Curso Informática para Internet 

# 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

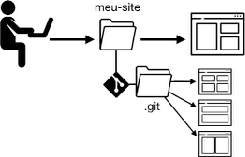
sistemas operacionais. O sistema do Git foi criado no ano de 2005 em poucos dias (10 dias, para ser mais exato), por conta de uma briga entre o *Linus* e o criador de um software chamado **BitKeeper**, que era utilizado para gerenciar as versões em desenvolvimento do *Linux*.

**QUER SABER MAIS SOBRE A TRETA?** Não vou usar espaço desse material para contar a treta entre **Linux 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.

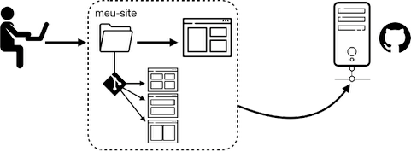


A grande vantagem de usar um **sistema de controle de versões** como o **Git** é poder voltar a qualquer momento para qualquer versão anterior do projeto de forma imediata, tudo 100% transparente para o programador.

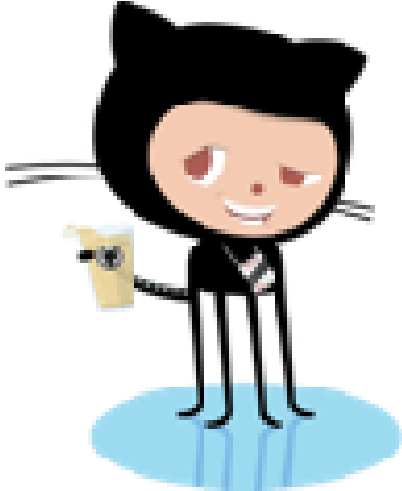
Como você já deve ter percebido, os repositórios locais resolvem o problema que apontamos na **situação 1**, mas se o nosso PC quebrar (**situação 2**) ou se quisermos mostrar o projeto para outra pessoa (**situação 3**), ainda não é possível usando esse tipo de sistema, mas não se desespere agora! Continue 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.

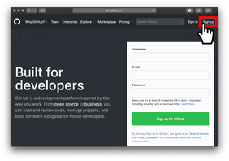
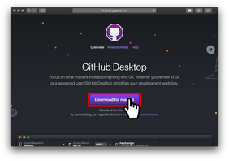
Nesse site, encontramos as versões para todos os sistemas: Windows, Linux e MacOS.

Para a instalação no Linux, o site mostra os comandos necessários para realizar a instalação via gerenciador de pacotes.

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.

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 email de verificação será enviado para ativar sua conta.

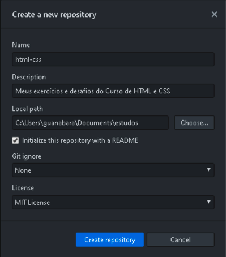


Se já tiver uma conta criada, clique sobre o botão **Sign in** ao lado.

**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

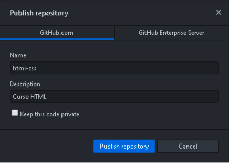
Abra o **GitHub Desktop** (com conexão ativa à Internet, claro) e faça login na sua conta criada no **GitHub** diretamente na tela do programa. Depois de confirmar a conexão, crie um novo repositório em File > New Repository ou pressione Ctrl+N.

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.

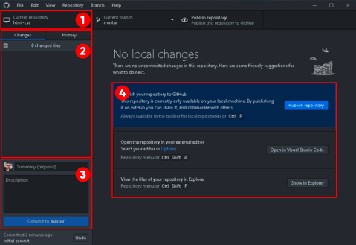
Assim que solicitar a publicação do repositório, deve escolher se vai deixá-lo **público** ou **privado** (padrão). Desmarque a opção *“Keep this Code private”,* principalmente se deseja utilizar o recurso do **GitHub Pages** e hospedar gratuitamente seu site HTML. Só projetos públicos podem ser hospedados 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.

**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.

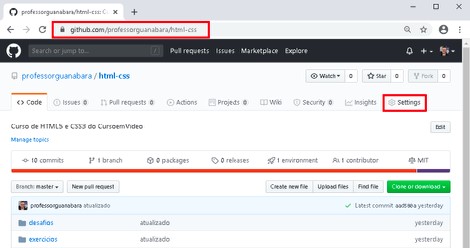
Na **área 4**, também vão aparecer botões muito importantes para abrir o projeto usando o Visual Studio Code ou abrir o gerenciador de arquivos do seu sistema com a pasta do projeto.

## Habilitando a Hospedagem Grátis

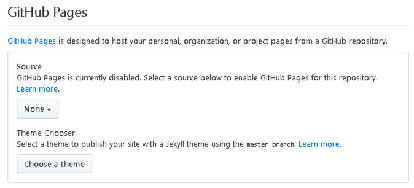
O recurso do **GitHub Pages** não está ativo por padrão em nossos repositórios remotos, mas é muito fácil de habilitar. Isso só pode ser feito diretamente no site do GitHub, individualmente para cada repositório que queremos usar para hospedar projetos HTML.

Acesse o seu perfil no site <https://github.com/nomeusuario> (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.

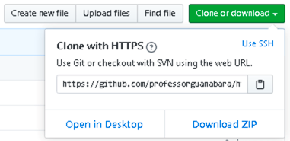


A segunda maneira é clicando sobre o botão **Choose a theme**, que vai permitir escolher um tema para exibir o conteúdo do arquivo README.md como uma página. Essa segunda técnica é mais usada quando vamos hospedar vários projetos em um mesmo repositório ou quando vamos guardar informações sobre um projeto 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](http://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?](https://www.youtube.com/playlist?list=PLHz_AreHm4dm7ZULPAmadvNhH6vk9oNZA)

[list=PLHz\_AreHm4dm7ZULPAmadvNhH6vk9oNZA](https://www.youtube.com/playlist?list=PLHz_AreHm4dm7ZULPAmadvNhH6vk9oNZA)

Curso Informática para Internet 

# DevWeb

Capítulo 19

**Imagens de Fundo**

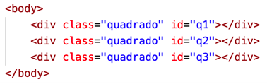
Nos capítulos 6 e 11 do nosso material, aprendemos técnicas para usar imagens como parte do nosso conteúdo. Agora, vamos aprender como usar algumas imagens para complementar visualmente um site, aplicando-as aos fundos dos nossos elementos HTML utilizando estilos. Também vamos ver como manter essas imagens 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-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. |

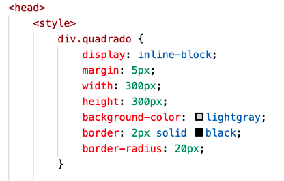


## O fundo não precisa ter só cor

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:



Agora vamos criar a configuração de estilo base para toda <div> que possui a classe quadrado:



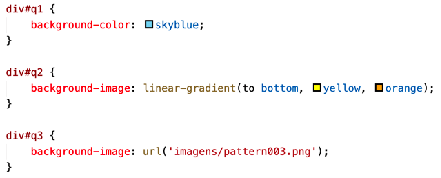
Com esse código aplicado, temos o seguinte resultado na tela:



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:



|  |
| --- |
| **CADÊ AS IMAGENS?** Se você quer as imagens que usaremos nesse capítulo, acesse nosso repositório em [https://github.com/](https://github.com/gustavoguanabara/html-css/tree/master/exercicios/ex022)  [gustavoguanabara/html-css/tree/master/exercicios/ex022](https://github.com/gustavoguanabara/html-css/tree/master/exercicios/ex022). Lá você vai encontrar a pasta com várias imagens que usaremos. |

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().

Ao adicionar as linhas acima ao código, o resultado já muda consideravelmente:



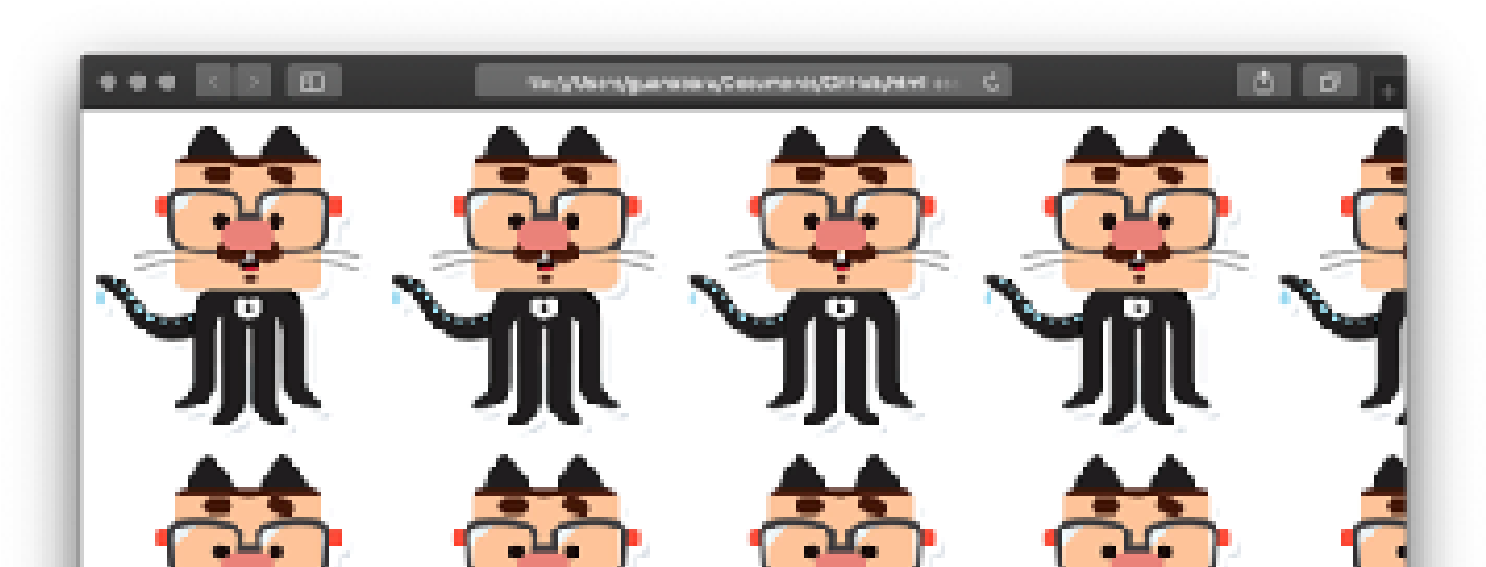
De forma geral, essas são as três maneiras mais simples de preencher uma caixa em HTML: cor sólida, degradê ou imagem de fundo.

## 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.



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 backgroundsize para alterar esse comportamento.

Os valores aceitos por essa propriedade são:

|  |
| --- |
| **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** 🖖🤓 |



|  |  |
| --- | --- |
| 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. |

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:



Podemos reunir tudo em uma única declaração:



## 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:



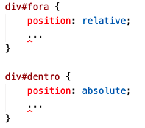
Agora vamos criar as configurações personalizadas para cada <div> dentro da área de <style> na área da cabeça <head> do código:



O resultado disso será algo como:



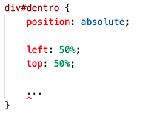
O nosso objetivo aqui é deixar o retângulo interno exatamente no meio do retângulo externo. Vamos começar configurando o posicionamento de cada uma. O retângulo externo terá posicionamento relativo, enquanto o interno terá posicionamento absoluto.



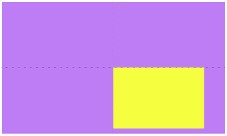
Não crie um novo seletor, apenas adicione as declarações que estou indicando.

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:



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.



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/ desfios/](https://gustavoguanabara.github.io/html-css/desafios/) e executar os **exercícios** **022** 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 d011**. 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](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?](https://www.youtube.com/playlist?list=PLHz_AreHm4dlAnJ_jJtV29RFxnPHDuk9o)

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

Curso Informática para Internet 

# 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 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 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.

|  |
| --- |
| **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> |

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.

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](https://www.recantodasletras.com.br/poesias/3186743)

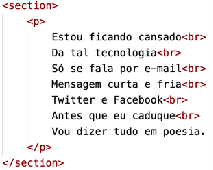
**O projeto pronto**

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.

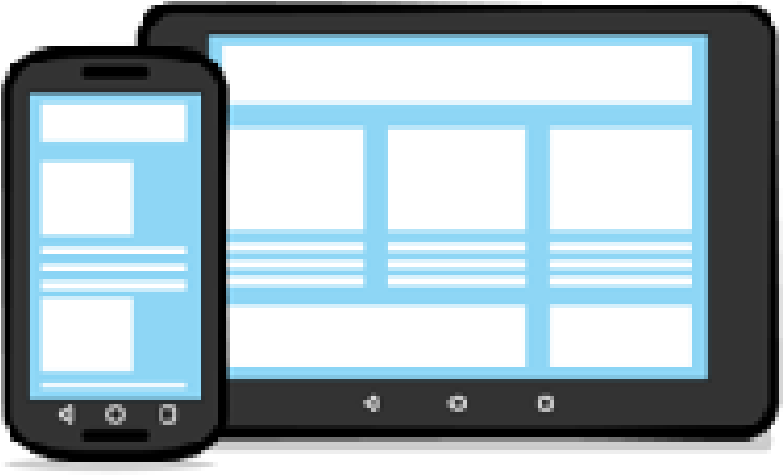
**Organizando o conteúdo em seções**

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.

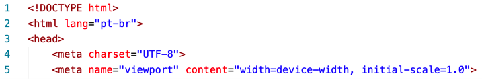


**Como criar um texto que aumenta ou diminui dinamicamente?**

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.



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:



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:



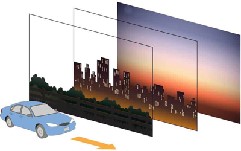
Isso vai significar que o tamanho da fonte será adaptado para que ocupe 10% da largura da viewport, ou seja, em uma tela do Galaxy, com 360px de largura, o tamanho da fonte será 36px. Mas basta deitar o celular para que a largura da tela vá para 740px, fazendo com que a fonte seja mudada para tamanho 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:



Todas as declarações acima foram vistas no capítulo anterior, quando falamos sobre aplicação de imagens em *background*.

**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 d012**. 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](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?](https://www.youtube.com/playlist?list=PLHz_AreHm4dlAnJ_jJtV29RFxnPHDuk9o)

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