

COLEÇÃO – PROFISSIONALIZANTE – EQUIPE PRAXIS

Coordenação Geral

Maiara Caroline Braz Sobrinho

Gerência de Projetos

Priscila Isaías da Silva

Projeto Gráfico, Editoração e Revisão

Matheus Carrara de Oliveira Gerth

Montagem da Capa

Leonardo Moreira Campos

AUTOR

Professor Alessandro Oliveira

2023

Caro estudante,

Estamos muito felizes em te receber e compartilhar com você nosso conhecimento e experiências, para que você tenha um ótimo proveito no curso em que está matriculado.

Nossos cursos têm por objetivo prepara-lo para o mercado de trabalho de forma abrangente, nos quais são aplicados conteúdos específicos da profissão em que você se formará, e também outros conteúdos de suma importância para a atuação no mercado, complementando assim sua formação profissional.

Enfatizamos que é importante você participar de todas as atividades propostas pelos seus instrutores e pela equipe de coordenação da Equipe Praxis. Você terá aulas teóricas em sala de aula, e também participará de workshops, oficinas, eventos e atividades práticas promovidas pela Equipe Praxis. Aqui a teoria e a prática andam juntas!

Aproveite cada momento em sala de aula, tire dúvidas, questione, pergunte e avalie! Assim poderá absolver o máximo de conhecimento estando preparado para o mercado de trabalho cada dia mais competitivo.

A média de formando que consegue emprego nos meses subsequentes a sua formatura é de 52% e com você estando na nossa lista de formandos temos a certeza que aumentaremos ainda mais nossa lista de empregabilidade.

Desejamos a você bons estudos!

Equipe Praxis

Sumário Designer Gráfico (web)

O HTML 5.....	7
O CSS3.....	8
O que é hipertexto.....	9
O que é um navegador.....	9
O que é uma página Web.....	10
Padrões web e especificações.....	11
Estrutura Básica de uma Página Web.....	12
Sintaxe e semântica do HTML.....	13
Elementos, atributos e Valores.....	16
A importância do UTF-8.....	17
Estrutura do documento HTML.....	19
Os novos elementos de HTML5.....	20
Header.....	21
Footer.....	21
Nav.....	21
Article.....	21
Aside.....	22
Section.....	22
Figure.....	22
Figcaption.....	22
Sua Primeira página na web.....	22
Adicionando um Título.....	23
Adicionando cabeçalhos.....	24
Adicionando Parágrafos.....	25
Adicionando imagens.....	26
Exercícios.....	27
Textos.....	30

Texto importante ou enfatizado.....	31
Citações.....	32
Abreviações.....	32
Alterações.....	32
Outras tags para modificar o texto.....	34
Exercícios.....	36
Imagens.....	37
Links.....	39
Atributos do elemento link.....	40
Criando Âncoras.....	40
Exercícios.....	42
Listas.....	43
Lista de descrições.....	43
Lista Ordenada.....	44
Lista não ordenada.....	45
Exercícios.....	46
Estrutura de uma tabela em HTML.....	49
Explicando cada tag da tabela.....	50
Exercícios.....	53
Parâmetros GET e POST.....	54
Formulários.....	55
Placeholders.....	59
Caixas de entrada específicas.....	60
Caixas de busca.....	60
Caixas de números.....	61
Caixas de e-mail, telefone e url.....	62
Caixas de datas e horas.....	63
Caixas de senha.....	64
Caixas de texto longo.....	65
Check boxes e Radios.....	65
Botões.....	66
Botões de reset.....	67

Botões de upload.....	67
Imagem como botão de submit.....	68
Lista Drop-down.....	68
Validação.....	69
Exercícios.....	70
CSS.....	71
Sintaxe básica.....	71
Seletor ID.....	71
Seletor class.....	73
Unidades para valores.....	74
Declaração única.....	75
Formas de se utilizar o CSS.....	76
Cores.....	81
Cor gradiente.....	84
Principais propriedades para textos.....	88
Shadow efeito de sombra.....	92
Exercícios.....	96
Propriedade para dimensões.....	96
Para fundos do site e de blocos.....	97
Para margens.....	99
Para preenchimentos.....	100
Para bordas.....	104
Display.....	106
Position.....	112
Formatação de links.....	113
Formatando o link como botão.....	114
Box sizing.....	117
@media layouts responsivos.....	120
Flexbox.....	124
Elementos.....	125
Propriedades para o elemento pai.....	127
Display.....	128

Flex direction.....	128
Flex wrap.....	129
Flex flow.....	130
Justify content.....	130
Align items.....	132
Align content.....	133
Propriedades para elementos filhos.....	135
Order.....	135
Flex Grow.....	136
Flex shrink.....	137
Flex.....	137
Align self.....	138
Projeto.....	139
Exercícios.....	155
Referências Bibliográficas.....	171

O HTML 5

O HTML começou no início de 1990 como um pequeno documento que detalhava diversos elementos usados para construir páginas web. Muitos desses elementos era para descrever conteúdos de páginas como cabeçalhos, parágrafos e listas. O número de versões de HTML aumentou conforme a linguagem evoluiu com a introdução de outros elementos e ajustes as suas regras. A versão mais atual é HTML5.

O HTML5 é uma evolução natural das versões anteriores do HTML e luta para refletir as necessidades tanto dos sites atuais quanto dos futuros sites. Ele herda a grande maioria dos recursos de suas predecessoras versões, fazendo com que, se você já codificava em HTML antes do HTML5, já saiba muito sobre esta. Isso também significa que muito do HTML5 funciona em navegadores novos e antigos. Ser compatível com versões anteriores é um princípio de design chave do HTML5.

O HTML 5 também adiciona um monte de novos recursos. Muitos são bem diretos, como elementos adicionais que são usados para descrever conteúdo. Outros são bem complexos e ajudam a criar aplicações web poderosas. Você precisará ter uma noção concreta sobre a criação de páginas na web antes que possa avançar para os recursos mais complexo que o HTML 5 oferece. Ele também introduziu áudio próprio e playback de vídeo para suas páginas.

O CSS3

CSS é uma linguagem para formatação de documentos de marcação, tais como HTML ou XML, arquivos CSS são denominados como folhas de estilo, pois, contém todo código para formatação visual / estilo da página web. Saber trabalhar bem com CSS é fundamental para que possamos formatar bem uma página web, hoje em dia, nos padrões atuais de desenvolvimento web, não usamos mais códigos “misturados”, ou seja, tipos diferentes de linguagem como HTML e CSS misturados pelo código afora, tudo no seu devido lugar, separados, isso facilita no desenvolvimento, manutenção e portabilidade do código.

A função principal das CSS é, justamente, extrair a formatação de uma página do código HTML, separando-a do conteúdo propriamente dito (informações). Além de aumentar o nível de organização, isso indica que elas podem definir, de antemão, a formatação de todos os elementos de uma ou várias páginas. Com isso, torna-se muito mais fácil manter um padrão de fontes, cores e estilos, na medida em que será mais prático modificar tais atributos, pois tal atividade será executada a partir do tipo de elemento ou classe (e não da informação). Toda alteração nos elementos e atributos das CSS modificam toda a página que a utiliza, dispensando a alteração tag a tag. Essa estilização também inclui mudança de propriedades em função de eventos. Então vamos começar que temos muito a aprender.

O QUE É HIPERTEXTO

Desde milhares de anos, o homem tem escrito textos em diferentes formatos e sobre diferentes superfícies. Temos sido

testemunhas da história da humanidade através de conteúdos gravados sobre pedras, sobre madeira e mais recentemente sobre papéis e outros tipos de materiais sintéticos. Porém, no final das contas, sem importar a forma ou o material sobre o qual os mesmos foram escritos, a informação sempre foi tratada de maneira estática, com apenas uma forma de ser lida. A chegada dos computadores deu uma reviravolta nesse conceito.

Ao falar de hipertexto nos referimos a textos interligados entre eles, de modo a nos permitir outras possibilidades além da leitura linear (única opção possível em um livro físico).

Além dessa funcionalidade, o hipertexto suporta outros conteúdos (não somente palavras) entre eles: imagens, sons, vídeos, entre outros.

O QUE É UM NAVEGADOR

Um navegador (mais especificamente um navegador Web) é um programa que funciona em um computador ou algum outro tipo de dispositivo para ler e interpretar arquivos acessados pela Internet.

Os navegadores, basicamente funcionam interpretando código do tipo hipertexto e mostrando os resultados ao usuário de uma forma amigável.

Quando surgiram, os navegadores interpretavam basicamente textos com um ou outro elemento gráfico e links para ir de um texto a outro. Estes tipos de funcionalidades têm crescido bastante desde então.

Os navegadores Web mais conhecidos (e mais usados) na atualidade são os seguintes:

- Chrome
- Edge
- Opera
- Firefox
- Safari

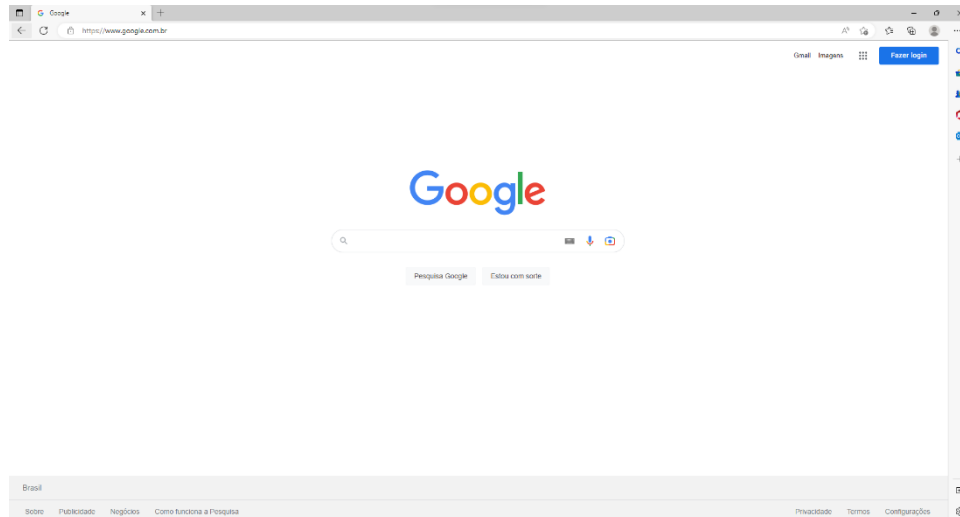
O QUE É UMA PÁGINA WEB

O que é uma página Web?

Uma página Web não é nada mais que um documento publicado na Rede Mundial de Computadores (WWW), que pode ser aberto usando um navegador. Essas páginas estão escritas em HTML com adições de linguagens específicas como CSS, JavaScript ou VBScript.

As páginas Web podem ser armazenadas localmente, isto é, dentro do equipamento onde se encontra o leitor, em alguma rede privada, ou em algum servidor remoto. O acesso a página pode ser público ou restrito. Da perspectiva da implementação de sites do lado do servidor, existem dois tipos de páginas web: estática e dinâmica. As páginas estáticas são recuperadas do sistema de arquivos do servidor web sem nenhuma modificação, enquanto as páginas dinâmicas devem ser criadas pelo servidor web em tempo real, normalmente utilizando um banco de dados para preencher um template web, antes de serem enviadas ao navegador do usuário

Página web também conhecido por navegador:



PADRÕES WEB E ESPECIFICAÇÕES

Em primeiro lugar você deve estar se perguntando quem criou o HTML e o

CSS, e quem continua a desenvolvê-los. O World Wide Web Consortium (W3C) dirigido pelo inventor da web e HTML, Tim Berners-Lee, é a organização responsável por guiar o desenvolvimento dos padrões web.

As especificações são documentos que definem os parâmetros de linguagens como HTML e CSS. Em outras palavras, as especificações padronizam as regras. Por diversas razões, outra organização, Web Hypertext Application Technology Working Group, desenvolve as especificações do HTML5. O W3C incorpora trabalho da WHATWG em sua versão oficial das especificações em progresso.

Com padrões definidos, nós podemos construir nossas páginas a partir de regras preestabelecidas, e navegadores como Chrome, Firefox, Edge, Opera e Safari, já mencionados, podem ser construídos para mostrar nossas páginas com aquelas regras em mente.

ESTRUTURA BÁSICA DE UMA PÁGINA WEB

Com a evolução da web, as páginas se tornam cada vez mais complexas, mas sua estrutura ainda permanece simples. A primeira coisa que você deve saber é que é impossível criar uma página web sem HTML, pois ele é responsável por descrever o conteúdo da página para o navegador processá-la de forma correta. Uma página web é composta basicamente por três elementos:

Conteúdo de texto: Simplesmente o texto aparece na página para demonstrar aos visitantes o conteúdo do seu site.

Referência a outros arquivos: Elas carregam itens como imagens, áudios, vídeos, links, folhas de estilo (responsável pelo layout), JavaScript e outros componentes que conectam o seu site a outros sites ou que dão funcionalidades extras ao código.

Marcadores: Os elementos HTML que descrevem o conteúdo de seu texto e fazem as referências funcionarem.

A maioria dos componentes da página web são basicamente textos com códigos HTML que não são vistos pelo usuário na visualização da sua página, eles são salvos com formato específico para que rodem em qualquer navegador, seja ele desktop (para computadores) ou mobile (para celular).

Os comandos em HTML são chamados de Tags, eles irão dizer ao navegador como o texto, a informação e as imagens serão formatadas e exibidas. Por exemplo, uma Tag pode dizer que um

determinado texto será exibido em negrito, itálico e com um tipo de fonte qualquer.

Exemplificando:

```
<!doctype html>
<html lang="pt-br">
  <head>
    <title>Título da página</title>
    <meta charset="UTF-8">
  </head>
  <body>
    <h1>Meu Título</h1>
    <p>Lorem ipsum dolor sit amet, consectetur adipiscing elit.
    Tempora, corporis?</p>
  </body>
</html>
```

SINTAXE E SEMÂNTICA DO HTML

E no HTML5 que usamos algumas práticas recomendadas:
Utilizar todos os elementos e atributos em minúsculas;
Incluir tag de abertura de fechamento para as tags que possuem conteúdo (no caso de tags órfãs);
Pode se inserir todos os valores de **propriedades** entre **aspas duplas**;

Sintaxe básica do HTML:

< tag > sinal de menor + tag + sinal de maior.

</tag> sinal de menor + barra de fechamento + tag + sinal de maior.

O HTML é um sistema inteligente para incluir marcações sobre o conteúdo em um documento de texto. Essas marcações têm como função descrever o significado de cada conteúdo, ou seja, possui funções de semântica. Diferente de outras versões anteriores, o HTML não define como os elementos visuais irão se apresentar para o usuário, este é um papel para o CSS.

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>Document</title>
</head>
<body>
  <article>
    <h1>Meu Título</h1>
    <p>Lorem ipsum dolor sit amet, adipisicing elit. Temporaporis?</p>
  </article>
</body>
</html>
```

Então revendo nosso código, temos um elemento que está abrangendo todo o nosso conteúdo, esse tal elemento é o article. Ele tem a função de definir um pedaço distinto do conteúdo, ou seja, ele deve ser usado quando um conteúdo é independente, autossuficiente. Um artigo deve fazer sentido por conta própria e que deve ser possível para distribuí-lo de forma independente do resto do site.

Logo em seguida vem o título do artigo, a tag <h> possui seis níveis de h1 até h6, com o h1 sendo o mais importante, o h2 é um subtítulo e assim por diante, funcionando da mesma forma quando você digita um documento com seus títulos e subtítulos em um editor de texto.

Depois temos o elemento p que define um parágrafo. Ele pode conter uma única sentença ou várias delas. Se fosse

necessário outro parágrafo, basta adicionar outra tag p e escrever o conteúdo dentro da tag.

Então, você já tem ideia da importância da semântica no HTML e se aplicada corretamente ela lhe trará diversos benefícios como:

Acessibilidade melhorada e interoperabilidade (o seu conteúdo vai estar disponível para tecnologias assistivas a visitantes com deficiências, e a navegadores desktop e para outros aparelhos móveis). Melhorias na otimização para sites de busca (SEO), ou seja, seu site ficará melhor ranqueado no Google.

Código mais leve e por consequência páginas mais rápidas. Manutenção e estilização de código mais fácil.

ELEMENTOS, ATRIBUTOS E VALORES

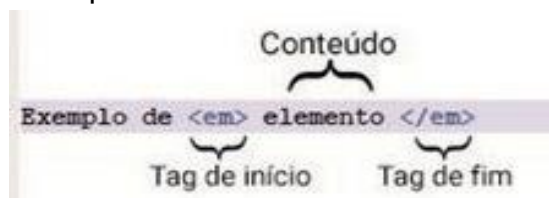
O HTML tem três componentes de marcação principais: elementos, atributos e valores.

Elementos

São como pequenos rótulos que descrevem as diferentes partes de uma página web, foram mostrados alguns deles em páginas anteriores como por exemplo a tag **<p>**. Alguns deles possuem um ou mais atributos, que descrevem o propósito e conteúdo do elemento.

Os elementos podem conter textos e outros elementos aninhados, ou simplesmente estarem vazios, ou seja, sem nenhum conteúdo entre a tag de abertura e de fechamento. Um elemento não-vazio consiste em uma tag de início, o conteúdo e a tag de fechamento.

Exemplo de elemento não-vazio:

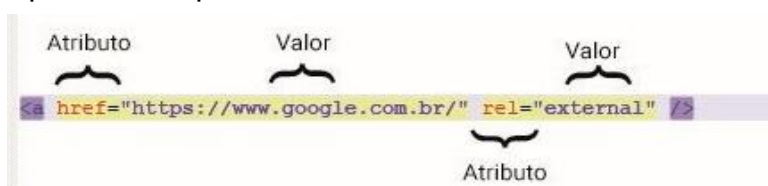


Exemplo de elemento vazio:

```

```

É comum digitar os nomes dos elementos em caixa baixa, embora o HTML5 não seja exigente quanto a isso, também permitindo o uso de caixa alta. No entanto no mercado de trabalho é raro encontrar alguém que codifique em maiúsculas pois trata-se de uma prática ultrapassada.



Atributos e valores

Os atributos contêm informações sobre o conteúdo no documento, ao contrário de estar contido em si mesmo. No HTML5, um valor de atributo pode opcionalmente vir entre aspas, mas é costume incluí-las, então é recomendável que você sempre o faça, e assim como os nomes dos elementos, use caixa baixa ao codificá-los.

Pais e Filhos

Se um elemento contém outro, ele é considerado pai do elemento anexo, ou filho. Quaisquer elementos contidos no

elemento filho são considerados descendentes do elemento pai. Em uma página você constrói uma relação hierárquica entre os elementos o que gera uma identificação de importância para eles. Com isso, o trabalho de estilização e aplicação do comportamento JavaScript é facilitado. Nesses casos é importante lembrar que o

```
<article>
  <h1> Exemplo de texto </h1>
  <p> Exemplo de parágrafo longo, <em> muito longo </em>, mas longo mesmo </p>
  <p> Tem outro aqui também </p>
</article>
```

elemento filho deve estar entre as tags de abertura e fechamento do elemento pai.

A IMPORTÂNCIA DO UTF-8

O texto contido dentro dos elementos é o ingrediente básico de uma página. Quando o navegador renderizam HTML, ele transforma espaços extras ou tabs em um único espaço. Outra restrição é que o HTML em sua forma original, processa os caracteres em ASCII, que basicamente são as letras da língua inglesa, numerais e alguns símbolos comuns, o que gera um certo trabalho para nós brasileiros que precisamos de acentos nas palavras e outros símbolos que não são tão comuns na língua inglesa. Para que esse problema seja corrigido é uma prática padrão utilizar o Unicode UTF8, pois ele aceita nossos caracteres. Para inserir o UTF-8 é necessário que dentro da sessão head do site seja adicionado o elemento meta. Este elemento é um metadado. (dados (informações) sobre os dados). A tag <meta> fornece metadados

sobre o documento HTML. Metadados não serão exibidos na página, mas serão passados ao navegador.

Os elementos meta são normalmente utilizados para especificar descrição da página, palavras-chave, autor do documento, modificar cada pela última vez, e outros metadados.

Eles podem ser usados por navegadores (como exibir conteúdo ou página(reload)), motores de busca (palavras-chave), ou outros serviços da web. O código para inserir o utf-8 seria mais ou menos assim:

```
<head>  
  <meta charset= "utf-8">  
</head>
```

Exemplo de página sem o UTF-8:

Exemplo de texto

Exemplo de parágrafo longo, *muito longo* , mas longo mesmo

Tem outro aqui também

Exemplo de página com UTF-8

Exemplo de texto

Exemplo de parágrafo longo, *muito longo* , mas longo mesmo

Tem outro aqui também

ESTRUTURA DO DOCUMENTO HTML

Veja na imagem a seguir a estrutura básica do arquivo em HTML 5:

```
<!DOCTYPE html>
<html lang="pt-br">
<head>
  <meta charset="UTF-8">
  <title>Document</title>
</head>
<body>
</body>
</html>
```

A `<!DOCTYPE html>` representa uma declaração/Tag que define este documento como HTML.

A Tag `<html>` representa o elemento raiz de uma página HTML.

A Tag `<head>` contém informações de metadados sobre a página.

A Tag `<title>` especifica um título para a página.

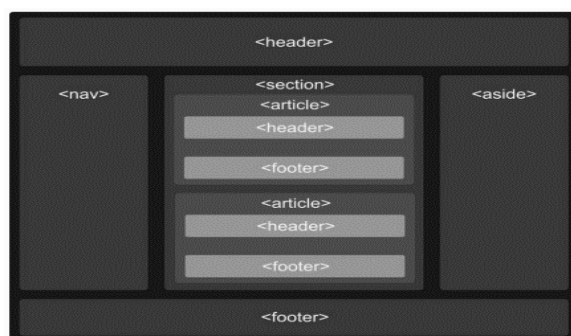
A Tag `<body>` contém todo conteúdo da página que estará visível.

O Meta é usado para especificar qual conjunto de caracteres é usado, descrição da página, palavras-chave, autor e outros metadados.

OS NOVOS ELEMENTOS DE HTML5

A seguir vamos entender os principais novos elementos para agrupamento de texto/conteúdo em HTML5, antes de começar é importante destacar que a tag `<div>` permanece ativa, embora a maioria dos tutoriais e textos sobre HTML5 encontrados na web não usam ou se referem à a tag `<div>` para formatação e disposição dos elementos, alguns materiais citam até uma certa proibição ou grande restrição no uso desta tag, a verdade é que não existe essa proibição, podemos usar a tag `<div>` da mesma forma que anteriormente que nossa página vai continuar sendo HTML5, o que se pode entender é uma nova forma de trabalho, uma segunda opção de estruturação da página. Nesta parte vamos entender a funcionalidade dos novos elementos para agrupamento de texto/conteúdo de HTML5, vamos entender como utilizar estes novos elementos, iremos criar duas estruturas usando somente os novos elementos, mas por razões de aprendizagem e não restritivas.

Veja na ilustração a seguir um modelo simples que mostra a semântica dos novos elementos de HTML5, é uma proposta interessante para mostrar a utilização dos novos elementos, se fizer uma pesquisa na web verá outras alternativas e aqui no próprio curso vamos usar de outras maneiras, não se preocupe com a maneira X ou Y ser a correta ou a errada, o importante é que você entenda estes novos elementos.



HEADER

O elemento header representa um cabeçalho, uma introdução ou um título de um artigo por exemplo. O elemento header pode ser usado para cabeçalho da página ou de um article por exemplo, devemos tomar cuidado para não confundir com a tag <head>, são tags distintas, tem funções bem diferentes. Com a tag <header> aplicada à página de uma forma geral, podemos inserir logotipo, nome do site, slogan, barra de pesquisa, etc, geralmente é a parte superior do site.

FOOTER

O elemento footer representa o rodapé da página. Seria o último elemento antes de fechar a tag HTML. O elemento footer pode ser usado para a página ou também no final de uma seção.

NAV

O elemento nav representa uma seção da página que contém links para outras partes do website. Nem todos os grupos de links devem ser elementos nav, apenas aqueles grupos que contém links importantes. O elemento nav é bem aplicado para criar o menu principal do site, links na barra lateral, grupo de links no rodapé, etc.

ARTICLE

Como o próprio nome diz, o elemento article representa um artigo, um post, uma notícia, de uma forma geral um bloco de texto comum.

ASIDE

O elemento aside representa uma barra lateral por exemplo bloco de conteúdo que referencia o conteúdo que envolve o elemento aside. O aside pode ser representado por conteúdos em sidebars em textos impressos, publicidade ou até mesmo para criar um grupo de elementos e outras informações separados do conteúdo principal do website.

SECTION

A tag `section` define uma nova seção “genérica” no documento. É uma tag de estruturação e organização da página, a `section` passa a ser o primeiro elemento de organização da página, dentro do `<body>` teremos várias tags `<section>` organizando cada parte da página. Por exemplo, a home de um website pode ser dividida em diversas seções: introdução ou destaque, novidades, informação de contato, artigos, notícias, etc. A tag `<section>` é bem genérica e tem a função de organizar os elementos dentro de uma página, basicamente uma `<section>` serve para agrupar elementos que fazem parte do mesmo bloco ou tipo.

FIGURE

Normalmente uma `<figure>` é uma imagem, ilustração, diagrama, trecho de código, etc., que é referenciado no fluxo principal de um documento, mas que pode ser movido para outra parte do documento ou para um apêndice sem afetar o fluxo principal

FIGCAPTION

Uma legenda pode ser associada ao elemento `<figure>` inserindo um `<figcaption>` dentro dele (como o primeiro ou o último filho). O primeiro elemento `<figcaption>` encontrado na figura é apresentado como legenda da figura.

SUA PRIMEIRA PÁGINA NA WEB

Pronto agora que já aprendemos sobre as principais novas tags de HTML5 já podemos iniciar o desenvolvimento de uma página, estas são só as tags principais para a nova semântica de HTML5, existem muitas outras tags que iremos aprender, para

vídeo, áudio e etc, a medida que formos evoluindo em nosso curso vamos aprendendo as novas tags de HTML5.

Para o desenvolvimento de uma página podemos usar vários aplicativos, no entanto iremos trabalhar com “Bloco de Notas” que está disponível no Windows o outro será o Visual Studio Code, um programa um pouco mais completo que você pode baixar pelo link oficial (<https://code.visualstudio.com/>).

Neste primeiro momento usaremos o notepad (Bloco de Notas) para melhor compreensão da estrutura e memorização das tags.

Abra o Bloco de Notas, clique no menu “Arquivo – Novo”, na página em branco digite o código básico do HTML5, mostrado a seguir:

```
<!DOCTYPE HTML>
<html lang="pt-br">
<head>
    <meta charset="utf-8">
    <title> </title>
</head>
<body>
</body>
</html>
```

Salve este arquivo com nome “pagina1.html” em alguma pasta de sua preferência, em seguida.

Adicionando um Título

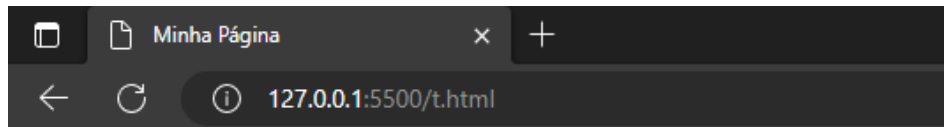
Agora vamos criar um título para a nossa página, ele deve ser curto, descritivo e único para cada página, na maioria dos navegadores ele

aparece na aba das páginas. O título também se mostra nas listas de histórico e favoritos e o mais importante, ele é utilizado por sites de busca.

O código ficaria assim:

```
<!DOCTYPE HTML>
<html lang="pt-br">
<head>
  <meta charset="utf-8">
  <title> Minha Página </title>
</head>
<body>
</body>
</html>
```

O resultado será esse:



Adicionando cabeçalhos

Para criar os cabeçalhos ou títulos de texto é usado o elemento **h**. Lembre-se que ele fornece seis níveis que estabelecem uma hierarquia nas informações de sua página.

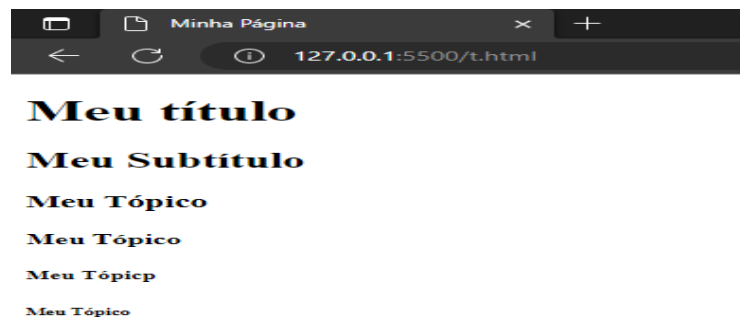
O elemento HTML **<header> representa um grupo de suporte introdutório ou navegacional. Pode conter alguns elementos de cabeçalho mas também outros elementos como um logo, seções de cabeçalho, formulário de pesquisa, e outros.*

Observe o exmplo descrito a baixo:

Escreva o seguinte código:

```
<body>
  <h1>Meu título</h1>
  <h2>Meu Subtítulo</h2>
  <h3>Meu Tópico</h3>
  <h4>Meu Tópico</h4>
  <h5>Meu Tópicp</h5>
  <h6>Meu Tópico</h6>
</body>
</html>
```

O resultado será esse:



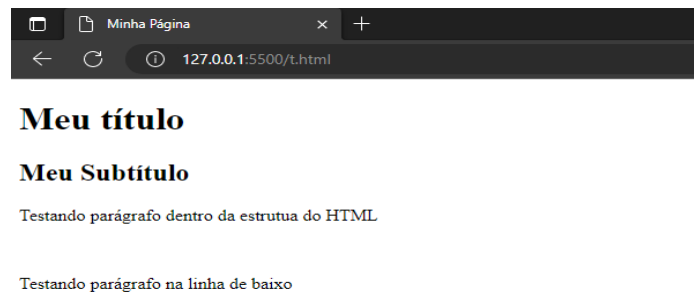
Adicionando Parágrafos

Vamos apagar agora alguns dos cabeçalhos usados a cima e vamos inserir um parágrafo. Lembre-se que o HTML não reconhece os espaços que você digita nos parágrafos sejam eles para quebrar uma linha ou acrescentar um espaço para o começo do parágrafo.

```
</head>
<body>
  <h1>Meu título</h1>
  <h2>Meu Subtítulo</h2>
  <p>Testando parágrafo dentro da estrutura do HTML</p> <br>
  <p>Testando parágrafo na linha de baixo</p>
</body>
</html>
```

Repare que dentro de um parágrafo foi utilizada um elemento `
`. Essa tag insere uma única quebra de linha. Lembrando que ela é uma tag vazia o que significa que não tem nenhuma marca de fim.

Resultado:



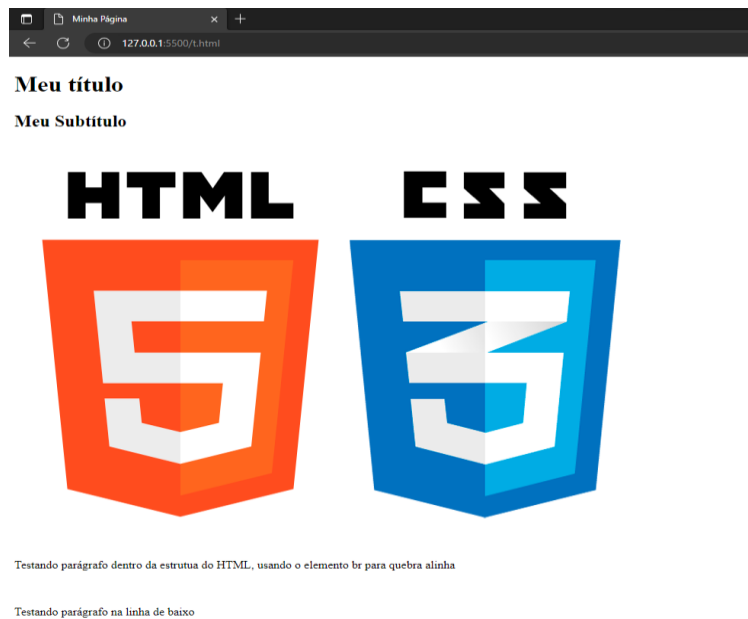
Adicionando imagens

Você pode colocar qualquer tipo de imagens em sua página web, desde logos a fotografias. As imagens aparecem de forma automática quando o usuário acessar sua página. Para inserir uma imagem em uma página faça esses passos:

1. Coloque o cursor no código HTML em que queira que a imagem apareça após o subtítulo `<h2>`.
2. Digite ``, você deve indicar a localização do arquivo de imagem no servidor. Não se esqueça de fechar a tag!

```
</head>
<body>
  <h1>Meu título</h1>
  <h2>Meu Subtítulo</h2>
  
  <p>Testando parágrafo dentro da estrutura do HTML</p> <br>
  <p>Testando parágrafo na linha de baixo</p>
</body>
</html>
```

Pronto, nosso primeiro site está pronto, o resultado dos nossos códigos deverão estar da seguinte forma como mostrado na imagem a baixo:



EXERCÍCIOS

- 1) Qual a estrutura básica de uma página HTML5 ?
- 2) Para que serve a tag TITLE?
- 3) Hierarquicamente, usamos:
 - a) `<html> <head> <title> </title> </head><body> </body> </html>`
 - b) `<html> <head> <title> <body> </body> </title> </head> </html>`

c) <html> <title> <head> </head><</title> body> </body> </html>

d) <html> <title> <head> <body> </body> </head> </title> </html>

4) A tag **<title>** define:

a) Um título para a janela que será exibido na Barra de Título do navegador.

b) Aparece sempre após a tag <body>

c) Aparece sempre na parte superior à tag <body>

d) As alternativas A e C estão corretas.

5) A tag **
** insere:

a) Uma linha em branco.

b) Um caractere em branco.

c) Uma quebra de linha.

d) Uma linha horizontal.

6) Qual a sintaxe do HTML5:

a) Sinal de maior + sinal de menor + tag e barra de fechamento.

b) Sinal de menor + tag + sinal de maior.

c) Tag + barra de fechamento + sinal de maior.

d) Tag + sinal de menor + sinal de maior

7) Quais são as tags usadas para formatação de títulos e subtítulos no HTML5:

- a) <h1></h1>, <h2></h2>
- b) <t1></t1>, <t2></t2>
- c) <h1></h1>, <t1></t1>
- d) <p1></p1>, <p2></p2>

8) Descreva as tags de acordo com suas declarações de estrutura do documento HTML5:

- a) A Tag <____> representa o elemento raiz de uma página HTML.
- b) A Tag <____> contém informações de metadados sobre a página.
- c) A Tag <____> especifica um título para a página.
- d) A Tag <____> contém todo conteúdo da página que estará visível.

9) Sobre as novas tags importantes destacamos as seguintes: section, nav, article, aside, header, footer e etc.

- a) <____> É uma tag de estruturação e organização da página, a section passa a ser o primeiro elemento de organização da página
- b) <____> Representa uma seção da página que contém links para outras partes do website.
- c) <____> O elemento que representa um artigo, um post, uma notícia, de uma forma geral um bloco de texto comum.
- d) <____> Representa um cabeçalho, uma introdução ou um título de um artigo por exemplo

10) Qual o comando usado para inserirmos uma imagem a nossa página:

- a)
- b) <img="local da imagem">
- c)
- d) <src img="local da imagem ">

TEXTOS

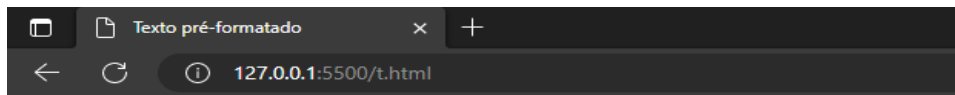
Texto pré-formatado

Os navegadores desconsideram os espaços excedentes entre as palavras contidas em um documento HTML assim como desconsideram as quebras de linha. Contudo, é possível inserir texto formatado com quantos espaços e quebras de linha desejarmos através do elemento <pre>. O texto contido no conteúdo de um elemento <pre> é exibido com todos os espaços e quebras de linha inseridos no documento HTML. Além disso, os navegadores costumam utilizar fonte mono espaçada para mostrar esse texto.

```
<title> Texto pré-formatado </title>
</head>
<body>
  <h1>Exemplo de texto pré-formatado</h1>
  <pre>Os espaços excedentes são considerados pelos
navegadores.
    Assim como as quebras de linha
    dentro da tag pre.
  </pre>
</body>
</html>
```

Por padrão, parágrafos e títulos ocupam todo o espaço horizontal do elemento pai.

O resultado será esse:



Exemplo de texto pré-formatado

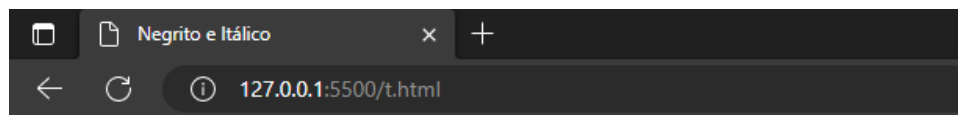
```
Os espaços excedentes são considerados pelos navegadores.  
Assim como as quebras de linha  
dentro da tag pre.
```

TEXTO IMPORTANTE OU ENFATIZADO

Podemos definir textos importantes ou enfatizados com os elementos `` e `` respectivamente. por padrão, nos navegadores, o conteúdo de um elemento `strong` é exibido em negrito e o conteúdo `em` é exibido em itálico.

```
<title> Negrito e Itálico </title>  
</head>  
<body>  
  <h2>Texto Negrito e Itálico</h2>  
  <p>O navegador interpreta a palavra em negrito  
<strong>importante</strong></p><br>  
  <em>Agora essa frase é enfatizada totalmente em  
itálico</em>  
</body>  
</html>
```

O navegador irá mostrar o código assim:



Texto Negrito e Itálico

O navegador interpreta a palavra em negrito **importante**

Agora essa frase é enfatizada totalmente em itálico

Citações

Podemos definir citações longas ou curtas com os elementos `<blockquote>` e `<q>` respectivamente. Por padrão, o elemento `blockquote` é exibido como elemento de bloco e o elemento `q` é exibido como elemento de linha. Esses dois elementos possuem o atributo `cite` que deve ser utilizado para indicar a fonte da citação.

Também podemos citar, com o elemento `cite`, títulos de trabalhos, livros, músicas, filmes, programas de TV, peças de teatro, entre outros. Por padrão, esse elemento é exibido como elemento de linha.

Abreviações

Podemos definir abreviações com o elemento `<abbr>`. Por padrão, esse elemento é exibido como elemento de linha. O atributo `title` desse elemento é utilizado para denir um tootip.

Alterações

Em alguns casos, é importante informar que o texto de um documento HTML sofreu alterações. Um texto que foi acrescentado

em um documento HTML deve ser definido com o elemento <ins>. Um texto que não faz mais parte do documento deve ser definido com o elemento . Um texto que deixou de ser correto, preciso ou relevante deve ser definido com o elemento <s>.

Centralizando textos

Para fazer com que um texto seja exibido de modo centralizado em HTML só é preciso usar a tag **center** e sua correspondente tag de fechamento, as mesmas delimitarão o texto que se pretende centralizar.

Vejamos um exemplo:

```
<html>
<head>
  <title>Exemplo 2</title>
</head>
<body>
  <center> Este texto deve aparecer centralizado
</center>
</body>
</html>
```

Sublinhando o texto

Para sublinhar uma parte do texto usando HTML, devemos somente inserir o texto correspondente dentro de um par de tags tipo u.

A tag **u** tem esse nome devido ao atributo underline (sublinhado).

Vejamos um exemplo:

```
<html>
<head>
  <title>Exemplo 7</title>
</head>
<body>
  Este texto contém uma palavra <u>sublinhada</u>.
</body>
</html>
```

Outras tags para modificar o texto

Existem muitos mais atributos que podem ser usados para mudar a apresentação do texto. Em geral, todos são utilizados da mesma forma, isto é, colocando a tag de abertura no princípio do texto a ser alterado e colocando a tag de fechamento ao final do mesmo.

É importante que o estudante não fique preocupado com a grande quantidade de tags HTML que existem para este tipo de tarefa. É possível ser um programador muito bom em HTML sem que nunca seja preciso utilizar nenhuma das tags apresentadas nesta seção. Dessa forma, nossa dica é a que você dê uma olhada na lista sem a obrigação de memorizá-las uma a uma.

Algumas dessas tags são as seguintes:

<small> Define um texto de menor tamanho.

<sub> Define um texto tipo subíndice (aparece mais abaixo e em menor tamanho).

<sup> Define um texto sobrescrito (aparece mais acima e em tamanho menor).

<ins> Define um texto inserido (geralmente equivale ao sublinhado).

**** Define um texto eliminado (geralmente se mostra tachado).

<code> Define uma porção de texto tipo programa (usualmente se mostra usando uma fonte não proporcional como o Courier).

<kbd> Define uma entrada de texto a partir do teclado (igual ao anterior).

- <samp>** Define um exemplo (igual ao anterior).
- <var>** Define uma variável (usualmente em itálico).
- <pre>** Define um texto pré-formatado (usualmente respeita os espaços em branco).
- <abbr>** Define uma abreviatura.
- <address>** Define um endereço pessoal.
- <bdo>** Define a direção do texto (direita a esquerda o vice-versa).
- <blockquote>** Define uma seção que corresponde a uma cita de outro texto (altera os margens).
- <q>** Define uma citação de uma ou poucas palavras.
- <cite>** Define uma citação de, por exemplo, um título de uma obra.
- <dfn>** Destaca uma definição (usualmente se faz em itálico).

O uso das tags que acabamos de mencionar passa a ter sentido ao complementar-se a página usando a linguagem CSS. Mediante seu uso, podemos definir exatamente como queremos que nossa página seja vista, por exemplo, todas as citações (tags tipo quote).

Tags para quebra de linha e para inserir uma linha horizontal no texto.

**
** É importante lembrar que a tag **br**, por ser um comando direto, não é necessário ter tag de fechamento.

<hr/> Permite desenhar uma linha horizontal na página, assim como a tag br, ela não precisa de tag de fechamento.

Essas são tags usadas com frequência na criação e configuração de texto.

EXERCÍCIOS

1. Reproduza a página a seguir:



2. Qual a tag usamos para sublinhar uma palavra?
3. Qual a tag usamos para quebrar uma linha?
4. Qual a tag define uma fonte menor?
5. Qual a tag mostra o texto tachado?
6. Qual a tag inseri uma linha?

7. Qual a tag usada para deixar a palavra em negrito?
8. Qual a tag para deixar a palavra em itálico?
9. Qual a tag define um texto pré-formatado?
10. Qual a tag define um abreviatura?

IMAGEM

Certamente, os sites seriam muito entediantes se não fosse possível adicionar imagens ao conteúdo das páginas. Podemos adicionar imagens em documento HTML com o elemento ``. Esse elemento possui um atributo chamado `src` que indica o caminho absoluto ou relativo da imagem que queremos adicionar. O elemento `` possui um atributo obrigatório chamado `alt`. Esse atributo define um texto alternativo que pode ser utilizado se houver um problema ao carregar a imagem ou por softwares de leitura de tela.

Segue o código do nosso exercício:

```
<title>Exemplo de imagens</title>
</head>
<body>
  <h1>Buscadores</h1>
  
  <h2>Cursos</h2>
  
</body>
</html>
```

Resultado:



Os links e as imagens podem ser adicionados em um documento HTML com URLs absolutas ou relativas.

```
<body>
  <h1>Buscadores</h1>
  
```

A URL absoluta contém todas as informações necessárias à localização de um recurso. A URL relativa localiza um recurso usando uma URL absoluta como ponto inicial. Na verdade, a "URL completa" do destino é especificada concatenando as URLs absoluta e relativa.

```
<body>
  <h1>Buscadores</h1>
  
```

LINKS

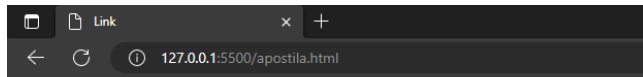
Um link tem duas partes principais: um destino e um rótulo. A primeira parte é sem dúvida a mais importante. Você utiliza para especificar o que vai acontecer quando o visitante acionar o link. Você pode criar links que vão para outras páginas, que pulem para dentro de uma página, mostrem uma imagem, baixem um arquivo, mande um e-mail e etc. A segunda parte do link é o rótulo, aquela parte que o visitante vê no navegador ou ouve em um leitor de telas para ser ativado e alcançar o destino. Ele pode ser um texto, imagem ou ambos. Os navegadores geralmente exibem o rótulo com um sublinhado em azul. Se você possui um site com mais de uma página na web é bem provável que queira criar links de uma página para outra, ou criar links para sites externos, sejam eles seus ou uma criação de outra pessoa. Para criar um link outra página siga esses passos:

1. Digite `` no lugar de `pagina.html` digite o caminho(URL) da página destino.
2. Digite o texto do rótulo, ou seja o texto que irá aparecer em destaque para que o link seja clicado, ou adicione um elemento `` como rótulo.
3. Digite `` para o fechamento do elemento.

Observe o código a seguir

```
<title>Link</title>
</head>
<body>
  <h1>Página Inicial do Google</h1>
  <p>Google é um dos buscadores mais usados do mundo, acesse:
  <a href="https://www.google.com.br/">clique aqui</a></p>
</body>
</html>
```

Repare que o link foi inserido dentro de um elemento de parágrafo. O resultado desse código ficou assim:



Página Inicial do Google

Google é um dos buscadores mais usados do mundo, acesse: [clique aqui](#)

ATRIBUTOS DO ELEMENTO LINK

Além do atributo href, podemos utilizar o atributo target para informar onde o destino de um link deve ser aberto. Esses atributos podem estar contido dentro do elemento link conforme sua necessidade:

- ◆ `_blank`: Abre o destino do link em uma nova janela ou aba.
- ◆ `_self`: Abre o destino do link na mesma janela ou no mesmo frame que exibe o documento que contém o link.
- ◆ `_parent`: Abre o destino do link na janela ou no frame onde está contido o frame 31 que exibe o documento que contém o link.
- ◆ `_top`: Abre o destino do link na janela que é “ancestral” do frame que exibe o documento que contém o link.

CRIANDO ÂNCORAS

Um link com target `_self` possui o mesmo comportamento de um link com target `_top` se esses links estiverem em um documento HTML que não esteja dentro de outro documento HTML.

Por padrão, o destino de um link é aberto na mesma página. Em outras palavras, se o atributo target não for definido explicitamente, o padrão é o comportamento do _self. Geralmente, a ativação de um link leva o usuário ao topo da página correspondente. Se quiser que ele pule para uma seção específica da página crie uma âncora e faça referência dela no link da seguinte forma: parágrafo 1 atribua um id de nome p1 para o parágrafo 1, e quando o link for clicado, o navegador irá mostrar na tela o parágrafo independentemente de sua posição.

```
<title>Links</title>
</head>
<body>
  <h1>Links referenciais</h1>
  <p><a href="#google">Google</a></p>
  <p><a href="#yahoo">Google</a></p>
  <p><a href="#bing">Google</a></p>
  <p id="google">Um motor de busca é feito para auxiliar a procura de informações armazenadas na rede mundial (WWW), dentro de uma rede corporativa ou de um computador pessoal. Ao se realizar uma consulta, a lista de ocorrências de assuntos é criada previamente por meio de um conjunto de softwares de computadores, conhecidos como Web crawler, que vasculham toda a Web em busca de ocorrências de um determinado assunto em uma página.</p>
  <p id="yahoo">Um motor de busca é feito para auxiliar a procura de informações armazenadas na rede mundial (WWW), dentro de uma rede corporativa ou de um computador pessoal. Ao se realizar uma consulta, a lista de ocorrências de assuntos é criada previamente por meio de um conjunto de softwares de computadores, conhecidos como Web crawler, que vasculham toda a Web em busca de ocorrências de um determinado assunto em uma página.</p>
  <p id="bing">Um motor de busca é feito para auxiliar a procura de informações armazenadas na rede mundial (WWW), dentro de uma rede corporativa ou de um computador pessoal. Ao se realizar uma consulta, a lista de ocorrências de assuntos é criada previamente por meio de um conjunto de softwares de computadores, conhecidos como Web crawler, que vasculham toda a Web em busca de ocorrências de um determinado assunto em uma página.</p>
</body>
</html>
```

Repare que as âncoras são criadas nos primeiros parágrafos, e logo após são criados parágrafos com os ids identificando cada um deles. O último link irá abrir em uma outra aba.

EXERCÍCIOS

1. Crie um documento HTML chamado ancoras.html e reproduza o seguinte código:

```
<!DOCTYPE html>
<html lang="pt-br">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<title>Exemplo de âncoras</title>
</head>
<body>
<h1>Exemplo de âncoras</h1>
<ul>
<li><a href="#sobre">Sobre o texto dessa página</a></li> </ul>
<p>Lorem ipsum dolor sit amet, consectetur adipiscing elit. Donec
justo 16 massa, sodales sit amet eleifend a, elementum eu nibh.
Donec egestas dolor 17 quis turpis dictum tincidunt. Donec blandit
tempus velit, sit amet 18 adipiscing velit consequat placerat.
Curabitur id mauris.</p>
<p>At nisi imperdiet lacinia. Ut quis arcu at nisl ornare viverra. 27
Duis vel tristique tellus. Maecenas ultrices placerat tortor.
Pellentesque 28 feugiat accumsan commodo. Proin non urna justo,
id pulvinar lacus.</p>
<h2 id="sobre">Sobre o texto dessa página</h2>
```

```
<p> O texto dessa página foi gerado através do site:  
<a href="http://www.lipsum.com/">http://www.lipsum.com/</a> </p>  
</body>  
</html>
```

LISTAS

A linguagem HTML tem três tipos distintos de listas.

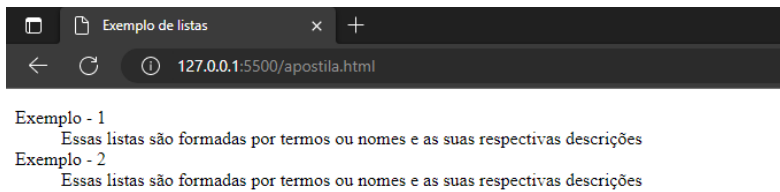
- ◆ Lista de descrições
- ◆ Listas ordenadas
- ◆ Listas não ordenadas

LISTA DE DESCRIÇÕES

Para criar uma lista de descrições, devemos utilizar o elemento `<dl>`. Essas listas são formadas por termos ou nomes e as suas respectivas descrições. Os termos ou nomes são definidos com o elemento `<dt>`. As descrições são definidas como elemento `<dd>`.

```
<title>Exemplo de listas</title> </head>  
<body>  
  <dl>  
    <dt>Exemplo - 1</dt>  
    <dd>Essas listas são formadas por termos ou nomes e as suas respectivas  
descrições</dd>  
    <dt>Exemplo - 2</dt>  
    <dd>Essas listas são formadas por termos ou nomes e as suas respectivas  
descrições</dd>  
  </dl>  
</body>
```

O resultado será esse:

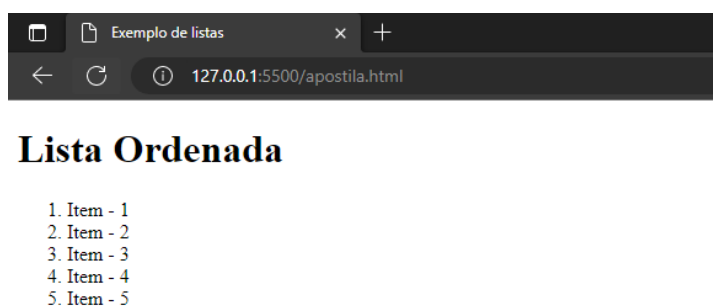


LISTA ORDENADA

Para criar uma lista ordenada, devemos utilizar o elemento ``. Os itens de uma lista com ordem são definidos com o elemento ``.

```
<h1>Lista Ordenada</h1>
<ol>
  <li>Item - 1</li>
  <li>Item - 2</li>
  <li>Item - 3</li>
  <li>Item - 4</li>
  <li>Item - 5</li>
</ol>
</body>
</html>
```

O resultado será esse:

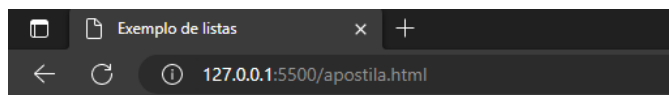


LISTA NÃO ORDENADA

Para criar uma lista sem ordem, devemos utilizar o elemento ``. Os itens de uma lista sem ordem são definidos com o elemento ``.

```
<body>
  <h1>Lista Não Ordenada</h1>
  <ul>
    <li>Item - 5</li>
    <li>Item - 3</li>
    <li>Item - 4</li>
    <li>Item - 1</li>
    <li>Item - 2</li>
  </ul>
</body>
</html>
```

O resultado será esse:



Lista Não Ordenada

- Item - 5
- Item - 3
- Item - 4
- Item - 1
- Item - 2

Observer a figura a cima que o resultado foi o mesmo do código descrito, porém ao não foi listado de forma crescente e sim aplicado o marcador no itens descritos.

Listas aninhadas

Uma lista pode ser definida dentro de outra lista. Quando listas sem ordem são aninhadas, normalmente, os navegadores alternam os marcadores dos itens.

```
<title>Exemplo de listas</title> </head>
<body>
  <h1>Lista Aninhada</h1>
  <ul>
    <li>Brasil
      <ol>
        <li>Brasília</li>
        <li>Rio de Janeiro</li>
        <li>São Paulo</li>
      </ol>
    </li>
  </ul>
</body>
</html>
```

EXERCÍCIOS

1- Reproduza a seguinte lista:

1. Brasil
2. Argentina
3. Uruguai

- 10.Paraguai
- 11.Chile
- 12.Venezuela
- 20.Colômbia
- 21.Bolívia
- 22.Peru

Resposta:

```
<title>Exemplo de listas</title> </head>
<body>
  <ol type="1">
    <li>Brasil</li>
    <li>Argentina</li>
    <li>Uruguai</li>
  </ol>
  <ol type="1" start="10">
    <li>Paraguai</li>
    <li>Chile</li>
    <li>Venezuela</li>
  </ol>
  <ol type="1" start="20">
    <li>Colômbia</li>
    <li>Bolívia</li>
    <li>Peru</li>
  </ol>
</body>
</html>
```

2- Reproduza o seguinte código:

```
<!DOCTYPE html>
<html lang="pt-br">
<head>
<meta http-equiv="Content -Type" content="text/html;
charset=UTF-8">
<title>Exemplo de listas aninhadas</title>
</head>
```

```
<body>
<h1>Exemplo de listas aninhadas</h1>
<h2>Continentes</h2>
<ul>
<li>
Europa
<ul>
<li>Portugal</li>
<li>França</li>
<li>Alemanha</li>
</ul>
</li>
<li> Ásia
<ul>
<li>Japão</li>
<li>China</li>
<li>Índia</li>
</ul>
</li>
</ul>
<h2>Cronograma da minha viagem</h2>
<ol>
<li> Europa <ol>
<li>Portugal</li>
<li>França</li>
<li>Alemanha</li>
</ol>
</li>
<li> Ásia <ol>
```


ESTRUTURA DE UMA TABELA EM HTML

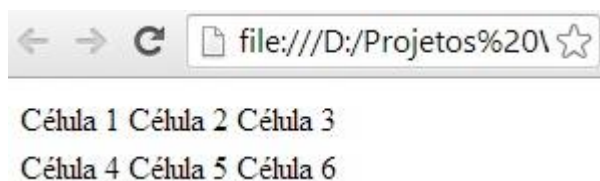
Uma tabela em linguagem HTML é formada por, no mínimo, três tags básicas que são: <table>, <tr> e <td>. Sendo que:

- A tag **<table>** significa “*tabela*” indica onde começa e termina uma tabela;
- A tag **<tr>** significa “*table row*” (linha de tabela) e indica onde começa e termina e uma linha horizontal da tabela; e
- A tag **<td>** significa “*table data*” (dados da tabela) e indica onde começa e termina cada célula contida nas linhas da tabela.

Vejamos no exemplo abaixo a codificação de uma tabela bem simples e, logo em seguida, vamos analisar cada linha do código para entender a diferença entre as tags.

```
<table>
  <tr>
    <td>Célula 1</td>
    <td>Célula 2</td>
    <td>Célula 3</td>
  </tr>
  <tr>
    <td>Célula 4</td>
    <td>Célula 5</td>
    <td>Célula 6</td>
  </tr>
</table>
```

Veja na imagem abaixo como o navegador exibe o exemplo acima.



Explicando cada tag da tabela

- No exemplo acima começamos uma tabela com a tag <table> e em seguida utilizamos a tag <tr> indicando o início de uma linha;
- Dentro dessa primeira linha da tabela inserimos três células (colunas) que são representados pelo conjunto de tags <td> e </td>, que são responsáveis por exibir o conteúdo de cada célula no navegador;
- A primeira linha termina com a tag </tr> e uma nova linha começa imediatamente com a tag <tr>, seguindo a mesma estrutura da linha anterior e, ao final dela, temos a tag <table> indicando o fim da tabela.

Observação: Uma tabela pode conter um número ilimitado de linhas e colunas.

Para uma maior legibilidade da nossa tabela podemos utilizar o atributo **border** para definir bordas entre as células da tabela. Alterando apenas a tag <table> para <table border="1">, temos o seguinte resultado.



Célula 1	Célula 2	Célula 3
Célula 4	Célula 5	Célula 6

Atributos colspan e rowspan

Outros dois atributos muito utilizados em tabelas são o colspan e rowspan.

O colspan (abreviação para “column span”) é utilizado na tag <td> para indicar quantas colunas estarão contidas em uma célula.

Veja no código abaixo um exemplo da utilização desse atributo.

```

<table border="1">
  <tr>
    <td colspan="3">Célula 1</td>
  </tr>
  <tr>
    <td>Célula 2</td>
    <td>Célula 3</td>
    <td>Célula 4</td>
  </tr>
</table>

```

O resultado será esse:



Célula 1		
Célula 2	Célula 3	Célula 4

Como você já deve ter imaginado, o rowspan especifica quantas linhas estarão contidas em uma célula. Veja no código abaixo um exemplo da utilização desse atributo.

```

<table border="1">
  <tr>
    <td rowspan="3">Célula 1</td>
    <td>Célula 2</td>
  </tr>
  <tr>
    <td>Célula 3</td>
  </tr>
  <tr>
    <td>Célula 4</td>
  </tr>
</table>

```

O resultado será esse:



Célula 1	Célula 2
	Célula 3
	Célula 4

Mais tags para as tabelas

Além das tags apresentadas até aqui, existem mais quatro que utilizamos para criar tabelas eficientes e que atendam as

recomendações da W3C. Visualmente elas mudarão pouca coisa na nossa tabela, porém, é importante que se conheça para que no próximo de curso de CSS seja possível aplicar estilos a essas marcações HTML. Segue abaixo essas quatro novas tags e seu significado.

th – define uma célula de cabeçalho da tabela

thead – define o cabeçalho da tabela

tbody – define o corpo da tabela

tfoot – define o rodapé da tabela

Para entender melhor a utilização dessas tags, observe atentamente o código abaixo e veja a aplicação de cada uma delas dentro do contexto criado na tabela apresentada no início dessa aula.

```
<table border="1">
  <thead>
    <tr>
      <th>Marca:</th>
      <th>Modelo:</th>
      <th>Ano:</th>
      <th>Valor:</th>
    </tr>
  </thead>
  <tbody>
    <tr>
      <td>Volkswagen</td>
      <td>Golf</td>
      <td>2009</td>
      <td>R$38.500</td>
    </tr>
    <tr>
      <td>Fiat</td>
      <td>Palio</td>
      <td>2010</td>
      <td>R$19.700</td>
    </tr>
    <tr>
      <td>Honda</td>
      <td>Civic</td>
      <td>2012</td>
      <td>R$61.500</td>
    </tr>
  </tbody>
  <tfoot>
    <tr>
      <td colspan="4">O melhor preço da região</td>
    </tr>
  </tfoot>
</table>
```

Marca:	Modelo:	Ano:	Valor:
Volkswagen	Golf	2009	R\$38.500
Fiat	Palio	2010	R\$19.700
Honda	Civic	2012	R\$61.500
O melhor preço da região			

Qual o motivo da existência dessas tags?

Essas quatro tags (**th**, **thead**, **tbody** e **tfoot**) possui muito mais valor semântico do que aplicação. No entanto, quando precisamos de criar estilos diferentes em partes diferentes da nossa tabela, a utilização dessas tags facilita muito a nossa vida.

Nos primórdios da internet as tabelas eram amplamente utilizadas para construir o layout dos sites, porém, **temos atualmente uma maneira muito mais racional de fazer isso com as regras de estilo CCS**. Dessa forma, a recomendação é que as tabelas sejam utilizadas unicamente para o seu real propósito, que é apresentar dados tabulares.

EXERCÍCIO

Crie uma página HTML em um arquivo chamado "horarios.html" adicione os códigos pertinentes ao exemplo citado a baixo, observe bem todas as informações contidas na imagem da tabela e reproduza no navegador.

Tabela	Hora da saída	Hora de chegada	Classe do ônibus	Tarifa normal	Frequência
Cidade A	06:00	14:00	Convencional	20,00	Diária
Cidade B	12:00	15:00	Convencional	10,00	Domingos
Cidade C	14:00	17:00	Leito	15,00	Diária
Cidade D	16:00	17:00	Convencional	10,00	Diária
Cidade E	18:30	20:00	Executivo	30,00	Domingos
Cidade F	20:00	23:00	Convencional	80,00	Sábados
Cidade G	21:00	23:30	Convencional	26,00	Diária
Cidade H	22:00	23:00	Executivo	16,00	Diária
Horários de ônibus					

PARÂMETROS GET E POST

Ao desenvolver um site dinâmico, existe a necessidade de passar alguns valores de uma página para a outra, para realizar operações como consultas e inserções no banco, autenticação de usuários, etc. Então para que essa funcionalidade funcione é necessário utilizar no formulário os parâmetros GET ou POST, pois esses parâmetros informam ao navegador a maneira que os dados deverão ser passados.

GET

Com capacidade de 1024 caracteres, este método é utilizado quando se quer passar poucas ou pequenas informações para realizar uma pesquisa ou simplesmente passar uma informação para outra página através da URL. Caso este limite seja ultrapassado, corre-se o risco de obter um erro na página, uma vez que as informações foram passadas de forma incompleta.

Por exemplo:

<http://www.boravencer.com.br/index.php?categoria=2&pag=1&tipo=8> Repare que com o parâmetro GET as informações ficam expostas. Então se sua página deseja passar dados confidenciais, o método GET não é indicado.

POST

Este método utiliza a URI para envio de informações ao servidor. A URL não é retornável ao cliente, o que torna o método POST mais seguro, pois não expõe os dados enviados no navegador. Como não tem limite de capacidade para envio de informações, este método se torna melhor que o GET. No POST, uma conexão paralela é aberta e os dados são passados por ela.

FORMULÁRIOS

Alguns sites e praticamente todas as aplicações web precisam obter informações enviadas pelos usuários. Por exemplo, considere uma empresa que deseja receber os pedidos dos seus clientes através do seu site. O site dessa empresa precisa oferecer alguma forma de interação que possibilite o recebimento de dados fornecidos pelos usuários. Então para tornar os sites e as aplicações web mais interativos, podemos utilizar formulários. Para criar um formulário, devemos utilizar o elemento `<form>`. Esse elemento possui um atributo chamado `action` que é responsável por indicar para qual endereço os dados do formulário serão enviados. Os formulários são compostos por caixas de texto, checkboxes, radios, caixas de seleção, botões, entre outros componentes.

```
<form action="/pagina-processa-dados-do-form" method="post">  
  
</form>
```

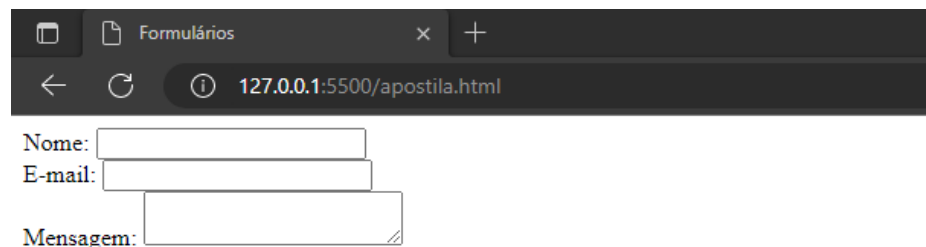
Provavelmente, você reconhecerá diversos desses componentes na imagem abaixo.

Todos os seus atributos são opcionais, mas é considerada a melhor prática sempre definir pelo menos o atributo `action` e o atributo `method`. Adicionar campos com os elementos `<label>`, `<input>`, e `<textarea>` O nosso formulário de contato é muito simples e contém três campos de texto, cada um com uma etiqueta. O campo de entrada para o nome será um campo básico texto de linha única("input"); o campo de entrada do e-mail será um campo de texto com uma única linha("input") que vai aceitar apenas um endereço de e-mail; o campo de entrada para a mensagem será um campo de texto de várias linhas("textarea").

Em termos de código HTML, teremos algo assim:

```
<form action="/pagina-processa-dados-do-form" method="post">
  <div>
    <label for="nome">Nome:</label>
    <input type="text" id="nome" />
  </div>
  <div>
    <label for="email">E-mail:</label>
    <input type="email" id="email" />
  </div>
  <div>
    <label for="msg">Mensagem:</label>
    <textarea id="msg"></textarea>
  </div>
</form>
```


O resultado será esse:



Nome:

E-mail:

Mensagem:

Os elementos como o **<div>** “elemento de container” estão lá para estruturar nosso código e deixar a estilização mais fácil. Observe o uso do atributo **for** em todos os elementos **<label>** ; é uma maneira para vincular uma label à um campo do formulário. Este atributo faz referência ao **id** do campo correspondente. Há algum benefício para fazer isso, é a de permitir que o usuário clique no rótulo para ativar o campo correspondente

No elemento **<input>** , o atributo mais importante é o atributo **type**. Esse atributo é extremamente importante porque define a forma como o elemento **<input>** se comporta. Ele pode mudar radicalmente o elemento, então preste atenção a ele. Em nosso exemplo, nós usamos somente o **type="text"**, valor padrão para este atributo. Ele representa um campo de texto com uma única linha que aceita qualquer tipo de texto sem controle ou validação. Nós também usamos o **type="email"** que define um campo de texto com uma única linha que só aceita um endereço de e-mail bem-formatados. Este último valor torna um campo de texto básico em uma espécie de campo "inteligente", que irá realizar alguns testes com os dados digitados pelo usuário. Por último, mas não menos importante, observe a sintaxe de **<input />** e **<textarea> </textarea>**. Esta é uma das esquisitices do HTML.

A tag **<input />** é um elemento que se auto-fecha, o que significa que se você quiser encerrar formalmente o elemento, você

tem que adicionar uma barra "/" no final do próprio elemento e não uma tag de fechamento. No entanto, o tipo **<textarea>** não é um elemento de auto-fechamento, então você tem que fechá-lo com a tag final adequada. Isso tem um impacto sobre um recurso específico de formulários HTML: a maneira como você define o valor padrão. Para definir o valor padrão de um elemento **<input>** você tem que usar o atributo value como este:

```
<input type="text" value="Por padrão, este elemento será  
preenchido com este texto " />
```

Pelo contrário, se você deseja definir o valor padrão de um elemento **<textarea>**, você só tem que colocar esse valor padrão no meio das tags, entre tag inicial e a tag final do elemento **<textarea>**, como abaixo:

```
<textarea>Por padrão, este elemento será preenchido com este  
texto </textarea>
```

E um elemento **<button>** para concluir o nosso formulário está quase pronto; nós temos apenas que adicionar um botão para permitir que o usuário envie seus dados depois de ter preenchido o formulário. Isto é simplesmente feito usando o elemento **<button>** :

```
<body>  
  <form action="/pagina-processa-dados-do-form" method="post">  
    <div>  
      <label for="name">Nome:</label>  
      <input type="text" id="name" />  
    </div>  
    <div>  
      <label for="mail">E-mail:</label>  
      <input type="email" id="mail" />  
    </div>  
    <div>  
      <label for="msg">Mensagem:</label>  
      <textarea id="msg"></textarea>  
    </div>  
    <div class="button">  
      <button type="submit">Enviar sua mensagem</button>  
    </div>  
  </form>  
</body>
```

O resultado será esse:



A screenshot of a web browser window. The address bar shows '127.0.0.1:5500/apostila.html'. The page title is 'Formulários'. The form contains three text input fields labeled 'Nome:', 'E-mail:', and 'Mensagem:'. Below the 'Mensagem:' field is a button labeled 'Enviar sua mensagem'.

Um botão pode ser de três tipos: **submit**, **reset**, ou **button**.

Um clique sobre um botão de submit envia os dados do formulário para a página de web definida pelo atributo action do elemento

<form> .

Um clique sobre um botão de **reset** redefine imediatamente todos os campos do formulário para o seu valor padrão. Um clique em um botão do tipo **button** faz ...ops, nada! Isso soa bobo, mas é incrivelmente útil para construir botões personalizados com JavaScript, ou seja, ele pode assumir qualquer comportamento através desta linguagem.

Note que você também pode usar o elemento **<input>** com o tipo correspondente para produzir um botão. A principal diferença com o elemento **<button>** é que o elemento **<input>** permite apenas texto sem formatação como seu valor, enquanto que o elemento **<button>** permite que o conteúdo HTML completo como seu valor.

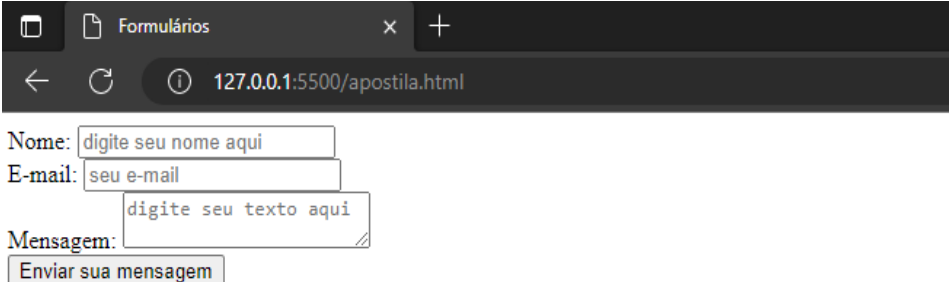
PLACEHOLDERS

Como vimos, os rótulos são utilizados para informar aos usuários quais dados devem ser preenchidos nos formulários.

Além dos rótulos, podemos utilizar placeholders para dar dicas ou exemplos do conteúdo que desejamos em cada caixa de entrada. Um placeholder é criado através do atributo placeholder do elemento **input**.

```
<label for="name">Nome:</label>  
  <input type="text" id="name" placeholder="digite seu  
nome aqui" />
```

Foi adicionado o placeholder nos demais campos, observe a imagem a baixo qual o resultado final.



A imagem mostra uma interface de usuário em um navegador web. No topo, há uma barra de endereço com o texto "127.0.0.1:5500/apostila.html". Abaixo, o formulário contém os seguintes elementos: um campo "Nome:" com o placeholder "digite seu nome aqui", um campo "E-mail:" com o placeholder "seu e-mail", e um campo "Mensagem:" com o placeholder "digite seu texto aqui". No final do formulário, há um botão "Enviar sua mensagem".

CAIXAS DE ENTRADA ESPECÍFICAS

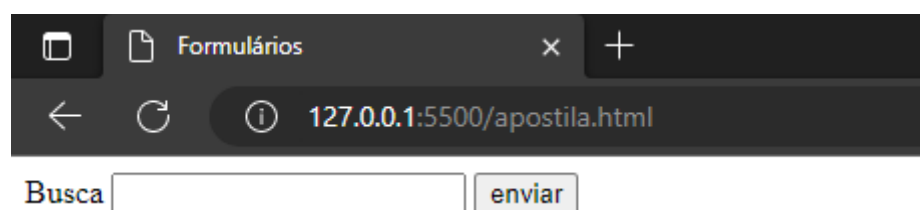
As caixas de texto são componentes muito genéricos. Antes do HTML5, informações de natureza bem distintas eram obtidas através desses componentes. Para melhorar a semântica dos documentos HTML, tipos específicos de caixas foram adicionados no HTML5.

CAIXAS DE BUSCA

Assim como as caixas de texto, as caixas de busca são adicionadas nos formulários como elemento input.

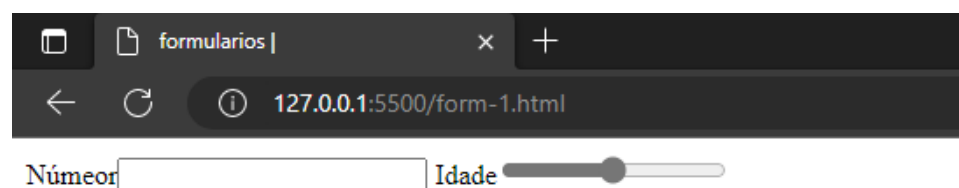
A diferença é que o valor do atributo type deve ser search ao invés de text. As caixas de busca devem ser utilizadas para coletar palavras chave que serão utilizadas em algum tipo de pesquisa. A princípio não há nenhuma diferença prática entre as caixas de texto e as caixas de busca. Contudo, essa diferenciação adiciona valor semântico aos documentos HTML e possibilita, por exemplo, que os navegadores diferenciem visualmente esses dois tipos de caixas.

```
<label for="keywords_id">Busca</label>
  <input id="keywords_id" name="keywords"
type="search">
  <input id="botao_id" type="submit" value="enviar">
```



CAIXAS DE NÚMEROS

Para coletar dados numéricos, podemos utilizar caixas específicas para números. No HTML5, há dois tipos de caixas para esse propósito. Os dois são definidos com o elemento **<input>**. O valor do atributo type é number para o primeiro tipo e range para o segundo tipo.



Esses dois tipos de componentes devem ser utilizados para coletar valores de sequências numéricas pré-denidas. A principal diferença entre eles é que o primeiro (type=number) deve oferecer um mecanismo preciso para os usuários selecionarem o valor desejado enquanto o segundo(type=range) não possui essa obrigação.

Para definir a sequência dos números que podem ser selecionados pelos usuários, podemos utilizar os atributos min, max e step. Por exemplo, para coletar um número da sequência {0; 0,2; 0,4; 0,6;0,8;1}, os valores dos atributos min, max e step devem ser 0,1 e 0.2 respectivamente.

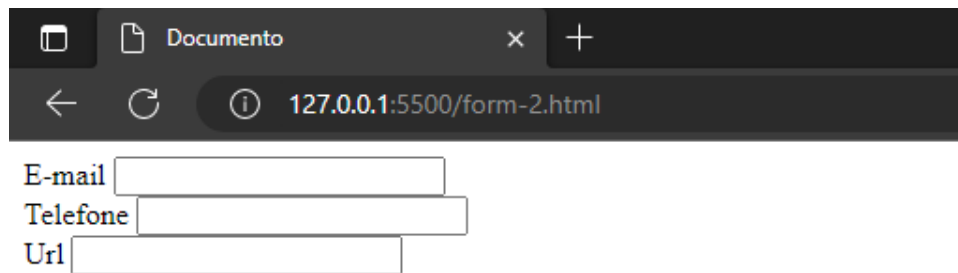
CAIXAS DE E-MAIL, TELEFONE E URL

No HTML5 foram denidas caixas de entradas específicas para e-mails, telefones e urls. Essas caixas são adicionadas com o elemento input. O valor do atributo type deve ser email, tel e url para e-mails, telefones e urls respectivamente.

A usabilidade das páginas web melhora com a utilização dessas caixas. Por exemplo, a configuração do teclado dos celulares ou tablets pode ser alterada de acordo com o tipo de caixa de entrada. Nas caixas de e-mail, o caractere “@” pode ser adicionado ao

```
<body>
  <form action="pagina.html"></form>
  <label for="email_id"> E-mail</label>
  <input id="email" name="email" type="email"> <br>
  <label for="telefone_id">Telefone</label>
  <input id="telefone_id" name="telefone" type="tel"><br>
  <label for="url">Url</label>
  <input id="url_id" name="url" type="url">
</body>
```

teclado. Nas caixas de telefone, o teclado não precisa conter as letras do alfabeto. Nas caixas de url, teclas especiais como “.com” ou “www” podem ser adicionadas ao teclado.



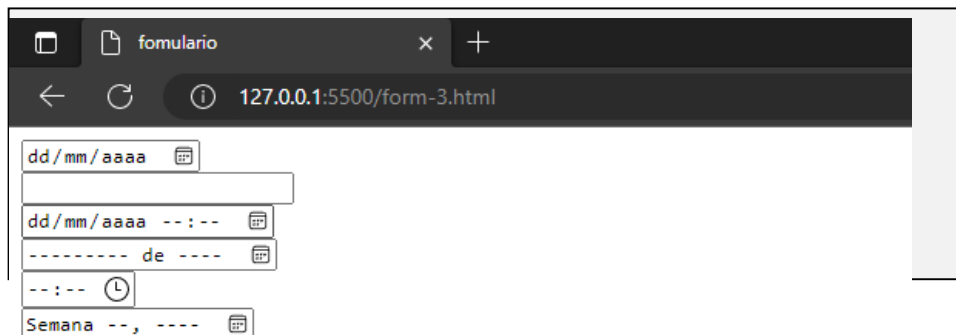
The image shows a web browser window with a dark theme. The address bar displays the URL "127.0.0.1:5500/form-2.html". Below the address bar, there are three input fields labeled "E-mail", "Telefone", and "Url". Each label is in blue text, and the input fields are white with a light blue border. The "E-mail" field is the first, followed by "Telefone", and then "Url".

CAIXAS DE DATAS E HORAS

Diversos tipos de caixas de entrada para coletar datas e horas foram adicionados no HTML5. Todas essas caixas são adicionadas com o elemento `<input>` e o valor do atributo `type` desse elemento assumirá um dos valores listados a seguir.

- ◆ `date`: Utilizado para coletar data (dia, mês e ano) sem fuso horário.
- ◆ `date time`: Utilizado para coletar data (dia, mês e ano) e hora (hora, minuto, segundo e fração de segundo) com fuso horário em UTC.
- ◆ `date time-local`: Utilizado para coletar data (dia, mês e ano) e hora (hora, minuto, segundo e fração de segundo) sem fuso horário.
- ◆ `month`: Utilizado para coletar data composta por mês e anos em fuso horário.
- ◆ `time`: Utilizado para coletar hora (hora, minuto, segundo e fração de segundo) sem fuso horário.
- ◆ `week`: Utilizado para coletar data composta por semana e anos em fuso horário.

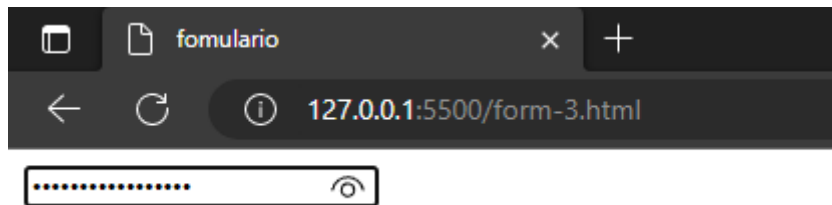
◆ A criação desses componentes permite que os navegadores melhorem a usabilidade das páginas web. A forma de exibição das caixas de datas e horas pode facilitar o processo de preenchimento dos formulários. Inclusive, os navegadores podem exibir esses componentes de formas diferentes de acordo com o dispositivo (computador, celular, tablet , entre outros).

A screenshot of a web browser window titled 'formulario'. The address bar shows '127.0.0.1:5500/form-3.html'. The form contains several input fields: a date picker with 'dd/mm/aaaa', a time picker with 'dd/mm/aaaa --:--', a dropdown menu with 'de', a clock icon, and a week picker with 'Semana --, ----'. Each picker has a small calendar icon to its right.

CAIXAS DE SENHA

As senhas devem ser coletadas com caixas específicas para esse tipo de informação. Para adicionar uma caixa de senha em um formulário, devemos utilizar o elemento input com o valor password para o atributo type. Normalmente, os navegadores utilizam símbolos como o asterisco ou o círculo para omitir o conteúdo das caixas de senha.

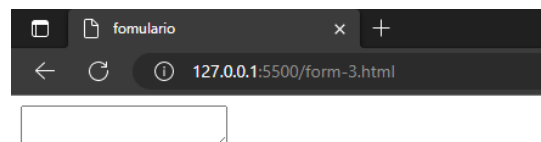
```
<body>
  <form action="pagina.html">
    <input id="senha_id" name="senha" type="password"> <br>
  </form>
</body>
```

CAIXAS DE TEXTO LONGO

Para coletar um texto com várias linhas, podemos utilizar o elemento `<textarea>`. A quantidade de linhas de um `textarea` é definida como atributo `rows` e a quantidade de colunas como atributo `cols`. Esse elemento também possui o atributo `name` que funciona como no elemento `input`. Podemos definir o limite de caracteres que podem ser inseridos no conteúdo do elemento `textarea` através do atributo `maxlength`.

```
<form action="pagina.html">
  <textarea id="mensagem_id" name="mensagem" maxlength="400"></textarea>
</form>
```

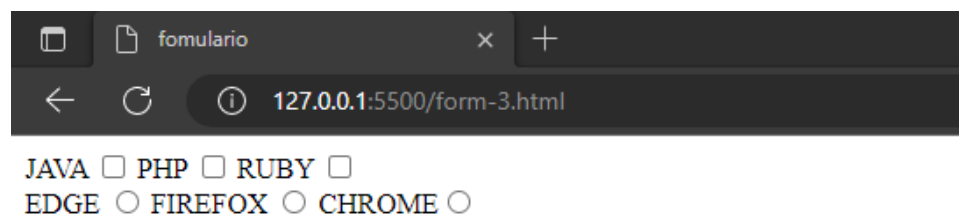


CHECK BOXES E RADIOS

Para adicionar um checkbox em um formulário, devemos utilizar o elemento `input` com `type` igual a `checkbox`. Ao utilizar esse componente, é importante definir um valor para o atributo `value`. No envio do formulário, esse valor é transmitido se o checkbox correspondente estiver marcado. Eventualmente é interessante

agrupar um determinado conjunto de checkboxes. Por exemplo, considere um formulário que coleta as linguagens de programação que os usuários conhecem. Para cada linguagem, podemos definir um checkbox. Para agrupar esses checkboxes, basta definir o atributo name como mesmo valor para eles. Para adicionar um radio em um formulário, devemos utilizar o elemento input com type igual a radio. Ao utilizar esse componente, é importante definir um valor para o atributo value. No envio do formulário, esse valor é transmitido ao radio correspondente se estiver marcado.

```
<body>
  <form action="pagina.html">
    JAVA <input id="java_id" type="checkbox" value="java" name="linguagens" id="">
    PHP <input id="php_id" type="checkbox" value="php" name="linguagens" id="">
    RUBY <input id="ruby_id" type="checkbox" value="ruby" name="linguagens" id=""> <br>
    EDGE <input id="edge_id" type="radio" value="edge" name="linguagens" id="">
    FIREFOX <input id="firefox_id" type="radio" value="firefox" name="linguagens" id="">
    CHROME <input id="chrome_id" type="radio" value="chrome" name="linguagens" id="">
  </form>
</body>
```



BOTÕES

Botões genéricos

Para adicionar um botão genérico em um formulário, podemos utilizar o elemento input com type igual a button. As ações desse tipo de componente são definidas com JavaScript. Os textos desses botões são definidos com atributo value. Outra forma de adicionar um botão genérico em um documento HTML é utilizar o

`<input id="botao_id" type="button" value="botão">` elemento `<button>` com type igual a `button`. Diferentemente do elemento `input`, o elemento `button` permite a criação de botões com imagens além de texto.

```
<form action="pagina.html">
  <button id="botao_id" type="button">Botão Genérico
  
</button>
</form>
```

BOTÕES DE RESET

Para adicionar um botão de reset em um formulário, podemos utilizar o elemento `input` com type igual a `reset`. Esse tipo de botão reinicia os dados do formulário. Os textos desses botões são definidos com o atributo `value`. `<input id="botao_id" type="reset" value="reiniciar">` Outra forma de adicionar um botão de reset em um documento HTML é utilizar o elemento `button` com type igual a `reset`. Diferentemente do elemento `input`, o elemento `button` permite a criação de botões com imagens além de texto.

```
<button id="botao_id" type="reset">
  
</button>
```

BOTÕES DE UPLOAD

Para adicionar um botão de upload em um formulário, podemos utilizar o elemento `input` com type igual a `file`. Esse tipo de botão permite selecionar um arquivo para um eventual upload. O formulário que contém esse botão deve possuir o atributo `enctype` com o valor `multipart/formdata`.

`<input id="botao_id" name="file" type="file">`

IMAGEM COMO BOTÃO DE SUBMIT

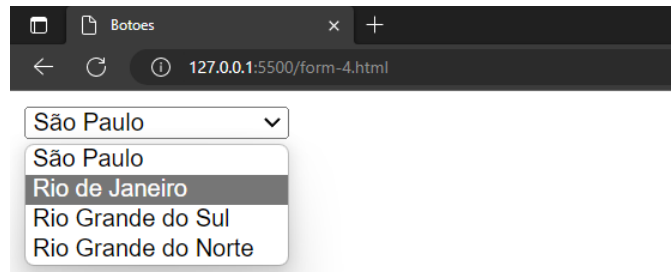
Uma imagem pode funcionar como um botão de submit. Para isso, devemos utilizar o elemento input com type igual a image. O caminho absoluto ou relativo da imagem que será utilizada deve ser definido com o atributo src. Um texto alternativo deve ser definido como atributo alt. Esse texto pode ser utilizado caso ocorra algum problema no carregamento da imagem.

```
<form action="parametros.php" method="post"
enctype="multipart/form-data"> <input id="botao_id" alt="enviar"
type="image" src="submit.png"> </form>
```

LISTA DROP-DOWN

Muitos formulários permitem que os usuários selecionem um ou mais itens de uma lista de opções. Essa seleção pode ser realizada através de uma lista drop-down. Para adicionar esse tipo de componente, devemos utilizar o elemento select. As opções devem ser definidas no conteúdo do elemento <select> e elas são adicionadas com o elemento option. O conteúdo do elemento option é exibido para os usuários. Esse elemento possui um atributo chamado value. Quando o formulário for enviado, o valor do atributo value é transmitido se a opção correspondente foi selecionada pelo usuário.

```
<select id="estados" name="estado">
  <option value="SP">São Paulo</option>
  <option value="RJ">Rio de Janeiro</option>
  <option value="RS">Rio Grande do Sul</option>
  <option value="RN">Rio Grande do Norte</option>
</select>
```



VALIDAÇÃO

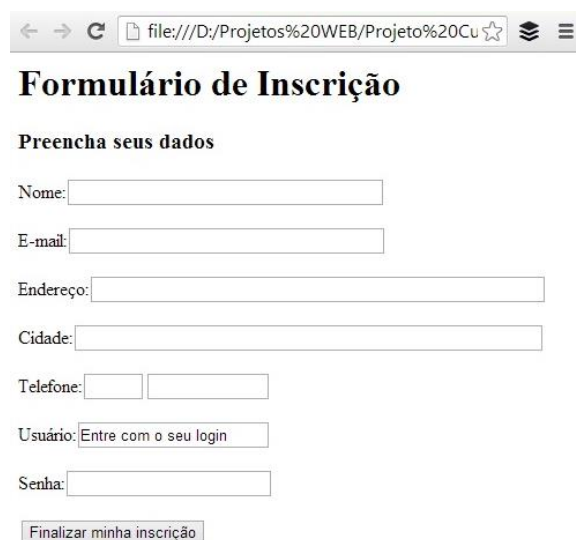
Alguns recursos para realizar a validação dos campos de um formulário foram adicionados no HTML5. Essas validações ocorrem antes do envio dos formulários. Por padrão, algumas validações são realizadas automaticamente de acordo com o tipo do campo. Por exemplo, os navegadores verificam se o conteúdo de uma caixa de e-mail se é um e-mail válido. Outras validações devem ser denidas explicitamente. Por exemplo, se um determinado campo é obrigatório, devemos utilizar o atributo **required**. Esse atributo não precisa de valor.

```
<form action="pagina.html">
  <input id="email_id" name="email" type="email" placeholder="seu e-mail">
  <input id="url_id" name="url" type="url" placeholder="seu site">
  <input id="nome_id" name="nome" type="text" placeholder="seu nome"
  required>
</form>
```

A screenshot of a web browser window showing a form. The address bar shows '127.0.0.1:5500/form-4.html'. The form contains three input fields: 'seu e-mail', 'seu site', and 'Obrigatório'. The 'Obrigatório' field is highlighted with a red border.

EXERCÍCIOS

1- Utilizando-se do conteúdo estudado nessa aula, escreva um código HTML que reproduza um formulário de inscrição semelhante ao da imagem abaixo.



A screenshot of a web browser window displaying a registration form. The address bar shows a file path: file:///D:/Projetos%20WEB/Projeto%20Cu. The form is titled "Formulário de Inscrição" and includes a section "Preencha seus dados" with input fields for Name, E-mail, Address, City, Telephone, User (with a placeholder "Entre com o seu login"), and Password. A "Finalizar minha inscrição" button is at the bottom.

file:///D:/Projetos%20WEB/Projeto%20Cu

Formulário de Inscrição

Preencha seus dados

Nome:

E-mail:

Endereço:

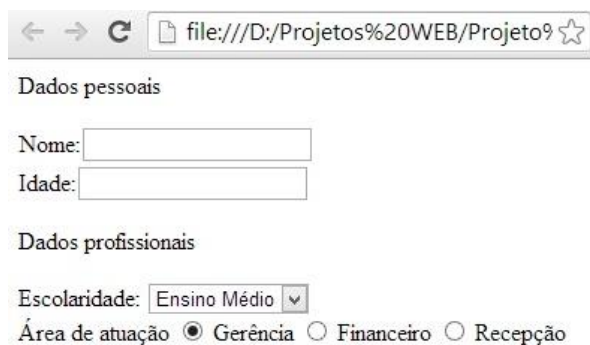
Cidade:

Telefone:

Usuário:

Senha:

2- Reproduza o código da seguinte imagem a baixo:



A screenshot of a web browser window displaying a form. The address bar shows a file path: file:///D:/Projetos%20WEB/Projeto%. The form has two sections: "Dados pessoais" with input fields for Name and Age, and "Dados profissionais" with a dropdown for Education (set to "Ensino Médio") and radio buttons for Area of Activity (Gerência, Financeiro, Recepção). The "Gerência" radio button is selected.

file:///D:/Projetos%20WEB/Projeto%

Dados pessoais

Nome:

Idade:

Dados profissionais

Escolaridade:

Área de atuação ☒ Gerência ☐ Financeiro ☐ Recepção

3- Qual o atributo usamos para tornar um campo obrigatório em nosso formulário?

4- Você já viu na aula anterior quais são os elementos usados para se criar uma lista em drop-down. Crie uma lista constando o título: Cursos Profissionalizantes: Na lista drop-down com as seguintes informações: Açougueiro, Operador de caixa, Camareiro, Estoquista e Babá.

CSS

Cascading Style Sheets, segundo o **W3C** é definido como: **“Folha de estilo em cascata”**. É um mecanismo usado para estilizar uma página web.” Então podemos perceber que ele foi criado para tirar formas de apresentação até então atribuídas a elementos e atributos no HTML. Com ele é possível definir em uma página HTML, cores e tipo de fontes para textos, fundos, bordas, espaçamentos, posicionar blocos, até fazer animações, entre outros. Saber trabalhar bem com CSS é fundamental para que possamos formatar bem uma página web, hoje em dia, nos padrões atuais de desenvolvimento web, não usamos mais códigos “misturados”, ou seja, tipos diferentes de linguagem como HTML e CSS misturados pelo código afora, tudo no seu devido lugar, isso facilita no desenvolvimento, manutenção e portabilidade do código.

Para que serve o CSS?

Agora que você já compreendeu como surgiu e como é, basicamente, seu funcionamento, está na hora de entender para que serve o CSS. A seguir, elencamos alguns pontos que o fazem essencial na programação.

SINTAXE BÁSICA

A sintaxe padrão de CSS é bastante simples, precisamos indicar um seletor que é uma tag, id ou classe, e dentro das chaves inserimos os comandos referente à formatação.

```
p { color: #F00; font-size: 20pt; }  
seletor { propriedade: valor; propriedade: valor; }
```

Observe que as cores nesse exemplo estão fazendo referência a sintaxe básica do CSS, para um melhor entendimento.

Seletor ID

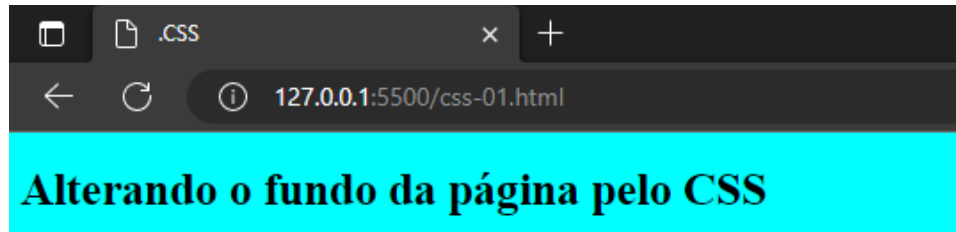
É um seletor individual usado para vincular somente um elemento por página web. Ele não pode ser usado em dois ou mais elementos. Para construí-lo basta que você crie um nome precedido pelo símbolo #.

```
<style>  
  #fundo-da-pagina {  
    background-color: aqua; /*Altera a cor de fundo da Página*/  
  }  
</style>
```

Agora veja como você irá construir seu código html para que ele funcione:

```
<body id="fundo-da-pagina">  
  <h1>Alterando o fundo da página pelo CSS</h1>  
</body>  
</html>
```


Resultado será esse:



Observe a imagem a cima, a cor do fundo foi alterada ou seja, adicionado o seletor ID no <body> do html e aplicado o css. Veremos mais a frente quais são as maneiras de ser usar o css.

Seletor class

Este seletor possibilita o uso em mais de um elemento da mesma página. Indicado quando você precisa atribuir algumas propriedades iguais em elementos diferentes. Para construí-lo basta que você crie um nome precedido por um **ponto**.

```
<style>
.cor-da-fonte {
    color: white;
}
</style>
```

Agora veja como você irá construir seu código html para que ele funcione:

```
<body id="fundo-da-pagina">
  <h1 class="cor-da-fonte">Alterando a cor da letra</h1>
</body>
</html>
```

O resultado será esse:



Vários seletores

Podemos indicar mais de um seletor para receber um conjunto de formatações CSS, basta indicar estes seletores separados por vírgulas, como no código de exemplo:

```
<style>
h1, h2, h3, p{
    color:#F00;font-size:20pt;
}
</style>
```

Mesmo que os seletores sejam de tipos diferentes podemos indicar como grupos.

Unidades para valores

Existem inúmeros valores específicos para propriedades. Só que alguns são de uso comum para muitas propriedades e vira e mexe você estará usando.

Veja uma pequena relação a seguir: **px** – é a unidade de pixels. Muita usada em dimensões de blocos e fontes.

Exemplo:

font-size: 14px;

width: 200px;

em – Indicada para tamanhos de fontes. Ao usarmos o tamanho padrão do dispositivo do usuário podemos expressar o tamanho da fonte de outros elementos através dessa unidade.

Assim um elemento com tamanho de fonte de 2em (font-size:2em} tem duas vezes o tamanho padrão. Caso o tamanho padrão for 16 pixels, uma fonte de 2em equivalerá a 32 pixels. Exemplo de sintaxe: font-size: 1.25em; font-size: 0.5em;

Hexadecimal – é um sistema de numeração que representa os números em base 16, empregando assim 16 símbolos. Este sistema é composto por 10 números, de 0 a 9, e seis letras adicionais de A a F. Usamos no CSS para atribuir valores para propriedades de cor para fontes (color), cor de fundo (background-color), cor de borda (border-color) entre outras.

Exemplo:

```
background-color: #cccccc;
```

```
color: #ff0000;
```

Neste caso temos a cor vermelha para textos. O valor em hexadecimal em CSS é precedido do símbolo #.

Declaração única

% – Unidade de porcentagens. Ela é relativa ao bloco onde o elemento está contido.

Se um bloco tem 150 pixels e for atribuído a outro bloco contido nele uma largura de 50%, equivalerá a 75 pixels. Exemplo font-size: 120%; width: 50%; Acontece quando você abrevia declarações reunindo todos os valores destas em um só.

E isto pode ser aplicado para fontes, margens, preenchimentos, fundos, bordas, etc. Por exemplo: Border é a propriedade responsável por reunir em uma declaração única os valores das propriedades, border-style, border-width e border-color.

Exemplo: Temos as seguintes declarações para uma div:

```
div {  
  
border-style: solid; border-width: 1px; border-color: black;  
  
}
```

Podemos implementar todas essas declarações acima em uma só, fazendo com que o nosso código fique mais limpo e ágil.

```
div {  
  
border: solid 1px black;  
  
}
```

FORMAS DE SE UTILIZAR O CSS

Existem algumas formas de vincular as folhas de estilos (CSS) em um documento HTML. Essas formas também definem a localização de cada tipo. Classificam-se em **três** tipos:

INLINE – INCORPORADO – EXTERNO

INLINE

O CSS inline é usado para dar estilo a um elemento HTML específico. Para este estilo de CSS< você somente vai precisar

adicionar o atributo style para cada tag HTML, sem usar os seletores.

Este tipo de CSS não é realmente recomendado, já que cada tag HTML precisa ser estilizada de maneira individual. Gerenciar o seu site pode se tornar uma tarefa bem difícil de você só usa o CSS inline.

Contudo, o CSS inline no HTML pode ser útil para algumas situações. Por exemplo, sem casos onde você não tem acesso aos arquivos CSS ou precisa aplicar estilos para um elemento único.

Vamos dar uma olhada num exemplo. Aqui, nós adicionamos um CSS inline para as tags <p> e <h1>:

```
<!DOCTYPE html>
<html>
  <body style="background-color:black;">

    <h1 style="color:white;padding:30px;">Hostinger Tutorials</h1>
    <p style="color:white;">Something usefull here.</p>

  </body>
</html>
```

Vantagens do CSS Inline :

Você pode inserir elementos CSS de maneira fácil e rápida numa página HTML. É por isso que esse método é útil para testar e pré-visualizar mudanças, assim como executar correções rápidas no seu site. Você não precisa criar e fazer upload de um documento separado como no estilo externo.

Desvantagens do Inline CSS:

Adicionar regras CSS para cada elemento HTML consome muito tempo e faz a sua estrutura HTML ficar bagunçada. Estilizar

múltiplos elementos pode afetar o tamanho da sua página o tempo para download.

INCORPORADO

O CSS incorporado ou interno requer que você adicione a tags `<style>` na seção `<head>` do seu documento HTML.

Este estilo de CSS é um método efetivo de estilizar uma única página. Contudo, usar esse estilo em múltiplas páginas pode consumir muito tempo, já que você precisa definir as regras CSS para cada página do seu site.

Confira abaixo como você pode usar o CSS interno:

1. Abra a sua página HTML e localize a tag de abertura `<head>`.
2. Coloque o seguinte código logo depois da tag `<head>`

```
<style type="text/css">
```

3. Adicione as regras CSS numa nova linha. Aqui está um exemplo:

```
body {  
    background-color: blue;  
}  
h1 {  
    color: red;padding: 60px;  
}
```

4. Digite a seguinte tag de fechamento:

```
</style>
```

O seu arquivo HTML ficará parecido com isso:

```
<!DOCTYPE html>
<html>
<head>
<style>
body {
  background-color: blue;
}
h1 {
  color: red;
  padding: 60px;
}
</style>
</head>
<body>

<h1>Curso Webdesigner</h1>
<p>Este é o nosso parágrafo.</p>

</body>
</html>
```

Vantagens de CSS Incorporado ou Interno:

Classes e **seletores de IDs** podem ser usados por stylesheet interno. Confira um exemplo abaixo:

```
.class {
  property1 : value1;
  property2 : value2;
  property3 : value3;
}
#id {
  property1 : value1;
  property2 : value2;
  property3 : value3;
}
```

Não há necessidade de carregar vários arquivos. HTML e CSS podem estar no mesmo arquivo.

Desvantagens de CSS Interno:

Adicionar o código para o documento HTML pode aumentar o tamanho da página e o tempo de carregamento.

CSS EXTERNO

Com o CSS externo, você vai linkar as páginas da internet com um arquivo **.css** externo, que você pode criar usando qualquer editor de texto no seu dispositivo (ex.: vscode).

Este tipo de CSS é um método mais eficiente, especialmente se você está estilizando um site grande. Ao editar um arquivo **.css**, você pode modificar um site inteiro de uma só vez.

Siga os passos abaixo para usar o CSS externo:

1. Crie um novo arquivo **.css** com um editor de texto e então adicione regras de estilo.

Veja o exemplo:

```
.xleftcol {  
  float: left;  
  width: 33%;  
  background:#809900;  
}  
.xmiddlecol {  
  float: left;  
  width: 34%;  
  background:#eff2df;  
}
```

2. Na seção `<head>` da sua planilha HTML, adicione uma referência para o seu arquivo **.css** logo depois da tag `<title>`:

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


Não se esqueça de modificar o `style.css` com o nome do seu arquivo `.css`.

Vantagens de CSS Externo:

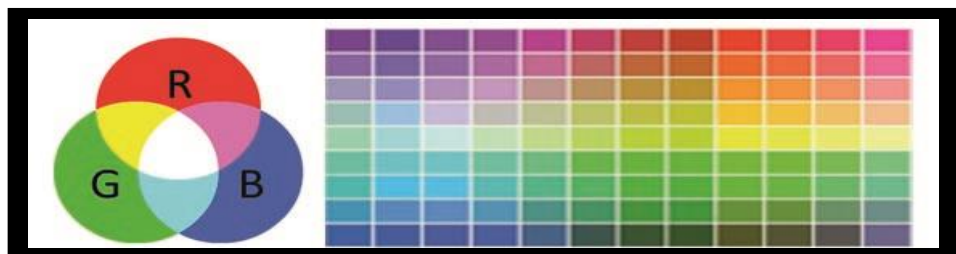
Como o código CSS está num documento separado, os seus arquivos HTML terão uma estrutura mais limpa e serão menores o mesmo arquivo `.css` pode ser usado em várias páginas.

Desvantagens de CSS Externo:

Até que o CSS externo seja carregado, a página pode não ser processada corretamente. Fazer o upload ou links para múltiplos arquivos CSS pode aumentar o tempo de download do seu site.

CORES

Trabalhar com cores para web é uma tarefa um pouco trabalhosa, principalmente no início quando ainda não se tem o costume e o domínio do assunto, como podemos configurar as cores usando caracteres hexadecimais. O sistema de cores usado na web se chama RGB, que são as letras para Red Green Blue (Vermelho, Verde e Azul). Esse é o sistema de cores usado por qualquer tela que não seja monocromática, seja CRT (tubo), LCD, Led, enfim, independentemente do tipo, veja a seguir algumas formações de pixels RGB.



Mas se a imagem a cima é composta somente de pixels de três cores, de onde vem as outras cores? Acredite, todas as cores que vemos na figura vem da mistura somente destas três cores, do branco ao preto!

O que acontece é que alteramos a intensidade destes pixels para formar as demais cores, veja bem, para formar o vermelho, basta diminuir a intensidade ao mínimo dos pixels Green (verde) e Blue (azul) e aumentar ao máximo a intensidade do pixel Red (vermelho), então teremos a cor vermelho, a mesma coisa para o verde e para o azul. Para formar as cores em RGB (vermelho, verde, azul), basta “misturar” as cores, pela intensidade de cada canal. Sendo “00” o valor mínimo e “FF” o valor máximo, respectivamente, a intensidade (força) mínima e máxima do canal. Assim, se informarmos Red=00, Green=00 e Blue=00, intensidade mínimo para todos os canais, vamos formar a cor preto, o contrário Red=FF, Green=FF e Blue=FF, intensidade máxima para todos os canais, vamos formar o branco.

Desta maneira vamos ver as cores básicas.

Preto = #000000

Branco = #FFFFFF

Vermelho = #FF0000

Verde = #00FF00

Azul = #0000FF

Misturando dois canais podemos obter as cores do sistema CMYK (Ciano, Magenta, Amarelo e Preto), vejamos:

Ciano = #00FFFF

Magenta = **#FF00FF**

Amarelo = **#FFFF00**

Agora que já vimos as 8 possibilidades de cores usando as intensidades mínima (00) e máxima (FF), vamos formar outras cores usando intensidades diferentes.

Cor laranja = **#FF8000** Confuso?

Quais tintas misturamos para criar a cor laranja? Vermelho e amarelo! Aqui aconteceu a mesma coisa!

Vejamos:

Vermelho = **#FF0000**;

Amarelo = **#FFFF00**;

Agora vamos adicionar um canal ao outro.

Cor	Red = vermelho	Green = Verde	Blue = Azul
Vermelho	FF	00	00
Amarelo	FF	FF	00
+	+	+	+
Resultado	FF	FF	00

R = Como o valor máximo é FF, então a soma de FF com FF é FF.

G = 00 + FF = FF

B = 00 + 00 = 00

Note então que a soma de vermelho (**#FF0000**) com amarelo (**#FFFF00**) foi igual ao próprio amarelo (**#FFFF00**), mas isso não resultou na cor laranja! Calma, vamos corrigir.

O que deu errado é que usamos o amarelo na sua intensidade máxima, vamos diminuir a intensidade do amarelo pela metade.

Amarelo menos intenso = **#808000** (Red = 80, Green = 80, Blue = 00)

Agora vamos somar novamente.

Cor	Red = vermelho	Green = Verde	Blue = Azul
Vermelho	FF	00	00
Amarelo menos intenso	80	80	00
+	+	+	+
Resultado	FF	80	00

$R = FF + 80 = FF$

$G = 00 + 80 = 80$

$B = 00 + 00 = 00$

Então, agora obtivemos a cor laranja **#FF8000**.

Vamos a mais uma soma? Como obter a cor roxo? Misturando vermelho e azul! Vamos a esta mistura.

Vermelho menos intenso = **#800000**

Azul = **#0000FF**

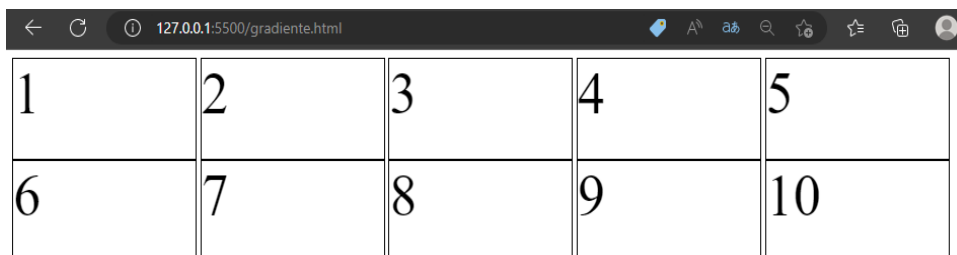
Bom, agora que entendemos um pouco melhor essa questão de cores RGB, iremos dar continuidade a outros assuntos relevantes ao nosso curso.

COR GRADIENTE

Esta é uma nova e incrível possibilidade implementada em CSS3, trabalhar com cores gradientes diretamente pelo código sem a possibilidade de precisar inserir uma imagem é muito interessante. Vamos ver como funcionam os gradientes em CSS. A sintaxe de utilização dos gradientes é: **tipo-de-gradiente** (direção, cor1, cor2, ...); São dois tipos de gradiente que podemos trabalhar, **linear** e **gradiente**, com uma pequena variação cada que é **repeating-linear-gradient** e **repeating-radial-gradient**. Vamos ao nosso código de exemplo, primeiro vamos adicionar 10 divs (como já vimos anteriormente div é um elemento de divisão um container para conteúdo de fluxo) e depois vamos aplicar os gradientes ao

background de cada div.

```
<style>
  div{
    width: 200px;
    height: 100px;
    border: #000 solid 1px;
    display: inline-table;
    color: #000;
    font-size: 60px;
  }
</style>
</head>
<body>
  <div id="grad1">1</div>
  <div id="grad2">2</div>
  <div id="grad3">3</div>
  <div id="grad4">4</div>
  <div id="grad5">5</div><br>
  <div id="grad6">6</div>
  <div id="grad7">7</div>
  <div id="grad8">8</div>
  <div id="grad9">9</div>
  <div id="grad10">10</div>
</body>
</html>
```



1	2	3	4	5
6	7	8	9	10

Gradativamente vamos aplicando os gradientes nas divs. Observem e leiam atentamente as variações tanto linear quanto radial nas informações a baixo:

LINEAR – GRADIENTE

O gradiente linear possui uma variação linear de um ponto inicial A até um ponto final B, para configurar este gradiente basta informar o sentido da variação linear e as cores.

Vamos a algumas configurações:

linear-gradient(to bottom, #F00, #FF0); ➡ Variação linear na vertical de cima para baixo com duas cores, vermelho e amarelo.

linear-gradient(45deg, #F00, #00F, #FF0); ➡ Variação linear em 45 graus com três cores, vermelho, azul e amarelo.

linear-gradient(to bottom, #F00 0%, #00F 25%, #FF0 50%);
➡ Variação linear na vertical de cima para baixo com três cores em posições definidas, vermelho no início, azul em 25% e amarelo em 50%.

linear-gradient(to right, #F00, #FF0); ➡ Variação linear na horizontal da esquerda para direita com duas cores, vermelho e amarelo.

RADIAL – GRADIENTE

O gradiente radial possui uma variação circular/radial do centro para fora, para configurar este gradiente informamos simplesmente as cores. Vamos a algumas configurações.

radial-gradient(#F00, #FF0); ➡ Variação radial do centro para fora com duas cores, vermelho e amarelo.

radial-gradient(#F00, #00F, #FF0); ➡ Variação radial do centro para fora com três cores, vermelho, azul e amarelo.

radial-gradient(#F00 0%, #00F 25%, #FF0 50%); ➡ Variação radial do centro para fora com três cores em posições definidas, vermelho no início, azul em 25% e amarelo iniciando no meio.

radial-gradient(circle, #F00, #FF0); ➡ Variação radial circular, independente do formato retangular, a variação do gradiente não será oval e sim circular, do centro para fora com duas cores, vermelho e amarelo.

REPEATING - LINEAR - GRADIENT & REPEAT - RADIAL - GRADIENT

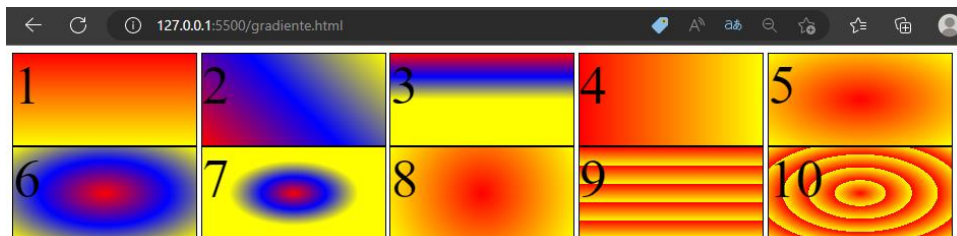
Estes gradientes são uma variação dos gradientes linear e radial, eles funcionam repetindo o gradiente configurado até preencher.

repeating-linear-gradient(#F00, #FF0 20%); ➡ Repete o gradiente linear.

repeating-radial-gradient(#F00, #FF0 20%); ➡ Repete o gradiente radial. A seguir vamos aplicar estes gradientes no background das divs e conferir os resultados.

Nossa próxima etapa será adicionar esses códigos ao nosso arquivo html.

```
<style>
div {
  width: 200px;
  height: 100px;
  border: #000 solid 1px;
  display: inline-table;
  color: #000;
  font-size: 60px;
}
#grad1 {
  background: linear-gradient(to bottom, #F00, #FF0);
}
#grad2 {
  background: linear-gradient(45deg, #F00, #00F, #FF0);
}
#grad3 {
  background: linear-gradient(to bottom, #F00 0%, #00F 25%, #FF0 50%);
}
#grad4 {
  background: linear-gradient(to right, #F00, #FF0);
}
#grad5 {
  background: radial-gradient(#F00, #FF0);
}
#grad6 {
  background: radial-gradient(#F00, #00F, #FF0);
}
#grad7 {
  background: radial-gradient(#F00 0%, #00F 25%, #FF0 50%);
}
#grad8 {
  background: radial-gradient(circle, #F00, #FF0);
}
#grad9 {
  background: repeating-linear-gradient(#F00, #FF0 20%);
}
#grad10 {
  background: repeating-radial-gradient(#F00, #FF0 20%);
}
```



Alguns browsers tem uma sintaxe específica para uso do gradiente, tenha bastante atenção no resultado de cada navegador que você optar por usar.

PRINCIPAIS PROPRIEDADES PARA TEXTOS

font-size: Esta define o tamanho da fonte, Os valores mais usados são px, em e porcentagem.

Observe a sintaxe:

```
p {  
font-size: 14px  
}
```

font-family: Define a família de fontes. Com ela você pode declarar uma fonte específica e uma genérica. É indicado que toda vez que você declarar uma fonte específica, declare também na sequência uma fonte de família genérica que corresponda à fonte específica declarada. Isso porque se a fonte específica não estiver disponível no dispositivo do usuário o navegador terá a liberdade para escolher uma fonte semelhante da família genérica. Por exemplo, se você declarou a fonte específica Verdana, declare também a genérica sans-serif porque Verdana é uma fonte sem serifa.

Observe a sintaxe:

```
p {  
font-family: Verdana, sans-serif;  
}
```

Font-weight: Esta propriedade serve para determinar o peso da fonte. É usada bastante com o valor “bold” para negritar trechos de textos. Os valores numéricos para essa propriedade vão de 100 a 900, mas você pode usar também normal, bold, bolder, lighter.

Observe a sintaxe:

```
p {  
font-weight: bold;  
}
```

Font-style: Usada para dar estilo. Você pode usar esta propriedade para valores “normal”, “oblique” ou “italic”.

Observe a sintaxe:

```
p {  
font-style: italic;  
}
```

Line-height: É uma propriedade de dimensionamento que permite estipular espaçamento entre linhas. Os valores usados para esta propriedade podem ser de diferentes unidades de medidas CSS px, cm, em, %, etc. E também um número (1, 2, 3).

```
p {
```

line-height: 36px;

}

Font: A propriedade “font” é indicada se quer diminuir o código reunindo todos os valores das propriedades acima em uma só declaração.

p {

font:bold 30px Verdana;

}

text-align: Esta é a propriedade usada para determinar o tipo de alinhamento que um texto possuirá. Os valores para esta propriedade são: left, right, center e justify.

Observe a sintaxe:

p {

text-align: center;

}

p {

color: #424242; /* cor cinza */

}

Color: A propriedade para inserir cor no texto. Você pode usar valores hexadecimais 67 nessa propriedade.

Observe a sintaxe:

letter-spacing: Controla o espaçamento entre as letras de um texto.

Observe a sintaxe:

p{

letter-spacing: 5px;

}

word-spacing: Controla o espaçamento entre as palavras de um texto. Exemplo:

p{

word-spacing: 10px;

}

text-decoration: Define um efeito decorativo no texto. Podendo entre eles ser: none (sem decoração); underline (sublinhado); line-through (linha cortando o texto); e blink (efeito piscante). Exemplo:

p{

text-decoration: underline;

}

text-transform: Controla os efeitos de capitalização do texto. Com ela, podemos definir a caixa das letras (minúsculas e maiúsculas). Os valores possíveis são: capitalize – transforma o primeiro carácter de cada palavra em maiúscula uppercase – transforma todas as letras de todas as palavras em maiúsculas lowercase – transforma todas as letras de todas as palavras em minúsculas none – cancela algum valor que tenha sido herdado text-transform: uppercase;

text-indent: Define o tamanho da endentação para a primeira linha do texto contida em um bloco, ou seja, o deslocamento para a direita de um parágrafo.

Exemplo:

```
p{  
text-indent: 20px;  
}
```

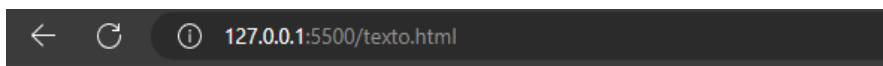
SHADOW – EFEITO DE SOMBRA

Existem duas propriedades para trabalharmos com sombras, são elas, **text-shadow** e **box-shadow**, obviamente a primeira para aplicar sombras em textos e a segunda nos demais elementos com <div> por exemplo.

text - shadow

Vamos começar com um código para sombra básica, simplesmente adicionando uma sombra a uma distância de 5 pixels na horizontal e 5 pixels na vertical.

```
<!doctype html>  
<html lang="pt-br">  
<head>  
  <title>Texto CSS</title>  
  <meta charset="UTF-8">  
  <style>  
    h1{  
      text-shadow:5px 5px;  
    }  
  </style>  
</head>  
<body>
```

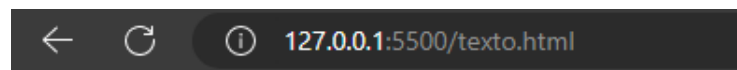


CURSO WEBDESIGNER

Observe o resultado acima, simples, text-shadow: distância Horizontal distância Vertical; Mas o resultado final foi uma sombra muito sólida e pesada, vamos a outras configurações para a sombra. text-shadow: distância Horizontal distância Vertical ; A propriedade consiste no embaçamento da sombra.

Vamos alterar nosso código e melhorar a sombra.

```
<style>
  h1 {
    color: rgb(107, 226, 244);
    text-shadow: 3px 3px 6px rgb(49, 49, 49);
  }
</style>
```



CURSO WEBDESIGNER

Vale destacar que podemos adicionar mais de uma sombra no mesmo texto, fazendo com que essa estilização fique de acordo com o projeto em que você esteja desenvolvendo, vamos adicionar duas sombras.

```
<style>
  h1{
    color:rgb(29, 29, 29);
    text-shadow:3px 3px 6px rgb(161, 241, 56), -3px -3px 6px
    rgb(252, 118, 0);
  }
</style>
```

Observe o resultado:



Podemos adicionar mais sombras, três, quatro, cinco, quantas forem preciso, mas lembre-se que exagerar nos efeitos torna o elemento mais difícil de ser renderizado pelo navegador.

Para finalizar iremos alterar a cor da fonte, ou seja, a cor da nossa letra na mesma cor do background da nossa página (branco), assim fazendo com que o brilho realça as letras.

```
<style>
  h1{
    font-size:50px;
    color:#FFF;
    text-shadow:1px 1px 2px #000, 0px 0px 7px rgb(83, 165, 220), 0px 0px 20px
    rgb(20, 24, 232);
  }
</style>
```

Observe o resultado final:



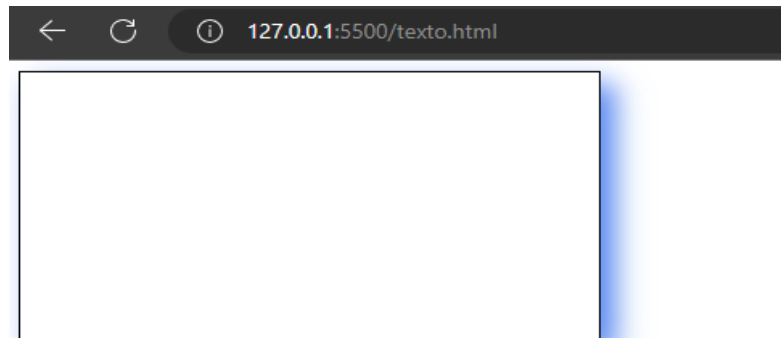
E assim podemos “brincar” criando vários efeitos, basta usar a criatividade.

Também podemos colocar uma ou mais sombras em uma caixa de texto, dando aquele efeito de sobreposição. Vejamos como:

BOX-SHADOW

A propriedade box-shadow tem a mesma sintaxe de text-shadow, porém aplica sombras aos outros elementos diferentes de texto, como <div> por exemplo.

```
<style>
  div {
    width: 400px; /*largura*/
    height: 200px; /*altura*/
    background-color: #FFF; /*cor do fundo */
    border: #000 solid 1px; /*cor e espessura da borda*/
    box-shadow: 10px 10px 20px rgb(82, 132, 241);
  } /*tamanho da caixa com sombra */
</style>
```



Lembre-se, podemos aplicar box-shadow em qualquer elemento que não seja um texto, para textos devemos usar text-shadow.

Observação:

*É muito útil podermos adicionar textos, comentários em nosso código que não serão interpretados, tanto no HTML quanto no CSS temos uma forma simples de adicionar esses comentários, no caso do **CSS**, basta usar **/* para iniciar e */ para finalizar**, todo conteúdo que for inserido como comentário não será interpretado como código de formatação.*

EXERCÍCIOS

1. Quais as formas de vincular as folhas de estilos (CSS) em um documento HTML?
2. Qual a tag usada para implementar CSS incorporado?
3. Descreva o código completo para implementar o CSS externo e sua localização no arquivo html:
4. Qual o sistema de cores usado na web ?
5. Quais os dois tipos de cores gradiente?
6. Qual a propriedade usada para definir o tamanho do texto?
7. **text-align:** Esta é a propriedade usada para determinar o tipo de alinhamento que um texto quais são os seus valores?
8. Qual a propriedade usada para inserir cor ao texto:
9. Existem duas propriedades para trabalharmos com sombras, quais são elas?
10. Crie uma <div> adicione um <h1>título, com <p> parágrafo entre 6 e 12 palavras, adicione bordas, cor de fund, largura, altura e sombra na <div>.

PROPRIEDADE PARA DIMENSÕES

Elementos estilizados com CSS possuem largura e altura.

Muitas vezes elas não precisam ser especificadas, pois resultam de outros fatores como tamanho do conteúdo na tela e

interferência de outras propriedades. Porém sempre existirá um momento em que você terá que estipular dimensões de um bloco, principalmente quanto à largura.

Veja um exemplo:

```
div {  
width: 400px;  
height: 200px;  
}
```

“**width**” define a largura do bloco e “**height**” a altura, quando você define valores de porcentagem, por exemplo, para uma largura, seu valor será calculado baseado na largura do elemento pai. Um dos possíveis valores é o “auto” (**width: auto**). Quando usado faz que o elemento se ajuste dentro do box pai.

PARA FUNDOS DO SITE E DE BLOCOS

A propriedade responsável por atribuir valores para características de fundo de sites, seções e blocos é a **background**. Com ele é possível atribuir valores de cor, posicionamento, imagens para fundos, entre outros.

Um exemplo comum:

```
div {  
background: #fff000 url(imagem-de-fundo.png) repeat;  
}
```

Este exemplo abaixo define uma imagem de fundo “imagem-de-fundo.png”. Ela se repetirá em toda a extensão do elemento no eixo x e y, e atrás desta temos uma cor de fundo amarela. Essa cor será notada se a imagem em png tiver transparência ou enquanto a imagem de fundo estiver sendo carregada no site.

```
div {
```

```
background: #fff000 url(imagem-de-fundo.png) repeat;  
}
```

Esta regra poderia ser obtida através de variantes de background.

Veja:

```
div {  
background-image: url(imagem-de-fundo.png); /* Defi ne a  
imagem de fundo */  
background-color: #fff000; /* Defi ne a cor do fundo amarela */  
background-repeat: repeat; /* Defi ne a se a imagem do fundo  
deve repetir ou não, ou somente no eixo x, ou no y. No caso se  
repete em todos os eixos.  
}
```

```
div {  
background-image: url(imagem-de-fundo.png); /* Defi ne a  
imagem de fundo */  
background-color: #fff000; /* Defi ne a cor do fundo amarela */  
background-repeat: repeat; /* Defi ne a se a imagem do fundo  
deve repetir ou não, ou somente no eixo x, ou no y. No caso se  
repete em todos os eixos.  
}
```

Outras variações:

background-attachment – define se a imagem fica fixa ou não enquanto rolamos a tela;

background-position – define onde a imagem de fundo é posicionada; **background-clip** – define a área onde a imagem de fundo é aplicada; **background-origin** – define a posição de

origem da imagem em um elemento; **background-size** – define as dimensões da imagem de fundo.

PARA MARGENS

As margens em CSS servem para que um bloco se distancie de blocos vizinhos e também da extremidade do navegador. E isso é muito útil para criar áreas de respiro entre elementos, definir posicionamentos e auxiliar na diagramação de um layout quando usado em conjunto com as propriedades **width** e **height**.

A propriedade responsável por definir margens para um elemento se chama “**margin**”.

Veja um exemplo:

```
div {  
margin: 10px;  
}  
div {  
margin: 10px;  
}
```

Aqui diz que o elemento “div” deve se distanciar 10 pixels em todos os lados em relação a outros elementos ao seu redor. Essa regra equivale a esta:

```
div {  
margin-top: 10px; margin-right: 10px; margin-bottom: 10px;  
margin--left: 10px;  
}  
div {  
margin-top: 10px; margin-right: 10px; margin-bottom: 10px;  
margin--left: 10px;  
}
```

Mais obviamente a regra anteriormente é melhor por economizar código, tempo e melhorar o desempenho do site.

```
div {  
margin-top: 30px; /* aplica uma margem de 30 pixels no topo  
fazendo com que o bloco se distancie em relação ao que estiver  
acima dele */;  
}  
div {  
margin-top: 30px; /* aplica uma margem de 30 pixels no topo  
fazendo com que o bloco se distancie em relação ao que estiver  
acima dele */;  
}
```

PARA PREENCHIMENTOS

Preenchimento ou espaçamento é similar às margens. Ele também cria uma área de respiro, só que desta vez é entre o conteúdo e extremidade de um box.

A propriedade responsável por isso se chama “**padding**”.

Veja um exemplo:

```
div {  
padding: 10px;  
}  
div {  
padding: 10px;  
}
```

Da mesma forma que vimos em **margin**, o **padding** também pode ser aplicado para os lados com **padding-top**, **padding-right**, **padding-bottom** e **padding-left**.



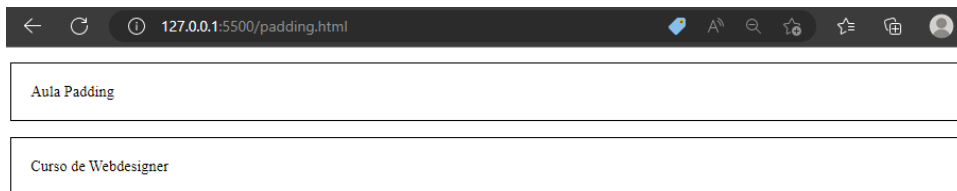
Note que a margem “cresce” para o lado de fora da borda e **padding** para o lado de dentro.

Em nosso código de exemplo vamos configurar o **padding** para o elemento `<p>`, mas podemos usar em outros elementos como `<div>` `<section>` `<article>` `<h1>` a `<h6>` e vários outros elementos, vamos ver no nosso código onde configuramos os elementos `<p>` com **padding** de 20 pixels.

Observe o exemplo a baixo:

```
<style>
  p {
    border: #000 solid 1px;
    padding: 20px;
  }
</style>
</head>
<body>
  <div>
    <p>Aula Padding</p>
    <p>Curso de Webdesigner</p>
  </div>
</body>
</html>
```

Observe o resultado a baixo:

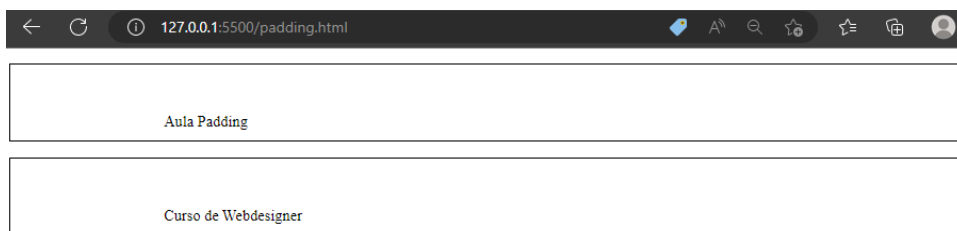


Existe um espaçamento de 20px, interno entre as bordas em relação as palavras dentro de cada box.

Iremos agora configurar valores diferentes para cada canto, começando pelo topo (top) 50px, direito(right) 0px, inferior(bottom) 10px, esquerdo(left) 160px.

```
<style>
  p{
    border:#000 solid 1px;
    padding:50px 0px 10px 160px;
  }
</style>
```

O resultado será esse:

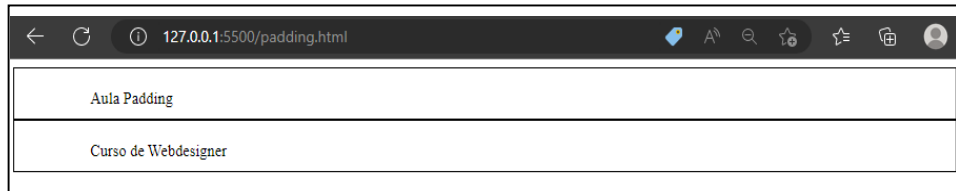


Naturalmente os elementos **<p>** tem uma margem externa, para eliminar esta margem basta configurar **margin** em 0 pixels.

Usaremos os seguintes comandos:

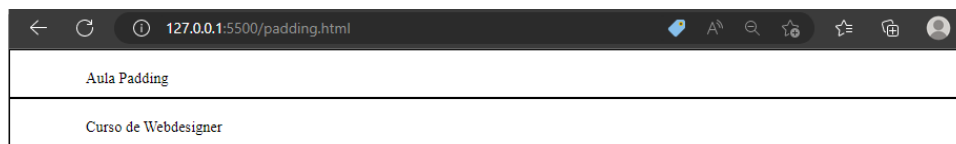
```
<style>
  p{
    border:#000 solid 1px;
    padding:20px 0px 10px 80px;
    margin:0px;
  }
</style>
```

O resultado será esse:



Observe que o espaço entre as duas tabelas deixou de existir. Vamos eliminar também a margem padrão do `<body>` configurando em zero.

```
<style>
  body{
    margin:0px;
  }
  p{
    border:#000 solid 1px;
    padding:20px 0px 10px 80px;
    margin:0px;
  }
</style>
```



Para finalizar o assunto de padding, também podemos configurar de forma individual, veja os parâmetros:

padding-top = Espaçamento interno superior

padding-right = Espaçamento interno direito

padding-bottom = Espaçamento interno inferior

padding-left = Espaçamento interno esquerdo

PARA BORDAS

Propriedade “**border**” é responsável por especificar a espessura, o estilo e a cor da borda de um elemento.

Existem variantes desta propriedade para cada tipo de valor que pretendes usar.

São elas:

Border-style: Especifica o estilo da borda.

Desta propriedade ainda é possível conseguir variações para os lados de um box:

border-top-style, **border-right-style**, **border-bottom-style** e **border-left-style**.

Exemplo:

```
div { border-style: solid;
}
div { border-style: solid;
}
```

Border-width: Especifica a espessura.

Suas variações: **border-top-width**, **border-right-width**, **border-bottom-width** e **border-left-width**.

Exemplo:

```
div {
border-width: 1px; /* Só funciona se border-style também
estiver declarado */
}
div {
border-width: 1px; /* Só funciona se border-style também
estiver declarado */
}
```

Border-color: Especifica a cor da borda.

Suas variações: **border-top-color**, **border-right-color**, **border-bottom-color** e **border-left-color**.


```
div {  
border-color: #000000; /* Só funciona se border-style também  
estiver declarado */ }
```

```
div {  
border-color: #000000; /* Só funciona se border-style também  
estiver declarado */ }
```

Exemplo:

Porém a forma mais utilizada é a que reúne as três em uma só declaração abreviada.

Veja:

```
div {  
border: 2px solid #000000;  
}
```

```
div {  
border: 2px solid #000000;  
}
```

Temos o valor da espessura (2px), do estilo (**solid**) e da cor da borda (#000000) tudo reunido na declaração.

Esta forma também representa outras variações de cada lado de um elemento. Veja:

```
div {  
border-top: 2px solid #000000; border-right: 2px solid  
#000000; border-bottom: 2px solid #000000; border-left: 2px  
solid #000000;  
}
```

```
div {  
border-top: 2px solid #000000; border-right: 2px solid  
#000000; border-bottom: 2px solid #000000; border-left: 2px  
solid #000000;  
}
```

DISPLAY

A propriedade **display** basicamente indica como um elemento será exibido, é muito importante entender o uso dessa propriedade, para construção dos nossos layouts, já até usamos algumas vezes anteriormente, mas iremos aprender como usar o display de uma forma mais direta e específica.

Todos os elementos têm um valor padrão para sua propriedade display, a maioria dos elementos tem seu display configurado em “**block**” ou “**inline-block**”.

BLOCK

Um elemento que tem o display configurado em block, sempre inicia em uma nova linha, não aceita elementos na mesma linha que ele, ou seja, quebra a linha após o elemento, e sua área ocupa toda a linha onde ele é inserido.

Podemos citar como exemplo alguns elementos que tem **display:block** configurados por padrão: **<div>**, **<h1>** até **<h6>**, **<p>**, **<form>**, **<header>**, **<footer>**, **<section>**, **<table>**.

É necessário observarmos com todo o cuidado e atenção o comportamento do display-block, pois fará toda a diferença na hora em que estamos montando no código, ou seja, na organização de todos os elementos em nosso site.

INLINE

Os elementos configurados com display:inline não iniciam em uma nova linha nem quebram de linha após o elemento, ou seja ao contrário do block, mencionado a cima, outro detalhe é que eles não ocupa a linha inteira, somente ocupam o espaço de seu conteúdo.

Podemos citar alguns exemplos como: ****, **<a>**, ****.

NONE

Este valor para propriedade display pode ser usada para ocultar o elemento pelo javascript.

INLINE-BLOCK

Parecido com inline, só que preserva o tamanho configurado do elemento.

INLINE-TABLE

O elemento é exibido como uma tabela, não iniciam em uma nova linha nem quebram de linha após o elemento, outro detalhe é que ele não ocupa a linha inteira, somente ocupam o espaço de seu conteúdo, podendo inclusive usar as propriedades de formatação do elemento `<table>`.

LIST-ITEM

Elemento se comporta como um `` inclusive passar a mostrar o marcador.

Observação: Um único valor list-item fará com que o elemento se comporte como um item de lista.

RUN-IN

Exibe um elemento como block ou inline, dependendo do contexto onde está inserido.

TABLE

O elemento se comporta exatamente igual a uma tabela, podendo inclusive usar as propriedades de formatação do elemento `<table>`.

TABLE-CAPTION

O elemento se comporta como se fosse um caption de uma tabela `<th>`.

TABLE-HEADER-GROUP

O elemento se comporta como se fosse um `<thead>` da tabela.

TABLE-FOOTER-GROUP

O elemento se comporta como se fosse um `<tfoot>` da tabela.

TABLE-ROW-GROUP

O elemento se comporta como se fosse um `<tbody>` da tabela.

TABLE-CELL

O elemento se comporta como se fosse uma célula de uma tabela, podendo receber todas as propriedades que usamos para `<td>`.

TABLE-COLUMN

Elemento se comporta como o elemento `<col>`.

TABLE-ROW

O elemento se comporta como se fosse uma linha de uma tabela, podendo receber todas as propriedades que usamos para `<tr>`.

Vejamos agora alguns exemplos.

Observe que por padrão a tag **``**, tem seu display configurado como inline.

Vejamos o código a seguir.

```
<head>
  <title>Curso de CSS</title>
  <meta charset="UTF-8">
</head>

<body>
  <p>Curso Webdesigner <span>PRAXIS</span>
  https://praxisdesenvolvimento.com.br/</p>
</body>
```



Iremos realizar algumas alterações, o display da tag `` para `block` e ver o resultado.

```
<style>
span{
  display:block;
}
</style>
```



Note que agora foram inseridas quebra de linha antes e depois do ``.

Vamos implementar mais configurações com mais alguns detalhes do ``, vejamos a seguir:

```
<style>
  span{
    display:block;
    width: 80px;
    height:80px;
    border:#000 solid 1px;
    text-align:center;
    vertical-align: middle;
  }
</style>
```

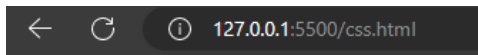


Observe com bastante atenção, verá que a propriedade “vertical-align:middle” não está funcionando, isso porque esta é uma propriedade para tabelas, então, para funcionar, vamos mudar o display do nosso elemento para table-cell.

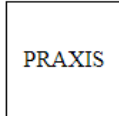
```
<style>
span{
  display:table-cell;
  width:50px;
  height:50px;
  border:#000 solid 1px;
  text-align:center;
  vertical-align:middle;
}
</style>
```

alteração do display:**table-cell**;

Veja o resultado com a



Curso Webdesigner

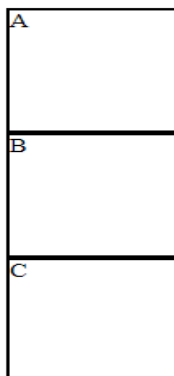
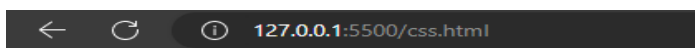


<https://praxisdesenvolvimento.com.br/>

Vamos ver outro exemplo usando divs.

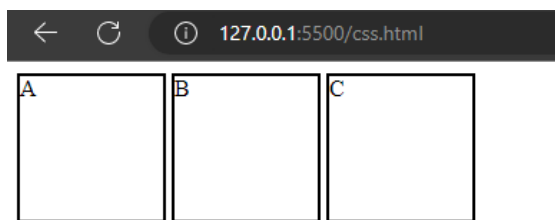
```
<style>
  div {
    width: 100px;
    height: 100px;
    border: #000 solid 2px;
  }
</style>
</head>

<body>
  <div id="divA">A</div>
  <div id="divB">B</div>
  <div id="divC">C</div>
</body>
```



Observe que no exemplo a cima o display padrão do elemento **<div>** é “**block**”, mas se precisarmos que todas estas divs sejam posicionadas uma ao lado da outra? simples, basta mudar o display para “**inline-block**”.

```
<style>
div{
width:100px;
height:100px;
border:#000 solid 1px;
display:inline-block;
}
</style>
```



POSITION

A propriedade position especifica como um elemento será posicionado na tela, podemos até posicionar em um ponto específico controlando ao parâmetros left e top.

Muitos programadores associam a propriedade position somente ao elemento **<div>** mas a verdade é que podemos utilizar em muitos outros elementos como **** e **<p>**.

São quatro tipos de posicionamento disponíveis, **static**, **relative**, **fixed** e **absolute**, vamos entende-los.

A única configuração que não permite escolher um posicionamento para o elemento é `static`, os demais podemos usar as propriedades `top`, `left`, `right` ou `bottom`.

top = Desloca o elemento na vertical (Y), o valor é a distância do elemento com o topo, ou seja, 0 pixels o elemento fica totalmente acima. Desloca no sentido de cima para baixo.

left = Desloca o elemento na horizontal (X), o valor é a distância do elemento com a borda esquerda, ou seja, 0 pixels o elemento fica totalmente à esquerda. Desloca no sentido esquerda para direita.

right = Desloca o elemento na horizontal (X), o valor é a distância do elemento com a borda direita, ou seja, 0 pixels o elemento fica totalmente à direita. Desloca no sentido direita para esquerda.

bottom = Desloca o elemento na vertical (Y), o valor é a distância do elemento com a borda inferior, ou seja, 0 pixels o elemento fica totalmente abaixo. Desloca no sentido de baixo para cima. Em nossos exemplos iremos usar `divs` simplesmente por facilitar o entendimento.

FORMATAÇÃO DE LINKS

Formatar os links é a mesma coisa de formatar um texto normal, a diferença é que podemos especificar uma formatação para cada estado do link. No primeiro código de exemplo vamos formatar somente a tag `<a>`, desta forma, todos os estados do link recebem a mesma formatação.

Observe o código a baixo:

```

<style>
  a {
    text-decoration: none;
    font-size: 20px;
    color: rgb(66, 9, 200);
  }
</style>
</head>

<body>
  <a href="https://praxisdesenvolvimento.com.br/" target="_blank">Curso
  Werdesigner</a>
</body>
</html>

```

← ↻ ⓘ 127.0.0.1:5500/link.html

Curso Werdesigner

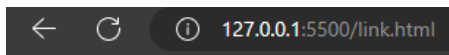
FORMATANDO O LINK COMO UM BOTÃO

Observe que podemos usar as propriedades do CSS para transformar o link na tag <a> normal como se fosse um botão.

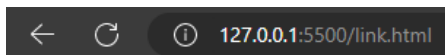
```

<style>
  a{
    text-decoration:none;
    font-size:20px;
    color:#FFF;
    background-color:rgb(14, 46, 142);
    padding: 15px 15px 15px 15px;
    display:inline-block;
  }
  a:hover{
    background-color:rgb(116, 129, 230);
  }
</style>

```



Curso Werdesigner



Curso Werdesigner

Quando o mouse estiver sobre o “botão” este ficará mais claro, exemplo da imagem ao lado.

MENU DROPDOWN SEM SCRIPT

Para construir usaremos alguns códigos que aprendemos até agora, faremos um menu estilo dropDown, sem nada de javascript, usando apenas propriedades CSS.

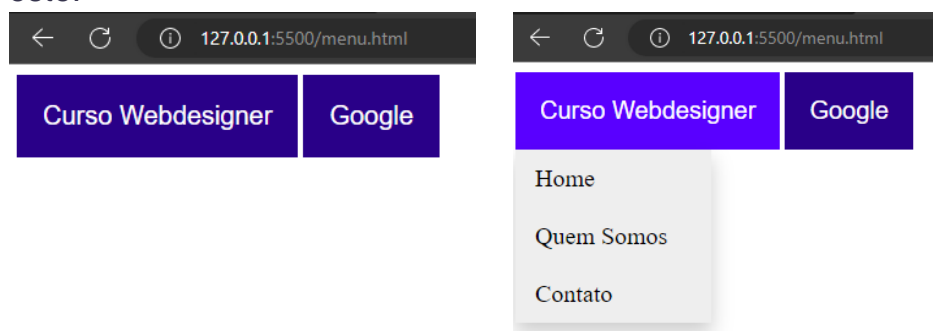
Observe com bastante atenção o código a baixo, pois usaremos o CSS incorporado ao nosso arquivo html.

```
<style>
  .menudd {
    position: relative;
    display: inline-block;
  }
  .menuBtn {
    background-color: #800;
    color: white;
    padding: 20px;
    font-size: 20px;
    border: none;
    cursor: pointer;
  }
  .menuConteudo {
    display: none;
    position: absolute;
    background-color: #EEE;
    min-width: 160px;
    box-shadow: 0px 8px 16px 0px rgba(0, 0, 0, 0.2);
  }
  .menuConteudo a {
    color: black;
    padding: 12px 16px;
    text-decoration: none;
    display: block;
    font-size: 20px;
  }
  .menuConteudo a:hover {
    background-color: #FDD;
    color: #800;
  }
  .menudd:hover .menuConteudo {
    display: block;
  }
  .menudd:hover .menuBtn {
    background-color: #F00;
  }
</style>
```

Observe que o código acima é referente ao CSS, ou seja, a estilização do nosso menu. A baixo veremos o código usado no `<body>` no corpo do nosso arquivo html.

```
<body>
  <div class="menudd">
    <button class="menuBtn">Curso Webdesigner</button>
    <div class="menuConteudo">
      <a href="https://praxisdesenvolvimento.com.br/"
target="_blank">Home</a>
      <a href="https://praxisdesenvolvimento.com.br/quemsomos.html"
target="_blank">Quem Somos</a>
      <a href="https://praxisdesenvolvimento.com.br/index.html#contatos"
target="_blank">Contato</a>
    </div>
  </div>
  <div class="menudd">
    <button class="menuBtn">Google</button>
    <div class="menuConteudo">
      <a href="http://www.google.com.br" target="_blank">Google</a>
      <a href="http://www.youtube.com.br" target="_blank">Youtube</a>
    </div>
  </div>
</body>
```

Reproduza o código a cima, o resultado deverá ser semelhante a este:



Observe a lista de dropdown no menu “Curso Webdesigner”.

BOX – SIZING

A propriedade box-sizing é muito interessante, ela permite incluir ao tamanho total de um elemento o padding e a borda.

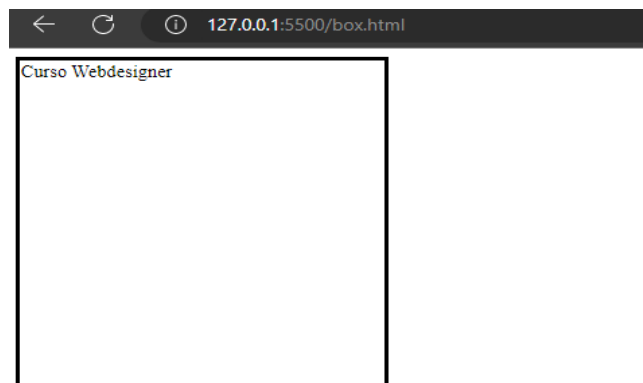
Isso resolver muitos problemas, na hora de adicionar um padding por exemplo o tamanho original do elemento não é alterado.

Vamos ao exemplo.

```
<style>
  div {
    border: #000 solid 4px;
    width: 300px;
    height: 300px;
  }
</style>
</head>

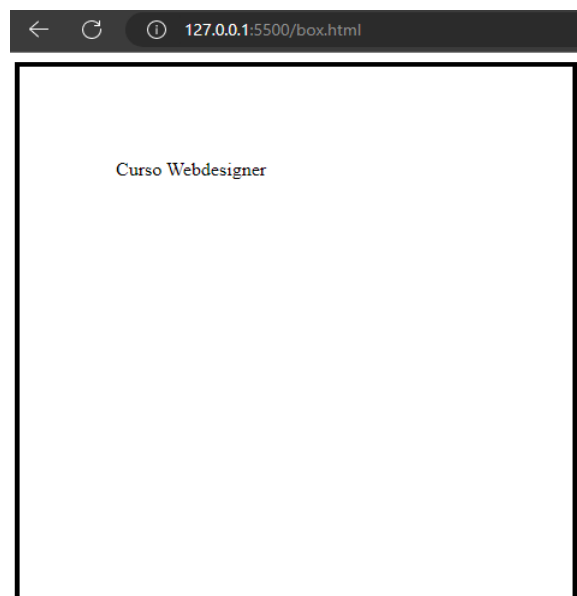
<body>
  <div>Curso Webdesigner</div>
</body>

</html>
```



Agora iremos adicionar 80 pixels de padding na div.

```
<style>
  div {
    border: #000 solid 4px;
    width: 300px;
    height: 300px;
    padding: 80px;
  }
</style>
```



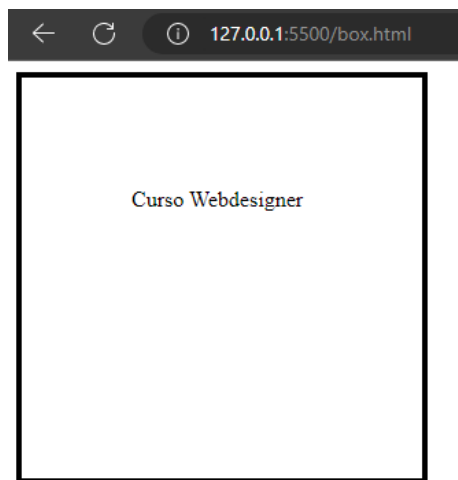
Observe a imagem a cima que o tamanho original da div foi alterado. Isso acontece porque o tamanho total de um elemento é calculado da seguinte forma.

$\text{width} + \text{padding} + \text{border} = \text{Largura atual do elemento.}$

$\text{height} + \text{padding} + \text{border} = \text{Altura atual do elemento.}$

Ao adiciona a propriedade box-sizing podemos mudar isto.

```
<style>
  div {
    border: #000 solid 4px;
    width: 300px;
    height: 300px;
    padding: 80px;
    box-sizing: border-box;
  }
</style>
```



Note que agora nem as bordas de 4 pixels nem os paddings de 80 pixels foram adicionado ao tamanho do elemento, neste caso, a div possui exatamente o tamanho de 300 pixels de largura por 300 pixels de altura originais.

Os valores possíveis são:

content-box (Padrão) O tamanho do elemento é calculado junto com o padding e o border.

border-box O tamanho do elemento é calculado pelo width e height, sem somar os valores de padding e border.

@MEDIA – LAYOUTS RESPONSIVOS

As regras **media** permitem configurar diferentes regras CSS para diferentes tipos de mídia, diante da necessidade de facilitar a configuração do site para se adaptar bem em dispositivos de diferentes tamanhos, daí vem o conceito de design responsivo ou layout responsivo.

Pode ter várias configurações de estilo para um mesmo site, para que se adapte aos diversos dispositivos, podemos ter estilos para computadores, celulares, impressão, televisão, tablets, etc.

As regras media queries podem verificar várias características dos dispositivos como, largura e altura da janela, largura e altura do dispositivo, orientação (se o dispositivo está na horizontal/ paisagem ou vertical/retrato), resolução.

A sintaxe é a seguinte:

```
@media not | only mediatype and (expressões) {  
Código CSS;  
}
```

O resultado de um comando @media é do tipo boolean (true/false), verdadeiro ou falso, se o tipo de dispositivo corresponde ao tipo de mídia terá um retorno true / verdadeiro e as regras CSS estabelecidas para esta @media serão aplicadas. Também podemos definir diferentes folhas de estilo para diferentes mídias.

```
<link rel="stylesheet" media="print()" href="impressora.css">  
<link rel="stylesheet" media="screen and (max-width: 500px)"  
href="celular.css">  
<link rel="stylesheet" media="screen and (min-width: 500px)" href="pc.css">  
<link rel="stylesheet" media="screen and (min-width: 1024px)"  
href="telasGrandes.css">
```


Os valores que podemos usar para os tipos de mídia são.

All - Todos os tipos de dispositivos

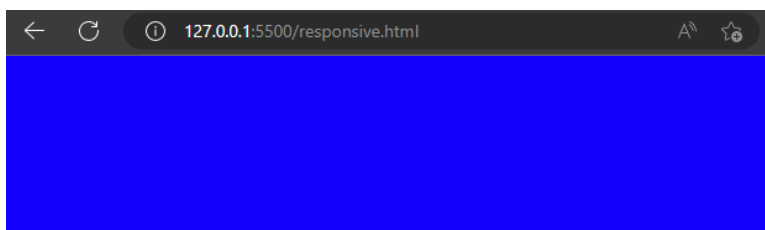
Print - Usado para impressoras

Screen - Usado para telas, computador, celulares, tablets, etc.

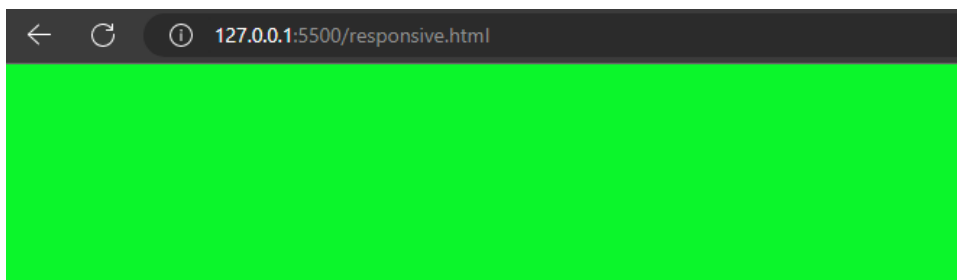
Speech - Usado para telas de leitores de conteúdo, leitores de ebook por exemplo, em voz alta.

Vamos a um primeiro código de exemplo onde iremos configurar a cor de fundo da tela para azul, mas iremos adicionar uma regra `@media` que irá muda a cor da tela para verde quando a largura for maior que 500 pixels, iremos definir a largura mínima para 500 pixels.

Tela com largura menor que 500 pixels.



Tela com largura maior que 500 pixels.



Vamos a um segundo exemplo, onde nosso layout com um menu e um texto se ajustará ao tamanho da tela, quando a janela tiver largura menor que 500 pixels se ajustará com o menu em cima do texto.

```

<style>
    .container {
        overflow: auto;
    }
    #txtPrincipal {
        margin-left: 214px;
    }
    #cxMenu {
        width: 200px;
        float: left;
    }
    #menu {
        margin: 0;
        padding: 0;
    }
    .menuitem {
        background: rgb(3, 43, 162);
        color: #FFF;
        border: #ddd solid 1px;
        border-radius: 4px;
        list-style-type: none;
        margin: 4px;
        padding: 2px;
    }
</style>
</head>

<body>
    <div class="container">
        <div id="cxMenu">
            <ul id="menu">
                <li class="menuitem">Menu 1</li>
                <li class="menuitem">Menu 2</li>
                <li class="menuitem">Menu 3</li>
            </ul>
        </div>
        <div id="txtPrincipal">
            <h1>Curso Webdesigner</h1>
            <p>Neste exemplo quando a tela for menor que 500 pixels, este texto que está ao lado
                do menu irá passar para baixo e o menu será redimensionado para ocupar o local onde o texto
                estava.</p>
        </div>
    </div>

```



Observe que o código gerou uma página sem a responsividade, independente da largura da janela, o layout sempre será o mesmo.

Agora vamos adicionar a rotina @media.

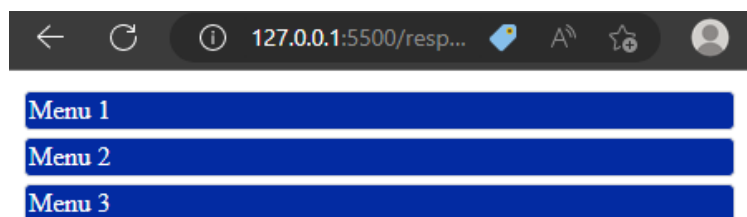
```
<style>
.container{overflow:auto;}
#txtPrincipal{margin-left:216px;}
#cxMenu{width:200px; float:left;}
#menu{
margin:0;
padding:0;
}
.menuitem{
background:#D55;
color:#FFF;
border:#ddd solid 1px;
border-radius:4px;
list-style-type:none;
margin:4px;
padding:2px;
}
@media screen and (max-width:500px){
#cxMenu{float:none; width:auto;}
#txtPrincipal{margin-left:5px;}
}
```

No código a cima inserimos a instrução `@media` indicamos:

`@media screen and (max-width:500px){` quando a largura máxima da tela for 500 pixels, ou seja, enquanto a tela for menor que 500 pixels, os comandos a seguir estarão atuando.

`#cxMenu{float:none; width:auto;}` Formata o menu sem flutuar e com a largura em auto para se ajustar ao tamanho atual da janela.
`#txtPrincipal{margin-left:5px;}` O texto principal recebe uma margem esquerda de 5 pixels para se distanciar um pouco da borda esquerda da janela.

Com esta nova implementação de `@media`, quando a janela estiver menor que 500 pixels se apresentará como na ilustração a seguir:



Curso Webdesigner

Neste exemplo quando a tela for menor que 500 pixels, este texto que está ao lado do menu irá passar para baixo e o menu será redimensionado para ocupar o local onde o texto estava.

FLEXBOX

O que é o Flexbox

Por muito tempo, as únicas ferramentas disponíveis para criar layouts em CSS e posicionar elementos com boa compatibilidade entre browsers eram float e position. Porém, essas ferramentas possuem algumas limitações muito

frustrantes, especialmente no que diz respeito à responsividade. Algumas tarefas que consideramos básicas em um leiaute, como centralização vertical de um elemento-filho com relação a um elemento-pai ou fazer com que elementos-filhos ocupem a mesma quantidade de espaço, ou colunas terem o mesmo tamanho independentemente da quantidade de conteúdo interno, eram impossíveis ou muito difíceis de serem manejadas com floats ou position, ao menos de forma prática e flexível.

A ferramenta Flexbox (de Flexible Box) foi criada para tornar essas tarefas mais simples e funcionais: os filhos de um elemento com Flexbox podem se posicionar em qualquer direção e pode ter dimensões flexíveis para se adaptar.

ELEMENTOS

O Flexbox é um módulo completo e não uma única propriedade; algumas delas devem ser declaradas no container (o elemento-pai, que chamamos de flex container), enquanto outras devem ser declaradas nos elementos-filhos (os flex itens). Se o leiaute "padrão" é baseado nas direções block e inline, o leiaute Flex é baseado em direções "flex flow". Veja abaixo um

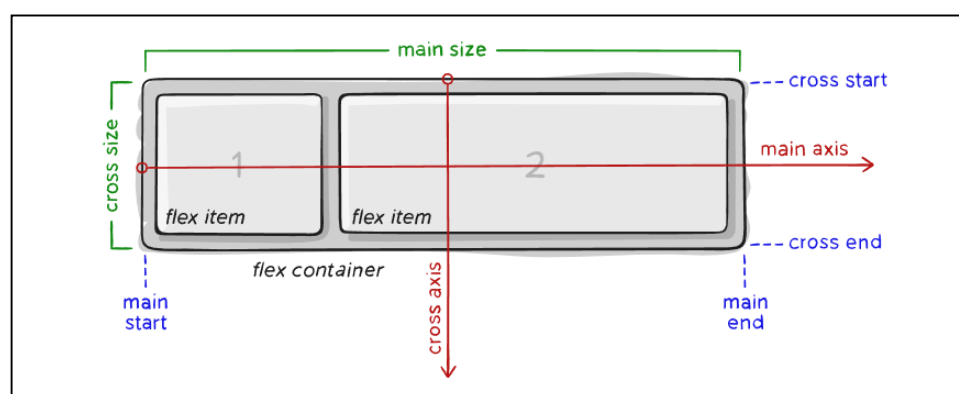


diagrama da especificação, explicando a ideia central por trás do layout Flex.

Os itens serão dispostos no layout seguindo ou o eixo principal ou o transversal.

Eixo principal: o eixo principal de um flex container é o eixo primário e ao longo dele são inseridos os flex items. Cuidado: O eixo principal não é necessariamente horizontal; vai depender da propriedade flex-direction (veja abaixo).

main-start | main-end: os flex items são inseridos dentro do container começando pelo lado start, indo em direção ao lado end.

Tamanho principal: A largura ou altura de um flex item, dependendo da direção do container, é o tamanho principal do item. A propriedade de tamanho principal de um flex item pode ser tanto width quanto height, dependendo de qual delas estiver na direção principal.

Eixo transversal: O eixo perpendicular ao eixo principal é chamado de eixo transversal. Sua direção depende da direção do eixo principal.

cross-start | cross-end: Linhas flex são preenchidas com itens e adicionadas ao container, começando pelo lado cross start do flex container em direção ao lado cross end.

cross size: A largura ou altura de um flex item, dependendo do que estiver na dimensão transversal, é o cross size do item. A propriedade cross size pode ser tanto a largura quanto a altura do item, o que estiver na transversal.

Flex container é o elemento que envolve sua estrutura. Você define que um elemento é um Flex Container com a propriedade `display` e valores `flex` ou `inline-flex`.

```
<div class="flex-container">
  <div>1</div>
  <div>2</div>
  <div>3</div>
</div>
```

```
.flex-container {
  display: flex;
}
```

Flex Item são elementos-filhos do flex container.

Eixos ou Axes são as duas direções básicas que existem em um Flex Container: `main axis`, ou eixo principal, e `cross axis`, ou eixo transversal.

PROPRIEDADES PARA O ELEMENTO-PAI

Container



Quando utilizamos o Flexbox, é muito importante saber quais propriedades são declaradas no elemento-pai (por exemplo, uma `div` que irá conter os elementos a serem alinhados) e quais serão declaradas nos elementos-filhos. Abaixo, seguem propriedades

que devem ser declaradas utilizando o elemento-pai como seletor (para alinhar elementos-filhos):

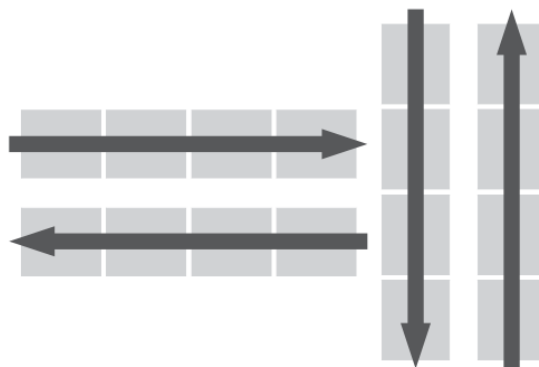
DISPLAY

Esta propriedade define um flex container; inline ou block dependendo dos valores passados. Coloca todos os elementos-filhos diretos num contexto Flex.

```
.container {  
  display: flex; /* or inline-flex */  
}
```

Note que a propriedade de CSS **columns** não tem efeito em um flex container.

FLEX-DIRECTION



Estabelece o eixo principal, definindo assim a direção em que os flex items são alinhados no flex container. O Flexbox é (com exceção de um wrapping opcional) um conceito de layout de uma só direção. Pense nos flex items inicialmente posicionados ou em linhas horizontais ou em colunas verticais.

.flex-container {


```
flex-direction: row | row-reverse | column | column-reverse;  
}
```

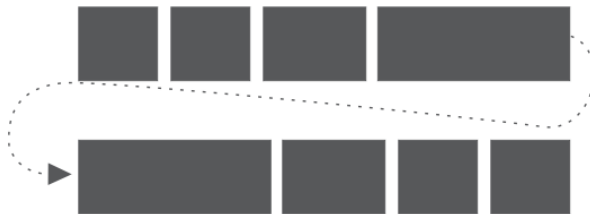
row (padrão): esquerda para a direita em ltr (left to right), direita para a esquerda em rtl (right to left)

row-reverse: direita para a esquerda em ltr, esquerda para a direita em rtl

column: mesmo que row, mas de cima para baixo

column-reverse: mesmo que row-reverse mas de baixo para cima

FLEX-WRAP



Por padrão, os flex items vão todos tentar se encaixar em uma só linha. Com esta propriedade você pode modificar esse comportamento e permitir que os itens quebrem para uma linha seguinte conforme for necessário.

```
.flex-container {  
  flex-wrap: nowrap | wrap | wrap-reverse;  
}
```

nowrap (padrão): todos os flex items ficarão em uma só linha

wrap: os flex items vão quebrar em múltiplas linhas, de cima para baixo.

wrap-reverse: os flex items vão quebrar em múltiplas linhas de baixo para cima

FLEX-FLOW

A propriedade flex-flow é uma propriedade shorthand (uma mesma declaração inclui vários valores relacionados a mais de uma propriedade) que inclui flex-direction e flex-wrap. Determina quais serão os eixos principal e transversal do container. O valor padrão é row nowrap.

```
.flex-container {  
  flex-flow: row nowrap | row wrap | column nowrap | column  
  wrap;  
}
```

JUSTIFY-CONTENT

flex-start



flex-end



center



space-between



space-around



space-evenly



Esta propriedade define o alinhamento dos itens ao longo do eixo principal. Ajuda a distribuir o espaço livre que sobrar no container tanto se todos os flex items em uma linha são inflexíveis, ou são flexíveis mas já atingiram seu tamanho máximo. Também exerce algum controle sobre o alinhamento de itens quando eles ultrapassam o limite da linha.

```
.flex-container {  
  justify-content: flex-start | flex-end | center | space-between | space-around | space-evenly;  
}
```

flex-start (padrão): os itens são alinhados junto à borda de início (start) de acordo com qual for a flex-direction do container.

flex-end: os itens são alinhados junto à borda final (end) de acordo com qual for a flex-direction do container.

start: os itens são alinhados junto à borda de início da direção do **writing-mode** (modo de escrita).

end: os itens são alinhados junto à borda final da direção do writing-mode (modo de escrita).

left: os itens são alinhados junto à borda esquerda do container, a não ser que isso não faça sentido com o flex-direction que estiver sendo utilizado. Nesse caso, se comporta como start.

right: os itens são alinhados junto à borda direita do container, a não ser que isso não faça sentido com o flex-direction que estiver sendo utilizado. Nesse caso, se comporta como start.

center: os itens são centralizados na linha.

space-between: os itens são distribuídos de forma igual ao longo da linha; o primeiro item junto à borda inicial da linha, o último junto à borda final da linha.

space-around: os itens são distribuídos na linha com o mesmo espaçamento entre eles. Note que, visualmente, o espaço pode não ser igual, uma vez que todos os itens tem a mesma

quantidade de espaço dos dois lados: o primeiro item vai ter somente uma unidade de espaço junto à borda do container, mas duas unidades de espaço entre ele e o próximo ítem, pois o próximo ítem também tem seu próprio espaçamento que está sendo aplicado.

space-evenly: os ítems são distribuídos de forma que o espaçamento entre quaisquer dois itens da linha (incluindo entre os ítems e as bordas) seja igual.

ALIGN-ITEMS

flex-start



flex-end



center



stretch



baseline



Define o comportamento padrão de como flex items são alinhados de acordo com o eixo transversal (cross axis). De certa forma, funciona de forma similar ao justify-content, porém no eixo transversal (perpendicular ao eixo principal).

```
flex-container {  
  align-items: stretch | flex-start | flex-end | center | baseline;  
}
```

stretch (padrão): estica os itens para preencher o container, respeitando o min-width / max-width).

flex-start/ start / self-start: itens são posicionados no início do eixo transversal. A diferença entre eles é sutil e diz respeito às regras de **flex-direction** ou **writing-mode**. **center**: itens são centralizados no eixo transversal.

baseline: itens são alinhados de acordo com suas baselines.

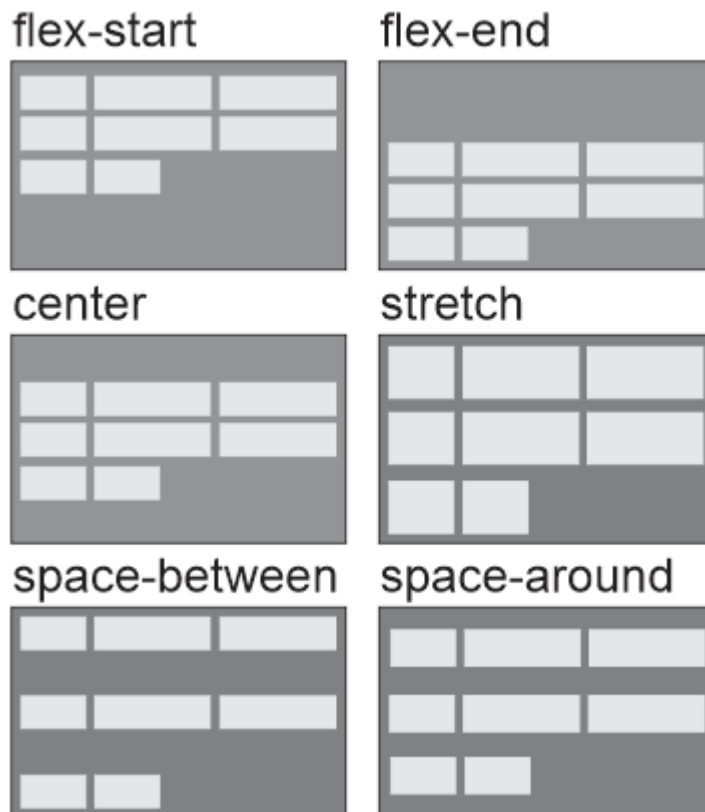
Os modificadores safe e unsafe podem ser usados em conjunto com todas essas palavras-chave (favor conferir o suporte de cada navegador) e servem para prevenir qualquer alinhamento de elementos que faça com que o conteúdo fique inacessível (por exemplo, para fora da tela).

ALIGN-CONTENT

Organiza as linhas dentro de um flex container quando há espaço extra no eixo transversal, similar ao modo como justify-content alinha itens individuais dentro do eixo principal.

Importante: Esta propriedade não tem efeito quando há somente uma linha de flex items no container.

```
.flex-container {  
  align-content: flex-start | flex-end | center | space-between  
| space-around | stretch;  
}
```



flex-start / start: itens alinhados com o início do container. O valor (com maior suporte dos navegadores) **flex-start** se guia pela **flex-direction**, enquanto **start** se guia pela direção do **writing-mode**.

flex-end / end: itens alinhados com o final do container. O valor (com maior suporte dos navegadores) **flex-end** se guia pela **flex-direction**, enquanto **end** se guia pela direção do **writing-mode**.

center: itens centralizados no container.

space-between: itens distribuídos igualmente; a primeira linha junto ao início do container e a última linha junto ao final do container.

space-around: itens distribuídos igualmente com o mesmo espaçamento entre cada linha.

space-evenly: itens distribuídos igualmente com o mesmo espaçamento entre eles.

stretch (padrão): itens em cada linha esticam para ocupar o espaço remanescente entre elas.

PROPRIEDADES PARA ELEMENTOS-FILHOS

A seguir, veremos propriedades que devem ser declaradas tendo como seletor os elementos-filhos, ou seja:

```
<div class="flex-container">
  <div class="flex-item">1</div>
  <div class="flex-item">2</div>
  <div class="flex-item">3</div>
</div>
```

Isso significa que, onde existe um elemento-pai com propriedade flex (o flex-container), é possível atribuir propriedades flex específicas também para as elementos-filhos (flex-item).

Você pode definir as propriedades abaixo para apenas um dos elementos-filhos através de um identificador, como uma classe específica.

ORDER

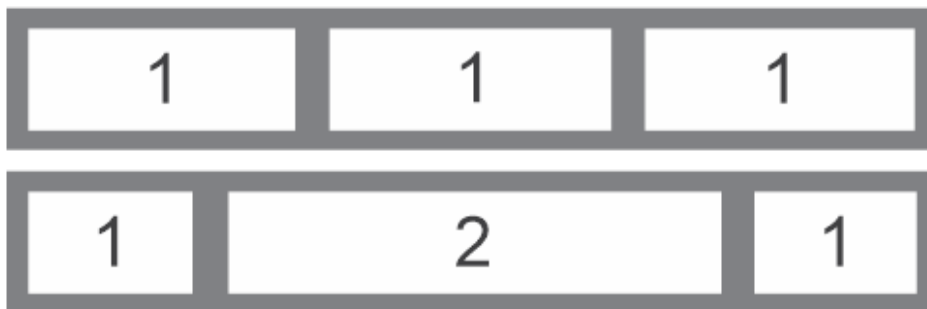
Determina a ordem em que os elementos aparecerão.

1	1	2	3	4
-1	1	2	7	
1				
2				
10				

Por padrão os flex-items são dispostos na tela na ordem do código. Mas a propriedade `order` controla a ordem em que aparecerão no container.

```
.flex-item {  
  order: <número>; /* o valor padrão é 0 */  
}
```

FLEX-GROW



Define a habilidade de um flex item de crescer, caso necessário. O valor dessa propriedade é um valor numérico sem indicação de unidade, que serve para cálculo de proporção. Este valor dita a quantidade de espaço disponível no container que será ocupado pelo item. Se todos os itens tiverem `flex-grow` definido em 1, o espaço remanescente no container será distribuído de forma igual entre todos. Se um dos itens tem o valor de 2, vai ocupar o dobro de espaço no container com relação aos outros (ou pelo menos vai tentar fazer isso).

```
.flex-item {  
  flex-grow: <numero>; /* o valor default(padão) é 0 */  
}
```

Valores negativos não são aceitos pela propriedade.

FLEX-SHRINK

Define a habilidade de um flex item de encolher, caso necessário.

```
.flex-item {  
  flex-shrink: <número>; /* o valor padrão é 0 */  
}
```

Valores negativos não são aceitos pela propriedade.

FLEX-BASIS

Define o tamanho padrão para um elemento antes que o espaço remanescente do container seja distribuído. Pode ser um comprimento (por exemplo, 20%, 5rem, etc) ou uma palavra-chave. A palavra-chave auto significa "observe minhas propriedades de altura ou largura" (o que era feito pela palavra-chave main-size, que foi depreciada). A palavra-chave content significa "estabeleça o tamanho com base no conteúdo interno do ítem" - essa palavra-chave ainda não tem muito suporte, então não é fácil de ser testada, assim como suas relacionadas: max-content, min-content e fit-content.

```
.flex-item {  
  flex-basis: flex-basis: | auto; /* o valor padrão é auto */  
}
```

Com o valor de 0, o espaço extra ao redor do conteúdo não é considerado. Com o valor de auto, o espaço extra é distribuído com base no valor de flex-grow do ítem.

FLEX

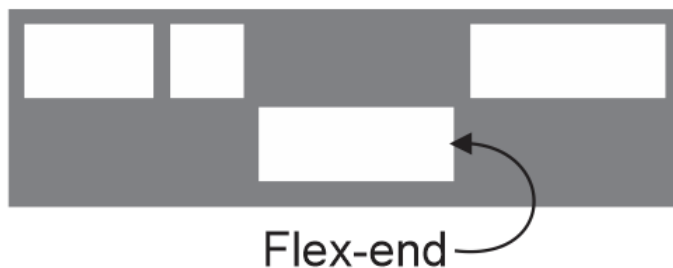
Esta é a propriedade shorthand para flex-grow, flex-shrink e flex-basis, combinadas. O segundo e terceiro parâmetros (flex-

shrink e flex-basis) são opcionais. O padrão é 0 1 auto, mas se você definir com apenas um número, é equivalente a 0 1.

```
.item {  
  flex: none | [ <'flex-grow'> <'flex-shrink'>? || <'flex-basis'>  
]  
}
```

É recomendado que você utilize esta propriedade shorthand ao invés de definir cada uma das propriedades em separado. O shorthand define os outros valores de forma inteligente.

ALIGN-SELF



Permite que o alinhamento padrão (ou o que estiver definido por **align-items**) seja sobrescrito para itens individuais.

Por favor veja a explicação da propriedade **align-items** para entender quais são os possíveis valores.

```
.item {  
  align-self: auto | flex-start | flex-end | center | baseline |  
stretch;  
}
```

Importante!

O CSS só enxerga a hierarquia pai-filho; não vai aplicar as propriedades Flex para elementos que não estejam diretamente relacionados;

Para que as propriedades funcionem nos elementos-filhos, as pais devem ter propriedade `display: flex`; As propriedades `float`, `clear` e `vertical-align` não têm efeito em `flex-items`.

PROJETO

A primeira coisa a se fazer é criar a estrutura de pastas do nosso site, primeiramente crie uma pasta de **nome projeto** e logo após dentro dela crie mais duas pastas, uma para imagem de nome **img** e outra para o arquivo **css**, chamada de `css`. Após isso crie um arquivo `index.html` que será responsável por ser a página principal do nosso site.

Nome	Data de modificação	Tipo	Tamanho
css	25/11/2022 17:03	Pasta de arquivos	
img	25/11/2022 17:03	Pasta de arquivos	
index.html	25/11/2022 17:04	Chrome HTML Do...	0 KB

Logo após montar a estrutura, faça o download de uma imagem no Google para servir de logo para seu site, e coloque a tal imagem dentro da pasta `img`.

Feito isso, vamos codificar o arquivo `index.html`.

Primeiramente vamos montar a estrutura básica do site.

```
<!DOCTYPE html>
<html lang="pt-br">
<head>
<meta charset="utf-8" />
<link rel="stylesheet" href="css/estilo.css"/>
<title> Blog Empregos DF</title> </head>
<body>
```

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

Repare que já foi incluído dentro da sessão head o elemento link, pois ele é o responsável por ligar o nosso arquivo HTML ao arquivo CSS.

Logo após isso vamos criar a estrutura no nosso menu, dentro da sessão body, digite o seguinte código:

```
<hr />
<header>
<figure>

</figure>
<nav id="menu_principal">
<ul>
<li><a href="index.html">Home</a></li>
<li><a href="#concursos">Concursos</a></li>
<li><a href="#estagio">Estágio</a></li>
<li><a href="#anunciodevagas">Anuncio de vagas</a></li>
<li><a href="#duvidas">Dúvidas</a></li>
</ul>
</nav>
</header>
```

Lembre-se que o elemento hr insere uma linha horizontal para fins decorativos na página, logo após foi usado o elemento header que é um elemento semântico do HTML que indica ao navegador que essa parte do código se refere a um cabeçalho.

Feito isso foi adicionado o logo, e logo abaixo dentro do elemento nav que serve principalmente para indicar que estamos em um menu de navegação, foi aberta uma lista () onde cada item () possui um link.

Agora vamos trabalhar no corpo do site, primeiramente vamos utilizar um elemento div para que na hora de estilizar o corpo do site a tarefa seja simplificada.

Dentro dessa div, vamos utilizar o elemento main, para indicar que essa é a parte principal do site. Logo após usamos o elemento section para separar semanticamente os conteúdos de cada site, e dentro do section usamos o elemento article para assim escrevermos o nosso texto. Insira os códigos abaixo do elemento header:

```
<div id="principal">
<main>
<h1>Últimas Vagas. </h1>
<section>
<div class="vagas">
<article class="vaga">
<a href="#"><h1>Ipsum Lorem.</h1></a>
<p>Morbi a metus. Phasellus enim erat, vestibulum vel,
aliquam a, posuere eu, velit. Nullam sapien sem, ornare ac,
nonummy non, lobortis a, enim.
Nunc tincidunt ante vitae massa. Duis ante orci, molestie
vitae, vehicula venenatis, tincidunt ac, pede. Nulla
accumsan, elit sit amet varius semper, nulla mauris mollis
quam, tempor suscipit diam nulla vel leo. Etiam commodo dui
eget wisi. Donec iaculis gravida nulla. Donec quis nibh at
felis congue commodo. <strong>Etiam</
strong> bibendum elit eget erat.</p>
<hr />
```

```
</article>
<article class="vaga">
<a href="#"><h1> Ipsum Lorem.</h1></a>
<p>Morbi a metus. Phasellus enim erat, vestibulum vel,
aliquam a, posuere eu, velit. Nullam sapien sem, ornare ac,
nonummy non, lobortis a, enim. Nunc tincidunt ante vitae
massa. Duis ante orci, molestie vitae, vehicula venenatis,
tincidunt ac, pede. Nulla accumsan, elit sit amet varius
semper, nulla mauris mollis
quam, tempor suscipit diam nulla vel leo. Etiam commodo dui
eget wisi. Donec
iaculis gravida nulla. Donec quis nibh at felis congue
commodo. <strong>Etiam</
strong> bibendum elit eget erat.</p>
<hr />
```

```
</article>
<article class="vaga">
<a href="#"><h1> Ipsum Lorem.</h1></a>
<p>Morbi a metus. Phasellus enim erat, vestibulum vel,
aliquam a, posuere eu, velit. Nullam sapien sem, ornare ac,
nonummy non, lobortis a, enim. Nunc tincidunt ante vitae
massa. Duis ante orci, molestie vitae, vehicula venenatis,
tincidunt ac, pede. Nulla accumsan, elit sit amet varius
semper, nulla mauris mollis quam, tempor suscipit diam
nulla vel leo. Etiam commodo dui eget wisi. Donec iaculis
gravida nulla. Donec quis nibh at felis congue commodo.
<strong>Etiam</strong> bibendum elit eget erat.</p>
</article> </div>
</section>
</main>
```

Logo após vamos criar uma divisão lateral com o elemento aside para ser possível que o usuário realize uma pesquisa e algumas publicidades no site. Insira o código a seguir a baixo de main:

```
<aside>
<section class="procura">
<h1> Pesquise sua vaga.</h1>
<form>
<input type="text" >
<input type="submit" value="Enviar">
</form>
<hr/>
</section>
<section>
<article class="publicidade">
<h1> Publicidade.</h1>
<p>Morbi a metus. Phasellus enim erat, vestibulum vel,
aliquam a, posuere eu, velit. Nullam sapien sem, ornare ac,
nonummy non, lobortis a, enim.
Nunc tincidunt ante vitae massa. Duis ante orci, molestie
vitae, vehicula venenatis, tincidunt ac, pede. Nulla
accumsan, elit sit amet varius semper, nulla mauris mollis
quam, tempor suscipit diam nulla vel leo. Etiam commodo dui
eget wisi. Donec iaculis gravida nulla. Donec quis nibh at
felis congue commodo. <strong>Etiam</strong> bibendum
elit eget erat.</p>
<hr/>
</article>
<aside>
</section>
</div>
```

Será necessário observar e replicar o código descrito a baixo pois o mesmo completo ficará assim:

```
<!DOCTYPE html>
<html lang="pt-br">
```

```
<head>
```

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

```
  <meta charset="utf-8" />
```

```
  <meta name="description" content="vagas de emprego em
Brasília" />
```

```
  <meta name="keywords" content=" vagas em brasilia,
concurso df, emprego, emprego brasilia, vagas para emprego" />
```

```
  <link rel="icon" type="image/png" href="/img/ficon.png" />
```

```
  <link rel="stylesheet" href="o nome do arquivo que você
criou .css" />
```

```
  <title> Blog Empregos DF</title>
```

```
</head>
```

```
<body>
```

```
  <hr />
```

```
  <header>
```

```
    <figure>
```

```
      
```

```
    </figure>
```

```
    <nav id="menu_principal">
```

```
      <ul>
```

```
        <li><a href="index.html">Home</a></li>
```

```
        <li><a href="#concursos">Concursos</a></li>
```

```
        <li><a href="#estagio">Estágio</a></li>
```



```
        <li><a href="#anuncio de vagas">Anuncio de
vagas</a></li>
```

```
        <li><a href="#duvidas">Dúvidas</a></li>
```

```
    </ul>
```

```
</nav>
```

```
</header>
```

```
<div id="principal">
```

```
    <h1>Últimas Vagas. </h1>
```

```
    <section>
```

```
        <div class="vagas">
```

```
            <article class="vaga">
```

```
                <a href="#">
```

```
                    <h1>Ipsum Lorem.</h1>
```

```
                </a>
```

```
                <p>Morbi a metus. Phasellus enim erat, vestibulum
vel, aliquam a, posuere eu, velit. Nullam sapien sem, ornare ac,
nonummy non, lobortis a, enim. Nunc tincidunt ante vitae massa.
Duis ante orci, molestie vitae, vehicula venenatis,
```

```
                tincidunt ac, pede. Nulla accumsan, elit sit amet
varius semper, nulla mauris mollisquam, tempor suscipit diam
nulla vel leo. Etiam commodo dui eget wisi. Donec
iaculis gravida nulla. Donec quis nibh at felis congue commodo.
```

```
                <strong>Etiam</strong> bibendum elit eget erat.</p>
```

```
            <hr />
```

```
        </article>
```

```
        <article class="vaga">
```

```
            <a href="#">
```

```
                <h1>Ipsum Lorem.</h1>
```

```
            </a>
```

```
            <p>Morbi a metus. Phasellus enim erat, vestibulum vel,
aliquam a, posuere eu, velit. Nullam sapien sem, ornare ac,
```

nonummy non, lobortis a, enim. Nunc tincidunt ante vitae massa. Duis ante orci, molestie vitae, vehicula venenatis, tincidunt ac, pede. Nulla accumsan, elit sit amet varius semper, nulla mauris mollis quam, tempor suscipit diam nulla vel leo. Etiam commodo dui eget wisi. Donec iaculis gravida nulla. Donec quis nibh at felis congue commodo. Etiam bibendum elit eget erat.</p>

<hr />

</article>

<article class="vaga">

<h1> Ipsum Lorem.</h1>

<p>Morbi a metus. Phasellus enim erat, vestibulum vel, aliquam a, posuere eu, velit. Nullam sapien sem, ornare ac, nonummy non, lobortis a, enim.

Nunc tincidunt ante vitae massa. Duis ante orci, molestie vitae, vehicula venenatis, incidunt ac, pede. Nulla accumsan, elit sit amet varius semper, nulla mauris mollis quam, tempor suscipit diam nulla vel leo. Etiam commodo dui eget wisi. Donec iaculis gravida nulla. Donec quis nibh at felis congue commodo. Etiam bibendum elit eget erat.</p>

</article>

</div>

</section>

<aside>

<section class="procura">

<h1> Pesquisa sua vaga.</h1>

<form>

<input type="text">

<input type="submit" value="Enviar">

```

</form>
<hr />
</section>
<section>
  <article class="public">
    <h1> Publicidade.</h1>

    <p>Morbi a metus. Phasellus enim erat, vestibulum
    vel, aliquam a, posuere eu, velit. Nullam sapien sem, ornare ac,
    nonummy non, lobortis a, enim.

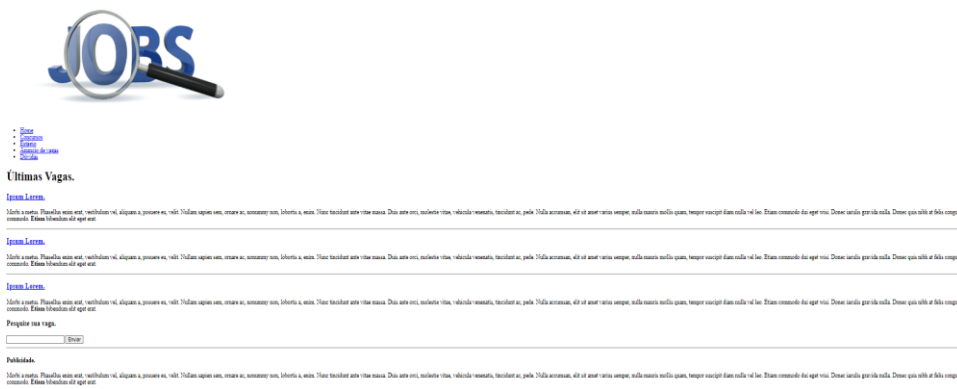
    Nunc tincidunt ante vitae massa. Duis ante orci,
    molestie vitae, vehicula venenatis, incidunt ac, pede. Nulla
    accumsan, elit sit amet varius semper, nulla mauris mollis
    quam, tempor suscipit diam nulla vel leo. Etiam
    commodo dui eget wisi. Donec iaculis gravida nulla. Donec quis
    nibh at felis congue commodo. <strong>Etiam</strong>
    bibendum elit eget erat.</p>

    <hr/>
  </article>
</section>
</aside>
</div>
</body>

</html>

```

Lembre-se de colocar as classes e os ids dos elementos, eles são importantes para o CSS identificá-los como seletores. Observe e veja se a sua página está com esse visual:



Agora vamos começar a estilizar a página, abra o arquivo `estilo.css` e repita o código a seguir:

```
@charset "utf-8";

@import
url("https://fonts.googleapis.com/css?family=Source+Sans+Pro")
;

* {
    color: #424242;
    margin: 0;
    padding: 0;
    font-size: 100%;
    box-sizing: border-box;
    font-family: "Source Sans Pro", sans-serif;
    outline: none;
    border: none
}

body {
    background-color: #ebf1f5;
}
```

```
a {
  text-decoration: none;
}

ul {
  list-style: none;
}

img {
  max-width: 100%;
}

hr {
  width: 80%;
  height: 0.3em;
  background-color: #d1e0e9;
  margin: auto;
}

header {
  width: 80%;
  background-color: #fff;
  margin: auto;
  height: 100%;
  overflow: hidden;
  box-shadow: 0.08em 0.08em 0.08em 0.08em rgba(224, 224,
224, 0.5);
  border-radius: 0.3em;
}

h1 {
  width: 100%;
```

```

    font-weight: 800;
    font-size: 170%;
    padding: 0% 2.5%;
    background-color: #d1e0e9;
    border-radius: 0.2em;
}

#logo {
    float: left;
    width: 20%;
    margin: 2% 5%;
}

#menu_principal {
    float: right;
    width: 70%;
    padding: 5.5% 0% 0% 10%;
}

#menu_principal li {
    float: left;
}

#menu_principal a {
    padding: 0.3em 0.9em;
    font-size: 130%;
    color: #337AB7;
    font-weight: 600;
}

#menu_principal a:hover {
    transition: 0.15s;

```

```

    font-weight: 600;
    border-bottom: 0.15em solid #337AB7;
}

#principal {
    width: 80%;
    background-color: #FFF;
    margin: 0.5% auto;
    border-radius: 0.3em;
    height: 100%;
    overflow: hidden;
    -webkit-box-shadow: 0.05em 0.05em 0.05em 0.05em
    rgba(224, 224, 224, 0.5);
    -moz-box-shadow: 0.0em 0.05em 0.05em 0.05em rgba(224,
    224, 224, 0.5);
    box-shadow: 0.05em 0.05em 0.05em 0.05em rgba(224, 224,
    224, 0.5);
}

.vagas {
    width: 71%;
    margin: 2% 1%;
    border: 0.1em solid #E0E0E0;
    border-radius: 0.5em;
    -webkit-box-shadow: 0.05em 0.05em 0.05em 0.05em
    rgba(224, 224, 224, 0.5);
    -moz-box-shadow: 0.0em 0.05em 0.05em 0.05em rgba(224,
    224, 224, 0.5);
    box-shadow: 0.08em 0.08em 0.08em 0.08em rgba(224, 224,
    224, 0.5);
    float: left;
}

```

```
.vaga h1 {  
    width: 50%;  
    font-weight: 600;  
    font-size: 130%;  
    background-color: #ffffff;  
    padding: 2% 0% 0% 4.5%;  
    color: #3BA1C6;  
}
```

```
.vaga p {  
    width: 95%;  
    text-align: justify;  
    text-indent: 4em;  
    margin: 2% auto;  
}
```

```
.vaga h1:hover {  
    color: #424242;  
}
```

```
.vaga hr {  
    width: 95%;  
    height: 0.12em;  
    background-color: #e0e0e0;  
    margin:  
        auto;  
}
```

```
.procura {  
    width: 20%;  
    margin: 2% 1%;  
    position: absolute;
```



```

    left: 68%;
}

.procura h1 {
    width: 100%;
    font-weight: 600;
    font-size: 130%;
    background-color:
        #ffffff;
    padding: 2% 0% 0% 4.5%;
}

.procura form {
    padding: 2% 5%;
}

.procura hr {
    width: 100%;
    height: 0.12em;
    margin-top: 5%;
    background-color:
        #e0e0e0;
}

.procura form input[type=text] {
    width: 74%;
    background-image: url();
    background-position: 0.4em 0.2em;
    background-repeat: no-repeat;
    padding-left: 1.7em;
    background-color: #ebf1f5;
    border-radius: 0.1em;
}

```

```
-webkit-box-shadow: 0.1em 0.1em 0.1em 0.1em rgba(224,
224, 224, 0.5);
-moz-box-shadow: 0.1em 0.1em 0.1em 0.1em rgba(224, 224,
224, 0.5);
box-shadow: 0.1em 0.1em 0.1em 0.1em rgba(224, 224, 224,
0.5);
}
```

```
.procura form input[type=text]:focus {
  background-color: #fff;
  border: 0.1em solid #d1e0e9;
}
```

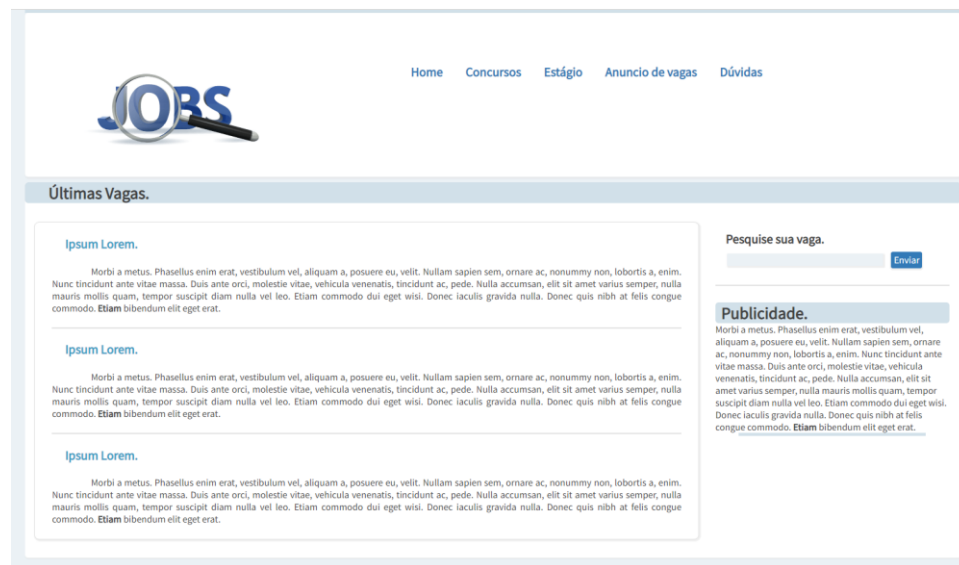
```
.procura form input[type=submit] {
  padding: 1% 1.27%;
  margin-left: 3%;
  background-color: #337AB7;
  color: #fff;
  font-style: bold;
  border-radius: 0.2em;
  -webkit-box-shadow: 0.1em rgba(224, 224, 224, 0.5);
  -moz-box-shadow: 0.1em rgba(224, 224, 224, 0.5);
  box-shadow: 0em 0.1em 0em 0.1em rgba(224, 224, 224, 0.5);
}
```

```
.procura form input[type=submit]:hover {
  background-color: #286090;
  cursor: pointer;
}
```

```
.public {
  width: 20%;
}
```

```
margin: 8.5% 1%;
position: absolute;
left: 68%;
}
```

O resultado final será esse:



EXERCÍCIOS

1. Primeiro crie uma pasta de nome projeto1. Logo após criar essa pasta crie mais duas pastas dentro da pasta projeto1. Uma pasta será para o arquivo css logo ela se chamara css, e outra para as imagens logo ela se chamara img.

Depois de criar a estrutura de seu projeto vá para a pasta de imagem e baixe uma imagem a seu gosto, mas lembre-se de ajustar o nome dela no código. Essa imagem servirá como logo. Quando terminar com as imagens crie um arquivo index.

html e implemente o seguinte código:

```
<!doctype html>
<html>
<head>
<meta charset="utf-8">
<title>Untitled Document</title>
<link href="css/estilo.css" rel="stylesheet" type="text/css">
</head>
<body>
<header id="cabecalho">
</img>
<ul id="menu">
<li><a href="#">Principal</a> </li>
<li><a href="#">Tecnologia</a></li>
<li><a href="#">Séries</a></li>
<li><a href="#">Filmes</a></li>
<li><a href="#">Games</a></li>
<li><a href="#">Curiosidades</a></li>
<li></li>
</ul>
</header>
<main id="principal">
</main>
</body>
</html>
```

Após codificar a nossa página index, vá para a pasta css e crie lá um arquivo chamado estilo.css, feito isso implemente esses códigos no arquivo.

```
@charset "utf-8";
/* CSS Document */
```

```

@import
url('https://fonts.googleapis.com/css?family=Oswald:400');
/*----- reset*/
* {
    color: #424242;
    margin: 0;
    padding: 0;
    font-size: 100%;
    border: none;
    outline: none;
    font-weight: 300;
    box-sizing: border-box;
    font-family: 'PT Sans Narrow', sans-serif;
}

body {
    background-color: #F5FCFC;
}

a {
    text-decoration: none;
}

ul {
    list-style: none;
}

img {
    max-width: 100%;
}

#cabecalho {

```

```

width: 75%;
background-color: #FFF;
height: 100%;
overflow: hidden;
margin: auto;
}

#logo {
  float: left;
  padding: 1% 0% 1% 13%;
  width: 20%;
  height: 10%;
}

#menu {
  float: right;
  width: 80%;
  padding: 2.5% 0% 1% 10%;
  margin-top: 0.5em;
}

#menu li {
  float: left;
}

#menu a {
  padding: 0.3em 0.7em;
  font-size: 120%;
  display: block
}

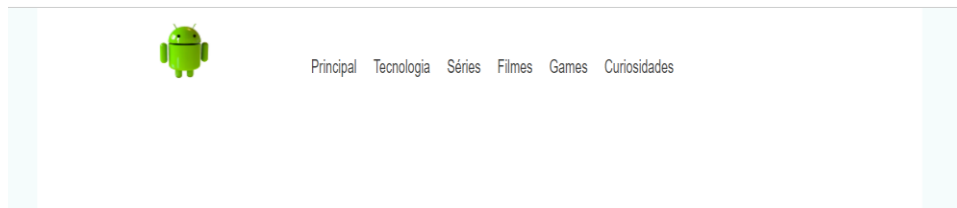
#menu a:hover {

```

```
font-weight: 600;  
border-bottom: solid 0.2em #424242;  
}
```

```
#principal {  
width: 75%;  
background-color: #FFF;  
height: 100%;  
overflow: hidden;  
margin: 0.5% auto;  
border-radius: 0.2em;  
}
```

O resultado será esse:



2. Primeiro crie uma pasta de nome projeto2. Logo após criar essa pasta crie mais duas pastas dentro da pasta projeto2. Uma pasta será para o arquivo css logo ela se chamara css, e outra para as imagens logo ela se chamara img.

Depois de criar a estrutura de seu projeto vá para a pasta de imagem e baixe uma imagem a seu gosto, mas lembre-se de ajustar o nome dela no código. Essa imagem servirá como logo. Quando terminar com as imagens crie um arquivo index. html e implemente o seguinte código:

```

<!DOCTYPE html>
<html lang="pt-br">
<head>
<meta charset="utf-8">
<title> Figuras </title>
<link href="estilo.css" rel="stylesheet">
</head>
<body>
<h1> Lorem Ipsum </h1>
<div>
<figure>

</figure>
<p>Lorem ipsum dolor sit amet, consectetur adipiscing elit.
estibulum auctor in ipsum vehicula consequat. Nullam nulla est,
aliquet ut turpis nec, porta ultrices orci. Aenean lobortis maximus
porta. Nulla dictum, ex et tempor rutrum, ex risus facilisis diam,
vestibulum cursus nibh nibh vel risus. Sed
tempus turpis nec ante iaculis, in porta dolor fermentum.
Suspendisse auctor sodales purus, id venenatis orci tincidunt ac.
Donec ut eros gravida, viverra lacus sed,
ullamcorper quam. Phasellus imperdiet nunc diam, eget sagittis
ante fe nibus vel. Duis vestibulum diam nec metus aliquet, vitae
porttitor risus sodales.</p>

<p>Nunc non gravida eros. Sed facilisis dapibus consequat.
Morbi pellentesque, urna quis condimentum dapibus, tortor
sapien pretium urna, eget tincidunt neque sem ac massa. Mauris
nec libero eleifend, tristique ipsum suscipit, accumsan neque.
Aliquam vitae purus faucibus, tempor odio a, pretium ante. Orci
varius natoque penatibus et magnis dis parturient montes,
nascetur ridiculus mus. In a fermentum mauris. Pellentesque

```


porta aliquet nulla eget lacinia. Etiam tincidunt purus lacus, in pharetra arcu efficitur id. Donec cursus placerat sodales. Nulla feugiat rutrum mi ut vehicula. Quisque dignissim egestas tempor. Sed maximus rutrum nisi sit amet ultricies. Duis at ligula ut dolor pulvinar tempus id eget neque.

Sed at commodo lectus. Duis sodales pulvinar nisi, in malesuada eros varius sed. Maecenas ut justo congue arcu porta facilisis vel egestas nisl. Pellentesque nec malesuada quam, ut placerat dolor. Etiam lacinia, massa sed fringilla vehicula, augue ante ullamcorper velit, eget fringilla elit orci eget dolor. Nullam tempus molestie tortor, at facilisis justo luctus nec. Mauris luctus pellentesque lacinia. Aliquam vestibulum nulla vitae pharetra suscipit. Orci at urna sed, volutpat dictum ex. Nulla eleifend massa non neque maximus condimentum. Proin dapibus est id ipsum fringilla maximus.

Após codificar a nossa página index, vá para a pasta css e crie lá um arquivo chamado estilo.css, feito isso implemente esses códigos no arquivo.

@import

url("https://fonts.googleapis.com/css?family=Roboto");

div {

background: #EEEEEE;

padding: 0.5% 1%

}

img {

```

width: 40%;
float: right;
padding: 0.5% 0.8%
}

p {
text-align: justify;
font-size: 1.2em;
font-family: "Roboto", sans-serif;
}

```

O resultado será esse:

Lorem Ipsum



3. Criação de uma landing page:

Primeiro crie uma pasta de nome landingpage. Logo após criar essa pasta crie mais três pastas dentro da pasta landingpage. Uma pasta será para o arquivo css logo ela se chamara css, e outra para as imagens logo ela se chamara img a terceira pasta se chamará fonts, ela irá conter as fontes da nossa página. Depois de criar a estrutura de seu projeto vá para a pasta de imagem e baixe uma imagem a seu gosto, mas lembre-se de ajustar o nome dela no código. Usaremos essa imagem no

momento certo. Quando terminar com as imagens crie um arquivo index. html e implemente o seguinte código:

```
<!DOCTYPE html>
<html lang="pt-br">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width,
initial-scale=1.0">
  <!-- Link para arquivo CSS-->
  <link rel="stylesheet" href="/css/style.css">
  <link rel="stylesheet" href="/css/fonts.css">
  <title>Manutenção ONLINE</title>
</head>
<body>
  <!--Usando tag semânticas-->
  <!--É a forma de deixar o site com suas informações bem
explicadas e compreensíveis para o computador ou seja
navegador "Interpretar de forma mais rápida as páginas"-->
  <!--usando a tag header para parte do cabeçalho-->
  <header>
    <div id="logo_title"> <!--Atribuido o id na Div para estilizar
somente a tag H1-->
      <h1>Manutenção</h1>
      <h1>Creative</h1>
    </div>

    <!--Tag "a" "ul" mais atributos para gerar um link e uma lista
desordenada -->
    <ul>
      <a href="#"><li>Início</li></a>
      <a href="#"><li>Serviços</li></a>
```

```

        <a href="#"><li>Sobre</li></a>
        <a href="#"><li>Contatos</li></a>
        <a id="t_conosco" href="#"><li>Trabalhe
Conosco</li></a>
    </ul>
</header>

<!--Parte principal do nosso site dentro do body-->
<main>
    <!--Tag aside semântica que possui por função um lado do
site-->
    <aside>

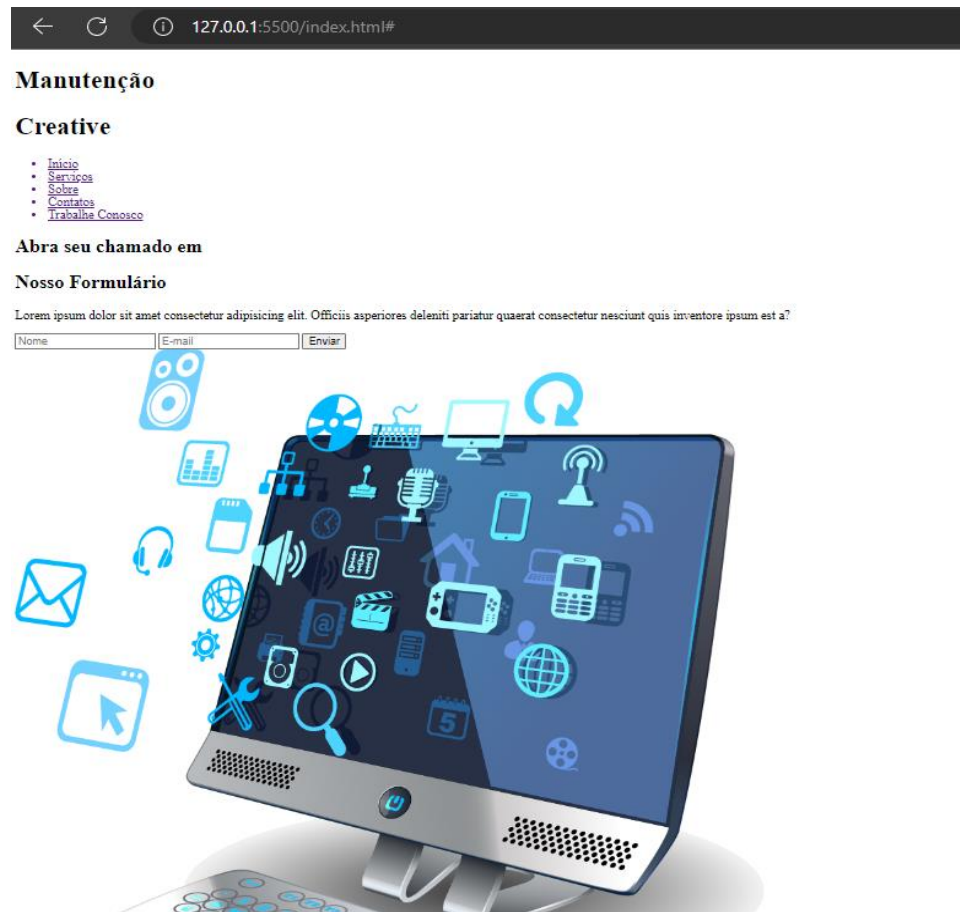
        <!--Tag de título-->
        <h2 id="h2_chamado"> Abra seu chamado em</h2>
<!--Id para estilização do h2-->
        <h2>Nosso Formulário</h2>
        <!--Tag de parágrafo-->
        <p>Lorem ipsum dolor sit amet consectetur
adipiscing elit. Officiis asperiores deleniti pariatur quaerat
consectetur nesciunt quis inventore ipsum est a?</p>
        <form>
            <input type="text" placeholder="Nome">
            <input type="email" placeholder="E-mail">
            <input id="enviar" type="submit" value="Enviar">
        </form>
    </aside>

    <!--Tag article é semântica pode ser usada para abrigar
também imagem-->
    <article>
        
    </article>

```

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

O resultado será esse:



Observe que temos uma página sem estilização, somente com os códigos inseridos no arquivo index.tml. Vamos inserir o CSS externo a nossa página.

```

/*Usaremos um reset no arquivo CSS para limpar qualquer
formatação de origem ou pré-configuração */
*{
    margin: 0; /* margem externa */
    padding: 0; /*magem do conteúdo apartir da tag*/
    box-sizing: border-box; /*indica que o tamanho agora levará
em conta até a borda */
    text-decoration: none; /* Qualquer decoração que tiver nos
textos*/
}
/* configurações na tag body, OBS: realizado o reset na font-
size:100%*/
body{
    font-size: 100%;
    background:linear-gradient(68.55deg, #E9FFF9 65.50%,
#a1f1d3 94.50%);
    color: blue; /*Aplicando color direto no body ele irá aplicar
somente nas fontes*/
    font-family: poppinsregular; /* Atribuindo a fonte descrita
no arquivo fonts.css*/
    max-width: 1280px; /*Tamanho da página para zoom vs
monitor* Obs: testar com scroll do mouse*/
    margin: 0 auto; /*Margem automática da página*/
    padding: 20px; /*Margem interna da página*/
}
/* configurações na tag header*/
header{
    display: flex; /*possibilita o deslocamento de conteudo em
todo o site desde, top, right, bottom e left*/
    flex-direction: row; /*tranforma o conteúdo cabeçalho qu por
padrão vem em coluna para linha*/

```

```

    justify-content:space-between; /*cria o espaço no contúdo
que está dentro do header*/
    align-items: center; /*alinha os conteúdos ou seja os itens */
}
/* Div criada para configurar o título "logo" na parte superior da
página*/
#logo_title{
    flex-direction: column; /* colocando a direção em coluna*/
    line-height: 30px; /*distância entre as palavras entre
coluna*/
}
h1{
    font-weight:100;
}

li{
    display: inline-block; /*desfaz o block e realiza o
seguimento das palavras em linha*/
    margin: 20px;
}
a{
    font-family: poppinsblack;
    color: rgb(58, 28, 194);
}
a:hover{
    color: rgb(97, 245, 220);
    transition: 0.2s all;
}
#t_conosco{
    border: 3px solid;
    padding: 10px;
    border-radius: 15px;

```

```

}
#t_conosco:hover{
    background-color: blue;
    color: rgb(97, 245, 220);
    border-color: blue;
}
main{
    display: flex;
    flex-direction: row;
    margin-top: 100px;
}

h2{
    font-size: 50px;
    line-height: 50px;
    font-family: poppinsbold;
}
#h2_chamado{
    color: rgb(66, 247, 217);
    font-size: 40;
    font-family: poppinsbold;
}

p{
    line-height: 20px;
    max-width: 500px;
    padding-top: 40px;
}
img{
    width: 580px;
    padding-right: 0;
}

```



```

form{
  display: flex;
  flex-direction: column;
  width: 70%;
  border-color: blue;
}

form [type="submit"]{
  color:rgb(97, 245, 220);
  font-family: poppinsregular;
  height: 20px;
  width: 60%;
  background-color: blue;
  padding: 5px 10px 35px 10px;
}

input{
  margin-top: 20px;
  height: 20px;
  padding: 25px;
  border-radius: 15px;
  font-size: 20px;
}

```

Nosso próximo passo é criar um arquivo com a extensão .css para adicionarmos as fontes a nossa página. Exemplo de como deve ser o nome desse arquivo: **fonts.css**.

OBS: Não esquecer de link esse arquivo no nosso index.html.

```

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

```

Agora vamos ao nosso código do fonts.css:

/ Sintaxe seletor e propriedade para puxar fonte. */*

```
@font-face {  
  font-family:poppinsblack ;  
  src: url(/fonts/Poppins-Black.ttf);  
  font-weight: normal;  
  font-style: normal;  
}  
  
@font-face {  
  font-family:poppinslight ;  
  src: url(/fonts/Poppins-Light.ttf);  
  font-weight: normal;  
  font-style: normal;  
}  
  
@font-face {  
  font-family:poppinsregular ;  
  src: url(/fonts/Poppins-Regular.ttf);  
  font-weight: normal;  
  font-style: normal;  
}  
  
@font-face {  
  font-family:poppinsbold ;  
  src: url(/fonts/Poppins-Bold.ttf);  
  font-weight: normal;  
  font-style: normal;  
}
```

Após criarmos as pastas necessárias com os respectivos arquivos e suas extensões, inserimos os códigos mencionados a cima, nossa landing page terá o seguinte resultado:



Referências Bibliográficas

- Jefferson Sales dos Santos - Coleção – Profissionalizante – Equipe Praxis Web Designer – Brasília/DF 2022.
- Renata Miyagusku - Desvendando os Recursos do CSS – São Paulo: Digerati Books.
- Bruno P. Campos - HTML e CSS - CFB Cursos - BH 2019
- Wallace Fragoso - Guia prático HTML & CSS - DevAcademy.
- Alfredo Limongi - HTML Apartir do Zero - ForenSys Caribbean.
- Jon Duckett - HTML & CSS Design and build Websites - John Wiley & Sons, Inc.
- Lucas Mazza – HTML5 e CSS3 Domine a Web do Futuro - Casa do Código.
- Eric Freeman & Elisabeth Robson – Use a Cabeça Programação em HTML5 – Alta Books.