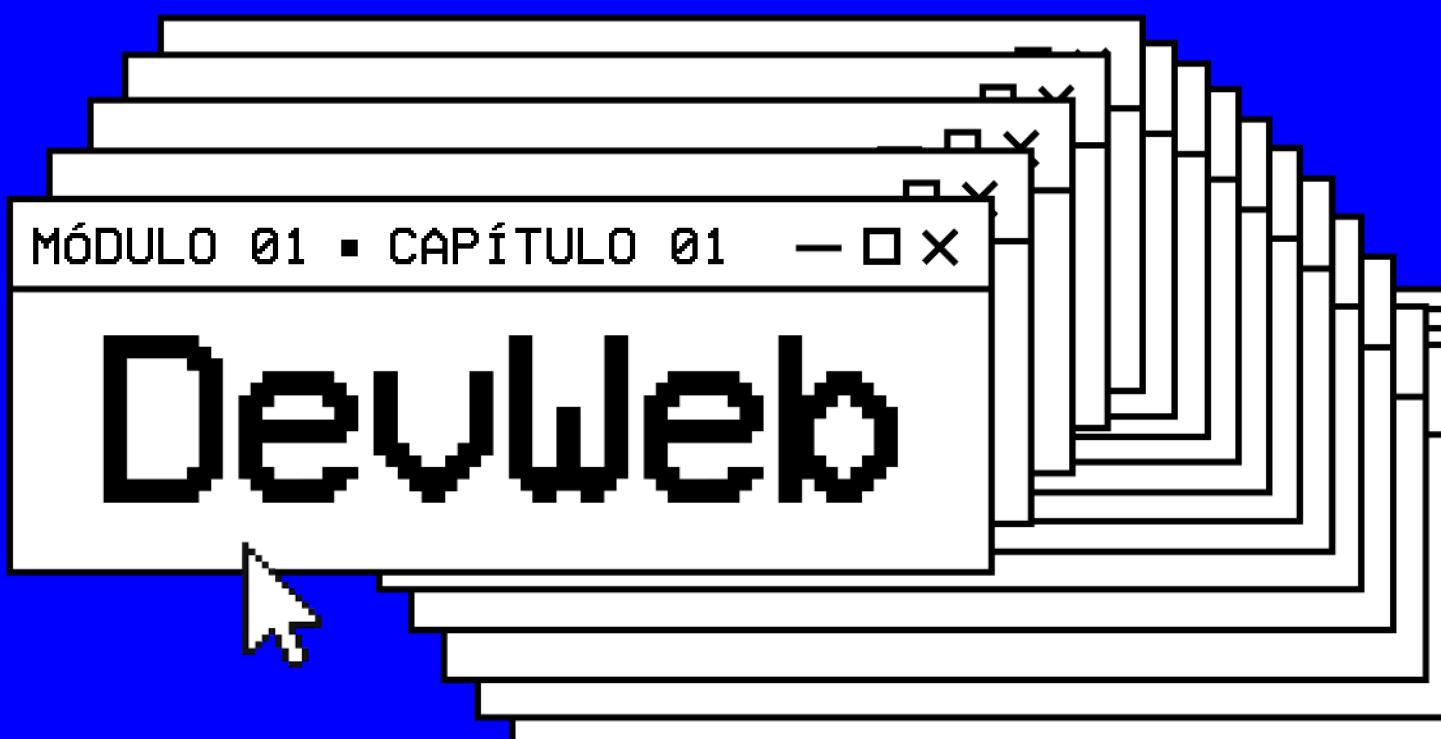
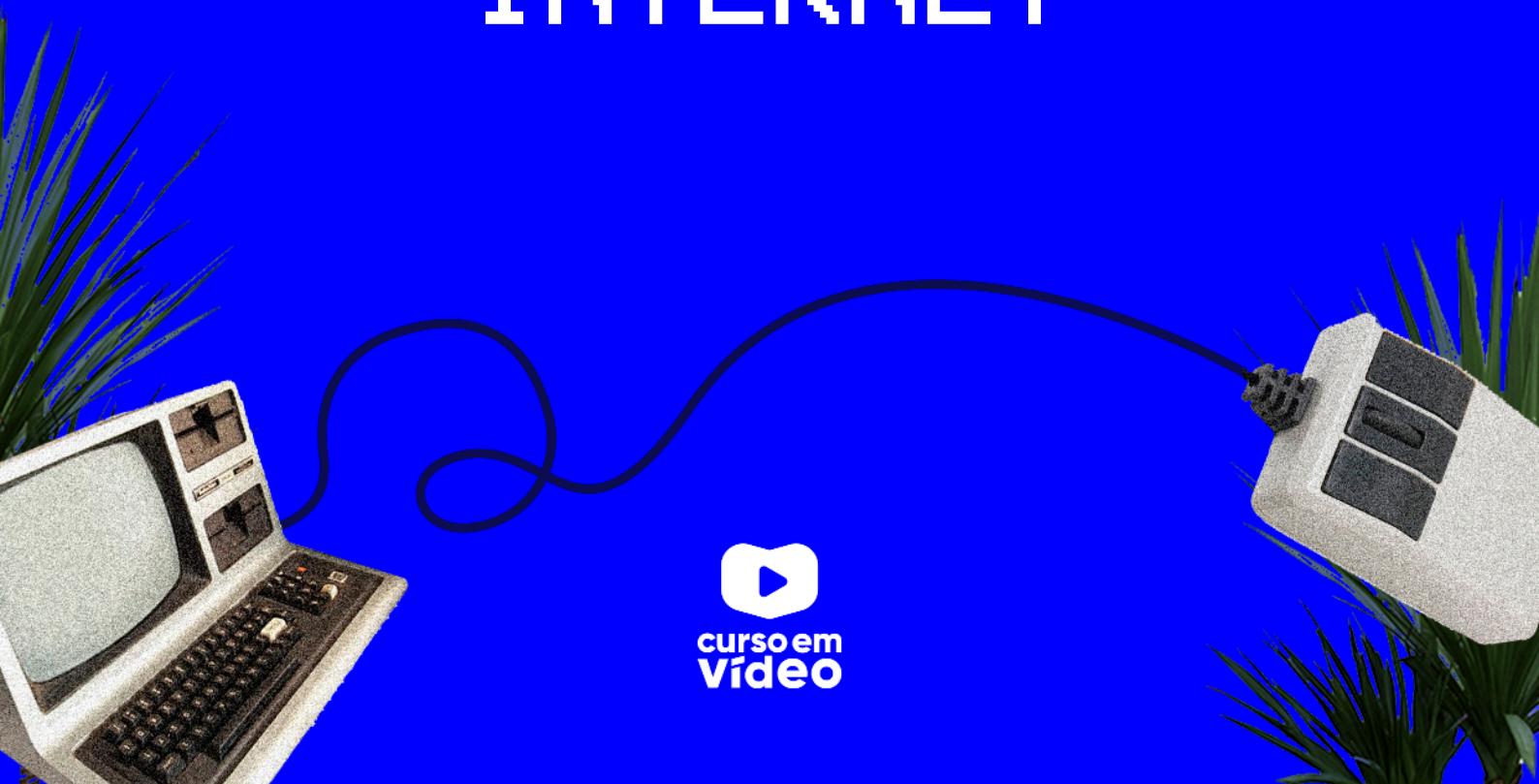


CURSO DE INFORMÁTICA PARA INTERNET

Gustavo Guanabara



# HISTÓRIA DA INTERNET



M01C01

veja em vídeo



# HISTÓRIA DA INTERNET

Todos nós temos uma História. Quando entendemos de onde viemos, conseguimos compreender a evolução até chegar onde estamos. E com a Internet não foi diferente. Tudo começa logo após a Segunda Guerra Mundial, com uma treta gigante entre Estados Unidos e União Soviética, que antes eram aliados mas agora se tornam inimigos. Acompanhe o desenrolar dessa novela e entenda como uma Guerra deu origem à maior rede do mundo. Vamos lá?



Você tem todo o direito de usar esse material para seu próprio aprendizado. Professores também podem ter acesso a todo o conteúdo e usá-lo com seus alunos. Porém todos o que usarem esse material - seja para qual for a finalidade - deverão manter a referência ao material original, criado pelo **Prof. Gustavo Guanabara** e disponível no endereço do seu repositório público <https://github.com/gustavoguanabara/>. Este material não poderá ser utilizado em nenhuma hipótese para ser replicada - integral ou parcialmente - por autores/editoras para criar livros ou apostilas, com finalidade de obter ganho financeiro com ele.



# A Internet veio da Guerra (infelizmente)

Depois da Segunda Guerra, EUA e URSS começaram a ter seus desentendimentos, dando origem à **Guerra Fria** em 1949. Neste contexto, em que os dois blocos ideológicos e politicamente antagônicos exerciam enorme controle e influência no mundo, qualquer mecanismo, qualquer inovação, qualquer ferramenta nova poderia contribuir nessa disputa liderada pela **União Soviética** e pelos **Estados Unidos**: as duas superpotências compreendiam a eficácia e a necessidade absoluta dos meios de comunicação.



**APRENDA MAIS:** Quer aprender mais sobre a Guerra Fria? Dá uma olhada aqui nesse vídeo de 9 minutos e com certeza você vai entender mais sobre essa treta toda.

Canal Descomplica: <https://youtu.be/cAwsLa04HGQ?t=49>

Nessa perspectiva, o governo dos Estados Unidos temia um ataque russo às bases militares. Um ataque poderia trazer a público informações sigilosas, tornando os EUA vulneráveis.

Então foi idealizado um modelo de troca e compartilhamento de informações que permitisse a descentralização das mesmas. Assim, se o Pentágono fosse atingido, as informações armazenadas ali não estariam perdidas. Era preciso, portanto, criar uma rede, a **ARPANET**, criada pela DARPA, sigla para **Defence Advanced Research Projects Agency**.



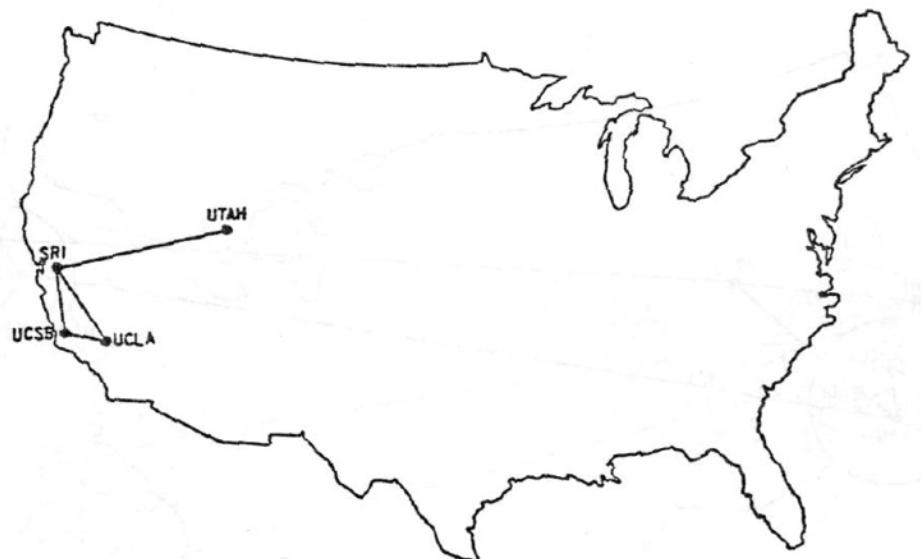
O ataque inimigo nunca aconteceu, mas o que o Departamento de Defesa dos Estados Unidos não sabia era que dava início ao maior fenômeno midiático do século 20', único meio de comunicação que em apenas 4 anos conseguiria atingir cerca de 50 milhões de pessoas.

## O começo de tudo

A **ARPANET** funcionava através de um sistema conhecido como *chaveamento de pacotes*, que é um sistema de transmissão de dados em rede de computadores no qual as informações são divididas em pequenos pacotes, que por sua vez contém:

- trecho dos dados
- o endereço do destinatário
- informações que permitiam a remontagem da mensagem original.

Em 29 de Outubro de 1969 ocorreu a transmissão do que pode ser considerado o **primeiro E-mail da história**. O texto desse primeiro e-mail seria "LOGIN", conforme desejava o Professor Leonard Kleinrock da Universidade da Califórnia em Los Angeles (UCLA), mas o computador no Stanford Research Institute, que recebia a mensagem, parou de funcionar após receber a letra "O".



A ARPANET no início, em 1969, só tinha 4 pontos

Já na década de 1970, a tensão entre URSS e EUA diminui. As duas potências entraram definitivamente naquilo em que a história se encarregou de chamar de **Coexistência**

— □ ×

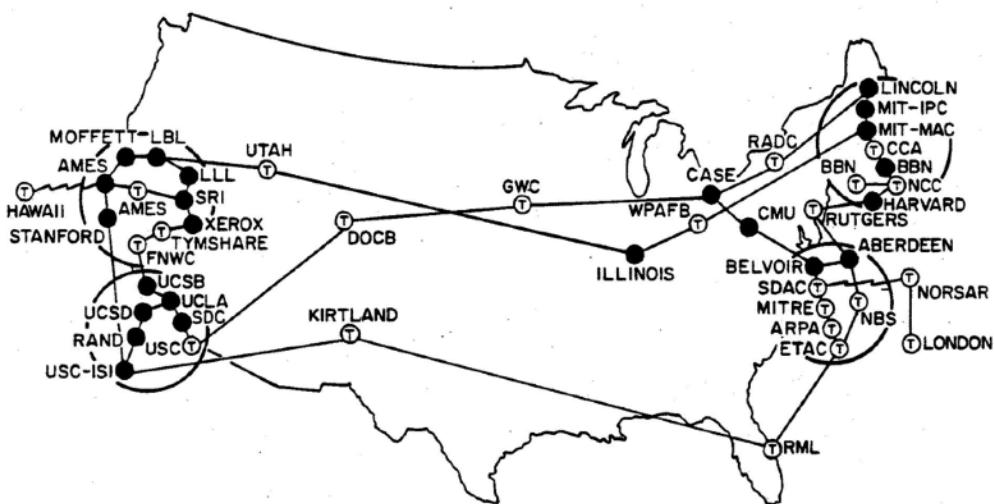
Cursos grátis de tecnologia  
que te preparam para o  
mercado de trabalho

RECODE

Scannable QR code linking to RECODE's free technology courses.

**Pacífica.** Não havendo mais a iminência de um ataque imediato, o governo dos EUA permitiu que pesquisadores que desenvolvessem, nas suas respectivas universidades, estudos na área de defesa pudessem também entrar na ARPANET.

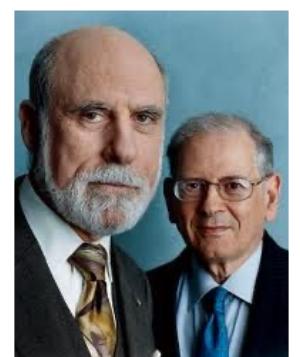
Com isso, a ARPANET começou a ter dificuldades em administrar todo este sistema, devido ao grande e crescente número de localidades universitárias contidas nela. Dividiu-se então este sistema em dois grupos, a **MILNET**, que possuía as localidades militares e a **nova ARPANET**, que possuía as localidades não militares. O desenvolvimento da rede, nesse ambiente mais livre, pôde então acontecer. Não só os pesquisadores como também os alunos e os amigos dos alunos, tiveram acesso aos estudos já empreendidos e somaram esforços para aperfeiçoá-los.



Já na década de 70, depois da abertura da rede para pesquisadores

Além desses backbones, existem os criados por empresas particulares. A elas são conectadas redes menores, de forma mais ou menos anárquica. É basicamente isto que consiste a Internet, que não tem um dono específico.

Com a entrada de muitos pontos na rede e com métodos de comunicação diferentes entre eles, alguma atitude tinha que ser tomada, já que o antigo protocolo **NCP** já não estava mais aguentando. *Robert Kahn* da DARPA e ARPANET recrutaram *Vint Cerf* da Universidade de Stanford para trabalhar com ele nesse problema. Em 1973, eles logo trabalharam com uma reformulação fundamental, onde as diferenças entre os protocolos de rede eram escondidas pelo uso de um protocolo inter-redes comum, e, ao invés da rede ser a responsável pela confiabilidade, como no ARPANET, os hospedeiros ficaram como responsáveis.



A especificação do protocolo resultante contém o primeiro uso atestado do termo internet, como abreviação de **internetworking**; então a palavra começou como um adjetivo, ao invés do nome que é hoje. Com o papel da rede reduzida ao mínimo, ficou possível a junção de praticamente todas as redes, não importando suas características, assim, resolvendo o problema inicial de Kahn. O DARPA concordou em financiar o projeto de desenvolvimento do software, e depois de alguns anos de trabalho, a primeira demonstração de algo sobre gateway entre a rede de Packet

Radio na Baía de SF área e a ARPANET foi conduzida. Decorrentes das primeiras especificações do TCP em 1974, **TCP/IP** emergiu em meados do final de 1978, em forma quase definitiva. Em 1º de janeiro de 1983, data conhecida como **Flag Day**, o protocolo TCP/IP se tornou o único protocolo aprovado pela ARPANET, substituindo o antigo protocolo NCP.

O cientista **Tim Berners-Lee** (foto ao lado), do **CERN**, criou a **World Wide Web**, a linguagem **HTML** e o protocolo **HTTP** em 1992. Essa linguagem simples, mas eficiente, era usada para a criação dos sites com o conceito de hipertexto (documentos ligados entre si).

A empresa norte-americana **Netscape** criou o protocolo **HTTPS** (HyperText Transfer Protocol Secure), possibilitando o envio de dados criptografados para transações comerciais pela internet.



**APRENDA MAIS:** A História da Internet tem muitos outros acontecimentos interessantes. Quer ver mais sobre isso? Então veja esse vídeo de 15 minutos que conta tudo com riqueza de detalhes, mas sem se tornar chato:

Canal TecMundo: <https://youtu.be/pKxWPo73pX0?t=27>

## A Internet no Brasil

Em 1989, o **Ministério da Ciência e Tecnologia** lança um projeto pioneiro, a **Rede Nacional de Ensino e Pesquisa** (RNP). Existente ainda hoje, a RNP é uma organização de interesse público cuja principal missão é operar uma rede acadêmica de alcance nacional. Quando foi lançada, a organização tinha o objetivo de capacitar recursos humanos de alta tecnologia e difundir a tecnologia Internet através da implantação do primeiro backbone nacional.

O primeiro backbone brasileiro foi inaugurado em 1991, destinado exclusivamente à comunidade acadêmica. Mais tarde, em 1995, o governo resolveu abrir o backbone e fornecer conectividade a provedores de acesso comerciais. A partir dessa decisão,

Soluções digitais para negócios

hostnet



surgiu uma discussão sobre o papel da RNP como uma rede estritamente acadêmica com acesso livre para acadêmicos e taxada para todos dos outros consumidores. Com o crescimento da Internet comercial, a RNP voltou novamente a atenção para a comunidade científica.

A partir de 1997, iniciou-se uma nova fase na Internet brasileira. O aumento de acessos a rede e a necessidade de uma infraestrutura mais veloz e segura levou a investimentos em novas tecnologias. Entretanto, devido a carência de uma infraestrutura de fibra óptica que cobrisse todo o território nacional, primeiramente, optou-se pela criação de redes locais de alta velocidade, aproveitando a estrutura de algumas regiões metropolitanas. Como parte desses investimentos, em 2000, foi implantado o backbone RNP2 com o objetivo de interligar todo o país em uma rede de alta tecnologia. Atualmente, o RNP2 conecta os 27 estados brasileiros e interliga mais de 300 instituições de ensino superior e de pesquisa no país, como o INMETRO e suas sedes regionais.

Outro avanço alcançado pela RNP ocorreu em 2002. Nesse ano, o então presidente da república transformou a RNP em uma organização social. Com isso ela passa a ter maior autonomia administrativa para executar as tarefas e o poder público ganha meios de controle mais eficazes para avaliar e cobrar os resultados. Como objetivos dessa transformação estão o fornecimento de serviços de infraestrutura de redes IP avançadas, a implantação e a avaliação de novas tecnologias de rede, a disseminação dessas tecnologias e a capacitação de recursos humanos na área de segurança de redes, gerência e roteamento.

A partir de 2005, a comunicação entre os point of presence (PoPs) da rede começou a ser ampliada com o uso de tecnologia óptica, o que elevou a capacidade de operação a 11 Gbps. A base instalada de computadores no Brasil atinge 40 milhões, de acordo com pesquisa da Escola de Administração de Empresas de São Paulo da Fundação Getúlio Vargas. O número, que inclui computadores em empresas e residências, representa um crescimento de 25% sobre a base registrada no mesmo período do ano passado.

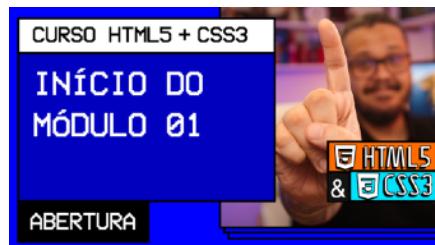


**APRENDA MAIS:** Quer ver mais informações sobre a chegada da Internet aqui no Brasil? Aqui vai mais um vídeo interessante que vai te contar todos os detalhes.

Canal TecMundo: [https://youtu.be/k\\_inQhpKprg?t=43](https://youtu.be/k_inQhpKprg?t=43)

# Quer acompanhar tudo em vídeo?

Eu sei que às vezes as pessoas gostam mais de assistir vídeos do que ler livros, e é por isso que eu lanço há anos materiais no canal Curso em Vídeo no YouTube. O link que vou compartilhar contigo faz parte da playlist completa onde você encontra o **Módulo 1 do Curso de HTML5 e CSS3**, completamente gravado com base nesse material.



Além de acessar o link a seguir, você também pode ter acesso às aulas apontando a câmera do seu celular para o código QR ao lado. Todo dispositivo smartphone ou tablet atualizado já possui esse recurso de leitura de códigos habilitado por padrão.

Módulo 1 do curso: [https://www.youtube.com/playlist?list=PLHz\\_AreHm4dkZ9-atkcmcBaMZdmLHft8n](https://www.youtube.com/playlist?list=PLHz_AreHm4dkZ9-atkcmcBaMZdmLHft8n)

## Teste seus conhecimentos

Terminou de ler esse capítulo e já acompanhou todos os vídeos e referências externas que indicamos? Pois agora, responda a essas 10 perguntas objetivas e marque em cada uma delas a única opção verdadeira. Aí sim, você vai poder comprovar que realmente entendeu o conteúdo.



1. Em qual período Histórico da humanidade a Internet surgiu?

- A Durante a 2ª Guerra Mundial
- B Durante a Guerra dos 100 anos
- C Durante a Guerra Fria
- D Durante a Guerra do Golfo

2. A agência militar americana DARPA foi a responsável pelo início dos estudos que deram origem à Internet. Logo no início, a rede teve um nome. Você sabe qual foi?

- A DARPANET
- B ARPANET
- C MILNET
- D NSFNET

3. A primeira transmissão registrada na rede foi no ano de \_\_\_\_\_. A primeira palavra enviada de um computador para outro foi \_\_\_\_\_.  
 A 1949 / HELLO  
 B 1955 / FIRST  
 C 1962 / WORKS  
 D 1969 / LOGIN

4. A primeira versão da rede que deu origem à Internet interligava quantos computadores?

- A 4 computadores
- B 40 computadores
- C 4 mil computadores
- D 400 mil computadores

5. O primeiro protocolo usado na transmissão de dados entre os controladores tinha o nome de:

- A TCP
- B NetBEUI
- C NCP
- D IP

6. No dia 1 de Janeiro do ano de \_\_\_\_\_ o TCP/IP passou a ser o único protocolo aceito pela Internet. Esse dia ficou conhecido como \_\_\_\_\_.

- A 1973 / Turn Point
- B 1983 / Flag Day
- C 1985 / New Hope
- D 1988 / Brand Tech

7. O conjunto de protocolos TCP/IP foi criado por um funcionário da DARPA e um pesquisador da universidade de Stanford. São eles, respectivamente:

- A Vint Cerf e Robert Kahn
- B Robert Kahn e Tim Berners-Lee
- C Tim Berners-Lee e Vint Cerf
- D Robert Kahn e Vint Cerf

8. O cientista inglês Tim Berners-Lee foi o responsável pela criação de três coisas muito importantes para a Internet. Foram elas:

- A a linguagem HTML, o protocolo HTTP e o primeiro navegador Mosaic
- B o TCP/IP, o primeiro navegador Mosaic e o nome WWW
- C a linguagem HTML, o protocolo HTTP e o nome WWW
- D o nome WWW, o primeiro navegador Mosaic e a linguagem HTML

9. O primeiro backbone brasileiro surgiu em \_\_\_\_\_ para acesso acadêmico, mas só foi liberado para empresas em \_\_\_\_\_.

- A 1991 / 1995
- B 1990 / 1997
- C 1989 / 1991
- D 1985 / 1995

10. Atualmente, a Internet Brasileira atinge quantos estados?

- A 19 estados, além do Distrito Federal
- B 23 estados, além do Distrito Federal
- C 25 estados, além do Distrito Federal
- D 26 estados, além do Distrito Federal

**Cursos que vão te levar  
ao próximo nível**

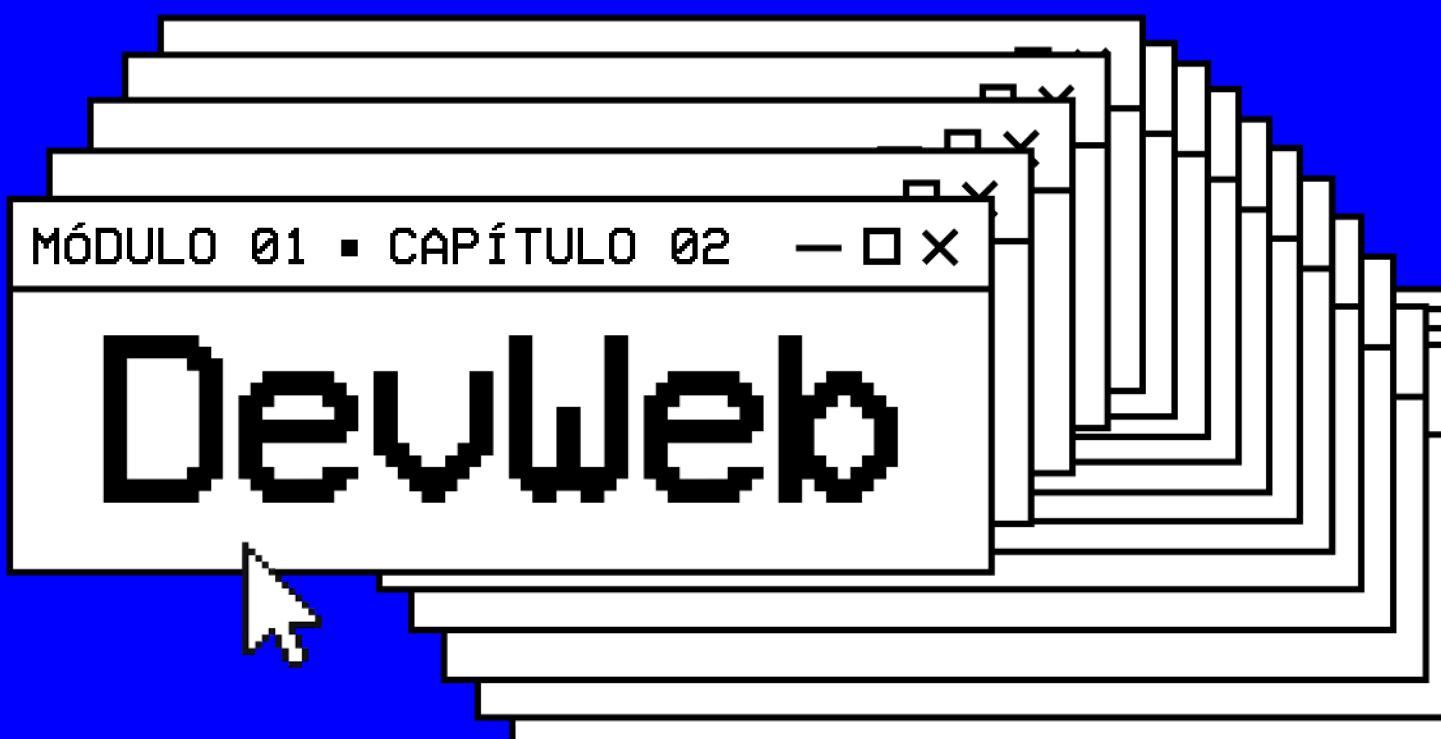


# **Suas anotações**

*Não guarde conhecimento. Ele é livre. Compartilhe o seu e veja ele se espalhando pelo mundo* 

CURSO DE INFORMÁTICA PARA INTERNET

Gustavo Guanabara



# COMO A INTERNET FUNCIONA?



M01C02

veja em vídeo



# COMO A INTERNET FUNCIONA?

Você é daquele tipo de pessoa curiosa, que não sossega enquanto não descobre como funcionam as coisas? Eu sou assim! Hoje em dia, a maioria das pessoas que conhecemos estão conectadas à Internet. Mas você sabe como funciona essa comunicação? Como é possível conversar ou mandar um arquivo para uma pessoa que está no Japão e tudo chegar em menos de 1 segundo? Esse é o nosso desafio nessa aula.



Você tem todo o direito de usar esse material para seu próprio aprendizado. Professores também podem ter acesso a todo o conteúdo e usá-lo com seus alunos. Porém todos que usarem esse material - seja para qual for a finalidade - deverão manter a referência ao material original, criado pelo **Prof. Gustavo Guanabara** e disponível no endereço do seu repositório público <https://github.com/gustavoguanabara/>. Este material não poderá ser utilizado em nenhuma hipótese para ser replicada - integral ou parcialmente - por autores/editoras para criar livros ou apostilas, com finalidade de obter ganho financeiro com ele.

# Tudo começa com o sinal

Você sabe que um computador é um equipamento eletrônico capaz de entender apenas um tipo de sinal, não sabe?

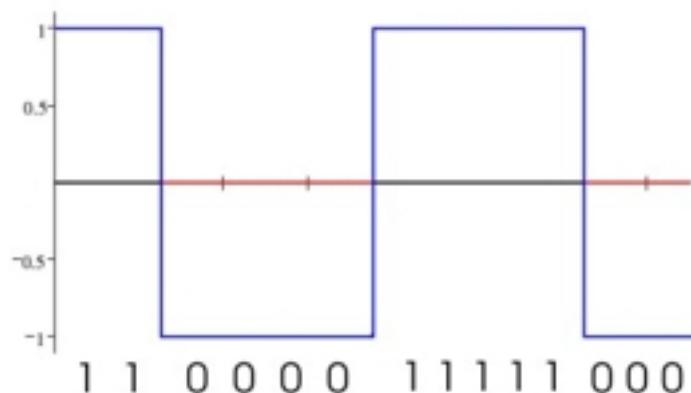
Tem gente que acha que um computador é um dispositivo super inteligente e cheio de capacidades de decidir nosso destino, mas na verdade ele é apenas uma máquina capaz de analisar sinais e fazer contas simples de uma maneira super rápida!

Eu sinto te informar, mas resumindo bastante aqui, o computador só é capaz de compreender duas coisas: 0 e 1.



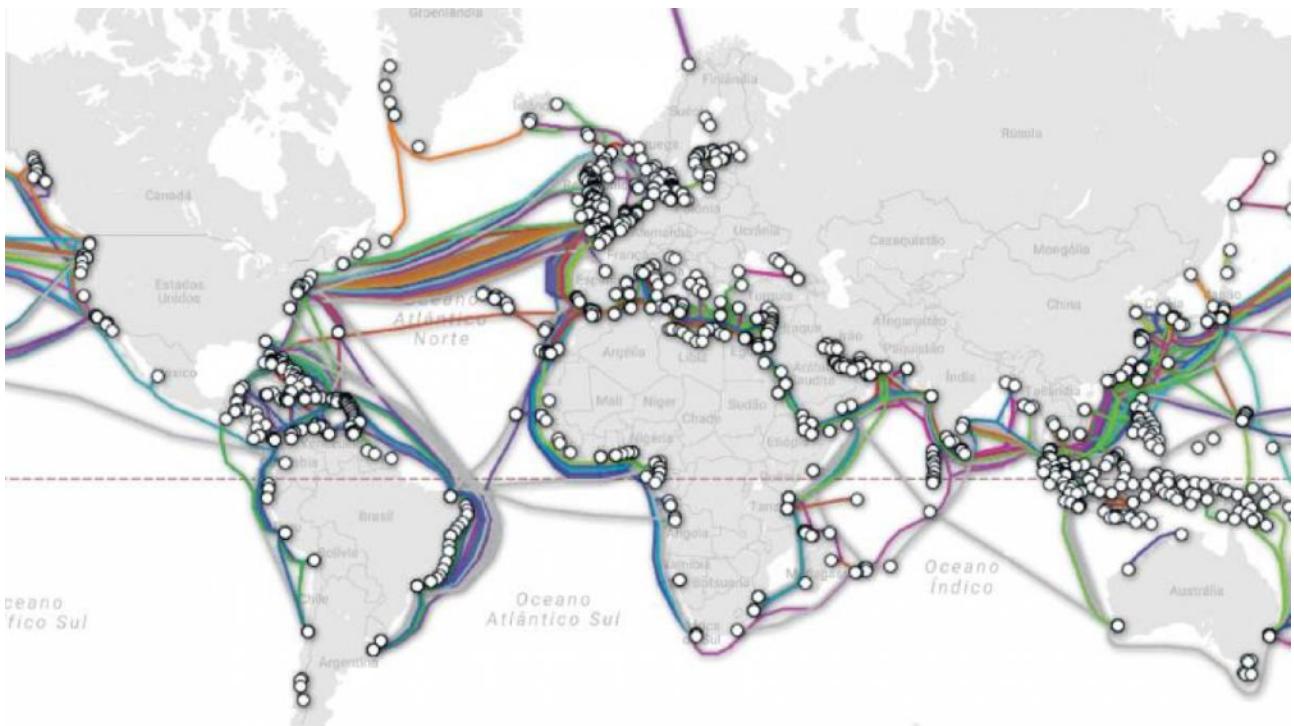
Bits, Bytes e seus múltiplos. Essa é a linguagem do seu computador

Mas é claro que o que circula dentro do seu computador não são pequenos números, são ondas (sinais). No caso de equipamentos eletrônicos processados, as ondas se parecem com as representadas a seguir.



Esta é uma onda que representa os bits, também chamada de onda digital

Como vimos na aula anterior, a Internet é uma rede gigantesca que interliga várias outras redes ao redor do mundo. E precisamos ter meios físicos para levar esses sinais de um lado para o outro.



Este aí é nosso planeta e suas interligações feitas pelos oceanos



**APRENDA MAIS:** Quer descobrir detalhes de cada um dos cabos que estão representados acima? O site Submarine Cable Map cria um mapa interativo e você pode clicar em cada um deles.

Acesse agora: <https://www.submarinemap.com>

E se nesse exato momento você está confuso(a) com esse conceito, saiba que a maioria das transmissões entre continentes não é realizada pelos **Satélites**, como a maioria das pessoas costuma pensar. Os satélites possuem uma limitação de tráfego e sofrem muito com interferências, e é por isso que os cabos de fibra ótica devem ser passados pelos oceanos. Um trabalho incalculável, mas necessário.

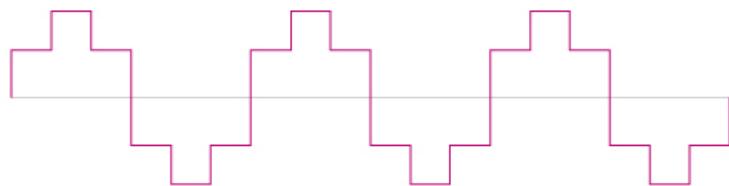


**APRENDA MAIS:** Quer aprender mais sobre os cabos submarinos? Assista esse vídeo de 9 minutos que explica mais detalhes.

Dicionário de Informática: <https://youtu.be/q-rBtDub3Hc>

Mas não dá para esses cabos submarinos saírem pela praia e seguirem caminho até a sua casa, e por isso precisam se interligar a outros sistemas de comunicação. Alguns dos sistemas utilizados sempre foram a telefonia tradicional, os sistemas de TV a cabo, os sinais via satélite e até as simples redes de radiofrequência.

sinal digital



sinal analógico



O problema é que os sistemas diferentes transmitem sinais em formatos diferentes, como você pode ver na imagem acima. Isso dificultaria a comunicação entre pontos, se não fosse um processo de "conversão", mais conhecido como **MODULAÇÃO**.

De uma maneira bem resumida, modular é conseguir ler uma onda no formato A compatível com um tipo de sistema de comunicação e convertê-la para um formato B, compatível com outro tipo de sistema.

E é exatamente para isso que servem aqueles aparelhos que você instala em sua casa para começar a receber Internet doméstica



Aparelho muito comum em casas com Internet

The banner features a purple background with white text and graphics. On the left, the text reads 'Soluções digitais para negócios'. On the right, there is a QR code. Below the text is the Hostnet logo, which consists of a stylized cloud icon followed by the word 'hostnet'.

Uma das funções desse aparelho é **MODULAR** os sinais que saem e **DEMODULAR** os sinais que chegam. E é por isso que chamamos esse aparelho de **MODEM**.

## Já que falamos de Roteadores

As rotas são outro assunto muito importante para o funcionamento da Internet. Pense na rede como se fosse como se fosse um mapa com várias ruas, como no desenho a seguir. De quantas maneiras diferentes podemos chegar do ponto A até o ponto B? Eu imaginei três rotas diferentes e representei usando as linhas coloridas. Quem vai decidir a melhor rota é o Aplicativo de GPS, já que podem existir engarrafamentos e ruas fechadas.

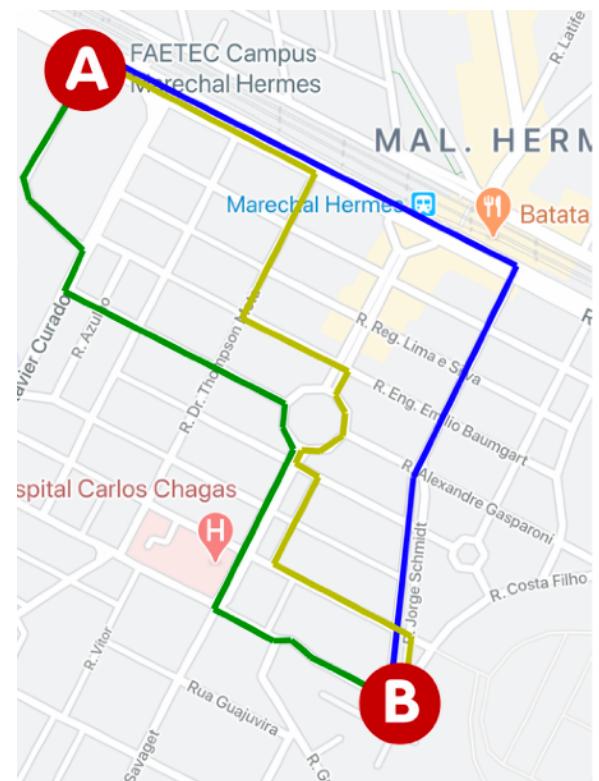
Na Internet também é assim. Para enviar um sinal de um dispositivo A para um dispositivo B, podemos ter várias rotas. Quem vai definir a melhor rota são os **ROTEADORES** que compõem a rede. Os pacotes de dados podem chegar em seu computador por diversas rotas diferentes, tudo vai depender do tráfego no momento da transmissão.

Agora que você já sabe como funcionam as rotas, vamos falar sobre os pontos.

## Cliente e Servidor

Volte uma página e olhe aquele mapa de novo. Imagine que o ponto A é você na escola pedindo uma pizza. O ponto B é a pizzaria, que vai te fornecer o pedido que vai matar sua fome. Quando o pedido for confirmado, o motoboy que vai fazer a entrega é o pacote. Ele vai levar seu pedido até você por uma rota.

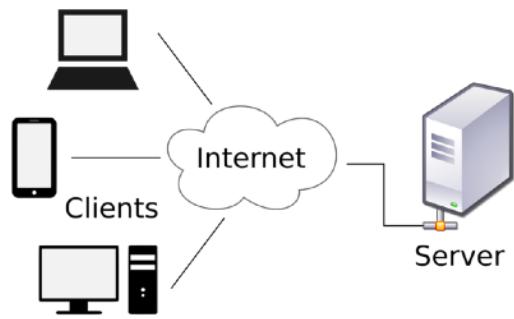
Nessa situação que descrevi acima, você no ponto A é o **CLIENTE**. A pizzaria no ponto B é o **SERVIDOR**. O motoboy é o **PACOTE** e sua pizza é o **DADO**. A Internet também vai funcionar dessa maneira.



### IMPORTANTE!

Quando você estudar um pouco mais sobre redes de comunicação, vai descobrir que a função **MODEM** desses aparelhos é apenas uma das características do produto. Na verdade, esse dispositivo é um **GATEWAY**, que vai se ligar aos **ROTEADORES** do seu provedor de acesso, mas resolvi simplificar a explicação para não aprofundar tanto assim no início de tudo.

Olhe agora o desenho ao lado. Ele é bem parecido com a história do mapa. Imagine que você está no seu celular tentando acessar um site. O seu celular é o **CLIENTE** e está pedindo algo pela Internet. Ao descobrir onde está o site, a máquina que está hospedando ele será o **SERVIDOR**, que vai fornecer os arquivos que compõem o site. O caminho que vai criar uma ligação entre o servidor e você (cliente) vai ser decidido pelos roteadores da Internet.



Um servidor pode estar no seu bairro, na sua cidade, no seu país ou até mesmo do outro lado do mundo. Os pacotes podem girar o mundo todo em poucos segundos e o resultado será exibido na tela do seu celular/computador como se fosse magia, mas é pura TECNOLOGIA!

Na Internet existem vários servidores:

- Servidor de site (também chamado de WebHost)
- Servidor de streaming
- Servidor de arquivos
- Servidor de e-mail
- E muito mais...

Mas como será que o mecanismo da Internet consegue descobrir o local exato de um site? Como ele descobre em que servidor ele está? E como ele consegue encontrar a posição exata do servidor no Globo? Aí entramos no próximo assunto.

## Identificando os nós

Como vimos anteriormente, a Internet funciona baseada em um conjunto de protocolos chamado **TCP/IP**. Um protocolo garante que todas as comunicações seguirão um mesmo padrão, permitindo que dispositivos que são diferentes, com tecnologias completamente distintas, possam se trocar mensagens.

Uma das funções do TCP/IP, mais especificamente do IP, é identificar os nós. Mas o que seria esse nó?

A resposta é simples: um nó é cada ponto que está conectado à rede. Quando você "se conecta" à Internet, recebe uma identificação única. Essa identificação é um **ENDEREÇO IP**.

Cursos grátis de tecnologia  
que te preparam para o  
mercado de trabalho

RECODE

QR code

Os IPs mais antigos (IPv4) usam 4 octetos, que são conjuntos de 8 bits separados por pontos, totalizando 32 bits por identificador.

Ex: **123.45.67.89** = 01111011.00101101.01000011.01011001

Os IPs mais modernos (IPv6), usam 128 bits ao todo (o que é 4x mais bits que o IPv4).

Ex: **2001:0db8:85a3:08d3:1319:8a2e:0370:7344**

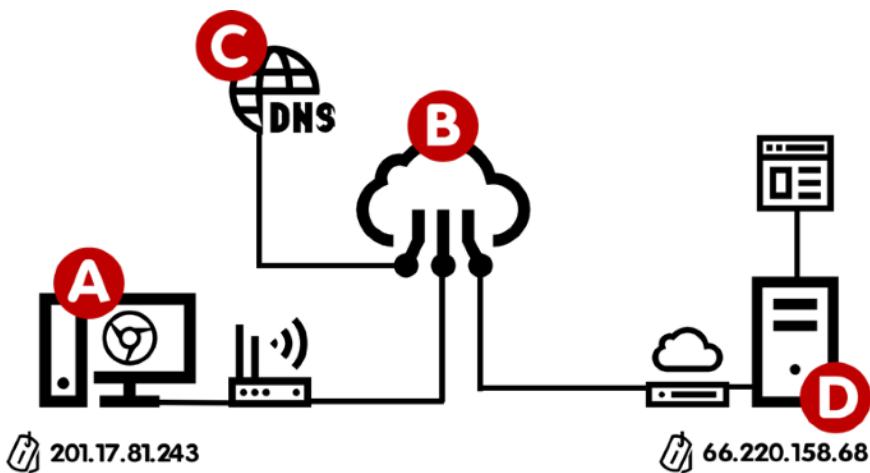


**APRENDA MAIS:** Entenda melhor qual é a necessidade de migrar do IPv4 para o IPv6. E o que acontecerá no futuro, quando a versão antiga parar de funcionar.

NICbr: [https://youtu.be/\\_JbLr\\_C-HLK](https://youtu.be/_JbLr_C-HLK)

## Acessando um servidor

Agora que você já sabe como os pontos são identificados, vamos criar um simples cenário aqui. Analise a imagem abaixo e veja que você estaria no ponto A, tentando acessar o site que está guardado no servidor que é o ponto D.



Você também deve ter notado que o ponto A tem um IP (201.17.81.243) e o ponto D também tem o seu (66.220.158.68). Agora imagine que você deva ter que decorar o IP do seu site favorito. Isso dificultaria todo o processo, não é?

E é para isso que existe o **DOMAIN NAME SYSTEM**, ou sistema de nomes de domínio. Eles são como grandes “listas telefônicas”, criando uma ligação entre o nome do site e o número de IP relacionado a ele.

Importante deixar bem claro: os números de IP mudam constantemente! Sempre que você desconecta o gateway da sua operadora (aquele aparelho que tem instalado na sua casa), o seu número de IP vai mudar.



**APRENDA MAIS:** Saiba mais sobre o DNS assistindo esse vídeo bem simples e ilustrado.

NICbr: <https://youtu.be/ACGuo26MswI>

É possível fazer um exercício simples para descobrir o seu próprio IP ou até mesmo descobrir o IP atual de um site que você esteja acessando. Descubra como, lendo o quadro informativo abaixo.

Agora vamos voltar ao desenho da página anterior e entender o passo-a-passo desse acesso.

1. Você está no **ponto A** (conectado à Internet) e digita o endereço do site que está querendo acessar (ex: [www.facebook.com](http://www.facebook.com)).
2. A arquitetura da Internet (**ponto B**) vai encontrar o **Servidor DNS** que terá o registro do IP referente ao nome que você digitou.
3. O Servidor DNS, que é o **ponto C** do nosso desenho, vai resolver o endereço e retornar o IP atual do site que você pediu (ex: no dia em que eu escrevi esse material, o IP atual do [facebook.com](http://facebook.com) era 66.220.158.68).
4. Uma vez que seu navegador já sabe o IP que deseja acessar, vai poder se direcionar diretamente ao servidor correto.
5. Assim que chega uma solicitação ao **ponto D**, ele vai procurar a página solicitada e te enviar uma cópia do documento para o seu computador.



**APRENDA MAIS:** Tente descobrir o seu IP e também tente descobrir o IP atual do site do Facebook usando o site a seguir:

IP2Location: <https://www.iplocation.net>

**EXTRA:** Depois que você descobrir o IP do Facebook, tente digitar o número dele no lugar do endereço do site no seu Google Chrome e aperte o enter. Surpresa!

**OBS:** Caso você não solicite um documento específico, o servidor vai te enviar o arquivo de índice (geralmente chamado `index.html`).

6. Agora que o seu computador no **ponto A** tem o arquivo HTML, vai poder analisá-lo para descobrir que arquivos extras ele vai precisar (fotos, vídeos, estilos, etc). A partir daí ele vai fazer outras solicitações ao **ponto D**, para que ele possa enviar esses recursos extras. É por conta disso que os sites que você visita vão aparecendo aos poucos.

# Domínio e Hospedagem

Tá aí um assunto que muita gente iniciante fica na dúvida. Até mesmo seus clientes vão perguntar sobre isso, pois serão duas contas pra pagar.

Quando você criar um site, vai querer que o maior número de pessoas tenha acesso a ele, não é? O problema é que enquanto você está desenvolvendo HTML em casa, na maioria das vezes esses arquivos estão guardados no seu próprio PC. Dessa maneira, não existe um endereço para outras pessoas poderem acessar o seu trabalho de fora.



## Vamos começar com o Domínio

Quando você acessa um site, precisa saber uma **URL** (*Uniform Resource Locator*) para poder acessá-lo. A parte principal dessa URL é o **domínio**.

O **domínio** é um nome único que vai conseguir identificar o seu servidor ou as suas páginas. Para conseguir um domínio, você deve pagar pelo direito de usá-lo por um período mínimo de 1 ano.

Vejamos alguns exemplos de domínio:

cursoemvideo.com  
faetec.rj.gov.br  
github.io  
universidadebrasil.edu.br



Analisando os endereços acima, temos domínios com várias terminações, como .com, .gov.br, .io, .edu.br. Essas terminações indicam tecnicamente que o site é de uma instituição comercial, governamental, educacional, ONGs, artistas, etc.

Além disso, alguns desses ainda indicam o país (.br). Esses são os **TLD** (*Top Level Domain*).

**GTLD**: São TLDs genéricos, sem indicação de país. Alguns dos domínios genéricos são .com, .net, .gov, .org, .io, .info, .online, .store, etc.

**ccTLD**: São TLDs com designação do país (*country code*). Alguns dos domínios desse tipo são .com.br, .edu.us, .co.fr, .jp, .es, etc.

Da junção entre o nome e a terminação, temos um domínio. Esse domínio pode ser usado para ter acesso ao seu servidor, seja ele web, servidor de ftp, servidor de e-mail (as contas de e-mail são no formato josesilva@cursoemvideo.com, onde o nome do usuário vem antes da @ e o domínio vem depois)

E já que eu falei de **sub-domínio**, podemos ter outros alem do www, ok? Um mesmo site pode ter vários sub-domínios. Vamos usar o Google como exemplo:

www.google.com.br - Dá acesso ao site principal do Google  
images.google.com.br - Você vai para o Google Imagens  
maps.google.com.br - Acessa o Google Maps  
mail.google.com.br - Entra no Gmail



**CUIDADO!** Tem muita gente que pensa que o **www** que vem antes da URL faz parte do domínio. Não faz! Esse www é um sub-domínio que aponta diretamente para o seu servidor web padrão.

Nesse caso o www, images, maps e mail são sub-domínios do domínio google.com.br. Sacou?

Além do sub-domínio e domínio, uma URL também é composta pelo protocolo utilizado, que pode ser `http://` ou `https://`, dependendo se o seu servidor tem ou não segurança por **SSL** (um serviço de criptografia de dados).

E por fim, uma URL também pode ter um caminho extra, que indica pastas a percorrer para achar um arquivo específico. Por exemplo:

`https://www.github.com/gustavoguanabara/html-css/tree/master/aulas-pdf`

Identifique na URL acima os componentes da URL. Faça esse exercício.

Como vimos, o domínio é uma maneira mais fácil para que outras pessoas possam chegar aos arquivos que vão compor o seu site. Mas onde estarão esses arquivos?

## Agora chegou a vez da Hospedagem

No momento em que você começa a desenvolver seu site, seus arquivos estão no PC da sua casa.



Mas seu computador fica ligado 24h por dia, 7 dias por semana, 365 dias ao ano? Se faltar luz, você tem uma sub-estação de energia pra manter tudo funcionando em caso de panes? Você tem uma banda de internet suficiente para suportar milhares ou até milhões de acessos simultâneos?

Pois para manter seus arquivos acessíveis, você vai precisar contratar um plano de hospedagem justamente para que tenha tudo aquilo que foi enumerado acima. Para isso, existem grandes **data centers** com toda a infraestrutura para fornecer disponibilidade, segurança e performance para guardar seus arquivos. Algumas **empresas de hospedagem** contratam esses data centers para fornecer espaços para colocar servidores e alugam esses servidores para nós.



**VEJA POR DENTRO:** Dá só uma olhada em como é um data center por dentro. E você aí achando que seu PC com no-break já dava conta do recado 😊

Data Center Ascenty: [https://youtu.be/qV1Mg-H\\_fBI](https://youtu.be/qV1Mg-H_fBI)

Data Center UOL: <https://youtu.be/flxzjA4tM1E?t=74>

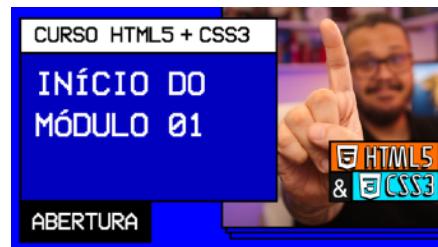
Data Center Locaweb: <https://youtu.be/a2vfVALjv18>

A **Hostnet**, que sempre foi nossa patrocinadora oficial do **Curso em Vídeo**, tem seus servidores na **Ascenty**, um dos maiores data centers do Brasil. Ao contratar uma **hospedagem** para o seu site, o plano normalmente inclui recursos como espaço em disco, memória, banco de dados, contas de e-mail, webmail, sub-domínios, certificados de segurança, etc. Nem todas as hospedagens oferecem tudo isso, por isso é muito importante saber escolher. Nós, é claro, recomendamos sempre a **Hostnet**.



## Quer acompanhar tudo em vídeo?

Eu sei que às vezes as pessoas gostam mais de assistir vídeos do que ler livros, e é por isso que eu lanço há anos materiais no canal Curso em Vídeo no YouTube. O link que vou compartilhar contigo faz parte da playlist completa onde você encontra o **Módulo 1 do Curso de HTML5 e CSS3**, completamente gravado com base nesse material.



Além de acessar o link a seguir, você também pode ter acesso às aulas apontando a câmera do seu celular para o código QR ao lado. Todo dispositivo smartphone ou tablet atualizado já possui esse recurso de leitura de códigos habilitado por padrão.

Módulo 1 do curso: [https://www.youtube.com/playlist?list=PLHz\\_AreHm4dkZ9-atkcmcBaMZdmLHft8n](https://www.youtube.com/playlist?list=PLHz_AreHm4dkZ9-atkcmcBaMZdmLHft8n)

# Teste seus conhecimentos

Terminou de ler esse capítulo e já acompanhou todos os vídeos e referências externas que indicamos? Pois agora, responda a essas 10 perguntas objetivas e marque em cada uma delas a única opção verdadeira. Aí sim, você vai poder comprovar que realmente entendeu o conteúdo.



1. Atualmente, na maioria das transmissões feitas mundialmente pela Internet, por onde a maior quantidade de sinal vai passar?

- A Por baixo d'água, em cabos submarinos
- B Por sinal de satélite, partindo de antenas
- C Por antenas de comunicação via celular
- D Por cabos enterrados nos continentes

2. Os dados que vão trafegar pelas redes e chegar aos computadores vão estar codificados em:

- A letras e símbolos
- B conjuntos de bits e bytes (0s e 1s)
- C ondas codificadas sem um padrão definido
- D arquivos de imagem / textos

3. Quando um sinal está representado em um determinado tipo de onda (codificação) e precisamos transformá-lo em outro tipo, dizemos que estamos realizando uma:

- A extração
- B composição
- C descompressão
- D modulação

4. Na maioria das vezes que estamos realizando algum tipo de acesso via rede, um lado será o \_\_\_\_\_ e vai solicitar o uso de um determinado serviço. Quem vai atender à essa solicitação é considerado um \_\_\_\_\_.

- A servidor / cliente
- B cliente / servidor
- C solicitante / provedor
- D provedor / servidor

5. Todo e qualquer ponto conectado à Internet recebe um identificador, que se chama “endereço IP”. Na versão IPv4, um endereço possui \_\_\_\_\_ bits, já a versão IPv6 necessita de \_\_\_\_\_ bits para identificar um ponto.

- A 32 bits / 64 bits
- B 64 bits / 128 bits
- C 32 bits / 128 bits
- D 16 bits / 64 bits

6. Qual é o serviço (e seu respectivo significado) responsável por resolver os endereços IP dos servidores a partir de um nome? Ele faz com que os usuários não precisem decorar números IP (que inclusive, mudam constantemente).

- A DNS - Domain Name System
- B DNS - Domain Name Service
- C DNS - Domain Name Server
- D DNS - Data Name Server

Cursos que vão te levar  
ao próximo nível



7. Uma URL é composta por vários componentes. Na URL `https://www.github.com/gustavoguanabara`, por exemplo, são respectivamente o domínio, o protocolo e o caminho os itens apontados na opção:

- A `github.com / https / gustavoguanabara`
- B `github.com / www / gustavoguanabara`
- C `github / https / gustavoguanabara`
- D `www.github.com / www / gustavoguanabara`

8. Quando chegar a hora de construir nossos sites, devemos informar aos nossos clientes que geralmente o domínio tem uma taxa de pagamento \_\_\_\_\_, enquanto as hospedagens são de pagamento \_\_\_\_\_ de forma geral.

- A mensal / anual
- B anual / diário
- C mensal / diário
- D anual / mensal

9. Os domínios possuem seus Top Level Domains. Vejamos como exemplo os domínios `estudonauta.com.br` e `cursoemvideo.com`. No primeiro caso, temos um \_\_\_\_\_ e no segundo caso, temos um \_\_\_\_\_.

- A GTLD / ccTLD
- B ccTLD / GTLD
- C DTLD / CTLD
- D CTLD / DTLD

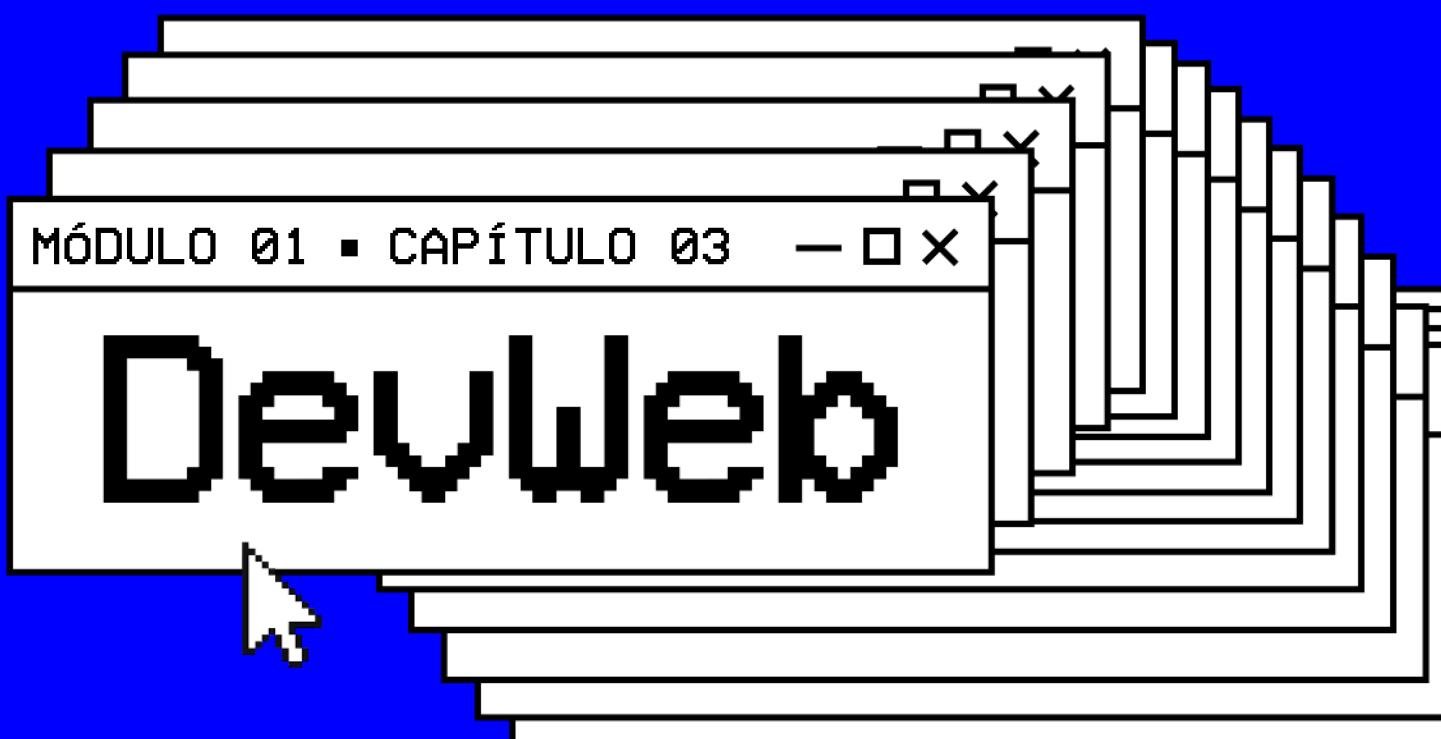
10. Todo bom serviço de hospedagem deve fornecer recursos e serviços valiosos para seus clientes. Entre os itens a seguir, qual é o único que NÃO É oferecido por uma empresa de hospedagem?

- A atendimento de suporte
- B espaço em disco para armazenamento
- C manutenção periódica no PC do cliente
- D backup constante dos arquivos e bancos de dados

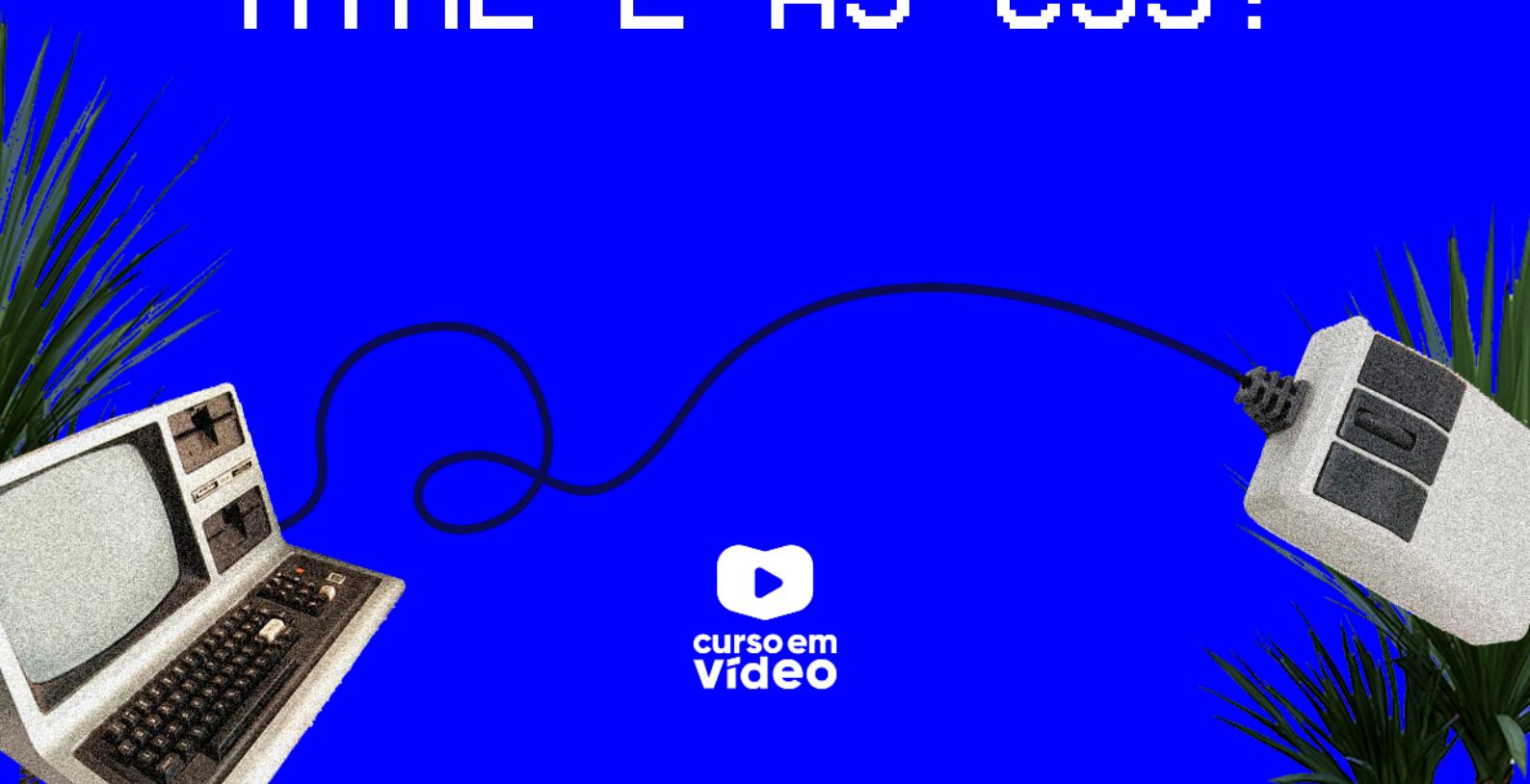
# **Suas anotações**

*Não guarde conhecimento. Ele é livre. Compartilhe o seu e veja ele se espalhando pelo mundo* 

Gustavo Guanabara



# COMO FUNCIONA A HTML E AS CSS?





# COMO FUNCIONA A HTML E AS CSS?

Pode ser estranho pra você ler “a HTML” e também “as CSS”. Muita gente trata as duas tecnologias com o artigo masculino na maioria das vezes. Na verdade, isso nem está totalmente errado, mas eu sempre vou considerar defini-las pela tradução dos seus termos. HTML é uma Linguagem de Marcação Hipertexto. CSS são as Folhas de Estilo em Cascata. Notou que são termos em feminino? Pois as polêmicas não param por aqui não...



Você tem todo o direito de usar esse material para seu próprio aprendizado. Professores também podem ter acesso a todo o conteúdo e usá-lo com seus alunos. Porém todos o que usarem esse material - seja para qual for a finalidade - deverão manter a referência ao material original, criado pelo **Prof. Gustavo Guanabara** e disponível no endereço do seu repositório público <https://github.com/gustavoguanabara/>. Este material não poderá ser utilizado em nenhuma hipótese para ser replicada - integral ou parcialmente - por autores/editoras para criar livros ou apostilas, com finalidade de obter ganho financeiro com ele.

# “Eu programo em HTML”

Talvez você já tenha ouvido alguém falar a frase acima. Geralmente ela é dita por pessoas iniciantes ou então aqueles que se baseiam apenas na capa de um único livro da série **Head First** (use a cabeça), que estampa na capa “*programação em HTML5*”.

Acontece que a própria sigla já entrega seu objetivo: **Hypertext Markup Language** traduzido para o bom e velho Português significa **Linguagem de Marcação Hipertexto**. Você vai notar daqui a pouco que ela não funciona com instruções, como toda linguagem de programação. A HTML é baseada em **marcações** chamadas **tags**, e elas comandam tudo.



Além disso, o termo “**programação**” envolve estruturas especializadas que dependem do uso de **variáveis simples** e **compostas**, **condições**, **laços** e até coisas mais complexas como **objetos**. Nada disso existe na HTML nem nas CSS. Por outro lado, todas essas características estão presentes na linguagem **JavaScript**. Essa sim é uma **Linguagem de Programação**.

HTML trabalha fundamentada apenas nas **marcas** ou **etiquetas** (do Inglês **tag**) e as CSS funcionam baseadas nos **seletores**, **propriedades** e **valores**. Você vai entender mais sobre isso mais pra frente.

Sendo assim, em breve você terá a base suficiente para conseguir dizer com certeza que HTML e CSS não são linguagens de programação.



**IMPORTANTE!** Dizer que HTML e CSS não são linguagens de programação não as torna **RUINS** de maneira nenhuma! O que estamos fazendo aqui é uma simples classificação. Não há nada de errado com elas e nenhuma outra linguagem é melhor ou pior que elas por conta disso, ok?

## Para que serve HTML, CSS e JS?

Uma das coisas mais importantes para quem está começando o desenvolvimento de sites é compreender para que serve esse trio de tecnologias, que geralmente são estudados em conjunto. Basicamente, de forma resumida, temos um panorama simples:

HTML	Conteúdo
CSS	Estilo
JavaScript	Interatividade

Guarde bem a tabela anterior sempre que você precisar decidir qual linguagem vai utilizar em cada situação.

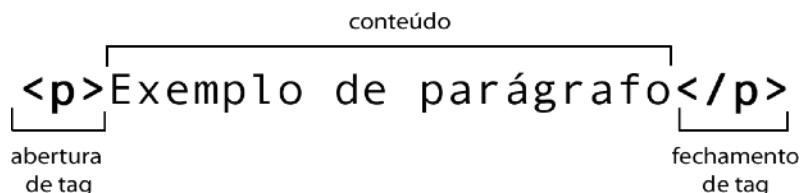
Abra aí o seu site de notícias favorito. Ao abrir uma determinada notícia, você vai ver o texto, as imagens, os vídeos e todo aquele conteúdo que compõe a notícia em si. Isso tudo foi criado em **HTML**. Ela é focada em **conteúdo**.

Agora preste atenção nas cores, na posição dos componentes e organização visual do conteúdo em colunas, blocos visuais e tudo mais. Tudo foi definido em **CSS**. Ela é focada no **design/estilo**.

Finalmente, provavelmente existe o menu do site. Quando você clica nele, acontece uma animação. Ao mover o mouse sobre as sessões, é possível que aconteçam algumas interações interessantes. Isso foi desenvolvido com ajuda de **JavaScript**. Ela é uma linguagem focada nas **interações**.

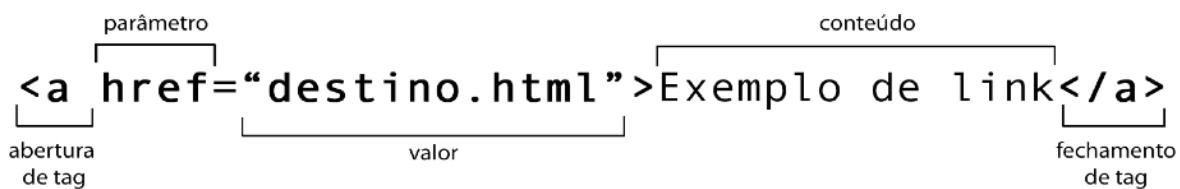
## Tags HTML, aí vamos nós

Como eu já disse anteriormente, a HTML funciona baseada em marcações específicas chamadas **tags**. Uma tag é um conjunto de palavras entre sinais de **colchete angular**, conforme representado a seguir.



Na imagem anterior, você consegue perceber o uso da tag <p> para a criação de um parágrafo simples. A maioria das tags possuem uma **abertura** e um **fechamento**, e você identifica isso pela presença da barra no fechamento da tag.

Além disso, as tags também podem ter atributos e valores, que vão configurar seu comportamento:



O banner tem uma barra superior branca com ícones de minimização, maximização e fechar. O fundo é escuro com formas geométricas. No topo, o texto "Cursos que vão te levar ao próximo nível". Abaixo, o logo "estudonauta" com uma figura de óculos. À direita, há um QR code.

Uma mesma tag pode ter vários parâmetros, cada um com seu valor. Entretanto, algumas tags não possuem a necessidade de **conteúdo** interno e por isso não possuem fechamento. É o caso, por exemplo, das tags `<br>` e `<img>`. Isso é algo natural, não se preocupe com isso agora.

## Eu ainda uso `<font>`, `<center>`, `<s>`, `<u>`, ...

Com o surgimento da versão 5 da HTML, algumas tags simplesmente deixaram de existir ou tornaram-se **obsoletas**. Uma tag obsoleta pode até estar funcionando no seu navegador hoje em dia, mas a própria **W3C** - consórcio responsável por manter as especificações da linguagem - recomenda que elas **não sejam mais usadas** pelos profissionais e aos poucos não serão mais suportadas pelos navegadores nas suas futuras versões.



**APRENDA MAIS:** Sempre que você quiser saber quais são as tags que estão sendo consideradas obsoletas pelo **W3C**, basta consultar a referência oficial da linguagem, disponível no site abaixo e acessar o **item 15: Obsolete Features**.

HTML Standard: <https://html.spec.whatwg.org/multipage/>

De forma simples e direta (vou até escrever “gritando” aqui, pra dar ênfase): **NÃO USE TAGS OBSOLETAS NO SEU SITE!** Ufa! Que alívio colocar isso pra fora e desabafar 😊

## Chegou a vez dos seletores CSS

Como já vimos anteriormente, as **CSS** são as **Cascading Style Sheets** (Folhas de Estilo em Cascata). Elas são usadas para configurar um **resultado visual** dos elementos HTML.

As configurações das CSS são realizadas através dos **seletores**. Vamos ver a anatomia de um seletor.

```
seletor
[ p {           declaração
    font-family: Arial;
    font-size: 12pt;
    color: blue;
}
propriedade   valor
```

O seletor apresentado anteriormente vai configurar o visual dos elementos de parágrafo do site corrente. O uso das chaves delimita todas as declarações relativas ao seletor atual. No seletor que eu te mostrei, serão feitas três configurações:

- A fonte escolhida foi *Arial*.
- O tamanho da letra será *12pt* (pontos).
- A cor da letra será *azul*.

Note que, ao final de cada **declaração**, temos que colocar ponto-e-vírgula para indicar que ela se encerrou.

Todas as **propriedades** devem ter seu **valor**, e eles devem ser separados por dois pontos. Você não é obrigado(a) a usar nenhuma declaração específica. Só utilize a propriedade que você realmente deseja alterar.

## Estrutura básica de um documento HTML

Ao criar um novo documento HTML, devemos sempre escrever a estrutura básica de um documento desse formato. Vamos analisar cada uma das 11 linhas que compõem esse documento base.

```
1  <!DOCTYPE html>
2  <html lang="pt-br">
3  <head>
4  |   <meta charset="UTF-8">
5  |   <meta name="viewport" content="width=device-width,
6  |   initial-scale=1.0">
7  |   <title>Document</title>
8  </head>
9
10 <body>
11 </html>
```

Cursos grátis de tecnologia  
que te preparam para o  
mercado de trabalho

RECODE



- **Linha 1:** Indica que o documento atual será escrito na versão mais atualizada da linguagem (no caso, HTML5)
- **Linhas 2 e 11:** Delimitam o documento HTML, que é sempre dividido em duas partes: a cabeça e o corpo. Na linha 2, também estamos indicando que o conteúdo desse site será no idioma Português do Brasil.
- **Linhas 3 e 7:** Delimitam a **cabeça** da página, local onde são realizadas algumas configurações iniciais como formatos, estilos, ícone de favoritos, etc.
- **Linha 4:** adiciona ao documento atual o suporte a caracteres acentuados. Remover essa linha pode causar erros de renderização de algumas letras na tela.
- **Linha 5:** Indica que o conteúdo aparecerá, por padrão, ocupando todo o espaço disponível da tela e com uma escala de 1:1.
- **Linha 6:** Configura o título da página, que aparecerá como identificação da aba do navegador, ao lado do *favicon*.
- **Linhas 8 e 10:** Delimitam o **corpo da página**, a maior porção do site, que vai aparecer na tela. É aqui onde colocaremos todo o nosso conteúdo.

## Por enquanto é isso!

Agora você já conhece a base suficiente para começar a criar seus próprios sites básicos. No próximo material vamos instalar os softwares necessários para a criação de documentos e começar a aprender as principais tags. Complemente sempre o nosso conteúdo com os vídeos que eu indico no final de cada material.

## Quer acompanhar tudo em vídeo?

Eu sei que às vezes as pessoas gostam mais de assistir vídeos do que ler livros, e é por isso que eu lanço há anos materiais no canal *Curso em Vídeo* no YouTube. O link que vou compartilhar contigo faz parte da playlist completa onde você encontra o **Módulo 1 do Curso de HTML5 e CSS3**, completamente gravado com base nesse material.



Além de acessar o link a seguir, você também pode ter acesso às aulas apontando a câmera do seu celular para o código QR ao lado. Todo dispositivo smartphone ou tablet atualizado já possui esse recurso de leitura de códigos habilitado por padrão.

Módulo 1 do curso: [https://www.youtube.com/playlist?list=PLHz\\_AreHm4dkZ9-atkcmcBaMZdmLHft8n](https://www.youtube.com/playlist?list=PLHz_AreHm4dkZ9-atkcmcBaMZdmLHft8n)

# Teste seus conhecimentos

Terminou de ler esse capítulo e já acompanhou todos os vídeos e referências externas que indicamos? Pois agora, responda a essas 10 perguntas objetivas e marque em cada uma delas a única opção verdadeira. Aí sim, você vai poder comprovar que realmente entendeu o conteúdo.



1. Qual das frases a seguir é a única tecnicamente CORRETA de se falar?

- A "Eu programo em linguagem HTML"
- B "Eu programo em linguagem CSS"
- C "Eu programo em linguagem JavaScript"
- D "Eu programo em linguagem VSCode"

2. A sigla HTML significa:

- A Host Text Makeup Language
- B HyperText Markup Language
- C Hyper Tree Makeup Language
- D Host Tree Markup Language

3. A sigla CSS significa:

- A Cascading Style Sheets
- B Cell Safety Science
- C Characteristic Score Style
- D Chief Scale Sheets

4. Correlacione a coluna da esquerda com a da direita, de acordo com o foco de cada uma das tecnologias:

( 1 ) HTML	(   ) interatividade
( 2 ) CSS	(   ) conteúdo
( 3 ) JS	(   ) estilo

- A 1 - 2 - 3
- B 3 - 2 - 1
- C 1 - 3 - 2
- D 3 - 1 - 2

5. Qual tag abaixo não tem fechamento?

- A <title>
- B <meta>
- C <strong>
- D <head>

6. Na tag <a>, o href é um(a):

- A conteúdo
- B parâmetro
- C característica
- D valor

7. Todas as configurações visuais dos elementos HTML são realizadas pela linguagem CSS. Agrupamos todas as declarações CSS de um mesmo elemento dentro de um(a):

- A seletor
- B parâmetro
- C valor
- D selecionador

8. Para mudar a cor de um texto em CSS, configuramos qual propriedade?

- A text
- B text-color
- C color
- D font-color

Soluções digitais para negócios

 hostnet



9. Para indicar que um determinado documento HTML está escrito na versão mais recente da linguagem, devemos adicionar a seguinte instrução:

- [A] <html lang="version5">
- [B] <title>HTML5</title>
- [C] <meta name="lang" type="html5">
- [D] <!DOCTYPE html>

10. Qual é a tag de um documento HTML adicionada pra manter a compatibilidade com os caracteres acentuados, muito comuns na língua Portuguesa?

- [A] <html lang="pt-br">
- [B] <meta charset="UTF-8">
- [C] <body lang="br">
- [D] <head charset="UTF-8">

# **Suas anotações**

*Não guarde conhecimento. Ele é livre. Compartilhe o seu e veja ele se espalhando pelo mundo* 

# DevWeb

## Capítulo 04

# Primeiros Passos com HTML5

Pronto! Já vai começar a codificação que você está esperando tanto! Chegou a hora de colocar a mão na massa e criar os primeiros documentos HTML. E faremos do jeito tradicional: devagar e sempre. Ao utilizar esse método que eu prefiro aplicar às minhas aulas, veremos exercícios bem simples, com a aplicação de cada tag nova isoladamente. Tem gente que acha melhor começar com um projeto grande, mas em sala eu opto sempre por esse método. Vamos lá?



Você tem todo o direito de usar esse material para seu próprio aprendizado. Professores também podem ter acesso a todo o conteúdo e usá-lo com seus alunos. Porém todos que usarem esse material - seja para qual for a finalidade - deverão manter a referência ao material original, criado pelo **Prof. Gustavo Guanabara** e disponível no endereço do seu repositório público <https://github.com/gustavoguanabara/>. Este material não poderá ser utilizado em nenhuma hipótese para ser replicada - integral ou parcialmente - por autores/editoras para criar livros ou apostilas, com finalidade de obter ganho financeiro com ele.



# Escolhendo um editor

Se você começar a frequentar grupos e fóruns de desenvolvedores, vai ver que tem muita gente perdendo um tempo precioso com discussões inúteis. Uma delas é: "qual é o melhor editor de códigos?" .

Na minha humilde opinião, o melhor editor de código é aquele que te atende! Geralmente as pessoas gostam mais do editor no qual elas começaram a desenvolver. Tem até gente que defende o uso de editores extremamente simples, como o Notepad (Windows) e o VIM (Linux). Se é o editor que você gosta e aprendeu a usar, continue nele!



Na imagem acima, coloquei alguns exemplos de editores de código bem famosos: na ordem, temos o **Visual Studio Code**, o **Atom**, o **Sublime**, o **Brackets** e o **Notepad++**. Todos são **ÓTIMOS** editores, cada um com suas vantagens e desvantagens. Nenhum deles é perfeito e todos podem travar.

Para construir o **curso de HTML5** que você encontra no YouTube, utilizei os editores **Notepad++** e o **WebStorm** da JetBrains. Naquela época eu usava muito esses dois. Hoje em dia, me adaptei 100% ao **Visual Studio Code**. Por conta disso, usaremos esse editor, que foi criado pela Microsoft, é distribuído gratuitamente e funciona no **Windows, Linux e Mac**.



**INSTALE O VS CODE:** Não adianta ficar apenas lendo esse material ou assistindo os vídeos. A instalação das ferramentas que usaremos no curso é **ESSENCIAL** para o seu aprendizado. Instale e use todos os dias! Crie muito!

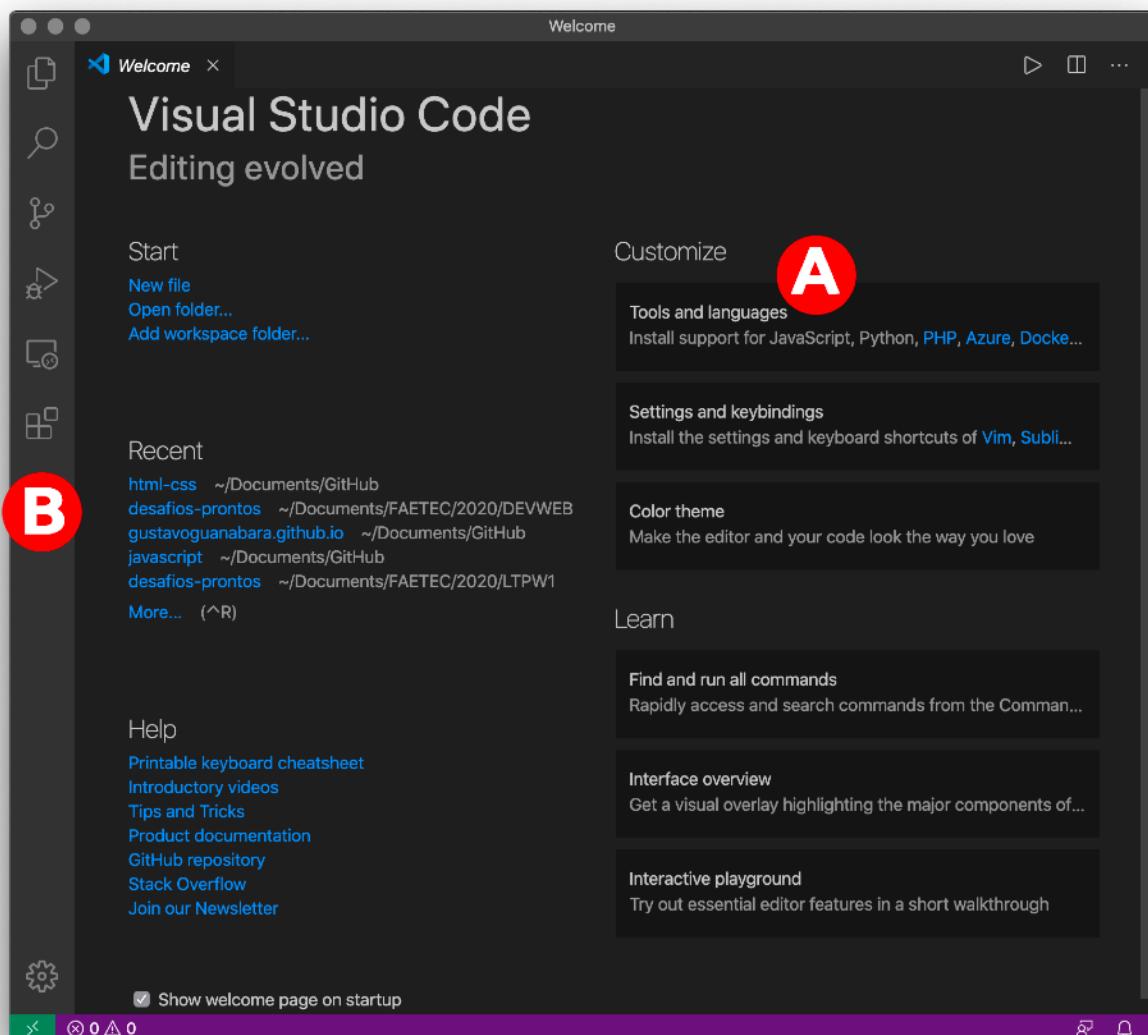
Baixe o editor em <https://code.visualstudio.com/#alt-downloads>

## Dicas básicas para configurar seu VS Code

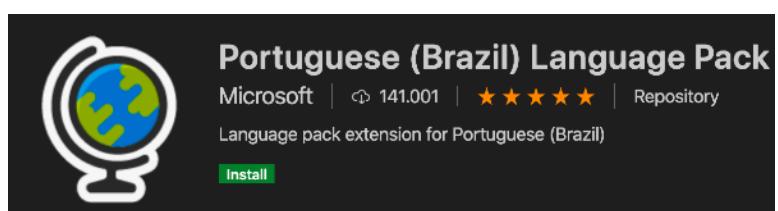
Depois de instalar o VS code, ele vai abrir no seu modo escuro (padrão) e recomendamos que você faça algumas configurações básicas para facilitar o seu dia-a-dia de desenvolvimento de sites.

Em primeiro lugar, vamos habilitar o suporte estendido para a linguagem JavaScript, que usaremos no nosso curso de JS. Na imagem que está a seguir, na tela de **Welcome** do seu editor, clique em "Install support for JavaScript" (**A** na imagem) na

seção “Tools and languages”. Em seguida, vamos acessar a **Toolbar** (**B** na imagem) e clicar no último botão (o de baixo), que permite instalar **Extensões** (Extensions).



Na tela de extensões, procure por “Portuguese” e escolha a extensão que está representada ao lado. Clique na opção **Install** e, ao final da instalação, o VS code vai solicitar que você reinicie o editor. Pronto! Seu code estará em Português!



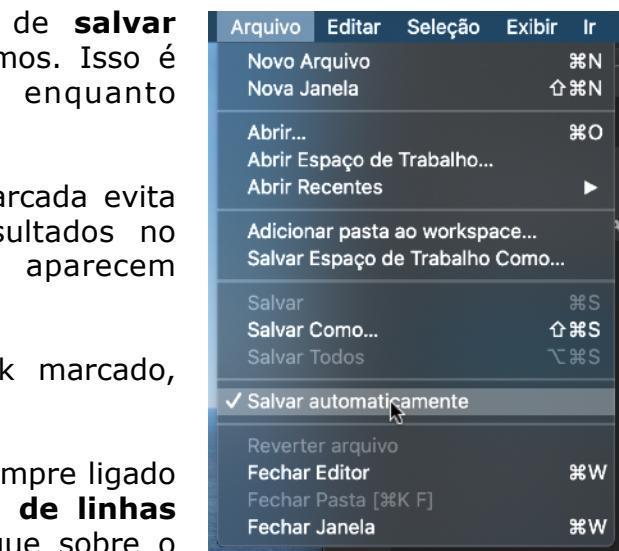
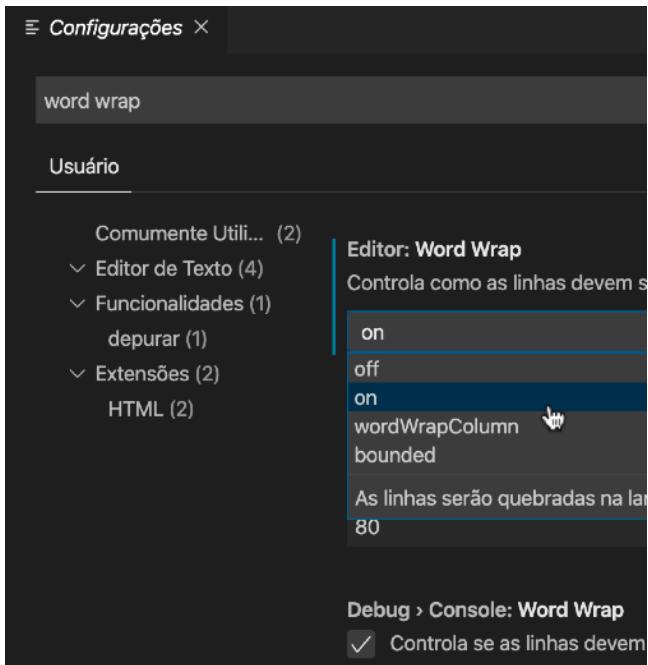
**IMPORTANTE:** Eu sei que tem gente que vai preferir manter o VS code em Inglês. Isso também é mais uma questão de opção pessoal. Nesse curso, utilizarei a versão em Português para eliminar possíveis barreiras de idioma, mas você é livre para optar pelo idioma original.

Em seguida, vamos habilitar o recurso de **salvar automaticamente** tudo aquilo que digitamos. Isso é muito importante para ganhar tempo enquanto desenvolvemos nossos códigos.

Além disso, manter essa opção sempre marcada evita problemas quando queremos ver os resultados no navegador, mas eles simplesmente não aparecem porque esquecemos de salvar.

Deixe sempre essa opção com um check marcado, exatamente como está aparecendo ao lado.

Outra coisa muito importante para deixar sempre ligado é a opção de **quebra automática de linhas** (Word Wrap). Para fazer isso, clique sobre o botão de opções que fica no canto inferior esquerdo do VS code (veja o ícone à esquerda) e escolha a opção **Configurações**.



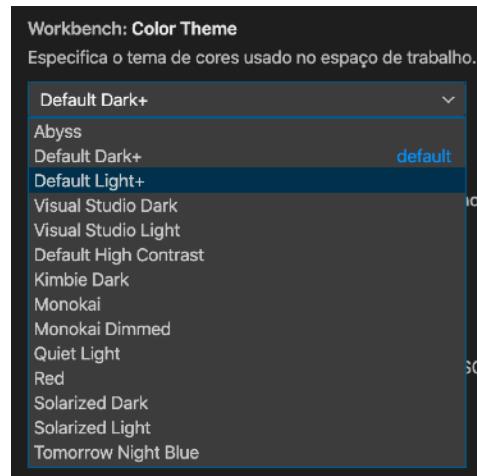
Quando abrir a tela de configurações, digite "word wrap" (sem as aspas) e modifique a opção **Editor: Word Wrap**, colocando o valor **on** na caixa suspensa de seleção (veja a imagem ao lado).

Habilitar essa opção, vai melhorar a visualização do seu código, já que você não vai precisar fazer a rolagem lateral da tela em situações onde a linha vai ficar muito comprida. Pode acreditar, escreveremos algumas linhas bem longas mesmo.

Outra configuração que você pode alterar caso ache necessário é o **Editor: Font Size**, que vai deixar aumentar ou diminuir o tamanho do código do editor.

## Modo Escuro ou Claro?

Tá aí mais um motivo para discussão inútil em comunidades em redes sociais. Particularmente, prefiro o modo escuro para trabalhar no dia-a-dia. O contraste entre as letras e o fundo escuro facilitam a minha leitura, já que sou meio idoso e preciso cuidar da minha saúde visual 😊

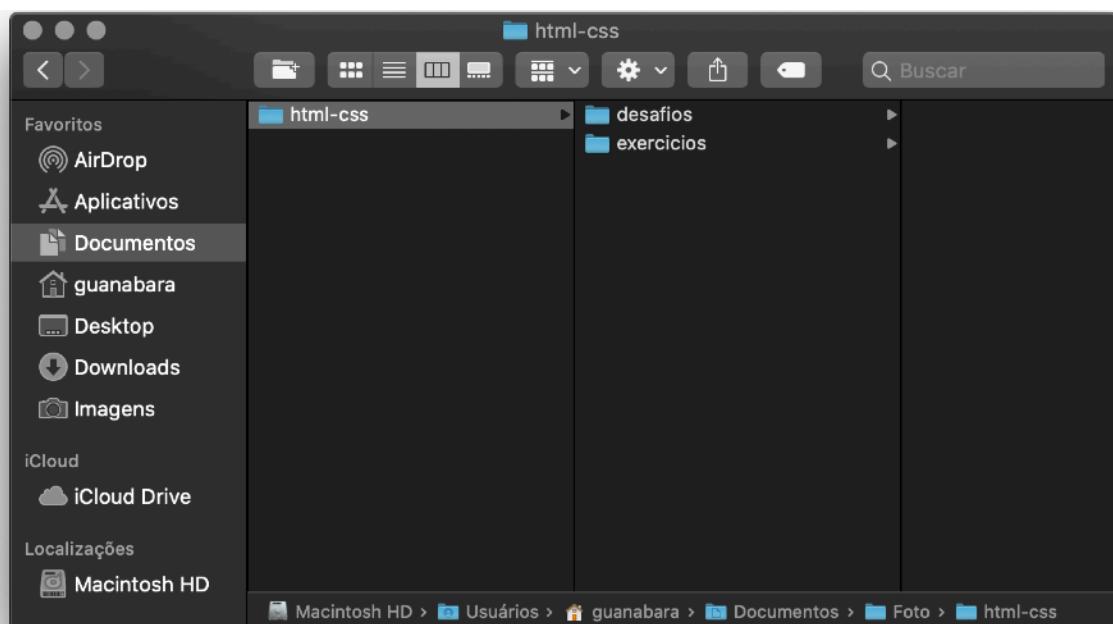


Mas como tudo aqui no curso, você tem total liberdade para modificar as configurações e deixar seu VS code da cor que quiser (até vermelho, mas não recomendo). Para isso, procure pela opção **Workbench: Color Theme** na sua área de configurações.

## Projetos em pastas

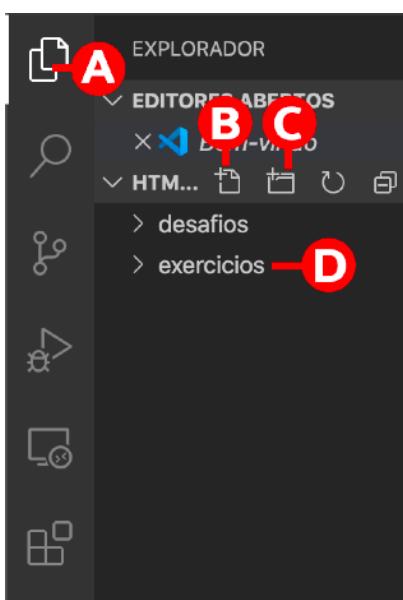
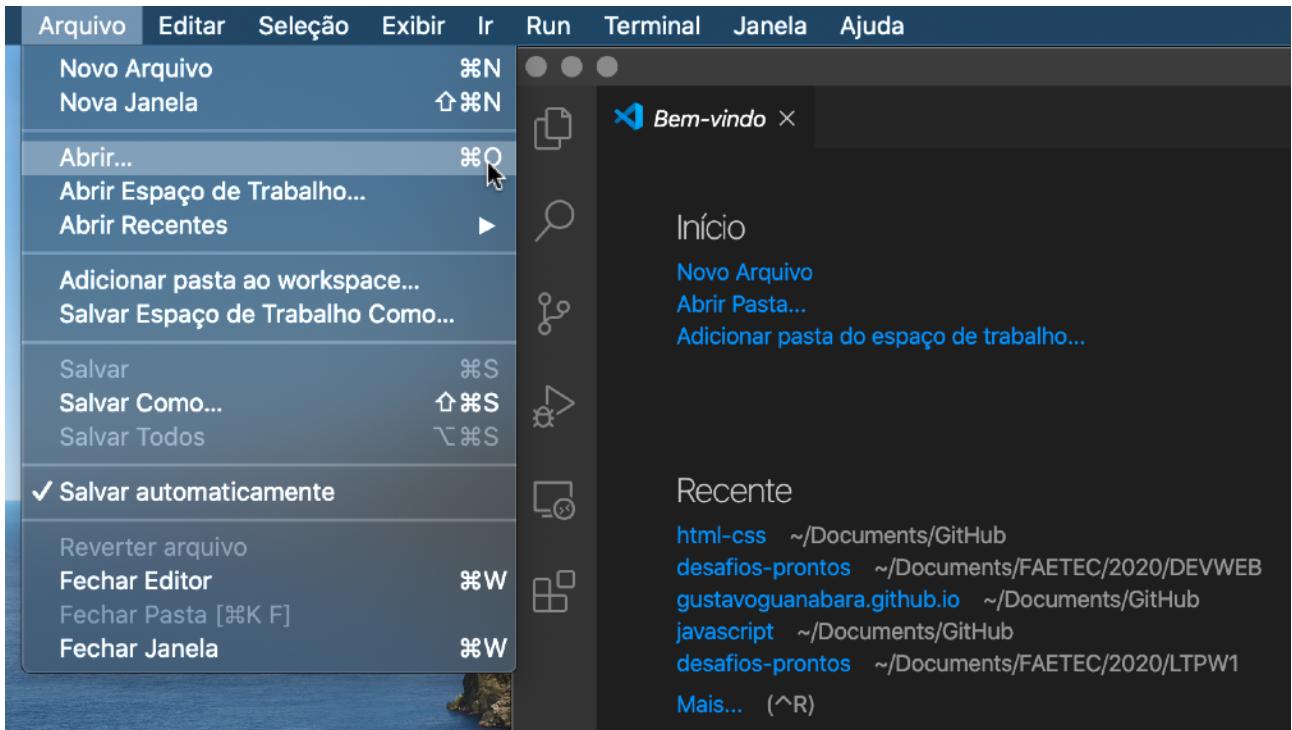
Se eu puder te dar um conselho - e eu me sinto nesse direito - ele conselho seria: **ORGANIZE-SE!** Quem começa a desenvolver coisas e joga tudo na **Área de Trabalho** do Windows ou dentro da pasta **Meus Documentos** do seu computador, em pouco tempo fica simplesmente perdido(a).

Para isso, vou te dar uma dica valiosa. Vá até a sua pasta de Documentos e crie uma pasta chamada **html-css**. Essa será a sua pasta principal do seu curso. Dentro dessa pasta, crie duas pastas: **desafios** e **exercícios** (note que eu não usei acentuações, nem espaços, nem letras maiúsculas).



Agora vamos abrir o VS code e abrir essa pasta que acabamos de criar. Se você ainda está com a tela de Bem-vindo (Welcome) aberta, basta clicar no link **Abrir Pasta...** que está na seção **Início** ao lado esquerdo (veja na foto) e escolher a pasta **html-css** que criamos anteriormente.

Além disso, você também pode clicar no menu do editor e escolher a opção **Abrir** (tecla de atalho Ctrl+O no teclado) e escolher a pasta.



Depois que você abrir a pasta, ela vai aparecer no lado esquerdo da tela, no **Explorador (A)**. Note que as duas pastas que criamos, **desafios** e **exercícios** também está aparecendo.

Vamos criar nosso primeiro exercício de maneira organizada. Clique primeiro no nome da pasta (**D**) para abri-la. Em seguida clique no botão para criar uma nova pasta (**C**) e dê o nome **ex001** para a pasta criada.

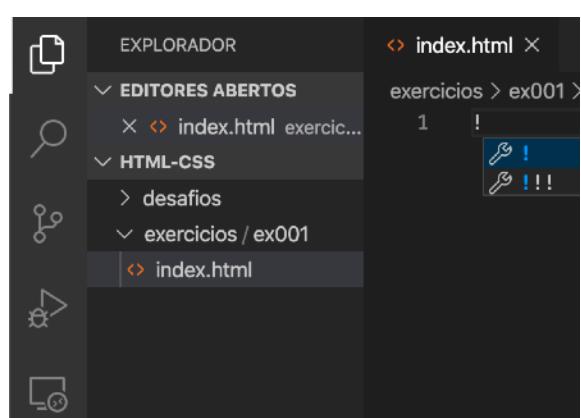
Depois clique na pasta que você acabou de criar e clique no botão para criar um novo arquivo (**B**) e dê o nome de **index.html** para o arquivo criado.

**IMPORTANTE:** Não se esqueça de usar apenas letras minúsculas, não usar acentos e não colocar espaços.

Feitos todos esses passos, a sua tela deve se parecer com a imagem à direita.

Note que o arquivo que foi criado já aparece aberto na área principal do editor, e o arquivo com extensão **.html** vai aparecer com um pequeno ícone **<>**, indicando que foi criado no formato correto.

Para criar o código base HTML que vimos no capítulo anterior, vá até as linhas do arquivo **index.html** e digite apenas um ponto de exclamação **!** e em seguida pressione a tecla **Enter** no seu teclado.



Viu só o resultado? O código foi criado automaticamente e precisa apenas de algumas pequenas atualizações. Complete seu código, adicionando as linhas para deixar o código igual ao que está representado abaixo.

```
1 <!DOCTYPE html>
2 <html lang="pt-br">
3 <head>
4     <meta charset="UTF-8">
5     <meta name="viewport"
6         content="width=device-width, initial-scale=1.0">
7     <title>Primeiro site</title>
8 </head>
9 <body>
10    <h1>Olá, Mundo!</h1>
11    <hr>
12    <p>Este foi o primeiro site que eu criei, com um
13        título, uma linha e um parágrafo.</p>
14 </body>
15 </html>
```

A **linha 2** foi alterada, indicando que nossa página terá seu conteúdo em Português Brasileiro. A **linha 6** também, para que nosso título seja *Primeiro site*.

Na **linha 9**, criamos um **título**. Nesse momento, vou te ensinar uma dica. No lugar de digitar `<h1></h1>` manualmente, digite apenas `h1`, sem os colchetes angulares e pressione Enter (veja a imagem). Automaticamente a tag será aberta e fechada. Isso facilita muito o nosso dia-a-dia também.

```
<body>
    h1
</body> ↵ h1
</html>
```

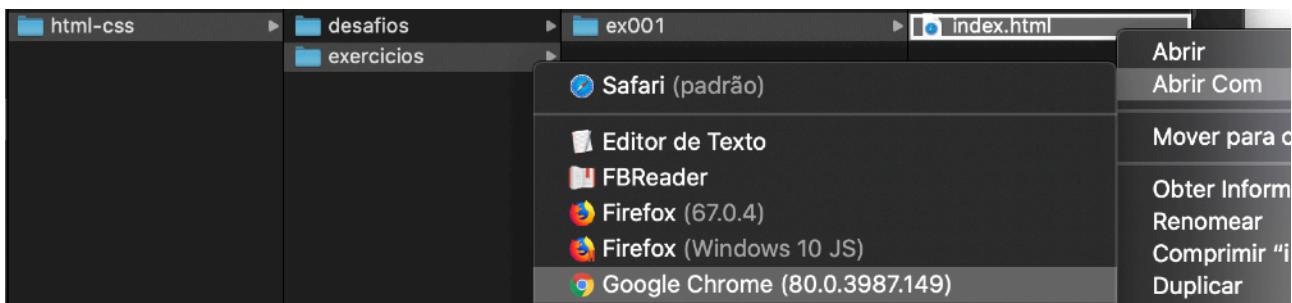
Na **linha 10** criamos uma **linha horizontal** (horizontal row). Perceba que não existe fechamento para essa tag. Isso é normal e nós já tínhamos previsto isso no material anterior.

Na **linha 11** estamos criando um **parágrafo** com um texto de conteúdo. Note que na foto, a linha ocupa duas linhas, mas a contagem se mantém no 11. Isso acontece porque eu habilitei o **Word Wrap** na página 4 desse material.

Se você também habilitou a opção **Salvar automaticamente** (página 4) no seu VS code, tudo aquilo que você digitou já está gravado na sua pasta.

O próximo passo é ver se está tudo funcionando corretamente.

# Chegou a hora de testar tudo



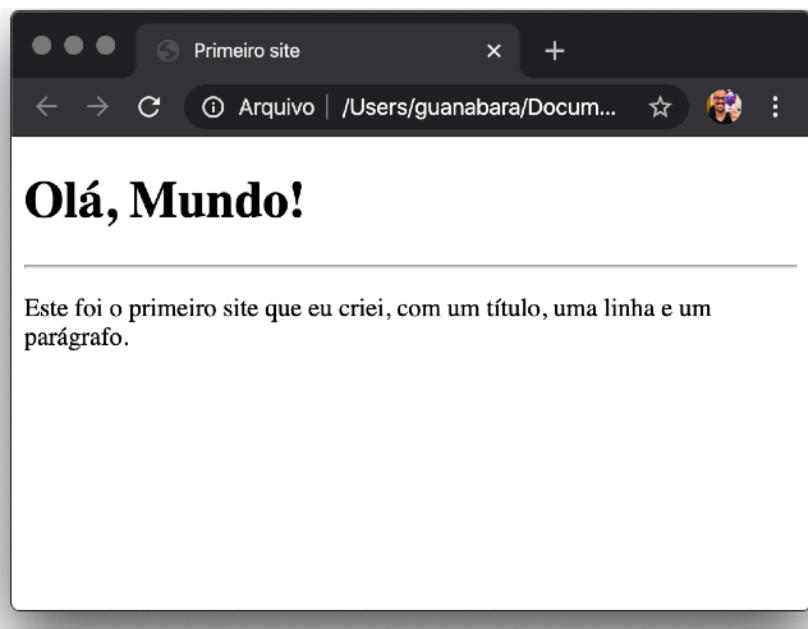
Se você se organizou direitinho e salvou o arquivo no local certo, vai poder abrir no seu explorador de arquivos a pasta Documentos/html-css/exercícios/ex001 e lá dentro vai encontrar o arquivo index.html. Clique com o botão direito sobre esse arquivo html e escolher a opção para **Abrir no Google Chrome**.

O resultado visual deverá ser semelhante ao apresentado ao lado.

Perceba na parte de cima do navegador, lá na aba do documento, aparecerá o termo **Primeiro site**. Isso foi o efeito da configuração do <title>.

Note que o uso da tag <h1> fez o seu conteúdo virar um título.

Já a tag <hr> fez aparecer uma linha horizontal, como já tínhamos previsto. Logo abaixo está o parágrafo.



## Eu já falei sobre isso no YouTube?

Eu sei que às vezes as pessoas gostam mais de assistir vídeos do que ler livros, e é por isso que eu lanço há anos materiais no canal Curso em Vídeo no YouTube. O link que vou compartilhar contigo tem o conteúdo explicado como você leu aqui, só que de forma mais ilustrada. Reserve um tempo dos seus estudos para assistir esse vídeo todo.



Curso em Vídeo: [https://www.youtube.com/playlist?list=PLHz\\_AreHm4dlAnJ\\_jJtV29RFxnPHDuk9o](https://www.youtube.com/playlist?list=PLHz_AreHm4dlAnJ_jJtV29RFxnPHDuk9o)

# Teste seus conhecimentos

Terminou de ler esse capítulo e já acompanhou todos os vídeos e referências externas que indicamos? Pois agora, responda a essas 10 perguntas objetivas e marque em cada uma delas a única opção verdadeira. Aí sim, você vai poder comprovar que realmente entendeu o conteúdo.



1. Qual dos programas a seguir é o único que não pode ser utilizado para escrever códigos em texto sem formatação para HTML + CSS + JS?

- A Atom
- B VIM
- C Brackets
- D Word

2. Qual é a empresa que fabrica o VSCode, o editor que escolhemos para escrever códigos para nosso curso?

- A Microsoft
- B Apple
- C Adobe
- D Google

3. É possível usar o VSCode em idioma Português do Brasil?

- A Sim. O editor já é instalado em PT-BR como padrão
- B Sim. Porém, precisamos escolher o idioma durante a instalação
- C Sim. Mas não existe suporte nativo para PT-BR, mas existe uma extensão para isso.
- D Não. O VSCode ainda não funciona em PT-BR.

4. Qual é a opção que devemos habilitar nas configurações para que as instruções muito longas sejam quebradas em várias linhas automaticamente?

- A Wrap Line
- B Multi Line
- C Auto Break
- D Word Wrap

5. Qual é o caminho correto que devemos seguir para ligar o salvamento automático de arquivos no VSCode?

- A Arquivo > Salvar Automaticamente

- B Terminal > Salvar Automaticamente
- C Configurações > Salvar Automaticamente
- D Editar > Salvar Automaticamente

6. Qual é o nome do arquivo principal de um site feito em HTML, que será aberto assim que o visitante solicitar a abertura do site?

- A first.html
- B www.html
- C index.html
- D 001.html

7. Ao criar um arquivo HTML, o VSCode tem um atalho simples para criar o código base para um arquivo desse tipo. Que atalho é esse?

- A ! + Enter
- B # + Enter
- C ^ + Enter
- D ? + Enter

8. Qual é a tag HTML para criar uma linha horizontal no corpo do navegador?

- A <line>
- B <row>
- C <!-->
- D <hr>

9. Para criar um parágrafo em HTML, usamos a tag:

- A <p>
- B <para>
- C <paragraph>
- D <h1>

10. A tag <title> é adicionada dentro de qual área de um documento HTML?

- A <h1>
- B <p>
- C <head>
- D <body>

# **Suas anotações**

*Não guarde conhecimento. Ele é livre. Compartilhe o seu e veja ele se espalhando pelo mundo* 

# DevWeb

## Capítulo 05

# Caracteres, parágrafos e quebras de linha

Eu sei que sites podem ter imagens, vídeos, animações e tudo mais. Mas o foco principal sempre é no texto em si. Palavras são valiosas. Sem palavras você não estaria nem lendo essa página. Então vamos dar o devido valor ao conteúdo escrito e ver as marcações importantes para os textos. Me acompanha?



Você tem todo o direito de usar esse material para seu próprio aprendizado. Professores também podem ter acesso a todo o conteúdo e usá-lo com seus alunos. Porém todos que usarem esse material - seja para qual for a finalidade - deverão manter a referência ao material original, criado pelo **Prof. Gustavo Guanabara** e disponível no endereço do seu repositório público <https://github.com/gustavoguanabara/>. Este material não poderá ser utilizado em nenhuma hipótese para ser replicada - integral ou parcialmente - por autores/editoras para criar livros ou apostilas, com finalidades de obter ganho financeiro com ele.



# Vamos começar com os parágrafos

No capítulo anterior já vimos como criar um parágrafo de forma simples, foi só colocar todo o conteúdo dentro da tag `<p></p>`. Mas o que eu não comentei na ocasião foi o comportamento do navegador em relação à posição das palavras.

Crie aí na sua casa uma pasta **ex002** dentro da sua pasta que criamos em Documentos/html-css/exercícios para deixar tudo organizando. Lá dentro, crie um documento **index.html** e crie o código base do HTML5 (já ensinei isso anteriormente).

Dentro da tag `<body></body>` crie um parágrafo com a seguinte característica:

```
<p>Você pode  
    escrever um  
    parágrafo  
    de qualquer jeito.  
    Basta colocar tudo  
    no meio do par de tags &lt;p&gt; e &lt;/p&gt;  
</p>  
  
<p>Se precisar quebrar  
    o texto em algum lugar  
    específico <br> como esse,  
    você pode usar a tag &lt;br&gt;  
</p>
```

O que nós criamos no código aí acima foram dois parágrafos. Note que eu criei várias quebras de linha dentro de cada parágrafo. Salve o seu projeto e veja o resultado.

Na verdade, todas aquelas quebras que causamos não servem de **ABSOLUTAMENTE NADA!** Isso acontece porque quem comanda quebras de linhas/parágrafos não é o fato de apertar **Enter**. Quem manda são as tags HTML. A única quebra que **REALMENTE** aconteceu foi após a palavra "*específico*" no segundo parágrafo. Isso porque colocamos a tag `<br>` nesse local.

A tag `<br>` significa literalmente "quebre a linha" (*break row*) e agora que você sabe o significado, não precisa nem de explicação né?



**IMPORTANTE:** Uma das **GAMBIARRAS** que muitos iniciantes em HTML fazem constantemente é criar espaços verticais maiores usando `<br><br><br><br><br>` várias vezes. Ai vai um grande conselho: **NÃO FAÇA ISSO!** Mais tarde vamos aprender como configurar o espaçamento vertical através das folhas de estilo (CSS). Quem cria espaço com vários `<br>` é NOOB! Não faça!

Outra coisa que você deve ter notado é que tivemos problemas para fazer uma tag aparecer no seu navegador. Isso acontece porque, se você colocar <br> no seu código, o navegador vai entender como “quebre a linha” e não pra fazer ela aparecer.

Para isso, no lugar de usar os colchetes angulares <>, usamos caracteres especiais referentes ao código desses símbolos. No código anterior, quando usamos:

- Um &lt; estamos pedindo pra colocar o símbolo menor que (less than).
- Um &gt; estamos pedindo para coloca o símbolo maior que (greater than).

Existem alguns outros símbolos que podem ser exibidos usando códigos. Esses códigos são chamados de **HTML Entities**, basta uma breve consulta para descobrir quais são as opções, mas vou te dar uma moral e colocar uma tabela com outras opções.

Símbolo	Descrição	Entitiey
	<i>Espaço em branco</i>	&ampnbsp
®	<i>Marca registrada</i>	&reg;
©	<i>Copyright</i>	&copy;
™	<i>Trade Mark</i>	&trade;
€	<i>Euro</i>	&euro;
£	<i>Libra (pound)</i>	&pound;
¥	<i>Yen</i>	&yen;
¢	<i>Cent</i>	&cent;
Ø	<i>Vazio</i>	&empty;
Σ	<i>Soma</i>	&sum;
Δ	<i>Delta</i>	&Delta;
←	<i>Seta esquerda</i>	&larr;
↑	<i>Seta acima</i>	&uarr;
→	<i>Seta direita</i>	&rarr;
↓	<i>Seta baixo</i>	&darr;



**DICA:** Você quer uma referência com vários símbolos e seus códigos, pois acesse o W3Schools, na área de Misc Symbols.

[https://www.w3schools.com/charsets/ref\\_utf\\_symbols.asp](https://www.w3schools.com/charsets/ref_utf_symbols.asp)

# Inserindo Emojis

Outra coisa que você pode adicionar às suas páginas são os emojis, onde cada símbolo possui um código Unicode. Para uma pesquisa mais precisa e atualizada, acesse o site da [emojipedia.org](http://emojipedia.org).

The screenshot shows a web browser window with the title bar "Emoji People and Smiley Me" and the URL "emojipedia.org/people/". The main content area is titled "Smileys & People" with the subtitle "Emojis for smileys, people, families, hand gestures, clothing and accessories." Below this, a list of emojis is displayed with their names: 😃 Grinning Face, 😄 Grinning Face with Big Eyes, 😆 Grinning Face with Smiling Eyes, 😇 Beaming Face with Smiling Eyes, 😅 Grinning Squinting Face, 😆 Grinning Face with Sweat, 😂 Rolling on the Floor Laughing, 😈 Face with Tears of Joy, 😃 Slightly Smiling Face, and 😊 Upside-Down Face. At the bottom of the page is the URL "https://emojipedia.org/grinning-face-with-smiling-eyes/".

The screenshot shows a web browser window with the title bar "Smiling Face with Open Mouth" and the URL "emojipedia.org/grinning-face-with-smiling-eyes/". The main content area includes sections for "Codepoints" (showing 😃 U+1F604), "Shortcodes" (showing :smile:), and "See also" (listing various other smiling face emojis like 😎, 😗, 😃, etc.). At the bottom of the page is the URL "https://emojipedia.org/emoji/ 😃 / smiling\_Eyes".

Escolha uma categoria, clique no emoji escolhido e procure o seu **Codepoints**. No caso do exemplo acima, escolhemos o emoji com o código **U+1F604**. Para inserir esse emoji no seu site, use o código adaptado **&#x1F604** em qualquer lugar que aceite palavras. É só substituir o **U+** por **&#x** (com a letra x minúscula).

# E começam os desafios!

Lá no repositório, além do material em PDF e dos códigos dos exercícios 100% disponíveis, também disponibilizamos alguns **desafios** que devem ser resolvidos. Esses desafios não incluem o código original e você deve tentar chegar à resposta sem copiar nenhum código.

Com todo o conteúdo que vimos até essa aula, você já pode resolver o **desafio d001**. Acesse o repositório público, abra a área do curso de HTML+CSS e clique no link de acesso aos desafios. Manda ver! Só não fica pedindo a resposta! Você consegue resolver isso sozinho(a)!

Repositório em: <https://gustavoguanabara.github.io>



## Eu já falei sobre isso no YouTube?

Eu sei que às vezes as pessoas gostam mais de assistir vídeos do que ler livros, e é por isso que eu lanço há anos materiais no canal Curso em Vídeo no YouTube. O link que vou compartilhar contigo tem o conteúdo explicado como você leu aqui, só que de forma mais ilustrada. Reserve um tempo dos seus estudos para assistir esse vídeo todo.



Curso em Vídeo: [https://www.youtube.com/playlist?list=PLHz\\_AreHm4dlAnJ\\_jJtV29RFxnPHDuk9o](https://www.youtube.com/playlist?list=PLHz_AreHm4dlAnJ_jJtV29RFxnPHDuk9o)

# DevWeb

## Capítulo 06

# Imagens e Favicon

Sabe aquela frase clássica que diz que “uma imagem vale mais do que mil palavras”? Pois é exatamente assim com a maioria dos sites. Tirando algumas pequenas exceções onde o público acessa um site apenas para “ler” seu conteúdo, as imagens são essenciais para a maioria dos casos. São elas que chamam a atenção e devem ser escolhidas com muita dedicação. Vamos nessa?



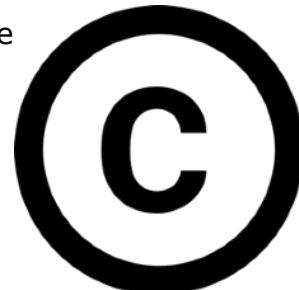
Você tem todo o direito de usar esse material para seu próprio aprendizado. Professores também podem ter acesso a todo o conteúdo e usá-lo com seus alunos. Porém todos os que usarem esse material - seja para qual for a finalidade - deverão manter a referência ao material original, criado pelo **Prof. Gustavo Guanabara** e disponível no endereço do seu repositório público <https://github.com/gustavoguanabara/>. Este material não poderá ser utilizado em nenhuma hipótese para ser replicada - integral ou parcialmente - por autores/editoras para criar livros ou apostilas, com finalidade de obter ganho financeiro com ele.



# Pra começar, vamos falar de direitos autorais

Sabe aquela foto que você pegou lá no **Google Imagens** e se encaixou perfeitamente no seu site? Pois é bastante provável que você simplesmente **NÃO POSSA USÁ-LA** e pode até mesmo ser processado(a) por isso.

" - *Mas como assim, Guanabara!?*" - você pode estar gritando, mesmo que apenas mentalmente, nesse exato momento.



É exatamente o que eu falei, pequeno Gafanhoto! Na verdade não só as imagens, todo conteúdo disponível na Internet pode estar protegido por direitos autorais. Livros em PDF, textos em blogs, matérias em jornais online, fotos, animações, vídeos. Tudo pode ser protegido e você nem fazia ideia disso.



**ANTES CONTINUAR:** Eu não sou profissional do Direito, então faço uma pausa aqui no material para te indicar um vídeo onde temos o Especialista em Direito Digital **Alan Moreira Lopes** falando sobre esse assunto com muito mais propriedade. A partir de 29min ele responde umas perguntas bem comuns.

Canal Revolution: <https://youtu.be/Bkym20Gq0oQ?t=171>

E se nesse momento você está pensando: "*mas quem vai processar um jovem aluno que está aprendendo a criar sites usando imagens do Google?*".

Minha maior preocupação aqui não é te **PROIBIR** de pegar imagens no Google Imagens e usar nos seus sites enquanto você está aprendendo HTML. O meu medo é você achar que esse ato é **100% LEGAL** e comece a criar sites profissionalmente para empresas usando imagens e conteúdos que você não pode.



**OLHA A TRETA:** Um caso que ficou muito famoso na Internet foi com o canal **Nostalgia**. O Felipe Castanhari, dono do canal, vive recebendo avisos de direitos autorais por usar pedaços de vídeos de documentários, filmes, clipes e séries com direito autoral.

Canal Nostalgia: <https://youtu.be/SJacdAbjdZI>

Concordando ou não com isso, o fato é que a lei existe. E ir contra a lei é ficar totalmente vulnerável aos efeitos causados por ela. Eu até sugiro que façamos uma breve discussão sobre isso em sala de aula. Se eu não for seu professor, bata um papo com ele sobre o assunto. Também sugiro que você converse sobre isso com algum advogado especializado nesse ramo digital caso fique alguma dúvida (toda família tem um tio ou vizinho advogado).

Sendo assim, para que você seja profissionalmente capaz de criar sites com imagens que não vão gerar processos, existem quatro maneiras simples:

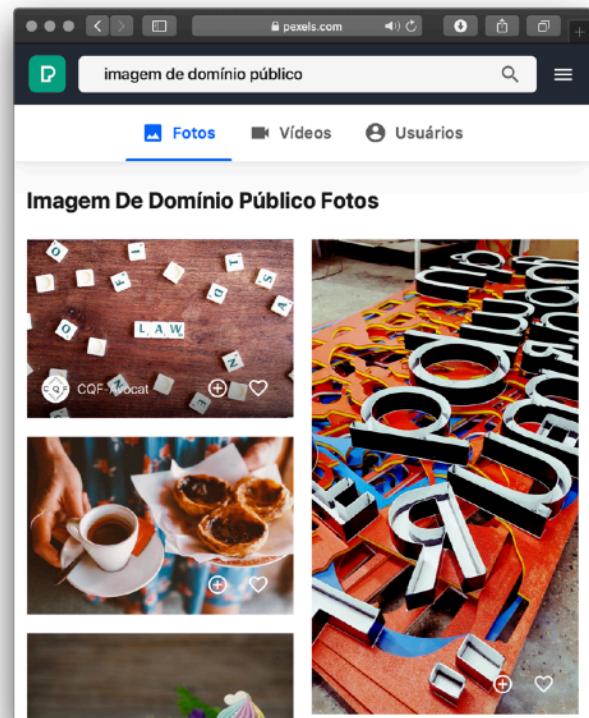
1. Seja o criador da própria arte. Use fotos que seu próprio cliente tirou ou que você mesmo criou.
2. Consiga uma autorização legal de quem é o detentor dos direitos autorais. Isso precisa de documentos específicos para tal.
3. Compre o direito de usar a imagem. Existem vários sites como o **iStockPhoto**, **DepositPhotos**, **ShutterStock**, etc. Só fica de olho aberto e leia as regras, pois na maioria dos casos essa imagem que você comprou só pode ser usada uma única vez ou em um único projeto.
4. Use imagens de domínio público, pois essas você tem a liberdade de colocar no seu site, seguindo as regras estabelecidas pelo autor.

## E onde encontro imagens de domínio público?

Existem vários sites que oferecem imagens de domínio público para serem usados em seus projetos. Mas a dica que te dei antes continua valendo: leia atentamente as instruções para uso dessas imagens. Algumas delas deixam você usar a arte apenas se o artista for creditado em um texto localizado no seu site.

Alguns dos sites que eu mais gosto de visitar quando quero usar imagens com domínio público são:

- [UnSplash](#)
- [Pexels](#)
- [FreePik](#)
- [Rawpixel](#)
- [Pixabay](#)
- [Libreshot](#)
- [Wikimedia Commons](#)



Mas muito **CUIDADO!** Alguns desses sites que disponibilizam imagens de domínio público possuem propagandas de serviços de venda de imagens e acabam te confundindo um pouco.

Vamos mudar de assunto, já que possivelmente você não está se preocupando muito com esse papo de direitos autorais, mesmo ele sendo muito importante para quando você virar um profissional de verdade.

# Vamos falar sobre formatos de imagem

Existem vários formatos de imagem, cada um com suas características, vantagens e desvantagens. Porém, vamos nos focar aqui nos dois formatos compactados mais usados para a criação de sites: **JPEG** e **PNG**.

O algoritmo de compactação **JPEG** é usado para gerar imagens fotográficas com um tamanho extremamente reduzido. Ele foi criado em 1983 por **Eric Hamilton** e hoje é gerenciado pelo **Joint Photographic Experts Group**. Ele é amplamente utilizado por câmeras digitais modernas e programas de tratamento de imagens.

A grande vantagem do uso de arquivos **JPG** (em formato JPEG) é gerar arquivos muito pequenos e que ocupam pouco espaço em disco. Isso é muito importante, pois quando colocarmos nosso site no ar, ele tem que ser leve e carregar as imagens muito rapidamente. Só tome cuidado para não exagerar na hora de configurar o nível de compactação. Isso pode fazer com que sua imagem fique horrível e toda borrada (dá só uma olhada na imagem abaixo).

O formato **Portable Network Graphics** (PNG) surgiu em 1996, desenvolvido pelo **W3C** (o mesmo órgão que gerencia a linguagem HTML) com o objetivo de substituir o formato **GIF** (que hoje voltou a ser popular graças ao WhatsApp e Instagram). Ele também é um formato compactado, mas não tanto quanto o JPEG.

A principal característica do **PNG** - e que o diferencia do JPEG - é a capacidade de configurar a opacidade de cada pixel (deixá-lo transparente ou com transparência limitada).

Mas para entendermos melhor a explicação acima, vamos a um exemplo visual (já que estamos falando de imagens né?)



As duas primeiras imagens estão compactadas no formato JPEG, mas na primeira eu coloquei o nível de compactação em qualidade 5% e ficou com 20KB e na segunda uma qualidade de 30% e ficou com 120KB. A última imagem está comprimida no

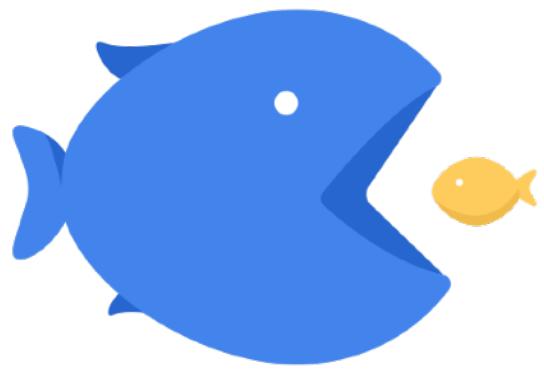
formato PNG, ficou com 300KB e teve o contorno preservado por causa do fundo transparente.

De forma resumida, na hora de escolher o formato de imagem para o seu site, opte sempre pelo formato JPEG com uma compactação entre 30% e 50%. O formato PNG só deverá ser usado quando precisarmos de transparência na foto. Combinado?

## Faltou falar sobre tamanho das imagens

Quando você está começando a desenvolver sites, acaba pensando que para obter o melhor resultado, sua imagem deve ter um tamanho **GIGANTE** para poder ter a maior qualidade.

Realmente, uma imagem com resolução grande (1920x1080) tem mais pontos que uma imagem com resolução pequena (500x280). Só que uma imagem 1920x1080 pode gerar um arquivo de até 3MB, enquanto uma 500x280 dificilmente vai passar de 500KB.



A regra de outro nesses casos é: **use imagens do tamanho certo!** Vai precisar de uma imagem que vai ter 200 pixels de largura? Gere um arquivo exatamente com esse tamanho! Nada de ficar salvando arquivos gigantes e diminuindo o tamanho da imagem com códigos.

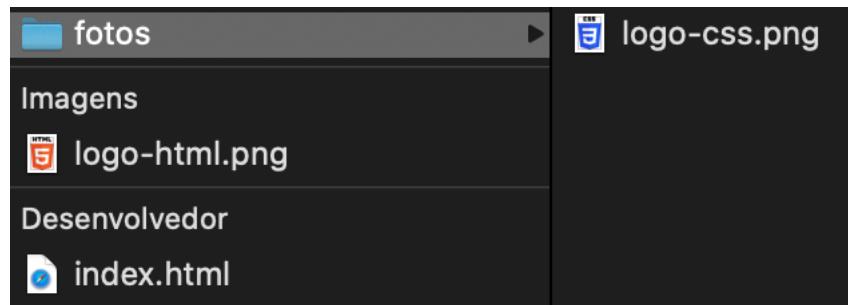
Uma imagem 1920x1080 de 3MB não vai ficar mais leve se você mudar a largura dela no seu CSS. Muito pelo contrário. Seu navegador vai levar um tempão pra carregar o arquivo do seu servidor e vai exibir a imagem minúscula na tela.

E no que isso te prejudica? Os mecanismos de busca como o Google **PENALIZAM** sites lentos e pesados, retirando-os da primeira página de buscas. Quem aqui quer criar um site que não aparece nos resultados do Google?

## Como carregar uma imagem em HTML

Agora que você está especialista em formatos e tamanhos de imagens, vamos gravar algum arquivo em uma pasta chamada **ex003** que você vai criar no seu repositório local. Se você visitar meu repositório em <http://gustavoguanabara.github.io/> e for até a parte de exercícios de HTML, vai ver que na pasta ex003 existem os arquivos logo-html.png e logo-css.png. Faça o download deles pra você ou salve outra imagem qualquer na sua pasta. Só fica de olho no formato dela: JPG ou PNG. Use o VScode para criar também o arquivo index.html e coloque dentro da pasta **ex003**, conforme vou apresentar a seguir.

Eu só te faço um alerta: analise a foto a seguir, onde eu mostro que o arquivo logo.html.png estará na pasta raiz do projeto (mesmo local onde está o arquivo index.html), enquanto o arquivo logo-css.png está localizado dentro da pasta fotos que está na mesma pasta do projeto.



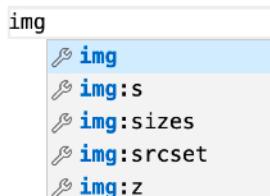
*Dentro da pasta ex003, crie os arquivos acima*

A posição dos seus arquivos vai fazer diferença e você deverá adaptar seu código para poder carregar as imagens sem problemas. Agora vá até o Visual Studio Code e edite o arquivo index.html, criando o código base de sempre e atualizando conforme o que apresento a seguir:

```
8 <body>
9     <h1>Testando carga de Imagens</h1>
10    <p>Abaixo você vai ver uma imagem que está na mesma pasta.</p>
11    
12    <p>Podemos também carregar imagens que estão em outra pasta, contanto
13       que ela esteja dentro da pasta atual.</p>
14    
15    <p>Também podemos carregar imagens externas, e para isso devemos ter a
16       sua URL completa.</p>
17    
19 </body>
```

O código acima só está começando na **linha 8** por questões de praticidade, pois as **linhas de 1 até 7** são aquelas criadas automaticamente pelo VScode. Não apague essas linhas, apenas vamos analisar o código do <body>.

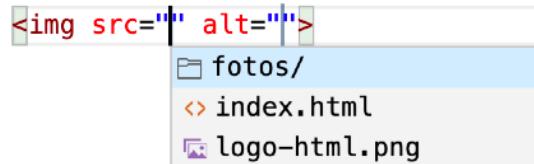
Na **linha 11**, adicionamos a tag <img>, responsável por carregar imagens que estão na pasta do projeto atual ou em links externos. Essa tag tem como parâmetros básicos o src (que vem de *source*, origem) e alt (que vem de *alternative* ou texto alternativo).



Aqui vai uma **DICA IMPORTANTE**: quando for carregar uma imagem no Visual Studio Code, não digite a tag <img> com os colchetes angulares. No lugar disso, digite apenas img e o próprio editor vai sugerir a inserção da imagem (veja foto ao lado).

Agora pressione a tecla **Enter**. Automaticamente, uma tag padrão será preenchida no seu código, incluindo as propriedades básicas src e alt. Mas a dica não para por aí. Sigamos.

Depois do código aparecer magicamente, o cursor do teclado vai aparecer dentro das aspas do parâmetro src. Pressione **Ctrl+Espaço** e mais uma mágica vai acontecer. Como a imagem ao lado mostra, o editor vai te Mostar uma listagem com todos os arquivo disponíveis na pasta do seu projeto. Escolha o arquivo clicando em logo-html.png para que você nem precise digitar o nome dele. Isso evita muitas falhas causadas por erros de digitação!



Agora volte para a página anterior e analise as linhas 13 e 15. A **linha 13** vai carregar a imagem logo-css.png que está dentro da pasta fotos, como vimos anteriormente. Note que foi necessário colocar o nome da pasta seguido de uma barra antes de colocar o nome do arquivo. Se isso não for obedecido, simplesmente a imagem não vai carregar!

Por fim, na **linha 15** fizemos a carga de um arquivo que não estava na pasta do projeto. A imagem js-small.gif está localizada em outro domínio, o jsdotgit.com. Nesse caso, precisamos indicar o caminho completo (**URL**) da imagem.

Você pode obter a URL completa de qualquer imagem, abra um site no Google Chrome, clique com o botão direito do mouse sobre a imagem e escolha a opção "*Copiar endereço da imagem*". Aí é só voltar no seu código HTML, colocar o cursor do teclado dentro das aspas da propriedade src da sua imagem e apertar **Ctrl+V**. Faça uma experiência em casa agora mesmo.

## Para que servem os textos alternativos?

Toda imagem deve ter um texto alternativo, mesmo você achando que isso é muito chato de fazer. Textos alternativos ajudam muito na indexação do seu site em mecanismos de busca e também ajudam muito na **Acessibilidade**, pois se um visitante for deficiente visual, seu navegador vai ser capaz de descrever que tipo de foto está sendo mostrada ali.

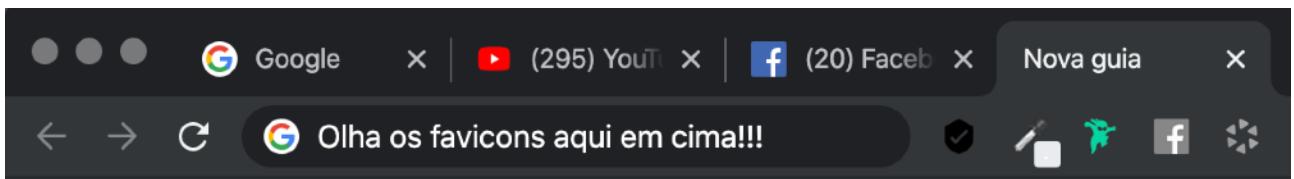
Tente ser objetivo na descrição da sua imagem, mas crie textos que fazem sentido. Tem um monte de gente que coloca o texto "foto" como descrição de uma imagem. Aí não dá!

Os textos alternativos também auxiliam o Google a saber o que tem dentro da sua foto e exibi-la nos resultados de busca do **Google Imagens**.

## Usando Ícone de Favoritos (favicon)

Você já ouviu falar em **favicon**? Talvez você não os identifique pelo nome esquisito, mas com certeza já viu aqueles pequenos ícones que aparecem ao lado dos sites que visitamos, na parte superior do seu navegador. É o mesmo local onde aparecem o texto que você colocou na tag <title> do seu site.

Na imagem abaixo, você consegue ver os favicons do Google, YouTube e Facebook. A última guia não tem favicon, porque não tem site nenhum aberto nela.



Para usar um favicon no seu site, você precisa ter o arquivo do ícone, que geralmente está no formato **ICO**. Se quiser baixar alguns ícones prontos, recomendo o site [IconArchive](#).

Se você tiver o dom e muita paciência, pode desenhar seu favicon no site [favicon.cc](#).

Agora, se seu objetivo é criar um ícone personalizado para seu site baseado em imagens que você já tem, recomendo o site [favicon.io](#), onde você pode submeter qualquer imagem e baixar um pacote com vários formatos de favicon. Lá você também vai poder criar um ícone de favoritos a partir de um texto ou então baseado em um emoji (veja foto ao lado).

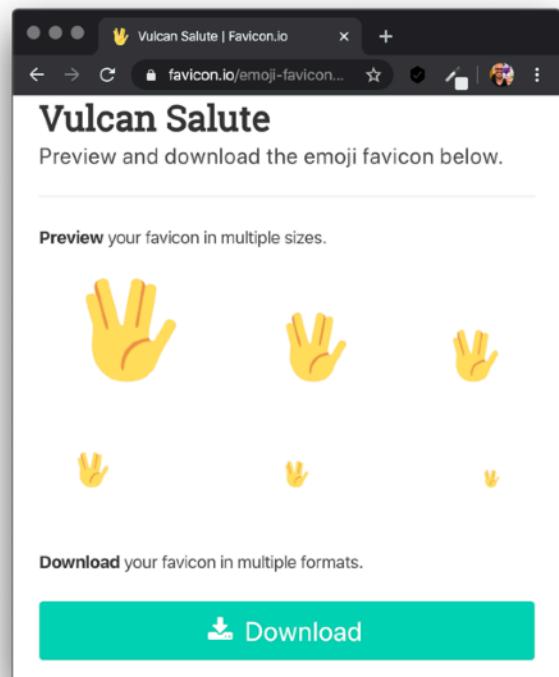
Ao baixar o pacote com os ícones de favoritos, descompacte-os (geralmente é um arquivo ZIP), pegue o arquivo com extensão .ICO e coloque na pasta principal do seu projeto. Normalmente, colocamos o nome do arquivo como favicon.ico.

Agora que você já tem seu favicon salvo na pasta do seu projeto (recomendo criar o **ex004** no seu repositório local), vamos nos focar no código HTML para carregá-lo.

Crie seu arquivo index.html dentro da pasta **ex004**, incluindo seu código base automaticamente. Depois faça uma simples alteração na sua área `<head>`, incluindo uma tag `<link>`, digitando apenas a palavra link em uma linha em branco e escolhendo a opção `link:favicon`.

```
3  <head>
4    <meta charset="UTF-8">
5    <meta name="viewport" content="width=device-width, initial-scale=1.0">
6    <link rel="shortcut icon" href="favicon.ico" type="image/x-icon">
7    <title>Testando o ícone de Favoritos</title>
8  </head>
```

Pronto! É só dar uma olhada no nosso código acima e prestar atenção na **linha 6**. Se você usou a dica que dei acima, não precisou digitar mais nada! Todo o código foi feito automaticamente!



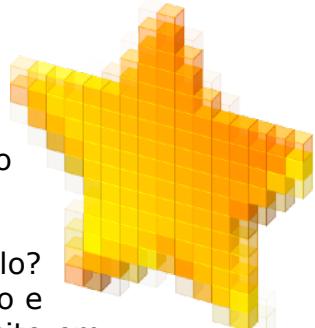
**OBS:** se seu arquivo não se chamar favicon.ico, você deve adaptar a linha 6, colocando o nome do seu ícone.

# Quais são os formatos aceitos para Favicon?

Apesar da minha indicação para usar o arquivo no formato **ICO**, existem outros formatos suportados para o seu favicon, como o próprio **PNG** que estudamos anteriormente e até o formato **SVG**, que é vetorizado e seria a melhor das opções.

Mas se o SVG é a melhor das opções, por que não usá-lo? Infelizmente nem todo navegador é compatível com esse formato e acaba causando algumas inconsistências na exibição do nosso site em outras plataformas.

Sendo assim, pelo menos por enquanto, continuo dando o meu conselho: use o formato ICO.



## Vamos aos desafios!

Lá no repositório, além do material em PDF e dos códigos dos exercícios 100% disponíveis, também disponibilizamos alguns **desafios** que devem ser resolvidos. Esses desafios não incluem o código original e você deve tentar chegar à resposta sem copiar nenhum código.

Com todo o conteúdo que vimos até essa aula, você já pode resolver do **desafio d001** até o **desafio003**. Acesse o repositório público, abra a área do curso de HTML+CSS e clique no link de acesso aos desafios. Manda ver! Só não fica pedindo a resposta! Você consegue resolver isso sozinho(a)!



Repositório em: <https://gustavoguanabara.github.io>

## Eu já falei sobre isso no YouTube?

Eu sei que às vezes as pessoas gostam mais de assistir vídeos do que ler livros, e é por isso que eu lanço há anos materiais no canal Curso em Vídeo no YouTube. O link que vou compartilhar contigo tem o conteúdo explicado como você leu aqui, só que de forma mais ilustrada. Reserve um tempo dos seus estudos para assistir esse vídeo todo.



Curso em Vídeo: [https://www.youtube.com/playlist?list=PLHz\\_AreHm4dlAnJ\\_jJtV29RFxnPHDuk9o](https://www.youtube.com/playlist?list=PLHz_AreHm4dlAnJ_jJtV29RFxnPHDuk9o)

# DevWeb

## Aula 07

### Hierarquia de Títulos

Vamos agora começar a falar sobre organização conteúdo. Ainda teremos muito mais capítulos falando sobre isso daqui pra frente, mas tudo começa nos títulos. São eles que dizem quando existe um assunto principal ou um sub-assunto, dentro de um desses principais. Fazer isso em HTML5 é muito fácil e você está prestes a aprender isso. Vamos começar?



Você tem todo o direito de usar esse material para seu próprio aprendizado. Professores também podem ter acesso a todo o conteúdo e usá-lo com seus alunos. Porém todos os que usarem esse material - seja para qual for a finalidade - deverão manter a referência ao material original, criado pelo **Prof. Gustavo Guanabara** e disponível no endereço do seu repositório público <https://github.com/gustavoguanabara/>. Este material não poderá ser utilizado em nenhuma hipótese para ser replicada - integral ou parcialmente - por autores/editoras para criar livros ou apostilas, com finalidade de obter ganho financeiro com ele.



# Entendendo a hierarquia de títulos

Vamos começar com um exercício prático. Te convido a analisar uma matéria da revista **VOCÊ S/A**, de Fevereiro de 2015. Não precisa se focar em ler o conteúdo, apenas olhe para a imagem a seguir e tente identificar seus componentes visuais.

CARREIRA → LIDERANÇA

## FALTA DE SINTONIA

Um chefe nem sempre simpatiza com seus subordinados. Mas há estratégias para o gestor contornar as diferenças sem prejudicar ninguém ou comprometer o resultado do time *Por Marcia Di Domenico*

**P**assamos tantas horas no escritório que é comum enfrentar sentimentos de amor, ódio e ansiedade. Também é inevitável criar laços de amizade e, claro, de inimizade. Quando o conflito se dá entre chefe e subordinado, a situação pede cautela para não colocar em risco o clima e a produtividade — fatores tão importantes em um período em que a palavra de ordem nas empresas é eficiência. “Não se espera que o líder goste, da mesma forma, de todos na equipe nem que seja igualmente querido por cada integrante”, diz Liane Davey, vice-presidente da Knightsbridge Human Capital Solutions, consultoria em gestão de pessoas em Ontário, no Canadá. “Mas manter uma relação cordial com cada um é obrigatório.” O relacionamento entre chefe e equipe vai bem não quando todo mundo se adora, mas quando o convívio não compromete o trabalho. Gestores que precisam motivar (e gerenciar) profissionais com os quais não se dão bem devem adotar algumas atitudes para que o desafeto não prejudique o relacionamento profissional.

### Pense antes de agir

O fato de se sentir incomodado com alguém diz mais sobre si mesmo do que sobre o outro. Antes de tomar qualquer atitude, o líder deve refletir sobre se o comportamento do funcionário pode demonstrar, na verdade, que há algum problema com a própria gestão. “O chefe deve se perguntar se há alguma coisa que possa mudar em si mesmo para contornar a situação”, diz Jair Moggi, consultor da Adigo DEF, consultoria em gestão, de São Paulo. Isso evita agir por simples impaciência, poupando sua imagem de um desgaste desnecessário.

### Passe mais tempo com seu desafeto

Evitar o contato ou ignorar o subordinado com quem você não se dá bem só piora as coisas. “Ele se sente desestimulado e desmoralizado dentro do time”, diz a consultora Liane Davey. Crie maneiras de passar mais tempo juntos — pode ser um almoço ou um café. Se achar que a iniciativa soa artificial, converse com outras pessoas, o que diminui o desconforto. Essa atitude ajuda a enxergar traços de personalidade às vezes invisíveis e, consequentemente, a entender determinados hábitos e atitudes.

### Saiba com quem está lidando

Tão importante quanto entender seus pontos fortes e fracos como chefe é investir tempo e disposição para aprender sobre a biografia, a personalidade e as limitações de cada integrante do time. Num momento de conflito, esse conhecimento vai auxiliar o gestor a entender o que pode estar por trás de determinadas ações e a escolher o tratamento adequado. “O funcionário sente que pertence ao grupo, o que é um dos princípios fundamentais da motivação”, diz Jair. Deixar claro o que a empresa espera do profissional e se certificar de que o subordinado comprehende o que precisa fazer também é importante para estimular a equipe.

### Tenha objetividade

Independentemente do motivo de não gostar de alguém, nunca leve a questão para o lado pessoal. A razão do mal-estar deve ser objetiva, assim o gestor conseguirá abordar o problema com racionalidade. A melhor maneira de falar sobre o assunto é em uma conversa a sós com o subordinado para evitar o desgaste da pessoa com o grupo. “Todo feedback precisa estar embasado na observação de alguma atitude ou episódio real e na reflexão do que isso representa para o trabalho”, diz Maria Cândida Baumer, diretora da People & Results, consultoria de São Paulo. Caso contrário, pode ser interpretado simplesmente como crítica e não repercutir como desejado. Outro erro é mandar recado por um colega da equipe, usar ironia ou jogar indiretas. “A agressividade

34 FEVEREIRO DE 2015 | VOCÊ S/A

Agora que você já deu uma olhada na estrutura, vamos procurar identificar a sua hierarquia de títulos. Com toda certeza, se eu te perguntar qual é o título principal da matéria, a resposta vem imediatamente: “*Falta de sintonia*”. Isso acontece porque esse é o texto que está em destaque principal no texto.

Mas aí, te faço outra pergunta: existem outros títulos nessa página? Uma breve análise visual nos leva a um resultado:

- *Pense antes de agir*
- *Passe mais tempo com seu desafeto*
- *Saiba com quem está lidando*
- *Tenha objetividade*

Outra coisa que você também consegue identificar é um parágrafo destacado logo após o título principal, onde a fonte do texto é até maior e mais destacada, mas não chega a ser um título. Em seguida, existe um parágrafo introdutório, onde encontramos um texto que aponta a essência da matéria.

Depois disso, começamos com os sub-títulos, que com certeza são partes que falam sobre o assunto "Falta de sintonia", mas separa o tema principal em sub-temas para melhorar a organização dos "pensamentos" de quem escreveu a matéria. Isso facilita muito na organização dos textos.

Dentro de cada um desses sub-títulos temos um parágrafo que desenvolve mais sobre cada assunto.



**COMECE A PRESTAR ATENÇÃO NISSO:** Algumas matérias em jornais e revistas simplesmente não separam o conteúdo em sub-títulos. Geralmente isso gera textos longos e cansativos. Separar seu texto em assuntos secundários prendem bastante o leitor.

Quantas vezes você lê um livro e fica constantemente virando as páginas para saber quanto falta para acabar o capítulo atual? Geralmente livros técnicos possuem capítulos separados por sub-títulos, exatamente como estou fazendo aqui. Isso não foi uma escolha arbitrária, prefiro ler coisas assim.

Acho que deu pra entender meu ponto aqui, não é? Organizar o nosso conteúdo quando criamos páginas é essencial para ter um bom site.

## Como criar títulos em HTML?

Títulos em HTML são conhecidos como *headings* (que, traduzindo do Inglês, significa *título* mesmo 😅). Os títulos possuem **SEIS NÍVEIS** de hierarquia, e esses níveis servem para organizar nosso conteúdo. Para isso, usamos as tags `<h1>`, `<h2>`, `<h3>`, `<h4>`, `<h5>` e `<h6>` para demarcar nossos títulos.

## Mas cuidado! Tamanho não é documento!

Tem muita gente por aí que acha que `<h1>` significa "*texto com fonte grande*" e `<h6>` significa "*texto com fonte pequena*". Isso não tem fundamento algum! Por favor, se alguém te disse isso algum dia, essa pessoa estava completamente equivocada.

Um `<h1>` significa que esse é um assunto principal e é sobre ele que vamos escrever. Já o `<h2>` significa que esse é um sub-assunto do `<h1>` que está imediatamente acima dele. O `<h3>` significa que esse é um sub-assunto do `<h2>` que está acima dele. E assim sucessivamente até o `<h6>`.

Faça um outro exercício aí. Estamos agora na página 3 (é só olhar aqui embaixo). Vá para a página 1 desse capítulo e olhe o nosso `<h1>`: "*Hierarquia de Títulos*". Nessa página 2, temos o sub-título `<h2>`: "*Entendendo a hierarquia de títulos*". Ele é um sub-assunto do nosso `<h1>`.

Nessa página 3 também temos outro `<h2>`: "*Como criar títulos...*" e logo abaixo temos um `<h3>`: "*Mas cuidado...*". Note que esse é um sub-assunto de "*como criar títulos...*". É claro que o nosso H3 tem letra menor que nosso H2, mas não se trata de tamanho. É questão de hierarquia!

# Só um H1 em uma página: fato ou lenda?

Rola por aí muita gente falando que um documento HTML só deve ter um título H1 e que todos os demais devem ser sub-títulos deles. Isso surgiu com a galera que falava sobre otimização de páginas para mecanismos de busca (*SEO - Search Engine Optimization*).

Hoje em dia, o próprio Google (maior mecanismo de busca do mundo) já diz que isso é uma lenda. Para eles, o que mais importa é dar significado ao seu H1. Ele é um tema principal, e se a sua página tem vários temas principais, use vários H1 sem medo.



**OUÇA O ESPECIALISTA:** Aqui embaixo você vai encontrar um link para o vídeo do canal do **Google para Webmasters** onde o especialista **John Mueller** fala sobre esse assunto. O conteúdo está em inglês, mas você pode clicar no botão da engrenagem no canto inferior direito do vídeo e mandar **traduzir automaticamente** as legendas para o Português. Fica ótimo!

Acesse: <https://youtu.be/WsgrSxCmMbM>

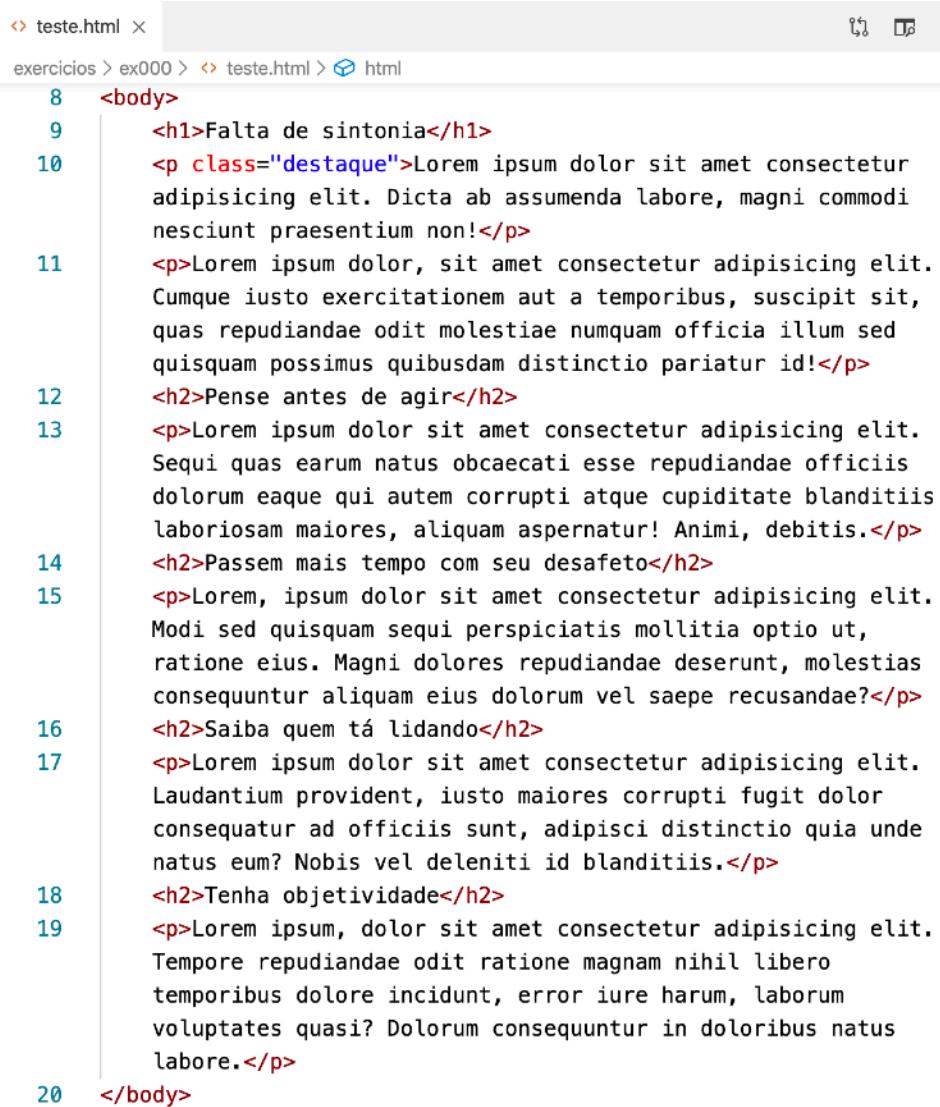
## Vamos criar a hierarquia da matéria da revista

A missão agora é transformar o conteúdo da matéria que te mostrei no início do capítulo em uma página HTML. Eu não vou me focar em digitar o texto dos parágrafos de cada seção e no lugar deles vou usar os famosos *Lorem Ipsum* genéricos.



**LOREM O QUE?** O *Lorem Ipsum* é um texto padrão em Latim utilizado na produção gráfica para preencher os espaços de texto em publicações (jornais, revistas, e sites) para testar e ajustar aspectos visuais (layout, tipografia, formatação, etc.) antes de utilizar conteúdo real.

Vamos começar abrindo o **Visual Studio Code** e criando um documento teste.html em uma pasta qualquer. Dentro desse documento, crie o código base HTML usando a exclamação, como já ensinei algumas vezes durante esse material (consulte o Capítulo 4, página 6). Dentro da tag <body>, criaremos a estrutura do conteúdo.



```
8 <body>
9   <h1>Falta de sintonia</h1>
10  <p class="destaque">Lorem ipsum dolor sit amet consectetur
11    adipisicing elit. Dicta ab assumenda labore, magni commodi
12    nesciunt praesentium non!</p>
13  <p>Lorem ipsum dolor, sit amet consectetur adipisicing elit.
14    Cumque iusto exercitationem aut a temporibus, suscipit sit,
15    quas repudiandae odit molestiae numquam officia illum sed
16    quisquam possimus quibusdam distinctio pariatur id!</p>
17  <h2>Pense antes de agir</h2>
18  <p>Lorem ipsum dolor sit amet consectetur adipisicing elit.
19    Sequi quas earum natus obcaecati esse repudiandae officiis
20    dolorum eaque qui autem corrupti atque cupiditate blanditiis
21    laboriosam maiores, aliquam aspernatur! Animi, debitis.</p>
22  <h2>Passem mais tempo com seu desafeto</h2>
23  <p>Lorem, ipsum dolor sit amet consectetur adipisicing elit.
24    Modi sed quisquam sequi perspiciatis mollitia optio ut,
25    ratione eius. Magni dolores repudiandae deserunt, molestias
26    consequuntur aliquam eius dolorum vel saepe recusandae?</p>
27  <h2>Saiba quem tá lidando</h2>
28  <p>Lorem ipsum dolor sit amet consectetur adipisicing elit.
29    Laudantium provident, iusto maiores corrupti fugit dolor
30    consequatur ad officiis sunt, adipisci distinctio quia unde
31    natus eum? Nobis vel deleniti id blanditiis.</p>
32  <h2>Tenha objetividade</h2>
33  <p>Lorem ipsum, dolor sit amet consectetur adipisicing elit.
34    Tempore repudiandae odit ratione magnam nihil libero
35    temporibus dolore incident, error iure harum, laborum
36    voluptates quasi? Dolorum consequuntur in doloribus natus
37    labore.</p>
38 </body>
```

Você viu que a imagem acima começa na **linha 8**. As anteriores são o código HTML automático criado pelo VS Code.

Na **linha 9** temos nosso título principal. Nas **linhas 12, 14, 16 e 18** temos os sub-títulos do principal, como analisamos anteriormente.



**criando lorem automaticamente:** Até na hora de criar um *Lorem Ipsum*, o VS Code te ajuda. Dentro de um parágrafo recém criado, digite apenas a palavra **lorem** e pressione **Enter**. Pronto!

**OBS:** Na **linha 10**, tivemos uma novidade que foi a atribuição de uma classe ao parágrafo após o `<h1>`. Mais para frente falaremos mais sobre isso, mas basicamente é para conseguirmos criar uma configuração visual diferente para esse parágrafo especial usando folhas de estilo CSS.

# Hora de exercitar

Agora chegou a hora de praticar. Acesse agora mesmo o endereço <https://gustavoguanabara.github.io/html-css/exercicios/> e execute o **exercício 006** no seu computador e tente atingir esse mesmo resultado em casa, sem copiar o código que eu criei. Nesse momento, a prática é algo que você mais precisa. Se por acaso ficar difícil, pode acessar o repositório público de HTML e CSS e dar uma olhada nos comandos, mas **EVITE COPIAR**.



## Eu já falei sobre isso no YouTube?

Eu sei que às vezes as pessoas gostam mais de assistir vídeos do que ler livros, e é por isso que eu lanço há anos materiais no canal Curso em Vídeo no YouTube. O link que vou compartilhar contigo tem o conteúdo explicado como você leu aqui, só que de forma mais ilustrada. Reserve um tempo dos seus estudos para assistir esse vídeo todo.



Curso em Vídeo: [https://www.youtube.com/playlist?list=PLHz\\_AreHm4d1AnJ\\_jJtV29RFxnPHDuk9o](https://www.youtube.com/playlist?list=PLHz_AreHm4d1AnJ_jJtV29RFxnPHDuk9o)

# DevWeb

## Capítulo 08

# Formatação de Textos

Eu já consigo sentir que estamos evoluindo na HTML, você não? Agora nossos documentos começam a tomar uma organização maior com os títulos e parágrafos, mas às vezes precisamos de algo a mais. Vamos ver algumas formatações importantes e que usaremos quando começarmos a produzir nossos conteúdos. Não se esqueça de manter tudo anotado, pois veremos muitas tags novas.



Você tem todo o direito de usar esse material para seu próprio aprendizado. Professores também podem ter acesso a todo o conteúdo e usá-lo com seus alunos. Porém todos o que usarem esse material - seja para qual for a finalidade - deverão manter a referência ao material original, criado pelo **Prof. Gustavo Guanabara** e disponível no endereço do seu repositório público <https://github.com/gustavoguanabara/>. Este material não poderá ser utilizado em nenhuma hipótese para ser replicada - integral ou parcialmente - por autores/editoras para criar livros ou apostilas, com finalidade de obter ganho financeiro com ele.



# Você já ouviu falar de semântica?

Tá aí mais uma palavra bonita pra gente aprender: “semântica”. Se você nunca ouviu falar nela, provavelmente não está entendendo direito, mas saiba que a maior mudança da HTML4 para a HTML5 é o acréscimo da semântica aos elementos.

Vamos procurar no dicionário, e vou fazer isso no **Michaelis**, que tem uma versão online e gratuita. Segundo o pai-dos-burros:

“Semântica é o significado dos vocábulos, por oposição à sua forma.”



Analisando a frase acima percebemos que uma palavra pode ter **forma** e **significado**, e que a **semântica** dá mais valor ao significado.

Na HTML4, tínhamos tags como `<b>` que colocava um texto em negrito, `<u>` que colocava o termo sublinhado e `<blink>` que fazia o texto piscar. Essas eram tags que representavam apenas uma **forma**. Você dizia que queria um texto sublinhado, mas qual era o motivo? Qual era o sentido de sublinhar alguma coisa? Vamos ver um exemplo:

Juvenal era um sujeito de muita sorte. E já começou de pequeno, onde morou na Rua Marquês de Lira Filho, um local de fácil acesso ao Centro da cidade.

Em HTML4, colocamos o par de tags `<u>` e `</u>` para delimitar o termo Rua Marquês de Lira Filho. Isso seria uma maneira de determinar somente um formato visual para chamar atenção para o endereço onde o cara nasceu. O sublinhado é apenas uma **forma**, sem **significado** explícito. Sublinhamos só pra chamar atenção visualmente.

Já a HTML5 chegou com o conceito de valorizar a **semântica**, logo suas tags tentam levar um **significado** embutido muito forte. Logo, a frase acima ficaria assim:

Juvenal era um sujeito de muita sorte. E já começou de pequeno onde morou na `<address>Rua Marquês de Lira Filho</address>`, um local de fácil acesso ao Centro da cidade.

Note que agora, usamos a tag `<address>` para dar um significado ao destaque que fizemos. Estamos chamando atenção para Rua Marquês de Lira Filho por se tratar do endereço da pessoa. No caso, um navegador de celular pode até sugerir que você veja o mapa do local e trace a rota para chegar lá. Viu? **SIGNIFICADO!**

Sendo assim, em HTML5, vemos de forma bastante evidente a presença do chamado **HTML semântico** ou **tags semânticas** ou ainda o **conteúdo semântico**. Você quer que um endereço apareça na forma de um texto sublinhado? Use CSS para configurar isso, a HTML serve para dar sentido ao conteúdo. É assim que tudo vai funcionar.

Desde a mudança de versões, a **W3C** - consórcio responsável por normatizar a HTML - tem dado muito valor por adicionar novas tags que tenham mais significado e a tirar

algumas tags que só se focam no efeito visual (forma) de apresentação. O intuito é deixar a apresentação gráfica por conta das CSS.

## Tags morrem, você sabia?

Existem tags que ainda funcionam hoje em dia, mas estão prestes a serem consideradas obsoletas, como as tags `<font>` e `<center>`. E existem outras que simplesmente foram excluídas das versões mais atuais da HTML5, como é o caso do `<applet>` e `<blink>`.



Isso acontece porque a linguagem evolui, e nesse processo algumas tags param de fazer sentido pois existem outras bem melhores e que fazem mais sentido.

Sendo assim, a própria W3C sugere que no lugar de `<b>`, que significa **bold** ou **negrito** e que seria simplesmente uma forma de apresentar um dado, passemos a usar a tag `<strong>`, que tem um significado de **força** ou **potência** dentro da frase.



**FIQUE SEMPRE DE OLHO:** Existe um documento oficial do Consórcio da World Wide Web (W3C) que é atualizado constantemente com as tags que estão ficando obsoletas e algumas substituição desejáveis que devemos fazer.

Diferenças entre HTML4 e 5: <https://www.w3.org/TR/html5-diff/#absent-attributes>

Elementos obsoletos na HTML5: <https://dev.w3.org/html5/pf-summary/obsolete.html>

Sendo assim, se você está aprendendo HTML com o uso das tags `<font>`, `<big>`, `<center>`, `<srtike>` e muitas outras, pode mandar esses links **OFICIAIS** que estão aí em cima para quem está te ensinando desse jeito. Talvez essa pessoa nem saiba ainda que deverá atualizar seus materiais o quanto antes.

## Antes de começar, fique de olho nos exercícios

Como você já deve saber, esse material é para ser consumido juntamente com os demais conteúdos disponibilizados no nosso repositório público do **Curso de HTML+CSS**, disponível em <https://gustavoguanabara.github.io>. Pois abra o código disponível dos exercícios resolvidos e analise os códigos dos exercícios **ex007** e **ex008**. Abra também o link onde você pode executar os exercícios e rode esses mesmos exemplos 007 e 008. As coisas vão ficar muito claras pra você daqui pra frente, eu garanto!



# Negrito e Itálico

Vamos ver agora algumas formatações bem usadas das últimas versões da linguagem, começando pelos famosos **negrito** e *itálico*.

Como vimos anteriormente, existem as tags `<b>` e `<i>` para essa tarefa, mas elas não possuem significado e focam apenas na forma, sendo assim, são pouco semânticos. Sendo assim, recomendamos que você passe a usar as tags `<strong>` e `<em>` para realizar essas mesmas formatações visuais, só que agora com sentido.

A tag `<strong>` significa que o termo delimitado possui força dentro da frase. Logo, ele aparecerá em **negrito**.

Já a tag `<em>` significa que queremos dar **ênfase** (do Inglês *emphasis*) ao termo. Logo, ele aparecerá em *itálico*.

Note que, ao usar `<strong>` e `<em>` no lugar de `<b>` e `<i>`, damos mais significado aos nossos termos e conteúdos. Como eles vão ser representados visualmente (forma), vai depender das nossas folhas de estilo CSS.

## Você sabe usar marca texto?



Provavelmente você já viu ou usou uma dessas canetas marca texto em seu dia-a-dia. Elas servem para você marcar uma parte do texto na qual você quer dar uma ênfase descomunal, já que se trata de um trecho muito valioso para você, assim como acabamos de fazer.

Para fazer essas marcas em HTML5, usamos a tag `<mark>...</mark>` para delimitar o texto que queremos demarcar, como se estivéssemos usando uma caneta marcador.



**COMO FAÇO PARA MUDAR A COR DA CANETA?** Com certeza você já deve estar se perguntando como usar outras cores. Pois saiba que isso é totalmente possível, contanto que as configurações sejam especificadas nas folhas de estilo CSS.

# <big> morreu, mas o <small> ainda sobrevive

Tem certas coisas que simplesmente não são fáceis de entender. A tag `<big>` (que deixava o texto maior) está depreciada pela HTML5, mas a tag `<small>` (que deixa o texto menor) segue firme e forte na vida da linguagem.



E não adianta ficar me olhando com essa cara de reprovação. Eu não tenho culpa nenhuma por conta dessa decisão, sou apenas um mero professor que se foca em mostrar os fatos, não em fazê-los ter um sentido claro. Tem certas coisas que a gente não questiona, só aceita que dói menos.

## Texto deletado

Antigamente existia a tag `<strike>`, que hoje também está depreciada pela última versão da HTML. No lugar dela entrou a tag `<del>`, que significa que ~~o texto está ali, pode até ser lido, mas deve ser desconsiderado~~ pelo leitor. Exatamente como eu acabei de fazer.

## Texto inserido

O texto inserido é o exato oposto do texto deletado que vimos anteriormente. Nesse caso, se colocarmos um texto qualquer dentro de `<ins>` e `</ins>`, estamos dizendo que o texto está ali, deve ser lido e você deve prestar atenção nele. E se provavelmente você está se perguntando qual é a diferença entre usar `<ins>` ou `<em>` (que vimos anteriormente), te digo com a maior paz na alma: "*O dia que você descobrir, me explica porque eu também quero saber*".

## Textos sobrescrito e subscrito

Quantas vezes você tentou escrever um  $x^2$  ou um  $H_2O$  em um documento e ficou na dúvida de como fazer pra esses números ficarem pequeninos ali em cima ou embaixo. Pois a HTML tem as tags `<sub>` e `<sup>` justamente para essa finalidade. Ao analisar os exemplos de código que apontei no início desse documento, você vai perceber melhor a diferença entre eles.

A large blue stylized letter 'X' is positioned next to a smaller blue 'X' with a superscript '2' above it, representing mathematical notation.

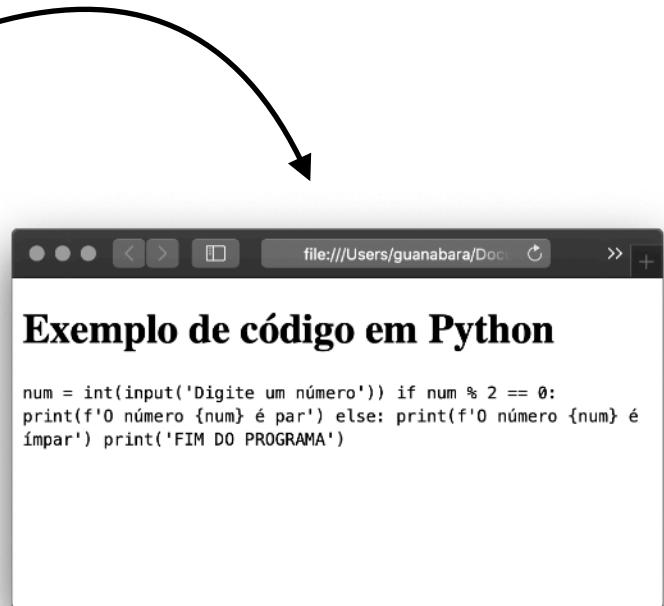
## Trechos de código

Nós somos (ou seremos) programadores, e por isso compartilhamos muito código-fonte em diversas linguagens. Para isso, existe a tag `<code>` da HTML onde você pode delimitar seu código. A principal vantagem no uso dessa tag é a o valor semântico que ela representa, indicando ao navegador que se trata de um código de computador.

Porém, existe também um efeito visual, pois as letras ficam no modo mono-espaçadas (monospace), o que facilita bastante a leitura do código. Analise, por exemplo, o trecho de código a seguir:

```
<h1>Exemplo de código em Python</h1>
<code>
num = int(input('Digite um número'))
if num % 2 == 0:
    print(f'O número {num} é par')
else:
    print(f'O número {num} é ímpar')
print('FIM DO PROGRAMA')
</code>
```

Você pode achar que o código vai aparecer lindamente no seu site, porém o que acontece é simplesmente uma mudança das letras, para que elas fiquem mono-espaçadas (veja a imagem).



Para resolver esse problema, vamos usar uma outra tag HTML chamada `<pre>`, que mantém o texto pré-formatado, exatamente da mesma maneira na qual ele foi digitado, incluindo quebras de linhas, espaços e tabulações.

```
<h1>Exemplo de código em Python</h1>
<pre><code>
num = int(input('Digite um número'))
if num % 2 == 0:
    print(f'O número {num} é par')
else:
    print(f'O número {num} é ímpar')
print('FIM DO PROGRAMA')
</code></pre>
```

Notou agora a diferença? A junção das tags `<pre>` e `<code>` em conjunto nos trouxe um resultado visual bem mais interessante.



**AS TAGS, UNIDAS JAMAIS SERÃO VENCIDAS!** No exemplo acima você pode perceber que adicionamos uma tag dentro da outra. Quando isso acontece, é como se elas juntassem suas forças para gerar um resultado ainda mais poderoso. É possível então juntar `<strong>` e `<em>` e gerar um resultado como esse: ***em negrito e em itálico ao mesmo tempo!***

# “Citações”

Se você já escreveu um texto sequer na sua vida, com certeza já teve que fazer citações. Uma citação é um trecho de texto, escrito ou dito por outra pessoa, que vai ilustrar perfeitamente algo que você quer explicar.

“

”

Normalmente, uma citação aparece entre aspas ou com as margens deslocadas em relação ao texto. Confuso pra você? Pois volte na página 2 desse capítulo e lá você vai ver três citações escritas. Com certeza você vai descobrir!

Para criar uma citação em HTML, podemos usar a tag `<q>` (do Inglês *quote*, que significa citar). O texto que estiver entre `<q>` e `</q>` já vai receber automaticamente as aspas, mas não terá nenhum deslocamento. Essa técnica é mais usada quando queremos uma citação no meio de um parágrafo.

Também podemos criar citações mais longas (em bloco) e que tenham um parágrafo só para si. Nesse caso, colocaremos tudo dentro de `<blockquote>` e `</blockquote>` e o texto ganha um recuo automaticamente. Podemos também colocar um link para o texto original, usando o parâmetro `cite` dentro da tag.

```
<blockquote cite="https://www.martinsfontespaulista.com.br/
php-a-biblia-53304.aspx/p">
```

O PHP é uma linguagem para a criação de scripts para a Web do lado servidor embutidos em HTML, cujo código-fonte é aberto, e que é compatível com os mais importantes servidores Web (especialmente Apache).

```
</blockquote>
```

# Abreviações

Essa é uma novidade da HTML5 e que ajuda muito em áreas como a de Tecnologia, que usa muitas siglas e abreviações. Sempre que você quiser escrever uma sigla, mas deixar claro ao usuário (e aos mecanismos de busca) o significado dela, use a tag `<abbr>`.

```
<p>Eu estou estudando <abbr title="Hyper Text Markup
Language">HTML</abbr> para criar sites.</p>
```

Note no código acima que usamos o `title` para indicar o significado da sigla. Ao lado, mostro o funcionamento desse código no navegador.

Eu estou estudando HTML para criar sites.

Hyper Text Markup Language

Quando passamos o mouse sobre a sigla abreviada, um pequeno texto aparece com o seu significado.

# ODITREVNI OTXET

Não entendeu nada desse título? Pois volte ali em cima e leia da última letra até a primeira. Achou inútil? Eu também. Pois é exatamente essa a função da tag `<bdo>`.

Para começo de conversa, BDO significa *bi-directional override*. Ao usar essa tag, coloque também o parâmetro `dir` para indicar uma das duas direções possíveis:

`rtl` = da direita para a esquerda (*right-to-left*)

`ltr` = da esquerda para a direita (*left-to-right*)

```
<h2>Texto Invertido</h2>
<bdo dir="rtl">Este texto todo está invertido no formato
RIGHT-T0-LEFT.</bdo>
```

## E aí, já acabou?

Nesse capítulo, aprendemos vários tipos de formatação de textos. Mas gostaria de deixar bem claro que existem muitos outros. O que fiz aqui foi uma seleção dos mais usados (e também o `<bdo>`, que é o mais inútil) e vamos dar prosseguimento na matéria. Se por acaso, mais pra frente, precisarmos usar alguma outra tag de formatação de textos, eu explico pontualmente. Combinado?

## Hora de exercitar

Eu já dei esse conselho lá em cima, mas não custa nada repetir. Acesse agora mesmo o endereço do nosso repositório público em <https://gustavoguanabara.github.io/html-css/exercicios/> e execute o **exercício 007** e o **exercício 008** no seu computador e tente atingir esse mesmo resultado em casa, sem copiar o código que eu criei. Nesse momento, a prática é algo que você mais precisa. Se por acaso ficar difícil, pode acessar o repositório público de HTML e CSS e dar uma olhada nos comandos, mas **EVITE COPIAR**.



## Eu já falei sobre isso no YouTube?

Eu sei que às vezes as pessoas gostam mais de assistir vídeos do que ler livros, e é por isso que eu lanço há anos materiais no canal Curso em Vídeo no YouTube. O link que vou compartilhar contigo tem o conteúdo explicado como você leu aqui, só que de forma mais ilustrada. Reserve um tempo dos seus estudos para assistir esse vídeo todo.



Curso em Vídeo: [https://www.youtube.com/playlist?list=PLHz\\_AreHm4d1AnJ\\_jJtV29RFxnPHDuk9o](https://www.youtube.com/playlist?list=PLHz_AreHm4d1AnJ_jJtV29RFxnPHDuk9o)

# DevWeb

## Capítulo 09

### Listas com HTML

- ✓ Listas são importantes
- ✓ Facilitam a leitura
- ✓ Organizam o conteúdo
- ✓ Simplificam a apresentação dos itens
- ✓ Viu como é simples?
- ✓ Quer aprender como se faz?
- ✓ Vem comigo!



Você tem todo o direito de usar esse material para seu próprio aprendizado. Professores também podem ter acesso a todo o conteúdo e usá-lo com seus alunos. Porém todos que usarem esse material - seja para qual for a finalidade - deverão manter a referência ao material original, criado pelo **Prof. Gustavo Guanabara** e disponível no endereço do seu repositório público <https://github.com/gustavoguanabara/>. Este material não poderá ser utilizado em nenhuma hipótese para ser replicada - integral ou parcialmente - por autores/editoras para criar livros ou apostilas, com finalidade de obter ganho financeiro com ele.



# O poder das listas

Pare para pensar por um só minuto: quantas listas você faz na sua vida? Eu, sinceramente não vivo sem listas: a lista do supermercado, lista de tarefas, a lista de chamada dos meus alunos, listas dos conteúdos a trabalhar cada semestre, lista de cursos que quero fazer, lista de cursos que quero produzir. Acho que já deu pra entender, não é?

Para quem produz conteúdo, listas também são muito úteis. Elas simplificam e sintetizam o conteúdo a ser passado e faz a demarcação de cada item para facilitar a visualização.



Por conta disso, a linguagem HTML disponibiliza vários tipos de lista para produzirmos nosso conteúdo e deixá-lo mais claro e eficiente.

## Listas Ordenadas

A HTML chama de **ordered lists** todas aquelas listas onde a ordem dos itens é algo muito importante. Um passo-a-passo para criar um bolo, uma lista de aprovados no vestibular e uma lista com os carros mais caros do mundo são exemplos de listas ordenadas.

Para criar uma *ordered list*, vamos usar a tag `<ol>` para delimitar a lista e `<li>` (*list item*) para identificar cada item da lista.

```
<ol>
  <li>Item 1</li>
  <li>Item 2</li>
  <li>Item 3</li>
  <li>Item 4</li>
  <li>Item 5</li>
</ol>
```



**IMPORTANTE!** Segundo a **W3C**, a tag `<ol>` é de fechamento obrigatório (ou seja, devemos sempre usar `</ol>`). Já a tag `<li>` tem seu fechamento **opcional** a partir da HTML5.

A tag `<ol>` possui um parâmetro `type`, onde configuramos o tipo de marcador da lista atual. As opções de valores para esse parâmetro são:

- ▶ 1 - Valor padrão. Cria listas numeradas. Ex: 1, 2, 3, 4, ...
- ▶ A - Cria listas alfabéticas em maiúsculas. Ex: A, B, C, D, ...
- ▶ a - Cria listas alfabéticas em minúsculas. Ex: a, b, c, d, ...
- ▶ I - Cria listas com algarismos romanos em maiúsculas. Ex: I, II, III, IV, ...
- ▶ i - Cria listas com algarismos romanos em minúsculas. Ex: i, ii, iii, iv, ...

Você também pode indicar o início da contagem usando o parâmetro `start`.

Por exemplo, a tag `<ol type="I" start = "5">` vai gerar itens numerados como V, VI, VII, VIII, IX, ...

## Listas não Ordenadas

Se você compreendeu a criação de listas ordenadas, com certeza vai entender as ***unordered lists***, também chamadas de listas com marcadores, que são aquelas onde a ordem dos itens não influenciará no significado da lista. Ela é apenas uma ótima maneira para organizar os itens que não apresentam uma classificação necessariamente.

Para criar uma *unordered list*, vamos usar a tag `<ul>` para delimitar a lista e a tag `<li>` para criar cada um dos seus itens internos.

```
<ul>
  <li>Item 1</li>
  <li>Item 2</li>
  <li>Item 3</li>
  <li>Item 4</li>
  <li>Item 5</li>
</ul>
```

O marcador padrão é a bolinha preta totalmente preenchida (circle), mas existe a opção de configurar a propriedade `type` da tag `<ul>` com os seguintes valores:

- ▶ disc - padrão. Uma bola preta totalmente pintada
- ▶ circle - Uma bola com uma borda preta e sem preenchimento
- ▶ square - Um pequeno quadrado preto totalmente pintado

## Misturando as coisas

Podemos também criar listas mistas, configurando listas dentro de outras listas. Veja o exemplo a seguir:

```
<ol>
  <li>Item 1</li>
  <li>Item 2</li>
    <ol type="a">
      <li>Item 2.1</li>
      <li>Item 2.2</li>
      <li>Item 2.3</li>
    </ol>
  <li>Item 3</li>
  <li>Item 4</li>
  <li>Item 5</li>
</ol>
```

O resultado visual do código que vimos anteriormente será semelhante ao que temos a seguir:

1. Item 1
2. Item 2
  - a. Item 2.1
  - b. Item 2.2
  - c. Item 2.3
3. Item 3
4. Item 4
5. Item 5

Note que os itens da lista interna (a, b, c, ...) está deslocado para a direita em relação ao item 2, do qual essa sub-lista é filha.



**DICA:** Além de aninhar listas ordenadas, podemos juntar listas `<ul>` com `<ol>` e vice-versa. As listas internas sempre terão deslocamento interno para a direita.

## Exercício proposto

Crie seu próprio código e faça a seguinte lista aninhada com seus jogos favoritos de acordo com a plataforma.

### **Lista dos meus jogos favoritos**

- Nintendinho
  - 1. Mario Bros
    - Super Mario Bros 3
    - Mario: Lost Levels
  - 2. Ninja Gaiden
    - Ninja Gaiden III
- Super Nintendo
  - 1. Super Mario World
  - 2. Donkey Kong
    - Donkey Kong Country III
    - Diddy's Kong Quest
- Playstation 1
  - 1. Final Fantasy
    - Final Fantasy VII
    - Final Fantasy IX
  - 2. Castlevania
    - Symphony of the Night

Tenha paciência e dedique-se, com certeza você vai conseguir fazer uma lista como essa aí em cima e vai aprender muito sobre HTML.

# Listas de Definições

É como se fosse um dicionário, temos os termos e as suas descrições. É uma lista sem demarcadores, mas bem útil em alguns casos.

Toda lista de definições está dentro de uma tag `<dl> </dl>` (*definition list*). Cada termo é um `<dt>` (*definition term*) e cada descrição é um `<dd>` (*definition description*). Assim como os itens da lista, essas duas últimas tags possuem fechamento opcional, segundo a referência oficial da HTML5.

Vamos ver um exemplo simples que cria uma lista com três definições que já conhecemos bem aqui pelo curso.

```
<dl>
  <dt>HTML</dt>
  <dd>Linguagem de marcação utilizada para criar o conteúdo de sites.</dd>

  <dt>CSS</dt>
  <dd>Linguagem de marcação para a especificação de estilos em sites.</dd>

  <dt>JavaScript</dt>
  <dd>Linguagem de programação para criar interatividades em sites.</dd>
</dl>
```

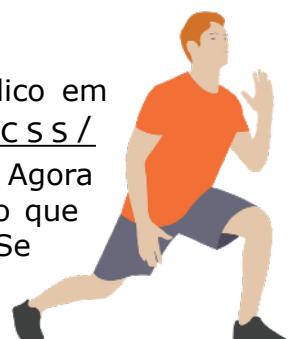
O resultado visual dessa lista aplicado pelo navegador é:

HTML  
Linguagem de marcação utilizada para criar o conteúdo de sites.  
CSS  
Linguagem de marcação para a especificação de estilos em sites.  
JavaScript  
Linguagem de programação para criar interatividades em sites.

Cada navegador pode mostrar um resultado ligeiramente diferente para listas de definição, mas o que mais importa é o significado desse tipo de listagem, pois ele relaciona diretamente os pares Termo + Descrição e isso nos ajuda bastante com mecanismos de busca.

## Hora de exercitar

Chegou a hora de acessar o endereço do nosso repositório público em <https://gustavoguanabara.github.io/html-css/exercicios/> e executar o **exercício 009** no seu computador. Agora tente atingir esse mesmo resultado em casa, sem copiar o código que eu criei. Nesse momento, a prática é algo que você mais precisa. Se por acaso ficar difícil, pode acessar o repositório público de HTML e CSS e dar uma olhada nos comandos, mas **EVITE COPIAR**.



# **Eu já falei sobre isso no YouTube?**

Eu sei que às vezes as pessoas gostam mais de assistir vídeos do que ler livros, e é por isso que eu lanço há anos materiais no canal Curso em Vídeo no YouTube. O link que vou compartilhar contigo tem o conteúdo explicado como você leu aqui, só que de forma mais ilustrada. Reserve um tempo dos seus estudos para assistir esse vídeo todo.



Curso em Vídeo: [https://www.youtube.com/playlist?list=PLHz\\_AreHm4dlAnJ\\_jJtV29RFxnPHDuk9o](https://www.youtube.com/playlist?list=PLHz_AreHm4dlAnJ_jJtV29RFxnPHDuk9o)

# DevWeb

## Capítulo 10

# Ligações em toda parte

O link ou ligação é a essência fundamental do hipertexto. Ela nos permite ligar documentos entre si e permitir a navegação entre essas páginas. Nesse capítulo vamos aprender a configurar as âncoras para vários comportamentos, desde o mais simples que é criar um link interno até ligações que habilitam downloads de arquivos.



Você tem todo o direito de usar esse material para seu próprio aprendizado. Professores também podem ter acesso a todo o conteúdo e usá-lo com seus alunos. Porém todos os que usarem esse material - seja para qual for a finalidade - deverão manter a referência ao material original, criado pelo **Prof. Gustavo Guanabara** e disponível no endereço do seu repositório público <https://github.com/gustavoguanabara/>. Este material não poderá ser utilizado em nenhuma hipótese para ser replicada - integral ou parcialmente - por autores/editoras para criar livros ou apostilas, com finalidade de obter ganho financeiro com ele.



# Joga a âncora, marujo



Os *hyperlinks* são um dos conceitos mais antigos da história da linguagem HTML. Eles permitem que você ligue um ponto a outro na *World Wide Web*. Toda vez que você está acessando um site e clica em um local para ir para outra página, outro site ou até para baixar um arquivo, você está interagindo com um *hyperlink*.



Até os mecanismos de busca se utilizam dos hyperlinks de um site. O **Google**, por exemplo, para achar um determinado site, fica vasculhando constantemente todos os outros sites da Internet procurando por links para descobrir novos conteúdos. Por isso é tão importante conseguir links válidos de outros sites para o nosso próprio site.



**APRENDA MAIS:** Veja com mais detalhes como funcionam os algoritmos fundamentais de busca assistindo esse vídeo do próprio **Google**, onde o engenheiro **Matt Cutts** explica o mecanismo básico da ferramenta mais valiosa do mundo. Habilite as legendas em PT-BR.



Google: <https://youtu.be/BNHR6IQJGZs>

Para criar um hyperlink, devemos criar **âncoras** através da tag `<a>`. O principal atributo dessa tag é o `href`, que cria uma referência hipertexto. Vamos ver um exemplo simples:

```
<h1>Vamos criar um link</h1>
<a href="https://gustavoguanabara.github.io">Acesse meu perfil GitHub</a>
```

Note que dentro do atributo `href`, o que colocamos foi uma **URL** completa para outro site.



**NÃO ENTENDEU?** Você sabe o que é uma URL? Pois volte para o nosso **capítulo 02** e vá até a parte onde falamos de domínios e hospedagem. Lá explicamos melhor o que são *Uniform Resource Locators* e quais são os seus principais componentes.

Outro atributo bem útil da tag de âncora é o `hreflang`, que permite indicar qual é o idioma principal do site para onde o link está desviando o fluxo de navegação. Isso vai permitir avisar ao navegador e a softwares de tradução como lidar caso o visitante opte por traduzir automaticamente os conteúdos.

```
<a href="https://www.w3schools.com/html/" hreflang="en">
    Site da W3Schools (em Inglês)
</a>
```

# Mira no Alvo



Por padrão, sempre que um visitante clique em um *hyperlink*, o site de destino abre na mesma janela do site que continha esse link. Ou seja, o conteúdo anterior vai deixar de ser exibido para mostrar o novo conteúdo.

Esse é um comportamento desejado quando o visitante vai continuar a visitar o nosso site, apenas mudando de um documento para outro. Mas e quando um clique leva o visitante para outro site e provavelmente ele nunca mais voltará ao nosso?

Para poder controlar onde o site de destino vai abrir, podemos usar o atributo `target`, que suporta os seguintes valores:

- ▶ `_blank` vai abrir o link em uma nova janela em branco
- ▶ `_self` vai abrir o link na janela ou frame atual (padrão)
- ▶ `_top` vai desfazer todos os frames e abrir o destino no navegador completo
- ▶ `_parent` similar ao uso do `_top` em uma referência à janela mãe
- ▶ nome-do-frame caso esteja usando frames, indicar o nome da janela a abrir

Como o uso de frames é uma técnica quase em desuso, vamos nos basear apenas nas duas primeiras opções `_blank` e `_self`.

```
<a href="pagina2.html" target="_self">  
|   Continuar navegando no site  
</a>
```

```
<a href="https://gustavoguanabara.github.com" target="_blank">  
|   Abrir perfil GitHub em nova janela  
</a>
```

## Esse link é seu ou dos outros?

Existe um recurso bem interessante para links que é indicar qual é a natureza do destino usando o atributo `rel`. Esse atributo aceita vários valores, entre eles vou citar:

- ▶ `next` indica que o link é para a próxima parte do documento atual
- ▶ `prev` indica que o link é para a parte anterior do documento atual
- ▶ `author` indica que é um link para o site do autor do artigo atual
- ▶ `external` indica que é um link para outro site que não faz parte do site atual
- ▶ `nofollow` indica que é um link para um site não endossado, como um link pago

```
<a href="pagina2.html" target="_self" rel="next">  
|   Continuar navegando no site  
</a>
```

```
<a href="https://gustavoguanabara.github.com" target="_blank" rel="external">  
|   Abrir perfil GitHub em nova janela  
</a>
```



No código anterior, o primeiro link é o que chamamos de **link local** ou **link interno**, já que ele leva o visitante a outra página dentro do nosso próprio site. Note que não é necessário nem indicar a URL completa nesses casos.

Já o segundo link vai nos levar para um outro site, o que chamamos de **link externo**. Nestes casos, devemos indicar a URL completa, incluindo o protocolo http:// ou https:// e o caminho que leve à uma página específica, se for necessário.



**NAVEGANDO POR PASTAS LOCAIS:** Não sei se você conhece um pouco do mundo **Linux**, mas servidores Web normalmente rodam esse sistema operacional. Se estiver se referindo à pasta atual do servidor, pode usar ./ antes do nome do arquivo. Se quiser se referir à pasta imediatamente superior na hierarquia, use ../ para voltar um nível para a pasta mais externa. E por favor, **aprenda Linux** !

## E para fazer Downloads?

Outra coisa que aparece bastante em sites são os links para efetuar download de algum material em PDF, ou de um arquivo ZIP qualquer. A partir da versão HTML5, as âncoras receberam atributos especiais para isso. Basta fazer o link diretamente para o arquivo que se deseja efetuar o download e adicionar o atributo download com o valor configurado para o nome do arquivo a ser baixado e o atributo type para indicar ao navegador que tipo de arquivo está sendo baixado. Vamos ver um exemplo:

```
<a href="arquivos/meulivro.pdf" download="meulivro.pdf" type="application/pdf">
| Baixe aqui o PDF do meu livro
</a>
```

Aqui vêm alguns *media types* bem usados no nosso dia-a-dia:

- |                   |              |              |
|-------------------|--------------|--------------|
| ▶ application/zip | ▶ video/mp4  | ▶ audio/mpeg |
| ▶ text/html       | ▶ video/H264 | ▶ font/ttf   |
| ▶ text/css        | ▶ video/JPEG | ▶ image/jpeg |
| ▶ text/javascript | ▶ audio/aac  | ▶ image/png  |



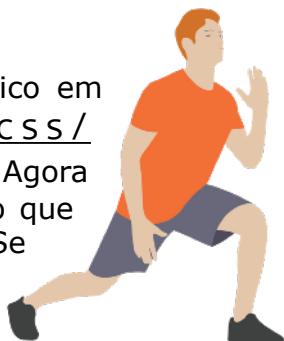
**LISTA DE MEDIA TYPES:** Se você quer saber o que escrever dentro do atributo type de uma âncora de hypertext, consulte a lista oficial da IANA.org disponível no link abaixo.

<https://www.iana.org/assignments/media-types/media-types.xhtml>



# Hora de exercitar

Chegou a hora de acessar o endereço do nosso repositório público em <https://gustavoguanabara.github.io/html-css/exercicios/> e executar o **exercício 010** no seu computador. Agora tente atingir esse mesmo resultado em casa, sem copiar o código que eu criei. Nesse momento, a prática é algo que você mais precisa. Se por acaso ficar difícil, pode acessar o repositório público de HTML e CSS e dar uma olhada nos comandos, mas **EVITE COPIAR**.



## Tenho um desafio pra você!

Lá no repositório, além do material em PDF e dos códigos dos exercícios 100% disponíveis, também disponibilizamos alguns **desafios** que devem ser resolvidos. Esses desafios não incluem o código original e você deve tentar chegar à resposta sem copiar nenhum código.



Com todo o conteúdo que vimos até essa aula, você já pode resolver o **desafio d005** e o **desafio006**. Acesse o repositório público, abra a área do curso de HTML+CSS e clique no link de acesso aos desafios. Manda ver! Só não fica pedindo a resposta! Você consegue resolver isso sozinho(a)!

Repositório em: <https://gustavoguanabara.github.io>

## Eu já falei sobre isso no YouTube?

Eu sei que às vezes as pessoas gostam mais de assistir vídeos do que ler livros, e é por isso que eu lanço há anos materiais no canal Curso em Vídeo no YouTube. O link que vou compartilhar contigo tem o conteúdo explicado como você leu aqui, só que de forma mais ilustrada. Reserve um tempo dos seus estudos para assistir esse vídeo todo.



Curso em Vídeo: [https://www.youtube.com/playlist?list=PLHz\\_AreHm4dlAnJ\\_jJtV29RFxnPHDuk9o](https://www.youtube.com/playlist?list=PLHz_AreHm4dlAnJ_jJtV29RFxnPHDuk9o)

# DevWeb

## Capítulo 11

# Imagens Dinâmicas, Áudio e Vídeo

Fotos, áudios e vídeos são essenciais para a construção dos sites hoje em dia. Ilustrar bem o seu conteúdo é imprescindível para deixar clara a mensagem que queremos passar. Mas é preciso tomar cuidado com formatos e tamanhos, pois muita gente abrirá os sites em seus celulares e dispositivos móveis. Vamos falar sobre esse assunto no capítulo que iniciamos agora.



Você tem todo o direito de usar esse material para seu próprio aprendizado. Professores também podem ter acesso a todo o conteúdo e usá-lo com seus alunos. Porém todos os que usarem esse material - seja para qual for a finalidade - deverão manter a referência ao material original, criado pelo **Prof. Gustavo Guanabara** e disponível no endereço do seu repositório público <https://github.com/gustavoguanabara/>. Este material não poderá ser utilizado em nenhuma hipótese para ser replicada - integral ou parcialmente - por autores/editoras para criar livros ou apostilas, com finalidade de obter ganho financeiro com ele.



# Seu site deve se adaptar ao tamanho da tela

Como eu disse ali na introdução do capítulo, atualmente podemos acessar sites em todo tipo de dispositivo: smartphones, tablets, notebooks, computadores desktop, aparelhos de smart TV e muito mais. Pensando nisso, você que está começando a desenvolver sites vai precisar adaptar seu conteúdo a tamanhos de tela diferentes.

A mídia que mais sofre com esse problema de tamanho de exibição são as imagens. Uma TV pode exibir imagens gigantes, mas se usarmos essa mesma foto para ser apresentada em um celular, teremos que redimensionar forçadamente a imagem com CSS.



Porém, essa prática não vai fazer com que o tamanho (em bytes) diminua também. Isso acaba aumentando o consumo de dados em dispositivos móveis e deixando seu site muito pesado, e ninguém tem paciência para acessar site lento.



**CUIDADO!** Sites lentos aumentam a **taxa de rejeição** dos usuários e prejudicam a indexação da sua página em mecanismos de busca como o **Google**. Veja no vídeo a seguir, um especialista em SEO (otimização para mecanismos de busca) falando sobre lentidão de sites, principalmente via 3G e 4G do celular.

Portal SEO: <https://youtu.be/jWnMfvSdo1E>

## Imagens Flexíveis



Nosso primeiro passo no caminho de adaptar nosso conteúdo ao tamanho da tela vai ser aprender a gerar imagens de tamanho diferentes e a fazer o navegador carregar a imagem certa para cada situação. Para isso, devemos conhecer as tags `<picture>` e `<source>`.



Para esse exemplo, criamos as três imagens ao lado: a menor tem 300x300px, a média tem 700x700px e a maior tem 1000x1000px. Usamos o programa **GIMP**, que é um editor de imagens 100% gratuito. Essas imagens serão carregadas pelo navegador de acordo com o tamanho da janela atual. Para isso, criamos o seguinte código base:

```
<picture>
  
</picture>
```

Note que colocamos a tag `<img>` exatamente como aprendemos no **capítulo 06** do nosso material. A novidade aqui é que inserimos essa imagem dentro da tag `<picture>`, que vai concentrar as outras fontes de imagem. Por padrão, a imagem **foto-g.png** (1000x1000px) será carregada.

O problema vai começar a surgir quando a janela do navegador chegar perto dos 1000 pixels de largura, pois a foto não vai mais caber lá. Vamos agora adicionar uma linha para resolver esse problema:

```
<picture>
  <source media="(max-width: 1050px)" srcset="foto-m.png" type="image/png">
    
</picture>
```

Note que a tag `<source>` possui três atributos:

- `type` vai indicar o *media type* da imagem que usamos (veja mais informações sobre *media types* no capítulo 10)
- `srcset` vai configurar o nome da imagem que será carregada quando o tamanho indicado for atingido
- `media` indica o tamanho máximo a ser considerado para carregar a imagem indicada no atributo `srcset`.



**ATENÇÃO!** Você pode até colocar o valor exato de 1000px na propriedade `max-width`, mas vai perceber que um valor ligeiramente acima vai gerar resultados mais interessantes.

Agora, recarregue seu código e mude o tamanho da janela do navegador. Você vai perceber que a imagem muda automaticamente conforme aumentamos ou diminuímos o tamanho da tela.

Vamos continuar e acrescentar mais um `<source>` à nossa imagem:

```
<picture>
  <source media="(max-width: 750px)" srcset="foto-p.png" type="image/png">
  <source media="(max-width: 1050px)" srcset="foto-m.png" type="image/png">
    
</picture>
```

É importante que existe uma ordem entre os `<source>`, e nessa nossa configuração, os itens mais acima sejam os menores tamanhos para `max-width` e que os seguintes sejam maiores, de forma crescente. O último item dentro de `<picture>` deve ser a imagem padrão.

Faça seus testes, modifique as ordens, entenda os resultados!

# Vamos falar sobre áudio

Quando comecei a produzir conteúdo para Internet, os vídeos ainda não eram uma realidade palpável, pois a Internet brasileira era muito lenta e cara. Iniciativas como Videolog e YouTube estavam começando a surgir, mas ainda mostravam vídeos muito curtos e com qualidade sofrível.

E foi em 2006 que comecei a produzir um podcast sobre tecnologia que chegou a ser bem conhecido e ter um número significativo de downloads por cada episódio.



Me lembro bem de que na época era bem complicado colocar conteúdo em áudio em um site. Precisava da adição de bastante JavaScript, manipulação de plugins específicos e um conhecimento bem consistente em RSS. A HTML não me ajudava em nada. Mais tarde, com a evolução dos vídeos e a melhoria da banda larga nacional, acabei optando por focar meus esforços no formato vídeo, mas tenho um carinho especial pelo áudio até hoje.



**QUER ME OUVIR?** Eu também participei de dois episódios de um dos maiores podcasts do Brasil: o **Nerdcast**. Dei meus pitacos sobre as profissões de Professor e de Programador. Você pode acessar o site abaixo e conferir.

[https://jovemnerd.com.br/?podcast\\_guest=gustavo-guanabara](https://jovemnerd.com.br/?podcast_guest=gustavo-guanabara)

Com a HTML5, veio também a facilidade em compartilhar áudio nos nossos sites e sem depender necessariamente de JavaScript ou plugins extras. A partir de agora, basta uma tag `<audio>` e alguns `<source>` para fazer seu site ser capaz de tocar qualquer áudio.

```
<audio preload="metadata" controls autoplay loop>
  <source src="midia/guanacast-33.mp3" type="audio/mpeg">
  <source src="midia/guanacast-33.ogg" type="audio/ogg">
  <source src="midia/guanacast-33.wav" type="image/wav">
  <p>Seu navegador não suporta áudio <a href="midia/guanacast-33.mp3" download="guanacast-33.mp3" type="audio/mpeg"></a></p>
</audio>
```

Vamos analisar os principais atributos da tag `<audio>` antes de mais nada:

- ▶ O atributo `preload` indica se o áudio será pré-carregado ou não e aceita três valores:
  - `metadata` vai carregar apenas as informações sobre o arquivo (tamanho, tempo, informações de direitos, etc)
  - `none` não vai carregar absolutamente nada até que o usuário clique no botão play ou um script inicie a reprodução

- auto (padrão) vai carregar o arquivo de áudio inteiro assim que a página for carregada, mesmo que o usuário nunca aperte o play
- O atributo controls vai apresentar o player na tela. Caso não seja colocado na tag <audio>, o controle será transparente e o usuário não poderá interagir com ele.
- O atributo autoplay, quando inserido, vai iniciar a reprodução do áudio assim que a página for carregada.
- O atributo loop vai fazer com que o áudio seja repetido eternamente assim que terminar a sua reprodução.

Dentro da tag <audio>, adicionamos vários <source> com formatos diferentes do mesmo áudio. Coloque na parte de cima o seu formato favorito. Os demais só serão carregados caso o de cima falhe. Caso todos falhem, criamos um parágrafo que permite o download do arquivo MP3 para ouvir no player padrão do dispositivo.

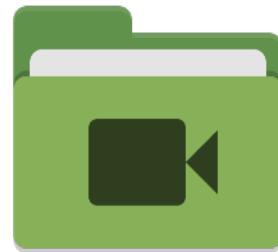


**GUANACAST:** Eu deixei no GitHub um episódio do meu antigo podcast. Basta acessar a área de exercícios de HTML, ir até a pasta ex011 e baixar o arquivo MP3 que está dentro da pasta midia.



## E os vídeos?

Com a expansão da banda larga no Brasil (sim, eu sei que ainda estamos longe de algo ideal na maioria do território nacional) tornou-se possível até assistir um vídeo em 4K em nosso plano 4G do celular (é caro, mas possível).



Para inserir um vídeo em nosso site, podemos utilizar a nova tag <video> da HTML5, caso o arquivo esteja hospedado no nosso próprio servidor.

```
<video width="600" poster="thumb.jpg" controls autoplay>
  <source src="meu-video.webm" type="video/webm">
  <source src="meu-video.mp4" type="video/mp4">
  <source src="meu-video.ogv" type="video/ogg">
  <p>Infelizmente seu navegador não conseguiu carregar o vídeo.</p>
</video>
```

Antes de mais nada, vamos criar a tag <video> e configurar alguns atributos importantes:

- width vai indicar a largura que o vídeo vai ter na tela. Nesse exemplo, 600px.
- poster configura uma imagem que vai aparecer como uma capa, enquanto o visitante não aperta o play para reproduzir o vídeo
- controls vai configurar se os controles do vídeo vão aparecer na parte inferior da mídia. Por padrão, os controles não aparecerão, mas basta colocar a palavra controls na tag <video>.
- autoplay diz para o navegador se o vídeo vai começar a tocar automaticamente, assim que a página for carregada.

# Que formatos são esses?

Arquivos de vídeo não são tão simples quanto imagens e áudios, onde o formato indica o padrão para abrir e reproduzir a mídia. Vídeos possuem formatos e codecs e isso pode tornar o vídeo inviável de ser reproduzido pela maioria dos navegadores na maioria dos dispositivos. É preciso prestar muita atenção nesse pequeno detalhe.

Os formatos suportados são MPEG, WEBM e OGG, mas os dois primeiros são os que possuem maior compatibilidade com os navegadores atualmente.



Navegador	Arquivos compatíveis
Microsoft Edge	.mp4 .m4v
Apple Safari	.mp4 .m4v
Google Chrome	.mp4 .m4v .webm .ogv
Mozilla Firefox	.webm .ogv
Opera	.webm .ogv



**UM ÓTIMO CONVERSOR:** Para gerar arquivos em vários formatos e usando codecs padronizados, recomendo usar o programa open source chamado **Handbrake**, disponível para várias plataformas.

<https://handbrake.fr/downloads.php>

## Hospedar seus próprios vídeos pode ser caro

Quando colocamos vídeos no nosso próprio servidor, podemos passar por problemas com alto consumo de banda, site lento e incompatibilidades com alguns navegadores por conta dos codecs. E geralmente só percebemos esses problemas quando colocamos nosso projeto no ar e lançamos oficialmente.

Vamos fazer uma conta simples: um vídeo simples, com poucos minutos, em formato mp4 com codec padrão deve ocupar uns 150MB com facilidade. Agora imagine que você lance seu site e 200 visitantes (um número super possível) acessem seu site e reproduzam o vídeo. Pronto, você acabou de utilizar 29GB de tráfego! Imagine o quanto isso pode deixar seu site lento ou consumir sua franquia de hospedagem, caso não seja ilimitada.



E é claro que seu site possivelmente não vai ter apenas um vídeo, não vai ter apenas 200 visitantes em um dia e aí a conta só aumenta. É preciso tomar muito cuidado quando decidimos guardar nossos próprios vídeos.

## E agora, quem poderá nos defender?

Euuuuuu! Ou melhor, o **YouTube** ou o **Vimeo**! Esses são serviços para a hospedagem de vídeos que vai evitar consumir nossos próprios recursos de host contratado. Cada um tem suas vantagens e desvantagens:



O **YouTube** é o serviço de hospedagem de vídeos mais popular do mundo e é gerenciado pelo Google. Sua principal vantagem é que seus servidores são ultra rápidos. Por outro lado, a ideia do Google é deixar todos os vídeos públicos e disponíveis, o que pode ser uma dor de cabeça caso você queira limitar quem vai ter acesso a determinado vídeo (uma escola online, por exemplo, onde queremos que apenas os alunos matriculados possam assistir).

O **Vimeo** resolve o problema que apontamos anteriormente. Ele permite limitar quem vai poder ver o vídeo, o que é especialmente vantajoso para quem quer criar produtos em forma de vídeo, entregues por demanda dentro de um site personalizado. Como desvantagens desse serviço, ele é pago por uma taxa anual e seus algoritmos não são tão eficientes quanto os do YouTube, logo os vídeos apresentam pequenos travamentos às vezes.



## E dá pra incorporar com esses aí em HTML?

Para incorporar vídeos que você subiu no **YouTube** ou **Vimeo**, existem recursos que te dão o código pronto em HTML5.

No **YouTube**, abra o vídeo que você quer incorporar e clique no link **COMPARTILHAR** que fica abaixo do título (veja imagem a seguir).

A screenshot of a YouTube video page for a video titled "Curso de HTML5 - 31 - Áudio e Vídeo em HTML5 - by Gustavo Guanabara". The video thumbnail shows Mario from Super Mario Bros. The video player interface is visible at the top. Below the video, there are standard YouTube controls: play/pause, volume, and a progress bar showing 0:23 / 16:59. To the right of the video player are several icons: a play button, a settings gear, a square, a rectangle, a person icon, and a double arrow. Underneath these are three more icons: a play button, a settings gear, and a double arrow. A red rectangular box highlights the "COMPARTILHAR" (Share) button, which is located next to the "SALVAR" (Save) button. Below the video player, there's a channel sidebar for "Curso em Vídeo" with 1,11 mi de inscritos, and two buttons: "ANALYTICS" and "EDITAR VÍDEO". At the bottom of the page, there are navigation links for "Home", "Trending", "Films", "Music", "Gaming", "Tech", "Science", "Sports", "Business", "Entertainment", "Howto", "Lifestyle", "Science", "Tech", "Business", "Entertainment", "Howto", "Lifestyle", and "Sports".

Curso de HTML5 - 31 - Áudio e Vídeo em HTML5 - by Gustavo Guanabara

150.231 visualizações

8,1 MIL

32

COMPARTILHAR

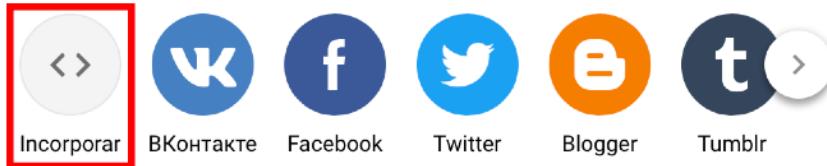
SALVAR

ANALYTICS

EDITAR VÍDEO

Ao clicar no link indicado anteriormente, uma janela vai aparecer, te dando as opções de compartilhamento. Escolha o item INCORPORAR.

Compartilhar



<https://youtu.be/SDAr3NIbIVM>

**COPIAR**

O código HTML personalizado vai aparecer em uma nova janela de contexto, incluindo um botão que permite COPIAR o código com a tag do <iframe> que vai aparecer diretamente na sua página. Volte ao seu editor de código e cole a tag no seu arquivo HTML que vai apresentar o vídeo.

Quando você está usando o Vimeo, o procedimento é bastante semelhante. Abra seu painel de controle do serviço e vá para a sua lista de vídeos hospedados. Na lista de mídias armazenadas, clique no botão com reticências e escolha a opção **Incorporação** (veja imagem abaixo). Na janela, você vai poder clicar no botão Código de incorporação e a tag <iframe> personalizada também vai aparecer.

The image shows a screenshot of the Vimeo media library. It lists three videos:

- M02A03 - W3TotalCache: configurações e teste de perfo... (duration 13:34, with 'Incorporação' button highlighted with a red box)
- M02A03 - W3TotalCache (duration 0:16:10, with 'Baixar' button)
- M02A02 - Seu WordPress muito mais rápido (duration 0:16:10, with 'Incorporação' button highlighted with a red box)

At the bottom right, there are buttons for 'Ocultar', 'Copiar link' (highlighted with a red box), and a three-dot menu icon (highlighted with a red circle).

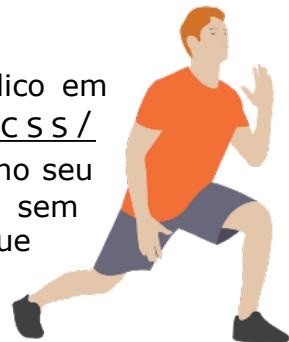
<https://vimeo.com/386829741>

**Código de incorporação**



# Hora de exercitar

Chegou a hora de acessar o endereço do nosso repositório público em <https://gustavoguanabara.github.io/html-css/exercicios/> e executar o **exercício 011** e o **exercício 012** no seu computador. Agora tente atingir esse mesmo resultado em casa, sem copiar o código que eu criei. Nesse momento, a prática é algo que você mais precisa. Se por acaso ficar difícil, pode acessar o repositório público de HTML e CSS e dar uma olhada nos comandos, mas **EVITE COPIAR**.



## Tenho desafios pra você!

Lá no repositório, além do material em PDF e dos códigos dos exercícios 100% disponíveis, também disponibilizamos alguns **desafios** que devem ser resolvidos. Esses desafios não incluem o código original e você deve tentar chegar à resposta sem copiar nenhum código.

Com todo o conteúdo que vimos até essa aula, você já pode resolver o **desafio d007**, o **desafio008** e o **desafio009**. Acesse o repositório público, abra a área do curso de HTML+CSS e clique no link de acesso aos desafios. Manda ver! Só não fica pedindo a resposta! Você consegue resolver isso sozinho(a)!



Repositório em: <https://gustavoguanabara.github.io>

## Eu já falei sobre isso no YouTube?

Eu sei que às vezes as pessoas gostam mais de assistir vídeos do que ler livros, e é por isso que eu lanço há anos materiais no canal Curso em Vídeo no YouTube. O link que vou compartilhar contigo tem o conteúdo explicado como você leu aqui, só que de forma mais ilustrada. Reserve um tempo dos seus estudos para assistir esse vídeo todo.



Curso em Vídeo: [https://www.youtube.com/playlist?list=PLHz\\_AreHm4d1AnJ\\_jJtV29RFxnPHDuk9o](https://www.youtube.com/playlist?list=PLHz_AreHm4d1AnJ_jJtV29RFxnPHDuk9o)

# DevWeb

## Capítulo 12

# Trabalhando com Estilos

Eu costumo sempre dizer que você pode ter o melhor conteúdo do mundo, mas se ele não for bem apresentado, o alcance dele diminui consideravelmente. Visitantes de sites gostam da beleza, mesmo que eles não conheçam nada de design. É uma sensação satisfatória ver um conteúdo organizado e bonito. Esse capítulo vai te mostrar os primeiros passos com o uso de CSS, aplicando esses conceitos em seus códigos.



Você tem todo o direito de usar esse material para seu próprio aprendizado. Professores também podem ter acesso a todo o conteúdo e usá-lo com seus alunos. Porém todos o que usarem esse material - seja para qual for a finalidade - deverão manter a referência ao material original, criado pelo **Prof. Gustavo Guanabara** e disponível no endereço do seu repositório público <https://github.com/gustavoguanabara/>. Este material não poderá ser utilizado em nenhuma hipótese para ser replicada - integral ou parcialmente - por autores/editoras para criar livros ou apostilas, com finalidade de obter ganho financeiro com ele.



# Você se lembra do que falamos sobre as CSS?

Nós já falamos sobre folhas de estilo em cascata, as famosas CSS no **capítulo 3**. Se por acaso você não se lembra direito, vale a pena voltar lá e dar uma segunda olhada nas definições. Vou considerar que você lembra claramente o que são as **Cascading Style Sheets** para podermos prosseguir.



Outro conteúdo muito importante que vimos está no **capítulo 8**, onde falamos sobre a íntima relação da HTML5 com a **semântica** das tags. Lá foi comentado que todo e qualquer efeito visual é responsabilidade das CSS. Vou partir daí e vamos trabalhar com os estilos, ok?

Caso você sinta qualquer dúvida a partir daqui, não se esqueça de revisitar os capítulos anteriores, pois a base foi dada gradativamente até esse momento. Vamos lá!

## A forma mais simples de aplicar estilos: CSS inline style

Vamos começar pela técnica mais básica para aplicar estilos em áreas pontuais em nosso site, que é usando as CSS dentro de parâmetros de HTML5. Crie mais uma pasta dentro da sua área de **exercícios** e crie um arquivo `index.html` com aquele código base que já fizemos várias vezes. Dentro da área `<body>`, crie um código como apresentado a seguir:

```
8  <body>
9    <h1>Capítulo 1</h1>
10   <h2>Capítulo 1.1</h2>
11   <p>Lorem, ipsum dolor sit amet consectetur adipisicing elit. Beatae
12     assumenda eveniet odit accusantium distinctio saepe.</p>
13   <h2>Capítulo 1.2</h2>
14   <p>Lorem ipsum, dolor sit amet consectetur adipisicing elit.
15     Necessitatibus doloribus pariatur deserunt in nobis labore aliquam
16     eos.</p>
17   <h1>Capítulo 2</h1>
18   <h2>Capítulo 2.1</h2>
19   <p>Lorem ipsum dolor sit amet consectetur adipisicing elit. Totam,
20     nobis quas! Eum saepe temporibus!</p>
21 </body>
```

Agora abra o arquivo recém criado no Google Chrome. O resultado visual deve ser semelhante ao apresentado a seguir:

# Capítulo 1

## Capítulo 1.1

Lorem, ipsum dolor sit amet consectetur adipisicing elit. Beatae assumenda eveniet odit accusantium distinctio saepe.

## Capítulo 1.2

Lorem ipsum, dolor sit amet consectetur adipisicing elit. Necessitatibus doloribus pariatur deserunt in nobis labore aliquam eos.

# Capítulo 2

## Capítulo 2.1

**RELEMBRANDO:** Não se esqueça que você pode criar esses parágrafos automáticos com texto "Lorem ipsum" apenas digitando o atalho `lorem` no VSCode.



Vamos começar nos focando na tag `<body>` e aplicando um estilo diferente ao corpo da página. Adicione o parâmetro `style` e digite as duas declarações de `font-family` e `color`, conforme apresentado a seguir:

```
<body style="font-family: Arial, Helvetica, sans-serif; color: #blue;">
```

Muito cuidado na hora de digitar esse código. Tudo deve ser seguido exatamente como fizemos acima, inclusive com letras maiúsculas e minúsculas. Não esqueça de adicionar os ponto e vírgulas para separar as declarações. Seu resultado visual deve ser esse:

# Capítulo 1

## Capítulo 1.1

**RELEMBRANDO:** Não se esqueça que você pode criar esses parágrafos automáticos com texto "Lorem ipsum" apenas digitando o atalho `lorem` no VSCode.

## Capítulo 1.2

**RELEMBRANDO:** Não se esqueça que você pode criar esses parágrafos automáticos com texto "Lorem ipsum" apenas digitando o atalho `lorem` no VSCode.

# Capítulo 2

## Capítulo 2.1

**RELEMBRANDO:** Não se esqueça que você pode criar esses parágrafos automáticos com texto "Lorem ipsum" apenas digitando o atalho `lorem` no VSCode.

Note que o formato da letra mudou (era Times e ficou em Arial) e a cor da fonte também foi alterado para **azul**. Se por acaso alguma dessas duas alterações não funcionou corretamente com você, confira seu código, pois algo foi digitado incorretamente. Lembre-se que o computador não é tão inteligente quanto você pode pensar. Temos que dar ordens bem claras e seguindo sempre as regras para que ele nos obedeça.



**CUIDADO!** Se por acaso você aprendeu em algum momento a tag `<font color="blue">` e acha muito mais prática usá-la, saiba que ela **NÃO É MAIS ACEITA** para as especificações da HTML5!

Nós falamos sobre especificações obsoletas de HTML no **capítulo 3**. Se precisar relembrar, volte lá e faça uma revisão do conteúdo.

Vamos fazer mais uma alteração, dessa vez na linha do primeiro título `<h1>` do nosso código:

```
<h1 style="color: ■red;">Capítulo 1</h1>
```

O resultado visual deve ser:

## Capítulo 1

### Capítulo 1.1

Lorem, ipsum dolor sit amet consectetur adipisicing elit. Beatae assumenda eveniet odit accusantium distinctio saepe.

### Capítulo 1.2

Lorem ipsum, dolor sit amet consectetur adipisicing elit. Necessitatibus doloribus pariatur deserunt in nobis labore aliquam eos.

## Capítulo 2

### Capítulo 2.1

Lorem ipsum dolor sit amet consectetur adipisicing elit. Totam, nobis quas! Eum saepe temporibus!

Note que apenas o **Capítulo 1** ficou vermelho, o **Capítulo 2** - que também é um `<h1>` - não teve alteração alguma. Isso acontece pois estamos fazendo **configurações pontuais** usando CSS.

# Estilizando de maneira mais interessante: CSS internal style

Para aplicar estilos de forma mais dinâmica e prática, podemos adicionar uma tag `<style>` dentro da área `<head>` do nosso documento HTML local. Volte lá no seu VSCode, e adicione o código dentro de `<head>`.

```
3 <head>
4   <meta charset="UTF-8">
5   <meta name="viewport" content="width=device-width,
6     initial-scale=1.0">
7   <title>Estilos pontuais</title>
8   <style>
9     body {
10       font-family: Arial, Helvetica, sans-serif;
11       background-color: #lightcyan;
12       color: #blue;
13     }
14     h1 {
15       color: #green;
16     }
17 </style>
18 </head>
```



**ATENÇÃO!** A tag `<style>` deve estar dentro da área `<head>` do seu documento HTML5. Se você colocá-la em qualquer outro local, como dentro da tag `<body>`, o resultado até pode funcionar, mas seu código estará fora dos padrões estabelecidos pela W3C. Siga sempre as regras!

Feitas as alterações, vamos ver o resultado e uma dúvida vai surgir:

## Capítulo 1

### Capítulo 1.1

Lorem, ipsum dolor sit amet consectetur adipisicing elit. Beatae assumenda eveniet odit accusantium distinctio saepe.

### Capítulo 1.2

Lorem ipsum, dolor sit amet consectetur adipisicing elit. Necessitatibus doloribus pariatur deserunt in nobis labore aliquam eos.

## Capítulo 2

### Capítulo 2.1

Lorem ipsum dolor sit amet consectetur adipisicing elit. Totam, nobis quas! Eum saepe temporibus!



Você sabe explicar por que o **Capítulo 1** ficou **vermelho** e não **verde**, como solicitamos?

Isso acontece porque as configurações pontuais (HTML style) vão prevalecer sobre as configurações gerais (CSS style). Volte ao seu código e remova todas as configurações de estilo que fizemos nas tags `<body>` e `<h1>` no início do capítulo.



**QUERO MAIS CORES:** Se você está achando que essa coisa de colocar cor pelo nome em Inglês é algo limitado, você está coberto de razão! No próximo capítulo, vou te mostrar outras técnicas de representar cores em CSS. Aquarede e confie!

# A técnica mais versátil: CSS external style

Manter as folhas de estilo fora do código HTML, além de uma maior organização faz com que tudo seja reaproveitado de maneira mais eficiente nas outras páginas do nosso site. Para isso, utilizamos a tag `<link>` especialmente configurada para trabalhar com arquivos externos de estilo. Essa tag deve ser colocada dentro da área `<head>` do seu documento HTML.

```
<html lang="pt-br">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>CSS external</title>
  <link rel="stylesheet" href="style.css">
</head>
```

A linha com o `<link>` pode estar em qualquer linha, contanto que seja dentro da área `<head>`. Particularmente, sempre procuro adicionar essa configuração após a tag `<title>` do documento atual.

## Dica para criar CSS externo com VSCode

O Visual Studio Code sempre trás algumas facilidades para o nosso dia-a-dia. Vá até o final da linha com o `<title>` e pressione Enter para criar uma nova linha. Depois comece digitando a palavra `link`, sem as marcas de tag (veja a imagem ao lado).

No menu de contexto que vai aparecer, escolha a opção `link:css` e a linha apresentada abaixo será magicamente preenchida.

```
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content=
  <title>CSS external</title>
  link
</head> link
link:atom
link:css
link:favicon
link:im
link:import
```

```
<title>CSS external</title> Siga o link (cmd + clique)
<link rel="stylesheet" href="style.css">
```

Agora passe o mouse sobre o nome do arquivo style.css e veja que existe um atalho "Siga o link", bastando pressionar Ctrl+clique (ou Cmd+clique, se estiver usando o sistema MacOS). A primeira vez que você segurar o Ctrl/Cmd e clicar sobre o link, o VSCode vai perguntar se você quer que ele crie o arquivo pra você. Clique em Sim ou Ok para aceitar a ajuda e seu arquivo será criado automaticamente.

Agora é só adicionar os seletores e todas as suas respectivas declarações nesse arquivo separado para que elas possam ser aplicadas ao documento que contiver um <link> para ele.

```
1 @charset "UTF-8";
2 /* Regra @charset caso haja problemas com acentuação */
3
4 body {
5     background-color: #lightgoldenrodyellow;
6     color: #chocolate;
7     font-size: 12pt;
8     font-family: Arial;
9 }
10
11 h1 {
12     color: #brown;
13 }
14
15 a {
16     color: #brown;
17     text-decoration: none;
18 }
```

Na **linha 1**, colocamos uma **regra** em CSS, que vai indicar a compatibilidade de codificação com o padrão UTF-8, assim como fizemos com o arquivo HTML5. Essa linha não é obrigatória e normalmente nem vai aparecer na maioria dos seus arquivos de configurações de estilo, mas caso você comece a ter problemas de compatibilidade com alguns caracteres, saiba que ela existe.

Na **linha 2**, adicionamos um **comentário** para facilitar a documentação do arquivo. Os comentários - assim como vimos em HTML - só servirão de explicação para que o desenvolvedor entenda o funcionamento de uma determinada linha ou trecho de código. O navegador não vai considerar nada que está entre os símbolos /\* e \*/ em CSS.

Nas demais linhas, fizemos as configurações dos seletores, da mesma maneira que criamos com as outras duas técnicas apresentadas no capítulo.



**TÁ CONFUSO?** Se você não está entendendo claramente todas as declarações, não se preocupe! Nos próximos capítulos nós vamos nos aprofundar nelas. Foque agora nas técnicas de uso das CSS.

# Qual técnica eu escolho pra usar?

Nesse capítulo, aprendemos as três técnicas de uso de folhas de estilo em cascata: **inline**, **interna** e **externa**. Mas em que situações devemos escolher cada um dos formatos?

Para falar com toda a sinceridade, a técnica **CSS inline style** deve ser evitada ao máximo. Ela acaba deixando seu código meio confuso, misturando a parte HTML e CSS em uma mesma linha. Mas se é para citar um momento específico em que podemos aplicar estilos inline em nossos códigos, use apenas em configurações muito pontuais e que não serão mais usadas em nenhum outro momento.



Já a técnica **CSS internal style** organiza melhor seu código, separando conteúdo e estilo em duas áreas bem definidas do seu documento. Use essa técnica quando for criar páginas isoladas com estilos próprios, que não serão replicados em outras páginas. Opte também por essa técnica apenas se a quantidade de configurações de estilo for pequena/média. Usar muitos seletores com muitas declarações vai fazer com que seu arquivo `.html` fique muito grande e seu conteúdo seja visualmente jogado lá pra baixo, dificultando manutenções futuras.

Por fim, opte sempre pela técnica **CSS external style** sempre que seu estilo for usado em várias páginas dentro do seu site. Usando a tag `<link>` em várias páginas, você pode compartilhar o mesmo estilo entre elas e não vai precisar ficar alterando vários arquivos quando o seu cliente solicitar uma pequena mudança no tom de uma determinada cor, por exemplo.

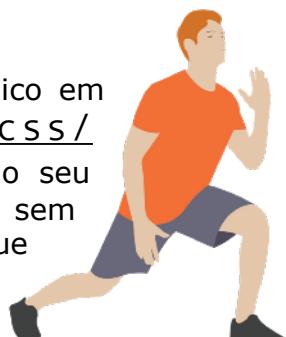
De forma resumida, guarde isso na sua cabeça:

CSS externo = use sempre que puder  
CSS interno = use para pequenas configurações  
CSS inline = procure evitar

Ainda é possível misturar as três técnicas, criando um CSS externo para as configurações globais, CSS interno para as configurações locais de um documento e CSS inline para pequenas configurações pontuais.

## Hora de exercitar

Chegou a hora de acessar o endereço do nosso repositório público em <https://gustavoguanabara.github.io/html-css/exercicios/> e executar os **exercícios 013, 014 e 015** no seu computador. Agora tente atingir esse mesmo resultado em casa, sem copiar o código que eu criei. Nesse momento, a prática é algo que você mais precisa. Se por acaso ficar difícil, pode acessar o repositório público de HTML e CSS e dar uma olhada nos comandos, mas **EVITE COPIAR**.



# **Eu já falei sobre isso no YouTube?**

Eu sei que às vezes as pessoas gostam mais de assistir vídeos do que ler livros, e é por isso que eu lanço há anos materiais no canal Curso em Vídeo no YouTube. O link que vou compartilhar contigo tem o conteúdo explicado como você leu aqui, só que de forma mais ilustrada. Reserve um tempo dos seus estudos para assistir esse vídeo todo.



Curso em Vídeo: [https://www.youtube.com/playlist?list=PLHz\\_AreHm4dlAnJ\\_jJtV29RFxnPHDuk9o](https://www.youtube.com/playlist?list=PLHz_AreHm4dlAnJ_jJtV29RFxnPHDuk9o)

# DevWeb

## Capítulo 13

### O poder das cores

Boa parte da apresentação de um determinado conteúdo parte da escolha das cores e fontes de uma página. Quando sabemos escolher bem uma paleta de cores que harmonize com o nosso conteúdo, já conseguimos dar o primeiro passo no caminho de um site bonito. Nesse capítulo, vou te mostrar as principais dicas para você começar a aplicar estilo aos seus projetos.



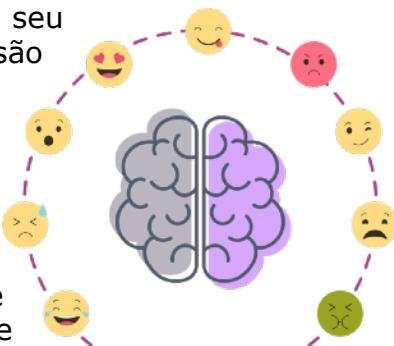
Você tem todo o direito de usar esse material para seu próprio aprendizado. Professores também podem ter acesso a todo o conteúdo e usá-lo com seus alunos. Porém todos os que usarem esse material - seja para qual for a finalidade - deverão manter a referência ao material original, criado pelo **Prof. Gustavo Guanabara** e disponível no endereço do seu repositório público <https://github.com/gustavoguanabara/>. Este material não poderá ser utilizado em nenhuma hipótese para ser replicada - integral ou parcialmente - por autores/editoras para criar livros ou apostilas, com finalidade de obter ganho financeiro com ele.



# A emoção das cores

Jamais subestime o poder das cores, elas podem influenciar na quantidade de tempo que seu visitante passa visitando o seu site e pode até mesmo ser um poderoso critério de decisão para uma compra.

Segundo um dos grandes especialistas na área de otimização de conteúdos **Neil Patel** (guarde bem esse nome, você provavelmente vai voltar a ouvir sobre ele) em seu artigo "Como cores afetam conversões" afirma que as pessoas levam cerca de 90 segundos para decidir se querem ou não um produto, e que 90% dessa decisão se baseia na sua cor.



As cores passam emoção para o subconsciente das pessoas, mesmo que na maioria dos casos isso não seja feito de forma totalmente consciente. Percebemos as cores e sentimos a sua emoção mesmo sem ter a plena certeza de que alguém usou a **psicologia das cores** para modelar um site ou produto.

Se você fizer uma breve busca pelo termo "psicologia das cores", vai ver várias sugestões de emoções para determinada cor. O **azul**, por exemplo, acaba nos remetendo a *harmonia, equilíbrio, confiança, profissionalismo, integridade e segurança*. Agora dê uma breve olhada nos logotipos do Facebook, Twitter, LinkedIn, Dell, HP e Intel. O que elas têm em comum? Será que isso é só coincidência?

Peguei o azul como exemplo principal, pois ela é citada como a cor favorita entre homens (46%) e mulheres (44%) e também é a cor com a menor taxa de rejeição (entre 1% e 2%).



**APRENDA MAIS.** A seguir, vou colocar alguns links para artigos onde você vai poder ver mais sobre a emoção das cores. Consuma esses conteúdos para entender melhor o poder das cores.

- <https://rockcontent.com/blog/psicologia-das-cores/>
- <http://www.matildefilmes.com.br/psicologia-das-cores-guia-avancado-para-profissionais/>
- <https://neilpatel.com/br/blog/psicologia-das-cores-como-usar-cores-para-aumentar-sua-taxa-de-conversao/>

Mas **muita atenção** ao seguir guias de cores e artigos, pois eles não devem ser considerados como uma verdade absoluta para todos os mercados e situações. Constantemente vemos casos de marcas que adotam uma determinada paleta de cores totalmente não recomendada por esses padrões e acabam fazendo muito sucesso. Meu sincero conselho: considere as recomendações, mas não se prenda a elas. Com isso na mente, acompanhe algumas sugestões de aplicação de algumas das cores mais usadas em sites.



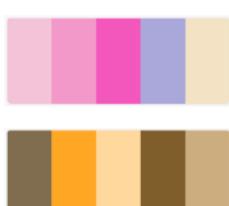
Cor	Associada a	Usar em	Evitar
vermelho	amor, emoção, energia, raiva, perigo	comida, moda, entretenimento, serviços de emergência e saúde	luxo, natureza, serviços em geral
amarelo	felicidade, alegria, otimismo, covardia	dar luz, dar calma e felicidade, chamar atenção	pode indicar que algo é barato ou spam
laranja	divertimento, ambição, calor, cautela	comércio eletrônico, entretenimento, call-to-action	pode se tornar cansativo se muito explorado
verde	saúde, natureza, dinheiro, sorte, inveja	relaxamento, turismo, financeiros, meio ambiente	luxo, tecnologia, meninas adolescentes
azul	competência, sabedoria, calma, frio	tecnologia, medicina, ciências, governo	comida (reduz apetite)
roxo	criatividade, poder, sabedoria, mistério	produtos de beleza, astrologia, ioga, espiritualidade, adolescente	não prende muito a atenção, indiferente
marrom	terra, robustez, estabilidade, amizade	alimentação, imobiliária, animais, finanças	cor considerada conservadora
preto	elegância, autoridade, mistério, morte	luxo, moda, marketing, cosméticos	desconforto e medo
branco	pureza, limpeza, felicidade, segurança	medicina, saúde, tecnologia, luxo (com preto, ouro, cinza)	não chama atenção, deve ser combinado
cinza	formalidade, sofisticação, frieza, indiferença	bens de luxo, efeito calmante	dá a sensação de frieza
rosa	amor, romance, sinceridade, cuidados	produtos femininos e cosméticos	pode tornar muito sentimental e doce

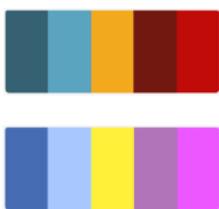
# Achei bonito, mas não sei explicar o motivo

Você provavelmente já olhou para um belo site ou para uma peça de propaganda bem produzida, teve aquela sensação de que tudo está em perfeita harmonia, mas não sabe explicar o porquê do seu cérebro perceber toda essa beleza e te fazer se sentir bem.



Pois saiba que boa parte de toda essa percepção que temos é por conta das cores e da simetria geométrica que aconteceu durante o planejamento desse site/propaganda. Um designer vai decidir qual



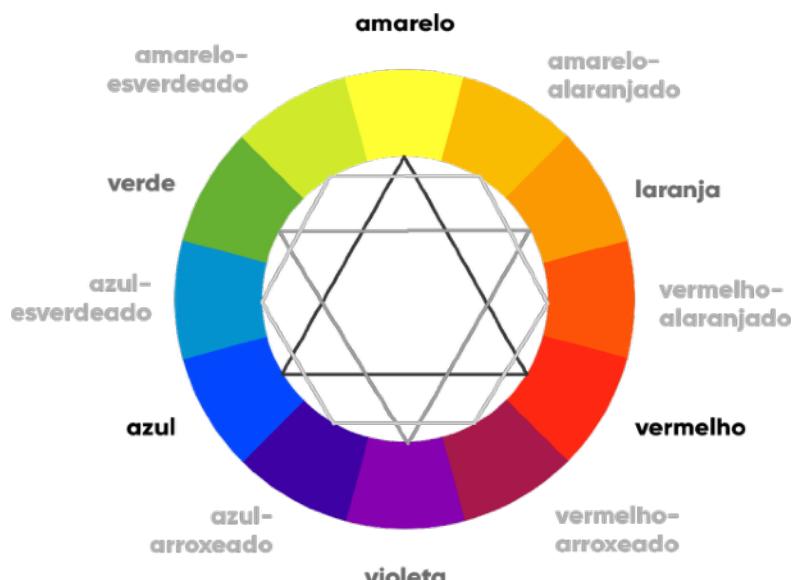


será a **paleta de cores** usada e fazer tudo fazer sentido quando as pessoas olharem para o resultado. Por exemplo, olhe para as cores apresentadas em cada linha ao lado. Elas são cores que fazem sentido quando usadas em conjunto. Você é capaz de perceber que elas possuem uma certa "harmonia" e talvez não saiba que existe toda uma ciência por trás disso. E é exatamente sobre isso que começaremos a falar daqui pra frente. Preciso antes te apresentar um amigo meu: o círculo cromático.

## O círculo cromático

Dentro da teoria das cores, precisamos separá-las em grupos para que possamos decidir se as escolhas que vamos fazer para o nosso site vão fazer um sentido harmônico e para que os nossos visitantes olhem para o nosso projeto e instintivamente pense: "- nossa, que bonito!".

A base para isso é conhecer o círculo cromático e compreender as suas sub-divisões. E ele está logo aí abaixo, olhe atentamente, se possível para uma versão colorida. Se por acaso você está vendo uma versão impressa em preto-e-branco, acesse agora o meu repositório e veja o PDF diretamente na tela do seu computador ou celular. Vai ficar tudo mais claro pra você, pode acreditar!



Analizando atentamente o círculo cromático, percebemos as três **cores primárias**, que estão destacadas com o texto mais escuro: **amarelo**, **vermelho** e **azul**.

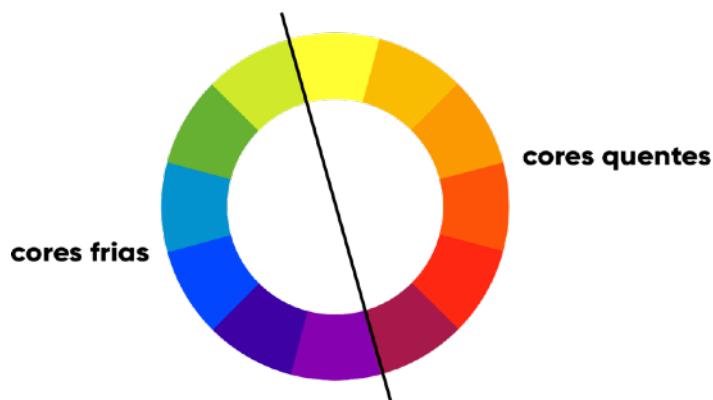
Da junção das cores primárias, temos as três **cores secundárias**, que são o **laranja** (amarelo+vermelho), o **violeta/roxo** (azul+vermelho) e o **verde** (azul+amarelo).

Da junção de uma cor primária com uma secundária, temos as seis **cores terciárias**:

- **Amarelo-esverdeado** (amarelo+verde)
- **Amarelo-alaranjado** (amarelo+laranja)
- **Vermelho-alaranjado** (vermelho+laranja)
- **Vermelho-arroxeadoo** (vermelho+roxo)
- **Azul-arroxeadoo** (azul+roxo)
- **Azul-esverdeado** (azul+verde)

# Temperatura e Harmonia

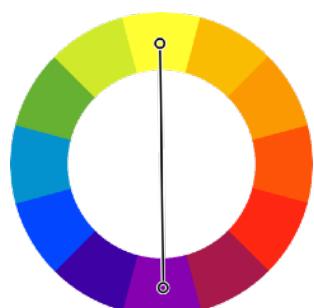
Olhando o círculo cromático, também conseguimos classificar as cores por sua temperatura. Dá só uma olhada na imagem a seguir:



As cores quentes, criam uma sensação de calor e proximidade. Já as cores frias, estão associadas a sensações mais calmas, de frescor e tranquilidade.

Além da classificação por temperatura, podemos classificar as cores por esquemas harmônicos.

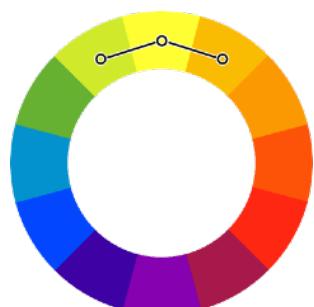
## Cores complementares



São aquelas que apresentam o maior contraste entre si. Elas estão localizadas do lado imediatamente oposto do círculo cromático.

Se pegarmos qualquer cor primária, a sua cor complementar é sempre uma cor secundária. De forma similar, qualquer cor terciária tem uma outra cor terciária como complementar. Quando juntamos duas cores complementares, sempre obtemos o cinza.

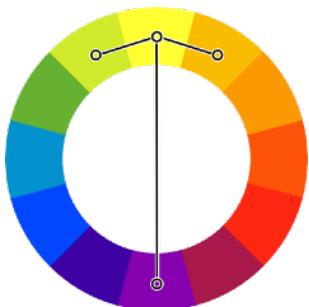
## Cores análogas



Diferente das cores complementares, que estão do lado extremo oposto no círculo cromático, as cores análogas são aquelas que são imediatamente vizinhas entre si.

Por serem cores consecutivas, as cores análogas possuem um baixo contraste entre elas, mas criam uma bela harmonia quando combinadas em um mesmo design.

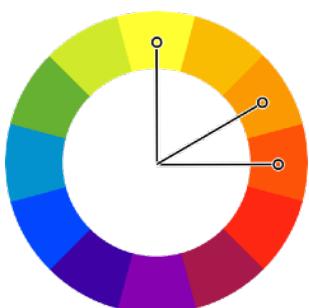
## Cores análogas mais uma complementar



Dá pra notar que essa aqui é uma combinação dos dois tipos anteriores, não é?

Essa técnica quebra um pouco o ritmo semelhante das cores análogas, adicionando uma cor que cria um grande contraste com as três análogas.

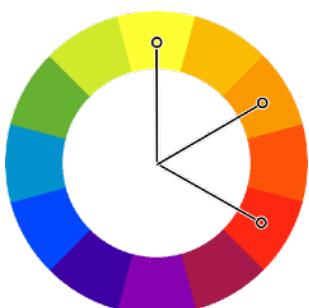
## Cores análogas relacionadas



Nesse tipo de harmonia, escolhemos duas cores análogas (consecutivas) e depois pulamos uma terceira cor (em qualquer direção) e escolhemos a quarta.

Com essa técnica, conseguimos um resultado parecido com o das cores análogas simples, mas com um pouco mais de contraste sem ter que escolher uma cor complementar.

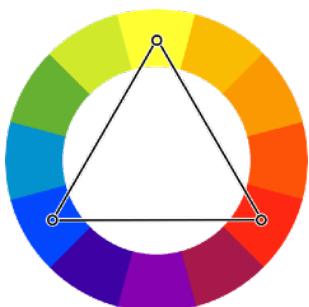
## Cores intercaladas



Um tipo menos usado de harmonia, já que às vezes não funciona tão bem assim. Vamos escolher a primeira cor e depois mais duas com intervalo constante entre elas.

Na imagem ao lado, criei um exemplo onde o intervalo é constante entre as cores selecionados.

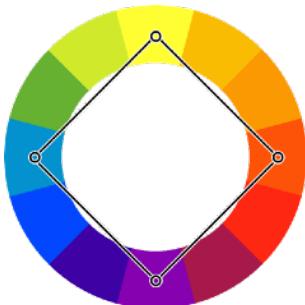
## Cores triádicas



Técnica bastante utilizada e que garante uma grande riqueza de cores, onde escolhemos três pontos equidistantes no círculo cromático.

Esse esquema gera sempre um triângulo equilátero e cria uma opção que sempre possui um ótimo contraste entre as cores.

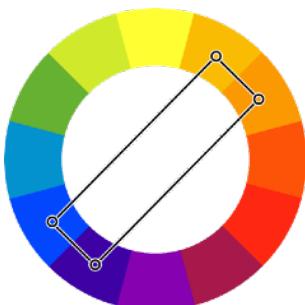
## Cores em quadrado



Bastante semelhante ao esquema triádico, mas permite selecionar quatro cores com um contraste razoável entre as cores escolhidas.

Esse esquema gera sempre um quadrado perfeito com os pontos selecionados.

## Cores tetrádicas



Com essa técnica, vamos escolher dois pares de cores complementares, que não serão necessariamente análogas ou consecutivas. Isso vai nos garantir dois pares de cores, com bastante contraste entre si.

## Monocromia

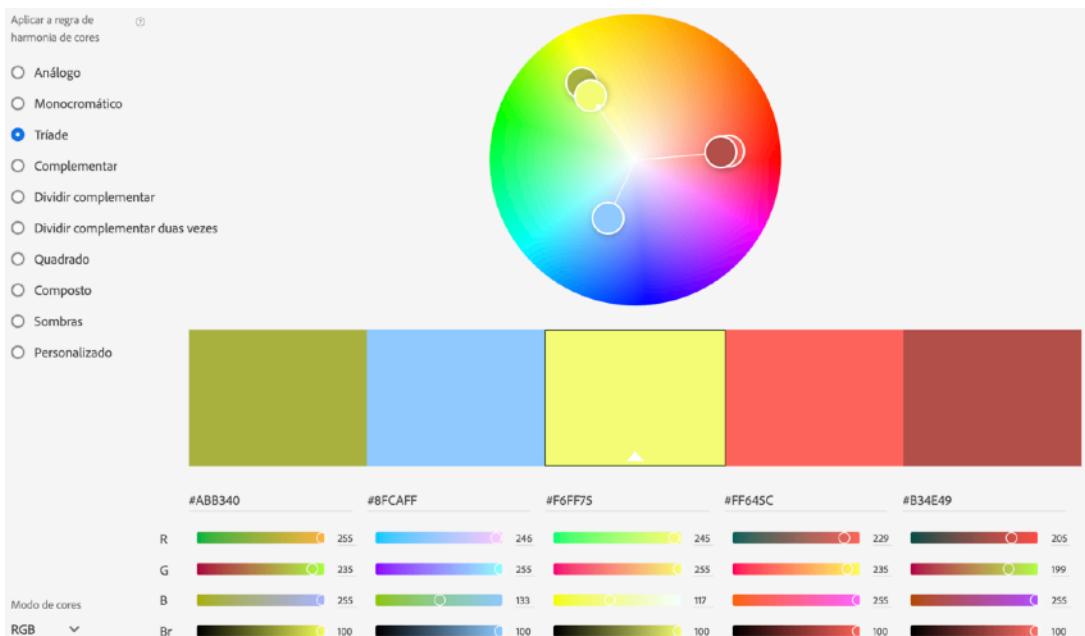


Uma harmonia bem diferente das anteriores, que usa apenas uma cor e varia apenas a sua saturação e o seu brilho. Essa combinação geralmente gera pouquíssimo contraste entre as cores escolhidas, mas acaba gerando um resultado visual bem agradável aos olhos, conhecido como “degradê”.

# E onde eu aplico esse conhecimento na prática?

Existem vários sites e serviços que vão te ajudar na escolha da paleta de cores do seu site. A que vai permitir mais opções, na minha opinião é o **Adobe Color** (disponível em <https://color.adobe.com/pt/>), que tem recursos gratuitos para te auxiliar na escolha das suas cores baseado nos esquemas de harmonia que vimos anteriormente.

No modo **Criar**, você vai escolher o modo de cores (para monitores é o RGB) e também a regra de harmonia que você quer usar. A partir daí ele vai te sugerir uma paleta com cinco cores perfeitamente harmônicas. Para mudar as tonalidades sem mudar a regra, arraste qualquer uma das cores e a regra vai se aplicar aos outros pontos. Já no modo **Explorar**, você vai ser apresentado a várias paletas prontas e vai poder copiá-las na maior cara de pau, pois tudo é grátis e liberado!

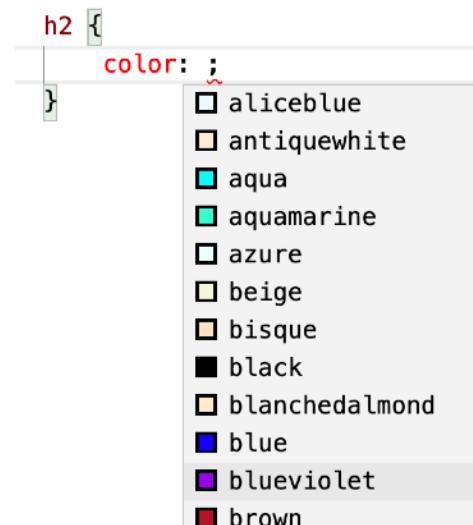


## Aplicando cores ao nosso site

Nos códigos CSS do capítulo anterior, vimos declarações voltadas para cores. Até o momento, usamos valores textuais como `blue`, `red`, `lightcyan`, e muitas outras.

No VSCode, ao criar uma propriedade relacionada a cores em CSS, podemos posicionar o cursor entre os dois pontos e o ponto e vírgula da declaração e pressionar `Ctrl+Espaço` para obter uma lista com os valores possíveis. Veja na imagem ao lado como esse recurso se comporta.

Porém, esse método de especificação de cores é muito limitado, pois uma tela moderna é capaz de exibir aproximadamente 65 milhões de cores.



Para conseguirmos mais possibilidades, devemos recorrer aos códigos hexadecimais ou então às funções CSS `rgb()`, `rgba()`, `hsl()` ou `hsla()`. Para usar esse recurso, adicione qualquer cor textual à sua propriedade e passe o mouse sobre o nome da cor (veja a imagem a seguir) e uma janela especial aparecerá.

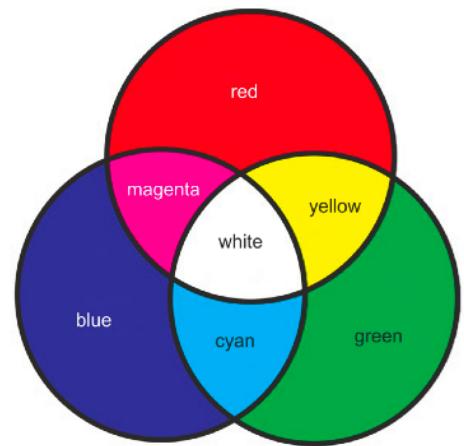
Na imagem ao lado, ao passar o mouse sobre a palavra `red`, a janela de seleção de cores vai

aparecer, permitindo escolher a cor, tom e transparência. O sistema do VSCode vai sugerir a melhor função a ser utilizada, mas você pode mudá-la clicando sobre o nome da função, também indicado na imagem.

Faça testes, experimente, mude cores, use sua criatividade. A prática leva ao aprendizado sólido e duradouro!

## Como representar uma cor?

Você já deve ter ouvido falar que as cores em uma tela são compostas da junção de três cores primárias: **vermelho** (red), **verde** (green) e **azul** (blue). Analisando a imagem ao lado, vemos que a junção de algumas cores primárias nos leva a outras cores como o **magenta**, **amarelo** e **ciano**. Se usarmos todas as cores primárias no máximo, chegamos ao **branco**. Com todas as três no mínimo, obtemos o **preto**.



Cada cor primária pode ter um valor **entre 0 e 255**, totalizando **256 possibilidades** para cada elemento.

Vejamos alguns exemplos de cores e seus respectivos códigos.

Lime #A4C400 RGB(164, 196, 0)	Green #60A917 RGB(96, 169, 23)	Emerald #008A00 RGB(0, 138, 0)	Teal #00ABA9 RGB(0, 171, 169)
Cyan #1BA1E2 RGB(27, 161, 226)	Cobalt #0050EF RGB(0, 80, 239)	Indigo #6A00FF RGB(106, 0, 255)	Violet #AA00FF RGB(170, 0, 255)
Pink #F472D0 RGB(244, 114, 208)	Magenta #D80073 RGB(216, 0, 115)	Crimson #A20025 RGB(162, 0, 37)	Red #E51400 RGB(229, 20, 0)
Orange #FA6800 RGB(250, 104, 0)	Amber #F0A30A RGB(240, 163, 10)	Yellow #E3C800 RGB(227, 200, 0)	Brown #825A2C RGB(130, 90, 44)
Olive #6D8764 RGB(109, 135, 100)	Steel #647687 RGB(100, 118, 135)	Mauve #76608A RGB(118, 96, 138)	Taupe #87794E RGB(135, 121, 78)

Vamos tomar como exemplo a cor **Teal** aqui ao lado. Seu código `rgb(0, 171, 169)` indica que existe quantidade **0** de vermelho nessa cor, **171** de verde e **169** de azul. No código de cores hexadecimal (iniciado sempre com #) indica que **00** é a quantidade de vermelho, **AB** é a quantidade de verde e **A9** é a quantidade de azul.

Esta mesma cor indicada acima, pode ser representada em CSS com um outro formato baseado na maneira como o olho humano enxerga as cores: o padrão **HSL**. A função `hsl(179, 100%, 34%)` indica que temos 179 de **hue** (matiz), 100% de **saturation** (saturação) e 34% de **lightness** (luminância).

Para obter versões de cores com transparência, basta arrastar a barra de transparência indicada à direita e perceber que mais um valor (**alpha**) será adicionado ao código.

# Usando Gradientes em CSS

Podemos gerar gradientes e aplicarmos a componentes visuais usando folhas de estilo. Vamos usar um exemplo simples no nosso exercício atual. Vá até o documento e modifique a declaração do nosso seletor body.

```
<style>
body {
    font-family: Arial, Helvetica, sans-serif;
    background-image: linear-gradient(90deg, yellow, red);
    color: black;
}
```

Pode parecer esquisito no início, mas um gradiente é considerado pelo navegador como se fosse uma imagem, por isso usamos a propriedade `background-image` na declaração CSS. A função `linear-gradient` é auto-explicativa e gera um gradiente linear angular. O primeiro parâmetro da função, indica o ângulo de inclinação de 90 graus (`90deg`) e as seguintes indicam as cores do degradê a ser criado. Você pode indicar quantas cores quiser e o navegador vai saber se virar pra gerar seu degradê personalizado. Experimente na sua casa outros valores de ângulo também, incluindo negativos (`45deg`, `-90deg`, `25deg`,...) e note as diferenças.

Também é possível gerar os chamados gradientes radiais, que também são meio auto-explicativos. Veja o exemplo:

```
background-image: radial-gradient(circle, red, yellow, green);
```

Altere o tipo de gradiente do body para usar o formato radial circular e veja o resultado. Você também pode personalizar ainda mais seu degradê colocando uma porcentagem ao lado da cor como `red 10%`, `yellow 40%`, `green 50%`. Experimente!

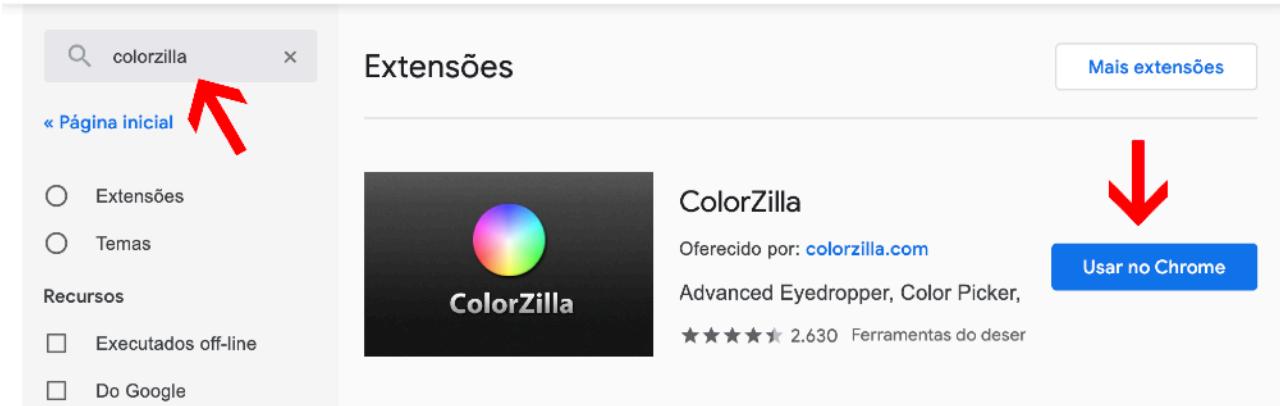
## Encontrei uma cor maravilhosa! Qual é o código dela?

Com certeza essa situação vai aparecer na sua vida, mais cedo ou mais tarde. Você vai entrar em um site e vai descobrir um tom perfeito daquele amarelo que estava procurando. Como descobrir exatamente o código dessa cor?

Uma das maneiras bem práticas de executar essa tarefa é usando uma extensão gratuita do **Google Chrome** chamada **Colorzilla**.

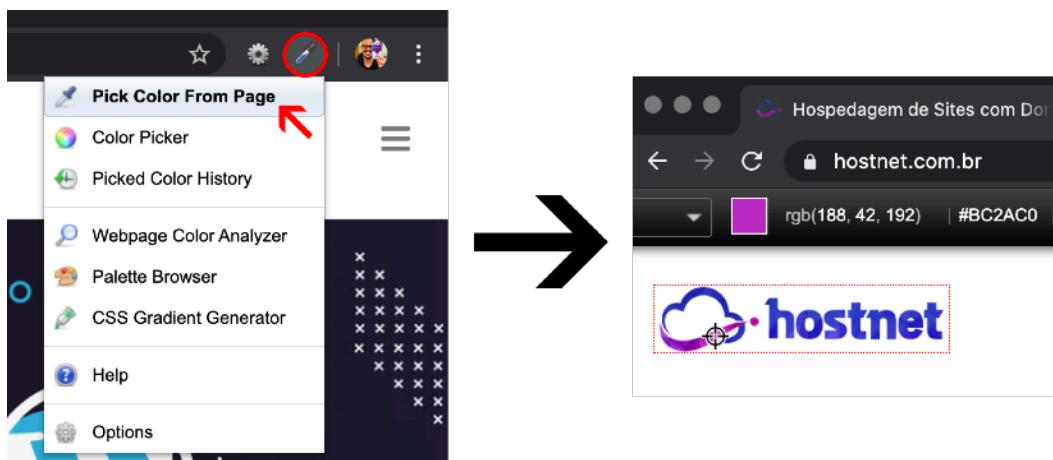


Para instalar uma extensão, abra o **Google Chrome** e acesse a **Chrome Web Store** no endereço <https://chrome.google.com/webstore/>. Na caixa de pesquisa no canto superior esquerdo, digite `colorzilla` e pressione Enter. A extensão vai aparecer no lado direito da tela e você deve clicar sobre o botão “Usar no Chrome” e clicar no botão autorizando usar a extensão (veja os passos na imagem a seguir).



A screenshot of the Chrome Web Store interface. In the search bar at the top left, the text "colorzilla" is typed. To the right of the search bar, the word "Extensões" is displayed. On the far right, there is a blue button labeled "Mais extensões". Below the search bar, there is a sidebar with links: "Extensões", "Temas", "Recursos", "Executados off-line", and "Do Google". The main content area shows the "ColorZilla" extension card. The card features a large circular icon with a color gradient, the text "ColorZilla", and the developer information "Oferecido por: colorzilla.com". Below this, it says "Advanced Eyedropper, Color Picker," and has a rating of "★★★★★ 2.630 Ferramentas do deserto". At the bottom right of the card is a blue button labeled "Usar no Chrome". A red arrow points upwards from the "Extensões" link in the sidebar towards the search bar.

Agora você vai perceber que ao lado da barra de endereço do navegador, apareceu um pequeno conta-gotas. Abra um site qualquer e clique sobre esse ícone. Em seguida, clique em "Pick Color From Page" e aí é só clicar no local que deseja capturar. A cor vai aparecer em formato `rgb()` e com seu código hexadecimal. Ao clicar, o código será copiado para a sua área de transferência.



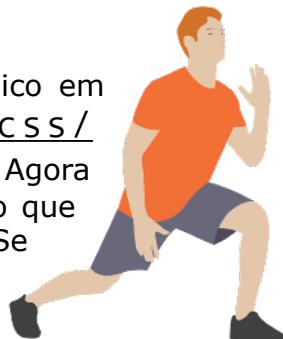
## Agora já sou especialista em CSS?

Ufa! Esse capítulo finalmente chegou ao fim. Mas não fique pensando que agora já sabe 100% das CSS. O caminho ainda é muito longo, nós só iniciamos! Nos próximos capítulos, vamos continuar estudando alguns conceitos de design, nos focando especialmente nos conceitos sobre fontes.



# Hora de exercitar

Chegou a hora de acessar o endereço do nosso repositório público em <https://gustavoguanabara.github.io/html-css/exercicios/> e executar o **exercício 016** no seu computador. Agora tente atingir esse mesmo resultado em casa, sem copiar o código que eu criei. Nesse momento, a prática é algo que você mais precisa. Se por acaso ficar difícil, pode acessar o repositório público de HTML e CSS e dar uma olhada nos comandos, mas **EVITE COPIAR**.



## Eu já falei sobre isso no YouTube?

Eu sei que às vezes as pessoas gostam mais de assistir vídeos do que ler livros, e é por isso que eu lanço há anos materiais no canal Curso em Vídeo no YouTube. O link que vou compartilhar contigo tem o conteúdo explicado como você leu aqui, só que de forma mais ilustrada. Reserve um tempo dos seus estudos para assistir esse vídeo todo.



Curso em Vídeo: [https://www.youtube.com/playlist?list=PLHz\\_AreHm4d1AnJ\\_jJtV29RFxnPHDuk9o](https://www.youtube.com/playlist?list=PLHz_AreHm4d1AnJ_jJtV29RFxnPHDuk9o)

# DevWeb

## Capítulo 14

# Vamos falar das Fontes

Sem dúvidas as cores são muito poderosas, como pudemos conferir no capítulo anterior. Mas em conjunto com elas, temos as fontes, que são um ótimo recurso visual para criar a identidade da página e mostrar a ideia que queremos passar com o nosso design. Vamos aprender um pouco mais sobre fontes e como aplicá-las aos nossos sites. Venha comigo.



Você tem todo o direito de usar esse material para seu próprio aprendizado. Professores também podem ter acesso a todo o conteúdo e usá-lo com seus alunos. Porém todos os que usarem esse material - seja para qual for a finalidade - deverão manter a referência ao material original, criado pelo **Prof. Gustavo Guanabara** e disponível no endereço do seu repositório público <https://github.com/gustavoguanabara/>. Este material não poderá ser utilizado em nenhuma hipótese para ser replicada - integral ou parcialmente - por autores/editoras para criar livros ou apostilas, com finalidade de obter ganho financeiro com ele.



# Tipografia? Que bicho é esse?

Você se lembra de que falamos no capítulo anterior que as cores podem transmitir emoções? Pois as fontes também possuem essa mesma capacidade. E se você é uma pessoa atenta, vai entender que quando somamos essas emoções, podemos ter resultados ainda mais fortes.

Para entender mais sobre as fontes, precisamos estudar os fundamentos básicos da **tipografia**, que é uma arte antiga que estuda técnicas de **escrita** (do Grego, *graphía*) para a apresentação de forma impressa (do Grego, *týpos*). Essa preocupação surgiu na época em que as grandes prensas físicas eram usadas para produzir livros/jornais. Os **tipos móveis** são aquelas peças de metal/madeira/argila (ao lado) que são usados para "carimbar" o papel e fazer as letras.



E o mundo da tipografia se inicia em 1450, com o inventor Alemão **Johannes Gutenberg** (foto ao lado), criador da prensa mecânica de tipos móveis. Na verdade, os Chineses foram os primeiros a criarem o conceito de prensa com tipos móveis, mas Gutenberg acabou sendo reconhecido como aquele que deu início à **Revolução da Imprensa**. Antes disso tudo, cada exemplar de um livro era reproduzido através de material manuscrito devidamente copiado, palavra por palavra, até atingir o resultado desejado.

## Fonte, letra e família

### Glifos, letras, caracteres

São os signos alfabéticos projetados para reprodução mecânica. O exemplo a seguir representa os glifos de **a** até **h**.

abcdefghijklm

### Família tipográfica

É o conjunto de glifos que possuem as mesmas características anatômicas, independente das suas variações.

Vou exemplificar esse conceito com o exemplo a seguir: a família tipográfica **Open Sans** possui várias configurações de peso (de 300 a 800). Mesmo parecendo representações bem diferentes, todos eles fazem parte da mesma família tipográfica.

Light 300

## Curso em Vídeo

Regular 400

## Curso em Vídeo

Semi-bold 600

## Curso em Vídeo

Bold 700

## Curso em Vídeo

Extra-bold 800

## Curso em Vídeo

# Fontes

As fontes são conjuntos de glifos que formam uma família tipográfica. O termo fonte também é aplicável ao arquivo digital que armazena todos os formatos de glifos que compõem uma determinada família tipográfica.

A 65	B 66	C 67	D 68	E 69	F 70	G 71	H 72	I 73	J 74	K 75	L 76	M 77
A	B	C	D	E	F	G	H	I	J	K	L	M
N 78	O 79	P 80	Q 81	R 82	S 83	T 84	U 85	V 86	W 87	X 88	Y 89	Z 90
N	O	P	Q	R	S	T	U	V	W	X	Y	Z
a 97	b 98	c 99	d 100	e 101	f 102	g 103	h 104	i 105	j 106	k 107	l 108	m 109
A	B	C	D	E	F	G	H	I	J	K	L	M
n 110	o 111	p 112	q 113	r 114	s 115	t 116	u 117	v 118	w 119	x 120	y 121	z 122
N	O	P	Q	R	S	T	U	V	W	X	Y	Z
0 48	1 66	2 67	3 68	4 69	5 70	6 71	7 72	8 73	9 74			
0	1	2	3	4	5	6	7	8	9			
\$ 36	* 42	+ 43	- 45	/ 47	& 38	~ 71	= 61	% 37	" 34	' 39	Ø 157	# 35
\$	*	+	-	/	&	~	=	%	"	'	ÿ	#
@ 64	_ 66	( 40	) 41	, 44	. 46	; 59	: 58	? 63	! 33	\ 92	124	{ 123
(@)	_	(	)	,	.	;	:	?	!	\		{
} 125	< 60	> 62	[ 91	] 93	^ 96	~ 94	® 166	° 167	½ 171	¼ 172	µ 230	Ø 157
}	<	>	[	]	^	~	TM	ʃ	Ω	º	μ	ÿ
À 183	Á 181	Â 182	Ã 199	à 133	á 160	â 131	ã 198	É 144	Ê 210	Í 214	í 161	Ó 224
ô 226	ö 229	ó 162	ô 147	õ 228	ú 233	û 154	ñ 165	ñ 164	ç 128	ç 135		
'	÷	1	^	~	€	>	"	Û	«	Ê		

# As fontes falam?

No capítulo anterior nós falamos sobre a importância de escolher uma boa paleta de cores para o nosso projeto. Também precisamos saber escolher as famílias tipográficas que utilizaremos em um site. As fontes também podem passar emoções. Vamos a um exemplo?

Imagine que eu tenha que representar a palavra “amor” através de uma determinada tipografia. Qual das opções abaixo você escolheria?



Amor

Amor

Com toda certeza, a maioria das pessoas escolheria a segunda opção. Sabe o por quê? A palavra “amor” tem mais a ver com uma representação mais suave e fluida, não algo mais robusto e forte. E mesmo sem te dizer isso, provavelmente a escolha do tipo fez isso por mim.

O problema é que a escolha não é tão simples assim. Na representação acima, com uma palavra curta e isolada, conseguimos ler facilmente a palavra “amor” em qualquer uma das opções de fontes escolhidas. Chamamos isso de **legibilidade**. Mas basta colocarmos uma frase maior para as coisas ficarem um pouco confusas.

*“Amor quando é amor não definha. É  
até o final das eras há de aumentar.  
Mas se o que eu digo for erro, e o meu  
engano for provado, então eu nunca terei  
escrito ou nunca ninguém terá amado.”*

*William Shakespeare*

No exemplo acima, mesmo que todas as palavras tenham uma **legibilidade** razoável (até dá pra entender), a **leitabilidade** não é tão boa assim. Essa segunda característica diz respeito à fluidez que conseguimos ter na leitura.

Sendo assim, escolher um bom tipo é essencial para cada caso. Não povoar nosso site com tipos diferentes também é uma ótima ideia. No máximo dois ou três tipos já estaria ótimo.

Vamos conhecer agora algumas características anatômicas dos tipos para nos ajudar a escolher boas fontes para nosso site.

# Anatomia do Tipo

Vamos analisar cada uma das partes de um tipo. Volte sempre para essa página ao ler a descrição de cada elemento:



1 - **Arco:** presente em letras minúsculas. Uma linha curva que nasce em na haste principal.

2 - **Barriga:** curva em uma letra maiúscula ou minúscula, fechada, ligada à haste vertical em dois pontos.

3 - **Braço:** traço horizontal ou inclinado, ligado à haste vertical principal de uma letra maiúscula ou minúscula.

4 - **Cauda:** apêndice do corpo de algumas letras (*g, j, J, K, Q, R*), que fica abaixo da linha base.

5 - **Enlace:** a forma como uma haste, linha ou filete se liga a um arremate, a uma serifa ou a um terminal. Pode ser angular ou curvilíneo.

6 - **Espinha:** curva e contracurva estrutural da letra **S**.

7 - **Esporão:** uma projeção que encontramos nas letras **b** e **G**.

8 - **Filete:** haste horizontal ou inclinada, fechada nas duas extremidades, por duas hastes ou por uma curva.

9 - **Haste:** traço principal de uma letra, geralmente vertical.

10 - **Olho:** espaço em branco, fechado, dentro de uma letra.

11 - **Orelha:** apêndice presente na letra **g**, que pode ser em gota, botão, bandeira ou gancho.

12 - **Pé:** terminal ou serifa horizontal que arremata uma perna na parte de baixo.

13 - **Perna:** haste vertical ou inclinada com um extremidade livre (ou com um pé) e outra extremidade ligada ao corpo da letra.

14 - **Serifa:** também chamada de apoio ou patilha. Pequenas retas que ornamentam as hastes de alguns tipos.



15 - **Terminal**: forma que arremata a extremidade de uma linha curva de uma letra.

16 - **Vértice**: também chamada de ápice. Formada pela convergência de duas hastas que se encontram. Pode ser pontiagudo, oblíquo, plano ou redondo.

# Categorias de fontes

Os tipos ou fontes tipográficas também são classificados por suas categorias. Elas são baseadas principalmente na presença ou ausência da serifa, o item 14 da lista anterior. As demais categorias, acabam derivando das duas principais (com e sem serifa) ou não se encaixam nessas características e por isso geram novas categorias.

## Fontes Serifadas

Esta é a categoria mais clássica de fontes, surgida lá na época das prensas que eu citei no início do capítulo. Tipicamente, os caracteres serifados sempre foram aplicados em grandes blocos de textos impressos em papel e se aproveitam de uma característica da nossa percepção: nós nunca lemos as palavras letra por letra, e sim por um conjunto. As serifas têm a capacidade de guiar nossos olhos graças aos pequenos prolongamentos que elas criam e fazem as letras “se juntarem” em palavras. A seguir, vemos quatro exemplos de fontes serifadas:

Bree Serif  
TypeTogether

Noto Serif  
Google

**Curso em Vídeo**   **Curso em Vídeo**

EB Garamond  
Georg Duffner

Bitter  
Huerta Tipográfica

**Curso em Vídeo**   **Curso em Vídeo**

Atualmente, não usamos fontes serifadas para apresentar textos longos na Web pois as tendências atuais nos levam a usar fontes um pouco mais leves visualmente. Porém, as fontes serifadas são bastante usadas em títulos, pois acabam chamando mais atenção por conta das características que citei.



**APRENDA MAIS SOBRE FONTES:** As classificações não param por aqui. Existem também sub-categorias para cada fonte serifada, como as *old style*, *transitional*, *didone*, *slab*, *clarendon* e *glyphic*, que apresentam características detalhadas para cada uma. Nesse material, nós vamos nos limitar apenas às classificações gerais por motivos práticos. Provavelmente o seu professor de *design* vai falar sobre o assunto de maneira mais aprofundada.

# Fontes não Serifadas

Mais conhecidas por seu “nome chique” em Francês *sans-serif* (significa “sem serifa”), são fontes que, como você já pode imaginar, não apresentam serifas. As primeiras fontes dessa categoria surgiram em 1816, mas foram consideradas avançadas demais para a época. Anos depois, ressurgiram em versão melhorada e vieram pra ficar, principalmente para a Web. Isso acontece porque elas são ótimas para a exibição em telas/monitores pois transmitem a sensação de limpeza, clareza e organização. Veja a seguir alguns exemplos de fontes não serifadas:

Open Sans

Steve Matteson

Oswald

Vernon Adams, Kalapi Gajjar, Cyreal

Curso em Vídeo

Titillium Web

Multiple Designers

Curso em Vídeo

Montserrat

Julieta Ulanovsky, Sol Matas, Juan Pablo del Peral,

Curso em Vídeo

Curso em Vídeo

A grande maioria dos textos que você está lendo nesse material desde o início do curso estão sendo escritos com uma fonte não serifada muito popular: a **Verdana**.

# Fontes Monoespacadas

Essa é uma das categorias de fontes que vieram derivadas das duas categorias que vimos anteriormente, por isso existem fontes monoespacadas com e sem serifas. A principal diferença desse tipo de fonte é o espaço horizontal (largura) ocupado por cada letra. Na maioria das fontes, a letra **i** ocupa muito menos espaço lateral do que a letra **M**, não é? Não para as fontes monoespacadas. Elas possuem a mesma largura para todas as letras.

Source Code Pro

Paul D. Hunt

IBM Plex Mono

Mike Abbink, Bold Monday

Curso em Vídeo

Curso em Vídeo

Roboto Mono

Christian Robertson

Ubuntu Mono

Dalton Maag

Curso em Vídeo

Curso em Vídeo

A principal vantagem no uso desse tipo de fonte é facilitar ao máximo a leitura das palavras, principalmente aquelas que requerem que você as reproduza. Usamos muito esse tipo de fonte para representar comandos de linguagens de programação de computadores. Por isso, nós também costumamos chamá-las de fonte de terminal ou fonte de console.

## Fontes Script

Também chamadas de fontes *handwriting*, são aquelas que tentam imitar a escrita humana. Seu uso deve ser bem controlado e jamais será aplicado a textos muito longos, pois causam cansaço visual e tornam-se difíceis de ler, como já provamos anteriormente no início do capítulo, dentro do item "As fontes falam".

Great Vibes  
TypeSETit

Pacifico  
Vernon Adams, Jacques Le Bailly, Botjo Nikoltchev,

*Curso em Vídeo*

*Curso em Vídeo*

Dancing Script  
Impallari Type

Reenie Beanie  
James Grieshaber

*Curso em Vídeo*

*Curso em Vídeo*

## Fontes Display

Toda fonte que foge completamente das definições feitas pelas classificações acima são consideradas fontes *display*. São fontes com bastante efeitos visuais, enfeitadas e até mesmo curiosas. Também são chamadas de fontes comemorativas e algumas delas sequer representam letras, podendo ser desenhos de animais, objetos, pessoas, personagens de quadrinhos, etc.

Lobster  
Impallari Type, Cyreal

Bungee Outline  
David Jonathan Ross

*Curso em Vídeo*

**CURSO EM VÍDEO**

Luckiest Guy  
Astigmatic

Press Start 2P  
CodeMan38

**CURSO em VÍDEO**

**Curso em Vídeo**

Essas fontes também são recomendadas para criar títulos em destaque e devem ser evitadas para textos médios ou longos.

## Como aplicar isso na prática?

Para configurar a família tipográfica que será aplicada a um determinado texto, usamos a propriedade `font-family` das CSS. Se indicarmos mais de uma família na sequência, estamos indicando ao navegador que dê preferência para a primeira. Caso ela não seja encontrada, tente a próxima. E essa estratégia se seguirá até a última, que geralmente é a família genérica `serif`, `sans-serif` ou `monospaced`.

Vamos fazer alguns exemplos aplicando famílias bem simples às nossas fontes. Vá até o seu exercício atual e aplique algumas declarações de font-family aos seletores de cada componente formatável do seu documento HTML.

```
<style>
  body {
    font-family: Arial, Helvetica, sans-serif;
    color: black;
  }
  h1 {
    font-family: 'Franklin Gothic Medium', 'Arial Narrow', Arial, sans-serif;
    color: #rgb(24, 97, 126);
  }
  h2 {
    font-family: 'Times New Roman', Times, serif;
    color: #rgb(33, 136, 161);
  }
  p {
    font-family: 'Courier New', Courier, monospace;
  }
</style>
```



**SEQUÊNCIAS SEGURAS:** Existem as chamadas sequências seguras para especificações de famílias de fontes. Para ver quais são elas, abra o Google e faça uma rápida busca por CSS Web Safe Font Combinations.

No código acima, seus títulos principais `<h1>` usarão preferencialmente a fonte **Franklin Gothic Medium**, uma fonte sem serifa e que tem seu espaço horizontal bem limitado. Porém, essa fonte geralmente não existe em smartphones, que possuem a fonte **Arial Narrow** que é bem parecida mas é menos densa. Caso nenhuma delas seja encontrada no aparelho do visitante, o navegador vai selecionar a fonte **Arial** normal. Em último caso, se tudo der errado, o sistema selecionará uma fonte genérica sem serifa.

## Vamos falar de tamanhos

Além da família, podemos configurar tamanhos e estilos extras de qualquer componente textual do nosso documento HTML5.

Para especificar tamanho de fontes, existem várias medidas como **cm** (centímetros), **in** (polegadas), **pt** (pontos), **pc** (paicas), **px** (pixels), etc. Para tamanhos de fonte a serem exibidos na tela, o W3C recomenda o uso do **px** ou do **em**.



**EU GOSTO DE USAR PT, MAS:** A medida **pt** é aquela usada em editores de texto como o **Microsoft Word**. A recomendação oficial é de usar **pt** apenas para referenciar conteúdos que serão impressos.

A medida **em** é uma das que gera mais dúvida nos alunos. Ela é uma medida referencial em relação ao tamanho original da fonte. O tamanho padrão de uma fonte é geralmente **16px**, isso equivale a **1em**. A partir daí, podemos configurar o tamanho de um título, por exemplo, como sendo 2 vezes maior que a fonte padrão usando o valor **2em** para a propriedade.

```
h1 {  
    font-family: 'Franklin Gothic Medium', 'Arial Narrow', Arial, sans-serif;  
    font-size: 2em;  
}  
  
h2 {  
    font-family: 'Times New Roman', Times, serif;  
    font-size: 1.5em;  
}
```

No exemplo acima, todo título `<h2>` do nosso documento será 1.5x o tamanho padrão da fonte de referência.



**MAIS INFORMAÇÃO:** Para saber mais sobre as medidas suportadas pelas CSS, acesse o site oficial da W3C em:

[https://www.w3.org/Style/Examples/007/units.pt\\_BR.html](https://www.w3.org/Style/Examples/007/units.pt_BR.html)

## Outros estilos

Existem outras formatações muito usadas em CSS, que são as propriedades `font-style` para aplicar o itálico e `font-weight` para aplicar o negrito, sem contudo existir o fator semântico discutido no **capítulo 08**.



O padrão para essas duas propriedades é o valor normal, mas podemos aplicar o valor itálico ao font-style usando italic (mais compatível) ou oblique (menos compatível). Já o negrito, pode ser aplicado por nomes como lighter, bold e bolder ou pelo peso numérico, como indicado na imagem.

## Me dá uma mãozinha ?

As formatações de fontes são tão importantes e tão usadas em CSS, que existem "atalhos" para usá-las. São as chamadas *shorthands*.

Existe uma shorthand para fontes que é a propriedade font. No lugar de fazer várias configurações em múltiplas linhas, podemos simplificar tudo de maneira muito simples.

Por exemplo, no lugar de configurar o estilo dos parágrafos do nosso site desse jeito:

```
p {  
    font-family: Arial, Helvetica, sans-serif;  
    font-size: 1em;  
    font-style: italic;  
    font-weight: bold;  
}
```

Podemos usar a shorthand font que vai simplificar tudo:

```
p {  
    font: italic bold 1em Arial, Helvetica, sans-serif;  
}
```

A ordem dos atributos de uma *shorthand* em CSS é importante. No caso da propriedade font, devemos informar, na ordem:

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

## Alinhamentos

Existem quatro tipos de alinhamento de textos:

text-align: left;

  Lorem Ipsum is simply dummy text of the printing and typesetting industry. Lorem Ipsum has been the industry's standard dummy text ever since the 1500s, when an unknown printer took a galley of type and scrambled it to make a type specimen book.

text-align: right;

  Lorem Ipsum is simply dummy text of the printing and typesetting industry. Lorem Ipsum has been the industry's standard dummy text ever since the 1500s, when an unknown printer took a galley of type and scrambled it to make a type specimen book.

```
text-align: center;  
  
Lorem Ipsum is simply dummy text of  
the printing and typesetting industry.  
Lorem Ipsum has been the industry's  
standard dummy text ever since the  
1500s, when an unknown printer took a  
galley of type and scrambled it to make  
a type specimen book.
```

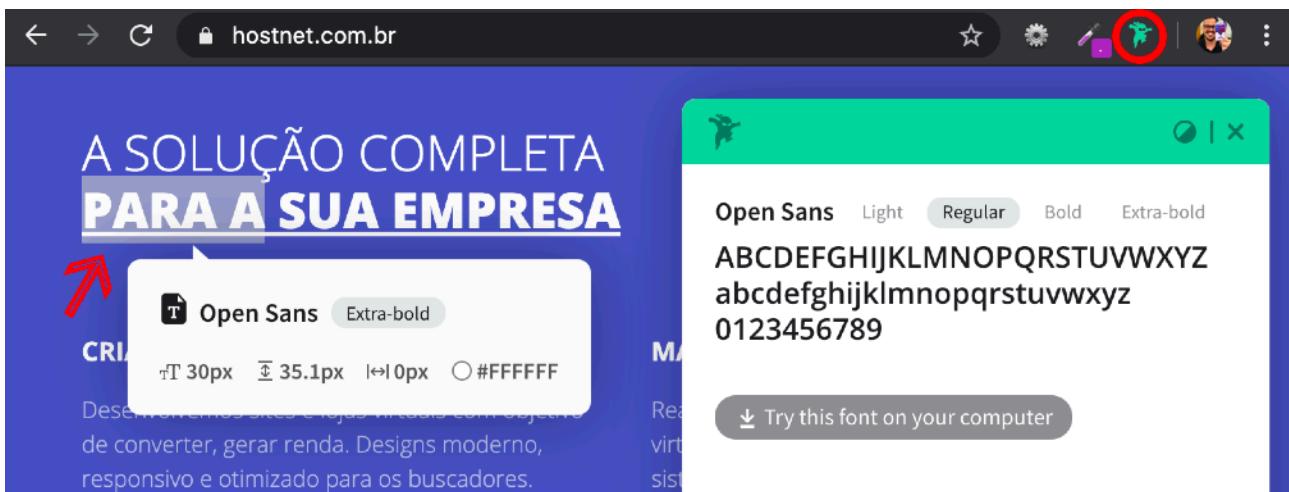
```
text-align: justify;  
  
Lorem Ipsum is simply dummy text of  
the printing and typesetting industry.  
Lorem Ipsum has been the industry's  
standard dummy text ever since the  
1500s, when an unknown printer took a  
galley of type and scrambled it to make  
a type specimen book.
```

## Como descobrir uma fonte que está sendo usada em outro site?

No capítulo anterior, te ensinei a usar a extensão **Colorzilla** para pegar uma cor que estava sendo usada em outro site. Agora vou te ensinar a usar a extensão **Fonts Ninja** do Google Chrome para capturar a fonte usada em componentes de texto.

Acesse novamente o site do **Chrome Web Store** e procure pela extensão **Fonts Ninja** (ensinei como fazer isso no capítulo anterior). Uma vez instalada e ativa, a extensão ficará ao lado da barra de endereços, assim como o Colorzilla.

Abra um site qualquer, selecione o trecho de texto que quer identificar (recomendo selecionar poucas palavras) e clique sobre o botão do **Fonts Ninja** (veja na imagem a seguir).



Além de mostrar qual foi a família tipográfica utilizada no texto selecionado, a extensão vai te indicar o tamanho e peso da fonte, o espaçamento vertical e horizontal e a cor aplicada a ele. Para isso, basta mover o mouse sobre o texto e um balão aparecerá com todas essas informações.

## Como usar fontes do Google Fonts

Além das famílias tipográficas e fontes padronizadas disponíveis para os navegadores, podemos usar fontes externas em nosso projeto sem a necessidade de baixar e instalar nenhuma fonte no computador do visitante.

Para isso, usaremos um serviço gratuito chamado **Google Fonts**, disponível em <https://fonts.google.com>. Ao acessar o site, algumas áreas são muito úteis:

The screenshot shows the Google Fonts homepage. At the top, there's a search bar (1) with a magnifying glass icon, a 'Custom' dropdown (2), a 'Curso em Vídeo' example text area (2), a '40px' size selector (3), and a 'Categories' dropdown (4). Below these are buttons for 'Language', 'Font properties', and 'Show only variable'. A 'View' dropdown shows 'Grid' selected. The main area displays 989 families of fonts. Two examples are shown: 'Roboto' (12 styles) and 'Jost' (Variable). Both examples show the text 'Curso em Vídeo' (5) in different font styles.

- 1 - Se você já sabe o nome de uma fonte, basta digitar nessa área.
- 2 - Na segunda área marcada, você pode escrever um texto de exemplo e vê-lo aplicado em várias fontes.
- 3 - É o tamanho da fonte que será apresentado na tela
- 4 - São as categorias das fontes que serão exibidas, suporta as opções Serif, Sans Serif, Display, Handwriting e Monospace. Você pode escolher mais de uma categoria.
- 5 - Uma lista com as fontes que satisfazem as configurações feitas e um exemplo do texto personalizado aplicado.

Uma vez escolhida a fonte, clique sobre o nome dela (como na área 5, acima) e uma outra tela será exibida, como a seguir. Clique sobre o botão **+ Select this style** e em seguida pressione o ícone superior, conforme marcado na próxima imagem.

This screenshot shows the 'Monoton' font details page. The font name 'Monoton' is displayed prominently, along with the designer's name, Vernon Adams. Below this, a section titled 'Styles' lists the available styles, including 'Regular 400'. A preview of the text 'CURSO EM VÍDEO' is shown in the 'Regular 400' style. To the right of the preview is a red arrow pointing right, followed by a blue button labeled '+ Select this style'. The top right corner of the page has a circular icon with a grid and plus sign inside.

Ao clicar no ícone superior direito, uma aba lateral chamada **Selected family** aparecerá.

Review

Embed

# Monoton

Designed by Vernon Adams

## Styles

Regular 400

**CURSO EM VÍDEO**

## About Monoton

Monoton is a contemporary take on metalpress fonts like the ones used in 1931 by Rudolf Koch. Monoton is a pure display web font with 30 points. Monoton has been designed to be used freely.

To embed a font, copy the code into the <head> of your html

<link> **@import**

```
<style>
@import url('https://fonts.googleapis.com/css2?family=Monoton&
display=swap');
</style>
```

**CSS rules to specify families**

```
font-family: 'Monoton', cursive;
```

[API docs](#)

Em primeiro lugar, clique em **Embed** e em seguida em **@import** para ter acesso aos códigos que serão colocados no seu arquivo CSS. O código de cima será colocado na primeira linha das suas declarações de estilo. Já o segundo código, especificado em **CSS rules** será colocado na propriedade font-family na declaração de todo seletor onde vamos querer aplicar a fonte.

## Hora de exercitar

Chegou a hora de acessar o endereço do nosso repositório público em <https://gustavoguanabara.github.io/html-css/exercicios/> e executar o **exercício 017** no seu computador.

Agora tente atingir esse mesmo resultado em casa, sem copiar o código que eu criei. Nesse momento, a prática é algo que você precisa. Se por acaso ficar difícil, pode acessar o repositório público de HTML e CSS e dar uma olhada nos comandos, mas **EVITE COPIAR**.



## Eu já falei sobre isso no YouTube?

Eu sei que às vezes as pessoas gostam mais de assistir vídeos do que ler livros, e é por isso que eu lanço há anos materiais no canal Curso em Vídeo no YouTube. O link que vou compartilhar contigo tem o conteúdo explicado como você leu aqui, só que de forma mais ilustrada. Reserve um tempo dos seus estudos para assistir esse vídeo todo.



Curso em Vídeo: [https://www.youtube.com/playlist?list=PLHz\\_AreHm4dlAnJ\\_jJtV29RFxnPHDuk9o](https://www.youtube.com/playlist?list=PLHz_AreHm4dlAnJ_jJtV29RFxnPHDuk9o)

# DevWeb

## Capítulo 15

# Seletores Personalizados

Agora que nós já passamos pelos elementos mais importantes do design, que são as cores, as imagens e as fontes, vamos falar um pouco sobre formatações especiais que vão permitir organizar esses conteúdos na tela, indicando seus tamanhos, espaçamentos, sombras e tudo mais. Eu te convido a aprender maneiras que vão facilitar no desenvolvimento web de sites eficientes e bonitos.



Você tem todo o direito de usar esse material para seu próprio aprendizado. Professores também podem ter acesso

a todo o conteúdo e usá-los com seus alunos. Porém todos os que usarem esse material - seja para qual for a finalidade - deverão manter a referência ao material original, criado pelo **Prof.**

**Gustavo Guanabara** e disponível no endereço do seu repositório público <https://github.com/gustavoguanabara/>. Este material não poderá ser utilizado em nenhuma hipótese para ser replicada - integral ou parcialmente - por autores/editoras para criar livros ou apostilas, com finalidade de obter ganho financeiro com ele.



# Personalizando Seletores

Para começar a dar mais poder às CSS, criando estilos personalizados, precisamos aprender a utilizar os seletores de id (#) e de class (.) de maneira eficiente. Ao criar nosso conteúdo em HTML, podemos **identificar** um determinado elemento único com um id, ou **agrupar** elementos múltiplos que tenham características semelhantes com um class. Vamos olhar um exemplo simples de documento HTML, com propriedades identificadoras:

```
22 <body>
23   <h1 id="titulo-principal">Aprenda Desenvolvimento Web</h1>
24   <h1>Aprenda HTML</h1>
25   <h2 class="topico">Semântica Web</h2>
26   <p class="texto">Lorem ipsum dolor sit, amet consectetur adipisicing elit.
    Optio, quibusdam? Tempore sequi commodi natus, vitae enim sed molestias. Rem,
    voluptas illum. Ut, debitis error quas explicabo corporis officia fugit.
    Doloribus.</p>
27   <h1>Aprenda CSS</h1>
28   <h2 class="topico">Estilos Web</h2>
29   <p>Lorem ipsum dolor sit, amet consectetur adipisicing elit. Dolores, repellat
    dicta. In distinctio quasi nesciunt blanditiis. Quas expedita et architecto
    alias laudantium, facere neque quo facilis! Ut est preferendis sed!</p>
30 </body>
```



Olhando atentamente para a **linha 23**, vimos que o `<h1>` principal se diferencia dos demais que estão nas **linhas 24 e 27**, pois ele possui um id.

Agora foque sua atenção nas **linhas 25 e 28** e perceba que os dois títulos `<h2>` possuem o mesmo class.

Um id vai **identificar** um elemento único dentro da página atual. Essa identificação vai nos permitir criar um estilo especial para um elemento isolado. Já um class vai identificar uma **classe** à qual um ou mais elementos pertençam, compartilhando características em comum a todos os que façam parte desse grupo.



**SE LIGA NA REGRA:** Em um mesmo documento HTML, só podemos usar um id para um único elemento, o que significa que não podemos ter dois elementos com um mesmo id dentro de uma mesma página. Porém, podemos atribuir um mesmo class para vários elementos que possuam essa mesma característica.

Em seguida, baseado no código HTML visto anteriormente, vamos criar um estilo local criando uma área `<style>` dentro da seção `<head>` do documento atual.

```

7   <style>
8     body {
9       font: normal 1em Arial, Helvetica, sans-serif;
10    }
11
12    h1 {
13      font-size: 2em;
14      color: darkgreen;
15    }
16
17    h2 {
18      color: green;
19    }
20  </style>

```

O que fizemos foi criar configurações simples de estilo baseadas nos elementos body, h1 e h2 do nosso HTML. Até o momento, não fizemos nenhuma configuração personalizada para nenhum seletor. O resultado é bem previsível:

## Aprenda Desenvolvimento Web

### Aprenda HTML

#### Semântica Web

Lorem ipsum dolor sit, amet consectetur adipisicing elit. Optio, quibusdam? Tempore sequi commodi natus, vitae enim sed molestias. Rem, voluptas illum. Ut, debitis error quas explicabo corporis officia fugit. Doloribus.

### Aprenda CSS

#### Estilos Web

Lorem ipsum dolor sit, amet consectetur adipisicing elit. Dolores, repellat dicta. In distinctio quasi nesciunt blanditiis. Quas expedita et architecto alias laudantium, facere neque quo facilis! Ut est preferendis sed!

No código ao lado, criamos seletores com simbologias novas. Isso porque não são seletores para elementos HTML, e sim seletores para id e class. Na **linha 21**, criamos um seletor personalizado para o identificador `titulo-principal`, que está atribuído ao h1. Você deve ter percebido que colocamos um símbolo `#` antes do id.

Já as **linhas 27 e 31** são seletores que receberão declarações para as classes `topico` e `texto`. Todos os seletores personalizados para uma class são identificados por um ponto que vem antes do nome da classe.

Perceba no resultado ao lado que os dois primeiros títulos possuem apresentações idênticas, mesmo que o primeiro deles possua um id.

Isso aconteceu porque não fizemos nenhuma configuração específica para ele. Não adianta colocar apenas id ou class em um elemento e não personalizar o resultado visual usando seletores personalizados.

Agora vamos adicionar algumas declarações e seletores personalizados, ainda dentro da área `<style>` que criamos:

```

21  #titulo-principal {
22    background-color: darkgreen;
23    color: white;
24    text-align: center;
25  }
26
27  .topico {
28    text-align: right;
29  }
30
31  .texto {
32    text-align: justify;
33  }

```



**MAIS UMA REGRA:** Todo id em HTML é identificado nas CSS por um símbolo #. Toda class em HTML é identificada nas CSS por um ponto.

Ao adicionar os seletores personalizados ao nosso estilo local, o resultado fica bem diferente:

ANTES...	...DEPOIS
<p><b>Aprenda Desenvolvimento Web</b></p> <p><b>Aprenda HTML</b></p> <p><b>Semântica Web</b></p> <p>  Lorem ipsum dolor sit, amet consectetur adipisicing elit. Optio, quibusdam? Tempore sequi commodi natus, vitae enim sed molestias. Rem, voluptas illum. Ut, debitis error quas explicabo corporis officia fugit. Doloribus.</p> <p><b>Aprenda CSS</b></p> <p><b>Estilos Web</b></p> <p>  Lorem ipsum dolor sit, amet consectetur adipisicing elit. Dolores, repellat dicta. In distinctio quasi nesciunt blanditiis. Quas expedita et architecto alias laudantium, facere neque quo facilis! Ut est preferendis sed!</p>	<p><b>Aprenda Desenvolvimento Web</b></p> <p><b>Aprenda HTML</b></p> <p><b>Semântica Web</b></p> <p>  Lorem ipsum dolor sit, amet consectetur adipisicing elit. Optio, quibusdam? Tempore sequi commodi natus, vitae enim sed molestias. Rem, voluptas illum. Ut, debitis error quas explicabo corporis officia fugit. Doloribus.</p> <p><b>Aprenda CSS</b></p> <p><b>Estilos Web</b></p> <p>  Lorem ipsum dolor sit, amet consectetur adipisicing elit. Dolores, repellat dicta. In distinctio quasi nesciunt blanditiis. Quas expedita et architecto alias laudantium, facere neque quo facilis! Ut est preferendis sed!</p>

Analise os dois resultados acima e perceba as diferenças. Note que apenas o título h1 identificado como título-principal ganhou uma formatação diferente e os demais títulos de mesmo nível se mantiveram como antes. Inclusive, compare o resultado diferente obtido entre o primeiro e o segundo parágrafos. Apenas o primeiro ficou justificado. Isso foi por conta da aplicação de uma class apenas ao primeiro (volte lá no código HTML inicial e confira que o segundo parágrafo não possui class).

## Propriedades herdadas

No **Capítulo 12**, discutimos três técnicas usadas para aplicar estilos aos nossos documentos HTML: **inline**, **local** e **externa**. Durante os nossos vídeos de acompanhamento do material em PDF, demonstramos que as três técnicas podem ser usadas em conjunto, e os estilos de cada serão combinados nessa ordem:

- 1º lugar: CSS externo
- 2º lugar: CSS interno/local
- 3º lugar: CSS inline



Sendo assim, caso haja duplicidade nas configurações, o que fica valendo no final é a propriedade especificada nas configurações inline. Cada camada de aplicação de estilos vai adicionando propriedades novas e sobrescrevendo as configurações do nível anterior.

Quando nós aplicamos configurações personalizadas de `id` ou de `class`, também vai existir uma combinação de configurações, como você já deve ter percebido ao analisar o exemplo que apresentamos nas páginas anteriores.

Voltando ao exemplo dado anteriormente (que vou replicar aqui para facilitar o entendimento), logo no início do corpo, tínhamos dois elementos `h1`:

```
<body>
  <h1 id="titulo-principal">Aprenda Desenvolvimento Web</h1>
  <h1>Aprenda HTML</h1>
```

Logo em seguida, na área de estilo, criamos uma configuração especial para os elementos `h1`:

```
h1 {
  font-size: 2em;
  color: darkgreen;
}
```

Essas duas configurações de tamanho e cor da fonte serão aplicadas aos dois títulos, já que ambos são do tipo `h1`. Por fim, adicionamos uma configuração específica para o elemento que tem um `id` personalizado:

```
#titulo-principal {
  background-color: darkgreen;
  color: white;
  text-align: center;
}
```

Dessa maneira, o primeiro `h1` do código HTML (aquele que tem um `id`) ganhará mais três configurações - cor de fundo, cor da fonte e alinhamento do texto - graças a esse seletor extra. Note que na primeira configuração CSS dos elementos `h1`, dissemos que a cor era `darkgreen`.

Porém, quando aplicamos a segunda configuração de `color` no seletor `#titulo-principal`, ela vai sobrepor o valor da propriedade para `white`. As outras duas propriedades (`background-color` e `text-align`) serão adicionadas às configurações desse primeiro título. O segundo título não sofrerá alterações, já que apenas o primeiro possui o `id` referido no seletor.



## Pseudo-classes e pseudo-elementos

Uma pseudo-classe CSS é uma palavra-chave adicionada às declarações de um seletor após um sinal de dois pontos e especificam um estado especial de um elemento. Existem várias pseudo-classes para estilos, podemos citar `:hover`, `:visited`, `:active`, `:checked`, `:empty` e `:focus`.

Já um pseudo-elemento CSS é uma palavra-chave adicionada às declarações de um seletor após dois sinais de dois pontos e permitem que você formate um pedaço específico do elemento referenciado. Os principais pseudo-elementos usados nas CSS são ::before, ::after, ::first-letter, ::first-line.

Vamos começar criando um exemplo bem simples e bastante usado para exemplificar o uso de pseudo-classes e pseudo-elementos: a personalização de links.

```
43  <body>
44      <h1>Personalizando um link</h1>
45      <p>
46          Para aprender HTML5 e CSS3, acesse o
47          <a href="https://gustavoguanabara.github.io">
48              GitHub do Guanabara
49          </a>
50      </p>
51      <p>
52          Se quiser conhecer mais sobre os padrões, visite o site do
53          <a href="https://www.w3c.br">Escritório do W3C no Brasil</a>
54      </p>
55      <p>Para acompanhar os cursos, acesse o
56          <a href="https://www.youtube.com/cursoemvideo/" class="especial">
57              canal do CursoemVideo
58          </a>
59      </p>
60  </body>
```

Começando pelo corpo do documento, contendo só o HTML, vamos criar um texto com um título e três parágrafos, com três links. Note que o último link recebeu a atribuição de uma classe específica. O resultado visual está sendo apresentado a seguir:

# Personalizando um link

Para aprender HTML5 e CSS3, acesse o [GitHub do Guanabara](https://gustavoguanabara.github.io)

Se quiser conhecer mais sobre os padrões, visite o site do [Escritório do W3C no Brasil](https://www.w3c.br)

Para acompanhar os cursos, acesse o [canal do CursoemVideo](https://www.youtube.com/cursoemvideo/)

Na imagem acima, o primeiro link apareceu com a cor violeta pois já tínhamos visitado o perfil do GitHub anteriormente. Por padrão, os navegadores mostram links inéditos em azul e os visitados em violeta. Todos os links também aparecem sublinhados por padrão.

Vamos alterar essa apresentação padrão com nossas configurações de estilo. Crie a área `<style>` dentro de `<head>` e coloque os seguintes seletores.

```
8   body {
9     font: normal 1em Arial, Helvetica, sans-serif;
10    }
11
12   a {
13     font-weight: bold;
14     text-decoration: none;
15     color: red;
16   }
17
18   a:visited {
19     color: darkred;
20   }
21
22   a:hover {
23     text-decoration: underline;
24 }
```

Nas declarações acima, criamos três configurações para os links: a primeira (**linha 12**) para links inéditos (não visitados), colocando o texto em negrito, removendo o sublinhado e colocando-os em cor vermelha.

No segundo seletor para links (**linha 18**), configuramos as âncoras já visitadas (com a pseudo-classe :visited). Nele, apenas mudamos a cor para vermelho escuro.

Já na terceira declaração (**linha 22**), fizemos configurações para todos os links, quando passarmos o mouse por cima (pseudo-classe :hover) e o sublinhado volta a aparecer.

O resultado visual está apresentado abaixo, compare com o resultado anterior e veja as diferenças:

## Personalizando um link

Para aprender HTML5 e CSS3, acesse o [GitHub do Guanabara](#) 

Se quiser conhecer mais sobre os padrões, visite o site do [Escritório do W3C no Brasil](#)

Para acompanhar os cursos, acesse o [canal do CursoemVideo](#)

Por fim, vamos adicionar algumas configurações relacionadas à classe especial, criada no documento HTML, para o terceiro link.

```

26     .especial:hover {
27         color: white;
28         background-color: black;
29         text-decoration: none;
30     }
31
32     .especial::before {
33         content: '»';
34         font-weight: lighter;
35     }
36
37     .especial::after {
38         content: '«';
39         font-weight: lighter;
40     }

```

Na primeira declaração do código acima (**linha 26**), dizemos que o link da classe especial vai ter letra branca, fundo preto e perderá o sublinhado apenas quando movermos o mouse sobre ele.

Nas próximas declarações (**linhas 32 e 37**), vamos adicionar um símbolo » antes e outro símbolo « depois usando os pseudo-elementos ::before e ::after, respectivamente. O resultado dessas declarações está apresentado a seguir:

## Personalizando um link

Para aprender HTML5 e CSS3, acesse o [GitHub do Guanabara](#)

Se quiser conhecer mais sobre os padrões, visite o site do [Escritório do W3C no Brasil](#)

Para acompanhar os cursos, acesse o » [canal do CursoemVideo](#) «

## Mais um exemplo

Vamos criar mais um exemplo para o uso de pseudo-classes. Nosso HTML vai ter o seguinte corpo:

```

18 <body>
19     <h1>Conteúdo especial</h1>
20     <div>
21         Passe o mouse aqui
22         <p>SURPRESAAAA!</p>
23     </div>
24     <p>Veja o conteúdo oculto aparecendo!</p>
25 </body>

```

Na **linha 20**, criamos um bloco especial com a tag <div>. Uma das grandes vantagens em usar divs é que elas podem ter outras tags dentro dela, assim como o parágrafo interno que criamos na **linha 22**.

Vamos criar nosso estilo agora, dentro da área <head>:

```
7   <style>
8     div > p {
9       display: none;
10    }
11
12    div:hover > p {
13      display: block;
14      background-color: yellow;
15    }
16  </style>
```

O primeiro seletor, na **linha 8**, vai esconder o parágrafo que está dentro da div (representado em CSS como div > p) através da propriedade display com o valor none.



**MAIS UM SÍMBOLO:** Toda vez que usamos o símbolo > em um seletor, indicamos os filhos (*children*) imediatos de um elemento. No exemplo HTML que criamos, dizemos que o primeiro parágrafo é *direct children* da div.

O segundo seletor, na **linha 12**, vai fazer o parágrafo escondido reaparecer, com o fundo pintado de amarelo apenas quando passarmos o mouse sobre ele.

SEM MOUSE	COM MOUSE
<h2>Conteúdo especial</h2> <p>Passe o mouse aqui</p> <p>Veja o conteúdo oculto aparecendo!</p>	<h2>Conteúdo especial</h2> <p>Passe o mouse aqui </p> <p><b>SURPRESAAAA!</b></p> <p>Veja o conteúdo oculto aparecendo!</p>

# DevWeb

## Capítulo 16

### O Modelo de Caixas

Entender o modelo das caixas é um dos primeiros passos para construir interfaces web e começar a dar forma aos seus sites. Nos últimos quatro capítulos, estudamos a essência das folhas de estilo e já sabemos criar seletores, identificá-los e personalizá-los. Agora chegou a hora de colocarmos tudo isso em caixas configuráveis e vamos começar a desenhar nossas primeiras páginas.



Você tem todo o direito de usar esse material para seu próprio aprendizado. Professores também podem ter acesso

a todo o conteúdo e usá-los com seus alunos. Porém todos o que usarem esse material - seja para qual for a finalidade - deverão manter a referência ao material original, criado pelo **Prof.**

**Gustavo Guanabara** e disponível no endereço do seu repositório público <https://github.com/gustavoguanabara/>. Este material não poderá ser utilizado em nenhuma hipótese para ser replicada - integral ou parcialmente - por autores/editoras para criar livros ou apostilas, com finalidade de obter ganho financeiro com ele.



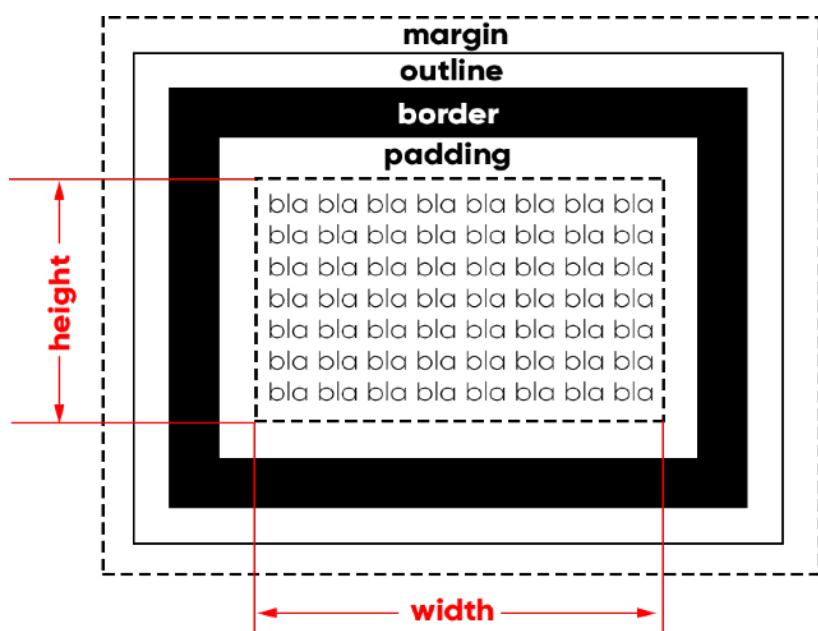
# O que é uma caixa?

De forma simples e objetiva, baseado em um conceito chamado “*box model*”, a grande maioria dos elementos HTML que temos no nosso site são como caixas. Elas são *containers* que armazenam conteúdos ou até mesmo outras caixas.



## Anatomia de uma caixa

Vamos analisar como uma caixa vai ser apresentada por todos os navegadores. Olhe atentamente o diagrama a seguir, que é exatamente o já citado **modelo de caixa**:



Tudo começa a partir do **conteúdo** (content), que representamos acima com o bla bla bla... Por padrão, toda caixa é composta apenas pelo conteúdo e não possui padding, nem border, nem outline e nem margin. Uma exceção curiosa é o elemento <body> que já vem com uma margin de 8px.

Todo conteúdo possui uma **largura** (width) e uma **altura** (height) e a esse conjunto de propriedades, damos o nome de box-size (tamanho da caixa). O tamanho da caixa não inclui as medidas de padding, border, outline e margin.

Depois do conteúdo e de seu tamanho, vamos nos focar na **borda** que fica em volta dele. Ela pode ter uma espessura, uma cor e um formato.

Entre a borda e o conteúdo - da borda para dentro - temos o **preenchimento** (padding) e da borda para fora, temos a **margem** (margin).

Entre a margem e a borda, podemos determinar o **contorno** (outline) que é muito pouco utilizado, mas existe. Ele é um traçado visual que podemos criar fora da borda e o cálculo da sua espessura faz parte da margem estabelecida.

Vamos criar um exemplo simples para exemplificar todos esses componentes, configurando as propriedades do modelo de caixa de um título `<h1>`. Acompanhe o trecho de código a partir das definições de estilo.

```
7 <style>
8   h1 {
9     width: 300px;
10    height: 50px;
11    background-color: lightgray;
12    border-width: 10px;
13    border-style: solid;
14    border-color: red;
15    padding: 20px;
16    outline-width: 30px;
17    outline-style: solid;
18    outline-color: blue;
19    margin: 50px;
20  }
21 </style>
22 </head>
23 <body>
24   <h1>Exemplo de Caixa</h1>
25 </body>
26 </html>
```

Todas as configurações serão aplicadas ao elemento `<h1>`, que é uma caixa e foi criado na **linha 24** do código acima. As **linhas 9 e 10** configuram o size da caixa (largura e altura, respectivamente) e fará com que ela tenha 300x50 pixels.

As **linhas de 12 a 14**, configuram uma borda sólida, vermelha e com 10 pixels de espessura.

A **linha 15** vai criar um espaço interno de preenchimento (da borda para dentro) de 20 pixels no elemento e a **linha 19** vai criar um espaço externo (da borda para fora) de 50 pixels.

As **linhas de 16 a 18** vão usar parte da margem para criar um contorno azul, sólido e com 30 pixels de espessura.



**TAMANHO TOTAL:** Para calcular a largura e altura total de um elemento na tela, some os tamanhos do **conteúdo + preenchimento + borda + margem**. O contorno não vai entrar nessa conta, pois utiliza parte da medida da margem.

O resultado visual do código anterior será:



Olhando de perto, podemos analisar as medidas configuradas no código apresentado. As medidas de height e width (300x50) são medidas apenas pela parte pontilhada do conteúdo.

A border de 10px ficou em vermelho e o outline de 30px ficou em azul. O padding de 20px fica da borda para dentro e a margin de 50px fica da borda para fora.

Sendo assim, a medida total que essa caixa vai ocupar é de  $50 + 10 + 20 + 300 + 20 + 10 + 50 = \textbf{460px de largura}$  e  $50 + 10 + 20 + 50 + 20 + 10 + 50 = \textbf{210px de altura}$ .



**NOVIDADE DAS CSS3:** Existe a nova propriedade box-sizing onde podemos definir que as dimensões height e width não são medidas apenas a partir do conteúdo (content-box) e sim pela borda (border-box).

## Dá pra simplificar?

As configurações de borda e contorno também possuem *shorthands* para simplificar o código anterior. A ordem para as duas configurações é sempre a mesma para as duas shorthands: largura (-width), estilo (-style) e cor (-color).

MODO COMPLETO	SHORTHAND
<code>border-width: 10px; border-style: solid; border-color: red; outline-width: 30px; outline-style: solid; outline-color: blue;</code>	<code>border: 10px solid red; outline: 30px solid blue;</code>

# Preenchimento e margem personalizados

Todo elemento de caixa possui quatro valores para padding e quatro para margin, sempre nessa mesma ordem: superior (-top), direita (-right), inferior (-bottom), esquerda (-left). Quando colocamos um único valor de dimensão para o preenchimento ou margem, esse mesmo valor é aplicado simetricamente a todas as direções, mas também podemos fazer códigos como:

MODO COMPLETO	SHORTHAND
<pre>padding-top: 10px; padding-right: 15px; padding-bottom: 20px; padding-left: 25px;  margin-top: 0px; margin-right: 10px; margin-bottom: 20px; margin-left: 30px;</pre>	<pre>padding: 10px 15px 20px 25px; margin: 0px 10px 20px 30px;</pre>

Também existe a opção de indicar cada *shorthand* das propriedades de preenchimento e borda usando apenas duas medidas:

MODO COMPLETO	SHORTHAND
<pre>padding-top: 10px; padding-right: 20px; padding-bottom: 10px; padding-left: 20px;  margin-top: 0px; margin-right: 15px; margin-bottom: 0px; margin-left: 15px;</pre>	<pre>padding: 10px 20px; margin: 0px 15px;</pre>

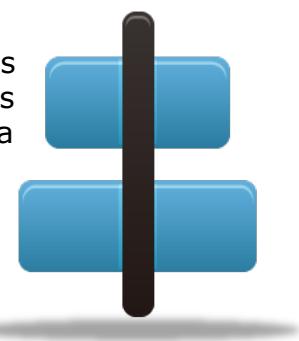
Essa simplificação só é possível quando as medidas -top e -bottom forem iguais entre si e o mesmo também ocorrer entre as medidas -right e -left.

# Margens no automático

Um recurso que também vai ser muito usado em nossos exercícios é a centralização de blocos. Para que isso seja feito, devemos pedir que o navegador calcule automaticamente as margens da esquerda e da direita para que o bloco seja colocado no meio do navegador, independente do tamanho da janela.

Para centralizar uma caixa, use a seguinte declaração no seu seletor:

```
margin: auto;
```



## Tipos de Caixa

Dependendo do comportamento da caixa, podemos classificar um elemento em uma de duas categorias:

### Caixa do tipo block-level

Um elemento dito *block-level* sempre vai se iniciar em uma nova linha e vai ocupar a largura total do elemento onde ele está contido. Se não estiver contido em nenhuma outra caixa, ele vai ocupar 100% da largura do <body>.

O elemento *block-level* mais conhecido é o <div> e suas variações semânticas modernas da HTML5, como <main>, <section>, <aside>, etc.

Na lista a seguir, coloquei alguns elementos HTML que são block-level:

<address>	<article>	<aside>	<blockquote>	<canvas>	<dd>
<div>	<dl>	<dt>	<fieldset>	<figcaption>	<figure>
<footer>	<form>	<h1> - <h6>	<header>	<hr>	<li>
<main>	<nav>	<noscript>	<ol>	<p>	<pre>
<section>	<table>	<tfoot>	<ul>		<video>

### Caixa do tipo inline-level

Um elemento do tipo *inline-level* não vai começar em uma nova linha, e sim no ponto exato onde foram definidos. E a largura dele vai ocupar apenas o tamanho relativo ao seu conteúdo.

Abaixo, listei alguns elementos *inline-level* usados pela HTML:

<a>	<abbr>	<acronym>	<b>	<bdo>	 
<button>	<cite>	<code>	<dfn>	<em>	<i>
<img>	<input>	<kbd>	<label>	<map>	<object>
<output>	<q>	<samp>	<script>	<select>	<small>
<span>	<strong>	<sub>	<textarea>	<tt>	<var>

## Grouping Tags e Semantic Tags

A linguagem HTML padrão tinha apenas duas tags de agrupamento genérico: a `<div>` e a `<span>`. A diferença básica entre elas é que a primeira é um elemento agrupador do tipo *block-level* e o segundo é *inline-level*. No mais, eles agem exatamente da mesma maneira, servindo para juntar vários outros elementos HTML.

Com o surgimento da HTML5, surgiram as tags semânticas de agrupamento. Isso não significa que as `<div>` e `<span>` (agora chamadas de não-semânticas) deixaram de existir ou ficaram obsoletas, mas seu uso agora faz menos sentido, pois temos tags para dividir as partes do nosso documento HTML.

Vamos compreender a partir de agora os principais agregadores semânticos da HTML5.

### Header

Cria áreas relativas a cabeçalhos. Pode ser o cabeçalho principal de um site ou até mesmo o cabeçalho de uma seção ou artigo. Normalmente inclui títulos `<h1>` - `<h6>` e subtítulos. Podem também conter menus de navegação.

### Nav

Define uma área que possui os links de navegação pela estrutura de páginas que vão compor o website. Um `<nav>` pode estar dentro de um `<header>`.



### Main

É um agrupador usado para delimitar o conteúdo principal do nosso site. Normalmente concentra as seções, artigos e conteúdos periféricos.

# Section

Cria seções para sua página. Ela pode conter o conteúdo diretamente no seu corpo ou dividir os conteúdos em artigos com conteúdos específicos. Segundo a documentação oficial da W3C, “uma seção é um agrupamento temático de conteúdos, tipicamente com um cabeçalho”.

# Article

Um artigo é um elemento que vai conter um conteúdo que pode ser lido de forma independente e dizem respeito a um mesmo assunto. Podemos usar um `<article>` para delimitar um post de blog ou fórum, uma notícia, etc.



**MÚLTIPLOS NÍVEIS:** A sua criatividade e planejamento vai definir a estrutura do seu site. Sendo assim, é possível ter um ou mais `<article>` dentro de uma `<section>` ou até mesmo criar `<section>` dentro de um `<article>`. Não existem limitações quanto a isso.

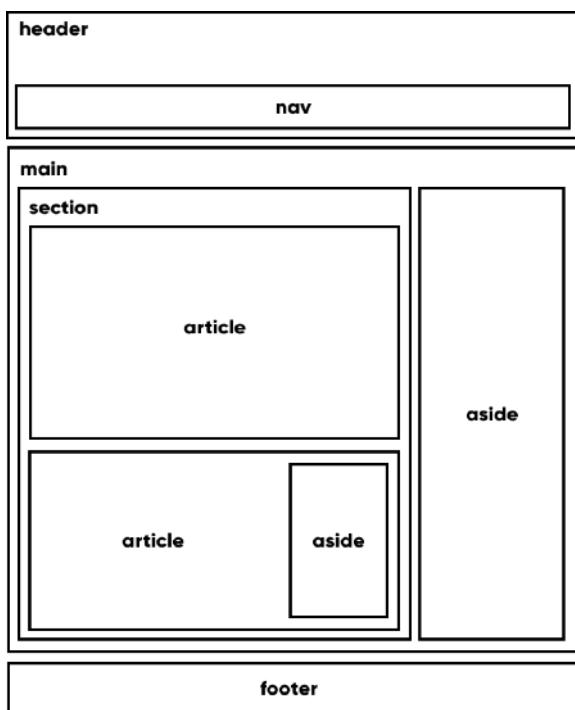
# Aside

Delimita um conteúdo periférico e complementar ao conteúdo principal de um artigo ou seção. Normalmente um conteúdo `<aside>` está posicionado ao lado de um determinado texto ou até mesmo no meio dele, exatamente como fizemos no bloco de texto apresentado anteriormente, falando sobre “MÚLTIPLOS NÍVEIS”.

# Footer

Cria um rodapé para o site inteiro, seção ou artigo. É um conteúdo que não faz parte diretamente do conteúdo nem é um conteúdo periférico (o que caracterizaria um `<aside>`), mas possui informações sobre autoria do conteúdo, links adicionais, mapa do site, documentos relacionados.

A seguir, vou criar uma proposta de estrutura para um projeto de site. Não tome ela como a única possibilidade de criar o posicionamento de elementos de agrupamento semântico.



```

9   <header>
10  <h1>Meu Site</h1>
11  <nav>link link link link...</nav>
12  </header>
13  <main>
14    <section>
15      <article>
16        <h2>Título</h2>
17        <p>Texto do artigo</p>
18      </article>
19      <article>
20        <h2>Título</h2>
21        <p>Texto do artigo</p>
22        <aside>
23          | conteúdo periférico do artigo
24        </aside>
25      </article>
26    </section>
27    <aside>
28      | conteúdo periférico do site
29    </aside>
30  </main>
31  <footer>
32    | conteúdo do rodapé
33  </footer>

```

Analise o diagrama do lado esquerdo e o código do lado direito da imagem acima. Veja a hierarquia entre os elementos e quais deles estão dentro um do outro.

## Sombras nas caixas

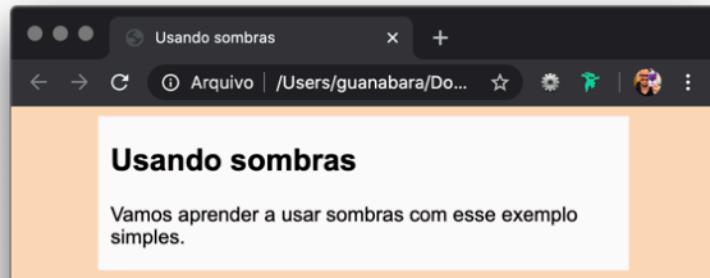
As sombras são muito úteis para dar volume, para dar a sensação de que as caixas estão ali realmente. Para exemplificar, vamos criar o seguinte código base:

```

7   <style>
8     body {
9       font: 1em Arial, Helvetica, sans-serif;
10    background-color: #peachpuff;
11  }
12  article#caixa {
13    background-color: #snow;
14    width: 400px;
15    margin: auto;
16    padding: 1px 10px;
17    /* sombra aqui */
18  }
19  </style>
20 </head>
21 <body>
22  <article id="caixa">
23    <h1>Usando sombras</h1>
24    <p>Vamos aprender a usar sombras com esse
25      exemplo simples. </p>
26  </article>
27 </body>
28 </html>

```

Olhando para o corpo da página, temos apenas um `<article>` (**linha 22**) com um breve conteúdo. A configuração de estilo (**linha 7** em diante) faz com que esse artigo seja configurado como uma pequena caixa centralizada. O resultado visual está apresentado a seguir:



Para criar uma sombra nessa caixa, vamos adicionar uma declaração especial na **linha 17**, substituindo o comentário que deixei lá.

```
box-shadow: 3px 5px 4px black;
```



Veja que uma sombra bem forte já pode ser percebida, assim que adicionamos a propriedade `box-shadow` e seus quatro valores. A ordem é sempre essa:

1. **Deslocamento horizontal** (*h-offset*): quanto a sombra vai andar para o lado direito (valores negativos causam deslocamento para a esquerda)
2. **Deslocamento vertical** (*v-offset*): quanto a sombra vai andar para baixo (valores negativos causam deslocamento para cima)
3. **Embaçamento** (*blur*): quanto a sombra vai se espalhar pelo fundo
4. **Cor** (*color*): cor da sombra. É possível usar transparência.



**MUITO CUIDADO!** Não exagere no uso de sombras, pois elas podem tornar o seu efeito visual muito pesado. Evite também usar sombras coloridas. Olhe ao seu redor e perceba que as sombras são sempre pretas. Use cores `rgba()` para obter uma transparência que cause efeitos mais suaves.

## Bordas decoradas

As bordas das caixas não precisam ser sempre retangulares e podem ter alguns detalhes especiais. Vamos usar o mesmo exercício que estamos criando desde o item anterior onde aprendemos a usar sombras.

# Vértices arredondados

Podemos arredondar os vértices usando uma declaração simples usando a propriedade border-radius. Adicione o seguinte comando ao seletor do artigo do exemplo que estamos criando:

**border-radius: 10px;**

## Usando sombras

Vamos aprender a usar sombras com esse exemplo simples.

Na declaração acima, todos os vértices foram levemente arredondados (*10px*) de forma simétrica. Se for necessário, podemos indicar quatro medidas diferentes, uma para cada vértice. Olhe atentamente para o resultado abaixo e perceba que cada ponta está diferente.

**border-radius: 10px 20px 30px 40px;**

## Usando sombras

Vamos aprender a usar sombras com esse exemplo simples.

Assim como fizemos com as margens, também é possível indicar apenas dois valores, o que vai agir em vértices intercalados, partindo do canto superior esquerdo.

**border-radius: 10px 30px;**

## Usando sombras

Vamos aprender a usar sombras com esse exemplo simples.

# Tenho desafios pra você!

Lá no repositório, além do material em PDF e dos códigos dos exercícios 100% disponíveis, também disponibilizamos alguns **desafios** que devem ser resolvidos. Esses desafios não incluem o código original e você deve tentar chegar à resposta sem copiar nenhum código.

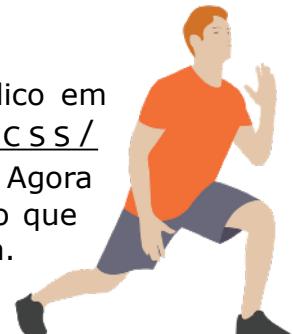
Com todo o conteúdo que vimos até essa aula, você já pode resolver o **desafio d010**. Acesse o repositório público, abra a área do curso de HTML+CSS e clique no link de acesso aos desafios. Manda ver! Só não fica pedindo a resposta! Você consegue resolver isso sozinho(a)!



Repositório em: <https://gustavoguanabara.github.io>

## Hora de exercitar

Chegou a hora de acessar o endereço do nosso repositório público em <https://gustavoguanabara.github.io/html-css/exercicios/> e executar o **exercício 017** no seu computador. Agora tente atingir esse mesmo resultado em casa, sem copiar o código que eu criei. Nesse momento, a prática é algo que você mais precisa. Se por acaso ficar difícil, pode acessar o repositório público de HTML e CSS e dar uma olhada nos comandos, mas **EVITE COPIAR**.



## Eu já falei sobre isso no YouTube?

Eu sei que às vezes as pessoas gostam mais de assistir vídeos do que ler livros, e é por isso que eu lanço há anos materiais no canal Curso em Vídeo no YouTube. O link que vou compartilhar contigo tem o conteúdo explicado como você leu aqui, só que de forma mais ilustrada. Reserve um tempo dos seus estudos para assistir esse vídeo todo.



Curso em Vídeo: [https://www.youtube.com/playlist?list=PLHz\\_AreHm4d1AnJ\\_jJtV29RFxnPHDuk9o](https://www.youtube.com/playlist?list=PLHz_AreHm4d1AnJ_jJtV29RFxnPHDuk9o)

# DevWeb

## Capítulo 17

# Primeiro Mini-Projeto

Chegou a hora de colocar em prática tudo aquilo que aprendemos até aqui e unificar tudo em um projeto simples. Mas não se engane, você ainda vai aprender coisas novas para poder criar um site básico com uma estrutura interessante. Vamos começar a analisar esse projeto e a aprender conceitos adicionais para torná-lo real.



Você tem todo o direito de usar esse material para seu próprio aprendizado. Professores também podem ter acesso

a todo o conteúdo e usá-los com seus alunos. Porém todos os que usarem esse material - seja para qual for a finalidade - deverão manter a referência ao material original, criado pelo **Prof.**

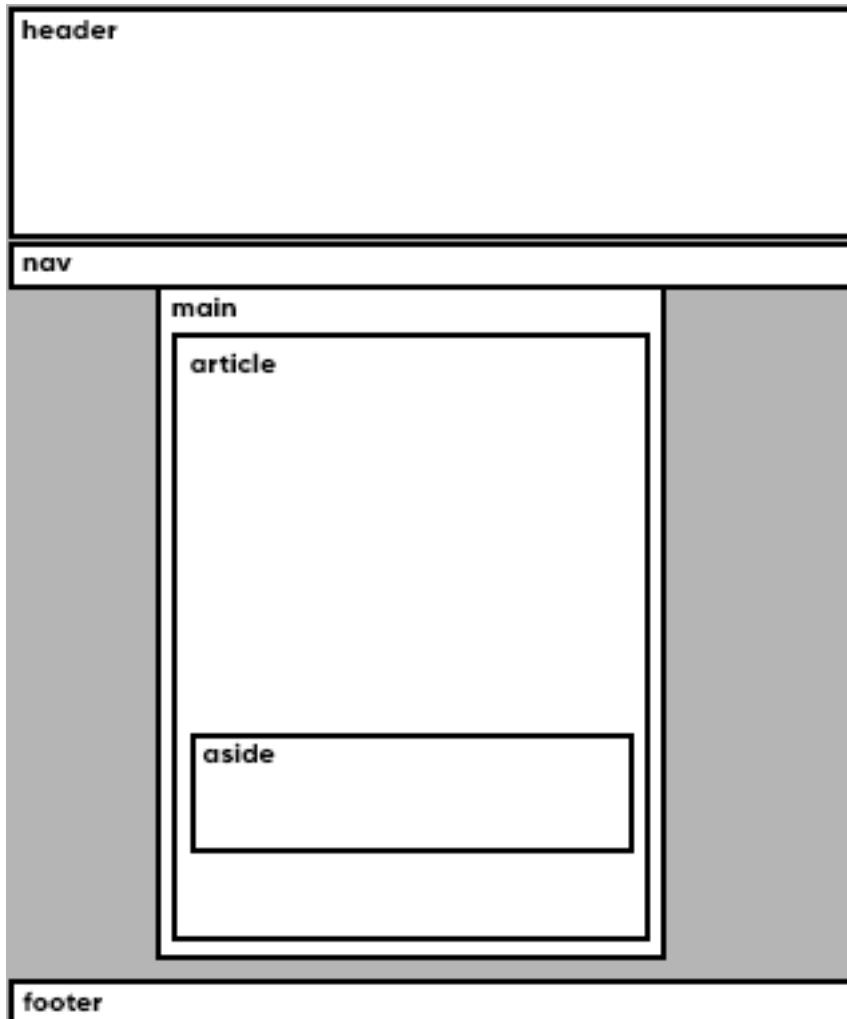
**Gustavo Guanabara** e disponível no endereço do seu repositório público <https://github.com/gustavoguanabara/>. Este material não poderá ser utilizado em nenhuma hipótese para ser replicada - integral ou parcialmente - por autores/editoras para criar livros ou apostilas, com finalidade de obter ganho financeiro com ele.



# A ideia do projeto

A ideia inicial é criar uma espécie de site de notícias, só que com uma notícia só 😅. O objetivo aqui é apenas ensinar como organizar o conteúdo e apresentá-lo em forma de página Web.

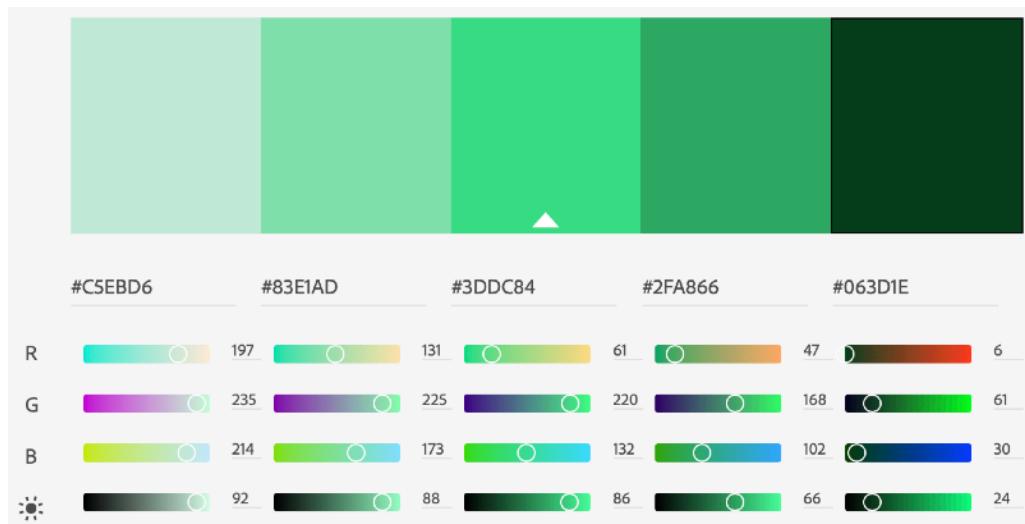
Tudo começa desenhando o que chamamos de *wireframe*, planejando qual vai ser a estrutura e comportamento do site. Ele vai servir de base na hora de planejar as caixas que farão parte da página. Para esse nosso primeiro mini-projeto, planejei o seguinte *layout*:



**CRIE SEUS WIREFRAMES:** Existe uma ferramenta muito legal para planejar seus *layouts*, tanto para sites quanto para aplicativos desktop e até apps de celular: o **MockFlow**. Como podemos imaginar, existem limitações na versão *free*, mas dá pra imaginar como o site vai ficar antes de começar a mexer no código.

Em seguida, vamos decidir a paleta de cores que será utilizada. Optei por uma paleta baseada em **monocromia** a partir de tons de verde (já que vamos falar sobre

Android), mas você pode usar a técnica que quiser para decidir suas cores básicas que serão usadas. Usei o **Adobe Color** (que aprendemos a usar no capítulo 13) e optei pelas seguintes cores:



Por fim, vamos falar das fontes escolhidas. A primeira, **Bebas Neue**, será a fonte para os destaques principais. Já que vamos fazer uma matéria sobre o mundo Android, optei por uma fonte chamada **iDroid**. Já para o texto padrão, escolhi a **Arial**.

LOREM IPSUM

Bebas Neue, cursive

LOREM IPSUM

iDroid, sans-serif

Lorem, ipsum dolor sit amet consectetur adipisicing elit. Eum, quo temporibus voluptates mollitia eius voluptas qui, officia blanditiis voluptatum ipsum exercitationem incident. Nemo aperiam laboriosam corporis magnam aliquid persiciatis quae?

Arial, Helvetica, sans-serif

## Já ouviu falar em responsividade?

Ainda que tenha optado por criar um projeto simples, não vamos abrir mão da versatilidade, pois queremos que nosso site possa ser visualizado em vários dispositivos com tamanhos diferentes de tela. O nome que damos a essa característica é **responsividade**. Um site dito responsivo vai ser capaz adaptar os tamanhos dos seus componentes para manter o site plenamente visível em qualquer tela. É claro que por enquanto, não aprendemos muita coisa para permitir essa adaptação completamente para todo tipo de situação, mas com o que vimos até aqui já dá pra começar.



# Largura e altura adaptáveis

No capítulo anterior, vimos as propriedades para largura (`width`) e altura (`height`) de uma caixa. Se não configurarmos a largura, por exemplo, ela vai ocupar 100% do seu contêiner. Porém, para que nosso site se adapte mais facilmente ao tamanho da tela, uma das técnicas básicas que podem ser utilizadas é deixar algumas caixas estratégicamente um pouco mais flexíveis indicando valores mínimos (`min-width` / `min-height`) e máximos (`max-width` / `max-height`). Para o nosso projeto, vamos definir os limites da largura da caixa `<main>` onde estará contido o conteúdo principal do artigo.

```
min-width: 400px;  
max-width: 800px;
```

A configuração acima vai garantir que a caixa em questão não fique com largura menor que 400px (para telas pequenas) e nem seja maior que 800px (para telas muito grandes).

Para medidas entre 400px e 800px, o tamanho será adaptável e a caixa vai ocupar 100% da largura do contêiner. Isso vai facilitar bastante a adaptação do conteúdo a diversos tipos de tela.



**DÁ PRA SER MAIS ADAPTÁVEL?** É possível aplicar outras técnicas adicionais para dar ainda mais capacidade de adaptação do conteúdo, como as *media queries* com a regra `@media` e as caixas flexíveis com as *flex boxes*. Mais pra frente, abordaremos esses conteúdos no curso, mas por enquanto as medidas adaptáveis já vão quebrar um galho.

## IMPORTANTE: variáveis em CSS

Com essa dica, você vai levar suas folhas de estilo a um outro nível! Vamos aprender a utilizar variáveis personalizadas com CSS para cadastrar os esquemas de cores e fontes do nosso site e isso vai facilitar muito na hora de efetuar eventuais mudanças estéticas no nosso site.

Para criarmos variáveis para nossas configurações, devemos definir uma área de definições dentro do seu estilo para uma pseudo-classe chamada `:root`, que definem as configurações para a raiz de uma árvore, que vai servir para o documento inteiro.

Note que, pelas declarações criadas ao lado, definimos nove variáveis com as configurações de cores e fontes que definimos no início desse capítulo.

```
:root {  
  --cor0: #c5ebd6;  
  --cor1: #83E1AD;  
  --cor2: #3DDC84;  
  --cor3: #2FA866;  
  --cor4: #1A5C37;  
  --cor5: #063d1e;  
  --fonte0: Arial, Helvetica, sans-serif;  
  --fonte1: 'Bebas Neue', cursive;  
  --fonte2: 'Android', sans-serif;  
}
```

A partir de agora, definir cores e fontes em nossos elementos HTML ficará extremamente mais fácil e personalizável, utilizando a função var().

```
body {  
    font-family: var(--fonte0);  
    color: var(--cor0);  
    background-color: var(--cor1);  
}
```



**IMPORTANTE:** As variáveis personalizadas em CSS devem ter seus nomes iniciando com dois traços obrigatoriamente.

Olhando as declarações feitas para o seletor de body acima, pode parecer inicialmente que elas ficaram um pouco mais confusas. Mas imagine que seu cliente mude as definições de cores e fontes. Quantas alterações teriam que ser feitas para deixar tudo em dia com as novas especificações? Ao usar variáveis, tudo se simplifica muito, pois basta atualizar as variáveis definidas em :root.

## Uso do seletor \* em CSS

Existe também um seletor especial das CSS que é o asterisco (\*), ele tem uma função muito especial, pois basicamente ele aplica uma configuração padrão para **TODOS** os elementos do código HTML ao qual o estilo está sendo aplicado.

No nosso caso, para o nosso mini-projeto, nós vamos utilizar esse seletor global para eliminar as eventuais margens que os navegadores (*user agents*) adicionam a alguns elementos. Isso vai facilitar bastante, pois vai permitir personalizar as medidas que vão aparecer na tela para cada elemento individualmente.

## Espaçamento entrelinhas em Textos

Outra configuração muito importante que podemos fazer para textos muito longos é o de espaçamento entre as linhas do nosso texto. Usando a propriedade line-height, podemos dizer qual é o tamanho do espaço entre uma linha e a outra do texto. O valor padrão na maioria dos navegadores é algo próximo ao 1.2em, mas veja a seguir a aplicação de outros valores.

Provavelmente você sabe que o sistema operacional <b>Android</b> , mantido pelo <b>Google</b> é um dos mais utilizados para dispositivos móveis em todo o mundo. Mas tavez você não saiba que o seu simpático mascote tem um nome e uma história muito curiosa? Pois acompanhe esse artigo para aprender muita coisa sobre esse robozinho.	Provavelmente você sabe que o sistema operacional <b>Android</b> , mantido pelo <b>Google</b> é um dos mais utilizados para dispositivos móveis em todo o mundo. Mas tavez você não saiba que o seu simpático mascote tem um nome e uma história muito curiosa? Pois acompanhe esse artigo para aprender muita coisa sobre esse robozinho.	Provavelmente você sabe que o sistema operacional <b>Android</b> , mantido pelo <b>Google</b> é um dos mais utilizados para dispositivos móveis em todo o mundo. Mas tavez você não saiba que o seu simpático mascote tem um nome e uma história muito curiosa? Pois acompanhe esse artigo para aprender muita coisa sobre esse robozinho.
line-height: 1.2em;	line-height: 1.5em;	line-height: 2.0em;

# Sombras em Textos

No capítulo anterior, falamos sobre as sombras em caixas, utilizando a propriedade box-shadow. Pois saiba que também existe uma propriedade específica para criar sombras em textos: o text-shadow.

A propriedade text-shadow também pode ter quatro parâmetros principais:

1. **Deslocamento horizontal** (*h-offset*): quanto a sombra vai andar para o lado direito (valores negativos causam deslocamento para a esquerda)
2. **Deslocamento vertical** (*v-offset*): quanto a sombra vai andar para baixo (valores negativos causam deslocamento para cima)
3. **Embaçamento** (*blur*): quanto a sombra vai se espalhar pelo fundo
4. **Cor** (*color*): cor da sombra. É possível usar transparência.

```
text-shadow: 2px 2px 0px #rgba(0, 0, 0, 0.466);
```

CURIOSIDADES DE TECNOLOGIA	
Tudo aquilo que você sempre quis saber sobre o mundo Tech, em um único lugar.	Tudo aquilo que você sempre quis saber sobre o mundo Tech, em um único lugar.
<b>Sem</b> text-shadow	<b>Com</b> text-shadow

Como você deve ter percebido olhando as imagens acima, a sombra em textos serve para destacar a letra e seu fundo, criando um contraste entre elas.

## Personalizando ainda mais as listas

No **capítulo 9** nós aprendemos a criar listas de vários tipos. Agora, vou te mostrar como criar mais personalizações e a criar um ótimo resultado visual.

No nosso projeto, quero adicionar uma lista com todas as 14 versões principais do sistema Android. Se fizermos usando apenas os elementos comuns sem configurá-los, teremos uma listagem que ocupa um grande espaço vertical. A solução aqui é dividir a lista em duas colunas e modificar o marcador para personalizar ainda mais a exibição do conteúdo.

```
ul {  
    list-style-type: '\2714\0020\0020';  
    columns: 2;  
    list-style-position: inside;  
}
```

A primeira linha de declarações faz com que o marcador seja personalizado com o parâmetro list-style-type. O valor \2714 corresponde ao símbolo ✓ que tem o

código Unicode U+2714 (confira no site da Emojipedia). O valor \0020 corresponde a um espaço em branco (também pode ser \00A0).

A segunda declaração vai organizar a lista em duas colunas. O total de elementos da lista com `<li>` será dividido em duas partes iguais (ou quase) e o resultado será colunado.

Por fim, a última declaração vai fazer com que os marcadores sejam exibidos na parte interna da caixa que contém a lista. Analise as imagens abaixo e perceba que, por padrão, a caixa de uma lista não inclui os marcadores. Alteramos essa característica usando a declaração `list-style-position` com o valor `inside`, já que a lista vai estar dentro de um `<aside>` no nosso documento HTML.

 <code>list-style-position: outside;</code>	 <code>list-style-position: inside;</code>
---	---

## Vídeos do YouTube mais flexíveis

Aprendemos no **capítulo 11** como deixar as imagens mais “flexíveis” usando a tag `<picture>`. Também aprendemos a adicionar vídeos usando várias técnicas, inclusive aqueles que estão hospedados em serviços especializados como **Vimeo** e **YouTube**.

O problema é que, quando incorporamos um vídeo do YouTube ou Vimeo, isso é feito através de uma tag `<iframe>` que já vem com as configurações de um tamanho fixo e precisamos alterar isso para que nossa interface possa se tornar mais responsiva e adaptar o tamanho do vídeo dinamicamente, principalmente para telas pequenas.

Para isso, vamos colocar o `<iframe>` de incorporação dentro de uma `<div>` para que o vídeo esteja limitado dentro de um contêiner.

```
<div class="video">
    <!-- AQUI VAI O CÓDIGO DO IFRAME -->
</div>
```

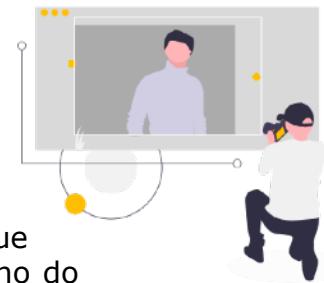
A partir daí, vamos fazer configurações de estilo para os dois elementos:

```
div.video {
    position: relative;
    background-color: var(--cor4);
    height: 0px;
    margin-left: -20px;
    margin-right: -20px;
    margin-bottom: 15px;
    padding-bottom: 59%;
}

div.video > iframe {
    position: absolute;
    top: 5%;
    left: 5%;
    width: 90%;
    height: 90%;
}
```

O que nos importa mais aqui é entender o funcionamento da propriedade position, que é a única que não vimos até aqui. O valor padrão para essa propriedade de posicionamento é static, que mantém a hierarquia conforme estabelecido no documento HTML.

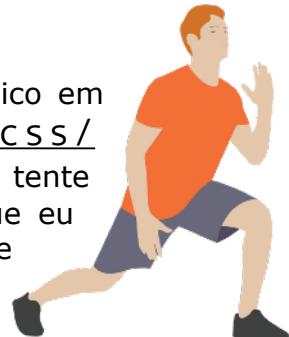
Na div, nós colocamos o valor como relative para que seja considerado o posicionamento atual do elemento de divisão e que ele se mantenha adaptável para o caso de alteração no tamanho do navegador.



Já dentro do iframe, nós usamos o posicionamento absolute para que a div - que é o seu contêiner - torne-se o ponto de partida para o posicionamento do frame. A partir daí, podemos utilizar propriedades para configurar o deslocamento à esquerda (left) e ao topo (top) e seu tamanho em largura (width) e altura (height), todos em porcentagem de tela.

## Hora de exercitar

Chegou a hora de acessar o endereço do nosso repositório público em <https://gustavoguanabara.github.io/html-css/desafios/> e executar o **desafio 010** no seu computador. Agora tente atingir esse mesmo resultado em casa, sem copiar o código que eu criei. Nesse momento, a prática é algo que você mais precisa. Se por acaso ficar difícil, pode acessar o repositório público de HTML e CSS e dar uma olhada nos comandos, mas **EVITE COPIAR**.



## Eu já falei sobre isso no YouTube?

Eu sei que às vezes as pessoas gostam mais de assistir vídeos do que ler livros, e é por isso que eu lanço há anos materiais no canal Curso em Vídeo no YouTube. O link que vou compartilhar contigo tem o conteúdo explicado como você leu aqui, só que de forma mais ilustrada. Reserve um tempo dos seus estudos para assistir esse vídeo todo.



Curso em Vídeo: [https://www.youtube.com/playlist?list=PLHz\\_AreHm4d1AnJ\\_jJtV29RFxnPHDuk9o](https://www.youtube.com/playlist?list=PLHz_AreHm4d1AnJ_jJtV29RFxnPHDuk9o)

# DevWeb

## Capítulo 18

### Aprendendo Git e GitHub

Talvez o conteúdo desse capítulo seja um dos que você mais vai usar durante a sua carreira, que está começando agora. Vamos aprender agora o que são os repositórios, tanto os locais quanto os remotos e qual é a diferença entre eles. E por fim, vamos aprender a colocar nossos projetos no ar, para que possamos acessar os sites em qualquer lugar com conexão à Internet.



Você tem todo o direito de usar esse material para seu próprio aprendizado. Professores também podem ter acesso

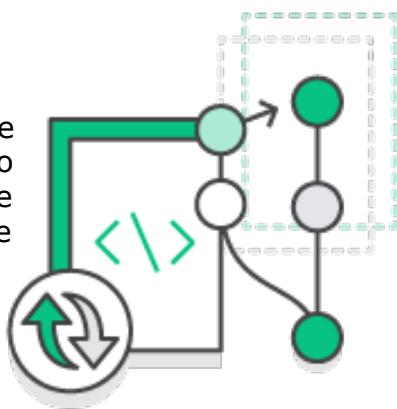
a todo o conteúdo e usá-los com seus alunos. Porém todos o que usarem esse material - seja para qual for a finalidade - deverão manter a referência ao material original, criado pelo **Prof.**

**Gustavo Guanabara** e disponível no endereço do seu repositório público <https://github.com/gustavoguanabara/>. Este material não poderá ser utilizado em nenhuma hipótese para ser replicada - integral ou parcialmente - por autores/editoras para criar livros ou apostilas, com finalidade de obter ganho financeiro com ele.



# Você precisa de um repositório para ser feliz

Talvez nesse exato momento você não faça ideia do que seja um **repositório** e esteja achando que minha afirmação acima foi um pouco exagerada, mas saiba que vou me esforçar para que você consiga entender o uso desse recurso e adote na sua vida de hoje em diante. Repositórios não vão te ajudar apenas no desenvolvimento em **HTML** e **CSS**, eles também serão úteis na criação de qualquer tipo de código em qualquer linguagem de programação que você vá decidir aprender de hoje em diante.



Para explicar a utilidade de cada um dos dois tipos de repositórios que vamos usar (**locais** e **remotos**), criei três pequenas histórias que podem acontecer na sua vida.

## Situação 1: como guardar várias versões do seu site?



Quando estamos criando um projeto, começamos aos poucos criando a sua primeira versão dentro de uma pasta local em nosso computador.

Quando atingir uma versão estável, para manter um “*backup seguro*”, sempre usamos ideias brilhantes como gerar uma versão compactada dessa pasta. Isso vai permitir continuar a desenvolver uma versão aprimorada sem perder a versão anterior. Se por acaso algo der errado (e acredite, geralmente dá) basta descompactar nosso arquivo de versão estável, substituir o conteúdo da pasta original e recomeçar os trabalhos da criação da próxima versão. Usando essa prática, com certeza vai acabar se deparando com vários arquivos ZIP com nomes esquisitos.



Esses arquivos compactados ficam se acumulando no seu computador e são úteis caso você precise “voltar no tempo” e desfazer as bobagens que fizemos na madrugada passada quando tentamos desenvolver algo com muito sono.

## Situação 2: seu HD foi pro saco, e agora?

Quem nunca perdeu arquivos de trabalho, que atire o primeiro HD queimado! Os dispositivos de hardware sempre param de funcionar no momento em que mais precisamos. E se utilizamos apenas a técnica descrita na situação anteriormente, é

possível que todo o trabalho seja perdido em um estalar de dedos.

Uma das saídas utilizadas é salvar cópias de segurança em *pendrives* e em HDs externos, além de manter um *backup* em serviços de armazenamento como **Dropbox**, **Google Drive**, **OneDrive**, **iCloud**, etc.



O problema de manter esses *backups* isolados é que, no momento de ligar nossa “máquina do tempo” e voltar para uma versão anterior, precisamos fazer o *download* do ZIP desejado e substituir manualmente a versão atual para a última estável.

## Situação 3: mostrar seu site para o mundo

E o que acontece quando você precisar mostrar o projeto de um site que você criou para um amigo, para seu professor ou até mesmo para um possível cliente? Vai mandar o link de um arquivo ZIP para ele(a) baixar, descompactar e abrir no navegador local? É assim que as pessoas estão acostumadas a acessar sites?

Os serviços de armazenamento que eu citei na situação anterior são muito úteis para fazer *backup*, mas possuem uma grande limitação quando queremos praticidade. Por exemplo, salvar os arquivos de um site no **Dropbox** não vai garantir de forma alguma que ele vai ficar disponível para o acesso externo. Ninguém vai poder “acessar seu site”, apenas digitando uma URL no navegador. Tudo vai ficar armazenado como um grupo de arquivos, não como um site hospedado.



**ATÉ DÁ, MAS...** Existe um recurso do **Google Drive** para a hospedagem de sites, mas nada muito simples e intuitivo. A melhor maneira é sempre usar serviços especializados em hospedagem de sites.

## Os repositórios vão resolver isso

Todas as três situações descritas podem ser resolvidas com repositórios locais/remotos de maneira muito simples. E se por acaso você já tentou aprender sobre isso, achou guias com dezenas de comandos e não conseguiu se adaptar, saiba que tudo evoluiu muito nos últimos meses e vamos ver tudo de uma maneira muito simples, sem decorar nenhum comando! Para começar, vamos aprender que existem dois tipos de repositórios: os **locais** e os **remotos**.

# Repositório Local

Um **repositório local** tem esse nome porque vai estar sempre no seu computador. A função dele é facilitar a gestão de diversas versões do seu projeto de forma simples e automática.



Lembra da **situação 1** que descrevemos anteriormente? Pois quando instalamos um **Sistema de Controle de Versões** (do inglês *Version Control System - VCS*), deixamos a responsabilidade por gerenciar as versões dos nossos projetos nas mãos de um *software* especializado, que vai fazer tudo automaticamente e vai permitir que você volte a qualquer ponto no momento em que achar melhor.

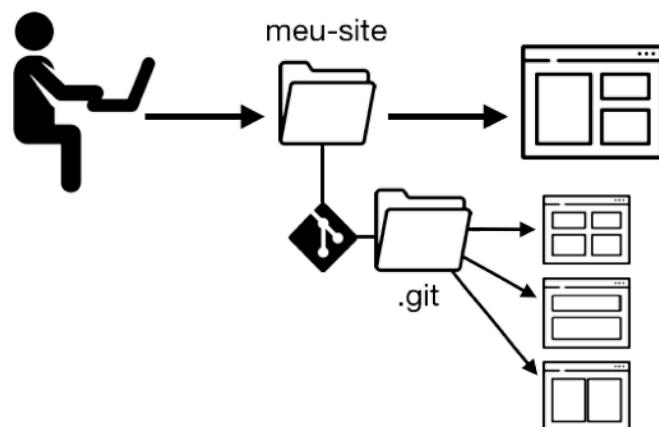
Um dos *softwares* VCS mais famosos é o **Git**, feito por **Linus Torvalds**, o mesmo criador do núcleo **Linux** dos sistemas operacionais. O sistema do Git foi criado no ano de 2005 em poucos dias (10 dias, para ser mais exato), por conta de uma briga entre o *Linus* e o criador de um software chamado **BitKeeper**, que era utilizado para gerenciar as versões em desenvolvimento do *Linux*.



**QUER SABER MAIS SOBRE A TRETA?** Não vou usar espaço desse material para contar a treta entre **Linus Torvalds** e **Larry McVoy**, mas já contei essa história no YouTube. Se você ficou curioso(a) para saber o que rolou, acesse o link a seguir:

Curso de Git e GitHub: <https://youtu.be/CJtrNuTTs4Q>

O esquema de funcionamento do Git é totalmente focado no nosso computador. Analisando a imagem a seguir, vemos que o software está monitorando uma pasta chamada *meu-site* que tem a versão atual do projeto e uma pasta especial chamada *.git* com várias versões pelas quais o site passou durante sua evolução.

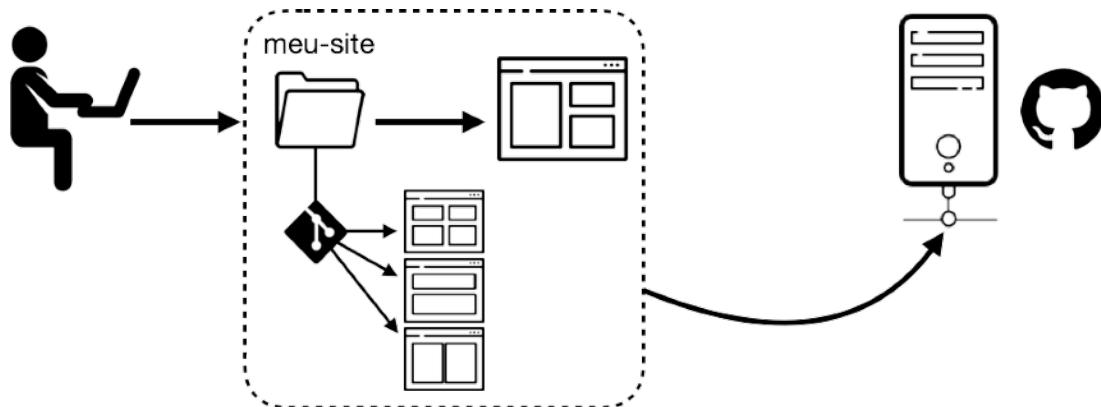


A grande vantagem de usar um **sistema de controle de versões** como o **Git** é poder voltar a qualquer momento para qualquer versão anterior do projeto de forma imediata, tudo 100% transparente para o programador.

Como você já deve ter percebido, os repositórios locais resolvem o problema que apontamos na **situação 1**, mas se o nosso PC quebrar (**situação 2**) ou se quisermos mostrar o projeto para outra pessoa (**situação 3**), ainda não é possível usando esse tipo de sistema, mas não se desespere agora! Continue lendo...

## Repositório Remoto

Para solucionar os problemas levantados nas duas últimas situações desse capítulo, vamos precisar de um lugar na nuvem para guardar nossos repositórios locais. E é aí que entra o nosso segundo grande personagem: o **GitHub**.



Criado em 2008 por quatro amigos, o **GitHub** é um serviço que nos permite criar um **repositório remoto** na nuvem para guardar nossos projetos e versionamentos. Ele não é a única opção que existe no mercado (ainda temos o **GitLab**, o **Bitbucket**, e muitos outros) mas provavelmente é a mais popular, tanto que hoje pertence à gigante **Microsoft**, que comprou o serviço em 2018 por 7.5 bilhões de dólares.

Com o tempo, o GitHub começou a ganhar funcionalidades extras, que foram transformando o serviço em uma grande **rede social para programadores**. Além de guardarmos nossos códigos nos servidores, podemos nos comunicar com outros desenvolvedores e até mesmo colaborar com outros projetos que estão disponíveis publicamente para todos.



Ao colocar nossos códigos na nuvem, automaticamente resolvemos o problema da **situação 2**. E um recurso chamado **GitHub Pages** vai permitir a hospedagem gratuita de sites simples, que usem HTML + CSS + JS e disponibilize o acesso através de uma URL. Por exemplo, o mini-projeto que construímos no capítulo anterior está disponibilizado no endereço a seguir:

<https://professorguanabara.github.io/projeto-android/>

O código original também está disponibilizado publicamente no endereço abaixo:

<https://github.com/professorguanabara/projeto-android>

# Como instalar e usar isso tudo?

Para poder aproveitar todos esses maravilhosos recursos que explicamos até aqui, precisamos instalar dois softwares no nosso computador: o **Git-SCM** e o **GitHub Desktop**. Também precisamos criar um perfil gratuito no site **GitHub**.

Para fazer o download e instalação do **Git**, acesse o site [git-scm.com](http://git-scm.com) e clique sobre o botão de *download* da versão mais recente e instale o programa.

Nesse site, encontramos as versões para todos os sistemas: Windows, Linux e MacOS.

Para a instalação no Linux, o site mostra os comandos necessários para realizar a instalação via gerenciador de pacotes.

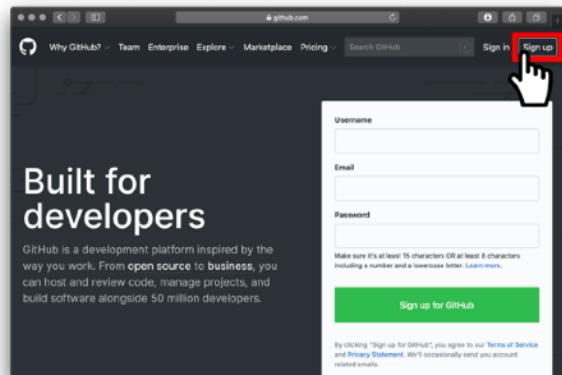


Depois de instalar as duas ferramentas acima, acesse o site [github.com](http://github.com) e crie o seu perfil clicando no botão **Sign up** indicado na imagem a seguir. Você só vai precisar de um nome de usuário, e-mail de contato e uma senha segura. Um e-mail de verificação será enviado para ativar sua conta.

Se já tiver uma conta criada, clique sobre o botão **Sign in** ao lado.

Para fazer o download e instalação do programa **GitHub Desktop**, acesse o site [desktop.github.com](http://desktop.github.com) e clique sobre o botão de *download* da versão atual e instale.

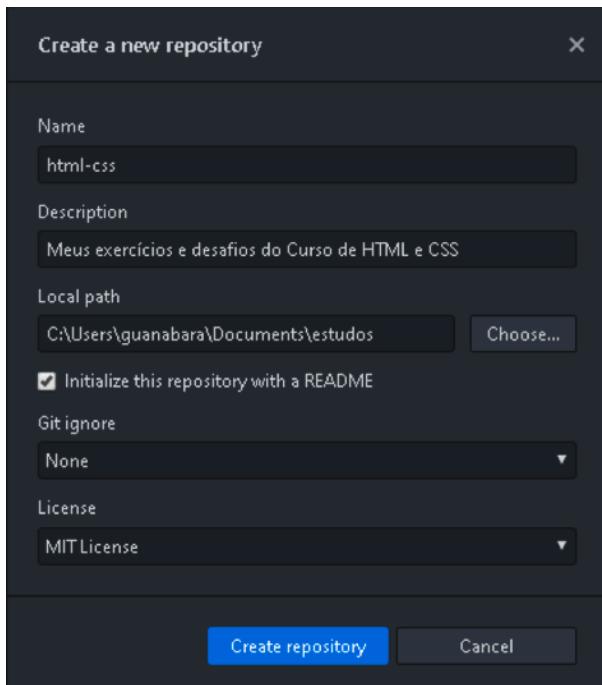
O GitHub Desktop também existe em versões para os sistemas Windows, Linux e MacOS. A versão para Windows só existe para sistemas 64-bits. Em Linux, devemos instalar via repositório.



**FAÇA ESCOLHAS SÁBIAS!** Tenha muito cuidado na hora de escolher seu nome de usuário. Ele fará parte do seu futuro profissional e pode ser que você precise divulgar seu perfil em uma entrevista de emprego. Fuja daqueles nomes engraçadinhos e agressivos.

# Criando o seu primeiro repositório

Abra o **GitHub Desktop** (com conexão ativa à Internet, claro) e faça login na sua conta criada no **GitHub** diretamente na tela do programa. Depois de confirmar a conexão, crie um novo repositório em File > New Repository ou pressione Ctrl+N.



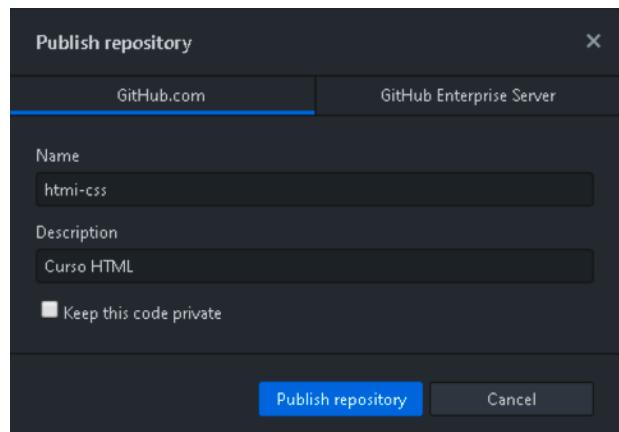
Assim que solicitar a publicação do repositório, deve escolher se vai deixá-lo **público** ou **privado** (padrão). Desmarque a opção "Keep this Code private", principalmente se deseja utilizar o recurso do **GitHub Pages** e hospedar gratuitamente seu site HTML. Só projetos públicos podem ser hospedados e disponibilizados para outros.

A interface do **GitHub Desktop** é bem simples, mas concentra muitas funcionalidades que unificam recursos do **Git** e do **GitHub** em um só lugar.

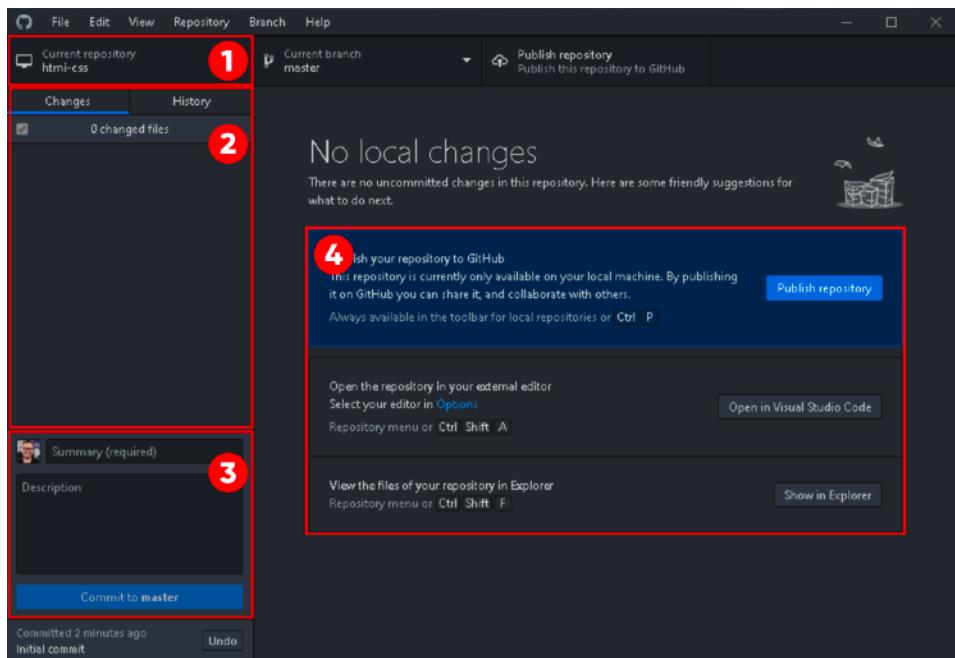
Para começar a versionar os arquivos que criamos durante o curso, vamos criar o repositório de nome `html-css` (mesmo nome da pasta que criamos) dentro da pasta `estudos` que está dentro dos nossos Documentos. Não se esqueça de marcar a opção para criar o arquivo `README.md` e escolher uma licença. No fim, clique em **Create repository**.

Este procedimento vai criar nosso **repositório local** via **Git**.

Depois de tudo, clique no botão **Publish Repository** para criar o nosso **repositório remoto** no **GitHub** e todos os arquivos serão enviados para a nuvem.



**MUITO CONFUSO? TEREMOS VÍDEOS!** Não se desespere se não estiver conseguindo realizar os procedimentos seguindo apenas esse material escrito. Ele será complementado com uma série de vídeos que estão sendo cuidadosamente criados pra você.



Na **área 1**, escolhemos o repositório ativo no momento. Clicando nessa área, podemos mudar entre todos os repositórios locais que temos no computador.

Já na **área 2**, vão aparecer todas as mudanças detectadas pelo Git nos arquivos que estão na pasta do repositório.

Usaremos a **área 3** sempre que quisermos estabelecer uma versão estável ou submeter uma alteração importante do nosso projeto. Ao colocar uma descrição na caixa **Summary** (resumo) e clicar no botão **Commit** (pode ser entendido como **compromisso** ou como **enviar**). Fazer um *commit* ou “comitar” (jargão técnico em Português) vai criar uma versão **local** do repositório usando o Git.

Depois de *comitar* um repositório local, podemos realizar uma operação de **Push** (empurrar) que vai transferir todos os códigos para o **GitHub**. O botão *push* vai aparecer na **área 4** logo após um *commit*. Também podemos realizar uma operação dessas clicando no menu Repository > Push ou ainda pressionar Ctrl + P.

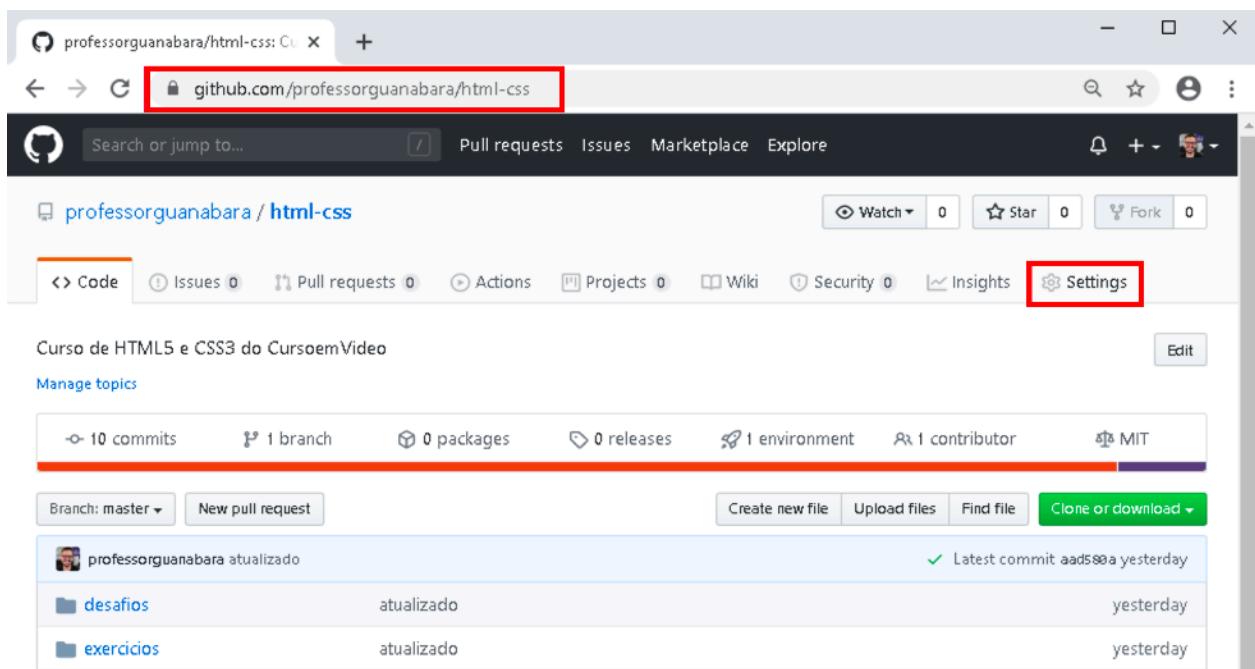
Na **área 4**, também vão aparecer botões muito importantes para abrir o projeto usando o Visual Studio Code ou abrir o gerenciador de arquivos do seu sistema com a pasta do projeto.

## Habilitando a Hospedagem Grátis

O recurso do **GitHub Pages** não está ativo por padrão em nossos repositórios remotos, mas é muito fácil de habilitar. Isso só pode ser feito diretamente no site do GitHub, individualmente para cada repositório que queremos usar para hospedar projetos HTML.

Acesse o seu perfil no site <https://github.com/username> (substituindo pelo nome do seu usuário, claro) e abra o repositório que quer habilitar a hospedagem. Em seguida, clique sobre a aba **Settings** e role a tela até a área **GitHub Pages**.

A imagem a seguir mostra a tela principal do repositório no site. Observe atentamente as áreas demarcadas, pois elas mostram a posição dos componentes.



Na área de opções, existem duas maneiras de habilitar o GitHub Pages. Na primeira, clique no botão **None** e escolha a opção **master branch**. Com essa maneira, você vai precisar ter um arquivo `index.html` com o código da página principal que será exibida ao acessar o site. Essa técnica é mais usada quando queremos hospedar um site único e receber uma URL para apresentar o projeto a outra pessoa.

## GitHub Pages

[GitHub Pages](#) is designed to host your personal, organization, or project pages from a GitHub repository.

### Source

GitHub Pages is currently disabled. Select a source below to enable GitHub Pages for this repository.  
[Learn more](#).

[None](#) ▾

### Theme Chooser

Select a theme to publish your site with a Jekyll theme using the `master` branch. [Learn more](#).

[Choose a theme](#)

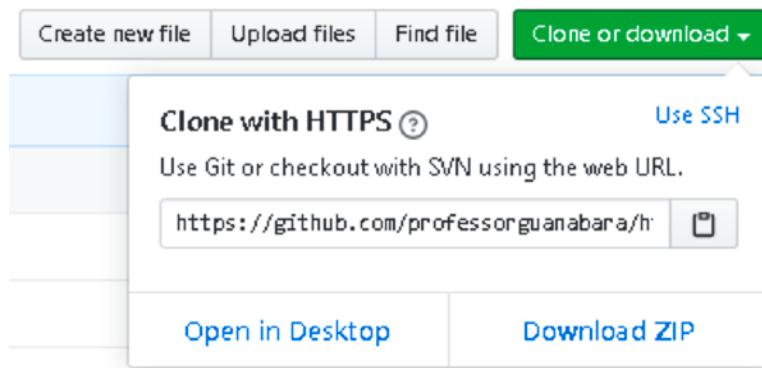
A segunda maneira é clicando sobre o botão **Choose a theme**, que vai permitir escolher um tema para exibir o conteúdo do arquivo `README.md` como uma página. Essa segunda técnica é mais usada quando vamos hospedar vários projetos em um mesmo repositório ou quando vamos guardar informações sobre um projeto que não é um site, como download de instaladores e coisas do tipo.



**IMPORTANTE:** O GitHub pode ser usado para guardar códigos em todas as linguagens, incluindo o PHP. Porém, o recurso **GitHub Pages** não vai servir para hospedar o projeto, gerando link para acesso à sua execução. Ele só serve para hospedar HTML + CSS + JavaScript.

# Clonando um repositório

Outra operação essencial para quem vai usar o **GitHub** para hospedar repositórios remotos é **clonar** um projeto. Para isso, devemos acessar a URL do repositório no site [github.com](https://github.com) e procurar o botão **Clone or download** ao lado direito da tela.



Ao clicar sobre o botão verde, escolha a opção **Open in Desktop** e o programa **GitHub Desktop** vai abrir automaticamente e os arquivos serão baixados para o seu computador, criando um repositório local.

Podemos clonar qualquer projeto do **nosso** repositório remoto (público ou privado) ou até mesmo repositórios públicos de **outras pessoas**. Se o repositório for seu, você ainda vai poder atualizá-lo com as alterações que fizer. Já para repositórios de outras pessoas, você não vai poder efetuar alterações e efetuar operações de *push*, a não ser que o proprietário autorize (através das *Pull Requests*).

## Eu já falei sobre isso no YouTube?

Eu sei que às vezes as pessoas gostam mais de assistir vídeos do que ler livros, e é por isso que eu lanço há anos materiais no canal Curso em Vídeo no YouTube. O link que vou compartilhar contigo tem o conteúdo explicado como você leu aqui, só que de forma mais ilustrada. Reserve um tempo dos seus estudos para assistir esse vídeo todo.



Curso em Vídeo: [https://www.youtube.com/playlist?list=PLHz\\_AreHm4dm7ZULPAmadvNhH6vk9oNZA](https://www.youtube.com/playlist?list=PLHz_AreHm4dm7ZULPAmadvNhH6vk9oNZA)

# DevWeb

## Capítulo 19

### Imagens de Fundo

Nos capítulos 6 e 11 do nosso material, aprendemos técnicas para usar imagens como parte do nosso conteúdo. Agora, vamos aprender como usar algumas imagens para complementar visualmente um site, aplicando-as aos fundos dos nossos elementos HTML utilizando estilos. Também vamos ver como manter essas imagens adaptáveis ao tamanho do navegador dos nossos visitantes. Vamos nessa?



Você tem todo o direito de usar esse material para seu próprio aprendizado. Professores também podem ter acesso

a todo o conteúdo e usá-lo com seus alunos. Porém todos os que usarem esse material - seja para qual for a finalidade - deverão manter a referência ao material original, criado pelo **Prof.**

**Gustavo Guanabara** e disponível no endereço do seu repositório público <https://github.com/gustavoguanabara/>. Este material não poderá ser utilizado em nenhuma hipótese para ser replicada - integral ou parcialmente - por autores/editoras para criar livros ou apostilas, com finalidade de obter ganho financeiro com ele.



# O fundo não precisa ter só cor

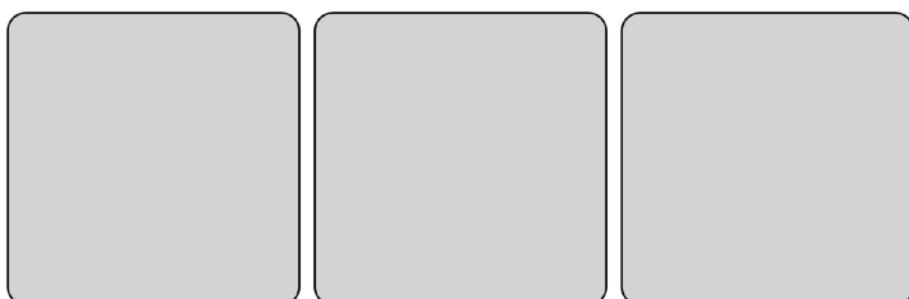
Em capítulos anteriores, aprendemos a aplicar cores sólidas ou degradê em qualquer elemento de caixa. Mas você não precisa se limitar a essa possibilidade e pode aplicar imagens ao fundo de qualquer elemento exibido visualmente em HTML. Vamos a um exemplo simples com três `<div>` no corpo de um documento:

```
<body>
  <div class="quadrado" id="q1"></div>
  <div class="quadrado" id="q2"></div>
  <div class="quadrado" id="q3"></div>
</body>
```

Agora vamos criar a configuração de estilo base para toda `<div>` que possui a classe quadrado:

```
<head>
  <style>
    div.quadrado {
      display: inline-block;
      margin: 5px;
      width: 300px;
      height: 300px;
      background-color: lightgray;
      border: 2px solid black;
      border-radius: 20px;
    }
  </style>
</head>
```

Com esse código aplicado, temos o seguinte resultado na tela:



Viu? Resultado simples, três quadrados exatamente iguais na tela.



**NÃO ENTENDEU?** Se você criou o código acima, mas não obteve o resultado apresentado, é sinal de que talvez você esteja tentando correr demais com seu aprendizado. Se eu puder te dar um conselho, volte ao capítulo 16 e faça todos os exercícios.

Agora vamos adicionar algumas configurações individualmente a cada quadrado, usando os identificadores diferentes entre eles:

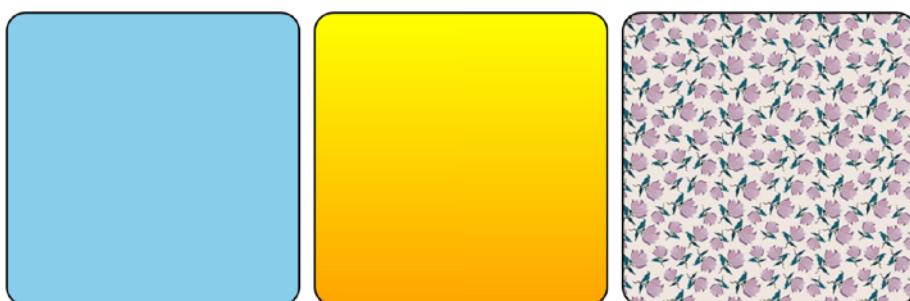
```
div#q1 {  
    background-color: #skyblue;  
}  
  
div#q2 {  
    background-image: linear-gradient(to bottom, #yellow, #orange);  
}  
  
div#q3 {  
    background-image: url('imagens/pattern003.png');  
}
```

Note que na definição da <div> que tem o id com o valor q1, usamos uma cor sólida, já para q2, usamos um preenchimento linear. Já na caixa q3, vamos aplicar uma imagem de fundo através de um endereço informado pela função url().



**CADÊ AS IMAGENS?** Se você quer as imagens que usaremos nesse capítulo, acesse nosso repositório em <https://github.com/gustavoguanabara/html-css/tree/master/exercicios/ex022>. Lá você vai encontrar a pasta com várias imagens que usaremos.

Ao adicionar as linhas acima ao código, o resultado já muda consideravelmente:



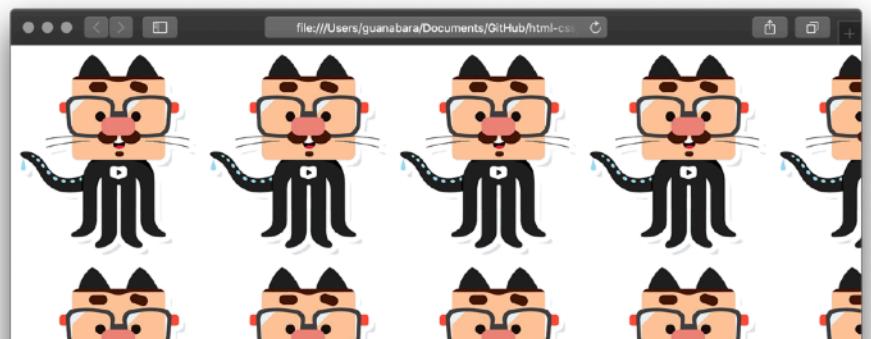
De forma geral, essas são as três maneiras mais simples de preencher uma caixa em HTML: cor sólida, degradê ou imagem de fundo.

## E não se esqueça: o **body** é uma grande caixa

Esse recurso também pode ser usado ao <body> do seu site e o papel de parede será aplicado. Vamos considerar um exemplo bem simples que usa uma imagem disponível no nosso repositório.

```
<style>
  body {
    background-image: url('https://gustavoguanabara.github.io/html-css/imagens/mascote.png');
  }
</style>
```

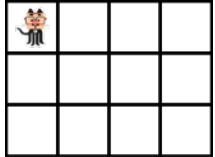
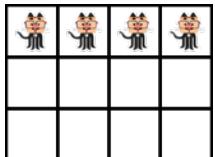
Analisando o código acima, estamos usando uma `url()` externa, já que a imagem estará em outro servidor (no caso, no servidor do **GitHub**). A imagem em questão é razoavelmente pequena, então não vai ser suficiente para cobrir toda a tela. Sendo assim, o resultado será o seguinte:



## Personalizando a aplicação do background

Quando o tamanho da caixa é maior que o tamanho da imagem, por padrão, a imagem será **repetida** nos dois eixos (*eixo x* e *eixo y*) quantas vezes for necessário para cobrir a extensão da caixa contêiner.

É possível alterar esse comportamento usando a propriedade `background-repeat`, que aceita os seguintes valores:

background-repeat: repeat;	
background-repeat: no-repeat;	
background-repeat: repeat-x;	
background-repeat: repeat-y;	

Note que ao usar o no-repeat, isso **não obriga** o navegador a aumentar ou diminuir o tamanho da imagem para caber no tamanho da caixa. Para realizar essa adaptação, devemos usar outra propriedade, que veremos mais adiante.

Além de escolher o nível de repetição do *background*, também podemos mudar a **posição de referência** de início das repetições. Por padrão, é considerado o canto esquerdo superior (left top), mas podemos ter várias opções. Use a imagem abaixo como referência sempre que precisar definir a posição do fundo com a propriedade background-position no seu código.

left top	center top	right top
left center	center center	right center
left bottom	center bottom	right bottom

Outra coisa que podemos fazer é **redimensionar** a imagem para forçá-la a caber na caixa. Por padrão, nenhum redimensionamento será aplicado, e a imagem será exibida do seu tamanho natural. Porém, podemos usar a propriedade background-size para alterar esse comportamento.

Os valores aceitos por essa propriedade são:

auto	(padrão) a imagem de fundo será aplicada em seu tamanho original.
[length]px [length]%	Redimensiona a largura da imagem e faz a altura se adaptar automaticamente. Podemos também informar as duas dimensões na sequência ou também usar valores percentuais.
cover	Muda o tamanho da imagem para que ela seja sempre totalmente exibida na tela, sem nenhum corte.
contain	Redimensiona a imagem para que ela cubra o contêiner, mesmo que para isso ocorram alguns eventuais cortes.



**SÓ CONSIGO DEMONSTRAR NA PRÁTICA.** Essas propriedades de personalização de imagens de fundo precisam de demonstração prática. Nesse exato momento, já gravei vários vídeos ensinando o uso de cada uma delas. Em breve você verá, basta acompanhar o canal **Curso em Vídeo** no **YouTube**

A última propriedade que podemos configurar é o **vínculo** (*attachment*) da imagem de fundo com o resto do documento, principalmente se o conteúdo for maior do que a altura da página e seja necessário vazar uma rolagem vertical.

A propriedade `background-attachment` aceita os valores:

scroll	(padrão) a imagem de fundo vai rolar junto com o conteúdo.
fixed	A imagem de fundo vai ficar fixada enquanto o conteúdo vai sendo rolado.

## Simplificando as coisas

Assim como já vimos várias vezes em nosso material, existe também a possibilidade de usar uma *shorthand* para simplificar o uso de propriedades que se apliquem ao fundo de uma caixa. A propriedade abreviada `background` pode ser declarada agrupando as seguintes configurações:

- `background-color`
- `background-image`
- `background-position`
- `background-repeat`
- `background-attachment`

Sendo assim, no lugar de usar:

```
background-color: black;  
background-image: url('imagens/wallpaper002.jpg');  
background-position: center center;  
background-repeat: no-repeat;  
background-attachment: fixed;
```

Podemos reunir tudo em uma única declaração:

```
background: black url('imagens/wallpaper002.jpg') center center no-repeat fixed;
```

## Centralização vertical em contêineres

Antes de começar a explicar o assunto sobre o qual vamos falar, preciso desabafar: plural de *contêiner* é muito esquisito, não acha? A propósito, o significado de *contêiner* (versão do Inglês *container*) é simples e direto: "aquele que contém coisas".



Aprendemos no **capítulo 16** que existem elementos que podem conter outros elementos. As `<div>` são um exemplo de elemento *contêiner*. Quando queremos centralizar blocos horizontalmente, aprendemos a usar o `margin:auto;` nas folhas de estilo. Mas como fazer a centralização vertical?

No **capítulo 17**, onde criamos o nosso primeiro mini-projeto, tivemos que arrumar uma maneira de centralizar e redimensionar um vídeo dentro de um *contêiner*. Agora vou te mostrar uma outra técnica.

Vamos começar criando uma hierarquia simples entre dois blocos:

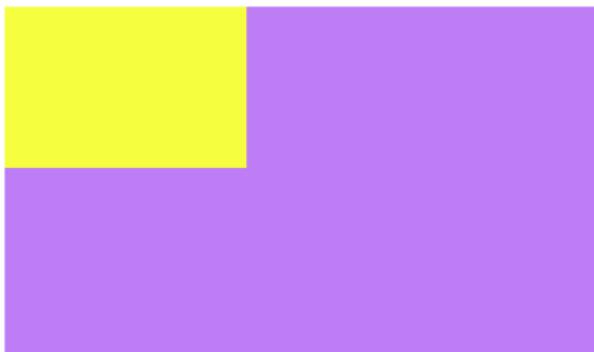
```
<div id="fora">
  <div id="dentro">
    </div>
  </div>
```

Agora vamos criar as configurações personalizadas para cada `<div>` dentro da área de `<style>` na área da cabeça `<head>` do código:

```
<style>
  div#fora {
    height: 96vh;
    background-color: #BD7DF5;
  }

  div#dentro {
    height: 200px;
    width: 300px;
    background-color: #F5FF40;
  }
```

O resultado disso será algo como:



O nosso objetivo aqui é deixar o retângulo interno exatamente no meio do retângulo externo. Vamos começar configurando o posicionamento de cada uma. O retângulo externo terá posicionamento relativo, enquanto o interno terá posicionamento absoluto.

```
div#fora {
  position: relative;
  ...
}

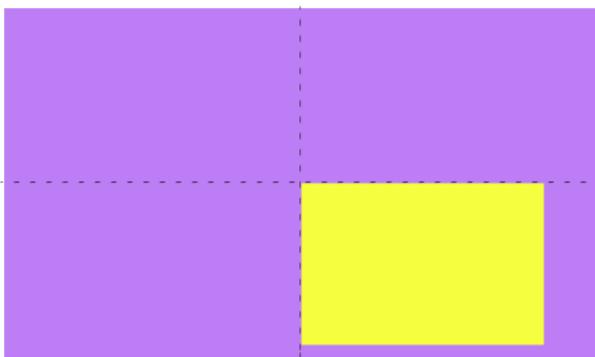
div#dentro {
  position: absolute;
  ...
}
```

Não crie um novo seletor, apenas adicione as declarações que estou indicando. Qualquer dúvida, assista o último vídeo relativo ao **capítulo 19** do curso.

Quando definimos um bloco com posicionamento absoluto, podemos personalizar a sua posição exata através das propriedades `left` e `top`. Como queremos posicionamento centralizado, vamos configurar os dois na metade do contêiner:

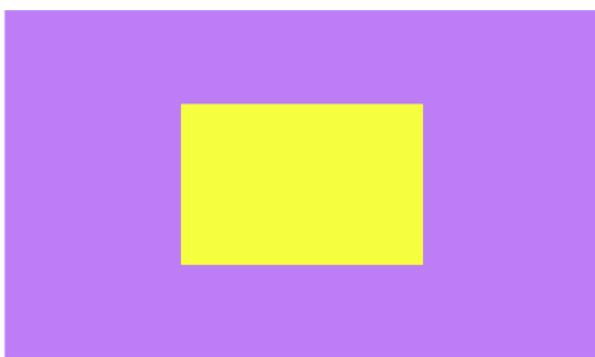
```
div#dentro {  
    position: absolute;  
  
    left: 50%;  
    top: 50%;  
  
    ...  
}
```

Porém, infelizmente, o resultado não será exatamente o que esperamos. Mas não se desespere! Nem tudo está perdido!



Note que, pelas linhas pontilhadas, o retângulo interno foi realmente posicionado a 50% da tela, mas pelo canto superior esquerdo da caixa. Vamos realizar uma transformação e mover o retângulo interno para a esquerda e para cima, para que ele fique efetivamente centralizado.

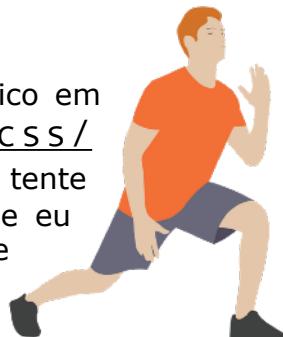
```
div#dentro {  
    ...  
    transform: translate(-50%, -50%);  
    ...  
}
```



E está feito! Essa é apenas uma das técnicas de centralização de conteúdo, mas as outras requerem aprender outros conceitos mais aprofundados das folhas de estilo, como as caixas flexíveis (Flexbox).

# Hora de exercitar

Chegou a hora de acessar o endereço do nosso repositório público em <https://gustavoguanabara.github.io/html-css/desafios/> e executar o **desafio 010** no seu computador. Agora tente atingir esse mesmo resultado em casa, sem copiar o código que eu criei. Nesse momento, a prática é algo que você mais precisa. Se por acaso ficar difícil, pode acessar o repositório público de HTML e CSS e dar uma olhada nos comandos, mas **EVITE COPIAR**.



## Eu já falei sobre isso no YouTube?

Eu sei que às vezes as pessoas gostam mais de assistir vídeos do que ler livros, e é por isso que eu lanço há anos materiais no canal Curso em Vídeo no YouTube. O link que vou compartilhar contigo tem o conteúdo explicado como você leu aqui, só que de forma mais ilustrada. Reserve um tempo dos seus estudos para assistir esse vídeo todo.



Curso em Vídeo: [https://www.youtube.com/playlist?list=PLHz\\_AreHm4d1AnJ\\_jJtV29RFxnPHDuk9o](https://www.youtube.com/playlist?list=PLHz_AreHm4d1AnJ_jJtV29RFxnPHDuk9o)

# DevWeb

## Capítulo 20

### Mini-projeto Cordel

Vamos começar agora mais um mini-projeto onde será possível aplicar conceitos importantes que vimos até o momento. Será um projeto rápido, mas vai incluir o efeito *parallax* às imagens de fundo e uma adaptação dinâmica do tamanho da fonte de acordo com o tamanho da tela. E o conteúdo que vamos dar vida, transformando em um site é um belo cordel.



Você tem todo o direito de usar esse material para seu próprio aprendizado. Professores também podem ter acesso

a todo o conteúdo e usá-los com seus alunos. Porém todos os que usarem esse material - seja para qual for a finalidade - deverão manter a referência ao material original, criado pelo **Prof.**

**Gustavo Guanabara** e disponível no endereço do seu repositório público <https://github.com/gustavoguanabara/>. Este material não poderá ser utilizado em nenhuma hipótese para ser replicada - integral ou parcialmente - por autores/editoras para criar livros ou apostilas, com finalidade de obter ganho financeiro com ele.



# A cultura da Literatura de Cordel

O Brasil é realmente um país múltiplo quando se fala em cultura, principalmente quando falamos de cultura popular. Uma dessas manifestações populares é a **literatura de cordel**. Ela tem esse nome pois os pequenos folhetos com os poemas são impressos ficam presos em cordas para a exposição.



A ideia dos folhetos não surgiu no Brasil e sim em Portugal, mas foi aqui que ganharam uma forma peculiar ao estampar as capas usando *xilogravuras*, formato muito popular em cidades do nordeste brasileiro.

Outra característica dos folhetos de cordel é ter poesias escritas em forma de versos que podem ser recitados de forma melodiosa e cadenciada, geralmente acompanhadas de uma música tocada com instrumentos tipicamente nordestinos.



**UM DOS MAIS FAMOSOS:** Quer conhecer um pouco mais sobre os cordéis? Acompanhe aqui um dos mais populares, que versa sobre o a história de Virgulino Ferreira, vulgo cangaceiro Lampião.

Causos de Cordel: <https://youtu.be/56zDjaM1iLg>

E se você está lendo essa introdução e se perguntando “*o que tem a ver cordel com HTML e CSS?*”, tenho um recado sincero pra você: a cultura faz parte da nossa identidade como seres humanos. Nunca desmereça a cultura, amplie seus conhecimentos e valorize sempre os movimentos nacionais. Você não perdeu tempo lendo tudo isso, você ganhou culturalmente. De nada!

Levantei esse assunto aqui, pois escolhi um poema em forma de cordel de **Milton Duarte** chamado “*Cordel Moderno - Tecnologia do agora*” que fala de Tecnologia (olha aí o assunto!) de uma maneira bem simpática e atual. Esse será o conteúdo do site que vamos criar.

## Download do pacote básico

Os arquivos básicos que usaremos para criar nosso site já estão disponíveis no nosso repositório oficial do GitHub. Comece acessando o endereço a seguir:

<https://github.com/gustavoguanabara/html-css/tree/master/desafios/d012>

Agora faça o download do arquivo pacote-d012.zip e descompacte os arquivos, colocando todo o conteúdo na sua pasta de desafios, dentro da sub-pasta d012.

Dentro desse arquivo compactado, você vai encontrar:

- Duas imagens que aplicaremos ao site, adicionando o efeito parallax a elas
- O texto original, criado por Milton Duarte e disponível no [Recanto das Letras](#)

## O projeto pronto

Para ver o projeto desse capítulo funcionando completamente, basta acessar o endereço <https://professorguanabara.github.io/projeto-cordel/> no seu navegador, mas não vale ficar olhando e copiando o código, pois esse é mais um desafio.

Acesse, role por todo o conteúdo, aumente e diminua o tamanho da janela (principalmente na largura) e veja como ele vai se comportar.

## Organizando o conteúdo em seções

Agora vamos começar a organizar o conteúdo em seções, que serão formatadas para intercalar áreas brancas com áreas de imagens. Você pode usar qualquer técnica, mas a que eu achei mais coerente semanticamente foi dividir tudo em `<section>` com um ou dois parágrafos e usando quebras de linha para organizar as estrofes.

```
<section>
  <p>
    Estou ficando cansado<br>
    Da tal tecnologia<br>
    Só se fala por e-mail<br>
    Mensagem curta e fria<br>
    Twitter e Facebook<br>
    Antes que eu caduque<br>
    Vou dizer tudo em poesia.
  </p>
</section>
```

## Como criar um texto que aumenta ou diminui dinamicamente?

Nesse exercício também usamos um recurso que faz com que o texto aumente ou diminua de acordo com o tamanho (mais especificamente a largura) da janela do navegador.

A dica aqui é usar as medidas relativas das CSS, como vh e vw. Basicamente vh significa **viewport height** e vw significa **viewport width**.

Mas que *diabos* é **viewport**?, você pode perguntar. Bem, de forma resumida a *viewport* é a área visível de uma janela. No início de um documento HTML, dentro da área <head>, nós fizemos a configuração básica do tamanho de uma viewport usando uma tag <meta> desde o nosso primeiro exercício.



```
1  <!DOCTYPE html>
2  <html lang="pt-br">
3  <head>
4      <meta charset="UTF-8">
5      <meta name="viewport" content="width=device-width, initial-scale=1.0">
```

Na **linha 5** do nosso código base, definimos que a viewport terá a largura máxima baseada na largura do dispositivo e que vai usar uma escala (zoom) inicial da tela para o valor padrão 1.0 (zoom de 100%).

Sendo assim, vamos considerar uma tela de celular popular como o **Samsung Galaxy S9**, que tem uma viewport de 360x740 pixels ou um **iPhone X** com 375x812 pixels. Isso significa que - com o celular na posição vertical - a largura da tela é de 300 e poucos pixels disponíveis para exibir o nosso site.

Para usar uma medida fixa do tamanho de um título, podemos declarar:

```
h1 {
    font-family: Verdana, Geneva, Tahoma, sans-serif;
    font-size: 5em;
}
```

O que vai fazer com que a fonte fique **5 vezes** maior que o seu tamanho base (aproximadamente 16px, dependendo do navegador). E esse tamanho será mantido, tanto para telas pequenas quanto para telas grandes.

Mas e se quisermos adaptar o tamanho da fonte ao tamanho da tela? Aí podemos mudar um pouco a declaração acima e usar:

```
h1 {
    font-family: Verdana, Geneva, Tahoma, sans-serif;
    font-size: 10vw;
}
```

Isso vai significar que o tamanho da fonte será adaptado para que ocupe 10% da largura da viewport, ou seja, em uma tela do Galaxy, com 360px de largura, o tamanho da fonte será 36px. Mas basta deitar o celular para que a largura da tela vá para 740px, fazendo com que a fonte seja mudada para tamanho 74px, o que significa 10% da nova dimensão.

Esse recurso fica ainda mais visível quando estamos em um computador e modificamos o tamanho da janela do navegador. Faça seus testes usando medidas de fontes configuradas com `em` e depois mude para `vw`.

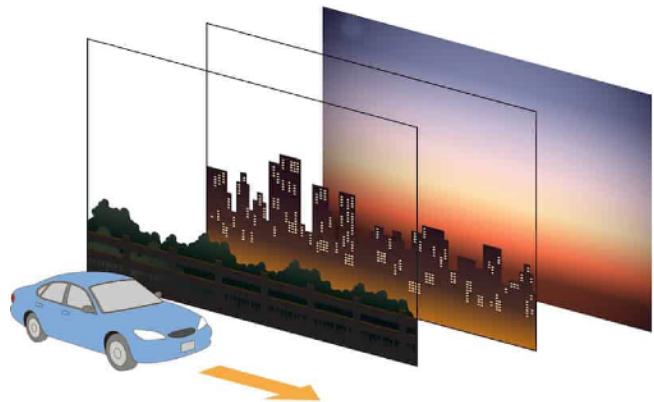


**USE COM MODERAÇÃO!** Esse recurso de tamanho dinâmico de fontes deve ser usado apenas em pontos muito específicos do nosso site. Não exagere e aplique isso em todos os textos, pois seu site vai acabar difícil de adaptar a telas. Se quiser um recurso melhor, estudaremos as Media Queries mais para frente.

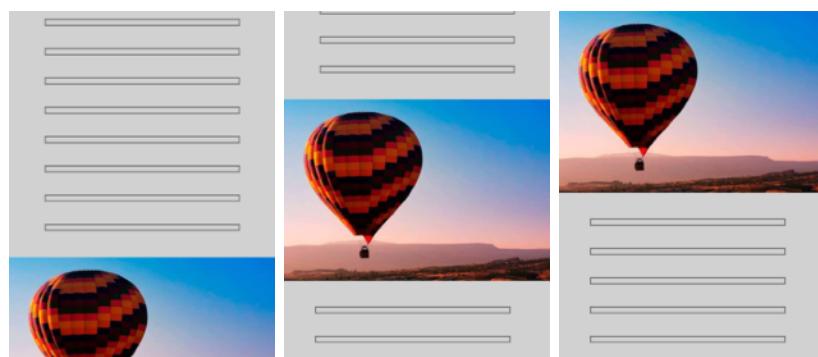
## O que é o efeito parallax?

Em vários pontos durante esse capítulo, eu citei esse efeito e provavelmente você percebeu um comportamento diferente nas imagens do site do projeto que te apresentei anteriormente. Elas se movem de um jeito diferente. Vou te explicar esse fundamento com uma situação do dia-a-dia.

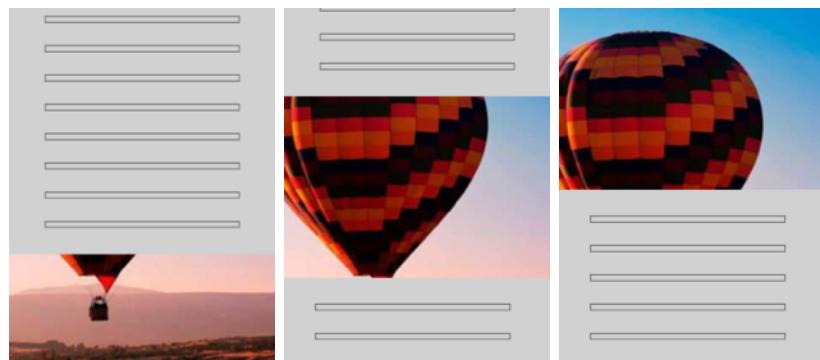
Você já deve ter percebido que quando estamos em movimento (em um carro, ônibus ou trem) e olhamos para a paisagem na janela lateral, aquilo que está mais perto de você parece se mover muito mais rápido e o que está mais longe tem um deslocamento muito mais lento? Pois isso é o que chamamos de **efeito parallax**.



Podemos simular esse mesmo princípio em sites criando diferenças de posicionamentos na rolagem do conteúdo. Em um site comum, imagine que temos um bloco de texto seguido de uma imagem, seguida novamente por um texto. Se não fizermos nenhuma configuração adicional, a rolagem da tela acontecerá da seguinte maneira:



Note que a imagem do balão vai subindo juntamente com os blocos de texto. Agora, quando aplicamos o **efeito parallax**, a rolagem fica um pouco diferente:



Agora o conteúdo vai sendo rolado pela tela, enquanto a imagem vai sendo revelada em pedaços diferentes, dando a impressão de que o bloco com texto está mais perto e a imagem está mais distante.

Talvez o impacto visual gerado por esse efeito não seja 100% percebido enquanto eu tento te explicá-lo em forma de um texto em uma página, mas com certeza você percebeu quando abriu o site do projeto. E se você ainda não abriu, aí vai outra oportunidade. Acesse o link a seguir, ou abra o app de câmera e aponte seu celular para o código QR:

<https://professorguanabara.github.io/projeto-cordel/>



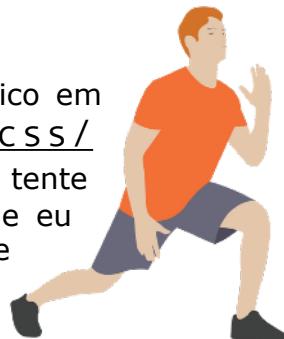
Para obter esse efeito, vamos dar um id à `<section>` que vai conter a imagem e realizar as seguintes configurações:

```
section#img01 {  
    background-image: url('imagens/background001.jpg');  
    background-repeat: no-repeat;  
    background-position: right center;  
    background-size: cover;  
    background-attachment: fixed;  
}
```

Todas as declarações acima foram vistas no capítulo anterior, quando falamos sobre aplicação de imagens em *background*.

# Hora de exercitar

Chegou a hora de acessar o endereço do nosso repositório público em <https://gustavoguanabara.github.io/html-css/desafios/> e executar o **desafio 010** no seu computador. Agora tente atingir esse mesmo resultado em casa, sem copiar o código que eu criei. Nesse momento, a prática é algo que você mais precisa. Se por acaso ficar difícil, pode acessar o repositório público de HTML e CSS e dar uma olhada nos comandos, mas **EVITE COPIAR**.



## Eu já falei sobre isso no YouTube?

Eu sei que às vezes as pessoas gostam mais de assistir vídeos do que ler livros, e é por isso que eu lanço há anos materiais no canal Curso em Vídeo no YouTube. O link que vou compartilhar contigo tem o conteúdo explicado como você leu aqui, só que de forma mais ilustrada. Reserve um tempo dos seus estudos para assistir esse vídeo todo.



Curso em Vídeo: [https://www.youtube.com/playlist?list=PLHz\\_AreHm4d1AnJ\\_jJtV29RFxnPHDuk9o](https://www.youtube.com/playlist?list=PLHz_AreHm4d1AnJ_jJtV29RFxnPHDuk9o)

# DevWeb

## Capítulo 21

### Conteúdos em Tabelas

Ao construir os conteúdos que vão compor o seu site, provavelmente você vai se deparar com a necessidade de criar uma ou mais tabelas. A HTML tem uma série de tags para a organização tabular de dados e precisamos dominá-las. Este capítulo vai te mostrar muitas peculiaridades das tabelas e suas várias possibilidades de personalização.



Você tem todo o direito de usar esse material para seu próprio aprendizado. Professores também podem ter acesso

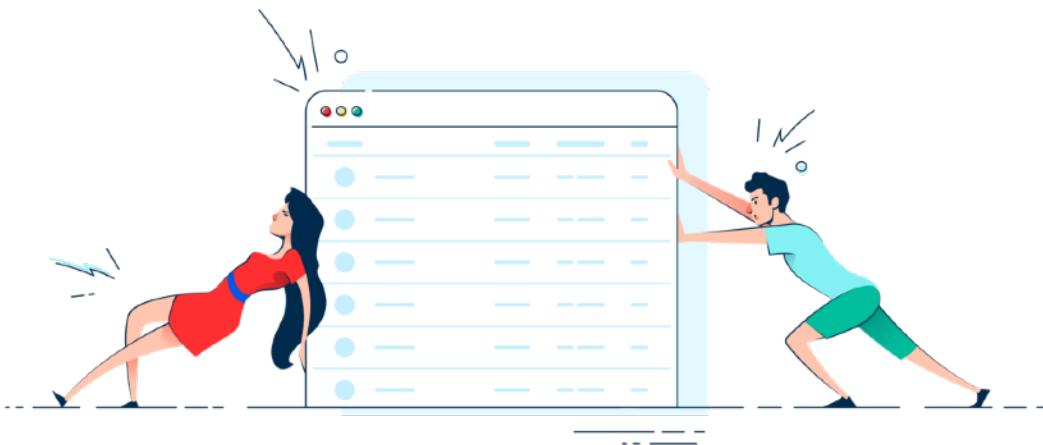
a todo o conteúdo e usá-lo com seus alunos. Porém todos os que usarem esse material - seja para qual for a finalidade - deverão manter a referência ao material original, criado pelo **Prof.**

**Gustavo Guanabara** e disponível no endereço do seu repositório público <https://github.com/gustavoguanabara/>. Este material não poderá ser utilizado em nenhuma hipótese para ser replicada - integral ou parcialmente - por autores/editoras para criar livros ou apostilas, com finalidade de obter ganho financeiro com ele.



# Tableless: tabelas são para apresentar conteúdos, não para construir estruturas de sites

Antes de começar a estudar as tabelas, vamos retirar um assunto da frente. Houve uma época em que os sites eram desenvolvidos como se fossem tabelas. Foi um tempo sombrio e que ficou no passado!



Com a evolução das CSS e com o surgimento da HTML na sua versão 5, os sites podem ser criados de forma mais organizada e dinâmica e não dependem mais das tabelas para existir, um movimento do design que ficou conhecido como *tableless layout* (traçado sem tabelas).

Mas não vamos demonizar as tabelas! Elas são boas e importantes, só não precisam ser usadas para criar a estrutura do seu site. Deixe as tabelas para o seu objetivo principal: apresentar dados de forma tabular. Só isso!

## Nossa primeira tabela

Vamos começar com uma tabela simples para podermos entender a estrutura básica dos dados tabulados:

População das Unidades Federativas	
Cidade	População
São Paulo	45.919.049
Minas Gerais	21.168.791
Rio de Janeiro	17.264.943
Bahia	14.873.064
Paraná	11.433.957
Total	110.659.804



**FONTE DOS DADOS:** A tabela anterior foi construída com base nos dados disponíveis em Junho de 2020 no site da Wikipedia, no link a seguir.

[https://pt.wikipedia.org/wiki/Lista\\_de\\_unidades\\_federativas\\_do\\_Brasil\\_por\\_população](https://pt.wikipedia.org/wiki/Lista_de_unidades_federativas_do_Brasil_por_população)

Vamos começar analisando a anatomia básica da tabela, demarcando algumas áreas importantes:

População das Unidades Federativas	
Cidade	População
São Paulo	45.919.049
Minas Gerais	21.168.791
Rio de Janeiro	17.264.943
Bahia	14.873.064
Paraná	11.433.957
Total	110.659.804

Em primeiro lugar, temos o **título** (*caption*) da tabela, marcado com o **número 1**. Em seguida, temos o **cabeçalho** (*thead*) da tabela com o **número 2**, que serve para identificar cada coluna de dados. Já o **corpo** (*tbody*) da tabela apresenta todos os dados de forma tabular na área demarcada com o **número 3**. Por fim, temos uma área final que totaliza todos os dados, que é o **rodapé** (*tfoot*) da tabela, delimitado pela área de **número 4**.

O código base para criar a tabela acima será:

```
<table>
  <caption>
    <!-- título da tabela -->
  </caption>
  <thead>
    <!-- cabeçalho da tabela -->
  </thead>
  <tbody>
    <!-- corpo da tabela -->
  </tbody>
  <tfoot>
    <!-- rodapé da tabela -->
  </tfoot>
</table>
```

Compare o código HTML com a estrutura da imagem com as quatro áreas delimitadas e tudo ficará bem claro. Esse será o "esqueleto" da nossa tabela que vai receber os dados.



**SUA TABELA É GRANDE OU PEQUENA?** Ao criar tabelas pequenas, não é tão necessário delimitar as áreas de `<thead>`, `<tbody>` e `<tfoot>`. Essas tags semânticas são mais usadas para tabelas com muito conteúdo, segundo a W3C. Mas eu sempre aconselho a usá-las para deixar claro a estrutura da sua tabela.

Agora vamos nos focar na estrutura das linhas de cada área para entender como demarcar a hierarquia de células:

Cidade	População
São Paulo	45.919.049
Minas Gerais	21.168.791

Em primeiro lugar, cada área é dividida em **linhas** (`table rows - tr`), que é a área marcada com o **número 5** e ocupa a largura total da tabela. Dentro das linhas, temos **células de dados** (`table data - td`) ou **células de cabeçalho** (`table header - th`). Sendo assim, para fazer o código relativo ao trecho que marcamos na imagem acima, vamos escrever:

```
<tbody> <!-- corpo da tabela -->
  <tr> <!-- primeira linha -->
    | <td>São Paulo</td> <!-- primeira célula da primeira linha -->
    | <td>45.919.049</td> <!-- segunda célula da primeira linha -->
  </tr>
  <tr> <!-- segunda linha-->
    | <td>Minas Gerais</td> <!-- primeira célula da segunda linha -->
    | <td>21.168.791</td> <!-- segunda célula da segunda linha -->
  </tr>
  ...
</tbody>
```



**QUANDO USAR `<td>` OU `<th>`?** Uma dúvida muito comum é quando usar cada uma dessas tags. De forma resumida, vamos usar o `<td>` quando a célula contiver um dado (**São Paulo**, por exemplo) e usaremos o `<th>` quando a célula contiver um identificador de dados (**Cidade**, por exemplo). Porém, ambos estarão dentro de linhas `<tr>`.

# Ao usar o <th>, defina o escopo

Agora que você já sabe que a tag `<th>` é usada para definir títulos de colunas ou linhas, agora é importante saber o que é escopo desse título. Ainda considerando a tabela que estamos construindo, vamos nos focar nas células marcadas na imagem a seguir. Elas são as nossas células de título, marcadas com `<th>`.

População das Unidades Federativas	
Cidade	População
São Paulo	45.919.049
Minas Gerais	21.168.791
Rio de Janeiro	17.264.943
Bahia	14.873.064
Paraná	11.433.957
<b>Total</b>	<b>110.659.804</b>

Perceba também que as células de “Cidade” e “População”, possuem seus dados abaixo delas (o que chamamos de **escopo de coluna**). Já a célula “Total” tem seu dado ao lado dela (o que chamamos de **escopo de linha**). Sendo assim, vamos fazer as declarações pontuais em cada caso:

```
<tr>
  <th scope="col">Cidade</th>
  <th scope="col">População</th>
</tr>
...
<tr>
  <th scope="row">Total</th>
  <td>110.659.804</td>
</tr>
```

Existem ainda os escopos de `colgroup` e de `rowgroup`, mas só vamos falar sobre essas configurações nos vídeos do curso, pois precisamos antes falar de mesclagem de células com `rowspan` e `colspan`.

# Minha tabela ficou feia, e agora?

Antes de falar em aplicar folhas de estilo a tabelas, precisamos entender que a hierarquia dos componentes de uma tabela é algo **primordial!** Qualquer erro na hierarquia, qualquer tag no lugar errado, vai causar a desorganização dos dados. Dessa maneira, se por acaso os dados não estiverem aparecendo na organização correta, é sinal de que existe um erro na estrutura da tabela.

Se a sua tabela estiver sendo apresentada na tela de forma organizada, mas sem bordas, sem cores e sem graça, chegou a hora de criar as configurações de CSS. A primeira configuração será configurar tamanhos e bordas:

```
table {  
    width: 600px;  
    border: 1px solid black;  
    border-collapse: separate;  
}  
  
td, th {  
    border: 1px solid black;  
}
```

O resultado será uma tabela com as células delimitadas por bordas finas. Note que existe um pequeno espaço entre as células, que talvez incomode um pouco. Essa distância pode ser personalizada através da propriedade border-spacing.

População das Unidades Federativas	
Cidade	População
São Paulo	45.919.049
Minas Gerais	21.168.791
Rio de Janeiro	17.264.943
Bahia	14.873.064
Paraná	11.433.957
Total	110.659.804

Para isso, mude o parâmetro border-collapse do valor separate para o valor collapse. A tabela vai mudar um pouco:

População das Unidades Federativas	
Cidade	População
São Paulo	45.919.049
Minas Gerais	21.168.791
Rio de Janeiro	17.264.943
Bahia	14.873.064
Paraná	11.433.957
Total	110.659.804

Agora vamos adicionar um pequeno padding (8px) às configurações de td e th que fizemos anteriormente e adicionar mais alguns seletores bem simples e intuitivos:

```
caption {  
    font-weight: bolder;  
    font-size: 1.5em;  
    text-align: center;  
    background-color: #rgb(236, 236, 236);  
    padding: 5px;  
}
```

```

thead {
    background-color: #636363;
    color: white;
}

tfoot {
    background-color: #636363;
    color: white;
    font-weight: bolder;
}

```

Todas as configurações acima já foram explicadas nos capítulos anteriores. Se surgir alguma dúvida, consulte o material impresso ou assista os vídeos relativos a cada capítulo. Como eu sempre digo, não tente fazer tudo na correria ou o resultado do seu aprendizado pode ser bem frustrante.

O resultado visual do código CSS acima aplicado à nossa tabela será:

População das Unidades Federativas	
Cidade	População
São Paulo	45.919.049
Minas Gerais	21.168.791
Rio de Janeiro	17.264.943
Bahia	14.873.064
Paraná	11.433.957
Total	110.659.804

## Alinhamento do conteúdo da célula

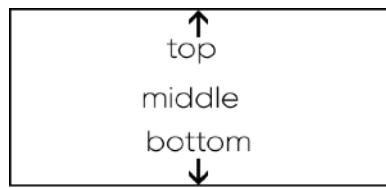
Alinhar o conteúdo de uma célula é muito simples e pode ser realizado em duas dimensões: a **horizontal** e a **vertical**.

O **alinhamento horizontal** é realizado exatamente da mesma maneira de uma div comum ou qualquer outro elemento exibido como um bloco: usando a propriedade `text-align`. Os valores aceitos para o alinhamento horizontal são:



Existe também o valor `justify`, que vai alinhar o conteúdo primordialmente pelo lado esquerdo, mas se o texto for grande, vai alinhar também à direita.

Já o **alinhamento vertical** é realizado pela propriedade `vertical-align`, que aceita os seguintes valores:



## Como criar o efeito zebrado?

Nunca ouviu falar no "efeito zebrado" em tabelas? Pois saiba que o conceito é muito simples e consiste em intercalar as cores das células pares e ímpares, criando um efeito "listrado" (tá aí a referência a uma zebra) que facilita na leitura dos dados em tabelas muito grandes e com muito conteúdo.



Para obter esse resultado na tabela que estamos criando, vamos acrescentar os seguintes seletores ao estilo:

```
tbody > tr:nth-child(even) {  
    background-color: □#dddddd;  
}  
  
```

```
tbody > tr:nth-child(odd) {  
    background-color: □#ffffff;  
}
```

A pseudo-classe `nth-child()` vai permitir selecionar as linhas **pares** (`even`) ou as linhas **ímpares** (`odd`), configurando uma cor para cada uma delas. No exemplo acima, inclusive, se o fundo da página já está branco, o segundo seletor (para linhas ímpares) torna-se desnecessário.

## Mudando o alcance das células

É possível modificar o alcance das células para obter resultados ainda mais organizados. Isso é feito através das propriedades HTML nas células: `colspan` e `rowspan`. Dê uma boa olhada na tabela a seguir:

Grupo	Nomes	Filmes Favoritos		
Mulheres	Ana Maria Santos	Alien	Rambo	Vingadores
	Beatriz Souza	Hulk	Inception	Batman
	Cláudia Melo	Oblivion	Matrix	Big Hero
Homens	Bruno Mendonça	Intocáveis	Amnésia	Gladiador
	Daniel Lourenço	Wall-E	Oldboy	Dangal
	Fabiano Mota	Star Wars 5	Taxi Driver	Toy Story

Note que, no cabeçalho da tabela, temos uma célula para "Filmes Favoritos" que está ocupando o espaço de três colunas. Enquanto isso, na lateral esquerda da tabela, as células com as palavras "Mulheres" e "Homens", que estão ocupando três linhas cada. Para efetuar essas mesclagens, basta usar as propriedades colspan e rowspan, respectivamente.

```
<table>
  <thead>
    <tr>
      <th>Grupo</th>
      <th>Nomes</th>
      <th colspan="3">Filmes Favoritos</th>
    </tr>
  </thead>
  <tbody>
    <tr>
      <td rowspan="3">Mulheres</td>
      <td>Ana Maria Santos</td>
      <td>Alien</td>
      <td>Rambo</td>
      <td>Vingadores</td>
    </tr>
    <tr>
      <td>Beatriz Souza</td>
      <td>Hulk</td>
      <td>Inception</td>
      <td>Batman</td>
    </tr>
  </tbody>

```

Para treinar um pouco mais, crie aí no seu ambiente de testes as seguintes tabelas: a primeira contendo números e com algumas mesclagens pontuais.

1	2		4
5	6	3	7
8		9	
10	11		12
	13	14	15

A segunda tabela vai ser um pouco diferente, vai conter letras e terá as seguintes mesclagens:

A			B
C	D	E	F
	G	H	I
	J	L	M
	N		

Se você conseguiu fazer as tabelas acima, vamos ao desafio final e reproduza a seguinte tabela:

Área	Disciplinas	Notas		Média
		Nota 1	Nota 2	
Exatas	Matemática	0.0	0.0	0.0
	Física	0.0	0.0	0.0
	Química	0.0	0.0	0.0
	Biologia	0.0	0.0	0.0
Média de Exatas				0.0
Humanas	História	0.0	0.0	0.0
	Geografia	0.0	0.0	0.0
Média de Humanas				0.0



**PRATIQUE MUITO!** Os desafios acima usam algumas configurações muito importantes. Saber montar tabelas assim é essencial para provar a si mesmo que aprendeu direito! Não pule os desafios acima, pratique cada um deles em seu computador.

## Personalização de colunas

Podemos personalizar uma linha inteira ou um conjunto de linhas de uma tabela criando seletores voltados para os `<tr>` da nossa tabela. Mas como podemos criar configurações específicas para as colunas?

Antes de mais nada, vamos criar uma tabela simples, com 4 linhas e 5 colunas:

```

<table>
  <tr>
    <td>1A</td>
    <td>1B</td>
    <td>1C</td>
    <td>1D</td>
    <td>1E</td>
  </tr>
  <tr>
    <td>2A</td>
    <td>2B</td>
    <td>2C</td>
    <td>2D</td>
    <td>2E</td>
  </tr>
  <tr>
    <td>3A</td>
    <td>3B</td>
    <td>3C</td>
    <td>3D</td>
    <td>3E</td>
  </tr>
  <tr>
    <td>4A</td>
    <td>4B</td>
    <td>4C</td>
    <td>4D</td>
    <td>4E</td>
  </tr>
</table>

```

O resultado visual dessa tabela, com algumas bordas e preenchimentos aplicados no estilo, será:

1A	1B	1C	1D	1E
2A	2B	2C	2D	2E
3A	3B	3C	3D	3E
4A	4B	4C	4D	4E

Note que temos 5 colunas, representadas pelos itens com as letras A, B, C, D e E respectivamente. Podemos criar um agrupamento de colunas, definindo um `<colgroup>` dentro da tabela, logo abaixo da tag `<table>`:

```

<colgroup>
  <col class="c1">
  <col class="c2" span="2">
  <col class="c3">
  <col class="c1">
</colgroup>

```

Isso vai atribuir uma classe a cada coluna (ou conjunto de colunas, quando usamos `span`). Ao criar uma configuração de cor para cada uma dessas três classes (`c1`, `c2` e `c3`), podemos criar um efeito de configuração em grupos de colunas:

```

.col.c1 {
    background-color: #E3C386;
}

.col.c2 {
    background-color: #DEA12F;
}

.col.c3 {
    background-color: #AC7C24;
}

```

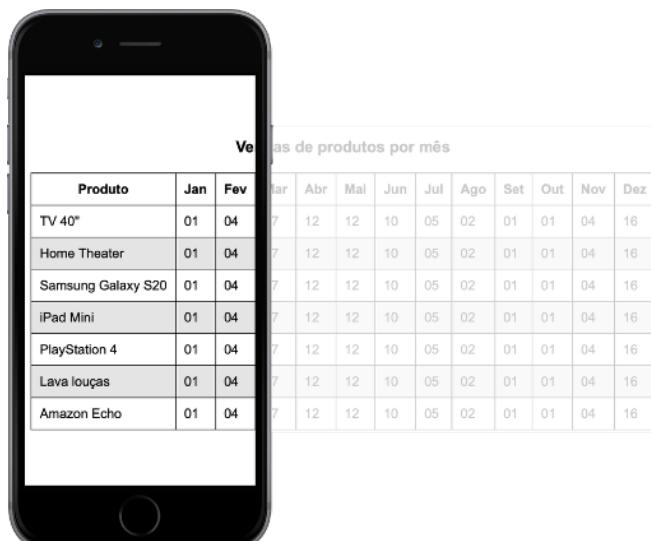
O resultado visual disso será:

1A	1B	1C	1D	1E
2A	2B	2C	2D	2E
3A	3B	3C	3D	3E
4A	4B	4C	4D	4E

Note que usar o `span` no HTML fez com que a segunda e a terceira colunas fizessem parte de um mesmo conjunto de colunas que receberam a mesma formatação de cor. Já a primeira e a última coluna tiveram a mesma cor de fundo aplicada, já que as duas `<col>` possuem a mesma classe configurada.

## Tabelas largas e responsivas

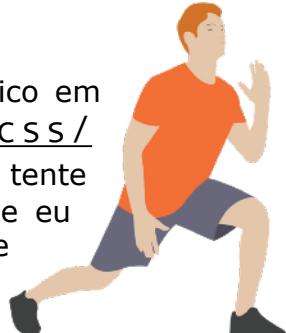
O que será que acontece quando carregamos uma página que contenha uma tabela que tem muitas colunas e que esse conteúdo não caiba na largura da tela? A imagem a seguir mostra exatamente essa situação:



Para resolver essa limitação, principalmente para telas de celular, e garantir a visualização de todos os dados, uma das soluções mais simples é colocar a tabela dentro de uma caixa qualquer (pode ser uma div) e configurar a sua propriedade CSS overflow-x para os valores auto ou scroll. Isso vai fazer com que todo e qualquer conteúdo que "transborde" a largura (eixo x) do tamanho do box, vai ganhar uma barra de rolagem horizontal (por se tratar do eixo x apenas) caso seja necessário.

## Hora de exercitar

Chegou a hora de acessar o endereço do nosso repositório público em <https://gustavoguanabara.github.io/html-css/desafios/> e executar o **desafio 013** no seu computador. Agora tente atingir esse mesmo resultado em casa, sem copiar o código que eu criei. Nesse momento, a prática é algo que você mais precisa. Se por acaso ficar difícil, pode acessar o repositório público de HTML e CSS e dar uma olhada nos comandos, mas **EVITE COPIAR**.



## Eu já falei sobre isso no YouTube?

Eu sei que às vezes as pessoas gostam mais de assistir vídeos do que ler livros, e é por isso que eu lanço há anos materiais no canal Curso em Vídeo no YouTube. O link que vou compartilhar contigo tem o conteúdo explicado como você leu aqui, só que de forma mais ilustrada. Reserve um tempo dos seus estudos para assistir esse vídeo todo.



Curso em Vídeo: [https://www.youtube.com/playlist?list=PLHz\\_AreHm4dlAnJ\\_jJtV29RFxnPHDuk9o](https://www.youtube.com/playlist?list=PLHz_AreHm4dlAnJ_jJtV29RFxnPHDuk9o)

# DevWeb

## Capítulo 22

### Inline Frames

Aposto contigo uma bananada que mesmo já conhecendo HTML há anos, provavelmente você não sabia que a tag `<iframe>` significava *inline frame* ou, no bom e velho Português, algo como “quadro em linha”. Nesse capítulo, vamos nos aprofundar um pouco nos *iframes*, que já apareceram aqui no curso durante o capítulo 11, enquanto nós falávamos sobre inserção de vídeos vindos de serviços como YouTube e Vimeo.



Você tem todo o direito de usar esse material para seu próprio aprendizado. Professores também podem ter acesso

a todo o conteúdo e usá-los com seus alunos. Porém todos os que usarem esse material - seja para qual for a finalidade - deverão manter a referência ao material original, criado pelo **Prof.**

**Gustavo Guanabara** e disponível no endereço do seu repositório público <https://github.com/gustavoguanabara/>. Este material não poderá ser utilizado em nenhuma hipótese para ser replicada - integral ou parcialmente - por autores/editoras para criar livros ou apostilas, com finalidade de obter ganho financeiro com ele.



# Entendendo quadros em linha

Antes de mais nada, é bom deixar claro que essa tradução porca que levou um *iframe* a ser chamado de *quadro em linha* na sua versão brazuca não faz parte da referência oficial do W3C. Esse termo traduzido para o idioma tupiniquim veio única e exclusivamente da minha vontade em fazer você lembrar que aquele “i” vem de *inline* e não tem nenhuma relação a produtos da **Apple** como *iPod*, *iPhone* e *iPad* (juro que um aluno já me perguntou se havia alguma ligação).



Um *iframe* é basicamente uma “janela aberta” dentro da nossa página para apresentar o conteúdo de outras páginas. Esse é um recurso bem antigo, presente desde as versões anteriores da HTML, mas que até hoje são utilizados por alguns sites, o que causa muita discussão em relação a segurança. Vamos abordar esse assunto mais pra frente, mas antes vamos aprender a usar esse recurso.



**NÓS JÁ VIMOS ISSO ANTES:** No **capítulo 11** aprendemos como incorporar um vídeo do **YouTube** ou **Vimeo** a uma página. Basta voltar lá e constatar que o código fornecido pelo serviço é uma tag `<iframe>` com vários parâmetros pré-configurados e que “magicamente” apresentava um vídeo dentro do nosso HTML. Pois saiba que podemos usar os quadros para apresentar qualquer tipo de conteúdo que esteja em outro site de forma simples.

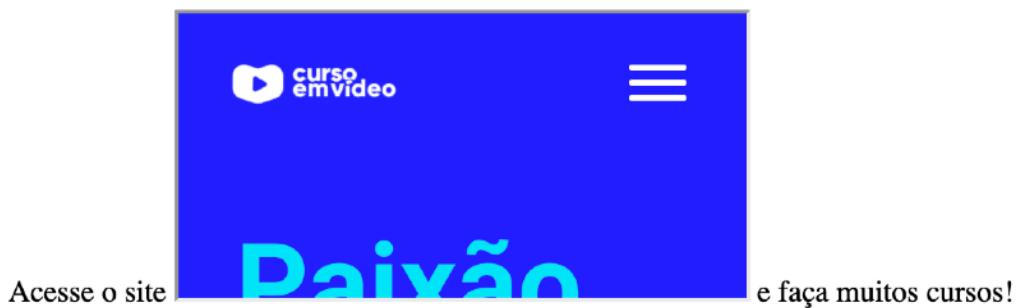
## Como criar um *iframe*?

Vamos começar inserindo um quadro bem simples:

```
<h1>Teste de iframe</h1>
<p>
    Acesse o site
    <iframe src="https://www.cursoemvideo.com"></iframe>
    e faça muitos cursos!
</p>
```

O código acima cria um título e um parágrafo com um *iframe* no meio da frase. Analisando o resultado, percebemos o motivo do uso do “i” antes da palavra “frame” no elemento. Ele cria um bloco *inline* (na mesma linha) que não quebra a frase, mesmo que o elemento tenha uma altura muito maior que o texto. Veja só a imagem a seguir:

# Teste de iframe



O site do **CursoemVideo** foi aberto dentro da nossa página, permitindo inclusive que o visitante faça a rolagem pelo conteúdo.

Podemos ainda configurar detalhes visuais e comportamentais como o tamanho do *frame*, sua borda, a forma de rolagem e até mesmo tratar eventuais incompatibilidades com alguns navegadores:

```
<p>
    Acesse o site
    <iframe src="https://www.cursoemvideo.com"
        frameborder="0" width="500" height="500"
        scrolling="auto">
        [ Seu navegador não suporta iframes ]
    </iframe>
    e faça muitos cursos!
</p>
```

O código acima remove a borda 3D que é criada por padrão (podemos ainda personalizá-la com CSS mais pra frente) e também configura um *frame* quadrado, com 500 pixels de largura (*width*) e de altura (*height*).

A propriedade *scrolling* aceita três valores:

- no - não permite a rolagem da página, mesmo que ela seja maior que o tamanho estabelecido para o *frame*.
- yes - habilita a rolagem do conteúdo, mesmo que seu conteúdo não seja grande o suficiente para preencher o *frame*.
- auto - (padrão) a rolagem é habilitada apenas se o conteúdo for maior que o tamanho do quadro estabelecido.

Por fim, note que acrescentamos uma frase entre `<iframe>` e `</iframe>` para que seja exibida caso o navegador do usuário não suporte o uso desse tipo de recurso. Alguns celulares mais antigos possuem navegadores que bloqueiam a exibição de quadros por questões relacionadas ao tamanho da tela, uso de memória e até segurança.

Talvez seja até interessante adicionar um link `<a>` a essa mensagem de erro para que o usuário possa visitar a página caso seu navegador não seja compatível com *iframe*.

# Páginas locais dentro do iframe

Claro que não precisamos apenas carregar sites externos dentro dos quadros que vão compor nosso site. Também podemos ter páginas locais carregadas no meio do nosso conteúdo.

```
<iframe src="outra-pagina.html" width="600" height="400">
    <p>Seu navegador não é compatível com a exibição desse recurso.</p>
</iframe>
```

É claro que para o elemento acima funcionar, devemos ter um arquivo chamado `outra-pagina.html` dentro da mesma pasta e o conteúdo será exibido em um quadro de dimensões `600x400`, conforme as configurações estabelecidas.

Podemos até ter mais de um `iframe` em uma mesma página, lembrando que por padrão eles serão exibidos no formato `inline-block` (isso pode até ser mudado via CSS). Apenas considere que cada quadro aberto dentro da página é semelhante a uma nova instância do navegador aberta. Sendo assim, sobrecarregar um site com muitos `iframes` pode deixar o dispositivo do seu visitante mais lento, consumindo um pouco mais de memória.



## IFRAMES VÃO DEIXAR O COMPUTADOR MAIS LENTO?

A melhor resposta para essa pergunta é: **DEPENDE**. Acabamos de ver que um `iframe` é como uma nova instância do navegador abrindo em um quadro. De forma grosseira, é como se abrissemos uma nova aba do navegador para ver o conteúdo. Abrir uma nova aba deixa seu computador muito mais lento? Pode ser que sim, pode ser que não. É a mesma linha de raciocínio quando usamos *quadros em linha*.

# Gerando conteúdo dentro do iframe

No lugar de usar o parâmetro `src` para indicar a **origem** (*souce*) do conteúdo de um quadro, podemos usar o parâmetro `srcdoc` para criar um conteúdo simples estaticamente dentro do `iframe`.

```
<iframe srcdoc=<h2>Teste de título</h2><p>Esse documento foi gerado
dinamicamente</p><img src='cursoemvideo.png'>" frameborder="0">
    <p>Seu navegador não é compatível com a exibição desse recurso.</p>
</iframe>
```

A estrutura acima vai criar um quadro de tamanho padrão (`300x150`) e vai colocar um título, um parágrafo e uma imagem dentro dele. Não existe nenhum arquivo adicional ou site externo para carregar o conteúdo.

# Personalizando um quadro via CSS

Já que todo `iframe` é uma caixa em linha (mais informações no [capítulo 16](#)), ele pode ser personalizado usando folhas de estilo sem problema algum. O princípio é exatamente o mesmo e o comportamento também.

```
iframe {  
    width: 500px;  
    height: 500px;  
    border: 5px solid black;  
    border-radius: 15px;  
    box-shadow: 3px 3px 10px #0000006c;  
}
```

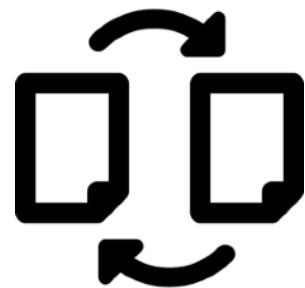


**UMA QUESTÃO DE PRIORIDADES:** O código acima define o tamanho do `iframe` para uma resolução de 500x500 pixels. Se por acaso o código HTML tivesse definido para outro tamanho (800x600, por exemplo), qual configuração fica valendo? Pois saiba que as folhas de estilo **SEMPRE** vão sobrescrever toda e qualquer pré-configuração feita pela HTML. O estilo nesse caso é soberano. Lembre-se disso.

## Direcionando novo conteúdo para um `iframe`

Podemos dar ordens a um quadro para trocar o seu conteúdo através de *links* ou códigos *JavaScript*. O primeiro passo é dar um nome a ele através do parâmetro `name`.

```
<iframe name="quadro" src="pagina01.html">  
    <p>Seu navegador não é compatível  
        com a exibição desse recurso.</p>  
</iframe>
```



No código acima, o quadro terá 300x150 pixels de tamanho e vai exibir o conteúdo do arquivo `pagina01.html` (se for compatível, claro). A diferença aqui é que atribuímos um nome a ele, que agora se chama **quadro**.

A qualquer lugar do documento original (o que contém o `frame`), podemos criar um link da seguinte forma:

```
<a href="pagina02.html" target="quadro">Abrir segundo site</a>
```

O *link* que criamos vai abrir a `pagina02.html` dentro do `iframe` sempre que o usuário clicar sobre ele, substituindo o conteúdo anterior imediatamente.

# Porém, nem tudo são flores...

Quando escolhemos usar *iframes* em nosso conteúdo, devemos estar cientes de que essa opção traz consequências. Muita gente diz que usar códigos de <iframe> é algo ultrapassado e não recomendado. Não compartilho da mesma opinião, apenas quero te deixar ciente do que pode acontecer, e por isso resolvi enumerar alguns tópicos e falar brevemente sobre eles.

## Não confunda frames e iframes

Minha avó sempre dizia: “- *não confunda alhos com bugalhos*” e isso me fez aprender o que são bugalhos. Pesquise aí também se quiser aprender. A aula aqui é de *iframe*!



E sem fugir mais do assunto, muita gente confunde *iframes* com os antigos *frames* da versão 4 da HTML e por isso ficam dizendo que todos são ultrapassados. Antigamente, os sites usavam as tags <frameset> e <frame> para organizar o layout do site. Ainda hoje, alguns sites ainda os usam, mas para criar uma barra fixa superior enquanto mostra outro conteúdo na parte de baixo da tela.

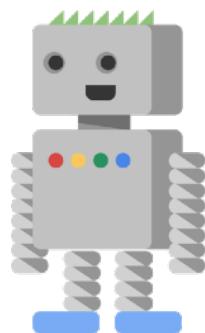
Os *frames* não se tornaram aquelas *tags* ditas **obsoletas**, mas estão em um caminho acelerado para isso. Hoje existem técnicas mais modernas e convenientes para que não seja necessário utilizá-los.

Os *iframes* são mais modernos e muitos sites ainda os utilizam, como veremos mais adiante. Então não cometa o engano de confundir os dois ou colocá-los em um “mesmo saco”.

Porém, como o próprio título dessa sessão sugere, nem tudo são flores e mesmo sendo modernos, optar por usar *iframes* pode trazer algumas consequências indesejadas em algumas situações.

## Problemas com mecanismos de busca

Quando criamos sites, uma das coisas que mais vamos querer é que ele apareça nos resultados de busca do **Google**, certo? Pois saiba que existe um algoritmo que varre o conteúdo dos sites - um robô batizado de *Googlebot* - que analisa todo o conteúdo de um HTML e indexa seu conteúdo.



**UMA OLHADA NAS GUIDELINES:** Ao acessar a documentação oficial do Google, encontramos referências a esse problema de indexação de conteúdo pelo robô. Dê uma breve conferida no link a seguir para mais informações.

Google Search Console Help: <https://support.google.com/webmasters/answer/72746?hl=pt-br>

Segundo o próprio **Google**, podem acontecer problemas com a indexação de conteúdos contidos dentro de *iframes* e talvez não haja uma boa relação entre o que está fora (na página principal) e o que está dentro (na parte interna do *iframe*).

## A usabilidade mandou um abraço

Ao decidir usar *iframes* para apresentar conteúdos em nosso site, saiba que existem alguns incômodos relativos à usabilidade e acessibilidade desse conteúdo.

Para quem é *deficiente visual*, alguns leitores de tela simplesmente se perdem completamente quando o conteúdo está dentro de um quadro desse tipo. E além disso, alguns outros problemas pontuais:



- O botão de **voltar** para a página anterior no navegador fica meio perdido, sem saber se quer voltar no site externo ou dentro do *iframe*.
- Confunde os usuários quando um site que está dentro de um *iframe* tenta abrir uma página em uma nova janela.
- Se o site a ser aberto não for responsivo, provavelmente vai ficar visualmente quebrado ou esquisito dentro de um *iframe* na tela de um celular.

Não estou escrevendo tudo isso pra te desmotivar, mas também acho falta de consideração esconder esses inconvenientes de você. Os *iframes* são ótimos, muito práticos e fáceis de usar, mas podem dar dor de cabeça em alguns navegadores/dispositivos.

## Cuidado com quem você chama pra sua casa



E por fim, mas não menos importante (na verdade a coisa mais importante de todas), a questão da **segurança** é muito discutida em relação ao uso de *iframes*.

Porém, em relação a esse assunto, tenho uma revelação importante a fazer: **em parte, isso pode ser responsabilidade SUA!** Isso porque devemos tomar todo o cuidado na hora de colocar um quadro no nosso site, abrindo um site de terceiros.

Você por acaso deixa aberta a porta da frente da sua casa pra quem quiser entrar e, eventualmente, beber um copo d'água? É uma atitude muito arriscada, não é? Pois esse conselho também serve para o seu site.

Ao usar *iframe* em projetos que vão efetivamente ao ar, devemos tomar o cuidado de dificultar as ações cruzadas e as relações entre o nosso site e aquele apresentado dentro do quadro. Isso pode deixar seu site vulnerável a ataques do tipo *cross-site* e seus visitantes suscetíveis a ataques XSS.

Mas nem tudo está perdido e podemos proteger um pouco mais nosso site usando configurações de *sandbox* e definindo *políticas de referência* mais restritivas. E é sobre isso que falaremos nos próximos tópicos desse capítulo.

# Tornando as coisas mais seguras

## Aprendendo a usar a caixa de areia

Cientes dos problemas que um `iframe` mal configurado pode trazer, a HTML5 implementou o novo recurso de *sandbox*. Uma caixa de areia (*sandbox*) é um ambiente controlado que permite exibir outros sites para seu visitante sem que ele possa “tomar controle” do site principal e aos dados do visitante.



Quando habilitamos a *sandbox* em nosso `iframe`, automaticamente o site que está dentro do quadro perde algumas funcionalidades, dentre elas:

- Não pode mais enviar dados de formulários
- Não pode mais executar scripts
- Desabilita todo tipo de API, janelas modais e *popups*
- Desabilita todo tipo de plugin com `<embed>`, `<object>`, `<applet>`
- Evita que o site dentro do `iframe` assuma a navegação *top level* do navegador
- Bloqueia recursos como autoplay e foco automático em elementos de formulário

Para carregar um site dentro de um quadro *iframe* em ambiente protegido, basta declarar:

```
<iframe src="https://www.cursoemvideo.com" frameborder="0" width="400" height="400" scrolling="yes" sandbox="sandbox"> </iframe>
```

Usar a propriedade `sandbox` com o valor `sandbox` (entre aspas) vai ligar todos os bloqueios citados acima.

Caso queira abrir alguma exceção e desbloquear algum dos recursos pontualmente, poderá usar um ou mais valores da lista a seguir:

- |                              |                                     |
|------------------------------|-------------------------------------|
| • <code>allow-forms</code>   | • <code>allow-popups</code>         |
| • <code>allow-scripts</code> | • <code>allow-same-origin</code>    |
| • <code>allow-modals</code>  | • <code>allow-top-navigation</code> |



**REFERÊNCIA COMPLETA:** Para uma lista completa de todos os valores suportados pela propriedade `sandbox`, acesse o site.

W3Schools: [https://www.w3schools.com/tags/att\\_iframe\\_sandbox.asp](https://www.w3schools.com/tags/att_iframe_sandbox.asp)

## Definindo a política de referência

Outra maneira de limitar a ação de sites que carregarmos em nossos `iframes` é configurando a **política de referência** (*referral policy*). Sem entrar muito em termos

técnicos, normalmente os sites conseguem monitorar nossos rastros de navegação através de um campo no cabeçalho http chamado *Referer Header*. Ele sempre indica onde o visitante estava quando decidiu ir para o seu site. Esse é um dos recursos que as redes sociais como Twitter, Facebook e Instagram e sites como Google e YouTube usam para entender nosso padrão de navegação e mostrar propagandas. Mas não é só pra isso que serve.



Ao configurar a propriedade `referrerpolicy` em um *iframe*, conseguimos definir quantos dados serão compartilhados com o site de destino. A maneira de manter a navegação mais “anônima” é usar o valor `no-referer` para esse atributo.

Existem outros valores, como:

- `no-referer-when-downgrade` (padrão)
- `origin`
- `origin-when-cross-origin`
- `same-origin`
- `strict-origin`
- `strict-origin-when-cross-origin`
- `unsafe-url`



**REFERÊNCIA COMPLETA:** Para uma lista completa de todos os valores suportados pela propriedade `referrerpolicy`, acesse o site.

W3C: <https://www.w3.org/TR/referrer-policy/>

## Outros exemplos do uso de iframes

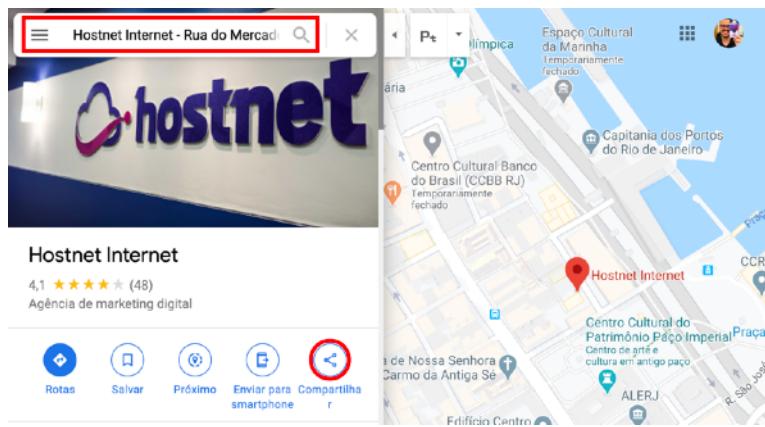
Além do **YouTube** e do **Vimeo**, que já usamos para incorporar vídeos aos nossos sites, também podemos usar outros serviços para acrescentar ainda mais conteúdos aos nossos projetos com HTML.



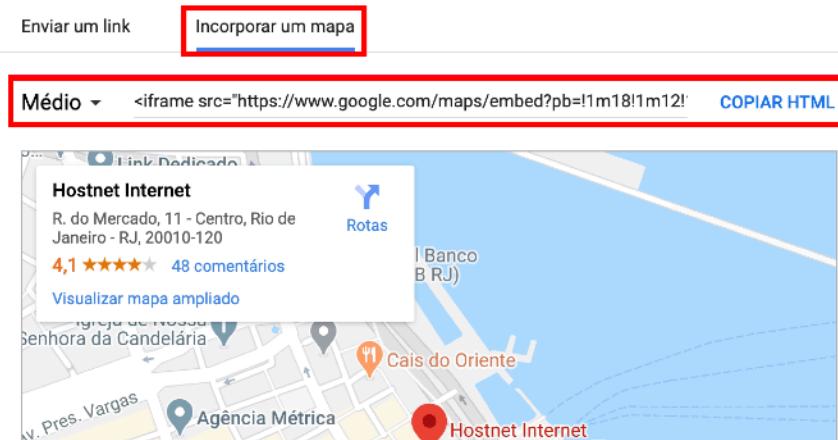
**NADA DE AREIA AQUI, RAPAZINHO:** Nenhum desses recursos, incluindo vídeos de YouTube e Vimeo, funcionam corretamente se habilitarmos o recurso de `sandbox` que vimos anteriormente.

## Google Maps

Podemos também adicionar mapas do Google Maps ao seu site. Para isso, basta acessar o site [maps.google.com](https://maps.google.com) e fazer uma busca de local previamente cadastrado no serviço ou colocar um endereço específico. Logo em seguida, clique sobre o botão redondo com a identificação **Compartilhar** (veja a imagem a seguir)



Ao pressionar o botão de compartilhamento, na tela a seguir, escolha a opção **Incorporar um mapa** e escolha um dos tamanhos disponíveis. Para finalizar, clique sobre o botão **COPIAR HTML** e o código do iframe será colocado na sua área de transferência. Agora vá até seu código HTML no **VSCode** e pressione **Ctrl+V** para colar o elemento copiado.



## Google Docs

A suite de documentos do **Google** também permite incorporar conteúdos diretamente em sites através de *iframes*. Para isso, acesse sua conta no Google Docs e crie um documento novo (pode ser texto, planilha ou apresentação). Assim que terminar, clique na opção **Arquivo > Publicar na Web** que fica na parte superior esquerda da aplicação (cuidado pra não confundir com o menu do seu navegador).

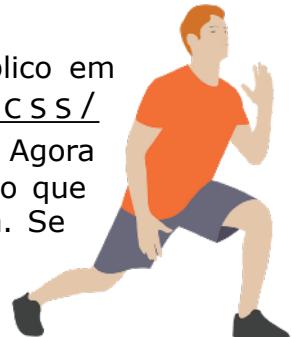
Publique seu conteúdo na Web para que ele fique visível para qualquer pessoa.  
Você pode adicionar um link para o documento ou incorporá-lo. [Saiba mais](#)



A tela que vai aparecer tem a opção de **Incorporar** o documento e - dependendo do tipo de documento - você fará algumas configurações básicas e pode clicar no botão **Publicar**. O código do *iframe* será disponibilizado e você já pode colocá-lo no seu HTML.

## Hora de exercitar

Chegou a hora de acessar o endereço do nosso repositório público em <https://gustavoguanabara.github.io/html-css/desafios/> e executar os **exercícios 024** no seu computador. Agora tente atingir esse mesmo resultado em casa, sem copiar o código que eu criei. Nesse momento, a prática é algo que você mais precisa. Se por acaso ficar difícil, pode acessar o repositório público de HTML e CSS e dar uma olhada nos comandos, mas **EVITE COPIAR**.



## Tenho desafios pra você!

Lá no repositório, além do material em PDF e dos códigos dos exercícios 100% disponíveis, também disponibilizamos alguns **desafios** que devem ser resolvidos. Esses desafios não incluem o código original e você deve tentar chegar à resposta sem copiar nenhum código.

Com todo o conteúdo que vimos até essa aula, você já pode resolver o **desafio d013**. Acesse o repositório público, abra a área do curso de HTML+CSS e clique no link de acesso aos desafios. Manda ver! Só não fica pedindo a resposta! Você consegue resolver isso sozinho(a)!



Repositório em: <https://gustavoguanabara.github.io>

## Eu já falei sobre isso no YouTube?

Eu sei que às vezes as pessoas gostam mais de assistir vídeos do que ler livros, e é por isso que eu lanço há anos materiais no canal Curso em Vídeo no YouTube. O link que vou compartilhar contigo tem o conteúdo explicado como você leu aqui, só que de forma mais ilustrada. Reserve um tempo dos seus estudos para assistir esse vídeo todo.



Curso em Vídeo: [https://www.youtube.com/playlist?list=PLHz\\_AreHm4dlAnJ\\_jJtV29RFxnPHDuk9o](https://www.youtube.com/playlist?list=PLHz_AreHm4dlAnJ_jJtV29RFxnPHDuk9o)

# DevWeb

## Capítulo 24

# Formulários

Estamos na reta final do nosso aprendizado básico com HTML5 e agora chegou a hora de aprender a criar formulários, pois eles são muito usados em sites. Toda tela de login, toda página de cadastro, todo campo de busca vai precisar obrigatoriamente de um formulário HTML para começar. E o que vamos aprender aqui é mais do que apenas criar formulários que funcionam visualmente. Acompanhe comigo em detalhes a partir de agora!



Você tem todo o direito de usar esse material para seu próprio aprendizado. Professores também podem ter acesso a todo o conteúdo e usá-lo com seus alunos. Porém todos os que usarem esse material - seja para qual for a finalidade - deverão manter a referência ao material original, criado pelo **Prof. Gustavo Guanabara** e disponível no endereço do seu repositório público <https://github.com/gustavoguanabara/>. Este material não poderá ser utilizado em nenhuma hipótese para ser replicada - integral ou parcialmente - por autores/editoras para criar livros ou apostilas, com finalidade de obter ganho financeiro com ele.



# Formulários são essenciais

Vamos começar com uma curiosidade (onde também vou vergonhosamente reconhecer que estou ficando velho) e lembrar algo que talvez nem faça parte da sua memória. Provavelmente você ou seus familiares fazem a **Declaração de Imposto de Renda** no início de cada ano usando o computador ou celular. Pois saiba que antigamente, antes dos computadores pessoais se tornarem algo popular, essa declaração era feita em papel, preenchendo um formulário de papel gigantesco que era entregue fisicamente.



De lá pra cá, as coisas evoluíram e hoje é possível usar até mesmo o seu celular para se manter em dia com a Receita Federal. O formulário gigantesco ainda está lá, mas pode ser preenchido de forma digital e muito mais organizada e sem dores de cabeça (além daquelas naturais, no momento de pagar os impostos).



Eu já deveria ter ensinado formulários desde as primeiras aulas de HTML5, mas uma avalanche de *tags* e *propriedades* acabou me impedindo de fazer isso mais cedo, mas você pode ter certeza que ter deixado esse conceito por último (até depois dos *iframes*) não é um desmerecimento ou algum tipo de “castigo”. Na verdade, tudo aquilo que você aprendeu até hoje te trouxe até aqui e os formulários são a cervejinha desse bolo.

Atualmente, encontramos formulários em várias ocasiões, enquanto navegamos em sites:

- Toda tela de login, onde digitamos usuário e senha (e às vezes outras coisas)
- Aquelas janelas onde informamos nosso e-mail para cadastrar em *newsletter* e muitas vezes nos arrependeremos, pois começamos a receber muito SPAM.
- Toda página de cadastro, onde informamos dados pessoais como nome, e-mail, endereço, etc.
- Aqueles campos de busca que encontramos geralmente no cabeçalho do site e usamos para encontrar conteúdos perdidos em meio de tantos artigos.

# Um formulário simples não pode ser tão simples

Quando aprendemos a criar um formulário, é comum encontrar exemplos como no código a seguir:

```
<form>
  Nome:
  <input type="text" name="fnome">
  <input type="submit" value="Enviar">
</form>
```

A primeira e a última linhas vão delimitar o formulário. Tudo aquilo que estiver dentro desse par de *tags* será considerado como conteúdo dos campos. No exemplo acima, temos apenas um campo que vai aceitar um texto relativo ao nome da pessoa e um botão de submissão, que vai permitir enviar o formulário. O resultado do código acima será:

Nome:  Enviar

E aí você pode pensar: "*-Pronto! Está funcionando!*". Infelizmente não! Precisamos fazer algumas configurações adicionais para que esse formulário funcione de maneira mais semântica e correta.

- Em primeiro lugar, vamos adicionar dois atributos importantes à tag `<form>`: o `method` e o `action`. O primeiro vai indicar o método de envio dos dados e o segundo vai dizer para onde os dados serão enviados.
- Não existe nenhuma ligação semântica entre a caixa de texto e seu identificador. Para isso, vamos adicionar um `<label>` para que possa ser feito o relacionamento entre o campo e seu identificador.
- Para que o `<label>` possa funcionar corretamente, a caixa de texto deverá ter um `id`.

Depois de fazer as três alterações indicadas acima, teremos o novo código:

```
<form method="GET" action="cadastro.php">
  <label for="camponome">Nome:</label>
  <input type="text" name="fnome" id="camponome">
  <input type="submit" value="Enviar">
</form>
```

E o resultado visual será exatamente o mesmo:

Nome:  Enviar

E nesse momento você também deve estar imaginando: "*-Mas antes estava igual, pra que adicionar essas coisas?*". A resposta é simples.

O formulário está **visualmente** igual, mas agora ele está mais **semanticamente** correto e o seu navegador vai conseguir relacionar os campos e enviar os dados da maneira certa para o lugar certo. Na primeira versão, mais simples, o seu navegador seria capaz de captar um dado, mas não saberia para onde enviar.

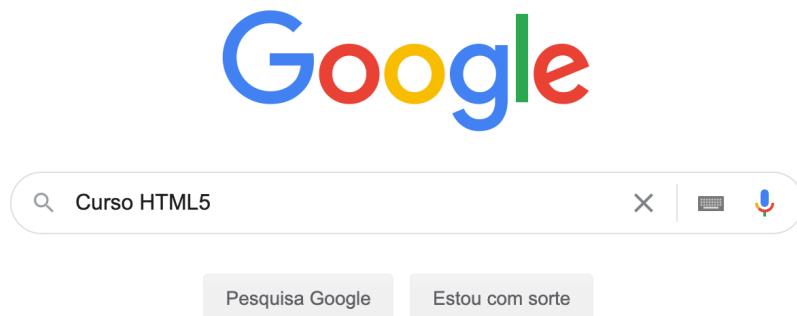


**E NADA DE PREGUIÇA!** Todas as vezes em que você se pegar pensando "mas aí eu estou escrevendo muito...", pense que você está estudando para exercer uma profissão! E um profissional que desenvolve sites realmente deve estar disposto a - eventualmente - ter bastante trabalho. Faça as coisas do jeito certo, não economize no código e seus projetos serão mais relevantes. Fica a dica!

## E quais métodos posso usar?

No exemplo anterior, configuramos o `method` do formulário para o valor `GET`, mas eu não expliquei como funciona esse método e nem quais são as opções disponíveis para esse atributo. Basicamente, existem dois métodos para enviar os dados de um formulário: `GET` e `POST`.

Para exemplificar o método `GET`, acesse agora o site do **Google** e faça uma busca por *Curso HTML5*.



Depois de enviar o formulário (sim, a página principal do Google é um formulário), preste bastante atenção à barra de endereços do navegador e veja a URL que você está acessando:

`https://www.google.com.br/search?q=Curso+HTML5|`

Note que aquilo que você digitou no formulário apareceu na *URL* que está sendo apresentada no seu navegador. Isso é uma característica presente nos formulários com método `GET`, pois ele condensa todos os dados digitados pelo visitante e envia como parte do endereço acessado.

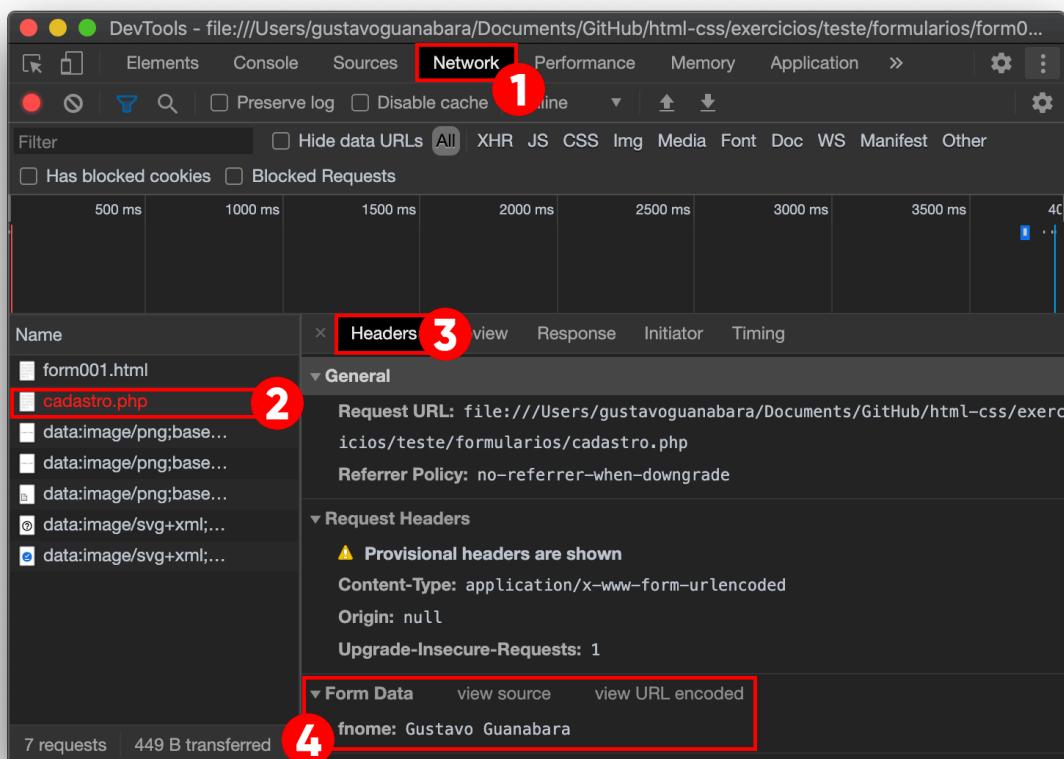
Já o método `POST` vai pegar os dados digitados e enviar diretamente no corpo da requisição HTTP que será feita quando seu visitante clicar para enviar o formulário. Sendo assim, os dados não vão aparecer na *URL*.

# Mas quais são as diferenças entre GET e POST?

Já vi muita gente dizendo que o método POST tinha mais “segurança” que o método GET só porque os dados não aparecem na *URL*. Isso não é bem verdade. Basta mudar o método do seu formulário para POST, abrir o seu formulário no **Google Chrome** e chamar o **DevTools**. Em seguida, digite seu nome no formulário e pressione Enviar.

Nome:

Agora vá até a aba **Network** e escolha a chamada para o arquivo configurado no *action* do formulário, no nosso caso, *cadastro.php*. Clique sobre a aba **Headers** e em seguida procure a seção **Form Data**. Pronto! Lá estão os dados.



E já que esclarecemos que não é uma questão de segurança, vamos analisar em que momentos vamos usar cada um dos métodos.

Em primeiro lugar, podemos usar GET em um formulário contanto que:

- Os dados a serem enviados não sejam sensíveis, como endereços, números de documentos, senhas, etc.
- Tivermos formulários simples e que possam ser compartilhados facilmente através de uma URL. Por exemplo, o Google usa GET no seu formulário, pois você pode enviar um link para qualquer pessoa, já com a ação da busca embutida, como em <http://google.com.br/search?q=Gustavo+Guanabara>

- Os dados a serem enviados nunca ultrapassem 3000 bytes.
- Não contenha envio de arquivos como imagens (i.e. fotos de perfil), arquivos PDF, etc.

Já o método POST é mais recomendado para casos em que:

- Não queremos que os dados apareçam explicitamente na URL do navegador.
- Capturamos dados como senhas, documentos e outro tipo de dado sensível.
- Precisamos enviar muitos dados (acima de 3000 bytes ~ sem limites).
- Precisamos enviar imagens ou outros tipos de arquivos.



**A DISCUSSÃO É AINDA MAIS PROFUNDA**, porém não vou me aprofundar aqui nos assuntos que envolvem as utilizações de métodos de envio de formulários, já que o curso aqui é para conceitos introdutórios. Mas se quiser mais detalhes, consulte esse tópico (em inglês):

Stack Overflow: <https://stackoverflow.com/questions/46585/>

Outra discussão que é bastante levantada nas comunidades é em relação à velocidade ou performance de cada método. De forma resumida, um formulário que usa método GET é ligeiramente mais rápido que um formulário que usa o POST. Porém, não use isso como critério para decidir qual método aplicar ao seu formulário. Foque-se primordialmente nos pontos levantados logo acima e sua escolha será muito mais sábia.

## Usar name ou id: qual é o mais útil e recomendado?

Outra dúvida bem comum entre os meus alunos é em relação ao uso dos atributos name e id em elementos de formulários. Pode parecer que os dois servem para identificar os elementos e que o uso de ambos é algo desnecessário, mas não é. Vamos dar uma olhada na linha do exercício que estamos criando:

```
<input type="text" name="fnome" id="camponome">
```

De forma objetiva, **use sempre os dois** em todos os seus elementos, pois cada um terá sua utilidade específica.

O name será bem útil durante o envio dos dados do formulário, independente do método escolhido para enviá-los. É ele que aparece na *URL* (ao usar GET) e no corpo do *HTTP request* (ao usar POST). É também através do name escolhido que vamos nos referenciar quando for a hora de programar o arquivo em *PHP* que vai receber esses dados.

Já o id será muito útil para fazer a ligação com o `<label>` (sempre use *labels*, por favor) e também será referenciado nos casos em que quisermos tratar esses dados

usando *JavaScript* para realizar qualquer ação, como por exemplo, fazer validação local dos campos antes de enviá-los.

Sendo assim, já disse mas repito: use sempre os dois atributos. Todo elemento de formulário que contenha dados deverá ter `name` e `id`. Nada de preguiça!

## As múltiplas faces do <input>

A tag `<input>` é uma das mais versáteis da linguagem HTML5 (linguagem de marcação, não de programação, nunca se esqueça), e digo isso porque é possível criar muitos componentes de formulário usando apenas essa *tag*. Veja a seguir os seguintes exemplos:

Foto do perfil:  Nenhum arquivo selecionado

Idade:

Sexo

Masculino  
 Feminino

Telefone:

E-mail:

Senha

Mês de aniversário:  

Dia da compra:  

Hora da compra:  

Cor:

Nível de satisfação: 

Aceito os [termos de uso](#)

Todos os elementos acima, por mais que possam parecer visualmente diferentes, foram criados essencialmente usando `<label>` e `<input>` na sua construção (exceto no campo sexo, onde foi preciso criar um `<fieldset>`). O trecho completo do código usado para criar o formulário acima será apresentado a seguir, e analisaremos suas principais configurações logo depois, sempre referenciando o número da linha para melhor organização e entendimento.

Mas antes de olhar a próxima página, um aviso: não se assuste! Vai ficar tudo bem!

```

1   <form action="cadastro.php" method="get">
2     <p><label for="ifoto">Foto do perfil:</label>
3     <input type="file" name="foto" id="ifoto" accept="image/*"></p>
4     <p><label for="idade">Idade:</label>
5     <input type="number" name="idade" id="idade" min="1" max="120"
6       value="20"></p>
7     <p><fieldset style="width: 100px;">
8       <legend>Sexo</legend>
9       <input type="radio" name="sex" id="isexmasc" checked>
10      <label for="isexmasc">Masculino</label>
11      <input type="radio" name="sex" id="isexfem">
12      <label for="isexfem">Feminino</label>
13    </fieldset></p>
14    <p><label for="itel">Telefone:</label>
15    <input type="tel" name="tel" id="itel" pattern="\\([0-9]{2}\\)([\\s])
16      [0-9]{4,6}-[0-9]{4}" placeholder="(XX) XXXX-XXXX"></p>
17    <p><label for="imail">E-mail:</label>
18    <input type="email" name="mail" id="imail" size="30"
19      maxlength="50"></p>
20    <p><label for="isenha">Senha</label>
21    <input type="password" name="senha" id="isenha" minlength="3"
22      maxlength="8" size="8"></p>
23    <p><label for="imes">Mês de aniversário:</label>
24    <input type="month" name="mes" id="imes"></p>
25    <p><label for="idata">Dia da compra:</label>
26    <input type="date" name="data" id="idata"></p>
27    <p><label for="ihora">Hora da compra:</label>
28    <input type="time" name="hora" id="ihora"></p>
29    <p><label for="icor">Cor:</label>
30    <input type="color" name="cor" id="icor" value="#000000"></p>
31    <p><label for="isat">Nível de satisfação:</label>
32    <input type="range" name="sat" id="isat" min="0" max="4"
33      value="2"></p>
34    <p><input type="checkbox" name="aceito" id="iaceito">
35      <label for="iaceito">Aceito os <a href="termos.html">termos de
36      uso</a></label></p>
37    <p><input type="submit" value="Enviar">
38      <input type="reset" value="Limpar"></p>
39  </form>

```

O formulário se inicia na **linha 1** e termina na **linha 33**. Até aí, tudo bem! Note que cada um dos campos do formulário foi devidamente limitado dentro de um parágrafo. Use esses parágrafos para servir de referência e aprender a usar cada um dos elementos.

Nas **linhas 2 e 3**, criamos um campo do tipo file, que permite o visitante fazer o upload de arquivos em qualquer formato. No exemplo dado, usamos o atributo accept

para limitar o upload a arquivos de imagem, já que queremos o envio a penas de fotos de perfil.



**LISTA COMPLETA DE FORMATOS:** Você pode consultar o site da IANA para uma listagem com todos os formatos suportados.

Media Types: <http://www.iana.org/assignments/media-types/media-types.xhtml>

Nas **linhas 4 e 5**, usamos o tipo number para criar uma caixa que aceita apenas números. Inclusive, usamos os atributos min e max para definir os limites aceitos e o atributo value para sugerir um valor inicial, que pode ser alterado para qualquer outro valor dentro do limite durante o preenchimento do formulário.

As **linhas 6 até 12** definem um grupo de campos na forma de um <fieldset>. Por padrão, é exibida uma linha em volta do controle (pode ser personalizada via CSS) e permite usar a tag <legend> para descrever qual será o texto que identifica o grupo.

Note que na **linha 8** e na **linha 10**, os botões de rádio (aqueles redondinhos que podem ser selecionados individualmente) possuem o mesmo name. Isso é necessário para indicar que os dois fazem parte do mesmo grupo a ser selecionado. Apenas um dele pode ser marcado e o primeiro já vai aparecer pressionado, pois usamos checked nele. Seus id são diferentes, para que possamos usar os *labels* corretamente.



**CUIDADO!** Quando estiver usando botões de radio, todos aqueles que fazem parte do mesmo grupo, devem ter o mesmo nome! Faça uma experiência, coloque nomes diferentes e veja o resultado ao tentar marcá-los!

Nas **linhas 13 e 14**, usamos um campo do tipo tel para ler um número de telefone no atributo pattern configuramos uma expressão regular para definir qual será o formato aceito para o valor. Já o atributo placeholder vai configurar uma “dica” para que seu visitante saiba exatamente qual é o formato que vai usar para informar seu telefone.

- \([0-9]{2}\) vai indicar que é necessário colocar dois dígitos de 0 a 9 entre parênteses.
- (\s) indica que logo em seguida, teremos um espaço, logo após o fechamento dos parênteses.
- [0-9]{4,6} - configura que podem existir entre 4 e 6 dígitos numéricos entre 0 e 9 e logo depois, um traço.
- [0-9]{4} por fim, o número se encerra com 4 dígitos entre 0 e 9.

As **linhas 15 e 16** servem para solicitar um e-mail, e isso vai exigir que o campo tenha uma @ para ser considerado válido. Também configuramos o tamanho máximo com maxlen e nenhum endereço digitado poderá conter mais de 50 caracteres. Também configuramos o tamanho físico da caixa para exibir 30 caracteres por vez ao

usar o `size`. Esses últimos dois atributos também podem ser usados em vários outros tipos de campo.

Isso se comprova, pois nas **linhas 17 e 18** criamos um campo para solicitar a senha e também usamos eles. Além disso, adicionamos o atributo `minlength` que em conjunto com o `maxlength`, vai indicar que nossa senha vai ter entre 3 e 8 caracteres.

No trecho de código que engloba as **linhas 19 até 24**, usamos controles especiais para data/hora. Experimente clicar no ícone que fica no final da caixa para ver que o navegador exibe blocos que facilitam a seleção em cada caso.

Nas **linhas 25 e 26**, criamos um controle que permite a seleção de cores e também usamos o atributo `value` para definir a cor padrão, que também pode ser alterada pelo visitante.

As **linhas 27 e 28** foram configuradas para exibir um `range`, exibido como uma pequena barra horizontal com um controle deslizante. Nesse elemento, usamos os atributos `min` e `max` para definir os limites e o `value` para indicar qual será o valor selecionado inicialmente.

Configuramos nas **linhas 29 e 30** um único checkbox, que se diferem dos botões de radio porque podem ter vários itens selecionados dentro de um mesmo grupo. Por conta disso, siga exatamente a regra **oposta** àquela que vimos anteriormente para os radio: **nunca coloque o mesmo nome** em vários checkbox, ok? Cada um deve ter um nome e um id diferentes.



**CUIDADO!** Ao usar um grupo de checkbox em um mesmo formulário, cada um deve ter seu próprio nome e id. Jamais repita um mesmo nome ou id para checkboxes diferentes.

Por fim, nas **linhas 31 e 32**, criamos dois botões: um `submit` e um `reset`. O primeiro vai capturar todos os dados digitados pelo visitante nos campos do formulário e enviar para o documento indicado dentro do atributo `action` da tag `<form>` onde ele está contido. Já o segundo botão vai limpar todos os dados digitados e restaurar todos os valores padrão dos elementos (caso existam).

## Meu controle de formulário não apareceu como o seu. E agora?

Provavelmente, ao criar seu formulário em casa, os componentes foram apresentados de forma um pouco diferente, mas saiba que isso é super natural, pois se tratam de plataformas diferentes (estou desenvolvendo o material no Mac OS X, por exemplo) e também existe diferença de renderização ao usar outros navegadores como o **Firefox**, o **Edge**, o **Safari** e o **Opera**.



As dicas para criar uma experiência próxima do ideal é fazer testes no maior número de navegadores e usar folhas de estilo para formatar visualmente cada componente para não deixar o navegador decidir o formato deles. Outro grande problema é em relação à compatibilidade com alguns tipos de elementos da HTML5. Alguns navegadores sequer suportam campos como color, range e time, por exemplo. Isso se resolve com o tempo, conforme as versões atuais do *browser* aumente sua compatibilidade com as novidades constantes da HTML5 e das CSS.

## E paramos por aqui? Só temos <form> e <input>?

E não é só de `input` que nossos formulários podem ser construídos. Existem outras opções de elementos para aumentar as possibilidades. Vamos ver mais algumas possibilidades:

Modelo do celular:

Especificações técnicas:

Profissão:

No formulário acima, temos uma caixa combinada, daquelas que podemos clicar e escolher um dos itens que vão aparecer em uma lista pré-definida. Logo a seguir, temos uma área de texto, onde digitamos conteúdo em vários parágrafos. Por fim, temos uma caixa de texto comum, mas que possui também uma lista incorporada. Podemos escolher um dos itens disponíveis nessa listagem ou digitar nossa própria resposta.

Mais uma vez, vamos ver como foi possível construir cada um dos elementos do formulário acima, através do trecho de código que utilizamos no documento HTML. Analise cada uma das linhas do código a seguir para poder entender como foi possível criar os elementos.

```

1  <form action="cadastro.php" method="get">
2      <p><label for="imod">Modelo do celular:</label>
3      <select name="modelo" id="imod">
4          <option value="">Escolha...</option>
5          <optgroup label="Apple">
6              <option value="AIP1132">iPhone 11 32GB</option>
7              <option value="AIP11P64">iPhone 11 Pro 64GB</option>
8              <option value="AIP11PM256">iPhone 11 Pro Max 256GB</option>
9          </optgroup>
10         <optgroup label="Samsung">
11             <option value="SGXYS10">Galaxy S10 Lite</option>
12             <option value="SGXYS20U128">Galaxy S20 Ultra 128GB</option>
13             <option value="SGXYN10256">Galaxy Note10 256GB</option>
14         </optgroup>
15     </select></p>
16     <p><label for="idesc">Especificações técnicas:</label><br>
17     <textarea name="desc" id="idesc" cols="40" rows="10"></textarea></p>
18     <p><label for="iprof">Profissão:</label>
19     <input type="text" name="prof" id="iprof" list="lstprof">
20     <datalist id="lstprof">
21         <optgroup label="Negócios">
22             <option value="Administrador"></option>
23             <option value="Contabilista"></option>
24             <option value="Gestor de RH"></option>
25         </optgroup>
26         <optgroup label="Artes e Design">
27             <option value="Animador"></option>
28             <option value="Ator"></option>
29             <option value="Fotógrafo"></option>
30         </optgroup>
31         <optgroup label="Ciências Exatas">
32             <option value="Analista de Sistemas"></option>
33             <option value="Estatístico"></option>
34             <option value="Matemático"></option>
35         </optgroup>
36         <optgroup label="Ciências Humanas">
37             <option value="Advogado"></option>
38             <option value="Linguista"></option>
39             <option value="Professor"></option>
40         </optgroup>
41     </datalist></p>
42     <p><input type="submit" value="Enviar"></p>
43 </form>

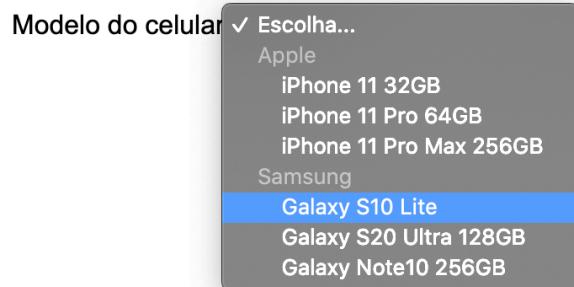
```

Com toda a certeza esses códigos estão ficando mais claros pra você agora. E se por acaso ainda estiver confuso, pode ser sinal de que você não está praticando o suficiente. Você vai ver que com a prática, tudo vai ficando mais simples. Vai por mim, invista um tempo para praticar.

Para começar, criamos um elemento de lista do tipo *dropdown* ao criar o código que está no intervalo que vai da **linha 2 até 15**. Tudo começa, criando a tag `<select>` e dando a ela a identificação necessária através do `id` e do `name`. Cada uma das opções deverá vir entre `<select>` e `</select>`. As opções também podem ser agrupadas em `<optgroup>`, separando-as por assunto caso seja necessário. Caso não exista essa divisão, você pode colocar todos os `<option>` diretamente dentro do `<select>`.

O resultado visual obtido com o código acima para a criação da lista está representado aí ao lado, incluindo a separação em grupos.

Cada `<option>` possui seu `value`, que é o valor que será enviado caso o seu visitante escolha essa opção. Geralmente, quando trabalhamos com listas desse tipo, enviamos pelo formulário o código de cada item, de acordo com o que temos cadastrado em um banco de dados, por exemplo.



Logo a seguir, nas **linhas 16 e 17**, temos uma área de texto que vai aceitar múltiplas linhas. Nesse caso, os parâmetros `rows` e `cols` definem respectivamente em quantas linhas e quantas colunas o conteúdo será exibido, mas saiba que o conteúdo completo pode ter muito mais que o número de linhas configurado. A capacidade de uma `<textarea>` é **ilimitada**.

Profissão:

- ▼
- Administrador
- Contabilista
- Gestor de RH
- Animador
- Autor
- Fotógrafo
- Analista de Sistemas
- Estatístico

O último elemento que criamos nesse formulário foi uma caixa de texto comum, definida diretamente nas **linhas 18 e 19**. A única diferença é a presença do atributo `list`, que vai apontar para uma `<datalist>` que está sendo criada logo abaixo, no intervalo que vai da **linha 20 até 41**.

Assim como `<select>`, uma `<datalist>` também pode conter vários `<option>` em seu interior. Visualmente, a caixa de texto vai aceitar qualquer valor personalizado, mas sempre vai sugerir os valores que definimos dentro das opções, inclusive filtrando os valores exibidos conforme vamos digitando o conteúdo da caixa de texto.

Perceba que durante a preparação desse material que você tem em mãos agora, o uso de `<optgroup>` dentro de uma `<datalist>` não está sendo suportado pelo **Google Chrome**. Faça seus testes em casa e veja se já existe compatibilidade ou não.

## Gerando saídas rápidas utilizando `<output>`

Um recurso que foi adicionado à HTML5 foi o elemento `<output>`, que pode ser usado para executar tarefas simples e gerar uma saída na tela. Essa execução é feita usando

a Linguagem de Programação JavaScript e pode ser executada diretamente no atributo oninput do formulário.

Vamos começar com um formulário bem simples, pedindo informações sobre uma determinada compra:

Produto:

Preço: R\$

Quantidade:

Total: R\$ 0

O código básico para criar o formulário acima será:

```
1   <form action="cadastro.php" method="get">
2     <p><label for="iprod">Produto:</label>
3     <input type="text" name="prod" id="iprod"></p>
4     <p><label for="ipreco">Preço: R$</label>
5     <input type="number" name="preco" id="ipreco"></p>
6     <p><label for="iquant">Quantidade:</label>
7     <input type="number" name="quant" id="iquant"></p>
8     <p><label for="itot">Total: R$</label>
9     <output name="tot" id="itot">0</output></p>
10    <p><input type="submit" value="Enviar"></p>
11  </form>
```

A única novidade é a **linha 9**, onde temos um campo de saída que não pode ser editado pelo seu visitante e sequer será enviado pelo seu formulário. O seu objetivo aqui é manipular o seu conteúdo através de instruções simples.

Para calcular o Total apresentado ali embaixo, basta pegar o preço e multiplicar pela quantidade, não é? Pois então vamos fazer isso pegando o valor de cada elemento identificado pelo seu id transformado em número, o que ficaria:

```
itot.value = Number(ipreco.value) * Number(iquant.value)
```

A forma de “ler” a linha acima é:

*“O valor dentro do total (**itot**) recebe (=) o valor do preço (**ipreco**) convertido para número multiplicado (\*) pelo valor da quantidade (**iquant**) convertido para número.”*

Vamos pegar essa instrução JavaScript que construímos e colocar na **linha 1** do HTML escrito acima, dentro do atributo oninput:

```
<form action="cadastro.php" method="get" oninput="itot.value =
Number(ipreco.value) * Number(iquant.value)">
```

Não se esqueça de conferir se as letras maiúsculas e minúsculas estão sendo digitadas corretamente, pois isso faz total diferença para o JavaScript.

Agora faça um teste e veja se a sua operação está funcionando corretamente!

Produto:

Preço: R\$

Quantidade:

Total: R\$ 540

Viu como o total é calculado automaticamente, assim que digitamos preço e quantidade?



**CUIDADO!** Se não funcionar, provavelmente é o seu código JavaScript digitado errado, verifique se digitou todas as letras maiúsculas e minúsculas exatamente como colocamos aqui no material.



**AS SAÍDAS NÃO SÃO ENVIADAS!** O valor gerado dentro de um elemento <output> não será enviado quando pressionarmos o botão de submissão. Ele serve apenas para mostrar saídas ao seu visitante. Caso queira enviar os dados, pesquise mais sobre como funcionam os elementos do tipo hidden.

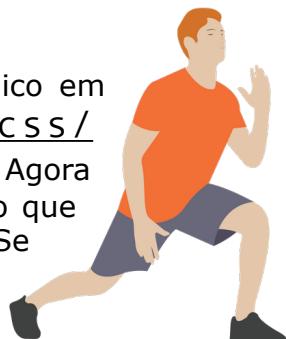


**QUER APRENDER MAIS JAVASCRIPT?** Se quiser se aprofundar um pouco mais no aprendizado de JavaScript, recomendo um curso completo com os primeiros passos na linguagem que eu criei e está completamente grátis:

Curso em Vídeo: [https://www.youtube.com/playlist?list=PLHz\\_AreHm4dlsK3Nr9GVvXCbpQyHQl1o1](https://www.youtube.com/playlist?list=PLHz_AreHm4dlsK3Nr9GVvXCbpQyHQl1o1)

# Hora de exercitar

Chegou a hora de acessar o endereço do nosso repositório público em <https://gustavoguanabara.github.io/html-css/desafios/> e executar os **exercícios ???** no seu computador. Agora tente atingir esse mesmo resultado em casa, sem copiar o código que eu criei. Nesse momento, a prática é algo que você mais precisa. Se por acaso ficar difícil, pode acessar o repositório público de HTML e CSS e dar uma olhada nos comandos, mas **EVITE COPIAR**.



## Tenho desafios pra você!

Lá no repositório, além do material em PDF e dos códigos dos exercícios 100% disponíveis, também disponibilizamos alguns **desafios** que devem ser resolvidos. Esses desafios não incluem o código original e você deve tentar chegar à resposta sem copiar nenhum código.

Com todo o conteúdo que vimos até essa aula, você já pode resolver o **desafio ???**. Acesse o repositório público, abra a área do curso de HTML+CSS e clique no link de acesso aos desafios. Manda ver! Só não fica pedindo a resposta! Você consegue resolver isso sozinho(a)!



Repositório em: <https://gustavoguanabara.github.io>

## Eu já falei sobre isso no YouTube?

Eu sei que às vezes as pessoas gostam mais de assistir vídeos do que ler livros, e é por isso que eu lanço há anos materiais no canal Curso em Vídeo no YouTube. O link que vou compartilhar contigo tem o conteúdo explicado como você leu aqui, só que de forma mais ilustrada. Reserve um tempo dos seus estudos para assistir esse vídeo todo.



Curso em Vídeo: [https://www.youtube.com/playlist?list=PLHz\\_AreHm4d1AnJ\\_jJtV29RFxnPHDuk9o](https://www.youtube.com/playlist?list=PLHz_AreHm4d1AnJ_jJtV29RFxnPHDuk9o)

# DevWeb

## Capítulo 25

# Media Queries

No início, só se podia ter acesso à Internet através de terminais específicos e com configurações limitadas. Hoje em dia a coisa está completamente diferente. Tente enumerar em quais tipos de aparelhos e dispositivos podemos ter acesso à rede. Computadores desktop, notebooks, tablets, smartphones, aparelhos de TV, vídeo-games, vestíveis como relógios e óculos, aparelhos com assistentes pessoais embutidos, eletrodomésticos,... A lista é gigante! Precisamos nos preocupar com essa exibição em múltiplos meios.



Você tem todo o direito de usar esse material para seu próprio aprendizado. Professores também podem ter acesso a todo o conteúdo e usá-lo com seus alunos. Porém todos os que usarem esse material - seja para qual for a finalidade - deverão manter a referência ao material original, criado pelo **Prof. Gustavo Guanabara** e disponível no endereço do seu repositório público <https://github.com/gustavoguanabara/>. Este material não poderá ser utilizado em nenhuma hipótese para ser replicada - integral ou parcialmente - por autores/editoras para criar livros ou apostilas, com finalidade de obter ganho financeiro com ele.



# O desafio de mostrar conteúdo, não importa a mídia

Dê uma bela olhada na imagem a seguir. Ela mostra como o conteúdo deve ser adaptado para poder aparecer em vários tamanhos e formatos de tela.



Esse sempre foi um grande desafio para todos aqueles que desenvolvem qualquer coisa que será exibida nessas telas, ainda mais agora que o tamanho, formato, resolução e capacidade dessas temas varia tanto.

O primeiro passo para adaptar o conteúdo ao tamanho da tela nós já demos! Quando começamos a estudar CSS, falamos bastante sobre recursos que facilitariam a **responsividade**, adaptando o conteúdo ao tamanho da tela usando valores percentuais e as medidas *vh* e *vw*. Porém, é preciso aprender mais para adaptar ainda mais os conteúdos e é aí que entram as **Media Queries**.

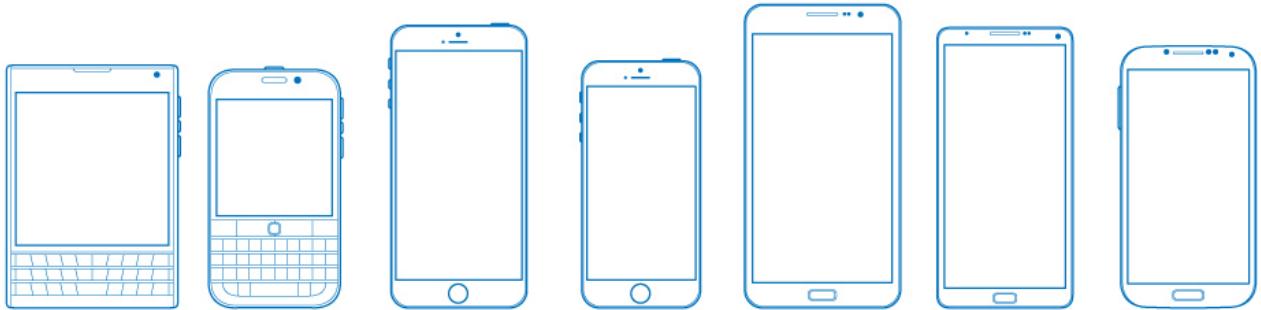


**SINGULAR, PLURAL.** Basta uma busca rápida pela Internet e você vai encontrar os termos **Media Query** e **Media Queries**. Pois saiba que os dois são corretos, e o primeiro é a representação do termo no singular e o segundo está no plural!

Esse termo não possui uma tradução para o Português, mas de forma simples, **media** significa *meio* ou *mídia* e **query** significa *solicitação* ou *indagação*. Sendo assim, a explicação para o termo seria algo como o seu navegador solicitando um formato especial (CSS) para a exibição em meios diferentes.

Os primeiros estudos sobre essa tecnologia surgiram com os **media types**, que basicamente criavam um formato específico para determinado tipo de mídia (tela, impressora, dispositivos para braille, portáteis handheld, projetores, conteúdo lido, TVs, telas de grade fixa, etc).

O principal problema é que só os **media types** não foram suficientes, pois quando falávamos em telas, tínhamos tamanhos, formatos, suporte máximo de cores, resoluções, proporções e compatibilidades muito diferentes. Um exemplo típico é comparar as telas de alguns modelos de smartphones:



E olha que a imagem acima nem mostrou os aparelhos mais recentes onde a tela ocupa a parte frontal completa do dispositivo e foram adicionados aqueles recortes bizarros para caber as câmeras e sensores.



### A ADAPTAÇÃO A TELAS DIFERENTES NÃO É AUTOMÁTICA?

Provavelmente você deve estar se perguntando isso. Pois eu tenho uma PÉSSIMA notícia para te dar. Como desenvolvedores, devemos nos preocupar MUITO com essas resoluções diferentes. Usuários comuns podem pensar que tudo é adaptado como mágica, mas é aí que está a beleza da Tecnologia!

É aí que surgiram as **media queries**. Elas são basicamente as **media types** somadas às **media features**, que são características extras que foram criadas para que possamos personalizar ainda mais nossas folhas de estilo. A seguir, enumero algumas dessas características que podem ser testadas:

- width e height
- device-height e device-width
- aspect-ratio e device-aspect-ratio
- orientation
- resolution
- color e color-index
- grid
- scan

## Como usar uma Media Query?

A declaração de um estilo personalizado para um mídia específica através de uma *media query* pode ser feita de duas maneiras: através da declaração de atributo media em arquivos HTML ou usando uma regra @media em arquivos CSS.

### Media query definida na HTML

No exemplo que vou apresentar a seguir, note que antes de mais nada, usamos uma configuração que está presente desde o início dos nossos exercícios. É uma linha que está dentro da área `<head>` do nosso documento e faz a configuração básica da viewport, fazendo com que ela ocupe 100% da largura disponível do dispositivo onde

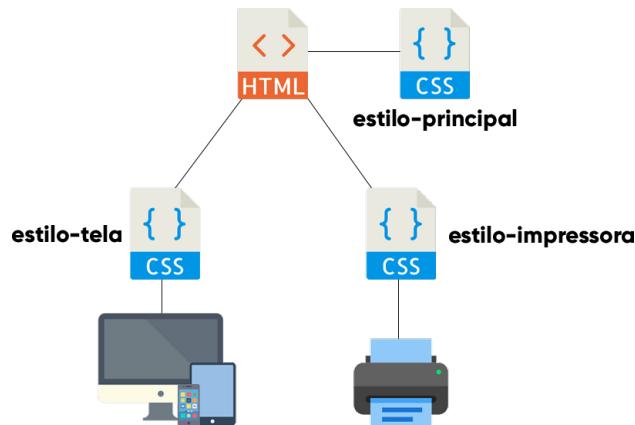
a página esteja sendo exibida, em uma escala 1:1. Já discutimos sobre isso anteriormente quando discutimos os princípios de responsividade.

```
1 <!DOCTYPE html>
2 <html lang="pt-br">
3 <head>
4   <meta charset="UTF-8">
5   <meta name="viewport" content="width=device-width, initial-scale=1.0">
6   <title>Testando o Media Query</title>
7   <link rel="stylesheet" href="estilo-principal.css" media="all">
8   <link rel="stylesheet" href="estilo-tela.css" media="screen">
9   <link rel="stylesheet" href="estilo-impressora.css" media="print">
10 </head>
```

Perceba que a **linha 5** do código acima faz a configuração da *viewport* e essa linha é gerada automaticamente pelo VSCode. Se você está usando outro editor de código, por favor, adicione essa linha! Sem ela, nada feito!

Na **linha 7**, adicionamos o primeiro documento de folha de estilo chamado *estilo-principal.css* que vai servir como base para todos (*all*) os dispositivos, independente da característica.

Já nas **linhas 8 e 9**, fazemos a carga dos documentos CSS com configurações específicas para **telas** e **impressoras**, respectivamente. O funcionamento disso será o seguinte:



Note que o arquivo HTML será o mesmo para todos os dispositivos, assim como o estilo principal CSS, que servirá para todos. Porém, na hora de carregar o site em dispositivos com tela ou em impressoras, teremos estilos específicos para cada um. Neste caso, o arquivo *estilo-principal.css* é quem vai fazer as configurações gerais e elas servirão para todo tipo de dispositivo. Já o *estilo-impressora.css* ou *estilo-tela.css* vão ter as configurações extras, que vão adicionar ou modificar características definidas no estilo principal.

O problema da declaração acima é que definimos apenas os *media types*, o que generaliza muito quando dizemos “dispositivos com tela” e diz que todos eles possuem a mesma configuração. Precisamos adicionar configurações de *media features* para aumentar a personalização. Vamos fazer algumas alterações nas linhas para aumentar as possibilidades:

```

3 <head>
4   <meta charset="UTF-8">
5   <meta name="viewport" content="width=device-width, initial-scale=1.0">
6   <title>Testando o Media Query</title>
7   <link rel="stylesheet" href="estilo-geral.css" media="all">
8   <link rel="stylesheet" href="small-screen.css" media="screen and (max-width: 600px)">
9   <link rel="stylesheet" href="phone.css" media="screen and (min-width: 600px) and (max-width: 768px)">
10  <link rel="stylesheet" href="tablet.css" media="screen and (min-width: 768px) and (max-width: 992px)">
11  <link rel="stylesheet" href="pc.css" media="screen and (min-width: 992px) and (max-width: 1200px)">
12  <link rel="stylesheet" href="tv.css" media="screen and (min-width: 1200px)">
13  <link rel="stylesheet" href="print-portrait.css" media="print and (orientation: portrait)">
14  <link rel="stylesheet" href="print-landscape.css" media="print and (orientation: landscape)">
15 </head>

```

Note que agora, além da configuração geral (**linha 7**) temos configurações especiais para telas pequenas (**linha 8**), smartphones (**linha 9**), tablets (**linha 10**), PCs desktop/notebooks (linha 11) e TVs/telas gigantes (**linha 12**) com tudo adaptável. Também adicionamos configurações específicas para impressoras no modo retrato (folha em pé, **linha 13**) e paisagem (folha deitada, **linha 14**).



**QUE MEDIDAS SÃO ESSAS?** As medidas de tela que usamos acima foram especificadas pela W3Schools como sendo os "*Typical Device Breakpoints*", mas você não é obrigado a seguir esses valores e pode adaptá-los de acordo com medidas mais atualizadas, já que as telas evoluem a cada dia e as resoluções estão melhorando bastante.

W3Schools: [https://www.w3schools.com/howto/howto\\_css\\_media\\_query\\_breakpoints.asp](https://www.w3schools.com/howto/howto_css_media_query_breakpoints.asp)

É claro que você não precisa de todas essas configurações para todos os sites que for construir, mas saiba que é possível criar estilos personalizados para todas as possibilidades que seu projeto necessitar.



**IMPORTANTE!** Perceba que nas declarações de uma *media query*, em primeiro lugar indicamos o *media type* e depois as *media features* delimitadas por parênteses. Isso é obrigatório. Usamos também os operadores *and*, *or*, *not* ou *only* para manter o relacionamento entre as especificações.

## Media query definida nas CSS

Como vimos anteriormente, existem duas maneiras de especificar estilos personalizados para mídias diferentes e já vimos como incluí-las dentro das declarações HTML. A outra maneira é especificando regras `@media` dentro de uma declaração de folhas de estilo.

Vamos criar uma declaração CSS que suporte vários formatos de tela, tudo junto em um único documento. Analise as linhas a seguir com muito cuidado, já que agora teremos níveis diferentes de seletores.

```

/* Configurações gerais */
body {
    background-color: ■blue;
    color: □white;
}

/* Tablets */
@media screen and (min-width: 768px) and (max-width: 992px) {
    body {
        background-color: ■red;
    }
}

/* Monitores desktop/notebook */
@media screen and (min-width: 992px) and (max-width: 1200px) {
    body {
        background-color: □yellow;
        color: ■red;
    }
}

/* Telas grandes e TVs */
@media screen and (min-width: 1200px) {
    body {
        background-color: ■orange;
        color: □yellow;
    }
}

/* Impressoras */
@media print {
    body {
        background-color: □white;
        color: ■black;
    }
}

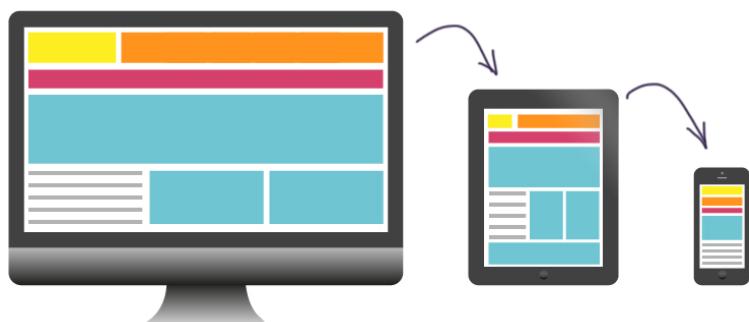
```

Olhando o código acima, perceba que as configurações para Tablets não possuem a configuração de color. Se ela não foi definida na media query personalizada, o que vai valer é o que foi definido nas configurações gerais no início do código.

Outra coisa que provavelmente você deve estar perguntando: "*mas onde estão as configurações especiais para telas pequenas e smartphones?*". Pois é, mas para conseguirmos falar sobre esse aparente "esquecimento", precisamos antes conversar sobre outro assunto.

## O conceito de Mobile First

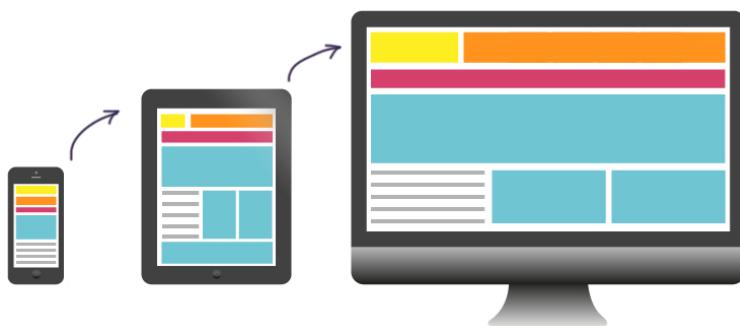
Em uma abordagem mais tradicional, quando pensamos na construção de um site, idealizamos toda a estrutura e os recursos visuais que ele pode ter, criamos o layout completo, implementamos o projeto em um computador e depois nos preocupamos em adaptar toda a sua estrutura para que as telas menores possam exibir o site sem maiores problemas. Isso é o que chamamos de **design responsivo**.



Porém, em uma abordagem um pouco diferente, podemos trilhar exatamente o caminho contrário, pensando inicialmente no layout e funcionalidades da versão móvel e depois adaptando tudo a telas maiores. Esse é o conceito proposto pela linha chamada de **mobile first** (o móvel primeiro), esquema proposto por **Luke Wroblewski** (foto) em 2009. Atualmente chefe de produtos **Google**, LukeW é considerado um dos “pais” dessa abordagem de desenvolvimento.



E esse conceito acaba fazendo muito sentido, pois segundo a GSMA- uma entidade mundial que representa as operadoras móveis - o Brasil é o país da América Latina com mais *smartphones* conectados de toda a América Latina, seguido da Argentina. Isso leva a uma grande probabilidade do seu site ser acessado em um dispositivo portátil conectado a um 4G (ou até 3G) e com tamanho limitado de tela.



E para contribuir ainda mais com essa visão, em 2018 o **Google** anunciou que passaria a valorizar mais na sua indexação os sites que tivessem a preocupação em gerar uma melhor experiência móvel ao usuário. E quem não quer ser valorizado na indexação orgânica da maior ferramenta de buscas do mundo?

Segundo o princípio fundamental da filosofia **mobile first**, pensar primeiro na versão móvel trás algumas vantagens:

- Uma maior divulgação do site, pois como vimos anteriormente, ferramentas de busca vão valorizá-lo.
- Uma melhor experiência do usuário, que vai ter acesso ao conteúdo que foi pensado para que pudesse ser consumido de forma mais confortável e sem confusões.
- Aumento na credibilidade, já que os visitantes vão ter a percepção (ainda que inconsciente) de que quem criou o site se preocupou com a sua experiência.
- Otimização do carregamento, uma vez que sofremos com conexões lentas e dispositivos populares sem tanto poder de processamento.

Dessa maneira, respondendo à dúvida que havia sido levantada anteriormente, o código CSS que criamos não possui uma regra @media específica para dispositivos móveis pois ela simplesmente é a configuração padrão das nossas folhas de estilo. Não esquecemos as configurações para pequenas telas, simplesmente estamos usando o conceito de **mobile first**.

# Colocando a mão na massa

Agora que já sabemos para que serve *media query* e entendemos o conceito de *mobile first*, chegou a hora de fazer um projeto que crie versões personalizadas de um site simples para vários tipos de mídia.

Vamos começar separando as imagens e ícones que usaremos no projeto. As imagens que vamos utilizar foram baixadas diretamente do site **Pexels**, que apresentamos durante o **capítulo 11** e os ícones foram baixados no site **Flaticon**. Esses dois serviços disponibilizam conteúdos livres para uso em nossos projetos.

Todos os ícones também foram redimensionados no **Gimp** para que tenham tamanho padrão de **200x200 pixels** e colocados todos na mesma pasta de imagens.



back-pc.jpg



back-phone.jpg



back-print.jpg



back-tablet.jpg



back-tv.jpg



icon-pc.png



icon-phone.png



icon-print.png



icon-tablet.png



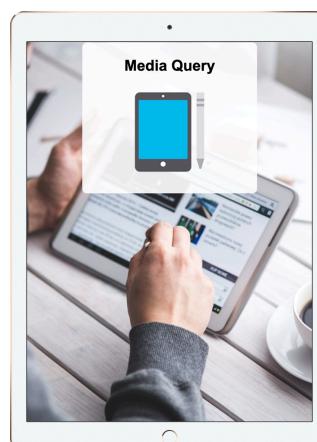
icon-tv.png

Os resultados esperados estão representados nas imagens a seguir. Note que a personalização simples que criamos para cada mídia gera resultados visuais que podem até fazer olhos destreinados perceberem que são sites diferentes, mas se trata da mesma página, apenas com *media query* aplicada.

## Versão para smartphone



## Versão para tablets



## Versão para desktop/ notebooks



## Versão para grandes telas/TVs



## Versão para impressão



**DOWNLOAD DAS IMAGENS:** Mesmo te dando o caminho das pedras e indicando os sites **Pexels** e **Flaticon**, estou deixando as imagens utilizadas no nosso repositório aberto do GitHub. Basta acessar a pasta **ex??** e baixar o arquivo **pacote-imagens-cap25.zip** que está lá.

Repositório:

Para iniciar, vamos criar o arquivo `index.html` e colocar no mesmo local a pasta `imagens` com todos os arquivos `JPG` e `PNG` que vamos usar no projeto.

A base do arquivo `HTML` será apenas uma área principal `<main>` com um título e todos os ícones.

```

1  <!DOCTYPE html>
2  <html lang="pt-br">
3  <head>
4      <meta charset="UTF-8">
5      <meta name="viewport" content="width=device-width, initial-scale=1.0">
6      <title>Testando o Media Query</title>
7      <!-- Aqui vão entrar os estilos e as media queries -->
8  </head>
9  <body>
10     <main>
11         <h1>Media Query</h1>
12         
13         
14         
15         
16         
17     </main>
18 </body>
19 </html>

```

Olhe atentamente nas **linhas de 12 a 16**, onde demos um id para cada `<img>`. Isso será importante para podermos filtrar qual será a imagem a ser exibida em cada mídia.

Agora vamos criar a chamada aos estilos dentro da área `<head>`, exatamente a partir do local indicado na **linha 7** do código acima. Serão ao todo cinco arquivos de folhas de estilo, o primeiro terá as configurações gerais (optamos por aplicar o fundamento de *mobile first* para ele) e os demais para cada uma das mídias restantes.

```

7      <link rel="stylesheet" href="estilos/style.css"
8          media="all">
9      <link rel="stylesheet" href="estilos/tablet.css"
10         media="screen and (min-width: 768px) and (max-width:
11             992px)">
12      <link rel="stylesheet" href="estilos/pc.css"
13          media="screen and (min-width: 992px) and (max-width:
14              1200px)">
15      <link rel="stylesheet" href="estilos/tv.css"
16          media="screen and (min-width: 1200px)">
17      <link rel="stylesheet" href="estilos/print.css"
18          media="only print">

```

Note que na **linha 11**, usamos o operador `only` para deixar claro que essas configurações só serão aplicados para a mídia `print` e não vai afetar de maneira alguma as demais mídias.

Agora vamos segurar a tecla `Ctrl` (`Command` no Mac) e clicar sobre o arquivo `style.css` que está na **linha 7** do código acima. Note que o VSCode também terá que criar a pasta `estilos` que também está indicada na linha.

A estrutura básica do nosso projeto deverá ser essa: o arquivo index e duas pastas, uma para **estilos** (com os arquivos .css) e outra para **imagens**.



O arquivo style.css vai ter um número maior de linhas, já que terá as configurações gerais:

```
1 @charset "UTF-8";
2
3 html {
4     font-family: Arial, Helvetica, sans-serif;
5     font-size: 1em;
6 }
7
8 body {
9     background: black url(../imagens/back-phone.jpg)
10    no-repeat center center;
11    background-size: cover;
12    background-attachment: fixed;
13    color: black;
14 }
15
16 main {
17     background-color: rgba(255, 255, 255, 0.911);
18     text-align: center;
19     width: 350px;
20     height: 300px;
21     margin: auto;
22     padding: 10px;
23     border-radius: 10px;
24 }
25
26 img {
27     display: block;
28     margin: auto;
29 }
30 img#phone { display: block; }
31 img#tablet { display: none; }
32 img#pc { display: none; }
33 img#tv { display: none; }
34 img#print { display: none; }
```

Analisando todas as linhas acima, com certeza você será capaz de compreender todas as declarações dos seletores, pois vimos todas elas detalhadamente em todos os capítulos anteriores. A jogada diferente aqui ficará por conta das **linhas 30 até 34**, onde vamos usar a propriedade display para esconder (valor none) quase todas as imagens, exibindo apenas (valor block) o <img> que vai mostrar o telefone.

O código anterior já vai gerar automaticamente a versão para smartphone. Nossa missão agora é criar as demais, configurando os arquivos para tablet, computadores, TVs e impressoras. O que vamos definir para cada um são apenas as configurações que serão diferentes daquelas que já foram feitas no arquivo geral que apresentamos acima.

## Código do arquivo tablet.css

```
1 @charset "UTF-8";
2
3 body {
4     background-image: url(..../imagens/back-tablet.jpg);
5 }
6
7 img#phone { display: none; }
8 img#tablet { display: block; }
9 img#pc { display: none; }
10 img#tv { display: none; }
11 img#print { display: none; }
```

## Código do arquivo pc.css

```
1 @charset "UTF-8";
2
3 body {
4     background-image: url(..../imagens/back-pc.jpg);
5 }
6
7 img#phone { display: none; }
8 img#tablet { display: none; }
9 img#pc { display: block; }
10 img#tv { display: none; }
11 img#print { display: none; }
```

## Código do arquivo tv.css

```
1 @charset "UTF-8";
2
3 ∵ body {
4     background-image: url(..../imagens/back-tv.jpg);
5 }
6
7 img#phone { display: none; }
8 img#tablet { display: none; }
9 img#pc { display: none; }
10 img#tv { display: block; }
11 img#print { display: none; }
```

# Código do arquivo print.css

```
1 @charset "UTF-8";
2
3 body {
4     background-image: url(..../imagens/back-print.jpg);
5 }
6
7 main {
8     border: 1px solid black;
9 }
10
11 img#phone { display: none; }
12 img#tablet { display: none; }
13 img#pc { display: none; }
14 img#tv { display: none; }
15 img#print { display: block; }
```



**IMAGEM DE FUNDO SUMIU?** Mesmo colocando uma background-image para a versão de impressora ela não será impressa! Não se preocupe, isso é um padrão para a impressão não ficar visualmente poluída. Se por acaso alguma outra imagem não abrir, verifique com toda atenção a digitação dos comandos e use o *autocomplete* do **VSCode** sempre que possível.

Viu como é fácil criar um site que se adapte a múltiplos tipos de mídia? É claro que você não precisa criar todas as possibilidades em cada projeto, mas isso é totalmente possível.

Com o tempo e muita prática depois, você vai ser capaz de montar o layout de sites já pensando em exibi-los em múltiplos dispositivos.



# E paramos por aqui?

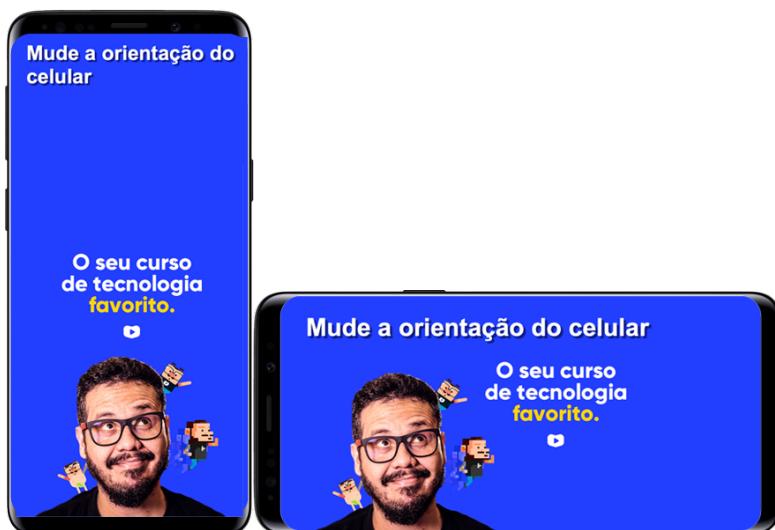
Se você acha que seus estudos sobre *media query* termina por aqui, aí é que você se engana! O objetivo é criar três projetos que vão testar nossos conhecimentos em múltiplas mídias e adaptar os conteúdos a cada uma delas.



**ONDE ESSES PROJETOS ESTARÃO?** O código para esses projetos serão apresentados em forma de vídeos, já que colocar tudo em formato impresso ficaria muito extenso. Vamos aproveitar as múltiplas mídias por aqui também. A essência de *media query* está explicada aqui, os projetos serão desenvolvidos em vídeos.

## Projeto orientação

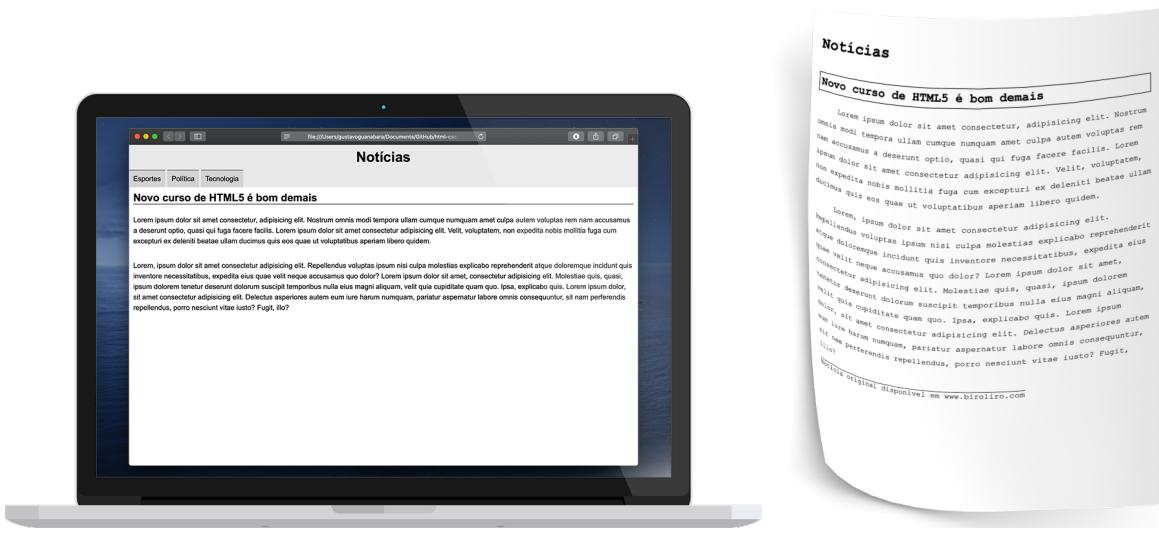
O primeiro projeto é o mais simples de todos e vai criar um site que se adapta ao formato do dispositivo, principalmente os móveis. Estando em pé, a imagem se adapta à parte de baixo do aparelho e a mensagem aparece escrita em cima da figura. Basta deitá-lo e a imagem ficará no canto esquerdo e a mensagem aparecerá na direita.



Nas imagens acima, apresento o resultado esperado. Vamos ver como isso é possível de uma maneira extremamente simples.

## Projeto site no papel

O segundo projeto vai mostrar as possibilidades em gerar uma versão especial para impressoras. Provavelmente você já viu isso acontecer em alguns sites, onde a exibição em tela apresenta o site completo, mas na hora de imprimir só será enviado o artigo principal e alguns poucos componentes extras.



Na versão para desktop, o site usa cores, tem um menu superior e o texto em um formato específico para leitura em telas. Já na versão impressa, o conteúdo deixa de exibir cores, perde o menu, ganha bordas, uma apresentação mais coerente com a leitura em papel e um rodapé com o link para acesso futuro.

## Projeto menu responsivo

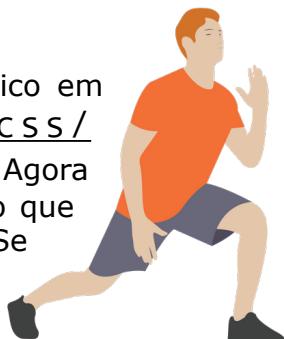
O terceiro e último projeto desse capítulo vai criar um menu adaptável a cada tipo de mídia. Olhe atentamente para as imagens a seguir:



Quando o site é apresentado em uma tela maior, o menu superior é apresentado na forma de pequenos botões organizados um ao lado do outro. Porém, em uma tela mais estreita, o menu muda de forma e dá lugar ao clássico “botão hambúrguer” que pode ser clicado e dá acesso à todas as opções disponíveis no menu. Um resultado muito mais agradável para quem estiver em dispositivos com telas menores.

# Hora de exercitar

Chegou a hora de acessar o endereço do nosso repositório público em <https://gustavoguanabara.github.io/html-css/desafios/> e executar os **exercícios ???** no seu computador. Agora tente atingir esse mesmo resultado em casa, sem copiar o código que eu criei. Nesse momento, a prática é algo que você mais precisa. Se por acaso ficar difícil, pode acessar o repositório público de HTML e CSS e dar uma olhada nos comandos, mas **EVITE COPIAR**.



## Tenho desafios pra você!

Lá no repositório, além do material em PDF e dos códigos dos exercícios 100% disponíveis, também disponibilizamos alguns **desafios** que devem ser resolvidos. Esses desafios não incluem o código original e você deve tentar chegar à resposta sem copiar nenhum código.

Com todo o conteúdo que vimos até essa aula, você já pode resolver o **desafio ???**. Acesse o repositório público, abra a área do curso de HTML+CSS e clique no link de acesso aos desafios. Manda ver! Só não fica pedindo a resposta! Você consegue resolver isso sozinho(a)!



Repositório em: <https://gustavoguanabara.github.io>

## Eu já falei sobre isso no YouTube?

Eu sei que às vezes as pessoas gostam mais de assistir vídeos do que ler livros, e é por isso que eu lanço há anos materiais no canal Curso em Vídeo no YouTube. O link que vou compartilhar contigo tem o conteúdo explicado como você leu aqui, só que de forma mais ilustrada. Reserve um tempo dos seus estudos para assistir esse vídeo todo.



Curso em Vídeo: [https://www.youtube.com/playlist?list=PLHz\\_AreHm4d1AnJ\\_jJtV29RFxnPHDuk9o](https://www.youtube.com/playlist?list=PLHz_AreHm4d1AnJ_jJtV29RFxnPHDuk9o)