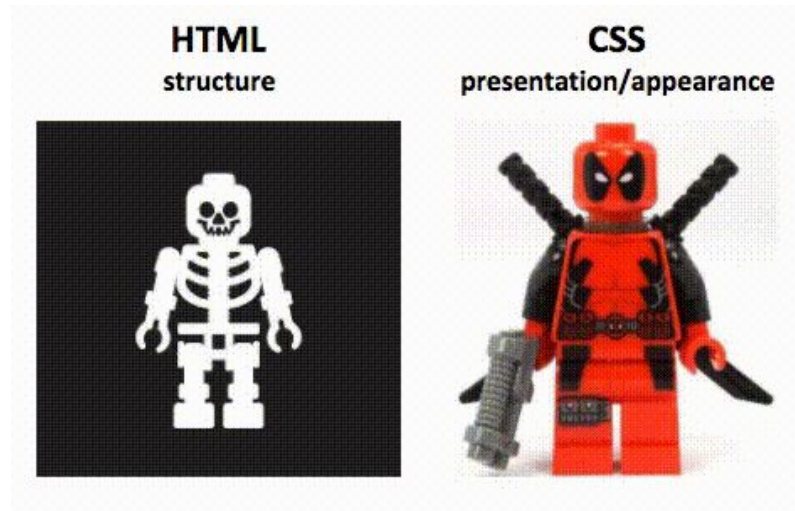
Uma ligação activa é sublinhado e vermelho

CSS

- O que é CSS?

Cascading Style Sheets (CSS) é um mecanismo simples para adicionar estilo (por exemplo, fontes, cores, espaçamento) a documentos da Web. Você pode ver mais [aqui](#).

Para entender a diferença de HTML e CSS, lembre-se do robzinho:



HTML estrutura

CSS apresentação/aparência

- Diferentes formas de adicionar estilo à página

Style

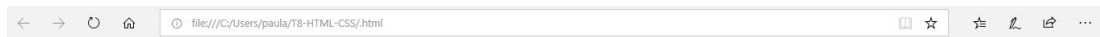
A tag ``<style>`` é usada para definir as informações de estilo para um documento HTML. Dentro do ``<style>`` você especifica características de estilo que devem ser processadas pelo navegador.

Essa tag fica dentro da tag ``<head>`` e um documento HTML pode conter vários ``<style>`` tags.

Um exemplo do seu uso seria:

```
1  <!DOCTYPE html>
2  <html lang="en">
3  <head>
4      <meta charset="UTF-8">
5      <meta name="viewport" content="width=device-width, initial-scale=1.0">
6      <meta http-equiv="X-UA-Compatible" content="ie=edge">
7      <style>
8          h1 {
9              color: blue;
10             font-size: 52px;
11         }
12     </style>
13     <title>Document</title>
14 </head>
15 <body>
16     <h1>
17         Essa é uma página testeeeee
18     </h1>
19 </body>
20 </html>
```

Resultado:



Essa é uma página testeeeee

Link

Outra forma de adicionar estilo ao documento HTML é através do arquivo **.css**. Nesse arquivo será escrito todo o código CSS. Dessa maneira, fica mais organizado e limpo. Depois esse arquivo é importado ao seu arquivo HTML de referência através da tag ``<link>``. Um exemplo de um arquivo css abaixo:


```
# style.css  X
T8-HTML-CSS > # style.css > ...
1  * {
2    box-sizing: border-box;
3  }
4
5  html {
6    width: 100%;
7    height: 100%;
8  }
9
10 body {
11   background-color: white;
12   font-family: 'Lexend Mega', sans-serif;
13 }
14
```

- Importando fontes externas (ex.: Google Fonts)

“A primeira escolha da web, quando ainda nem imagem havia, era usar uma fonte para o estilo do texto que o desenvolvedor teria certeza que o usuário teria em seu computador, com a opção de fallbacks, ou alternativas caso isso não ocorresse. Basicamente, quando o desenvolvedor escolhe, no CSS ou no HTML, uma fonte existente no computador do visitante do site, como Arial, Times (New Roman), Verdana, etc. através de um comando como o abaixo:” ([leia aqui](#))

```
.classe { font-family: Arial, Helvetica, sans-serif; }
```

Com o evoluir da web, hoje podemos adicionar as mais diversas fontes, de maneira que em qualquer navegador seja possível que o usuário a visualize - independentemente se ele possui aquela fonte instalada no seu computador. Um recurso muito usado é o [Google Fonts](#), em que você importa o link da fonte no seu documento HTML e depois o adiciona no arquivo css.

```
<link
href="https://fonts.googleapis.com/css?family=Lexend+Mega&display=swap"
rel="stylesheet">
```

Você pode ler mais sobre fontes [aqui](#).

- Estilo padrão (default) da página

Se eu definir vários estilos, de várias maneiras, qual o navegador irá usar?

Para responder a essa pergunta é necessário primeiro entender que um documento HTML possui um estilo **default** da página. Esse é o estilo padrão que será sempre usado, caso você não configure. Conforme novos estilos são adicionados, através da tag `<style>` ou de folhas de estilo, cria-se uma ordem de precedência, que nada mais é que uma hierarquia pela qual o navegador vai lendo os estilos e exibindo para o usuário (por isso o nome, “*folhas de estilo em cascata*”). Essa é a ordem:

1. **Estilo inline** (dentro de um elemento HTML)
2. **Folha de estilo Incorporada** (definida na tag `<head>` do documento);
3. **Folha de estilo Externa** (importada e linkada);
4. **Estilo padrão** (default) do navegador

- Uso de classes e identificadores nas tags

Classes

Uma classe é reutilizável: pode se repetir na página e também combinar-se com outras (podemos pôr mais de uma classe em um elemento). No arquivo css é chamado a partir do `..`.

ID

Uma id, como o nome diz, é uma identificação única: deve ser utilizada uma vez no documento. Normalmente é utilizada para identificar elementos estruturais da página. No arquivo css é chamado a partir do `#`.

É perfeitamente possível fazer um site apenas com ids, apenas com classes, com uma combinação das duas, ou sem nenhuma das duas. Tudo irá depender do seu estilo de fazer o código das páginas. O importante é definir um padrão e segui-lo.

CLASS	ID
Pouca especificidade Use bastante	Muita especificidade Use apenas uma vez

- Propriedades de estilo no CSS:

- Medidas usadas no CSS (px, %, em e rem)

[Artigo da Alura super completo \(:](#)

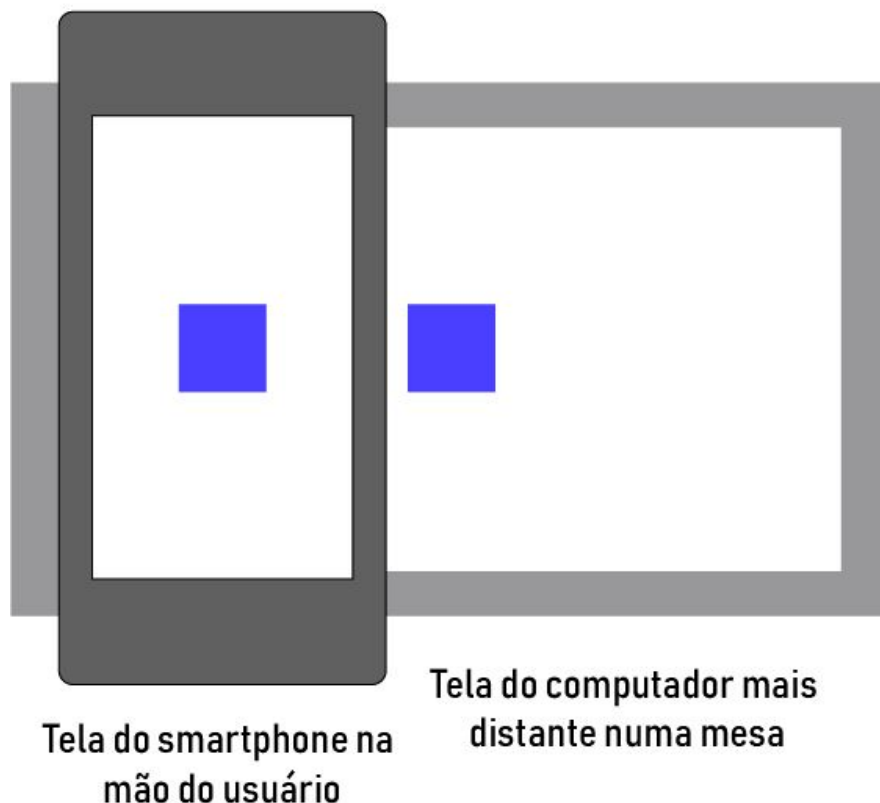
Pontos	Pixeles	Em	%
6pt	8px	0.5em	50%
7pt	9px	0.55em	55%
7.5pt	10px	0.625em	62.5%
8pt	11px	0.7em	70%
9pt	12px	0.75em	75%
10pt	13px	0.8em	80%
10.5pt	14px	0.875em	87.5%
11pt	15px	0.95em	95%
12pt	16px	1em	100%

Medidas absolutas são unidades de medidas definidas pela física, como o **píxel**, centímetro, metro, etc...

Essas medidas são fixas e não mudam de acordo com as especificações do dispositivo. Esse tipo de medida é indicada para quando conhecemos perfeitamente as características físicas e as configurações das mídias onde serão exibidos nossos projetos.

O pixel não é realmente uma medida perfeita, mas ele consegue manter um tamanho relativamente parecido em dispositivos diferentes, dependendo do ângulo.

Visão do ponto de vista do usuário



Medidas relativas essas são as que normalmente não estamos habituados. Essas medidas são calculadas tendo como base uma outra unidade de medida definida, como por exemplo **em** e o **rem** (veremos mais sobre essas duas medidas no decorrer do post). O uso delas é mais apropriado para que possamos fazer ajustes em diferentes dispositivos garantindo um layout consistente e fluido em diversas mídias.

Porcentagem (%)

Apesar de não ser uma unidade de medida, a porcentagem costuma ser bastante utilizada quando falamos de layout responsivo e fluido, por isso, não poderia deixá-la passar.

A porcentagem permite que criemos módulos que sempre vão se readaptar para ocupar a quantidade especificada. Por exemplo, se definirmos um elemento tendo um tamanho de 50%, independente do dispositivo em questão, esse módulo sempre ocupará metade do espaço que lhe cabe (caso esteja dentro de algum outro elemento). Usada em aula

Ems (em)

Difícilmente você achará algum navegador que não tenha suporte para essa medida, que está presente desde os primórdios. Até para o IE, nós teríamos que usar a versão abaixo da 3.0 para que tivéssemos algum problema.

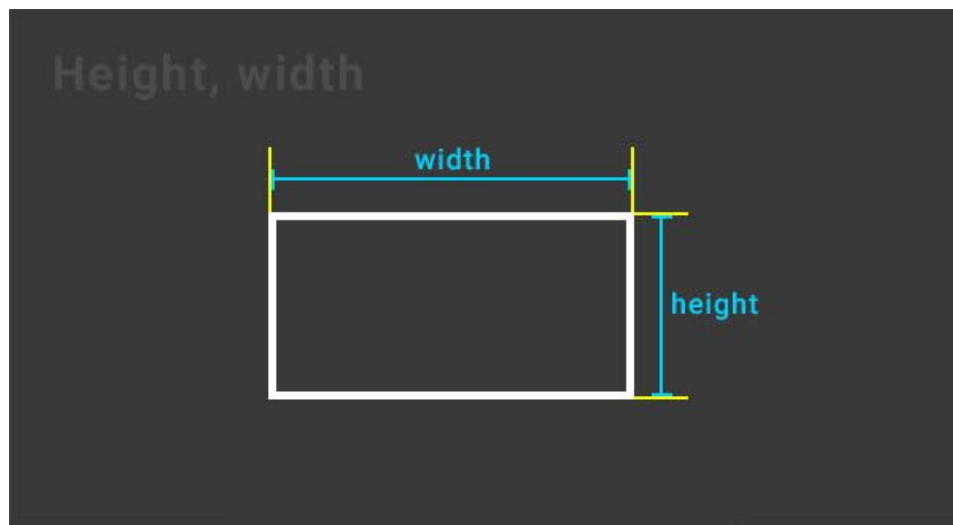
Esse definitivamente é um dos pontos que fazem o em tão popular. O segundo ponto, com certeza se dá a facilidade de criar layouts fluídos e responsivos.

Rems (rem, "root em")

O REM vem como sucessor do EM e ambos compartilham a mesma lógica de funcionamento (font-size), porém a forma de implementação é diferente. Enquanto o em está diretamente relacionado ao tamanho da fonte do elemento pai, o rem está relacionado com o tamanho da fonte do elemento root (raiz), no caso, a tag.

O fato de que o rem se relaciona com o elemento raiz resolve aquele problema que tínhamos com diversas divs (elementos) aninhados, uma vez que não haverá essa "herança" de tamanhos, lembra?! Ou seja, não precisaremos ter dor de cabeça tendo que realizar cálculos, uma vez que nos baseamos na tag raiz.

- Dimensões de elementos



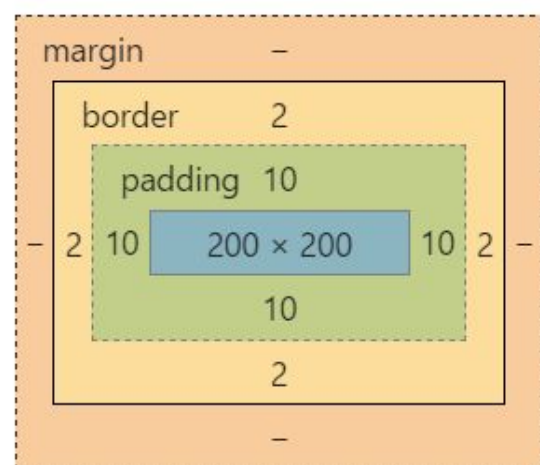
Largura - Width e Altura - Height

O width e o height podem ser definidos como:

Auto - o navegador calcula a altura e largura

Especificado em valores de comprimento - como px, cm, etc., ou em porcentagem (%) do bloco que contém.

- Margens e espaçamentos



No [Developer Mozilla](#) tem uma ótima explicação:

Em uma página WEB, cada elemento é representado como um **box retangular**. Determinar o tamanho, propriedades - como sua cor, fundo, estilo das bordas - e a posição desses boxes é o objetivo do mecanismo de renderização.

No CSS, cada um desses boxes retangulares é descrita usando o **box model padrão**. Este modelo descreve o conteúdo do espaço ocupado por

um elemento. Cada box possui 4 edges: **margin** edge, **border** edge, **padding** edge e **content** edge.

A **área de conteúdo** (content area) é a área ocupada pelo conteúdo real do elemento. Ele frequentemente possui um fundo, uma cor de fonte ou uma imagem (nessa ordem, uma imagem opaca esconde a cor de fundo) e é localizada dentro do content edge; suas dimensões são a largura do conteúdo, ou largura do box de conteúdo, e altura do conteúdo, ou altura do box de conteúdo.

Se a propriedade CSS box-sizing está configurada como padrão, as propriedades CSS **width**, **min-width**, **max-width**, **height**, **min-height** e **max-height** controlam o tamanho do conteúdo.

A **área de preenchimento** (padding area) estende-se para a borda em torno do enchimento. Quando a área de conteúdo tem um fundo, cor ou imagem, isso será estendido para a área de preenchimento, por esse motivo, você pode pensar o preenchimento como a extensão do conteúdo. O preenchimento está localizado dentro do padding edge, e suas dimensões são a largura do padding-box e a altura do padding-box.

O espaço entre os edges de preenchimento e conteúdo podem ser controlados utilizando as seguintes propriedades CSS **padding-top**, **padding-right**, **padding-bottom**, **padding-left** e na forma **generalizada padding**.

A **área de borda** (border area) estende a área de preenchimento para a área que contém as bordas. Esta é a área de dentro do border edge, e suas dimensões são a largura e a altura do border-box. Esta área depende do tamanho da borda que está definido pela propriedade border-width ou pela propriedade border.

A **área de margem** (margin area) estende a área de borda com um espaço vazio utilizado para separar o elemento dos elementos vizinhos. Esta é a área de dentro do margin edge, e suas dimensões são a largura e a altura do margin-box.

O tamanho da área de margem é controlada utilizando as seguintes propriedades CSS **margin-top**, **margin-right**, **margin-bottom**, **margin-left** e na forma **generalizada margin**.

Quando ocorre um colapso de margens, a área de margem não está claramente definida, uma vez que as margens são compartilhadas entre os boxes.

Finalmente, note que, para elementos não substituídos inline, o total de espaço ocupado (para a altura da linha) é determinado pela

propriedade line-height, mesmo que a borda e o padding apareçam visualmente em torno do conteúdo.

- Fontes e suas cores, famílias, tamanhos e ênfases

Algumas propriedades de fontes

font-family:

1. Nome da família de fonte: define-se pelo nome da fonte, exemplos: "verdana", "helvetica", "arial", etc.
2. Nome da família genérica: define-se pelo nome genérico da fonte, exemplos: "serif", "sans-serif", "cursive", etc.

font-size:

1. xx-small
2. x-small
3. small
4. medium
5. large
6. x-large
7. xx-large
8. smaller
9. larger
10. length: medida CSS de comprimento
11. exemplos: px, em, rem, % (porcentagem) ...

font-style:

1. **normal:** fonte normal (em pé)
2. **italic:** fonte inclinada
3. **oblique:** fonte oblíqua

font-variant:

1. **normal:** fonte normal
2. **small-caps:** transforma em maiúsculas de menor altura

font-weight:

1. normal

2. bold
3. bolder
4. lighter
5. 100
6. 200
7. 300
8. 400
9. 500
10. 600
11. 700
12. 800
13. 900

color:

1. **código hexadecimal:** #ffc6d9
2. **código rgb:** rgb(255,235,0)
3. **código rgba:** rgb(255,235,0, 0.7)
4. **código hsl:** hsl(210,100%,40%)
5. **código hsla:** hsla(155,80%,35%,0.4)
6. **palavra-chave:** red, blue, green...etc
7. **transparente:** transparent

[Veja mais.](#)

- Cores de fundo, transparências e gradientes

A propriedade que atribui cores de fundo é a **background-color**. Importante ressaltar que “é uma cor CSS <color> que descreve a cor uniforme do plano de fundo. Mesmo se um ou vários background-image são definidos, essa cor pode afetar a renderização, por transparência se as imagens não forem opacas. No CSS, transparent é uma cor.” [Veja mais.](#)

```
.exampleone {  
  background-color: teal;  
}  
  
.exampletwo {  
  background-color: rgb(153,102,153);  
}  
  
.examplethree {  
  background-color: #777799;  
}
```

A propriedade que atribui gradiente ao fundo é a **background-image**. Existem alguns tipos de gradientes, além dos lineares. Para criar um gradiente linear você deve definir pelo menos duas paradas de cor. paradas de cores são as cores que deseja para tornar transições suaves entre. Também é possível definir um ponto de partida e uma direção (ou ângulo), juntamente com o efeito de gradiente. [Veja mais](#).

```
background-image: linear-gradient(angle, color-stop1, color-stop2);
```

AULA 02

Revisão de conceitos aula anterior

Dúvidas

Introdução HTML e CSS

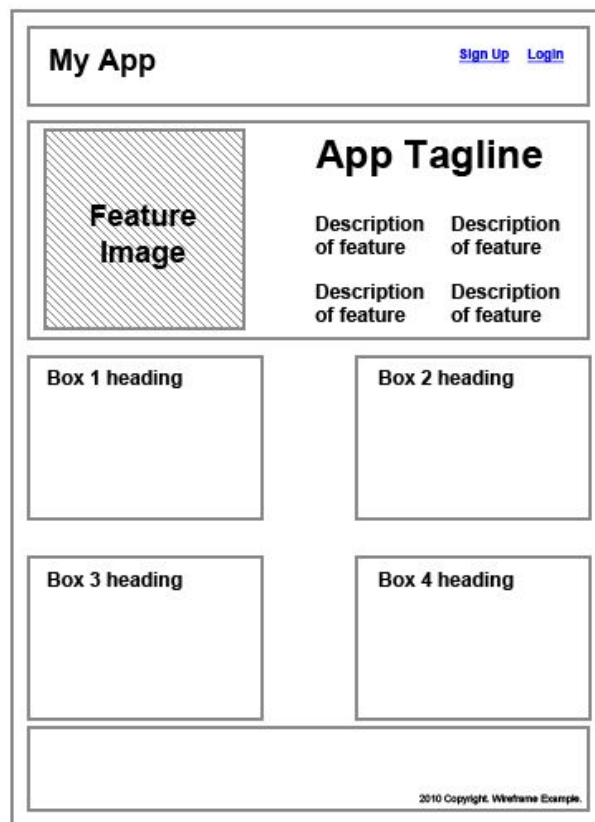
- Projetando a página - wireframe
- Hierarquia de Títulos (`<h1>` a `<h6>`)
- Parágrafos (`<p>`)
- Listas ordenadas e não ordenadas (``, `` e ``)
- Span
- Lorem ipsum (preenchendo espaços de texto)
- Método BEM - como nomear suas classes
- Seletores de elementos, classes e identificadores no CSS
- Seletores por referência e pseudo seletores
- Sobrescrita e precedência no CSS

Mãos no código

Exercício - Landing Page

HTML

- **Projetando a página - wireframe**



Ao dedicar tempo para criar, pelo menos, um wireframe simples, tenha certeza de que seu web design levará em conta todos os elementos das diferentes páginas que compõem o projeto e precisam ir para o design. Além disso, eles estarão posicionados na melhor localização a que o estudo destes elementos chegou.

Além disso, o processo de produção de wireframes é muito mais simples, fácil e barato de ser feito do que quando se “pula” essa etapa e se parte direto para o HTML/CSS: wireframes podem ser facilmente revistos, adaptados ou descartados.

Wireframes x Modelos x Protótipos

Wireframe, modelo (mockup) e protótipo são frequentemente usados como sinônimos, mas são três coisas diferentes (embora, frequentemente, haja alguma sobreposição entre eles). Cada um tem um propósito diferente do outro e sua própria importância no processo de design:

Wireframes. São “ilustrações básicas” da estrutura e componentes de uma página web. Geralmente são o primeiro passo no processo de design (depois da concepção mental, obviamente).

Modelos. Geralmente focam sobre os elementos de design visual do site. São muitas vezes bastante próximos ou idênticos ao web design final efetivo e incluem todos os gráficos, tipografia e outros elementos da página. Mockups geralmente são apenas arquivos de imagem.

Protótipos. São layouts semi-funcionais das páginas e servem para dar um preview de maior fidelidade do site real. Esta fase antecede a

programação da lógica de negócios do site. Enquanto eles não podem ter toda a funcionalidade, eles geralmente dão aos clientes a capacidade de interagir com os elementos e simular a forma como o site irá, eventualmente, trabalhar. Protótipos podem ou não incluir elementos de design finalizado. [Veja mais](#).

- Hierarquia de Títulos (`` a ``)

[Essas tags](#) são usadas para marcar cabeçalhos HTML. <H1> define o título mais importante. <H6> define o título menos importante.

[Veja](#) a ligação dessa hierarquia com técnicas de SEO e acessibilidade.

- Parágrafos (` `)

Define um parágrafo. Os Navegadores adicionam automaticamente algum espaço (margem) antes e depois de cada elemento <p>. [Veja mais](#).

- Span

É usada para agrupar elementos inline em um documento. A tag não fornece nenhuma alteração visual por si só, apenas uma maneira de adicionar um gancho a uma parte de um texto ou a uma parte de um documento.

- Lorem ipsum (preenchendo espaços de texto)

Segundo o [site oficial](#), **Lorem Ipsum** é simplesmente texto fictício da indústria tipográfica e de impressão. O Lorem Ipsum tem sido o texto fictício padrão da indústria desde os anos 1500, quando uma impressora desconhecida pegou uma galé do tipo e subiu para fazer um livro de espécimes de tipo. Ele sobreviveu não apenas cinco séculos, mas também o salto para a composição eletrônica, permanecendo essencialmente inalterado. Foi popularizado nos anos 60 com o lançamento de folhas de Letraset contendo passagens de Lorem Ipsum, e mais recentemente com software de editoração eletrônica como Aldus PageMaker incluindo versões de Lorem Ipsum.

[Lista tem 20 opções alternativas de Lorem Ipsum](#)

- Listas ordenadas e não ordenadas (` ` , ` ` e ` - `)

Listas não ordenadas

As listas não numeradas são usadas para listar itens, sem se preocupar com sua sequência. Chamamos de lista de marcadores apenas.

As tags usadas para criar uma lista não ordenada são e . A tag é usada para definir a lista e a tag é usada para cada item da lista.

```
<ul>
<li>Internet Explorer</li>
<li>Opera</li>
<li>Firefox</li>
<li>Safari</li>
</ul>
```

Listas ordenadas ou numerada

As listas ordenadas ou numeradas são usadas para indicar alguma sequência ou numeração

As tags usadas para criar uma lista não ordenada são e . A tag é usada para definir a lista e a tag é usada para cada item da lista.

```
<ol>
<li>São Paulo</li>
<li>Rio de Janeiro</li>
<li>Belo Horizonte</li>
<li>Brasília</li>
</ol>
```

CSS

- Método BEM - como nomear suas classes

Forma de nomear as classes do CSS seguindo um padrão fixo baseado em Blocos, Elementos e Modificadores. Oferece 3 principais benefícios:

- Traduz claramente a estrutura da página

- Torna possível ter estilos reutilizáveis, sem precisar ficar repetindo os mesmos estilos em vários elementos
- Torna seu código modular, ou seja, evita que estilos sejam passados "sem querer" por meio da cascata.

Para seguir o padrão BEM, basta seguir o seguinte formato para nomear suas classes:

`bloco__elemento--modificador`

Bloco é tudo aquilo que representa um componente independente da página. Possui um sentido se usado sozinho e pode ser reutilizado. *Ex: menu, botão, formulário*

Elemento é tudo aquilo que ajuda a construir o componente independente. Ou seja, se colocado sem uma "mãe" e sem "irmãos" não funciona bem. *Ex: item da lista, campo do formulário*

Modificador é tudo aqui que varia entre elementos ou componentes que podem ser iguais. Por exemplo, quando temos vários botões em uma página e um deles tem a cor diferente. Ele continua sendo um botão, mas precisa ser diferenciado dos outros. É sempre uma característica ou estado. *Ex: botão vermelho, menu horizontal ou vertical*

Importante: podemos usar modificadores tanto com elementos quanto com blocos diretamente.

Exemplos válidos:

`conteudo__item`
`conteudo__item-visitado`

`menu__item`
`menu--horizontal`

- Seletores de elementos, classes e identificadores no CSS**

Se você já assistiu Harry Potter, provavelmente se lembra do chapéu seletor que era colocado na cabeça de alguma pessoa bruxinha e dizia para qual casa essa pessoa iria. Lembrou?



No documento HTML os seletores vão funcionar de maneira parecida. Eles serão adicionados em alguma tag e, dentro do arquivo css, irão atribuir alguma propriedade (que possui algum valor). Algo como:



Seletor de classe: seleciona elementos com uma classe específica aplicada. Exemplo: **.destaque** seleciona todos os elementos com a classe "destaque".

Seletor de id: seleciona o elemento com a id especificada. Exemplo: **#cabecalho** irá selecionar o elemento com a id "cabecalho". Cada id é única e não pode ser repetida no mesmo documento.

Seletor de tipo: este é o tipo de seletor que utilizamos nos exemplos até agora. Com este seletor, selecionamos todas as tags de um mesmo tipo. Por exemplo, se digitarmos `<a>` estaremos selecionando todas as tags a (links) da página e poderemos aplicar estilos a elas. Útil para estilos gerais, mas para maior especificidade utilizamos outros seletores.

Seletor descendente: com este seletor, podemos escolher um ou mais elementos que estão dentro de outro, ou seja, que são descendentes do elemento principal. Exemplo: **p strong**. Com isso, selecionamos apenas tags strong que estão dentro de parágrafos. Podemos selecionar com ainda mais especificidade, escrevendo mais elementos, como: `div p strong` a. Neste exemplo, selecionamos links que estão dentro de tags strong que estão dentro de parágrafos que estão dentro de tags div.

Seletores por referência:

Pseudo seletores: uma pseudo-classe CSS é uma palavra-chave adicionada a seletores que especifica um estado especial do elemento selecionado. Veja [aqui](#) um exemplo. Segue abaixo a sintaxe.

```
seletor:pseudo-classe {  
    propriedade: valor;  
}
```

Tabela para seletores CSS

- Sobrescrita e precedência no CSS

Os seletores permitem, além da estilização dos elementos, a criação de padrões complexos de identificação de elementos. Um exemplo

AULA 03

Revisão de conceitos aula anterior

Dúvidas

Introdução HTML e CSS

- Links `` e atributo `target`

- Usando # para criar dead links

- Links externos e links internos

- Formulários (form, input, placeholders, checkbox, radio buttons, buttons)

- Propriedades de CSS

 - Alinhamento de elementos

 - Propriedades display (none, block, inline, inline-block e flex)

 - Position absolute e relative

Mãos no código

- Exercício** - Coding Dojo: form

HTML

- Links <a> e seu atributo e target

Outro atributo importante da tag <a> é o **target** que especifica onde abrir o documento vinculado, o valor "_blank" faz com que o link seja aberto em uma nova guia.

- Usando # para criar dead links

Substitua o valor do atributo href, na tag <a> por um #, também conhecido como um símbolo de hash, para criar um link inativo.

Por exemplo: href = "#"

- Links externos e links internos

Links internos são aqueles que apontam para uma página dentro do próprio site. São exemplos de links internos as páginas produtos, empresa, serviços, contato, localização, etc.

```
<a href="produtos.html">Página de produtos</a>
<a href="servicos.html">Página de serviços</a>
```

Links externos são aqueles que apontam para fora do próprio site. Quando criamos um link para outros sites estamos criando links externos. Tecnicamente links internos e externos são iguais, mudando apenas o destino dos mesmos.

```
<a href="https://www.google.com.br">Uol</a>
```

No caso de links externos é importante usarmos o atributo **target="_blank"** que indica que a página será aberta em uma nova janela.

```
<a target="_blank" href="https://www.google.com.br">Google</a>
```

É importante observar que ao fazermos um link externo, usamos o endereço completo da página destino. Isto é necessário pois caso informássemos apenas **www.google.com.br**, por exemplo, ele iria tentar abrir o site do Google dentro do nosso próprio site, portanto o https:// indica ao link para fazer uma busca externa.

- Formulários (form, input, placeholders, checkbox, radio buttons, buttons)

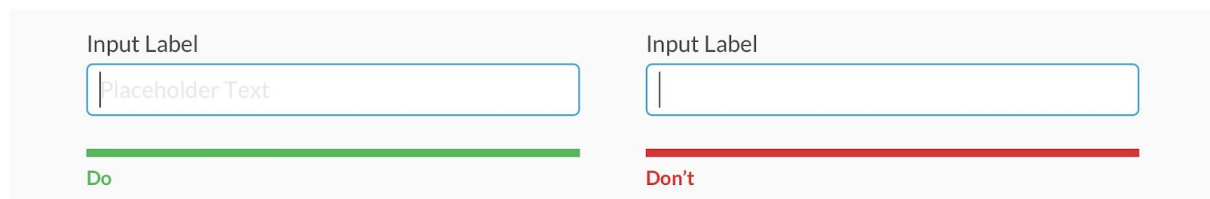
O `**<form>**` elemento HTML define um formulário que é usado para coletar a entrada do usuário:

```
<form>
  .
  form elements
  .
</form>
```

Um formulário HTML contém elementos de formulário. Os elementos de formulário são tipos diferentes de elementos de entrada, como campos de texto, caixas de seleção, botões de opção, botões de envio e muito mais.

O `**<input>**` elemento é o elemento de formulário mais importante. Pode ser exibido de várias maneiras, dependendo do atributo **type** (como text, radio, checkbox e submit).

O atributo **placeholder**, da `**<input>**`, especifica uma breve dica que descreve o valor esperado de um campo de entrada (por exemplo, um valor de amostra ou uma breve descrição do formato esperado).



The image shows two side-by-side input fields. The left field is labeled 'Input Label' and contains the text 'Placeholder Text' in a light gray font. Below it is a green progress bar and the word 'Do' in green. The right field is also labeled 'Input Label' but is empty. Below it is a red progress bar and the word 'Don't' in red.

CSS

-Propriedades de CSS

- Alinhamento de elementos

https://www.w3.org/Style/Examples/007/center.pt_BR.html

- Propriedades display (none, block, inline, inline-block e flex)

Display é a propriedade mais importante do CSS para controlar o layout. Cada elemento tem um valor padrão para o display dependendo de seu tipo. O valor padrão na maioria dos elementos é normalmente block ou inline. Um elemento com valor block é chamado de elemento de nível de bloco ou apenas elemento de bloco. Um elemento com valor inline é sempre chamado de elemento de linha.

Block

Div é o exemplo de bloco mais comum. O elemento de bloco sempre começa em uma nova linha e se expande pra esquerda e direita o tanto quanto for possível. Outros elementos de bloco comuns são p e form, e agora no HTML5 temos: header, footer, section e outros.

</div>

Inline

Span é o elemento de linha padrão. Um elemento de linha pode preencher algum texto dentro de um parágrafo como esse sem quebrar o fluxo daquele parágrafo. O elemento a é o elemento de linha mais comum, já que ele é utilizado para links.

None

Outro valor comum do display é o none. Alguns elementos específicos como o script utilizam este valor por padrão. Ele é normalmente utilizado através do JavaScript para esconder e exibir elementos sem realmente os remover nem recriar.

Ele é bem diferente de visibility. Marcando o display com o valor none vai exibir a página como se o elemento não existisse. Com visibility:hidden o elemento fica invisível, porém ele permanece ocupando o espaço em que estaria totalmente visível.

Inline-block

Você pode criar um grid de caixas que preenchem a largura do navegador e quebram suavemente. Isso tem sido possível por muito tempo utilizando o float, mas agora com o valor inline-block da propriedade display é bem mais fácil. Vejamos exemplos de ambas as abordagens.

<http://pt-br.learnlayout.com/display.html>

<http://pt-br.learnlayout.com/inline-block.html>

Flex

O novo modo de layout flexbox está pronto para redefinir como fazemos layouts com CSS. Infelizmente a especificação mudou bastante recentemente, então ele está implementado de forma diferente em diversos navegadores. Ainda assim, eu gostaria de compartilhar alguns exemplos para que você saiba o que está por vir. Esses exemplos só funcionam em alguns navegadores que utilizam a última versão do padrão.

Há vários recursos desatualizados sobre flexbox por aí. Se você quiser saber mais sobre Flexbox, comece aqui para aprender como identificar se o recurso é atual ou não. Eu também escrevi um artigo detalhado com a última versão do padrão.

<https://flexboxfroggy.com/>

- Position absolute e relative

Com a finalidade de desenvolvermos layouts mais complexos, precisamos discutir sobre a propriedade position. Esta propriedade possui diversos valores possíveis e seus nomes não fazem sentido e são impossíveis de lembrar. Vamos ver um por um, mas talvez seja bom você também marcar esta página.

O valor **static** é o valor padrão de todos os elementos HTML. Um elemento com position: static; não se posiciona de maneira especial, seria o mesmo que dizer que o elemento não tem posição definida ou então que um elemento com o atributo position definido seria posicionado.

Relative se comporta igualmente ao static, a menos que se adicione propriedades extras no estilo do elemento. Definindo as propriedades **top**, **right**, **bottom**, e **left** em um elemento position:relative o ajustamos em relação à posição atual dele. Outros conteúdos não vão se ajustar para se encaixarem em qualquer espaço deixado por esse elemento.

O valor **absolute** é o mais complicado. Este valor se comporta como o fixed, porém tendo como referência a posição do elemento relativo mais próximo de onde está contido, ao invés do viewport. Se um elemento position:absolute não possuir elementos ancestrais posicionados relativamente, ele utilizará o body como referência.

<http://pt-br.learnlayout.com/position.html>

AULA 04

Revisão de conceitos aula anterior

Dúvidas

Fundamentos HTML e CSS

- Tabelas (thead, tbody, td, th e tr)

- Tags semânticas (header, nav, footer, sections, video, article, entre outras)

- Propriedades de estilo no CSS:

 - Propriedade border-box

 - Cores RGB, RGBA e HEX Code

Mãos no código

Exercício - Tabela

HTML

- Tabelas (thead, tbody, td, th e tr)

Uma tabela HTML com um elemento <thead>, <tbody> e <tfoot>:

```
<table>
  <thead>
    <tr>
      <th>Month</th>
      <th>Savings</th>
    </tr>
  </thead>
  <tbody>
    <tr>
      <td>January</td>
      <td>$100</td>
    </tr>
    <tr>
      <td>February</td>
      <td>$80</td>
    </tr>
  </tbody>
  <tfoot>
    <tr>
      <td>Sum</td>
      <td>$180</td>
    </tr>
  </tfoot>
</table>
```

A tag `**<thead>**` é usada para agrupar o conteúdo do cabeçalho em uma tabela HTML.

O elemento `**<thead>**` é usado em conjunto com os elementos `**<tbody>**` e `**<tfoot>**` para especificar cada parte de uma tabela (cabeçalho, corpo, rodapé).

Os navegadores podem usar esses elementos para ativar a rolagem do corpo da tabela, independentemente do cabeçalho e do rodapé. Além disso, ao imprimir uma tabela grande que abrange várias páginas, esses elementos podem permitir que o cabeçalho e o rodapé da tabela sejam impresso na parte superior e inferior de cada página.

A tag `<thead>` deve ser usada no seguinte contexto: Como um filho de um elemento `<table>`, depois de qualquer elemento `<caption>` e `<colgroup>`, e antes de qualquer `<tbody>`, `<tfoot>` e `<tr>` elementos.

Nota: O elemento `<thead>` deve ter uma ou mais tags `<tr>` internas.

Dica: Os elementos `<thead>`, `<tbody>` e `<tfoot>` não afetarão o layout da tabela por padrão. No entanto, você pode usar CSS para estilizar esses elementos.

A tag `<th>` define uma célula de cabeçalho em uma tabela HTML. Uma tabela HTML possui dois tipos de células:

- Células de cabeçalho - contém informações de cabeçalho (criadas com o elemento `<th>`)
- Células padrão - contém dados (criados com o elemento `<td>`). O texto nos elementos `<th>` ficam em negrito e centralizados por padrão.

O texto nos elementos `<td>` são regulares e alinhados à esquerda por padrão.

- Tags semânticas (header, nav, footer, sections, video, article, entre outras)

O `<header>` é utilizado para representar o cabeçalho de um documento ou seção declarado no HTML. Nele podemos inserir elementos de `<h1>` a `<h6>`, até elementos para representar imagens, parágrafos ou mesmo listas de navegação.

O elemento `<nav>` é utilizado quando precisamos representar um agrupamento de links de navegação, que, por sua vez, são criados com os elementos ``, `` e `<a>`.

O elemento `<footer>` representa um rodapé de um documento, como a área presente no final de uma página web. Normalmente é utilizado para descrever informações de autoria, como nome e contato do autor, e data de criação do conteúdo.

O elemento `<section>` representa uma seção dentro de um documento e geralmente contém um título, o qual é definido por meio de

um dos elementos entre `**<h1>**` e `**<h6>**`. Podemos utilizar o `<section>`, por exemplo, para descrever as seções/tópicos de um documento.

Utilizamos o elemento `**<article>**` quando precisamos declarar um conteúdo que não precisa de outro para fazer sentido em um documento HTML, por exemplo, um artigo em um blog. É recomendado identificar cada `<article>` com um título.

[HTML Semântico: Conheça os elementos semânticos da HTML5](#)

CSS

- Propriedades de estilo no CSS:

- Propriedade border-box

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. É possível usar essa propriedade para emular o comportamento dos navegadores (browsers) que não suportam corretamente a especificação da propriedade CSS box model.

content-box

Essa é o estilo padrão, conforme especificado pela norma CSS. As propriedades width (largura) e height (altura) são medidas incluindo só o conteúdo, mas não o padding, border ou margin. Nota: Padding, border e margin serão fora da box. Exemplo.: Se `.box {width: 350px}` então se você aplicar uma propriedade `{border: 10px solid black;}` o resultado renderizado no navegador (browser) será `.box {width: 370px;}`

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. [Veja mais.](#)

- Cores RGB, RGBA e HEX Code

Não há diferenças entre uma cor RGB e hexadecimal.

Mas com o rgba (alpha) você pode adicionar uma variável alpha que adiciona uma opacidade à sua cor.