

HTML5 & CSS3

Técnicas Avançadas



Versionamento

Arquivos do Projeto

Git

Sistema de Controle de Versões

Controle de Versões

- Todo código criado deve ser versionado
 - A qualquer tempo uma versão anterior deve estar disponível
- Trabalhar de forma manual pode ser arriscada e muitas vezes ineficaz
 - Conseguir precisão de retornar à um ponto específico do código em um determinado momento é praticamente impossível manualmente
 - Muitas vezes se trabalha em equipe e cada membro programará no mesmo projeto, criando novas versões o tempo todo

Controle de Versões

- Existem diversos softwares que permitem controlar a versão do código



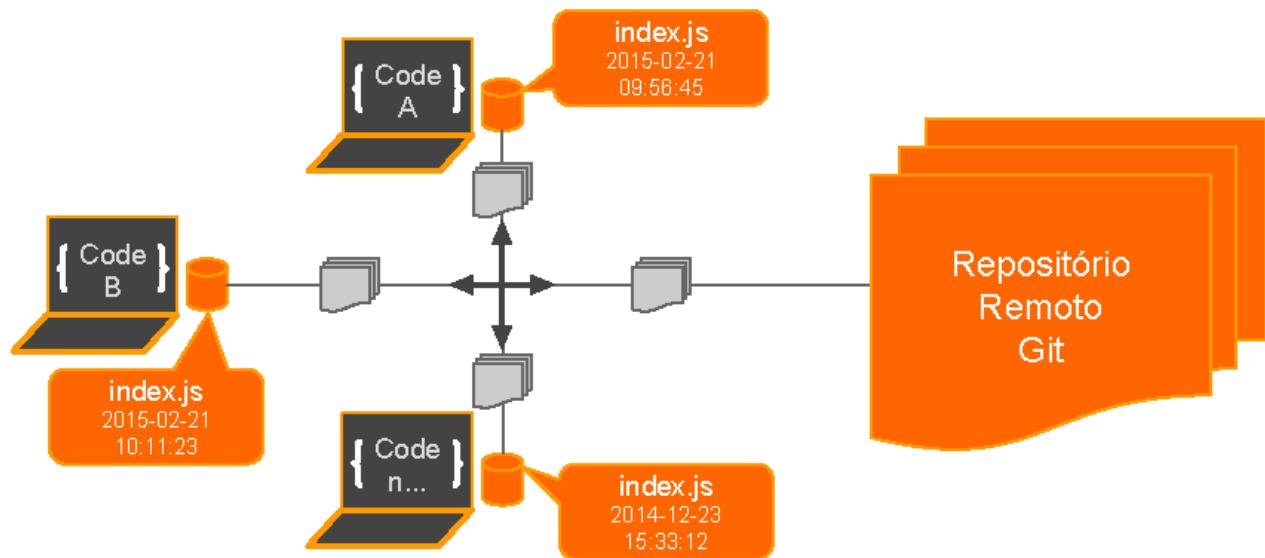
Git

- Mais utilizado pela comunidade de desenvolvedores
 - Criado após identificação de erros e falhas em outros modelos de versionadores
 - Gratuito
- Possui recursos pensados e elaborados para:
 - Desenvolvimento individual e de equipes grandes
 - Múltiplos projetos simultâneos
 - Permite ampla customização e formas de trabalhos

Git



Colaboração



Colaboração

- Cada membro da equipe tem total independência sobre seus arquivos trabalhados
 - Podem ser versionados localmente e posteriormente disponibilizados para integração
- Ao término cada colaborador disponibiliza o(s) arquivo(s) em um repositório central
 - Integração pode ocorrer parcialmente ou integral

Tipos de Arquivos

- Praticamente qualquer tipo de arquivo pode ser versionado
 - Texto
 - Imagens
 - Áudio
 - Vídeo

Repositórios Local e Remoto

- Local
 - Máquina do desenvolvedor
- Remoto
 - Servidor com Git instalado e configurado que possa ser acessado por mais de um desenvolvedor
 - Pode ser interno ou externo como em ambiente “cloud”
- Ambos guardam todo o histórico de mudanças

Nuvem

- Existem diversas empresas que disponibilizam o Git na Web para acesso remoto
 - Alguns serviços são gratuitos outros pagos
- Muitos, serviços gratuitos, permitem apenas versionamento público
 - Qualquer pessoa pode ter acesso
 - Bastante comum para projetos “open source”



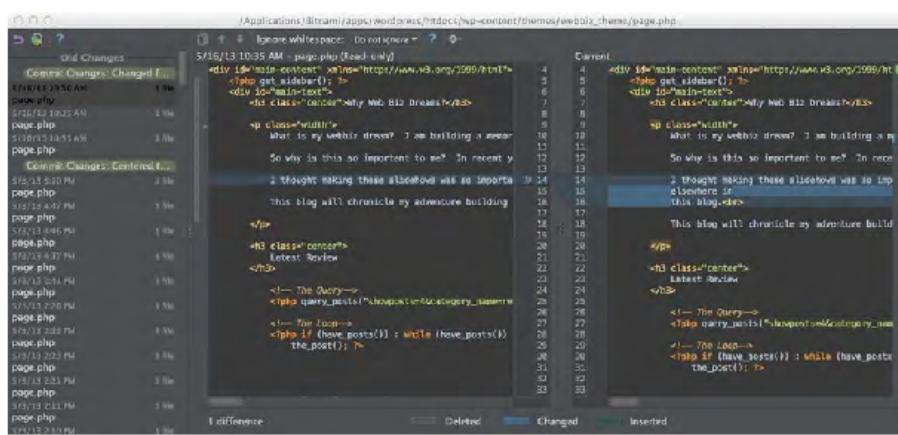
Alterações

- Cada arquivo alterado recebe uma marcação indicando:
 - Mudanças
 - Horário
 - Identificação feita pelo desenvolvedor

Conflitos

- Mesmo arquivo for alterado por colaboradores diferentes
 - Devem ser avaliados e comparados por um dos colaboradores
 - Git informa quais linhas são conflitantes
- Um bom IDE identifica facilmente as informações do Git e destaca na tela para o desenvolvedor
 - Quando configurado, o IDE oferece ferramentas para versionamento

Conflitos



The screenshot shows a diff tool comparing two versions of a PHP file, likely from a GitHub pull request. The left pane shows the 'Old Changes' (original code) and the right pane shows the 'Current' (modified code). A conflict is visible around line 14, where the original code has a block of text starting with 'So why is this so important to me? In recent' and the current code has a different block starting with 'I thought making these slideshows was so import'. The middle section, where the conflict occurs, is highlighted in yellow. Below the code panes, there are buttons for 'Difference', 'Deleted', 'Changed', and 'Inserted'.

```
Old Changes: Changed Line 14
5/15/13 10:35 AM - page.php (Read-only)
    <div id="main-content" xmlns="http://www.w3.org/1999/xhtml">
        <?php get_sidebar(); ?>
        <div id="main">
            <div class="center">My Web BIZ Dreams</div>
            <sp class="width1">
                What is my website about? I am building a new
                site for my website about my website about my website
                So why is this so important to me? In recent
                I thought making these slideshows was so import
                this blog will chronicle my adventure building
            </sp>
            <h3 class="center">
                Latest Review
            </h3>
            <!-- The Query -->
            <?php query_posts('&category__number=1' ?>
            <!-- The Loop -->
            <?php if (have_posts()) : while (have_posts()):
                the_post(); ?>
```

```
Current:
    <div id="main-content" xmlns="http://www.w3.org/1999/xhtml">
        <?php get_sidebar(); ?>
        <div id="main">
            <div class="center">My Web BIZ Dreams</div>
            <sp class="width1">
                What is my website about? I am building a new
                site for my website about my website about my website
                So why is this so important to me? In recent
                I thought making these slideshows was so import
                this blog is
            </sp>
            <h3 class="center">
                Latest Review
            </h3>
            <!-- The Query -->
            <?php query_posts('&category__number=1' ?>
            <!-- The Loop -->
            <?php if (have_posts()) : while (have_posts():
                the_post(); ?>
```

EP: Instalação

1. Acesse o endereço abaixo conforme seu OS
 - a. Mac → <http://git-scm.com/download/mac>
 - b. Win → <http://git-scm.com/download/win>
2. Clique nos arquivos e siga as instruções de instalação
 - a. Perceba que a instalação ocorrerá com sucesso se você possuir privilégios de administrador da máquina
 - b. Anti-virus, anti-spyware, entre outros podem comprometer o sucesso da instalação, recomenda-se desativá-los

Comandos

- Em geral, os comandos de terminal são compostos por:
`command action`
- Opções do comando podem conter ou não hífen,
 - Dependerá da instrução
 - Alguns ainda contém duplo hífen
`command -options action`

Comandos

- Em geral, todos os comandos permitem identificar quais opções usando uma das seguintes notações:

```
command -help
```

```
command --help
```

Navegando com o Terminal

- Navegação com o terminal é bastante simples
 - Basta digitar o comando `cd` (change directory) e colocar o caminho desejado

```
cd Desktop/www.mysite.com
```

Navegando com o Terminal

- Essencial saber em qual pasta o terminal está para criar o caminho correto
 - MAC → `pwd`
 - WIN → `cd`

Listando

- Para listar os arquivos e pastas existentes em uma pasta, basta digitar:
 - MAC → `ls`
 - WIN → `dir`

Configuração Inicial

- Uma vez instalado o software deverá ser configurado
- Comando abaixo deve ser invocado para iniciar o processo:
 - Informações serão exibidas na tela após a execução do comando

```
git config
```

Configuração Inicial

- Próximo passo é a identificação do usuário
 - Email será solicitado, porém não há necessidade de ser um valor real
 - Porém o valor deverá ser único entre os colaboradores que utilizarem
 - O parâmetro “--global” as informações de identificação para qualquer ação feita no Git
 - Em geral este comando é executado uma única vez

```
git config --global user.name "Your Name"
```

```
git config --global user.email "you@email.com"
```

Configuração Inicial

- Para verificar os dados cadastrados basta executar o comando:

```
git config --list
```

EP: Configuração Inicial

- Abra o terminal
 - Win → Menu inicial → Rodar → Digite: cmd
 - Mac → Spotlight → Digite: terminal
- Digite o seguinte comando no terminal
 - Git deve ter sido previamente instalado com sucesso
 - Uma lista de opções aparecerá na tela

```
git config
```

EP: Configuração Inicial

2. (cont.)

```
MacBook-Air:~ mac$ git config --global user.name "DRC"
MacBook-Air:~ mac$ git config --global user.email "drc@email.com"

git config [options] <name> <value>
  --global          use global config file
  --system          use system config file
  --file <file>    use given config file
  --blob <blob>    read config from given blob object

Get or
  --get             get value name [value-regex]
  --get-all         get all values: key [key-regex]
  --get-regexp     get values for regular name regex [value-regex]
  --get-value       get value spec if on the RR : see below
  --replace-all    replace all matching variables name value
  --set             add a new variable: name value
  --unset           remove a variable: name [value-regex]
  --unset-all       remove all matches: name [value-regex]
  --rename-section  rename section: old-name new-name
  --remove-section  remove a section: name
```

EP: Configuração Inicial

3. Digite os seguintes comandos:

- Substitua os caracteres X por seus valores

```
git config --global user.name "xxxxxxx"
```

```
git config --global user.email "xxxx@xxxx.com"
```

```
MacBook-Air:~ mac$ git config --global user.name "DRC"
MacBook-Air:~ mac$ git config --global user.email "drc@email.com"
```

EP: Configuração Inicial

4. Verifique a configuração do usuário:

```
git config --list
```

```
MacBook-Air ~ mact$ git config --list
user.name=DRC
user.email=drc@gmail.com
```

Repositório

- Refere-se ao local no qual os arquivos são versionados
- Existem duas formas de trabalhar com um repositório
 - Criar um novo repositório importando os arquivos de uma pasta local
 - Clonar um repositório do servidor para a máquina local

Importando os Arquivos

- Necessário navegar (utilizando o terminal) para a pasta a qual contém os arquivos a serem versionados
 - Geralmente a pasta do projeto
- Uma vez na pasta desejada, necessário iniciar a importação:

```
git init
```

Importando os Arquivos

- Após execução do comando uma pasta com o nome “.git” será criada
 - Dentro dela arquivos serão criados com as configurações do Git
 - Até este ponto nenhum arquivo ainda foi versionado, apenas o repositório foi configurado

EP: Importando

1. Abra o terminal
 2. Navegue para a pasta “www.tutorial.com”
 - a. O caminho correto dependerá da pasta na qual o terminal foi aberto
 - b. Verifique a pasta na qual o terminal iniciou usando os comandos “pwd” ou “cd” conforme seu OS
 - c. Para saber quais são os arquivos e pastas existentes digite “ls” ou “dir” conforme seu OS
- `cd Desktop/www.tutorial.com`

EP: Importando

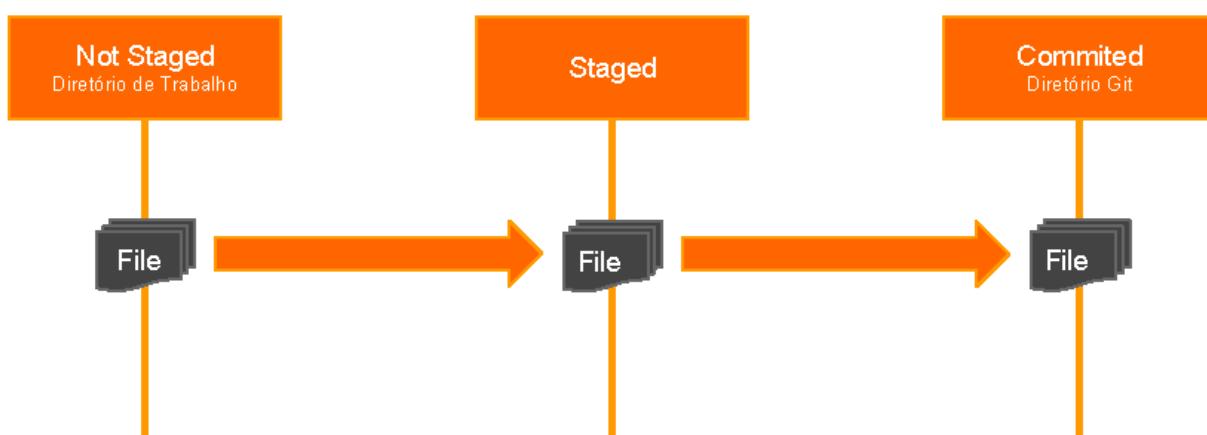
3. Inicie o Git no diretório

`git init`

Operações Locais

- Git possui 3 fases distintas para o versionamento
 - “Not staged”
 - “Staged”
 - “Committed”

Operações Locais



Not Staged

- Fase inicial do Git
- Identifica quais arquivos foram modificados ou criados
 - Nenhum dos comandos “add” ou “commit” foi executado

Stage

- Entre o “add” e o “commit” os arquivos modificados estão em um estado chamado “staged”
 - Modificados mas não foram versionados
 - Aguardando o comando do usuário para versionar o(s) arquivo(s)
 - Estão apenas sendo monitorados pelo Git
 - No terminal é possível verificar quais arquivos estão neste estado pela mensagem “Changes to be committed”

Committed

- Arquivos foram versionados e salvos no repositório
- Será utilizado para comparar as mudanças entre o arquivo “not staged” e o “committed”
 - Qualquer alteração no arquivo “not staged” refletirá a que existe uma mudança pendente

Versionando os Arquivos

- Git permite escolher qual(is) arquivo(s) serão versionados em um repositório
 - Ex: arquivos com código, imagens, fontes, vídeos, etc
- Nem todos os arquivos do projeto devem ser versionados
 - Git oferece o recurso para ignorar os arquivos que não devem ser versionados

Versionando os Arquivos

- Para escolher os arquivos deve-se utilizar o comando:

```
git add readme.txt
```

```
git add *.html
```

```
git add css
```

```
git add js
```

Versionando os Arquivos

- No caso em que todos os arquivos serão versionados basta utilizar o comando abaixo:

```
git add .
```

Versionando Pastas

- Quando uma pasta é adicionada todos os arquivos e subpastas serão adicionados recursivamente:

```
git add css
```

```
git add js
```

commit

- Uma vez escolhidos, estes devem ser versionados quando o desenvolvedor entender que estão prontos
 - `commit` → versionar os arquivos alterados
 - `-m 'some message'` → informação a ser visualizada identificando a versão ou alteração efetuada

```
git commit -m 'iniciando o projeto'
```

commit

- Possível efetuar o “commit” sem passar por “stage”
 - Dependendo do seu fluxo de trabalho esta fase pode ser suprimida
- Basta adicionar ao comando o parâmetro “-a”

```
git commit -a -m 'iniciando o projeto'
```
- Existe ainda a versão reduzida do mesmo comando

```
git commit -am 'iniciando o projeto'
```

EP: Versionando os Arquivos

1. Abra o terminal na pasta do projeto
2. Crie um novo arquivo de nome “readme.txt”
 - a. Adicione algum texto “lorem ipsum”
3. Adicione o novo arquivo ao Git:

```
git add readme.txt
```

EP: Versionando os Arquivos

4. Efetue seu primeiro “commit”:

```
git commit -m 'my first commit on git'
```

Status

- Possível visualizar as ações executadas pelo Git utilizando os comandos “status” e ou “log”
 - status → informações sobre o versionamento dos arquivos

```
git status
```

Status OK

```
git status  
On branch master  
nothing to commit, working directory clean
```

Status Untracked

```
git status  
On branch master  
Untracked files:   
(use "git add <file>..." to include in what will be committed)  
  logo.png   
  support.html   
nothing added to commit but untracked files present  
(use "git add" to track)
```

Status Staged

```
git add logo.png
```

```
git status
```

```
On branch master
```



```
Changes to be committed:
```

```
(use "git reset HEAD <file>..." to unstage)
```

```
new file: logo.png
```

```
modified: readme.txt
```

EP: Status

1. No terminal digite o comando para avaliar o “status” do versionamento de arquivos:

- a. Observe as informações retornadas

```
git status
```

2. Modifique o arquivo “readme.txt”

- a. Remova ou adicione mais “lorem ipsum”

EP: Status & Log

3. Verifique o “status”:

```
git status
```

4. Crie um novo arquivo “version.txt” com texto “1.0.0”

5. Verifique novamente o “status”

- a. Perceba que o novo arquivo foi adicionado a lista “Untracked files”

EP: Status & Log

6. Adicione o novo arquivo e verifique o “status”:

- a. Perceba que o arquivo “readme.txt” continua fora do “status staged”

```
git add version.txt
```

```
git status
```

EP: Status & Log

7. Adicione novamente o arquivo “readme.txt” e verifique o “status”

- a. Perceba que ambos os arquivos estão prontos para o “commit”

```
git add readme.txt
```

```
git status
```

EP: Status & Log

8. Efetue o “commit” com uma nova mensagem”

```
git commit -m "adding new files(readme and version)"
```

9. Verifique o “status” e o “log”

- a. Perceba as mensagens e informações disponíveis

Modificações Após Stage

- Eventualmente um arquivo pode ser modificado antes de seu “commit” e com o arquivo em “stage”
 - Algum erro pode ter sido encontrado e o arquivo deve ser corrigido antes do “commit”
- Neste caso o arquivo aparecerá em ambos estados:
 - “Changes not staged for commit”
 - “Changes to be committed”

Modificações Após Stage

- Se efetuado o “commit” o arquivo na versão em “stage” será usado e não o arquivo modificado
 - Necessário adicionar o arquivo modificado à “stage” antes de efetuar o “commit”

EP: Modificações Após Stage

1. Modifique novamente o texto do arquivo “version.txt”
 - a. Mude o texto para “1.0.1”
2. Verifique o “status” do Git
 - a. Perceba que ele aparece na lista de “Changes not staged for commit”
3. Adicione o arquivo ao “stage” e verifique o “status”
4. Modifique novamente o arquivo “version.txt”
 - a. Altere o texto para “1.0.2”

EP: Modificações Após Stage

5. Verifique o “status”
 - a. Perceba que o arquivo aparece em ambos os estados do Git
 - b. Se efetuado o “commit” agora o arquivo com o texto “1.0.1” será versionado no repositório principal e o arquivo com o texto “1.0.2” continuará pendente de versionamento

Ignorando Arquivos

- Nem todos os arquivos devem ser versionados
 - Os arquivos de configuração não são versionados
 - Referem-se à máquina do desenvolvedor e não ao projeto
 - Deve-se excluir aqueles referentes a configuração do ambiente de desenvolvimento

Ignorando Arquivos

- Muitas ferramentas para edição de código criam arquivos (alguns ocultos) próprios
 - Possuem configurações (caminhos, preferências, etc) do projeto de um dos desenvolvedores do projeto
 - Não podem ser compartilhados, caso contrário sobrescreverá o arquivo de outro desenvolvedor e apagará as configurações do ambiente dele

.gitignore

- Arquivo que contém a lista de pastas, arquivos e ou extensões a serem ignorados no versionamento
 - Arquivo não é criado automaticamente
 - O desenvolvedor deve criar na pasta do projeto o arquivo com as configurações desejadas
 - O arquivo deve possuir o nome de “.gitignore”
 - “.gitignore” pode ser versionado e ser comum à todos no projeto

.gitignore

- Podem ser usadas expressões regulares para maior abrangância (*.ext, *~, *[oa], etc)
 - Para fazer comentários basta começar com #
 - Linhas em branco não são consideradas
- ```
This is a comment
```

# .gitignore

```
ignore any .txt files
*.txt

ignore all files in the "temp" directory
temp/
```

# .gitignore

- Existe ainda alguns sites que indicam quais as configurações comuns de projetos que devem ser ignoradas
  - Alguns ainda geram o arquivo com as configurações
  - Um deles bastante interessante é o “[gitignore.io](https://www.gitignore.io/)”

<https://www.gitignore.io/>

# EP: .gitignore

1. Crie no projeto:
  - a. Arquivos "ignoreme.txt", "pack1.zip" e "pack2.zip"
  - b. Pasta com nome "temp"
  - c. Dentro da pasta "temp" adicione os arquivos de nomes "info.log"
2. Verifique o status
  - a. Observe que o Git aponta as modificações feitas mas os arquivos criados dentro da pasta não aparecem, e sim a pasta com estes
  - b. No caso do Mac aparece um arquivo .DS\_Store criado pelo OS

# EP: .gitignore

## 2. (cont)

```
MacBook_Air:www.tutorialsp.com mac$ git status
On branch master

Initial commit

Untracked files:
 (use "git add <file> ..." to include in what will be committed)

 .DS_Store
 ignoreme.txt
 pack1.zip
 pack2.zip
 temp/
```

## EP: .gitignore

3. Crie o arquivo “.gitignore” na pasta do projeto com o seguinte código:

- a. (Opcional) No caso do Mac adicione também o arquivo “.DS\_Store”

```
ignoreme.txt
```

```
*.zip
```

```
temp/
```

## EP: .gitignore

4. Verifique o status novamente

- a. Perceba que apenas o arquivo “.gitignore” está pendente de versionamento
  - b. Pode ser versionado e disponibilizado de forma comum à todos no projeto

```
git add .gitignore
```

```
git commit -m "gitignore added"
```

# Removendo Arquivos

- Remover um arquivo em “stage”

```
git rm cached filename
```

- Remover um arquivo em “stage” e também do diretório do projeto (apagar o arquivo do projeto)

```
git rm -f filename
```

# Removendo Arquivos

- Remover um arquivo em “committed”
  - Basta apagá-lo e executar o “commit”
- Deve-se utilizar o parâmetro “-u” (“update”) para adicionar ao “stage” antes do “commit”

```
git add -u
```

```
git commit -m "file deleted"
```

# Removendo Arquivos

- Ou efetuar o “commit” com a opção “-a” para não precisar passar por “stage”

```
git commit -a -m "file deleted"
```

## EP: Removendo Arquivos

1. Crie um novo arquivo no projeto de nome “removeme.txt”
2. Verifique o status
3. Adicione o arquivo ao “stage”

```
git add removeme.txt
```

# EP: Removendo Arquivos

4. Utilize o comando e remova do “stage”

```
git rm --cached removeme.txt
```

5. Verifique o status

- a. Perceba que o arquivo voltou a etapa “not staged”

6. Adicione novamente o arquivo à “staged”

```
git add removeme.txt
```

# EP: Removendo Arquivos

7. Tente remover o arquivo do “stage” com o comando:

- a. Uma mensagem aparecerá na tela informando para utilizar a opção “-f” que forçará a remoção

```
git rm removeme.txt
```

8. Adicione a opção “-f”

- a. Perceba que o arquivo foi apagado do projeto

```
git rm -f removeme.txt
```

# EP: Removendo Arquivos

9. Verifique o status
  - a. O arquivo foi removido do projeto e do “stage”
10. Finalmente apague o arquivo “readme.txt” e verifique o status

```
MacBook-A1:xxx.tutorial.com mac$ git status
on branch master
Changes not staged for commit:
 (use "git add <file>..." to update what will be committed)
 (use "git checkout -- <file>" to discard changes in working directory)

 deleted: readme.txt
```

# EP: Removendo Arquivos

11. Finalize com o “commit”
  - a. Outra opção seria adicionar à etapa “staged” e posteriormente ao “committed”  
`git commit -am "readme.txt apagado"`

# Renomeando Arquivos

- Infelizmente o Git não consegue verificar arquivos renomeados
- Quando um arquivo é renomeado, para o Git duas operações ocorreram:
  - Um novo arquivo foi criado
  - O arquivo original (que foi renomeado) foi apagado

# Renomeando Arquivos

- Para resolver isso o Git oferece o comando “mv”
  - Recebe 2 parâmetros, nome do arquivo no repositório e novo nome
- Deve ser executado antes da mudança de nome
  - O próprio comando se encarregará de alterar o nome do arquivo ao projeto e adicioná-lo ao “stage”
  - Posteriormente a operação deve ser finalizada com “commit”

```
git mv current-file-name new-file-name
```

## EP: Renomeando Arquivos

1. Altere o nome do arquivo “version.txt” para “version-number.txt”
2. Verifique o status
  - a. Perceba que o Git aponta duas condições, “not staged” e “untracked”

```
git add version-number.txt
git commit -m "Renomeando o arquivo"
git status
```

## EP: Renomeando Arquivos

3. Renomeei novamente o arquivo para o nome inicial “version.txt”
4. Execute o comando “mv” e verifique o status
  - a. Perceba que o arquivo foi renomeado no projeto e adicionado ao “stage”

```
git mv version.txt version-number.txt
git status
```

# EP: Renomeando Arquivos

## 4. (cont.)

```
MacBook-Air:www.tutorialspoint.com mac$ git mv version.txt version-number.txt
MacBook Air:www.tutorialspoint.com mac$ git status
On branch master
Changes to be committed:
 (use "git reset HEAD <file>..." to unstage)

 renamed: version.txt -> version-number.txt
```

# EP: Renomeando Arquivos

## 5. Finalize a operação com o comando “commit”

```
git commit -m "version.txt renomeado"
```

# Author & Commiter

- No histórico do Git existem duas qualificações:
  - Author → quem criou o arquivo
  - Commiter → qualquer usuário diferente de “author” que efetuou um “commit”

# Histórico de Commits

- Muitas vezes será necessário visualizar os “commits” efetuados
  - Procurar alguma alteração específica
  - Voltar à versão anterior de um ou mais arquivos
- O comando “log” permite visualizar o histórico de “commits” executados
  - Para sair do comando tecle “q + Enter”

# Log

```
git log

commit 158820df5355041a70ad17d1c97a6489906d2195

Author: drc <drc@email.com>

Date: Thu Feb 12 16:48:13 2015 -0200

my first commit on Git
```

# Log

- Comando permite configurar o resultado

| Comando          | Exibe                                                    |
|------------------|----------------------------------------------------------|
| git log -p       | Diferenças entre os arquivos versionados                 |
| git log -2       | Número de linhas a serem exibidas (no exemplo duas)      |
| git log --stat   | Abrevia o resultado                                      |
| git log --pretty | Permite customizar a exibição do resultado               |
| git log --since  | Estabelece a partir de qual data o resultado será obtido |

# EP: Log

1. Visualize todas as execuções de “commit” no projeto
  - a. Analise as informações “Author” e “Date”
  - b. Identifique as mensagens adicionadas ao comando “commit” e perceba a importância destas mensagens
  - c. Visualize que cada “commit” possui uma chave única (sequência de números e letras)

```
git log
```

# EP: Log

1. (cont.)

```
MacBook-Air:www.tutorial.com mac$ git log
commit ffe7323e5f8e2bb6dad71f1d58b6e4ae7f8d8996
Author: DRC <drc@email.com>
Date: Sun Feb 15 14:11:23 2015 -0200

 version.txt renomeado

commit 5971da9deb8de358eeeb1c8684d35f3d6da8392c
Author: DRC <drc@email.com>
Date: Sun Feb 15 13:45:48 2015 -0200

 readme.txt apagado

commit 4a6eea4fa4fc0f44732516c6c2b8e2e6e47f7b8b
Author: DRC <drc@email.com>
Date: Sun Feb 15 11:52:07 2015 -0200
```

## EP: Log

2. Adicione a opção “-2” ao comando para limitar as respostas à apenas 2 linhas

- a. Experimente outros números

```
git log -2
```

## EP: Log

3. Altere o parâmetro do comando para “-p”

- a. Perceba que informações com mais detalhes entre os arquivos são exibidos
  - b. Tecle “Enter” para mostrar mais resultados
  - c. A qualquer tempo tecle “q + Enter” para sair

# EP: Log

4. Combine os comandos anteriores

```
git log -p -1
```

```
MacBook Air:~/code/0107$ git log -p -1
commit 4fcf223c5f902b06cad71f1585d1a97e06996
Author: DRC <drc@east.com>
Date: Sun Feb 15 14:17:28 2015 +0000

 version.txt renomeado

diff --git a/version.number.txt b/version.number.txt
new file mode 100644
index 0000000..e6d5cb8
--- /dev/null
+++ b/version.number.txt
@@ -0,0 +1 @@
+1.0.0
\ No newline at end of file
diff --git a/version.txt b/version.txt
deleted file mode 100644
index e6d5cb8..0000000
a/version.txt
+++ /dev/null
@@ -1 @@
+1.0.0
\ No newline at end of file
```

# EP: Log

5. Modifique a alteração para:

- Perceba as novas informações disponíveis
- Identifique o número ao lado de cada arquivo

```
git log --stat
```

# EP: Log

6. Altere o arquivo “version-number.txt” para “1.0.3” e execute o “commit”
7. Avalie novamente o número que aparece ao lado do arquivo com o comando “log” e o parâmetro “-p”

## Customizando o Resultado

- O parâmetro “--pretty” do comando “log” permite customizar o resultado
  - Inúmeras possibilidades podem ser obtidas, conforme a configuração do parâmetro

```
git log --pretty
```

# Customizando o Resultado

- As combinações mais comuns são:

| Comando                                   | Exibe                                                                                                                            |
|-------------------------------------------|----------------------------------------------------------------------------------------------------------------------------------|
| git log --pretty=oneline                  | Resultados simplificados em uma única linha, muitas informações são omitidas                                                     |
| git log --pretty=format:"%an -- %ar : %s" | Formata o resultado para:<br>%an → email do autor<br>%ar → tempo decorrido desde a última alteração<br>%s → mensagem do “commit” |

## EP: Customizando o Resultado

- Digite o comando abaixo:

```
git log
```

- Utilize o novo comando e observe o resultado:

- Compare com as informações anteriores
- Perceba que muitas informações foram omitidas

```
git log --pretty=oneline
```

## EP: Customizando o Resultado

- Digite um novo comando:

```
git log --pretty=format:"%s"
```

- Repita o comando acrescentando tempo decorrido:

```
git log --pretty=format:"%ar --- %s"
```

- Finalmente adicione o nome do usuário:

```
git log --pretty=format:"%an : %ar --- %s"
```

## Histórico entre Datas

- Possível executar o comando “log” para que obtenha as alterações entre datas
  - Um resultado mais interessante pode ser obtido se usado em conjunto com os parâmetros “--author” ou “--committer”, ou ainda o “--pretty”

```
git log --since="2015-02-01" --until="2015-02-04" --
author="name"
```

# Histórico entre Datas

| <i>Comando</i>               | <i>Exibe</i>                                                     |
|------------------------------|------------------------------------------------------------------|
| git log --since="2015-02-01" | Resultados desde 01/02/2015                                      |
| git log --until="2015-03-01" | Resultados até 01/03/2015                                        |
| git log --before=1.hour      | Resultados antes de 1 hora atrás                                 |
| git log --after="14:11:23"   | Resultados depois das 14 horas, 11 minutos e 23 segundos de hoje |

## EP: Histórico entre Datas

1. Verifique o “log” completo dos “commits” feitos até aqui
2. Aleatoriamente, identifique uma data e ou horário de um deles
3. Utilize um dos comandos com os valores encontrados e avalie o resultado
  - a. Faça combinações para resultados mais pontuais

# EP: Histórico entre Datas

## 3. (cont.)

```
git log --since="YYYY-MM-DD"
git log --until="YYYY-MM-DD"
git log --before=2.hours
git log --after="HH:MM:SS"
```

# git commit --amend

- O parâmetro “--amend” refaz o último “commit”
  - Arquivos podem ser adicionados no último “commit”
  - Se desejado é possível alterar apenas a mensagem do “commit”

```
git commit -m "Some commit"
git add missing-file
git commit --amend
```

## EP: git commit --amend

1. Crie o arquivo “readme.txt” com texto “lorem ipsum”
2. Faça o “commit” do arquivo
  - a. Perceba que o arquivo “version.txt” não foi alterado para a nova versão (deveria ter sido alterado e fazer parte do mesmo “commit”)

```
git add readme.txt
```

```
git commit -m "readme.txt adicionado 1.0.4"
```

## EP: git commit --amend

3. Altere o texto do arquivo “version-number.txt” para “1.0.4” e verifique o status
4. Adicione a alteração do arquivo “version.txt” no último “commit”, utilizando o parâmetro “--amend”
  - a. Quando executado será possível alterar a mensagem também

```
git add version-number.txt
```

```
git commit --amend
```

# git reset

- Permite retirar arquivos de “staged” para “unstaged”
  - Mensagem lembrando este comando sempre aparece quando o comando “status” é executado após o comando “add”

```
git reset HEAD filename
```

## EP: git reset

1. Altere o texto do arquivo “version-number.txt” para “1.0.5”
2. Adicione o arquivo em “stage”
3. Verifique o status
  - a. Identifique a mensagem “use git reset HEAD <file>... to unstage”

# EP: git reset

4. Execute o comando e verifique o status

```
git reset HEAD version-number.txt
```

5. Verifique o status

- a. Perceba que ele voltou a ser “unstaged”
- b. Mantenha o arquivo com a versão “1.0.5” e “unstaged” para o próximo EP

# git checkout

- Permite reverter à um estado anterior
  - Recupera uma determinada versão
  - Novamente as mensagens do “commit” são muito importantes para facilitar a identificação de um determinado ponto
- Comando pode reverter:
  - Arquivos
  - “Commits” (parcial ou total)
  - “Branches” (alterna entre “branches”)

# git checkout -- file

- Reverte as alterações de um arquivo “unstaged” para a versão “commit” do mesmo arquivo
  - Traz de volta a versão do arquivo que está sendo gerenciada pelo Git e desfaz todas as alterações do arquivo feitas em “unstaged”
  - Esta mensagem também é lembrada quando o comando “status” é invocado

```
git checkout -- filename
```

# git checkout -- file

- Este comando deve ser executado apenas se tiver absoluta certeza do que está sendo feito
  - O arquivo “unstaged” sobreescrito terá suas alterações perdidas
  - Equivalente a copiar um arquivo sobre outro existente de mesmo nome (o arquivo existente perderá seu conteúdo em favor do outro)

## EP: git checkout -- file

1. Verifique se o arquivo “version-number.txt” está “unstaged”
  - a. Caso não esteja modifique a versão e deixe o arquivo “unstaged”
  - b. Perceba que a versão do mesmo arquivo em “commit” possui a versão com o texto “1.0.4”
2. Execute o comando

```
git checkout -- version-number.txt
```

## EP: git checkout -- file

3. Verifique o status
  - a. Perceba que não existe nenhuma pendência com relação ao arquivo “version-number.txt”
4. Verifique o texto do arquivo “version-number.txt”
  - a. O número voltou para “1.0.4”

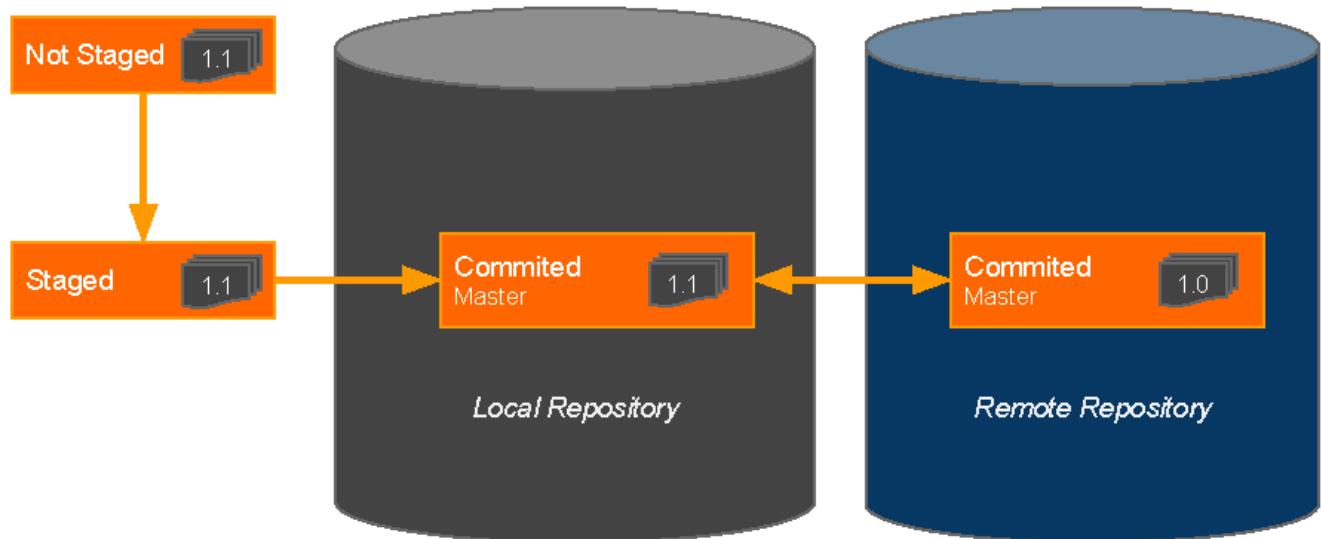
# Versionamento Remoto

- Repositório remoto permite colaborar os arquivos versionados
  - Essencial para que exista a colaboração de arquivos
  - Podem existir mais de um repositório remoto
- Consiste em enviar (“push”) e trazer (“pull”) arquivos entre usuários

# Versionamento Remoto

- Possível controlar o permissionamento de usuários
  - Podem existir repositórios “read only” no qual é possível apenas clonar os arquivos existentes
  - Comum em projetos “open source”

# Versionamento Remoto



## git clone

- Clona um repositório existente
  - Deve estar disponível e acessível em um servidor remoto
  - O nome padrão criado para o repositório é “origin”

```
git clone URL
```

# git clone

- O comando pode ser composto com a pasta de destino
  - Valor é opcional, se não usado o repositório será clonado na pasta atual na qual o comando foi executado no terminal

```
git clone URL FOLDER
```

## EP: git clone/remote

1. Abra o terminal e navegue até o seu “Desktop”
  - a. Utilize os comandos “cd”, “pwd” e ou “ls/dir” conforme seu OS
2. Digite o comando abaixo para clonar o repositório
  - a. A pasta “www.tomatoo.com” não pode existir em seu “Desktop”, caso contrário um erro será lançado

```
git clone https://github.com/drc-tutorial/www.tomatoo.com
```

# EP: git clone/remote

3. Navegue para a pasta “www.tomatoo.com”
4. Verifique o “log”
  - a. As informações de “log” podem diferir do mostrado na imagem abaixo

```
MacBook-Pro:~/Desktop$ git clone https://github.com/drc-tutorial/www.tomatoo.com
Cloning into 'www'...
remote: Counting objects: 7, done.
remote: Total 7 (delta 0), reused 0 (delta 0), pack-reused 0
Unpacking objects: 100% (7/7), done.
Checking connectivity... done.
MacBook-Pro:~/Desktop$ cd www.tomatoo.com
MacBook-Pro:~/Desktop$ git log
commit 92d5da9a233bf1ccdf73d9a18cd/e2e4cd9ba
Author: DRC <drc@drctutorial.com.br>
Date: Tue Feb 17 17:25:16 2015 -0300

Initial commit.
```

## git remote

- Lista o(s) repositório(s) remoto(s) configurado(s)

```
git remote
```

- Para obter informações mais detalhadas, basta acrescentar o parâmetro “-v”

```
git remote -v
```

# git remote add

- Repositório pode ser adicionado manualmente (ao invés de usar o comando “clone”)
  - Necessário iniciar o Git em uma pasta e executar o comando
  - Permite escolher um nome personalizado para o repositório
  - Se utilizar em um projeto clonado, os repositórios podem ficar duplicados com nomes diferentes

```
git remote add shortname url
```

# git remote rename

- Renomeia um repositório
  - Permite alterar o nome padrão “origin” para qualquer outro

```
git remote rename current-name new-name
```

# git remote rm

- Remove um respositório

```
git remote rm repository-name
```

# git remote show

- Exibe informações detalhadas sobre o repositório remoto

```
git remote show repository-name
```

## EP: git remote

1. Navegue até a pasta “www.tomatoo.com” no “Desktop”
2. Execute o comando abaixo para listar o nome dos repositórios:

```
git remote
```

3. Acrescente o parâmetro “-v” e compare as informações:

```
git remote -v
```

## EP: git remote

4. Adicione o repositório manualmente
  - a. Ao final existirão 2 repositórios duplicados, porém com nomes diferentes

```
git remote add too https://github.com/drc-tutorial/www.tomatoo.com
```

5. Liste novamente os repositórios

```
git remote
```

# EP: git remote

6. Renomeie o repositório com o nome “too”

```
git remote rename too tmt
```

7. Liste novamente os repositórios

```
git remote -v
```

8. Exclua o repositório duplicado

```
git remote rm tmt
```

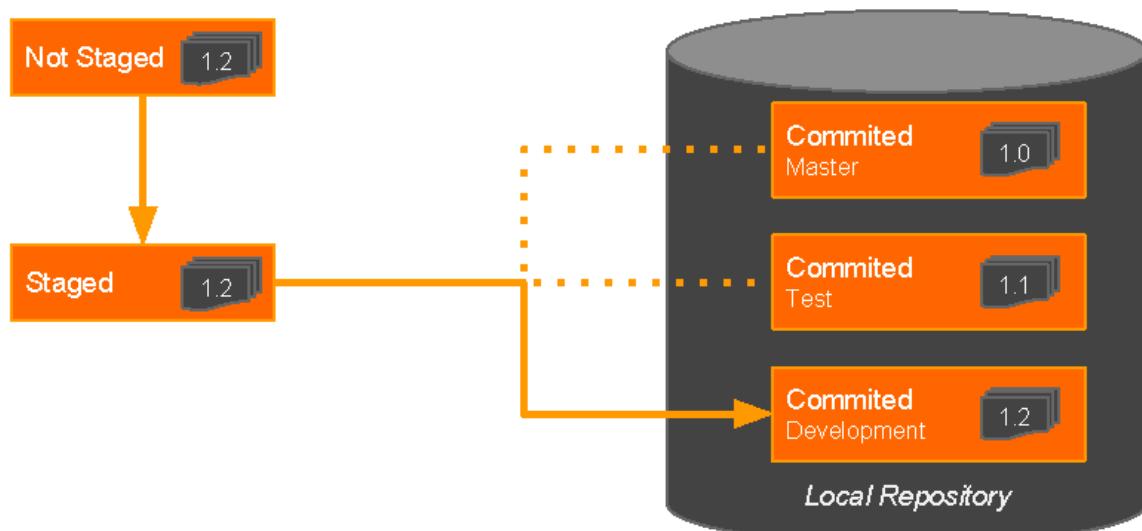
# Branch

- Derivações de um repositório de versionamento
  - Ramificações do repositório
- Importante, pois permite criar linhas diferentes de um mesmo repositório
  - Diferentes “branches” podem ser criadas, como produção, teste e desenvolvimento

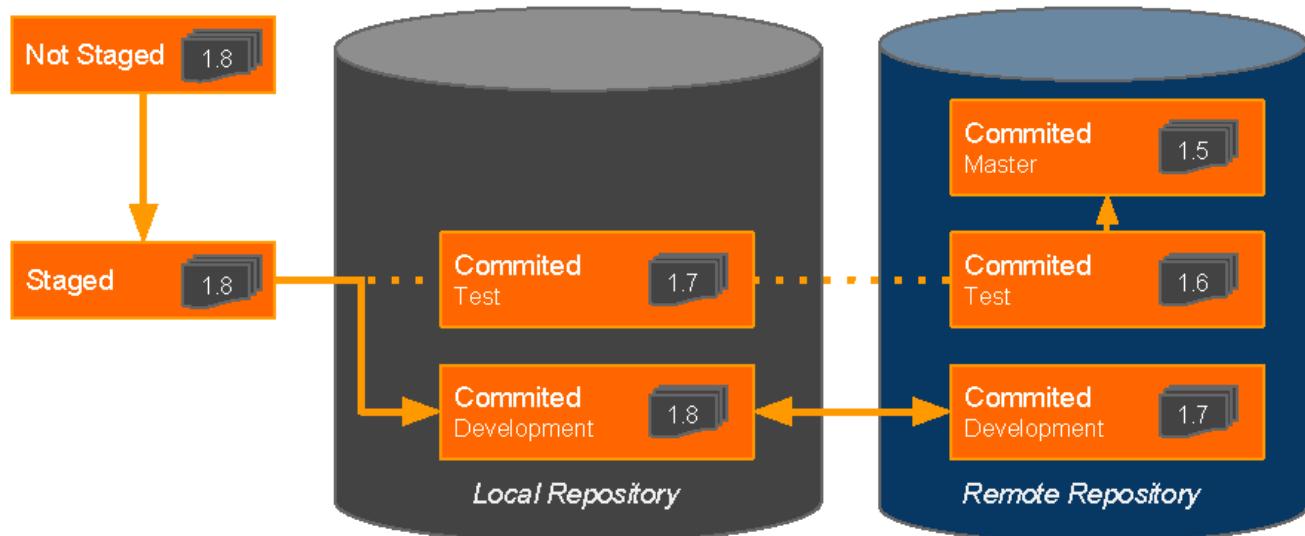
# Branch

- Cada “branch” permitirá o desenvolvimento contínuo e simultâneo sem interferência entre elas
- Resolve cenários como:
  - Como continuar modificando um arquivo enquanto outro igual ainda está em testes, o qual poderá necessitar de modificações (“bugs”)
  - Arquivos em produção necessitam de modificações e o desenvolvimento está em andamento de uma nova versão, ou seja, será necessário reparar o arquivo da versão em produção

## Local Branches



# Local & Remote Branches



## Criando Branches

- Quando executado o comando “git init” o repositório é criado com uma única “branch” de nome de “master”
  - Não confundir o nome do “branch” com o nome do repositório remoto padrão (“origin”)
- Novas “branches” podem ser criadas a qualquer tempo
  - Importante definir a estratégia de quais “branches” existirão e como serão usados junto com os demais usuários

# git branch

- Quando existem mais de um “branch” é possível escolher para qual o arquivo versionado será enviado
  - O mesmo ocorre para os demais comandos como “pull”, “fetch”, “clone”, “reset”, etc
- O comando apenas cria o “branch” não efetua a troca
  - As ações ainda continuarão a ocorrer na anterior em uso

```
git branch name
```

# git branch -r

- O parâmetro “-r” permite visualizar os “branches” remotos

```
git branch -r
```

# git branch -d

- “Branches” podem ser criadas temporariamente
  - Para resolver um “bug” por exemplo
  - As temporárias devem ser excluídas após seu uso
- A opção “-d” do comando remove a “branch”

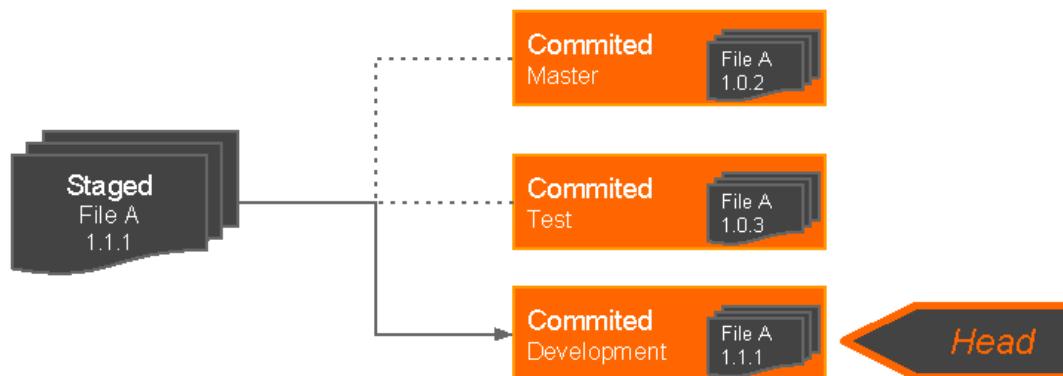
```
git branch -d name
```

# Head

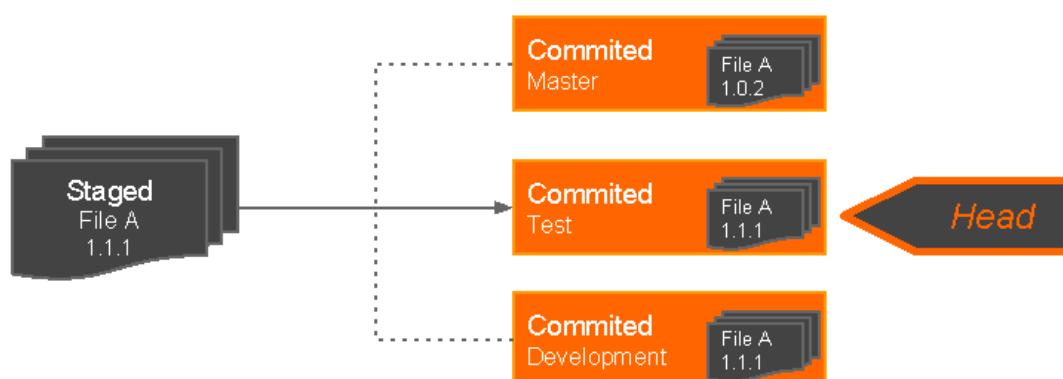
- Identifica qual “branch” está em uso
  - Aponta para o “branch” utilizando um ponteiro de nome “head”
- Para mudar de “branch” é necessário usar o comando “checkout” e o nome do “branch”
  - O ponteiro “head” apontará para o “branch” definido no comando
  - O “branch” deve existir para que a troca ocorra

```
git checkout branch-name
```

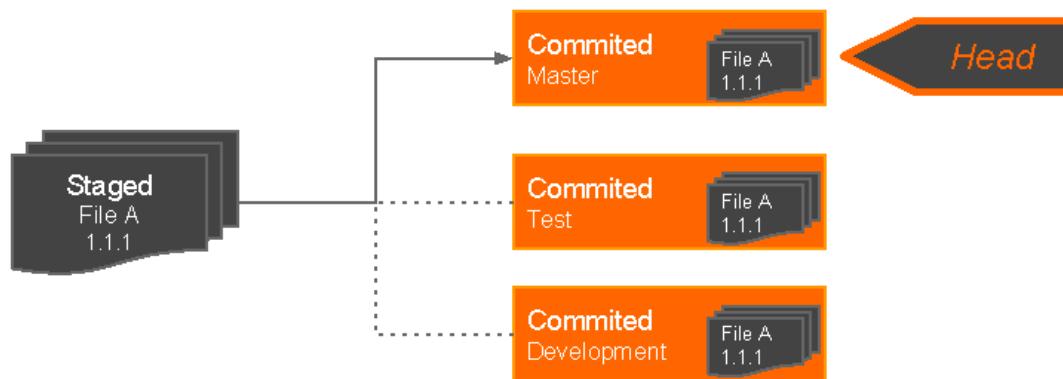
# Head



# Head



# Head



## Alternando Branches

- Prática bastante comum para que o desenvolvimento do projeto continue
  - Enquanto está em testes
  - “Bug” foi encontrado no ambiente de produção e precisa ser reparada a versão específica de um código
- Organização e senso comum são fundamentais
  - Convenção e regras entre os usuários são essenciais

# Alternando Branches

- Considere o seguinte cenário para o arquivo “A”:
  - a. Execução do “commit” no “branch” de nome “test”
  - b. Alterna para o segundo “branch” de nome “development”
  - c. Execução do “commit” de outra versão do arquivo “A” no “branch development”

# Alternando Branches

```
git checkout test
git commit -am "A file ready to test"
git checkout development
git commit -am "A file updated"
```

# Alternando Branches

- Os “branches” possuirão arquivos em versões diferentes
  - A qualquer tempo qualquer uma das versões poderá ser recuperada e alterada
  - Por fim o “merge” pode ser feito combinando e sincronizando ambos os “branches” com a mesma versão de arquivo

## Criando e Alternando Branches

- O comando “checkout” pode receber o parâmetro “-b” que cria e alterna para a nova “branch”

```
git checkout -b test
```

# EP: Branch

1. Abra o terminal na pasta “www.tomatoo.com”
2. Verifique o status
  - a. Tenha certeza de que nenhuma operação está pendente
  - b. Caso exista alguma pendência, resolva antes de prosseguir
3. Verifique as “branches” existentes
  - a. Perceba o caractere “\*” ao lado do nome “master”

```
git branch
```

```
MacBook-Air:www.tomatoo.com mat:$ git branch
 hotfix
* master
```

# EP: Branch

4. Adicione uma nova “branch”
  - a. A “branch” será criada apenas na máquina local

```
git branch hotfix
```

5. Verifique novamente as “branches” existentes
  - a. Perceba que a nova “branch” foi adicionada

```
git branch
```

6. Alterne para a nova “branch”
  - a. Use o comando “git checkout”

```
git checkout hotfix
```

# EP: Branch

7. Verifique a “branch” em uso
  - a. Perceba que o caractere “\*” mudou para o nome “hotfix”

```
git branch
```
8. No Sublime, altere o título da página “index.html” para “TOMATOO Co v1.0.0”
9. Verifique o status

# EP: Branch

10. Faça o “commit” da alteração
  - a. Identifique a mensagem retornada no terminal informando o nome do “branch” [hotfix ...]
  - b. Neste momento existem duas versões diferentes de um mesmo arquivo em “branches” diferentes

```
git commit -am "index title update"
```

# EP: Branch

11. Mais uma vez altere o título do arquivo “index.html” para “ToMaTOO Co v1.0.1”
12. Tente alternar para a “branch master”
  - a. Um erro será lançado indicando que modificações foram feitas e estão pendentes e devem ser resolvidas antes de prosseguir

```
git checkout master
```

```
MacBook Air:www.tomatoocn mac$ git checkout master
error: Your local changes to the following files would be overwritten by checkout:
 index.html
Please, commit your changes or stash them before you can switch branches.
Aborting
```

# EP: Branch

13. Adicione ao “stage” e corrija o “commit” com o parâmetro “--amend”

```
git add index.html
```

```
git commit --amend
```

14. Alterne de “branch”

```
git checkout master
```

# git merge

- Combina (“merge”) das “branches”
  - Atualiza os arquivos das “branches”
  - Conflitos podem ocorrer e devem ser resolvidos pelo desenvolvedor
- Deve estar em uso a “branch” na qual será atualizada pelos arquivos da segunda
  - Apenas a “branch” que será atualizada será modificada

```
git merge brach-name-with-updated-files
```

# git merge

- Para verificar quais “branches” já foram ou não feitos os “merges” existem os parâmetros do comando “branch”:

```
git branch --merged
```

```
git branch --no-merged
```

# EP: Merge

1. No terminal (ainda no projeto “www.tomatoo.com”) verifique qual a “branch” em uso
  - a. Tenha certeza de estar na “branch master”
2. Execute o comando abaixo:

```
git merge hotfix
```

```
MacBook-Air:www.tomatoo.com mac$ git merge hotfix
Updating c8a8810..fcb38ca
Fast-forward
 index.html 2 +-
 1 file changed, 1 insert(+), 1 deletion(-)
```

# EP: Merge

3. Apague a “branch hotfix”:

```
git branch -d hotfix
```

4. Verifique as “branches” existentes

```
git branch
```

```
MacBook-Air:www.tomatoo.com mac$ git branch
 * master
 hotfix
MacBook-Air:www.tomatoo.com mac$ git branch -d hotfix
Delete branch hotfix (was "fc38ca")
MacBook-Air:www.tomatoo.com mac$ git branch
 * master
```

# Conflitos

- Conflitos são bastante comuns e rotineiros quando existem mais de uma pessoa trabalhando no mesmo arquivo
  - Ambos os arquivos devem ter sido alterados para gerar conflito
  - Se apenas um dos arquivos foi alterado o conflito não ocorrerá
- Devem ser resolvidos antes de terminar a operação sendo executada

# Conflitos

- O Git adiciona marcação(ões) no(s) arquivo(s) conflitante(s) automaticamente

- Indicam os valores conflitantes, que devem ser resolvidos pelo usuário

```
<<<<<< HEAD
```

```
my current code on this branch
```

```
=====
```

```
my other code on the other branch
```

```
>>>>> BRANCH-NAME
```

# Conflitos

- Para resolver o conflito:
  - Apague as mensagens
  - Deixar apenas o código correto
  - Adicione ao “stage” e faça o “commit” do arquivo

## EP: Conflito

1. Crie e alterne para uma nova “branch” de nome “conflict”

```
git checkout -b conflict
```

2. Altere o título do arquivo “index.html” para “ToMaTOo Co v1.0.2”

3. Efetue o “commit” das alterações na “branch conflict”

## EP: Conflito

4. Alterne para a “branch master”
  - a. Perceba que o título voltou para “ToMaTOo Co v1.0.1”  
`git checkout master`
5. Modifique novamente o título do arquivo “index.html” para “ToMaTOo Co v1.0.3”
6. Efetue o “commit” das alterações na “branch master”

## EP: Conflito

7. Ainda na “branch master”, efetue o “merge” com a “branch conflict”
  - a. Visualize a mensagem resultante quando o comando for executado, alertando sobre o conflito e a pendência de resolução  
`git merge conflict`

```
MacBook Air:WWW tomatoo.com mads: git merge conflict
Auto-merging index.html
CONFLICT (content): Merge conflict in index.html.
Automatic merge failed; fix conflicts and then commit the result.
```

## EP: Conflito

8. Verifique o status
  - a. Perceba também a mensagem de conflito e pendência de resolução
9. Abra o arquivo “index.html” no Sublime
10. Identifique os textos adicionados pelo Git no arquivo

```
<<<<< HEAD
<title>ToMaTOo Co v1.0.3</title>
=====
<title>ToMaTOo Co v1.0.2</title>
>>>>> conflict
```

## EP: Conflito

11. Apague as mensagens e mantenha apenas o código com o texto “ToMaTOo Co v1.0.2”
12. Adicione ao “stage” e efetue o “commit”

```
git add index.html
git commit -m "title conflict fix"
```

# EP: Conflito

## 13. Verifique o status

- a. Nenhuma mensagem de conflito ou pendência deve existir
- b. Caso alguma mensagem ainda apareça, volte aos passos anteriores e resolva as pendências

```
git status
```

## 14. Apague o “branch conflict”

```
git branch -d conflict
```

```
git commit -m "title conflict fix"
```

# Versões dos Arquivos

- Quando o Git executa o versionamento uma chave que identifica o “commit” é gerada aleatoriamente
  - A chave geralmente é composta por números e letras
  - Deve ser uma identificação única
- Identifica a versão do “commit”
  - Seria análogo a criar versões 1.0, 1.1, 1.2, 1.n

```
c8b60199bf256173606e2afe5210c6139a5b5338
```

# Versões dos Arquivos

```
MacBook-Air:www.tomatoo.com mac$ git log --chatty=oneline
05e6b42561baee3e630220a8f511c07e89faaf68 title conflict fix
3479a35315684fbcc9a4cfb5525d641f9f531c6f title update to v1.0.3
3924b62561eb8959b083c07e8b677f43cf9e5778 title update to v1.0.2
fc33ccc4c264c17dd7b7c6fba4cf099632dc258a index title update
c5e60199cf256173606e2afe5210c6139a5b5338 Adding initial project files
92c5da9a368bfa15cd177d69a18cd7c284c2dfda Initial commit
```

## git checkout commit-id

- Todas as ações feitas no Git são memorizadas
  - A qualquer momento é possível desfazer uma ação e retornar à uma versão anterior
  - Ainda é possível recuperar e modificar uma determinada versão mais antiga (resolução de “bug”) e continuar trabalhando na atual

# git checkout commit-id

- Para recuperar um “commit” e desfazer todas as suas modificações basta utilizar o comando abaixo
  - Todas as alterações guardadas neste “commit” serão revertidas no projeto
- Arquivos serão adicionados em um nova “branch”

`git checkout commit-id`

# git checkout commit-id

- Após a execução do comando uma mensagem avisando que seu projeto está com os arquivos de uma versão anterior e foi “desconectado de HEAD”
  - Qualquer alteração feita aqui não impactará nos arquivos originais

```
MacBook-Air:www.tomatoo.com mac$ git checkout 3479af
Note: checking out '3479af'.
You are in 'detached HEAD' state. You can work around, make experimental
changes and commit them, and you can discard any commits you make in this
state without impacting any branches by performing another checkout.
```

# git checkout commit-id

- Utilizando mais um parâmetro com o nome do arquivo é possível recuperar apenas um arquivo do “commit”

```
git checkout commit-id file-name
```

## EP: git checkout commit-id

1. Abra o arquivo “index.html” e confirme o valor do título na tag <title>
2. Verifique o “log” do projeto:
  - a. Identifique os “ids” de cada uma das versões e suas mensagens

```
git log --pretty=oneline
```

## EP: git checkout commit-id

3. Copie o começo do “id” do “commit” que altera a versão no título
  - a. A mensagem do “commit” ajudará a identificá-lo
  - b. O valor será diferente do mostrado aqui, pois cada “commit” tem um valor único, basta apenas substituir pelo encontrado em sua máquina

## EP: git checkout commit-id

4. Digite o comando no terminal e cole o valor do “id”
  - a. Não há necessidade de usar o identificador todo, apenas parte dele (geralmente os 6 dígitos iniciais são suficientes)
  - b. No comando abaixo foram colocados todos os dígitos apenas para efeitos didáticos

```
git checkout 3479af35315684fbe9a4cfb5525d641f9f531b6f
```

## EP: git checkout commit-id

5. Verifique o status e as “branches”:

```
git status
```

```
git branch
```

```
MacBook-Air:www.tomatoo.com mac$ git status
HEAD detached at 3479af3
nothing to commit, working directory clean
MacBook-Air:www.tomatoo.com mac$ git branch
* (detached from 3479af3)
 master
```

## EP: git checkout commit-id

6. Volte para a “branch master”

- a. Para manter as alterações é necessário criar uma nova “branch” com os comandos “git checkout -b branch-name” e posteriormente fazer o “merger”

7. Verifique o status e as “branches”:

- a. Perceba que a “branch” criada temporariamente não existe

```
git status
```

```
git branch
```

# git revert

- Permite desfazer um “commit” sem perder o histórico
  - Usado apenas para reverter todo o “commit”
  - Garante mais segurança por preservar o histórico
- Mantém todos os “commits” anteriores e posteriores
  - Traz de volta ao projeto uma determinada versão de arquivo(s)

```
git revert HEAD
```

## EP: git revert

1. Altere o título do arquivo “index.html” para “ToMaTOo Co v1.0.5” e efetue o “commit”:

```
git commit -am "index title update v1.0.5"
```

2. Verifique o “log”
  - a. Identifique a mensagem do último “commit”

# EP: git revert

3. Execute o comando para desfazer o “commit”:

```
git revert HEAD
```

4. Verifique o “log” novamente

- a. Uma mensagem padrão (caso você não a tenha alterado) identifica a reversão
  - b. O arquivo “index.html” foi revertido ao estado inicial antes do “commit”

# CSS3

## Animações e Efeitos

# Transformações

## Alterações Visuais com CSS

### Transformação

- CSS3 traz um conjunto de funções para manipulação de transformações lineares 2D e 3D
- Modifica a posição e forma de um elemento
- Não altera o fluxo normal do documento

# Propriedades CSS

- As propriedades para efetuar a transformação:
  - `transform`
  - `transform-origin`
- Manipula as transformações:
  - `translate`
  - `rotate`
  - `skew`
  - `scale`

## transform

- Possui métodos para executar a transformação
  - `translate()`
  - `rotate()`
  - `scale()`
  - `skew()`
  - `matrix()`

```
transform: method(parameter)
```

# Vendor Provider

- Deve ser usada com os "vendors prefix" para suporte em mais de um navegador:
  - Safari e Chrome: `-webkit-transform: function(parameters);`
  - Firefox: `-moz-transform: function(parameters) ;`
  - Opera: `-o-transform: function(parameters) ;`
  - IE: `-ms-transform: function(parameters) ;`

# Suporte dos Navegadores

- A propriedade "transform" é suportada apenas no IE 10, Firefox e Opera
  - IE 9 suporta o método alternativo (-ms-transform) apenas para 2D
  - Safari e Chrome suportam o método alternativo (-webkit-transform) para 2D e 3D
  - Opera suporta apenas transformações 2D

[http://www.w3schools.com/cssref/css3\\_pr\\_transform.asp](http://www.w3schools.com/cssref/css3_pr_transform.asp)

# rotate

- Rotaciona um elemento no sentido horário sobre a sua origem
- Recebe o parâmetro com o valor em graus do ângulo a ser rotacionado

```
transform: rotate(degrees);
```

## EP: rotate

1. Crie um arquivo de nome "transformations.html"
2. Adicione o seguinte código HTML:

```
<div>I will be transformed</div>
```

## EP: rotate

3. Insira o código CSS:

```
div {
 width:200px;
 height:100px;
 background:salmon;
}
```

## EP: rotate

4. Crie a classe CSS e atribua ao elemento <div>:

```
.rotate {
 -webkit-transform: rotate(7deg);
 -moz-transform: rotate(7deg);
 -ms-transform: rotate(7deg);
 -o-transform: rotate(7deg);
 transform: rotate(7deg);
}
```

# scale

- Escala o elemento em 2D
- Recebe 2 parâmetros
  - sx → eixo X para escala
  - sy → eixo Y para escala
- Se omitido o parâmetro para o eixo Y, será usado o mesmo valor do eixo X

```
transform: scale(sx, sy);
```

## EP: scale

1. Abra o arquivo "transformations.html" e altere o CSS:

- a. Adicione as demais instruções para "vendor prefix"

```
.scale{ transform:scale(2,4); }
```

2. Altere o atributo "class" para "scale" da <div>
3. Salve e teste

# scaleX e scaleY

- Similares a função "scale", porém recebem apenas 1 único parâmetro
  - "scaleX" é similar a instrução "scale(sx, 1)"
  - "scaleY" é similar a instrução "scale(1, sy)"

```
transform: scaleX(sx);
```

```
transform: scaleY(sy);
```

# skew

- Gira o elemento no ângulo especificado
- Recebe 2 parâmetros
  - ax → ângulo X de giro
  - ay → ângulo Y de giro
- Se omitido o parâmetro para o eixo Y, será usado o mesmo valor do eixo X

```
transform: skew(ax, ay);
```

# skew

- Possui as variações "skewX" e "skewY" para modificar um único eixo
  - Regras são análogas a "scaleX" e "scaleY"

## EP: skew

1. Abra o arquivo "transformations.html" e altere o CSS:

- a. Adicione as demais instruções para "vendor prefix"

```
.skew { transform: skew(30deg,20deg); }
```

2. Altere o atributo "class" para "skew" da <div>
3. Salve e teste

# translate

- Move o elemento da posição atual
- Recebe 2 parâmetros
  - tx → distância X do movimento
  - ty → distância Y do movimento
- Pode receber valores numéricos ou em percentuais

`transform: translate(tx, ty);`

# translate

- Possui as variações "translateX" e "translateY" para modificar um único eixo
  - Regras são análogas a "scaleX" e "scaleY"

# EP: translate

1. Abra o arquivo "transformations.html" e altere o CSS:
  - a. Adicione as demais instruções para "vendor prefix"

```
.translate { transform: translate(50px,100px); }
```
2. Crie mais uma <div> como primeira "tag" de <body>
3. Altere o atributo "class" para "translate" da segunda <div>, salve e teste

# Transições

Suavizando a Mudança de Estados

# Mudança de Estado

- Objetos podem ter mais de um estado
  - Ex: tela de login expande quando usuário clica em "Esqueci minha senha"
- Mudança ocorre pela alteração de um estilo para outro
  - Transições (ainda experimental) lidam com o efeito visual durante a mudança de estado
  - Não há necessidade de uso de JS, tudo é feito apenas com CSS

## transition

- Basta especificar a propriedade CSS e a duração
- A duração é:
  - Obrigatória, caso contrário será usado o valor padrão zero, ou seja, a transição não será aplicada
  - Definida em segundos

```
transition:width 2s;
```

# transition

- Possui as seguintes configurações:

```
transition-property: all;
transition-duration: 0s;
transition-timing-function: ease;
transition-delay: 0s;
```

# transition

- Como as demais propriedades CSS, possui a versão "shorthand":  

```
transition: property duration timing-function delay;
```
- Mais de uma transição pode ser aplicada separando as instruções por vírgula  

```
transition:width 2s, height 2s
```

# Vendor Provider

- Deve ser usada com os "vendors prefix selectors" para suporte em mais de um navegador:
  - Safari e Chrome: `-webkit-transition`
  - Firefox: `-moz-transition`
  - Opera: `-o-transition`
  - IE: `-ms-transition`

## EP: transition

1. Crie um arquivo de nome "transitions.html"
2. Adicione um elemento `<div>` em `<body>`
3. Aplique o seguinte estilo:

```
div { width:100px; height:100px; background-color:red;
 transition:width 2s;
 -webkit-transition:width 2s;
 /* add other vendor prefix */
}
```

## EP: transition

4. Utilize a pseudo classe CSS:

```
div:hover {
 width:200px;
}
```

5. Salve e teste
6. Altere o valor da duração para 1 segundo e teste novamente

## EP: transition

7. Crie a transição para altura

- a. Adicione o parâmetro "height" com duração de 1 segundo
  - b. Adicione o valor de "height" de "200px" na pseudo classe

8. Crie a transição para cor de fundo

- a. Adicione o parâmetro "background-color" com duração de 1 segundo
  - b. Adicione o valor de "background-color: blue" na pseudo classe

# EP: transition

9. Adicione a instrução CSS no seletor "div":

```
-moz-transition:width 1s, height 1s, background-color 1s, -moz-transform 1s;

-webkit-transition:width 1s, height 1s, background-color 1s, -webkit-transform 1s;

-o-transition:width 1s, height 1s, background-color 1s, -o-transform 1s;

transition:width 1s, height 1s, background-color 1s, transform 1s;
```

# EP: transition

10. Adicione a instrução CSS na pseudo classe "div:hover":

```
-moz-transform:rotate(180deg) ;

-webkit-transform:rotate(180deg) ;

-o-transform:rotate(180deg) ;

transform:rotate(180deg) ;
```

11. Salve e teste

# Outras Possibilidades

- Existem mais configurações para a transição

[https://developer.mozilla.org/en-US/docs/Web/Guide/CSS/Using\\_CSS\\_transitions?redirectlocale=en-US&redirectslug=CSS%2FTutorials%2FUsing\\_CSS\\_transitions](https://developer.mozilla.org/en-US/docs/Web/Guide/CSS/Using_CSS_transitions?redirectlocale=en-US&redirectslug=CSS%2FTutorials%2FUsing_CSS_transitions)

- Possível também escutar os eventos de uma transição adicionando "listener" ao elemento com efeito
- JS pode enriquecer ainda mais a transição
  - Criar condições mais complexas
  - Aplicar e ou alterar valores em tempo de execução

# Animações

Melhorando a Usabilidade do Usuário

# Sobre

- Objetivo é substituir as animações feitas por "gif", "flash" e JS
- Controle é feito através de CSS
  - Opcionalmente pode ser usado JS para criar condições e funcionalidades mais complexas

# Animação vs Transição

- A animação e transição substituem um estilo por outro de forma gradual
- Diferem pela utilização e aplicação
  - Animação pode ser implementada a qualquer tempo quantas vezes forem necessárias
  - Animação permite criar vários pontos intermediários de estado, enquanto a transição tem apenas dois (inicial e final)

# @keyframes

- "At rule" que define um ponto para a animação
- Possui duas propriedades que definem o estilo inicial e final
  - from { ...css style... }
  - to { ...css style... }

# @keyframes

```
@keyframes myAnimation {
 from { background: red; }
 to { background: yellow; }
}

@-webkit-keyframes myAnimation {
 from { background: red; }
 to { background: yellow; }
}
```

# animation

- Propriedade que integra o seletor CSS com a animação
- Recebe 2 parâmetros
  - Nome da animação
  - Duração em segundos

# animation

```
div {
 animation: myAnimation 5s;
 -webkit-animation: myAnimation 5s;
}
```

## EP: animation

1. Crie um arquivo de nome "animation.html"
2. Adicione um elemento <div>
3. Adicione o seguinte código CSS, salve e teste:

```
div { width: 300px; height:200px;

 animation: myAnimation 5s;

 -webkit-animation: myAnimation 5s; }
```

## EP: animation

```
@keyframes myAnimation {
 from { background: red; }
 to { background: yellow; }
}

@-webkit-keyframes myAnimation {
 from { background: red; }
 to { background: yellow; }
}
```

## 0% a 100%

- As propriedades "from" e "to" podem ser representadas por valores em percentual "0%" e "100%" respectivamente
- Intervalos entre 0 e 100 podem ser inseridos

```
0% { background: red; }
25% { background: yellow; }
50% { background: blue; }
100% { background: green; }
```

## EP: 0% a 100%

1. Abra o arquivo "animation.html"
2. Comente as instruções "from" e "to"
3. Em seus lugares adicione o código, salve e teste

```
0% { background: red; }
25% { background: yellow; }
50% { background: blue; }
100% { background: green; }
```

## EP: 0% a 100%

4. Faça as alterações necessárias para que o estado final seja igual ao inicial
5. Adicione posicionamento relativo à <div>

`position:relative;`

## EP: 0% a 100%

6. Adicione o seguinte código nos "at rules":

```
0% { background:red; left:0px; top:0px; }

25% { background:yellow; left:200px; top:0px; }

50% { background:blue; left:200px; top:200px; }

75% { background:green; left:0px; top:200px; }

100% { background:red; left:0px; top:0px; }
```

## EP: 0% a 100%

7. Altere para repetição:

```
animation: myAnimation 5s linear infinite; -webkit-
animation: myAnimation 5s linear infinite;
```

## EP: 0% a 100%

8. Salve e teste
9. Adicione rotação ao final de cada instrução CSS dos "keyframes":

```
25% { ... -webkit-transform: rotate(90deg) ; }
50% { ... -webkit-transform: rotate(180deg) ; }
75% { ... -webkit-transform: rotate(270deg) ; }
100% { ... -webkit-transform: rotate(360deg) ; }
```

# Propriedades

[http://www.w3schools.com/css/css3\\_animations.asp](http://www.w3schools.com/css/css3_animations.asp)

## API HTML5

Localização e Armazenamento

# Geolocalização

## API de Localização

### Sobre

- Usa as coordenadas de latitude e longitude
- HTML5 possui uma API para acessar as informações
  - Possui métodos e propriedades para identificar o posicionamento
  - Trazem valores aproximados e com precisão acima de 95%, mas não de 100%
  - Necessária autorização do usuário para o compartilhamento de informações de localização

# Obtendo a Localização

- Navegador utiliza os seguintes recursos conforme disponibilidade:
  - GPS
  - Endereço IP
  - WiFi
  - Celular

## GPS

- Disponível em quase todos os celulares modernos
- Dados podem incluir altitude, velocidade e direção
- Dentre as formas usadas caracteriza-se por:
  - Ser o mais preciso
  - Usuário deve estar em local aberto
  - Maior consumo de bateria
  - Pode ser lento

# Endereço IP

- Utiliza um banco de dados externo que mapeia o IP a uma localização física
  - ISP - Internet Service Provider
  - <http://www.ipcatcher.net/>
- Dentre as formas usadas caracteriza-se por:
  - Funciona em qualquer lugar
  - Precisão de cidade (alguns casos bairros)

# WiFi

- Faz a triangulação com 1 ou mais pontos
- Dentre as formas usadas caracteriza-se por:
  - Funciona em ambientes fechados
  - Rápido
  - Geralmente com alta precisão
  - Requer conexão WiFi e idealmente mais de um ponto próximo

# Celular

- Faz a triangulação com 1 ou mais torres de celular
  - Localização e a distância para a torre são consideradas
- Dentre as formas usadas caracteriza-se por:
  - Menos precisa
  - Pode funcionar em ambientes fechados
  - Mais rápido que GPS
  - Depende diretamente da quantidade de torres para precisão

# Escolha do Recurso ?

- Não existe qualquer controle sobre qual recurso será utilizado
- Geralmente o navegador utiliza mais de uma forma
  - Da mais rápida para a mais lenta e precisa
  - Ex: utiliza a triangulação por celular e posteriormente GPS

# navigator

- Objeto que possui informações sobre o navegador do usuário
  - <https://developer.mozilla.org/en-US/docs/Web/API/Navigator>
- Propriedade "geolocation" possui os métodos e propriedades para localização geográfica
  - [https://developer.mozilla.org/en-US/docs/Web/API/Using\\_geolocation](https://developer.mozilla.org/en-US/docs/Web/API/Using_geolocation)
  - <http://www.w3.org/TR/geolocation-API/#navi-geo>

# navigator

Método	Descrição
getCurrentPosition()	Captura do posicionamento atual do usuário Retorna os objetos "position" com as propriedades de latitude e longitude e "positionError" com informações de erro (no caso de ocorrer erro)
watchPosition()	Captura as coordenadas de posicionamento do usuário de modo continuo
clearWatch()	Para a atualização continua feita pelo método "watchPosition"

# getCurrentPosition( )

- Método recebe os seguintes argumentos:
  - Primeiro argumento → função "callback" de sucesso
  - Segundo argumento → função "callback" de erro
  - Terceiro argumento → objeto opcional de configuração
    - <https://developer.mozilla.org/en-US/docs/Web/API/PositionOptions>

```
getCurrentPosition(success, error, options);
```

## getCurrentPosition(success)

- Função "callback" que será invocada quando o método "getCurrentPosition( )" retornar com sucesso
  - Argumento da função será o objeto com os dados de posicionamento

```
function success(position) { ... }
```

# position

- Objeto com as informações de geolocalização
- Possui 2 propriedades "coords" e "timestamp"

```
function success(position) {

 var coords = position.coords;

 var timestamp = position.timestamp;

}
```

## coords

Propriedade	Descrição
latitude / longitude	Valor em graus decimais relativos ao posicionamento sobre um eixo x / y imaginário
altitude	Altura do posicionamento medido em metros
altitudeAccuracy	Valor em metros, com precisão, da coordenada geográfica de altitude

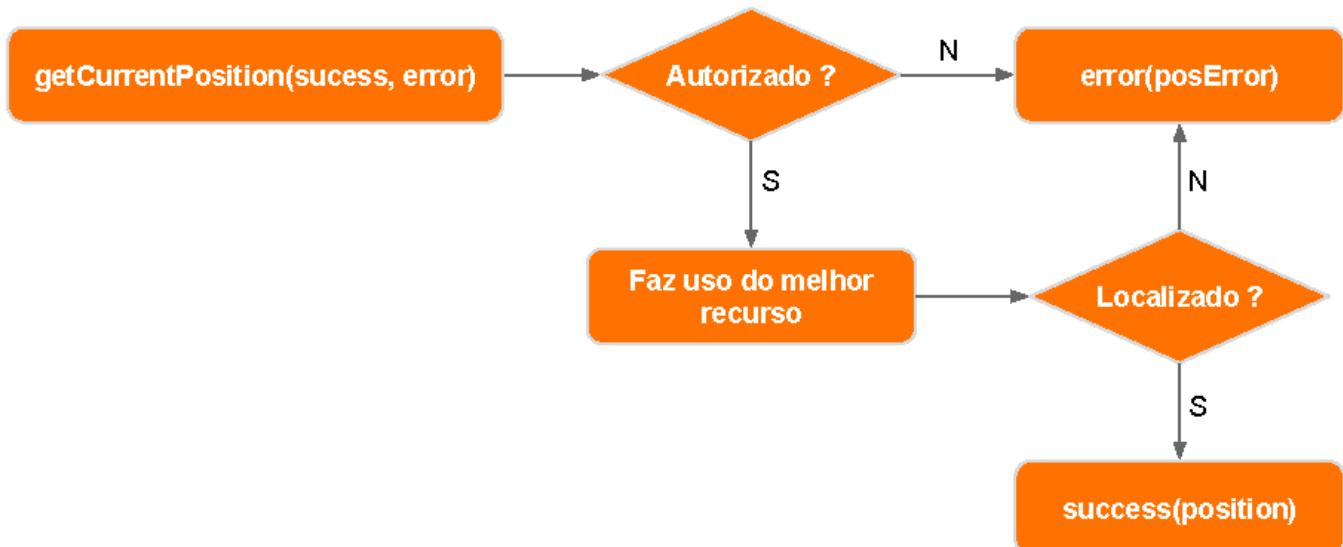
# coords

Propriedade	Descrição
accuracy	Precisão em metros das coordenadas de latitude e longitude
heading	Direção do deslocamento realizado pelo usuário, através do valor de um ângulo que varia de 0 a 360 graus no sentido horário
speed	Velocidade de deslocamento do usuário, medida em metros por segundo

# coords

```
navigator.geolocation.getCurrentPosition(function
 (position) {
 var coords = position.coords;
 var latitude = coords.latitude;
 var longitude = coords.longitude;
 var altitude = coords.altitude;
 var altitudeAccuracy = coords.altitudeAccuracy;
 var accuracy = coords.accuracy;
 var heading = coords.heading;
 var speed = coords.speed;
});
```

# Processo



## EP: position

1. Abra o console do Google Chrome
2. Inspecione o objeto "navigator"
  - a. Verifique os métodos e propriedades
  - b. Identifique a propriedade "geolocation" e os métodos seus disponíveis
3. Digite a seguinte instrução:
  - a. Observe a solicitação de permissionamento

```
navigator.geolocation.getCurrentPosition(
 function(position) {console.dir(position)})
```

<http://www.gps-coordinates.net/>

# EP: position

4. Crie um arquivo de nome "geolocation.html"
5. Adicione o seguinte código:

```
window.onload = function() {
 if (navigator.geolocation) {
 var message = 'Suas coordenadas são:

';
 navigator.geolocation.getCurrentPosition
 (success);
 function success(position) {
 /*add here the next slide code*/
 }
 }
}
```

# EP: position

6. Remova a linha comentada anteriormente e adicione:

```
var latitude = position.coords.latitude;
var longitude = position.coords.longitude;
var altitude = position.coords.altitude;
var altitudeAccuracy = position.coords.
altitudeAccuracy;
var accuracy = position.coords.accuracy;
var heading = position.coords.heading;
var speed = position.coords.speed;
var timestamp = position.timestamp;
```

# EP: position

7. Após a última instrução anterior adicione:

```
message += 'Latitude: '+latitude+'
';
message += 'Longitude:
' +longitude+'
';
message += 'Altitude: '+altitude+'
';
message += 'Direção: '+heading+'
';
message += 'Velocidade: '+speed+'
';
message += 'Tempo: '+timestamp+'
';
```

# EP: position

7. Após a última instrução anterior adicione (cont.):

```
message += 'Precisão da Altitude:
' +altitudeAccuracy+'
';
```

```
message += 'Precisão da Localização:
' +accuracy+'
';
```

# EP: position

8. Após a última instrução anterior adicione:

```
document.body.innerHTML = message;
```

9. Salve e teste no servidor

- a. <http://www.chrome-allow-file-access-from-file.com/>

## getCurrentPosition(succ, error)

- Segundo argumento do método define a função "callback" de erro
  - Argumento da função será o objeto com os dados de erro

```
function error(positionError) { ... }
```

# positionError

- Possui apenas propriedades que identificam o erro
  - code
  - message
- Os códigos da propriedade "code" são definidos pelas constantes do objeto "PositionError"

## PositionError.code

Constante	Código	Descrição
PERMISSION_DENIED	1	Usuário não permitiu o acesso
POSITION_UNAVAILABLE	2	Erro gerado pelo recurso que obtém as informações de geolocalização
TIMEOUT	3	Tempo máximo foi alcançado antes do retorno dos dados

## EP: error

1. Abra o arquivo "geolocation.html"
2. Adicione o segundo argumento ao método:

```
navigator.geolocation.getCurrentPosition(success, error);
```

3. Crie a função "callback" de erro como última instrução do "if":

```
function error(positionError) {
 //add here the next slide code
}
```

## EP: error

4. Remova a linha comentada anteriormente e adicione:

```
switch(positionError.code) {
 case 1: message = 'Permissão para obter posição negada!';
 break;
 case 2: message = 'Erro de conexão!'; break;
 case 3: message = 'Tempo de conexão esgotado!'; break;
 default: message = 'Não foi possível obter os dados!';
}
document.body.innerHTML = message;
```

5. Salve e teste no Google Chrome

# Google Maps

- Google Maps oferece uma API para indentificar no mapa a localização
  - Utiliza as coordenadas de latitude e longitude
  - Possui muitas outras funcionalidades
- Uma vez obtida a localização possível integrar com o serviço Google para visualizar no mapa
  - API Geolocation HTML5 != Google Maps API

# Google Maps

- API permite uso através de chave
  - Pode haver cobrança ou limitações dependendo da utilização

<https://developers.google.com/maps/documentation/javascript/examples/map-simple>

- O mapa é configurado no momento de sua criação
  - Caso necessário, é possível modificar as configurações posteriormente
  - Ex: deslocamento do usuário

## google.maps.LatLng( )

- API cria um objeto global de nome "google"
  - "LatLng" classe que recebe em seu método construtor os parâmetros de latitude e longitude

```
new google.maps.LatLng(-34.397, 150.644);
```

## google.maps.Map( )

- "Map" classe que recebe 2 parâmetros:
  - Primeiro → elemento HTML que conterá o mapa
  - Segundo → parâmetro as configurações

```
new google.maps.Map(
 document.getElementById('container'),
 mapOptions
) ;
```

# google

```
function initializeMap() {
 var mapOptions = {
 zoom: 8,
 center: new google.maps.LatLng(-34.397, 150.644)
 };
 new google.maps.Map(
 document.getElementById('container'),
 mapOptions
);
}
```

## EP: google

1. Crie os arquivos HTML, JS e CSS de nome "maps"
2. Adicione a "tag" <script> com o atributo "src":

<https://maps.googleapis.com/maps/api/js?v=3.1&sensor=false>

3. Adicione uma segunda "tag" <script> para "maps.js"
4. Adicione a <div>:

```
<div id="mapContainer"></div>
```

## EP: google

5. Atribua o arquivo "maps.css" à página HTML
6. Adicione o seguinte código CSS no arquivo "maps.css":

```
html, body, #mapContainer {
 height: 100%;
 margin: 0px;
 padding: 0px
}
```

## EP: google

7. No arquivo "maps.js" adicione o seguinte código:

```
window.onload = function() {
 var lat = -23.5854262, lng = -46.677014299999996;
 var mapContainer = document.getElementById
 ('mapContainer');
 var mapLatLng = new google.maps.LatLng(lat, lng);
 var mapOptions = { zoom: 8, center: mapLatLng };
 var map = new google.maps.Map (mapContainer,
 mapOptions);
}
```

# EP: google

9. Salve e teste
10. Altere o valor do "zoom" para 20, salve e teste novamente

## google.maps.Marker( )

- Classe que recebe em seu método construtor o objeto de configuração da marcação

```
var map = new google.maps.Map(mapContainer,
mapOptions);

var marker = new google.maps.Marker({
 position: myLatlng,
 map: map,
 title: 'Hello World!'
});
```

## EP: Marker( )

1. Abra o arquivo "maps.js" do EP anterior
2. Adicione a seguinte linha de código:

```
var marker = new google.maps.Marker({
 position: mapLatLng,
 map: map,
 title: 'Hello World!'});
```

3. Salve e teste

## Armazenamento Local

Persistindo Dados

# Cookies

- Arquivo de texto armazenado na máquina do usuário
- Permite a gravação e utilização de dados
- Possui uma série de limitações:
  - Tamanho de 45kb por "cookie"
  - 20 "cookies" por domínio e 300 "cookies" por navegador
  - São enviados pelo protocolo HTTP a cada requisição, o que gera em situações de tráfego de rede desnecessário
  - Não são seguros

# Web Storage

- API HTML5 para armazenamento local
- Armazena dados no navegador do usuário
- Como um "cookie", o objeto de armazenamento é único por navegador
  - Se o usuário abrir a página com outro navegador outro objeto existirá
- Dado é armazenado por "chave/valor"

# Web Storage vs Cookie

- Mais seguro e rápido
  - Permite armazenamento de grande quantidade de dados sem comprometer a performance
  - Dado não é transmitido em todas as requisições, apenas quando solicitado
  - Página pode acessar apenas os dados criados por ela

## localStorage

- Armazena dados sem data de expiração
- Dado não é apagado quando o navegador é fechado
- Estará sempre disponível a qualquer tempo

```
localStorage.key = value;
```

# EP: localStorage

1. Abra o console do Google Chrome
2. Digite a seguinte instrução:

```
localStorage.hello = "hello from storage";
```

3. Ainda no Chrome Dev Tools
4. Mude para a aba "Resources"
5. Expanda o item "Local Storage"

# EP: localStorage

6. Identifique a propriedade "hello" e seu valor
7. Crie os arquivo de nome "localStorage.html"
8. Adicione uma "tag" <script> com uma função auto-inicializável
9. Dentro da função crie um objeto conforme a seguir:

```
localStorage.name="John";
localStorage.lastName="Wayne";
```

# EP: localStorage

11. Salve e teste
12. Identifique os valores armazenados na aba "Resources"

## Métodos

- Possível obter e definir um valor apenas utilizando a notação "chave/valor"
- Também estão disponíveis os seguintes métodos:
  - `setItem("key", value)` → define um valor para a chave
  - `getItem("key")` → obtém o valor definido pela chave
  - `removeItem("key")` → remove o item definido pela chave
  - `clear( )` → remove todos os itens armazenados

## EP: Métodos

1. Abra o console do Google Chrome
2. Digite a seguinte instrução:  
`localStorage.setItem("age", 78);`
3. Localize o valor armazenado na aba "Resources"
4. Volte ao console e digite as instruções abaixo, observando os retornos:  
`localStorage.getItem("age");`  
`localStorage.getItem("name");`

## EP: Métodos

5. Digite a seguinte instrução no console:  
`localStorage.removeItem("age");`
6. Verifique se o valor permanece na aba "Resources"
7. Finalmente digite a instrução abaixo no console e verifique os itens armazenados na aba "Resources":  
`localStorage.clear();`

# sessionStorage

- Objeto é bastante similar ao "localStorage"
  - Difere apenas na duração da persistência dos dados
- Armazena e manipula dados enquanto a janela do navegador estiver ativa
  - No momento em que a janela é fechada, os dados são excluídos automaticamente
  - Dados gerados são visualizados apenas na janela que foram criados

## EP: sessionStorage

1. Abra o console do Google Chrome
2. Digite a seguinte instrução:  
`sessionStorage.hello = "hello from this session";`
3. Ainda no Chrome Dev Tools
  - a. Mude para a aba "Resources"
  - b. Expanda o item "Session Storage"
4. Feche a janela e abra novamente
5. Verifique se os ainda existem

# Armazenando Objetos

- O armazenamento funciona apenas com valores do tipo texto
  - Qualquer tipo será convertido para texto
- Este cenário é similar à uma requisição Ajax usando JSON
  - Os dados trafegados são convertidos em texto

# JSON

- Um objeto fornece 2 métodos para converter valores JavaScript para o formato JSON
  - **JavaScript Object Notation**
  - `JSON.stringify()` → serializa um valor JS para o texto JSON
  - `JSON.parse()` → desserializa o texto JSON para JS

## EP: JSON

1. Abra o console do Google Chrome
2. Digite a seguinte instrução e observe o retorno

```
var myNumber = 1;

myNumber;

typeof myNumber;
```

3. Repita o passo anterior, porém acrescente aspas:

```
var myNumber = "1";
```

## EP: JSON

4. Digite a seguinte instrução e novamente observe os valores retornados:

```
var parsed = JSON.parse(myNumber);

parsed;

typeof parsed
```

# EP: JSON

5. Repita o teste com um objeto literal JS
  - a. Crie uma variável chamada myObject
  - b. Adicione 3 propriedades com os seguintes valores
    - ✓ myBoolean: true
    - ✓ myArray: [1,2,3]
    - ✓ myNumber: 1
  - c. Utilize "typeof" e verifique o tipo
  - d. Adicione aspas ao valor do objeto e verifique o tipo
  - e. Aplique "JSON.parse()" e novamente verifique o tipo

# EP: JSON

6. Crie o arquivo JS de nome "utils"

7. Adicione o seguinte código:

```
var utils = {
 parser: /* place next slide code here */}
}
```

## EP: JSON

8. Remova o comentário do passo anterior e adicione:

```
setLocal: function(key, value) {
 /* code placeholder */
},
getLocal: function(key) {
 /* code placeholder */
}
```

## EP: JSON

9. Remova o comentário do passo anterior e adicione:

```
setLocal: function(key, value) {
 localStorage[key]=JSON.stringify(value);
}
```

## EP: JSON

10. Remova o comentário do passo anterior e adicione:

```
getLocal: function(key) {
 return JSON.parse(localStorage[key]);
}
```

## EP: JSON

11. Crie os arquivos JS e HTML de nome "custom"
12. Vincule como primeira "tag" <script> o arquivo "utils.js"
13. Vincule como segunda "tag" <script> o arquivo "custom.js"
14. No arquivo "custom.js" adicione uma função auto-inicializável

## EP: JSON

15. Dentro da função adicione o seguinte código:

```
var preferences = {
 language: "pt-br",
 theme: "sunny",
 newVisitor: true,
 perPage: 100,
 interests: ["politics", "business", "sports"]
};
localStorage.preferences = preferences;
```

## EP: JSON

16. Salve e teste no Google Chrome

- Verifique o valor salvo na aba "Resources"
- Obtenha o valor salvo na aba "Console" e observe seu valor

17. Volte ao código JS e apague a instrução "localStorage"

18. Em seu lugar adicione:

```
utils.parser.setLocal("preferences", preferences);
```

## EP: JSON

19. Salve e teste no Google Chrome
  - a. Verifique o valor salvo na aba "Resources"
  - b. Obtenha o valor salvo na aba "Console" e observe seu valor
20. Ainda no console:
  - a. Utilize a API criada e obtenha o valor salvo como objeto
  - b. Verifique o valor de "storageObject" utilizando "typeof"

```
var storageObject = utils.parse.getLocal
("preferences");
```

## Bibliotecas & Frameworks

Técnicas Avançadas de Desenvolvimento

# Modernizr

Detectando as Capacidades do Navegador

## Sobre

- Biblioteca capaz de identificar suporte disponível do navegador para HTML5 e CSS3
  - Avalia o suporte disponível
  - Não importa qual navegador esteja sendo utilizado
- Cada navegador pode suportar ou não os recursos
  - HTML5 e CSS3 são os mais sensíveis à falta de suporte
  - Quando não suportados, podem comprometer o layout da página

# Características

- Biblioteca de código JavaScript
  - Avalia quais são os recursos disponíveis quando a página carregar
- Não gera nenhuma solução alternativa automaticamente
  - Desenvolvedor deverá prover uma solução
  - Identifica os recursos e dispõe como classes HTML tradicionais
  - Mais de 40 recursos são verificados

# Fallback

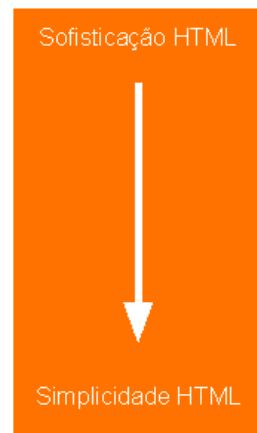
- Caso um recurso não esteja disponível ou solução mais simples deverá ser provida
  - Classes são criadas para <html> automaticamente
  - Através do CSS será possível acessar as classes e prover uma solução alternativa para resolver o problema
- Esta técnica também é conhecida como:
  - Graceful degradation → do sofisticado para o mais simples
  - Progressive enhancement → do simples para o mais sofisticado

# Elegância

Progressive Enhancement



Graceful Degradation



## Condicionais IE

- Muitas vezes o desenvolvedor adiciona condicionais à página HTML para identificar se o navegador é o IE
  - Classes são criadas para posteriormente adicionar uma solução através de CSS
  - Modelo que funciona apenas para IE e não abrange todos os navegadores

<https://msdn.microsoft.com/en-us/library/ms537512%28v=vs.85%29.aspx>

# Condicionais IE

```
<!--[if lt IE 7]>
 <html class="lt-ie9 lt-ie8 lt-ie7">
<! [endif]-->
```

# Condicionais IE

```
<!--[if IE 7]>
 <html class="lt-ie9 lt-ie8">
<! [endif]-->
```

# Condicionais IE

```
<!--[if IE 8]>
 <html class="lt-ie9">
<! [endif]-->
```

# Condicionais IE

```
.lt-ie7 a{
 display: block;
 float: left;
 background: url(drop-shadow.gif);
}
```

# Condicionais IE

```
.lt-ie8 a{
 display: inline-block;
 background: url(drop-shadow.png);
}
```

# Condicionais IE

```
.lt-ie9 a{
 display: inline-block;
 box-shadow: 10px 5px 5px rgba(0,0,0,0.5);
}
```

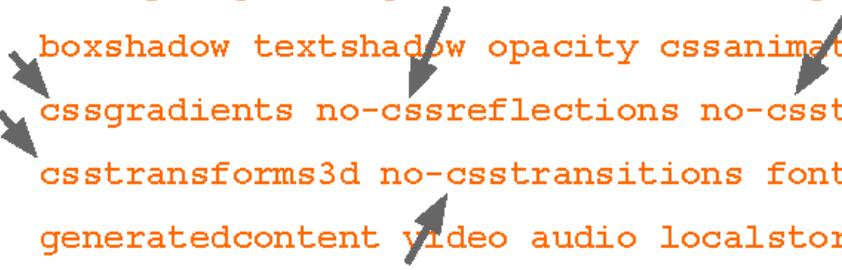
# Utilizando a Biblioteca

- Arquivo JS deve ser adicionado à página
  - Permite configurar quais recursos serão identificados
- Cria classes para <html>
  - Independe de qual navegador está sendo usado
  - Importante são os recursos disponíveis
  - Qualquer recurso não suportado será adicionada a palavra “no-...”

<http://modernizr.com/download/>

## Classes do Modernizr

```
<html class="js flexbox canvas canvastext webgl touch
geolocation postmessage websqldatabase indexeddb
hashchange history draganddrop websockets rgba hsla
multiplebgs backgroundsize borderimage borderradius
boxshadow textshadow opacity cssanimations csscolumns no-
cssgradients no-cssreflections no-csstransforms no-
csstransforms3d no-csstransitions fontface
generatedcontent video audio localstorage sessionstorage
webworkers applicationcache svg inlinesvg smil
- - - - -
```



# Classes do Modernizr

- Uma vez criadas as classes, basta utilizar o seletor CSS adequado
  - Comum o uso do “descendant selector”
  - Qualquer técnica CSS que selecione o elemento adequado poderá ser utilizada

# Classes do Modernizr

```
no-cssgradients .header{
 background-image:url('images/bg-gradient.png');
}

.cssgradients .header{
 background-image: linear-gradient(bottom,
 rgb(13,25,248) 36%, rgb(39,53,255) 68%,
 rgb(67,80,255) 84%);
}
```

# EP: Modernizr

1. Baixe o arquivo no endereço:
  - a. Selecione todas as opções de recursos
  - b. Clique no botão “Generate” e posteriormente em “Download”

<http://modernizr.com/download/>



# EP: Modernizr

2. Crie os arquivos HTML, JS e CSS de nome "modernizr"
3. Vincule o arquivo JS baixado no arquivo HTML
4. Salve e teste
  - a. Perceba as classes criadas em <html>
  - b. Abra a página em outros navegadores
5. Adicione o seguinte código HTML:

```
div#reflect>img[src=http://lorempixel.com/250/375/cats/3]
```

## EP: Modernizr

6. Crie o CSS:

```
#reflect { padding: 10px 0; width: 250px; height: 750px;
margin: 0 auto; }

.touch #reflect { background-color: deeppink; } .no-
touch #reflect { background-color: deepskyblue; }
```

7. Salve e teste

- a. Identifique a classe “no-touch” criada em <html>
- b. Perceba a cor de fundo de fundo aplicada

## EP: Modernizr

8. Adicione o CSS:

```
.cssreflections #reflect img {
-webkit-box-reflect: below 0px -webkit-gradient(linear,
left top, left bottom, from(transparent), color-stop
(50%, transparent), to(white));
}
```

9. Salve e teste

- a. Abra no Chrome e no Firefox

## EP: Modernizr

10. Finalmente complemente o CSS:

```
.no-cssreflections #reflect:after {
 content: ""; display: block;
 background: -moz-element(#reflect) no-repeat;
 width: auto;
 height: 375px;
 margin-bottom: 100px;
 -moz-transform: scaleY(-1);
}
```

## EP: Modernizr

11. Salve e teste

- Repita o teste anterior
- (Opcional) crie uma imagem no Photoshop para criar o efeito correto no Firefox

# Propriedades JS

- Possível utilizar o Modernizr no JS
  - Disponibiliza propriedades que identificam os recursos

```
if (Modernizr.touch) {
 console.log('Touch is supported!');
}
```

# Grid System

# Sobre

- Permite posicionar elementos na tela com base em um layout de grid de colunas
  - Baseado no modelo utilizado por jornais e revistas
  - Alto controle sobre a posição dos elementos na tela
- Forma bastante utilizada da atualidade
  - Implementado por diversas bibliotecas e “frameworks css”

# 4 Tipos

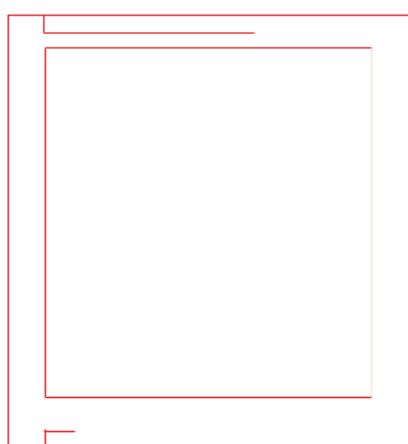
- Manuscrito
- Coluna
- Modular
- Hierárquico

<http://www.vanseodesign.com/web-design/grid-types/>

## 4 Tipos: Manuscrito

- Mais simples
- Estrutura principal é uma grande área retangular
- Chamado de “block grid”
- Feito para grandes quantidades de texto
- Utilização principal para livros

## 4 Tipos: Manuscrito



## 4 Tipos: Manuscrito

**Finding Art in the Sidewalk Cracks**

The Wooster Collective talks to the New York-based street artist Martin Sobey about his work.

About a year ago, we started to notice that the drain drain pipes and scaffolding poles were coming to life with vibrant, energizing colorful works. These are the work of Martin Sobey, an artist whose studio in NoHo is near many of the places he has brought to life. We believe that these small acts of creativity make New York a special city. Passersby know the city is alive and that the human element prevails—even in a construction zone. We caught up with Sobey to ask him about his most recent work.

**WOOSTER:** Why do you choose where you put your pieces?

**MARTIN SOBEY:** My placement depends on a variety of things—initially the piece of architecture or environment that strikes me and is applicable to my type of work—secondarily the location can I make this thing happen?

**W:** How does your work contribute to the community?

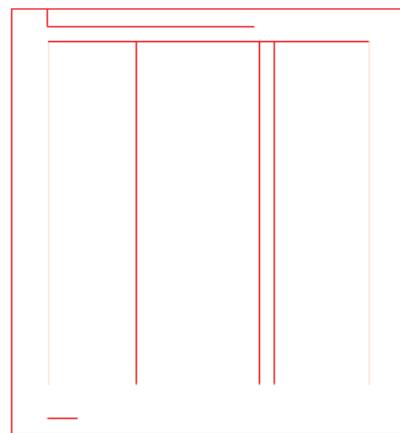
**MS:** My work adds to my community on many levels: it introduces work not of a typical nature, with a greater impact and a stronger aesthetic than the surrounding environment. It also comes down when I down it down—understanding its impermanence is crucial to the piece. Also pride my community has a large number of artists and I strive to maintain a public face of this. Represent, baby! While installing I usually come into contact with people and I get the chance to open a dialogue with neighbors and visitors, a great way to introduce people to my art, public art, and art in general.

**W:** How do people react to the work?

## 4 Tipos: Coluna

- Múltiplas colunas separadas
  - Fluxo de divisão vertical
  - Colunas independentes
  - Mais comum são 12 e 16 colunas, mas existem outras formatações
  - Cada separação entre as colunas é chamado de “gutter”
- Utilizado para informações descontinuadas
  - Elementos variados podem ocupar diferentes áreas
  - Extremamente flexível com relação ao design do layout

## 4 Tipos: Coluna



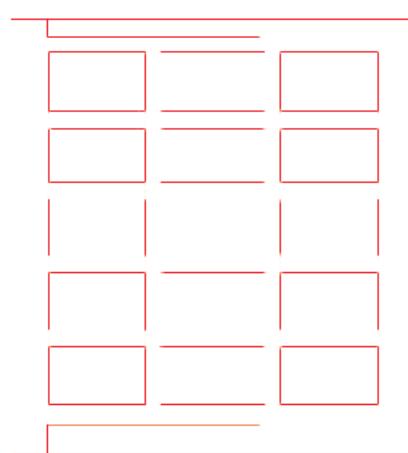
## 4 Tipos: Coluna



## 4 Tipos: Modular

- Similar ao de coluna, porém a adição de linhas
  - Fluxo de divisão vertical e horizontal
  - Define células/módulos
- Utilizado para dados tabulares
  - Ex: gráficos, agendas, etc
- Mais flexíveis e com maior precisão
  - Porém mais complexos de lidar e fazer a manutenção
  - Ex: módulos com 1px / 1px

## 4 Tipos: Modular



# 4 Tipos: Modular

The Grid System is an [article](#), not a guarantee. It permits a number of possible uses and each designer can pick for a solution appropriate to his/her needs. "But you must learn how to use the grid. And it's not something you have."

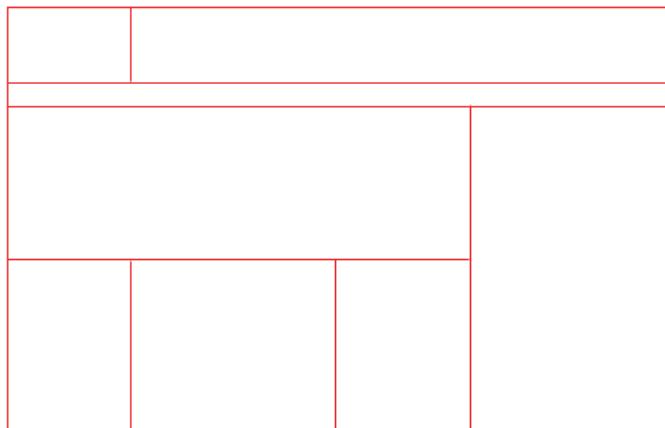
**Jeffrey Milner, Blackadder**

Articles	Tools	Books	Templates	Blog	Inspiration
<a href="#">Applying Mathematics to Web Design</a> Because of its beauty, elegance, mathematics has been a part of art and design for thousands of years. But that doesn't mean it's ignored. <b>02 Mar 2010</b>	<a href="#">#grid</a> A grid that covers a layout grid on web pages, allows you to float it above, and traps elements displaying it in a transparent manner. It is a framework.	<a href="#">Universal Principles of Design</a> Universal Principles of Design is the first comprehensive book on design.	<a href="#">The Golden Grid Template</a> A PSD template based on the Golden Ratio. The Golden Grid by Vladimir Cetkovic.	<a href="#">Grid Based Designs</a> A gallery showcasing some of the most beautiful designs based on a grid.	<a href="#">Art. Art. 178</a> <a href="#">Anodino</a> <a href="#">Athletics</a> <a href="#">BKK</a> <a href="#">Blanka</a> <a href="#">Build</a> <a href="#">Creative Risk Watch</a> <a href="#">Country Photo</a> <a href="#">David Aray</a> <a href="#">Design Assembly</a> <a href="#">Dusty Moose</a> <a href="#">Experimental Artist</a> <a href="#">Fours Fifty Five</a> <a href="#">Grid Magazine</a> <a href="#">Gram Edge</a> <a href="#">Graphic Mag</a> <a href="#">I Love Typography</a> <a href="#">Lamencia</a>
<a href="#">Mystery of Golden Ratio Explained</a> The equation supposedly used to guide the construction the Pyramids	<a href="#">Fluid 960 Grid System</a> A fully-featured fluid grid system template based on Ethan's Grids 500.	<a href="#">Designing for the Web</a> A Practical Guide to Designing for the Web has written explanations.	<a href="#">Photoshop 4 Column Grid</a> A free 4 column Photoshop grid template for a 1024x768 screen resolution.	<a href="#">Buy Slammer = Help Haiti</a> The proceeds from the sale of Slammer as part of the index+Retail collection will	

# 4 Tipos: Hierárquico

- Baseado nos limites de cada elemento
  - Não possui repetição de intervalos
  - Intervalos são definidos pelo conteúdo
- Utiliza espaçamento e margens iguais
- Forma mais tradicional de desenvolvimento na web

## 4 Tipos: Hierárquico



**960**

Grid System

# 960 Grid System

- Framework CSS
  - Biblioteca pré-configurada cujo objetivo é facilitar a criação de layouts o mais próximo possível dos padrões web
- Possui 960px de largura
  - (12 x 20px) + (12 x 60px)

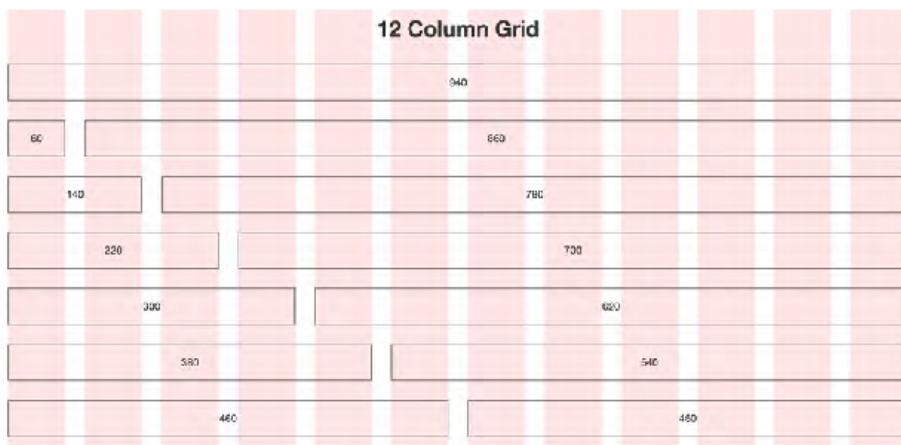
<http://960.gs/>



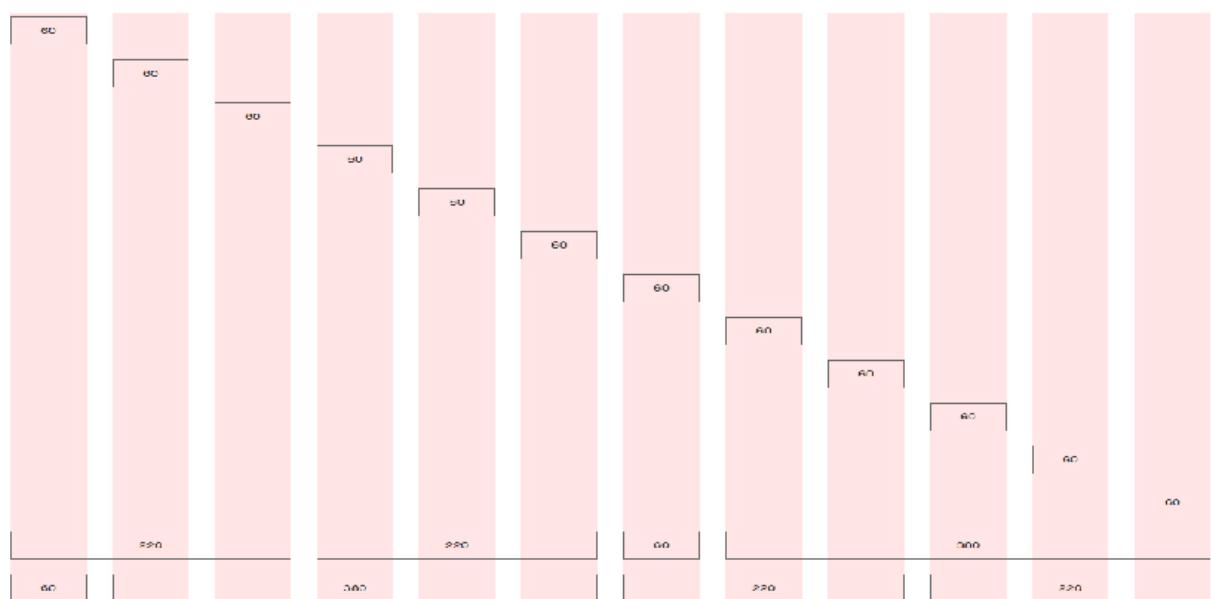
## 12 Colunas

- Modelo mais tradicional
  - Utiliza <div> e o atributo “class” para segmentar as colunas
- Cada coluna possui um tamanho de 60 pixels
- “Gutter” de 20px
  - Espaço entre as colunas
  - 10 pixels para direita e esquerda

# 12 Colunas



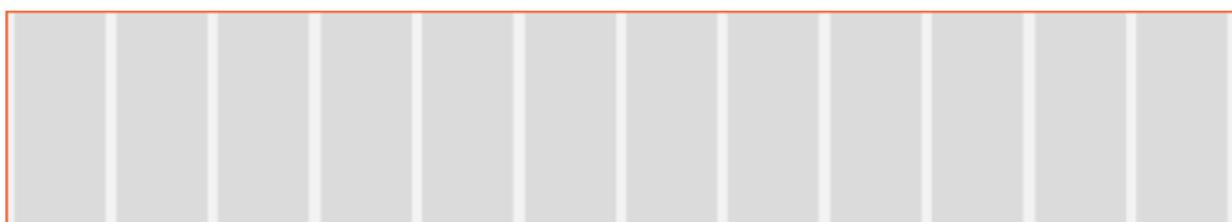
# 12 Colunas



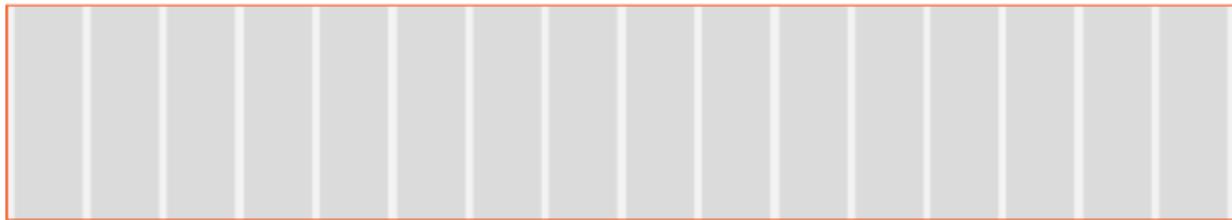
# Containers

- Cada coluna é chamada de contêiner
- Tamanho padrão é 12 ou 16 colunas
  - Ambos resultam em 960px de largura (caso de 16 colunas cada coluna tem menor largura)
  - Outros tamanhos podem ser criados
- Invisíveis no resultado final

# Containers



```
<div class="container_12">...</div>
```

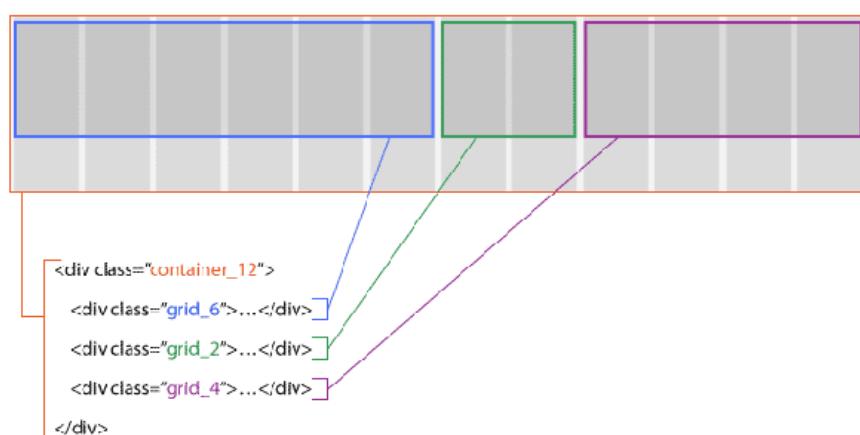


```
<div class="container_16">...</div>
```

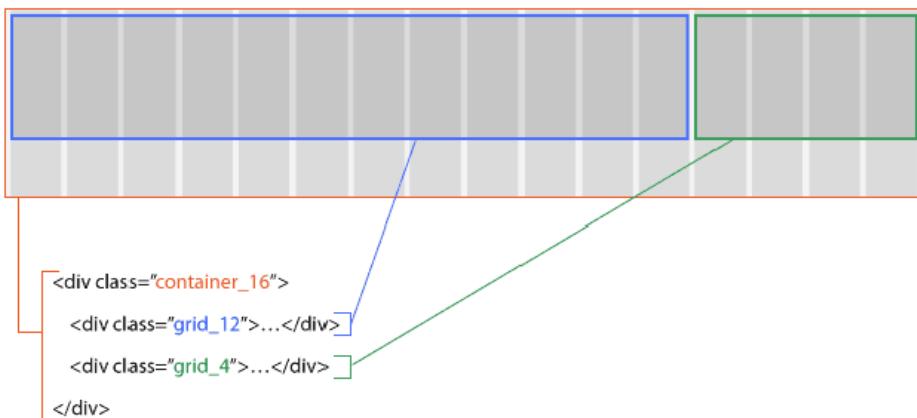
# Grid

- São blocos de conteúdo que utilizam colunas
- Classes identificam as <div>
  - container\_X → Container de X colunas
  - grid\_X → blocos de uma ou mais colunas

# Grid



# Grid



# Grid



# Arquivos no ZIP

- Minificados e não minificados
  - Minificados → “code/css/min/...”
  - Não minificados → “code/css/...”
- Arquivos específicos para 12/16/24 colunas
  - 960\_12\_col.css / 960\_16\_col.css / 960\_24\_col.css
- Genérico para qualquer tamanho
  - 960.css

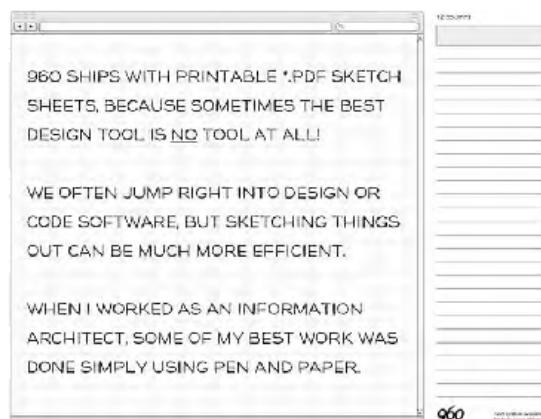
# Arquivos no ZIP

- Exemplos
  - “code/demo.html”
- PDFs para “sketch”
  - “sketch\_sheets/960\_sketch\_all.pdf”
- Plugins para programas de edição de imagens
  - Photoshop → “app\_plugins/phostoshop/960\_GRIDs.atn”
  - Fireworks → “app\_plugins/phostoshop/960.gs Grid Commands.mxp”

# Arquivos no ZIP

- Templates para diversos programas de edição de imagens
  - "templates/..."
- "\_rtl" desenvolvimento para idiomas da direita para esquerda
  - 960\_X\_col rtl.css

## Sketch



# EP: 960 Grid System

1. Navegue até o site abaixo e baixe o arquivo CSS

<http://960.gs/>

2. Extraia os arquivos do pacote “zip”
3. Coloque no projeto o arquivo “960.css”
  - a. Para fins de produção utilize apenas os arquivos minificados

# EP: 960 Grid System

4. Crie os arquivos HTML e CSS de nome “grid”
5. Vincule também os arquivos na ordem abaixo:
  - a. “reset.css”
  - b. “960.css”
  - c. “grid.css”

# EP: 960 Grid System

6. Adicione o seguinte CSS no arquivo “grid-960.css”:

```
.red { background-color: red; }

.green { background-color: green; }

.blue { background-color: blue; }

.red, .green, .blue { height: 768px; }
```

# EP: 960 Grid System

7. Adicione o seguinte código HTML:

```
<div class="container_12">

 <div class="grid_4 red"></div>
 <div class="grid_4 green"></div>
 <div class="grid_4 blue"></div>

</div>
```

8. Salve e teste

# EP: 960 Grid System

9. Altere o código HTML para:

```
<div class="container_12">
 <div class="grid_2 red"></div>
 <div class="grid_6 green"></div>
 <div class="grid_4 blue"></div>
</div>
```

10. Salve e teste

## Grids Fluídos

- O “framework” disponibiliza apenas os arquivos para layout fixo
- Possível obter os arquivos para layout fluído no “link” para customização do layout
  - Colunas serão criadas com base em percentual

<http://grids.herokuapp.com/>

# EP: Grids Fluídos

1. Duplique o arquivo “grid.html” para “fluid.html”
2. Navegue para o endereço abaixo:  
<http://grids.herokuapp.com/>
3. Identifique e clique no “link” de nome “preview”
  - a. Redimensione a página e observe o layout fluído das colunas

[http://grids.herokuapp.com/fluid\\_grid?column\\_amount=12](http://grids.herokuapp.com/fluid_grid?column_amount=12)

# EP: Grids Fluídos

4. Baixe o arquivo no “link” para “download”
  - a. Salve como “fluid-grid.css”
- [http://grids.herokuapp.com/fluid\\_grid.css?column\\_amount=12](http://grids.herokuapp.com/fluid_grid.css?column_amount=12)
5. No arquivo “fluid.html” altere o vínculo do arquivo “960\_12\_col.css” para “fluid-grid.css”
6. Salve e teste
  - a. Redimensione a tela

# Clear

- Entre as “linhas de conteúdos” é necessário utilizar o a propriedade “clear” do CSS
- O “framework” faz uso de técnicas mais robustas
  - Basta utilizar a classe “clear” do próprio “framework”
  - Necessário para resolver alguns “bugs” econtrados em alguns navegadores

## EP: Clear

1. Crie um novo o arquivo “clear.html”
  - a. Vincule os recursos necessários para o 960GS
2. Adicione o seguinte código CSS:

```
div[class*="grid"] { height: 200px; }
.skyblue { background-color: skyblue; }
.tomato { background-color: tomato; }
.graphite { background-color: #333; }
```

## EP: Clear

2. cont.:

```
.purple { background-color: purple; }
.cornsilk { background-color: cornsilk; }
.lime { background-color: lime; }
```

## EP: Clear

3. Adicione o código HTML:

```
<div class="container_12">
 <div class="grid_2 skyblue"></div>
 <div class="grid_4 graphite"></div>
 <div class="grid_2 tomato"></div>
</div>
```

4. Salve e teste

- a. Perceba que a somatória é inferior à 12 colunas

## EP: Clear

5. Adicione novos “grids”:

- a. Mantenha tudo dentro do “container”

```
<div class="grid_4 purple"></div>
<div class="grid_2 cornsilk"></div>
<div class="grid_4 lime"></div>
```

6. Salve e teste

- a. Perceba que a somatória é superior à 12 colunas

## EP: Clear

7. Adicione entre “tomato” e “purple” o “clear”

```
<div class="clear"></div>
```

8. Salve e teste

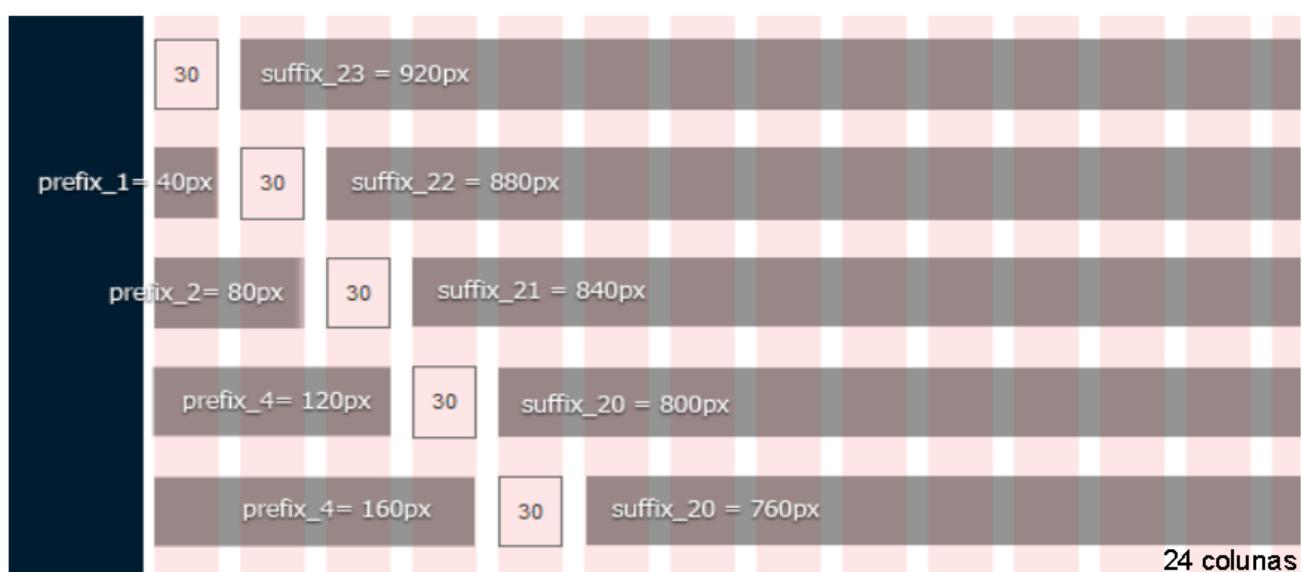
# Prefix & Suffix

- Permite ocupar as colunas anteriores ou posteriores
  - Prefix → anterior
  - Suffix → posterior
- Aumentam a flexibilidade e opções de desenvolvimento
  - Caso o conteúdo seguinte não tenha espaço irá para a linha debaixo

```
<div class="grid_2 prefix_5"></div>
```

```
<div class="grid_2 suffix_10"></div>
```

# Prefix & Suffix



# EP: Prefix & Suffix

1. Crie o arquivo “prefix-suffix”
  - a. Vincule os recursos necessários para o 960GS

2. Adicione o seguinte CSS:

```
div[class*="grid"] {
 height: 200px;
 background-color: #ccc;
}
```

# EP: Prefix & Suffix

3. Adicione o seguinte HTML:

```
<div class="container_12">
 <div class="grid_6">6 Columns</div>
 <div class="grid_4">4 Columns</div>
</div>
```

4. Salve e teste

## EP: Prefix & Suffix

5. Altere o código conforme a seguir:

```
<div class="grid_4 prefix_2">4Columns</div>
```

6. Salve e teste

- a. Perceba que "grid\_4" adicionou mais 2 colunas no início
- b. O texto está localizado exatamente após as 2 colunas adicionais

## EP: Prefix & Suffix

7. Altere o código para:

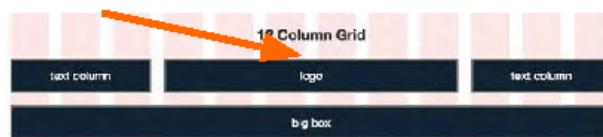
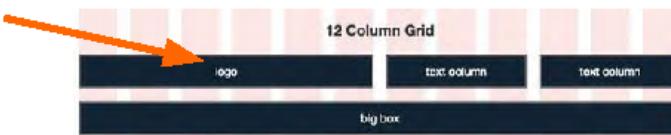
```
<div class="grid_4 suffix_2">4Columns</div>
```

8. Salve e teste

- a. Perceba que "grid\_4" adicionou mais 2 colunas ao final
- b. O texto está localizado no início da coluna

# Pull e Push

- Permitem reposicionar o conteúdo
  - “Puxam” e “empurram” o conteúdo de lugar



## EP: Pull e Push

1. Crie o arquivo “pull-push”
  - a. Vincule os recursos necessários para o 960GS
  - b. Vincule o CSS do EP anterior
2. Adicione o seguinte CSS, salve e teste:

```
<div class="container_12">
 <div class="grid_6">6 Columns</div>
 <div class="grid_3">3 Columns</div>
 <div class="grid_3">3 Columns</div>
</div>
```

# EP: Pull e Push

3. Altere o código da segunda <div> para:

```
<div class="grid_3 push_1">6 Columns</div>
```

4. Salve e teste

- a. Perceba que parte do conteúdo da terceira <div> foi coberto pela da segunda

# EP: Pull e Push

3. Altere o código da segunda <div> para:

```
<div class="grid_3 pull_6">6 Columns</div>
```

4. Salve e teste

- a. Perceba que parte do conteúdo da primeira <div> foi coberto pela da segunda

# EP: Pull e Push

3. Altere o código da primeira <div> para:

```
<div class="grid_6 push_3">6 Columns</div>
```

4. Salve e teste

- a. Perceba ambas as <div> trocaram de lugar

## Links

<http://baselinecss.com/>

<http://www.yaml.de/docs/index.html>

<http://www.blueprintcss.org/>

<http://www.sitepoint.com/understanding-css-grid-systems/>

# Bootstrap

## Bootstrap

- “Framework” mais popular para desenvolvimento HTML
  - Inclui responsividade e “mobile”
- Desenvolvimento rápido e fácil
- Utiliza recursos e arquivos HTML, CSS e JS

<http://getbootstrap.com/getting-started/>



# Bootstrap

- Criado por Mark Otto e Jacob Thornton do Twitter
- “Open source”
- Disponibilizado em Agosto de 2011
- Um dos principais projetos do GitHub



<https://github.com/search?q=stars:%3E1&s=stars&type=Repositories>

# Bootstrap



# Utilização

- Download dos arquivos
  - Bootstrap → compilados
  - Código Fonte → não compilados
  - SASS → gerador de CSS
- CDN

<http://www.bootstrapcdn.com/>



# Download

- Estão incluídos os arquivos compilados e minificados
  - Para produção o ideal é utilizar apenas os minificados
  - Estão caracterizados por possuir o sufixo “min” ao nome do arquivo  
`bootstrap.min.*`
- Os arquivos de fontes e temas são opcionais

<http://getbootstrap.com/components/#glyphicons>

# Download: CSS

```
bootstrap/
└── css/
 ├── bootstrap.css
 ├── bootstrap.css.map
 ├── bootstrap.min.css
 ├── bootstrap-theme.css
 ├── bootstrap-theme.css.map
 └── bootstrap-theme.min.css
```

# Download: JS e Fonts

```
bootstrap/
└── js/
 ├── bootstrap.js
 └── bootstrap.min.js
└── fonts/
 ├── glyphicons-halflings-regular.eot
 ├── glyphicons-halflings-regular.svg
 ├── glyphicons-halflings-regular.ttf
 ├── glyphicons-halflings-regular.woff
 └── glyphicons-halflings-regular.woff2
```

## EP: Download

1. Navegue até o site abaixo e baixe a última versão do Bootstrap:  
<http://getbootstrap.com/getting-started/>
2. Extraia o arquivo do “zip” e adicione a pasta como projeto do Sublime
3. Inspecione os arquivos baixados

## EP: Download

4. Remova o projeto do Sublime

# Exemplos

- O “framewrok” vem com muitos exemplos
  - “Templates” que podem ser customizados conforme necessidade
  - Podem servir de ponto de partida para o desenvolvimento
  - Já possuem responsividade

<http://getbootstrap.com/getting-started/#examples>

# Componentes

- Também são disponibilizados componentes
  - Ex: “slider”, “carousel”, “dropdown menu”, etc

<http://getbootstrap.com/getting-started/#examples-custom>

<http://getbootstrap.com/components/#dropdowns>

# EP: Exemplos

1. Navegue até o site abaixo e baixe o “Source code”:  
<http://getbootstrap.com/getting-started/>
2. Extraia o arquivo do “zip” e adicione a pasta como projeto do Sublime
3. Inspecione os arquivos no Sublime
  - a. Encontre os exemplos na pasta “bootstrap-3.3.4/docs/examples”
  - b. Abra os arquivos usando o Sublime Text

# EP: Exemplos

4. Remova o projeto do Sublime

# CDN

- Outra opção de utilização é através do CDN:

```
<link rel="stylesheet" href="http://maxcdn.bootstrapcdn.com/bootstrap/3.3.4
/css/bootstrap.min.css">

<script src="https://ajax.googleapis.com/ajax/libs/jquery/1.11.1/jquery.min.js"
></script>

<script src="http://maxcdn.bootstrapcdn.com/bootstrap/3.3.4/js/bootstrap.min.js"
></script>
```

# Usando os Arquivos

- Os arquivos estritamente necessários são:
  - bootstrap.min.css
  - bootstrap.min.js
  - jquery.min.js (> 1.9.1)
- Opcionalmente outros arquivos como o tema pode ser adicionado
  - bootstrap-theme.min.css

# Dependência

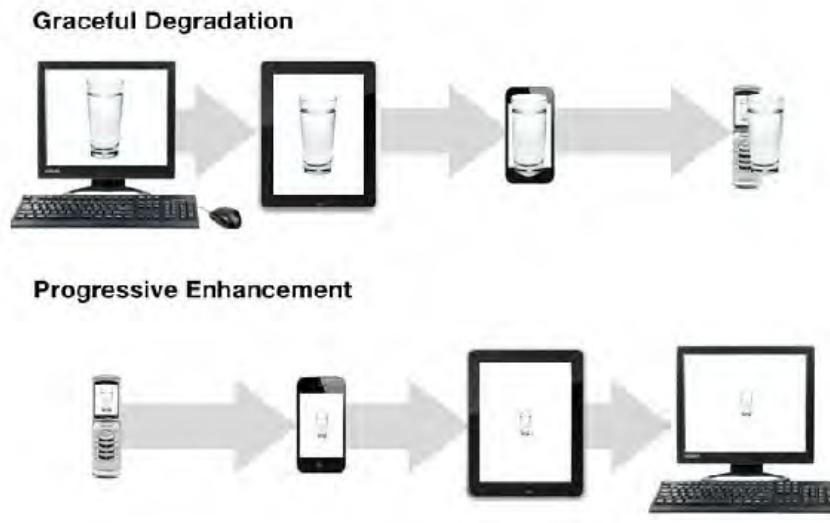
- A única dependência é a biblioteca jQuery superior a versão 1.9.1
- O download do “zip” com os arquivos do Bootstrap não incluem o jQuery
  - Deve ser baixado separadamente ou utilizar CDN

# Normalize

- O “framework” utiliza o “Normalize” para estabelecer os padrões em todos os navegadores
  - Similar ao CSSReset, porém mantém a tipografia das “tags”
  - Margem, “padding”, tamanho de fontes

<http://necolas.github.io/normalize.css/>

# Mobile First



## View Port

- Deve ser configurado para que a exibição seja correta em dispositivos móveis

```
<meta name="viewport"
content="width=device-width, initial-scale=1">
```

# Container

- Todo o conteúdo deve estar dentro de um contêiner
  - Obrigatório para funcionamento correto
  - Já o uso de linhas e colunas é opcional
- Podem existir mais de 2 contêineres “siblings”
  - Não é permitido o aninhamento
  - Idealmente apenas 1 contêiner principal com os arquivos deve ser criado

# Container

- Fixo utiliza a classe “.container”

```
<div class="container">...</div>
```
- Fixo utiliza a classe “.container-fluid”
  - Não possui tamanho fixo de largura
  - Largura será a largura disponível

```
<div class="container-fluid">...</div>
```

# EP: Container

1. Crie e vincule os arquivos HTML e CSS “container”
2. Adicione o seguinte código HTML:

```
<link rel="stylesheet" href="http://maxcdn.bootstrapcdn.com/bootstrap/3.3.4/
css/bootstrap.min.css">

<script src="https://ajax.googleapis.com/ajax/libs/jquery/1.11.1/jquery.min.js"
></script>

<script src="http://maxcdn.bootstrapcdn.com/bootstrap/3.3.4/js/bootstrap.min.js"
></script>
```

# EP: Container

3. Adicione o seguinte HTML:

```
<div class="container"></div>

<div class="container-fluid"></div>
```

4. Adicione o seguinte CSS:

```
.container { background-color: #f00; }

.container-fluid { background-color: #ff0; }
```

# EP: Container

## 4. cont.:

```
.container,.container-fluid {
 height: 200px;
}
```

## 5. Salve e teste

- a. Perceba a diferença entre o fixo e o fluído

# Grid

- Bootstrap utiliza “grid” para posicionamento e layout
  - 12 colunas
  - Pode ter tamanho fixo ou fluído
- Os conceitos de utilização são similares a outros “frameworks” que fazem uso de “grid”

# Grid

- Alguns requisitos e características do “grid”:
  - Obrigatório o uso do contêiner
  - Possui linhas (grupo horizontal de colunas) e colunas
  - Conteúdo deve ser colocado em colunas
  - Colunas criam “gutters”
  - Colunas do “grid” são criadas com o número dentre as 12 disponíveis
  - Se mais de 12 colunas forem colocadas em uma única linha o conteúdo extra irá para a próxima linha

# Grid

<i>Tela</i>	<i>Tamanho do Grid</i>
1200px ou maior	1170px
992px à 1199px	970px
768px à 991px	750px
Menor que 768px	auto

# Colunas

- Classes pré-definidas:

- xs → celulares
- sm → tablets
- md → desktop
- lg → telas grandes

```
<div class="col-sm-4">.col-sm-4</div>
```

# Responsividade

- Possível controlar a responsividade através das classes pré-definidas para as colunas
- Quando atingir o valor são convertidas para colunas, mudando o layout de horizontal para vertical
  - Exceto se utilizada a classe “xs”

# Responsividade

Classe	Significado	Tamanho
.col-xs-\$	Extra Small	< 768px
.col-sm-\$	Small Devices	768px à 991px
.col-md-\$	Medium Devices	992px à 1199px
.col-lg-\$	Large Devices	≥ 1200px

# Responsividade

```
/* Small Devices, Tablets */
@media (min-width : 768px) {...}

/* Medium Devices, Desktops */
@media (min-width : 992px) {...}

/* Large Devices, Wide Screens */
@media (min-width : 1200px) {...}
```

# Linhas

- A classe pré-definida “row” cria a linha
- Agrupa um conjunto de colunas

```
<div class="row">...</div>
```

# Estrutura

```
<div class="container">
 <div class="row">
 <div class="col-md-4">4 Columns</div>
 <div class="col-md-6">6 Columns</div>
 <div class="col-md-2">2 Columns</div>
 </div>
</div>
```

# EP: Estrutura

1. Crie e vincule os arquivos de nome “structure”
  - a. Vincule os arquivos necessários para o Bootstrap
2. Adicione o seguinte HTML utilizando Emmet:

```
.container>(.row.gap>.col-xs-4.item-${xs$}*3)+(.row.gap>.
col-sm-4.item-${sm$}*3)+(.row.gap>.col-md-4.item-${md$}*3)
+(.row.gap>.col-lg-4.item-${lg$}*3)
```

# EP: Estrutura

3. Adicione uma cor de fundo diferente para cada “item”:

```
.gap { margin-top: 50px; }

.item-1 { background-color: plum; }

.item-2 { background-color: aqua; }

.item-3 { background-color: khaki; }
```

# EP: Estrutura

4. Salve e teste
  - a. Inspecione o código e identifique a “media query” de cada classe do Bootstrap
  - b. Verifique os valores dos “breakpoints” de cada “media query”
  - c. Redimensione a tela e perceba a execução do “breakpoint”
  - d. Perceba que a classe “xs” não possui “breakpoint”
5. Altere a classe “container” para “container-fluid”
6. Salve e teste novamente

## Pull & Push

- Também é possível mudar as colunas de lugar utilizando “pull” e “push”

```
<div class="col-md-4 col-md-push-4">
```

# Off-set

- Controla o espaçamento entre as colunas
- Somatória ainda deve resultar em 12

```
<div class="col-md-4">4 Columns</div>

<div class="col-md-4 col-md-offset-4">4 Columns 4 offset</div>

...

<div class="col-md-3 col-md-offset-3">3 Columns 3 offset</div>

<div class="col-md-3 col-md-offset-3">3 Columns 3 offset</div>
```

# Aninhamento

- Layouts complexos podem ser criados aninhando linhas dentro das colunas
  - As linhas aninhadas podem conter outras colunas

# Aninhamento

```
<div class="row">
 <div class="col-sm-6">
 <div class="row">
 <div class="col-sm-6">Nested</div>
 <div class="col-sm-6">Nested</div>
 </div>
 </div>
</div>
```

## EP: Aninhamento

1. Crie e vincule os arquivos de nome “nesting”
  - a. Vincule os arquivos necessários para o Bootstrap
2. Adicione o seguinte HTML utilizando Emmet:

```
.container-fluid>.row>(.col-sm-6.left>h1{Left}+.row>(.col-sm-6.sub-$>h3{ Sub-$ })*2)+.col-sm-6.right>h1{Right}
```

# EP: Aninhamento

3. Crie e vincule os arquivos de nome “nesting”

```
.left {background-color: lightpink; }

.left .sub-1 { background-color: pink; }

.left .sub-2 { background-color: hotpink; }

.right { background-color: skyblue; }
```

4. Salve e teste

## Mixed Columns

- Possível combinar os seletores de colunas menores
  - Colunas responsivas serão criadas
  - Ao invés do layout mudar de horizontal para vertical, o layout ficará sempre na horizontal

```
<div class="col-xs-12 col-sm-6 col-lg-8">
```

## EP: Mixed Columns

1. Crie e vincule os arquivos de nome “responsive-columns”
  - a. Vincule os arquivos necessários para o Bootstrap
2. Adicione o seguinte HTML utilizando Emmet:

```
.container-fluid>(.row>.col-lg-8{8lg}+.col-lg-4{4lg})+(.row>.col-lg-4{4lg}+.col-lg-4{4lg}+.col-lg-4{4lg})
```

## EP: Mixed Columns

3. Adicione o CSS:

```
div[class*="col"] {
 border: 5px solid red;
}
```

4. Salve e teste
  - a. Redimensione a tela e perceba a mudança de horizontal para vertical

## EP: Mixed Columns

5. Adicione as colunas "md":

```
<div class="col-md-6 col-lg-8">6md 8lg</div>
<div class="col-md-6 col-lg-4">6md 4lg</div>
<div class="col-md-8 col-lg-4">8md 4lg</div>
<div class="col-md-2 col-lg-4">2md 4lg</div>
<div class="col-md-2 col-lg-4">2md 4lg</div>
```

6. Salve e teste

## EP: Mixed Columns

7. Adicione as colunas "xs":

```
<div class="col-xs-4 col-md-6 col-lg-8">4xs 6md 8lg</div>
<div class="col-xs-8 col-md-6 col-lg-4">8xs 6md 4lg</div>
<div class="col-xs-3 col-md-8 col-lg-4">3xs 8md 4lg</div>
<div class="col-xs-3 col-md-2 col-lg-4">3xs 2md 4lg</div>
<div class="col-xs-6 col-md-2 col-lg-4">6xs 2md 4lg</div>
```

8. Salve e teste

# Tipografia

- Bootstrap define estilos tipográficos padronizados

<http://getbootstrap.com/css/#type>

- Classes adicionais com estilos próprios estão disponíveis
  - Apelos visuais comuns utilizados no dia a dia

# Tipografia

Classe	Descrição
.lead	Aumenta a fonte do texto
.text-left, .text-center, .text-right, .text-justify	Alinhamento do texto
.text nowrap	Não “quebra” o texto para a próxima linha
.text-lowercase, .text-uppercase, .text-capitalize	Transforma o texto
.list-unstyled	Remove os “bullets” ou números da lista
list-inline	Modifica a exibição da lista de “block” para “inline”

# Imagens

Classe	Descrição
.img-rounded	Adiciona cantos arredondados (não disponível no IE8)
.img-circle	Transforma a imagem para a forma de círculo (não disponível no IE8)
.img-thumbnail	Transforma a imagem para “thumbnail” (contorno envolve a imagem)
.img-responsive	Escala a imagem ao “parent element”

## EP: Imagens

1. Crie e vincule os arquivos de nome “image”
  - a. Vincule os arquivos necessários para o Bootstrap
2. Adicione o seguinte HTML utilizando Emmet:

```
.container-fluid>(.row.row-$>(.col-xs-3>img[src=http://lorempixel.com/300/300/people/$@6])*4)*4
```

# EP: Imagens

3. Adicione o seguinte CSS:

```
.row-1 { background-color: salmon; }
.row-2 { background-color: gold; }
.row-3 { background-color: skyblue; }
.row-4 { background-color: firebrick; }
```

4. Salve e teste

- a. Redimensione a tela e observe o comportamento das imagens

# EP: Imagens

5. Adicione as seguintes classes em cada imagem:

- a. .img-rounded
  - b. .img-circle
  - c. .img-thumbnail
  - d. .img-responsive

6. Salve e teste

- a. Redimensione a tela e observe o comportamento das imagens

# EP: Imagens

7. Acrescente a classe “img-responsive” à classe “img-circle”:

```

```

8. Salve e repita o teste anterior

9. (Opcional) Altere a classe de “col-xs-3” para “col-md-3” e teste novamente

## Jumbotron

- “Callout” especial com fundo cinza, cantos arredondados e com fonte de tamanho grande
  - Caixa para chamar atenção à algum conteúdo ou informação
  - A fonte é aumentada proporcionalmente conforme a tag
- A classe “.jumbotron” define a caixa para o texto

```
<div class="jumbotron">...</div>
```

# Jumbotron

The screenshot shows a "EXAMPLE" section of the Bootstrap documentation. It features a large, light-gray "hero unit" with the text "Hello, world!" in a large font. Below the title is a smaller paragraph: "This is a simple hero unit, a simple jumbotron-style component for calling extra attention to featured content or information." At the bottom of the hero unit is a blue "Learn more" button. Below the hero unit, the HTML code for creating this component is displayed:

```
<div class="container">
 <div class="jumbotron">
 <h1>Hello, world!</h1>
 <p>...
```

## EP: Jumbotron

1. Crie e vincule os arquivos de nome “jumbotron”
  - a. Vincule os arquivos necessários para o Bootstrap
2. Adicione o seguinte HTML utilizando Emmet:

```
.container-fluid>(.row>.col-xs-12>div>h${Header $ } *3+p>lorem) *2
```

# EP: Jumbotron

3. Adicione a classe “.jumbotron”:

- a. Apenas na primeira <div>, deixe a segunda <div> sem a classe

```
<div class="col-xs-12">

 <div class="jumbotron">

 <h1>Header 1</h1> . . .
```

4. Salve e teste

# Page Header

- Cabecalho de página
- Adiciona uma linha
  - Com espaço adicional entre a linha e o texto
- A classe “.page-header” configura a <div> que conterá o texto

```
<div class="page-header">...</div>
```

# EP: Page Header

1. Crie e vincule os arquivos de nome “page-header”
  - a. Vincule os arquivos necessários para o Bootstrap
2. Adicione o seguinte HTML utilizando Emmet:

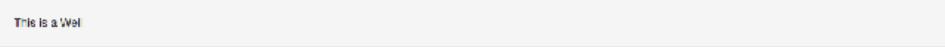
```
.container-fluid>(.row>.col-md-12>.page-header>h1 {Page
Header}) + (.row>.col-md-12>p*10>lorem*5)
```

3. Salve e teste

## Well

- Similar ao “jumbotron”, porém sem alterar o tamanho do texto
- Classe “.well” define o contêiner para o conteúdo

```
<div class="well">...</div>
```



# Well

- Possui 2 opções adicionais de tamanhos:

- well-sm
  - well-lg

```
<div class="well well-sm">...</div>
```

```
<div class="well well-lg">...</div>
```

## EP: Well

1. Crie e vincule os arquivos de nome “well”
  - a. Vincule os arquivos necessários para o Bootstrap
2. Adicione o seguinte HTML utilizando Emmet:

```
.container>(.well.well-lg>p>lorem) + (.well>p>lorem) + (.well.well-sm>p>lorem) + (div>p>lorem)
```

3. Salve e teste
  - a. Identifique na tela a <div> com a classe “well” e observe as diferenças

# Contextual

- Existem as seguintes classes contextuais
  - `*-success` → verde
  - `*-info` → azul
  - `*-warning` → amarelo
  - `*-danger` → vermelho
- Maioria das classes do Bootstrap tem estas variações
  - São classes secundárias que devem ser adicionadas à principal

# Alert

- Visualmente similares ao “.well” porém coloridos
  - Cada cor é definida pela classe de contexto

```
<div class="alert alert-success">...</div>

<div class="alert alert-info">...</div>

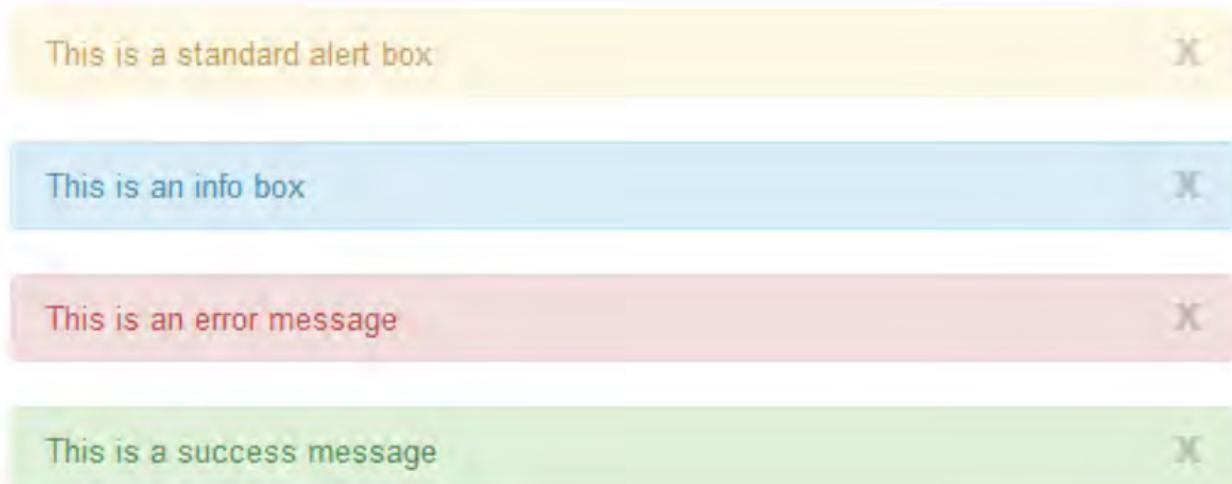
<div class="alert alert-warning">...</div>

<div class="alert alert-danger">...</div>
```

# Alert & Close

- Opcionalmente pode ser adicionado o botão fechar
  - O código JS para a ação faz parte do arquivo “bootstrap.min.js”
  - Pode também ser baixado e implementado separadamente como um “plugin” através do arquivo “alert.js” disponibilizado pelo Bootstrap
- Deve ser usada `<a>` configurada adequadamente para que o botão funcione
  - Usar a classe “close” e o atributo “data-dismiss” com valor “alert”

# Alert & Close



# Alert & Close

```
<div class="alert alert-success">
 <a href="#" class="close"
 data-dismiss="alert">×
</div>
```

## EP: Alert & Close

1. Crie e vincule os arquivos de nome “alert”
  - a. Vincule os arquivos necessários para o Bootstrap
2. Adicione o seguinte HTML utilizando Emmet:

```
.container>(.alert>a.close[data-dismiss=alert href=#]
×)+p>lorem10)*4
```

3. Salve e teste
  - a. Perceba que não existe cores para o contêiner

# EP: Alert & Close

4. Adicione as seguintes classes em cada um dos “alerts”:
  - a. alert-success
  - b. alert-info
  - c. alert-warning
  - d. alert-danger
5. Salve e teste
  - a. Identifique a diferença de cada um dos contêineres

# Buttons

- Classe principal é “.btn”
- Existem ainda 7 classes com estilos diferentes
  - .btn-default
  - .btn-primary
  - .btn-success
  - .btn-info
  - .btn-warning
  - .btn-danger
  - .btn-link



# Buttons

```
<button class="btn btn-default">Default</button>
<button class="btn btn-primary">Primary</button>
<button class="btn btn-success">Success</button>
<button class="btn btn-info">Info</button>
<button class="btn btn-warning">Warning</button>
<button class="btn btn-danger">Danger</button>
<button class="btn btn-link">Link</button>
```

# Configurações

- Tamanhos diferentes
  - .btn-lg
  - .btn-md
  - .btn-sm
  - .btn-xs

# Configurações



# Configurações

```
<button class="btn btn-default btn-lg">Default</button>
<button class="btn btn-primary btn-md">Primary</button>
<button class="btn btn-success btn-sm">Success</button>
<button class="btn btn-info btn-xs">Info</button>
```

# Configurações

- Exibição “block”
  - .btn-block
- Ativar/desativar
  - .active (visual de pressionado)
  - .disabled

# Configurações

```
<button class="btn btn-info btn-xs active">Info</button>
<button class="btn btn-danger disabled">Danger</button>
<button class="btn btn-link btn-block">Link</button>
```

# Elementos

- As classes de botão podem ser utilizadas com os elementos (todos tem o mesmo visual):
  - <a>
  - <button>
  - <input>

## EP: Buttons

1. Crie e vincule os arquivos de nome “buttons”
  - a. Vincule os arquivos necessários para o Bootstrap
2. Adicione o seguinte HTML utilizando Emmet:

```
.container>button.btn{Button $}*7
```
3. Salve e teste

# EP: Buttons

4. Acrescente as seguintes classes:

- a. .btn-default
- b. .btn-primary
- c. .btn-success
- d. .btn-info
- e. .btn-warning
- f. .btn-danger
- g. .btn-link

5. Salve e teste

# EP: Buttons

6. Escolha 3 botões aleatoriamente e adicione as seguintes classes

- a. .btn-block
- b. .active (visual de pressionado)
- c. .disabled

7. Salve e teste

## EP: Buttons

8. Escolha 4 botões aleatoriamente e adicione as seguintes classes
  - a. .btn-lg
  - b. .btn-md
  - c. .btn-sm
  - d. .btn-xs
9. Salve e teste

## EP: Buttons

10. Crie 2 novos botões utilizando os elementos <a> e <input>:

- a. Coloque dentro da <div> “container”:

```
input:button.btn.btn-warning[value=Warning]+a.btn.btn-
danger[href=#] {Danger}
```

11. Salve e teste

# Button Group

- Barra com agrupamento de botões
  - <div> com a classe “btn-group” que conterá os botões
- Bastante útil para criar menus
  - Pode ser definida na horizontal ou vertical
  - Permite ainda criar sub-menus

# Button Group

```
<div class="btn-group">
 <button class="btn btn-primary">Button 1</button>
 <button class="btn btn-primary">Button 2</button>
 <button class="btn btn-primary">Button 3</button>
</div>
```

# Button Group

- O tamanho pode ser definido pelas classes:

- o btn-group-lg
- o btn-group-sm
- o btn-group-xs



Left Middle Right

# Button Group

```
<div class="btn-group btn-group-lg">
 <button class="btn btn-default">Default</button>
 <button class="btn btn-primary">Primary</button>
 <button class="btn btn-warning">Warning</button>
</div>
```

# EP: Button Group

1. Crie e vincule os arquivos de nome “group-buttons”
  - a. Vincule os arquivos necessários para o Bootstrap
2. Adicione o seguinte HTML utilizando Emmet:

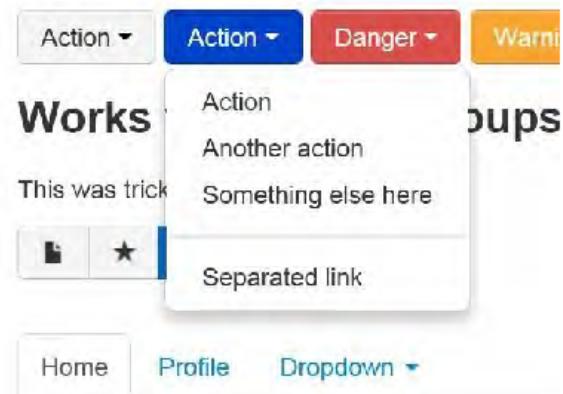
```
.container>(.col-md-12.btn-group.btn-group-lg>button.btn.btn-primary{Button$}*3)+(.col-md-12.btn-group>button.btn.btn-primary{Button$}*3)+(.col-md-12.btn-group.btn-group-sm>button.btn.btn-info{Button$}*3)+(.col-md-12.btn-group.btn-group-xs>button.btn.btn-warning{Button$}*3)+(.col-md-2.btn-group-vertical>button.btn.btn-success{Button$}*3)
```

3. Salve e teste

# Dropdown & Dropdown

- Possível criar sub-itens dentro de um botão
  - Pode ser um único botão ou uma barra de botões
- Existem duas classes para criar o componente:
  - .dropdown → “popup” exibido abaixo do botão
  - .dropdown → “popup” exibido acima do botão

# Dropdown & Dropup



# Dropdown & Dropup

- Configuração adequada requer:
  - Botão principal que mostrará sub-itens
    - Deve ser configurado com a classe "dropdown-toggle" e a propriedade "data-toggle" com o valor "dropdown"
  - "Caret" indicando a existência de sub-itens
    - <span> com a classe "caret"
  - Sub-itens, <ul> com <li>
    - <ul> deve utilizar a classe ".dropdown-menu"

# Dropdown & Dropdown

```
<div class="btn-group">

 <button type="button" class="btn btn-primary dropdown-toggle" data-
 toggle="dropdown">Main Button
 </button>

 <ul class="dropdown-menu">
 Sub-Item 1
 Sub-Item 2

</div>
```

# Dropdown & Dropdown

- Um divisor pode ser adicionado entre os sub-itens

```
<ul class="dropdown-menu">
 Sub-Item 1
 <li class="divider">
 Sub-Item 1

```

# EP: Dropdown & Dropup

1. Crie e vincule os arquivos de nome “dropdown”

- a. Vincule os arquivos necessários para o Bootstrap

2. Adicione o seguinte HTML utilizando Emmet:

```
.container>.dropdown>(button.btn.btn-danger.dropdown-
toggle[data-toggle=dropdown]{Dropdown}>span.caret)+ul.
dropdown-menu>(li*3>a[href=#]{Sub-Item$})+(li.divider)+
(li*3>a[href=#]{Sub-Item$@4})
```

3. Salve e teste

# EP: Dropdown & Dropup

4. Escolha alguns dos sub-itens aleatoriamente e adicione a classe “disabled”
5. Salve e teste

# Glyphicon

- Bootstrap disponibiliza ícones no formato de fontes
  - 250 ícones diferentes
  - Podem ter tamanhos e cores com base em uma instrução CSS
- Podem ser visualizados no endereço abaixo:  
<http://getbootstrap.com/components/#glyphicons>

GLYPHICONS.com

# Glyphicon

- O elemento `<span>` com a classe identificado o “glyphicon” exibe o ícone na tela
  - Não podem ser usadas com nenhum outro elemento e a “tag” não deve conter texto

```

```

# Glyphicon



## EP: Glyphicons

1. Crie e vincule os arquivos de nome “glyphicon”
  - a. Vincule os arquivos necessários para o Bootstrap
2. Adicione o código Emmet:

<http://pastebin.com/dEQiQJT0>

3. Adicione o CSS:

```
.glyphicon {font-size:50px;padding:50px;}
```

4. Salve e teste

# Badges

- Indicadores numéricos
- Classe “.badge” com `<span>` define o componente

```
Messages 3
```

Messages 8

# Badges

- `<span>` pode ser adicionado dentro de outras “tags” como um botão

```
<button class="btn btn-primary">
 Inbox 8
</button>
```

Inbox 8

# Labels

- Similares aos “badges”, porém apropriados à textos

```
Label
```

- Tamanho definido conforme a “tag” que o contiver

```
<h3>
```

```
 Header Label
```

```
</h3>
```

# Labels

```
<h1> Label
<h2> Label
<h3> Label
<h4> Label
<h5> Label
<h6> Label
```

# Labels

- Podem ser configurados com o seguinte contexto:
  - .label-default
  - .label-primary
  - .label-success
  - .label-info
  - .label-warning
  - .label-danger

**Default**   **Primary**   **Success**   **Info**   **Warning**   **Danger**

# Labels

```
<h1> Default
<h2> Primary
<h3> Success
<h4> Info
<h5> Warning
<h6> Danger
```

# EP: Labels

1. Crie e vincule os arquivos de nome “labels”
  - a. Vincule os arquivos necessários para o Bootstrap
2. Adicione o seguinte HTML utilizando Emmet:

```
.container>(h1{<h1>}>span.label.label-default{Default})+(h2{<h2>}>span.label.label-primary{Primary})+(h3{<h3>}>span.label.label-success{Success})+(h4{<h4>}>span.label.label-info{Info})+(h5{<h5>}>span.label.label-warning{Warning})+(h6{<h6>}>span.label.label-danger{Danger})
```

3. Salve e teste

# Progress Bar

- A classe “progress” para a <div> contêiner define a barra
  - A propriedade CSS “width” define o tamanho da barra
- A classe “progress-bar” para a <div> aninhada define o progresso
  - A propriedade CSS “width” define o tamanho do progresso

# Progress Bar

```
<div class="progress">
 <div class="progress-bar"
 style="width:10%">
 </div>
</div>
```

# Progress Bar

- Existem 5 tipos diferentes de barra de progresso:
  - Básica com ou sem “label”
  - Contextuais (4)
  - Listradas (“striped”)
  - Com animação ( $\geq$ IE9)
  - Empilhadas (“stacked”)

# Básica

Básica sem “label”



Básica com “label”



## Básica com “Label”

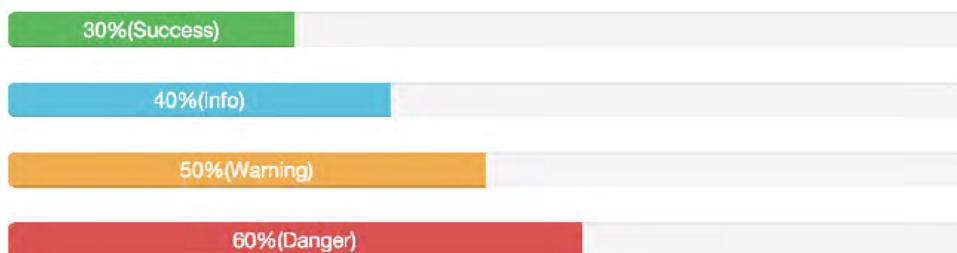
- Basta adicionar texto ao conteúdo de <div> com a classe “progress-bar”

```
<div class="progress">
 <div class="progress-bar" style="width:10%">
 10%
 </div>
</div>
```

# Contextual

- As classes definem as cores:
  - .progress-bar-success
  - .progress-bar-info
  - .progress-bar-warning
  - .progress-bar-danger

# Contextual



# Coloridas

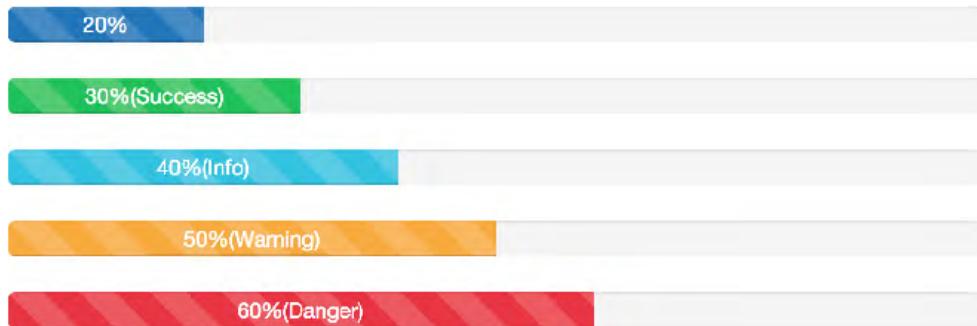
```
<div class="progress">
 <div class="progress-bar progress-bar-success"....
 <div class="progress-bar progress-bar-info"....
 <div class="progress-bar progress-bar-warning"....
 <div class="progress-bar progress-bar-danger"....
</div>
```

# Listradas

- A classe “progress-bar-striped” define a barra listrada

```
<div class="progress">
 <div class="progress-bar progress-bar-striped"....
</div>
```

# Listradas



## Com Animação

- Disponível apenas para IE9 ou superior
- A classe “active” deve ser adicionada à classe “progress-bar-striped”

```
<div class="progress">
 <div class="progress-bar progress-bar-striped active"...>
</div>
```

# Empilhadas

- Múltiplas barras devem ser aninhadas dentro de uma única <div> contêiner
  - <div> com a classe “progress”
- Somatória do andamento deve ser menor ou igual à 100%

# Empilhadas

```
<div class="progress">
 <div class="progress-bar progress-bar-success"
 style="width:30%">30% (Success)</div>
 <div class="progress-bar progress-bar-info"
 style="width:40%">40% (Info)</div>
</div>
```



# EP: Progress Bar

1. Crie e vincule os arquivos de nome “progress-bar”
  - a. Vincule os arquivos necessários para o Bootstrap
2. Adicione o seguinte HTML utilizando Emmet:

# EP: Progress Bar

```
.container>(.progress>.progress-bar[style=width:10%])+(.progress>.progress-
bar.progress-bar-striped[style=width:20%]{20%})+(.progress>.progress-bar.
progress-bar-success.progress-bar-striped.active[style=width:30%]{30%
(Success)})+(.progress>.progress-bar.progress-bar-info.progress-bar-
striped.active[style=width:40%]{40%(Info)})+(.progress>.progress-bar.
progress-bar-warning.progress-bar-striped.active[style=width:50%]{50%
(Warning)})+(.progress>.progress-bar.progress-bar-danger.progress-bar-
striped.active[style=width:60%]{60%(Danger)})+(.progress>.progress-bar.
progress-bar-success[style=width:30%]{30%(Success)})+.progress-bar.progress-
bar-info[style=width:40%]{40%(Info)})
```

3. Salve e teste

# Pagination

- Paginação pode ser criada com o uso de <ul> e a classe “pagination”

```
<ul class="pagination">
 1
 2
 3

```



# Pagination

- A classe “active” define qual elemento <li> está ativo/selecionado

```
<ul class="pagination">
 1
 <li class="active">2
 3

```



# Pagination

- Pode ser desativado com a classe “disabled” em <li>
- Tamanho padrão também pode ser alterado com o uso adicional das classes:
  - .pagination-lg
  - .pagination-sm

# Pager

- Permite criar paginação “próxima” e “anterior”

```
<ul class="pager">
 Próximo
 Anterior

```

[Próximo](#)[Anterior](#)

# Pager

- Podem ser alinhados separadamente com o uso das classes “previous” e “next” em <li>

```
<ul class="pager">
 <li class="previous">Próximo
 <li class="next">Anterior

```

[Próximo](#)[Anterior](#)

# List Group

- Listas para exibição de conteúdos complexos
  - A classe “list-group” em <ul> define o contêiner
  - Já a classe “list-group-item” em <li> define o item

Item 1
Item 2
Item 3

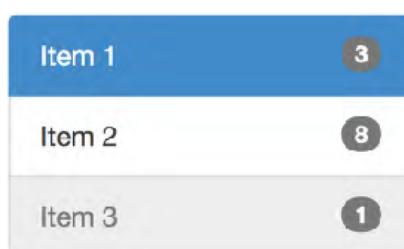
# List Group

```
<ul class="list-group">
 <li class="list-group-item">Item 1
 <li class="list-group-item">Item 2
 <li class="list-group-item">Item 3

```

# List Group

- A classe “active” define o item ativo/selecionado
  - Um `<span>` com a classe “badge” pode ser usada em conjunto
  - A classe “disabled” desabilita um dos itens da lista



# List Group

```
<ul class="list-group">
 <li class="list-group-item active">
 Item 1 3

 <li class="list-group-item">
 Item 2 8

 <li class="list-group-item disabled">
 Item 3 1


```

# List Group

- Também é possível utilizar `<div>` e `<a>`, eliminando os elementos `<ul>` e `<li>`

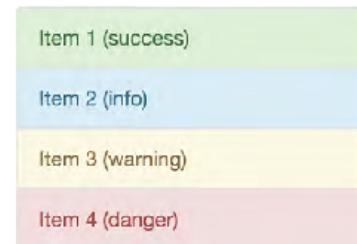
- Basta fazer uso das mesmas classes

```
<div class="list-group">
 Item 1
 Item 2
 Item 3
</div>
```

# List Group

- Existem as seguintes classes contextuais:

- .list-group-item-success
- .list-group-item-info
- .list-group-item-warning
- .list-group-item-danger



# List Group

```
<div class="list-group">
 Item 1
 Item 2
 Item 3
 Item 4
</div>
```

# List Group

- Textos podem ser adicionados na lista fazendo uso da classe “list-group-item-text”



# List Group

```
<div class="list-group">

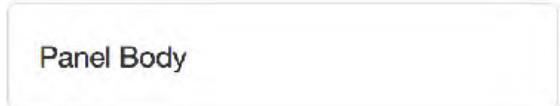
 <h4 class="list-group-item-heading">Header</h4>
 <p class="list-group-item-text">...</p>

</div>
```

# Panel

- Cria painel de conteúdo
  - Classe “panel” define o container

```
<div class="panel panel-default">
 <div class="panel-body">Panel Body</div>
</div>
```



Panel Body

# Panel

- Pode conter corpo, “header” e “footer”
  - .panel-heading
  - .panel-body
  - .panel-footer
- O título pode ser definido com a classe “panel-title”

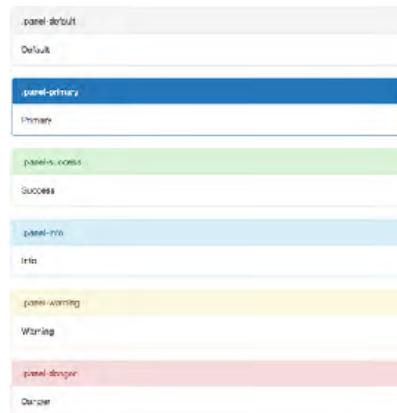


# Panel

```
<div class="panel panel-default">
 <div class="panel-heading">
 <h3 class="panel-title">Panel Title</h3>
 </div>
 <div class="panel-body">Panel Body</div>
 <div class="panel-footer">Panel Footer</div>
</div>
```

# Panel

- Possui as classes de contexto:
  - .panel-default
  - .panel-primary
  - .panel-success
  - .panel-info
  - .panel-warning
  - .panel-danger



# EP: Panel

1. Crie e vincule os arquivos de nome “panel”
  - a. Vincule os arquivos necessários para o Bootstrap
2. Adicione o HTML utilizando Emmet, save e teste:

```
.container>(.panel.panel-default>.panel-heading{.panel-default}+.panel-body{Default})+(>.panel.panel-primary>.panel-heading{.panel-primary}+.panel-body{Primary})+(>.panel.panel-success>.panel-heading{.panel-success}+.panel-body{Success})+(>.panel.panel-info>.panel-heading{.panel-info}+.panel-body{Info})+(>.panel.panel-warning>.panel-heading{.panel-warning}+.panel-body{Warning})+(>.panel.panel-danger>.panel-heading{.panel-danger}+.panel-body{Danger})
```

# Tabs

- Tabulação para conteúdo
  - Classe “nav” e “nav-tabs” definem o contêiner
  - Já “active” é a classe da aba ativa

```
<ul class="nav nav-tabs">
 <li class="active">Tab 1
 Tab 2
 Tab 3

```



# Pills

- Similar ao “tabs” porém com visual diferente
  - Também cria tabulação
  - Código também similar, porém com a troca pela classe “nav-pills”
  - A classe “nav-justified” centraliza o título e justifica o espaçamento

```
<ul class="nav nav-pills nav-justified">
 <li class="active">Pill 1
 Pill 2
 Pill 3

```

Pill 1 Pill 2 Pill 3

# Nav Bar

- Barras de navegação
  - Podem conter outros componentes do Bootstrap como “dropdown”
- Pode ser criada com o uso das seguintes classes:
  - navbar → contêiner principal (idealmente com <nav>)
  - navbar-default → estilo padrão de barra
  - nav → contêiner para os itens da barra
  - navbar-nav → identifica tipo do item

# Nav Bar

```
<nav class="navbar navbar-default">
 <ul class="nav navbar-nav">
 <li class="active">Home
 Page 1
 Page 2
 Page 3

</nav>
```

# Nav Bar

- A classe “navbar-default” pode ser substituída por “navbar-inverse” para inverter as cores
  - Torna a barra escura com texto claro

```
<nav class="navbar navbar-inverse">...
```



# Nav Bar

- Para colocar a barra no topo da página basta definir a `<div>` com a classe “container” dentro dela

```
<nav class="navbar navbar-default">
 <div class="container-fluid">
 <ul class="nav navbar-nav">
 <li class="active">Home
 Page 1...

 </div>
</nav>
```

# Nav Bar

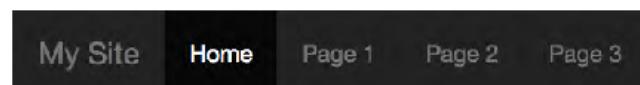
- A classe “navbar-fixed-top” torna a barra fixa no topo
  - Efeito oposto pode ser obtido com a classe “navbar-fixed-bottom”

```
<nav class="navbar navbar-default navbar-fixed-top">
 <div class="container-fluid">
 <ul class="nav navbar-nav">
 <li class="active">Home
 Page 1...

 </div>
</nav>
```

# Nav Bar

- Paralelamente as classes “nav navbar-nav”, podem ser definidos os elementos “header” e “brand”
  - Identificam o “cabeçalho” e “logotipo/marca” na barra



# Nav Bar

```
<nav class="navbar navbar-default navbar-fixed-top">
 <div class="container-fluid">
 <div class="navbar-header">
 My Site
 </div>
 <ul class="nav navbar-nav">
 <li class="active">Home
 Page 1...

 </div>
</nav>
```

# Nav Bar

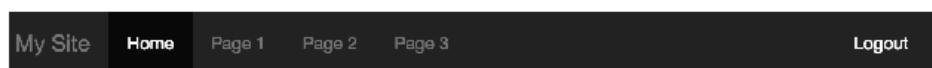
- Itens podem ser alinhados à direita adicionando a classe “navbar-right” junto com “nav navbar-nav”

```
<ul class="nav navbar-nav">...

<ul class="nav navbar-nav navbar-right">

 Logout


```



## EP: Nav Bar

1. Crie e vincule os arquivos de nome “nav-bar”
  - a. Vincule os arquivos necessários para o Bootstrap
2. Adicione o seguinte HTML utilizando Emmet:

```
nav.navbar.navbar-inverse.navbar-fixed-top>.container>(.
navbar-header>a.navbar-brand[href="#"] {My-Site}) + (ul.nav.
navbar-nav>li.active>a[href="#"] {Home} + li*7>a[href="#"]
{Page$}) + ul.nav.navbar-nav.navbar-right>li>a[href="#"]
{Logout}
```

3. Salve e teste

# Breadcrumb

- Identificam a hierarquia navegacional
  - Caminho percorrido na navegação feita pelo usuário
- Classe “breadcrumb” deve ser usada no elemento `<ul>` ou `<ol>`
  - Cada `<li>` deve conter `<a>` com o “link” de navegação
  - A página corrente não está inserida em `<a>` e o `<li>` possui a classe “active”

# Breadcrumb

```
<ol class="breadcrumb">
 Home
 Products
 Books
 <li class="active">Fifty Shades of Grey

```

Home / Products / Books / Fifty Shades of Grey

# Forms

- Existem 3 tipos de layouts:

- Vertical (padrão)
- Horizontal
- Inline

# Forms

- Regras de utilização

- Agrupe “labels” e “form controls” dentro de <div> com a classe “form-group”, a qual otimiza o espaçamento
- Adicione a classe “form-control” (100% de largura) para os elementos de texto <input>, <textarea> e <select>
- Controles individuais (como botão “submit”) não necessitam de agrupamento

# Forms

```
<form>
 <div class="form-group">
 <label for="login">Login</label>
 <input type="text" class="form-control" id="login">
 </div>
 <div class="form-group">
 <label for="pass">Password</label>
 <input type="text" class="form-control" id="pass">
 </div>
</form>
```

# Forms

- Possível colocar um texto estático ao lado de <label>
  - A classe “form-control-static” deve ser usada no elemento desejado
  - Por ser estático o usuário não será capaz de efetuar alterações

```
<div class="form-group">
 <label for="email">Email</label>
 <p class="form-control-static"
 id="login">email@email.com</p>
</div>
```

# EP: Forms

1. Crie e vincule os arquivos de nome “form-vertical”
  - a. Vincule os arquivos necessários para o Bootstrap
2. Adicione o seguinte HTML utilizando Emmet:

```
.container>form.col-md-6>(.form-group>label[for=input]{Input}+input.form-control#input)+(.form-group>label[for=select]{Select}+select.form-control#select>option[value=opt-$]{Option $}*7)+(.form-group>label[for=text-area]{Text Area}+textarea.form-control#text-area)+button.btn.btn-default[type=submit]{Submit}
```

3. Salve e teste
  - a. Remova o atributo “cols” de <textarea>

## Horizontal Form

- Os “labels” e “form controls” ficam dispostos lado a lado
  - Elemento seguinte ficará na linha abaixo do anterior
- Quando não há espaço o layout muda para vertical automaticamente
  - Comum quando ocorre responsividade da página

# Horizontal Form

- Classe “form-horizontal” deve ser aplicada em <form>
  - Aninhamentos dos demais elementos deve seguir as regras de “grid” para conseguir o posicionamento correto
  - “form-groups” se comportam como “rows”, assim não há necessidade de utilizar a classe “row”
  - Classe “control-label” alinha o “label” com o “control”

# Horizontal Form

```
<form form-horizontal>

 <div class="form-group">
 <label for="input1" class="col-sm-2 control-label">Input1</label>
 <div class="col-sm-10">
 <input type="text" class="form-control" id="input1">
 </div>
 </div>
</form>
```

# EP: Horizontal Form

1. Crie e vincule os arquivos de nome “form-horizontal”

- a. Vincule os arquivos necessários para o Bootstrap

2. Adicione o seguinte HTML utilizando Emmet:

```
.container>form.row.form-horizontal>(.form-group>label.col-sm-2.control-label[for=input$]{Input$}+.col-sm-10>input.form-control#input$)*5
```

3. Salve e teste

## Inline Form

- Todos os elementos ficam com posicionamento “inline-block”
  - Posicionados lado a lado
  - Alinhados à esquerda
- “Viewport” deve possuir no mínimo 768 pixels de largura
- Classe “form-inline” deve ser utilizada em <form>

# Inline Form

```
<form class="form-inline">
 <div class="form-group">
 <label for="input1">Input1</label>
 <input type="text" class="form-control" id="input1">
 </div>
 <div class="form-group">
 <label for="input2">Input2</label>
 <input type="text" class="form-control" id="input2">
 </div></form>
```

# Inline Form

1. Crie e vincule os arquivos de nome “form-inline”
  - a. Vincule os arquivos necessários para o Bootstrap
2. Adicione o seguinte HTML utilizando Emmet:

```
.container>form.row.form-inline>(.form-group>label
[for=input$] {Input$}+input.form-control#input$)*3+button.
btn.btn-default[type=submit] {Submit}
```

3. Salve e teste

# Checkbox & Radios

- Existem classes específicas para estes elementos
  - .disabled → quando utilizada em conjunto com <label> alterar o cursor do mouse para “not allowed”
  - .radio
  - .radio-inline
  - .checkbox
  - .checkbox-inline

# Estados de Validação

- Bootstrap define alguns estados visuais comuns em validações de formulários
  - Não efetua a validação, apenas disponibiliza o estilo para os estados
- As classes podem ser atribuídas aos “form-group”:
  - .has-warning → amarelo
  - .has-error → vermelho
  - .has-success → verde

# Estados de Validação

```
<div class="form-group has-success">
 <label for="input1">Input 1</label>
 <input type="text" class="form-control" id="input1">
</div>

<div class="form-group has-warning">
 <label for="input2">Input 2</label>
 <input type="text" class="form-control" id="input2">
</div>
```

# Estados de Validação

The screenshot shows a web form with three input fields and three checkbox options:

- Input with success:** An input field with a green border.
- Input with warning:** An input field with a yellow border.
- Input with error:** An input field with a red border.

Below the inputs is a group of three checkboxes:

- Checkbox with success
- Checkbox with warning
- Checkbox with error

# Tamanho dos Controles

- Controles que utilizam a classe “form-control” podem ter tamanhos alterados
  - Uso é conjunto com a classe “form-control”
- Adicionalmente ao tamanho médio (padrão) é possível definir os tamanhos com as classes:
  - .input-lg
  - .input-sm

# Tamanho dos Controles

```
<input class="form-control input-lg" type="text">
<input class="form-control" type="text">
<input class="form-control input-sm" type="text">

<select class="form-control input-lg">...</select>
<select class="form-control">...</select>
<select class="form-control input-sm">...</select>
```

# Tamanho dos Controles



# Tamanho dos Controles

- Classe “form-group-lg” e “form-group-sm” alteram o tamanho do <label> e dos controles aninhados

```
<div class="form-group form-group-lg">
 <label for="input1">Input 1</label>
 <input type="text" class="form-control" id="input1">
</div>

<div class="form-group form-group-sm">
 <label for="input2">Input 2</label>
 <input type="text" class="form-control" id="input2">
</div>
```

# Tamanho dos Controles



## Bloco de Ajuda

- Texto adicional exibido abaixo do controle
- A classe “help-block” pode ser adicionada à <span>

```
<div class="form-group">
 <label for="input1">Input 1</label>
 <input type="text" class="form-control" id="input1">
 Help text...
</div>
```

# Bloco de Ajuda

Password:

Help text...

## Input Group

- Agrupa elementos adicionais dentro do `<input>`
- As classes “`input-group`” e “`input-group-addon`” adicionam no início ou fim do elemento
  - Usualmente os elementos `<span>` e `<div>` recebem as classes, mas formas mais complexas podem ser criadas (botões, radios, checkbox)
  - Devem ser usadas apenas em conjunto com `<input>`



# Input Group

```
<form class="form-inline">
 <div class="form-group">
 <div class="input-group">
 <div class="input-group-addon">$</div>
 <input type="text" class="form-control" id="amount">
 <div class="input-group-addon">.00</div>
 </div>
 </div>
</form>
```

## EP: Input Group

1. Crie e vincule os arquivos de nome “input-group”
  - a. Vincule os arquivos necessários para o Bootstrap
2. Adicione o seguinte HTML utilizando Emmet:

```
.container>form.form-horizontal.col-sm-4>(.form-group>
 input-group>.input-group-addon{\$}+input.form-control+
 input-group-addon{.00})+(.form-group>.input-group>input.
 form-control+span.input-group-btn>button.btn.btn-default
 {Go})
```

3. Salve e teste

# Carousel

- Cria um “slideshow” de fotos e textos
  - Utiliza transições e animações CSS3
  - Não há suporte para IE 9 e versões anteriores
- Pode ser criado e configurado usando apenas HTML
  - As imagens, botões e textos podem ser criados no HTML
  - Exige o uso de algumas classes e “id” específicos para funcionar

# Carousel

Requisito	Descrição	Exemplo
id="myCarousel"	Contêiner para todo o componente deve possuir o identificador e classe com este valor	<div id="myCarousel" class="carousel">
class="carousel"		
class="slide"	Adiciona animação e transição entre cada “slide”	<div id="myCarousel" class="carousel slide">
data-ride="carousel"	Inicia as trocas entre os “slides”	<div id="myCarousel" class="carousel slide" data-ride="carousel">

# Carousel

- Composto pelos seguintes elementos aninhados ao contêiner do carrossel:
  - Indicadores
  - “Slides”
  - “Captions”
  - Controles

## Carousel Indicators

- Pequenos círculos indicando quantos “slides” existem e permitem a navegação entre eles
  - São opcionais e podem ser omitidos



# Carousel Indicators

- Adicionados através da classe “carousel-indicators” em <ul> ou <ol>
  - Elementos devem estar dentro do contêiner do carrossel
  - Os elementos <li> não possuem nenhum valor, mas devem usar os seguintes atributos:
    - data-slide-to="0" → número da posição (iniciando com zero) do "slide" o qual direciona quando for clicado
    - data-target="#myCarousel" → identificador do carrossel
    - A classe "active" indica o indicador ativo

# Carousel Indicators

```
<div id="myCarousel" class="carousel slide">
 <ol class="carousel-indicators">
 <li data-target="#myCarousel" data-slide-to="0"
 class="active">
 <li data-target="#myCarousel" data-slide-to="1">
 <li data-target="#myCarousel" data-slide-to="2">
 <li data-target="#myCarousel" data-slide-to="3">

 ...

```

# Carousel Slides

- Classe “carousel-inner” em uma `<div>` define os “slides”
  - Deve estar dentro do contêiner do carrossel
- Dentro de “carousel-inner” estarão aninhados os “slides” como items
  - `<div>` com a classe “item” e uma imagem aninhada a esta
- A classe “active” deve ser definida à um dos “slides”
  - Obrigatório o uso, caso contrário nenhum “slide” estará visível

# Carousel Slides

```
<div id="myCarousel" class="carousel slide">
 ...
 <div class="carousel-inner">
 <div class="item active"></div>
 <div class="item"></div>
 <div class="item"></div>
 </div>
 ...

```

# Carousel Caption

- Um texto pode ser adicionado à cada imagem
- Uma nova `<div>` com a classe “carousel-caption” deve ser aninhada dentro da `<div>` com a classe “item”
- O texto estará aninhado dentro da `<div>` com a classe “carousel-caption”

# Carousel Caption

```
<div class="carousel-inner">
 <div class="item active">

 <div class="carousel-caption">
 <h3>Image 1</h3>
 <p>Some caption for image 1</p>
 </div>
 </div>
</div>
```

# Carousel Controls

- Botões de controles do carrossel
  - Posicionados a esquerda e direita do componente
  - Permitem navegar para o “slide” anterior e posterior
- O elemento `<a>` deve ser configurado com:
  - Classe “left” ou “right”
  - Atributo “`href`” com o valor “myCarousel”
  - Atributo “`data-slide`” com o valor “`prev`” ou “`next`”
  - Elemento aninhado que será usado como botão

# Carousel Controls

```
<div id="myCarousel" class="carousel slide">
 ...
 <a class="left carousel-control" href="#myCarousel"
 role="button" data-slide="prev"><
 <a class="right carousel-control" href="#myCarousel"
 role="button" data-slide="next">>
</div>
```

# EP: Carousel

1. Crie e vincule os arquivos de nome “carousel”
  - a. Vincule os arquivos necessários para o Bootstrap
2. Adicione o seguinte HTML utilizando Emmet:

```
.container>#myCarousel.carousel.slide[data-ride=carousel]>(ol.carousel-indicators>li.active[data-target=#myCarousel data-slide-to=0]+li[data-target=#myCarousel data-slide-to=$]*9)+(.carousel-inner>(.item.active>img[src=http://lorempixel.com/800/600/cats/1 alt=Cat-$])+(.item>img[src=http://lorempixel.com/800/600/cats/$@2 alt=Cat-$])*9)+(a.carousel-control.left [href=#myCarousel data-slide=prev]>span.glyphicon.glyphicon-chevron-left)+(a.carousel-control.right [href=#myCarousel data-slide=next]>span.glyphicon.glyphicon-chevron-right)
```

3. Salve e teste

# EP: Carousel

4. Adicione o CSS abaixo para centralizar as imagens:

```
.carousel-inner img {
 width: 70%;
 margin: auto;
}
```

5. Salve e teste

# Modal

- Caixa de diálogos
- Composto por:
  - `class="modal"` → contêiner
    - `class="modal-dialog"` → largura e margens
      - `class="modal-content"` → conteúdo
        - `class="modal-header"` → cabeçalho
        - `class="modal-body"` → corpo
        - `class="modal-footer"` → rodapé

# Modal

```
<div id="myModal" class="modal fade">
 <div class="modal-dialog">
 <div class="modal-content">
 <div class="modal-header">...</div>
 <div class="modal-body">...</div>
 <div class="modal-footer">...</div>
 </div>
 </div>
</div>
```

# Modal

<i>Requisito</i>	<i>Descrição</i>	<i>Exemplo</i>
<code>id="myModal"</code>	Contêiner para todo o componente deve possuir o identificador e classe com este valor	
<code>class="modal"</code>		<code>&lt;div id="myModal" class="modal"&gt;</code>
<code>class="fade"</code>	Adiciona animação e transição entre cada “slide”	<code>&lt;div id="myModal" class="modal fade"&gt;</code>

# Dialog & Content

<i>Requisito</i>	<i>Descrição</i>	<i>Exemplo</i>
<code>class="modal-dialog"</code>	Define a largura e margens, deve estar aninhado como primeiro elemento filho do contêiner	<code>&lt;div class="modal-dialog"&gt;</code>
<code>class="modal-dialog modal-sm"</code> <code>class="modal-dialog modal-lg"</code>	As classes “modal-sm” e “modal-lg” definem o tamanho	<code>&lt;div class="modal-dialog modal-sm"&gt;</code>
<code>class="modal-content"</code>	Conteúdo do “modal”, deve estar aninhado como primeiro filho do “dialog”	<code>&lt;div class="modal-content"&gt;</code>

# Header, Body & Footer

Requisito	Descrição	Exemplo
class="modal-header"		<div class="modal-header">
class="modal-body"	Devem estar aninhados dentro de “content”	<div class="modal-body">
class="modal-footer"		<div class="modal-footer">

## Modal Behavior

- Comportamento de abertura e fechamento pode ser executado por um botão HTML ou código JS
- Atributo “data-*\**” identifica o “modal” e seu comportamento
  - Atributo deve ser usado no botão de abertura e no de fechamento do “modal”

# Modal Behavior

- A abertura do “modal” pode ser configurada com:

- `data-toggle="modal"` → abre a janela “modal”
  - `data-target="#myModal"` → identifica o “modal”

```
<button
 class="btn btn-default"
 data-toggle="modal"
 data-target="#myModal">Show</button>
```

# Modal Behavior

- Já o fechamento pode ser configurado por:
  - `data-dismiss="modal"` → remove o “modal”
- Geralmente utilizada dentro do “modal”

```
<button
 class="btn btn-default"
 data-dismiss="modal">Close</button>
```

# EP: Modal

1. Crie e vincule os arquivos de nome “modal”
  - a. Vincule os arquivos necessários para o Bootstrap
2. Adicione o seguinte HTML utilizando Emmet:

```
.container>(button.btn.btn-default.btn-lg[data-toggle=modal data-target=#modal]{Open})+.modal.fade#modal>.modal-dialog>.modal-content>(.modal-header>button.close[data-dismiss=modal]{×}+h2.modal-title{Title})+(>p{Body})+(>.modal-footer>button.btn.btn-default[data-dismiss=modal]{Close})
```

3. Salve e teste

# Tooltip

- Pequena caixa que aparece como “pop-up” quando o usuário mantém o cursor do “mouse” sobre o elemento
  - Tenha em mente que não existe “mouse over” dispositivos móveis
- Pode ser configurado pelos atributos:
  - title → texto que será usado dentro do “pop-up”
  - data-toggle="tooltip" → configura o componente do Bootstrap
  - data-placement="top" → configura a posição do “pop-up”, valores aceito são “top”, “bottom”, “left” e “right”

# Tooltip

```
<a href="#" data-toggle="tooltip" data-placement="top"
title="Info">Top

<a href="#" data-toggle="tooltip" data-placement="bottom"
title="Info">Bottom

<a href="#" data-toggle="tooltip" data-placement="left"
title="Info">Left

<a href="#" data-toggle="tooltip" data-placement="right"
title="Info">Right
```

# Tooltip JS

- Para que funcione, é necessário utilizar a seguinte instrução JS:

```
$(document).ready(function() {
 $('[data-toggle="tooltip"]').tooltip();
});
```

# EP: Tooltip

1. Crie e vincule os arquivos de nome “modal”
  - a. Vincule os arquivos necessários para o Bootstrap
2. Adicione o seguinte HTML utilizando Emmet:

```
.container>ul>+(li>a[href="# data-toggle=tooltip data-placement=left title=Info] {LEFT})+(li>a[href="# data-toggle=tooltip data-placement=right title=Info] {RIGHT})+(li>a[href="# data-toggle=tooltip data-placement=top title=Info] {TOP})+(li>a[href="# data-toggle=tooltip data-placement=bottom title=Info] {BOTTOM})
```

# EP: Tooltip

3. Adicione o seguinte JS:
  - a. Abaixo de todos os demais

```
$(document).ready(function() {
 $('[data-toggle="tooltip"]').tooltip();
});
```

4. Salve e teste

# Popover

- Similar ao “pop-up”, porém aparece quando o usuário clica sobre o elemento
  - Também pode ser configurado para “mouse over”, mas o clique abrange os dispositivos móveis
  - Pode conter mais conteúdo que o “pop-up”

# Popover

- Configurado através de:
  - `data-toggle="popover"` → cria a funcionalidade
  - `data-placement="top"` → configura a posição do “pop-up”, valores aceito são “top”, “bottom”, “left” e “right”
  - `data-trigger="focus"` → fecha o “popover” quando clicar fora deste
  - `data-trigger="hover"` → abre o “popover” com o “mouse over”
  - `data-content="Content..."` → define o conteúdo dentro do “popover”
  - `title` → cabeçalho do “popover”

# Popover

```
<a href="#"
 data-toggle="popover"
 title="Popover Header"
 data-content="Content...."
 data-trigger="focus">Toggle
```

# Popover JS

- Requer o seguinte código JS:

```
$(document).ready(function() {
 $('[data-toggle="popover"]').popover();
});
```

# EP: Popover

1. Crie e vincule os arquivos de nome “modal”
  - a. Vincule os arquivos necessários para o Bootstrap
2. Adicione o seguinte HTML utilizando Emmet:

```
a[href=# data-toggle=popover title=Header data-content=Content data-trigger=focus] {Toggle}
```

3. Adicione o JS:

```
$(document).ready(function() {
 $('[data-toggle="popover"]').popover();
});
```

4. Salve e teste

# Scrollspy

- Atualiza o “link” selecionado automaticamente conforme a posição da barra de rolagem (“scroll”)
- Configurado através de:
  - data-spy="scroll" → o elemento no qual ocorrerá o “scroll” (geralmente <body> com posicionamento relativo)
  - data-target=".navbar" → identifica o “id” ou a classe do elemento de navegação com os “links” (deve estar dentro da área com “scroll”)
  - data-offset="12" → altura da barra de navegação a ser ignorada

# Scrollspy

```
<body data-spy="scroll"
 data-target="#myNav"
 data-offset="15">

<div class="container">
 <div class="row">
 <nav id="myNav">...
```

## EP: Scrollspy

1. Crie e vincule os arquivos de nome “scrollspy”
  - a. Vincule os arquivos necessários para o Bootstrap
2. Adicione os seguintes atributos em <body>:

```
<body data-spy="scroll"
 data-target=".navbar"
 data-offset="50">
```

# EP: Scrollspy

3. Crie a barra de navegação usando o Emmet:

a. Toda a instrução deve estar em uma única linha para ser convertida

```
(nav.navbar.navbar-inverse.navbar-fixed-top>(.navbar-
header>button.navbar-toggle[data-toggle=collapse data-
target=#myNavbar]>span.icon-bar*3)+.container-fluid>.
collapse.navbar-collapse#myNavbar>ul.nav.navbar-nav>li*3>a
[href=#page$] {Page-$})+ (#page$>.container-fluid>h2{Page-$}
+p*10>lorem)*3
```

# EP: Scrollspy

4. Adicione o CSS:

```
body {position: relative; font-size: 24px;}

#page1 {background-color: salmon; margin-top: 50px;}

#page2 {background-color: gold}

#page3 {background-color: skyblue}
```

5. Salve e teste

# Links

- <http://responsive.vermillion.com/compare.php>
- <http://getbootstrap.com/javascript/#js-programmatic-api>
- <http://bootsnipp.com/>
- <http://bootswatch.com/>
- <http://startbootstrap.com/bootstrap-resources/>
- <http://tutorialzine.com/2013/07/50-must-have-plugins-for-extending-twitter-bootstrap/>
- <http://www.bootply.com/>
- <https://github.com/JasonMortonNZ/bs3-sublime-plugin>

# NodeJS

Servidor JavaScript

# Cases

- eBay
- [Google, Mozilla & Yahoo](#)
- [LinkedIn](#)
  - [LinkedIn](#)
- [PayPal](#)
- [Walmart](#)

# Popularidade

- Terceiro projeto mais popular no GitHub
- Mais de 2 milhões de downloads por mês
- Mais de 81000 módulos disponíveis (NPM)
- Vagas de empregos\* tem crescido mais que qualquer outra linguagem de programação back-end

\*fonte [indeed.com](#)

# Popularidade



## Sobre

- Servidor JavaScript
  - Aplicações usando JS como linguagem back-end
- Toda API básica JS é exatamente igual à utilizada na programação front-end
  - Condicionais
  - Variáveis
  - Objetos



# Sobre

- Porém os recursos do ambiente Web não existem:
  - window
  - alert
  - document

# V8 JavaScript Runtime

- “Engine” criado pelo Google
  - Utilizado no Google Chrome e outros navegadores
- Código escrito na linguagem C++
  - Alta performance de interpretação e execução do código



# Event Driven

- Programação dirigida à eventos
  - Executa ações conforme o disparo de eventos
- Múltiplas ações simultâneas (Multithread)
  - Processos ocorrem em paralelo
  - Assíncrono
  - “Non-blocking”

# Utilização

- Application Server
  - Serviços Web
  - Upload Client
- Websocket Server
  - Live Chat
  - Real-Time Data Apps

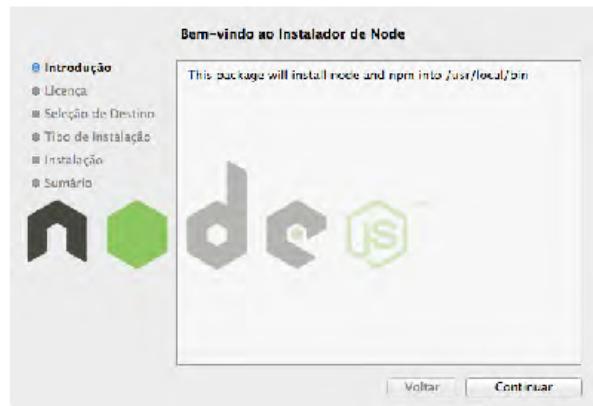
# Utilização

- Front-end Developer Tool
  - Grunt
  - Bower
  - JSDocs
  - Pre-processors Compiler

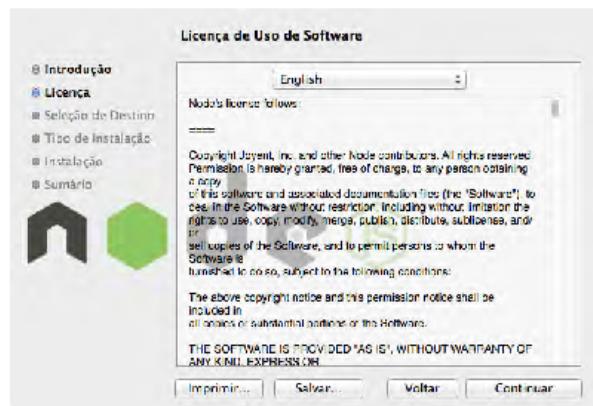
## EP: Instalação

1. Baixe o arquivo instalador do NodeJS
  - a. Clique no botão "Install"  
<http://nodejs.org/>
2. Efetue a instalação do software
  - a. Duplo clique sobre o arquivo e siga os procedimentos informados
  - b. Para a instalação padrão basta clicar em "Continuar"

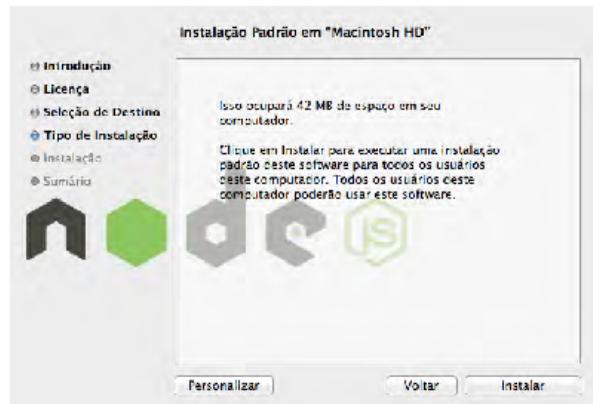
# EP: Instalação



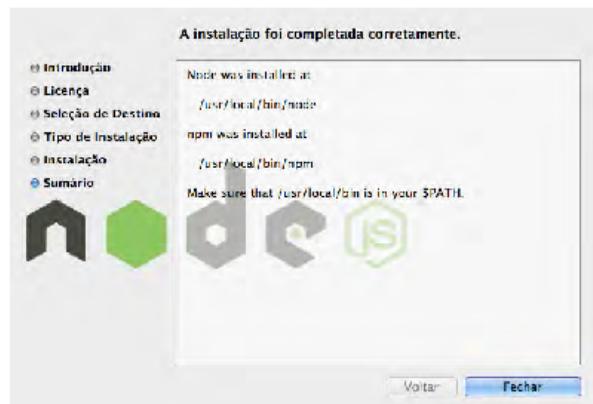
# EP: Instalação



# EP: Instalação



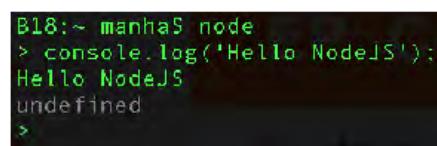
# EP: Instalação



# EP: Instalação

3. Após a instalação, abra o terminal e teste o NodeJS digitando:

- a. node
- b. console.log('Hello NodeJS');



```
B18:~ manha$ node
> console.log('Hello NodeJS');
Hello NodeJS
undefined
>
```

A screenshot of a terminal window titled 'B18:~ manha\$'. It shows the command 'node' being run, followed by 'console.log('Hello NodeJS');'. The output 'Hello NodeJS' is displayed in green, and 'undefined' is shown below it. A cursor is visible at the end of the line.

# Execução de Código

- NodeJS faz a leitura e execução de um arquivo JS tradicional
  - Criar o arquivo e adicionar o código
  - Executar o código do arquivo, utilizando o comando “node”:

```
node file-name.js
```

# EP: Execução de Código

1. Crie um novo arquivo de nome “hello.js” com o código:

```
var users = ['Raul', 'Silvio', 'Fausto'];

for (var i = 0; i < users.length; i++) {
 console.log('Hello ' + users[i]);
}
```

# EP: Execução de Código

2. Abra o terminal e navegue até o arquivo “hello.js”
3. Na mesma pasta do arquivo “hello.js” digite:

```
node hello.js
```

# Módulos

- Módulos são códigos JS
  - Como qualquer biblioteca JS o uso do seu código está condicionado ao carregamento do seu arquivo (módulo)
  - Similar ao “import” usado em diversas linguagens de programação
- Uma vez carregado e utilizado no seu arquivo JS o módulo passa a ser uma dependência
  - Similar a carregar e usar jQuery em uma página Web

# Módulos

- Quando carregado é retornado um objeto com a API carregada
- Módulos podem ser:
  - Bibliotecas
  - Funções
  - Qualquer tipo de objeto JS (Array, Object, etc)

# require( )

- Permite carregar módulos no arquivo JS
  - Podem ser bibliotecas próprias do NodeJS, bibliotecas de terceiros e até mesmo criadas pelo próprio desenvolvedor
- Comando “require( )” carrega e retorna o código JS
  - Importante armazenar o retorno em uma variável para uso posterior

```
var library = require('./filename');
```

# require( )

Parâmetro	Definição	Exemplo
Nome do arquivo JS	require('FILENAME')	require('./my-module')
Nome do arquivo JS + extensão “.js”	require('FILENAME.js')	require('./my-module.js')
Nome da pasta com arquivo “package.json” configurado	require('/FOLDERNAME')	require('/modules')
Caminho absoluto	require('ABSOLUTE-PATH')	require '~/Desktop/modules'
Caminho relativo ao diretório do arquivo atual	require('../FOLDER/FILE')	require('../modules/my-module')

# node\_modules

- Existe ainda a opção de utilizar uma pasta com o nome “node\_modules”
  - Nome padrão utilizado pelo NodeJS para encontrar módulos

```
require('./my-module')
//node_modules/my-module
```

# node\_modules

- Idealmente deve estar no mesmo diretório do arquivo que executa o método “require( )”
  - Se não encontrar o módulo, o NodeJS irá procurar no nível superior
  - Essa procurar ocorrerá até que o módulo seja encontrado ou chegar no “root folder”

```
./node_modules/my-module
../../node_modules/my-module
```

# Exportando Módulos

- Códigos JS podem ser exportados como módulos
  - Permitem o reaproveitamento em outros arquivos
- A exportação deve ocorrer na última linha do arquivo com o código a ser exportado
  - “module” é uma variável global do NodeJS
  - Qualquer objeto JS pode ser exportado

```
module.exports = MyModule
```

# Exportando Módulos

- Para utilizar um módulo exportado, basta usar o método “require( )” no arquivo desejado

```
var myModule = require('./my-module');
```

# EP: Módulos

1. Crie o arquivo de nome “my-module.js” e adicione o código:

```
var getSum = function(a,b){
 return a + b;
};

module.exports = getSum;
```

# EP: Módulos

2. Salve o arquivo com o código do módulo
3. Crie um novo arquivo de nome “sum.js”
4. Adicione o código abaixo:

```
var getSum = require('./my-module');

console.log(getSum(1,2));
```

5. Salve e execute o código:      `node sum.js`

# Core Modules

- São módulos compilados junto com o NodeJS
  - Estão presentes em qualquer instalação do NodeJS, sem necessitar baixar pacotes adicionais
  - Funcionalidades essenciais do servidor
- Fazem parte da API do NodeJS

<https://nodejs.org/api/>

## HTTP Module

- Permite criar e configurar um servidor

```
var http = require('http');
```
- O método “[createServer\(\)](#)” cria a instância do servidor
  - Recebe 2 parâmetros, requisição e resposta
  - Método retorna um objeto da classe “[Server](#)”

# HTTP Module

```
http.createServer(function(request, response){
 response.writeHead(200);
 response.write('Hello from NodeJS');
 response.end();
}).listen(8080);
```

## Alterações no Código

- Qualquer modificação no código, a execução deve ser interrompida e invocado novamente
  - A interrupção do código pode ser feita com o pressionando CTRL + C no terminal

# EP: HTTP Module

1. Crie um novo arquivo de nome “server.js” com:

```
var http = require('http');
var server=http.createServer(function(request, response) {
 response.writeHead(200);
 response.write('<h1>Hello from NodeJS</h1>');
 response.end();
});
server.listen(8080);
console.log('Server on URL localhost and PORT 8080');
```

# EP: HTTP Module

2. Salve o arquivo e execute o seu código:

```
node server.js
```

3. Abra o navegador e digite o endereço:

- a. Ambos os endereços são equivalentes

<http://localhost:8080> ou <http://127.0.0.1:8080>

# EP: HTTP Module

4. Interrompa a execução do código pressionando no terminal:

**CTRL + C**

5. Tente abrir no navegador o endereço novamente

- a. Perceba que não há resposta do servidor e portanto o navegador não consegue exibir a página HTML

# Path Module

- Permite lidar com os caminhos de arquivos ou pastas
- Resolve problemas como:
  - Encontrar e extrair caminhos
  - Normalizar caminhos ( Mac barra “/”, no Windows barra invertida “\” )
  - Concatenar textos para formar caminhos
  - Verificar a existência de caminhos

```
require('path')
```

# FS Module

- Funcionalidades para lidar com arquivos

- Criar
- Apagar
- Abrir
- Ler
- Escrever
- Fechar

```
require('fs')
```

## fs.stat( )

- Objeto com métodos e propriedades como:

- stat(path, callback) → carrega as informações de um arquivo, como tamanho, data de criação e modificação, etc

```
var fs = require('fs');

fs.stat('./filename.ext',
 function(err, stats){...}

);
```

## fs.stat( )

- O método “stat( )” retorna à função “callback” um objeto “stats” que possui diversas funções como:
  - stats.isFile() → verifica se é um arquivo
  - stats.isDirectory() → verifica se é um diretório

```
function(err, stats){
 if(stats.isFile()){...}
}
```

## EP: fs.stat( )

1. Crie o arquivo de nome “page.html” com a estrutura básica HTML e o texto “lorem ipsum” em <body>
2. Crie o arquivo de nome “stat.js” na mesma pasta do arquivo criado no passo anterior

## EP: fs.stat( )

3. Adicione o seguinte código:

```
var fs = require('fs');

fs.stat('./page.html', function(err, stats){

 console.log('Is file ? ' + stats.isFile());

 console.log(stats);

}) ;
```

4. Salve e execute o código do arquivo “stat.js”

## fs.readFile( )

- Abre e lê o arquivo definido no primeiro parâmetro

```
fs.readFile(path, callback)
```

- Quando carregado a função “callback” será invocada com as informações obtidas
  - A função “callback” receberá 2 parâmetros, erro e o “buffer” com os dados carregados

## fs.readFile( )

```
var fs = require('fs');

fs.readFile('./test.txt',
 function(err, buffer){ ... }

);
```

## EP: fs.readFile( )

1. Crie o arquivo de nome “read.js” com o código:

```
var fs = require('fs');

fs.readFile('./page.html', function(err, buffer){
 console.log(buffer.toString());
});
```

2. Salve e execute o arquivo

## EP: fs.readFile( )

3. (Opcional) Crie um novo arquivo que disponibilize um servidor e responda com a leitura do arquivo HTML

```
var html, http = require('http'), fs = require('fs');

fs.readFile('./page.html', function(err, buffer){
 html = buffer.toString();
}) ;
```

## EP: fs.readFile( )

```
http.createServer(function(request, response){
 response.writeHead(200, {'Content-Type': 'text/html'});
 response.write(html);
 response.end();
}).listen(8080);

console.log('Server running on localhost at port 8080');
```

# npm

- Node Package Modules  
<https://www.npmjs.org/>
- Gerenciador de pacotes do NodeJS
- Permite instalar novos recursos
  - Sintaxe de instalação do pacote é bastante simples:



```
npm install module-name
```

## Global vs Local

- A instalação de um módulo pode ocorrer:
  - Global → acessível em qualquer pasta ou local do projeto
  - Local → acessível apenas na pasta do projeto
- Para instalação global a instrução deve receber o parâmetro “-g”
  - Local é a forma padrão de instalação de um módulo

```
npm install -g module-name
```

# Global vs Local

- O diretório padrão para as instalações são:
  - Global → /usr/local/lib/node\_modules/
  - Local → CURRENT-FOLDER/node\_modules/
- Para operações globais, é necessário permissão de administrador ou super usuário

# Versão do Módulo

- Um determinado módulo pode ter mais de uma versão
- NPM permite especificar qual versão deve ser instalada
  - Expressões regulares podem ser usadas
  - Sinais “>”, “<”, “>=” e “<=” também podem ser usados

```
npm install module-name@version
```

# Mensagem Instalação

- Após a execução do comando “npm install” uma mensagem será exibida

```
npm install express
express@4.12.3 node_modules/express
 ├── merge-descriptors@1.0.0
 ├── utils-merge@1.0.0
 ...
```

# Mensagem Instalação

- A mensagem do exemplo indica que o módulo “express” depende de outros módulos que foram baixados:
  - Estão na pasta “node\_modules/express/node\_modules/”
  - merge-descriptors@1.0.0 → depende do módulo “merge-descriptors” na versão 1.0.0

# Comandos Úteis NPM

<i>Comando</i>	<i>Descrição</i>
npm install module-name npm install -g module-name	Instala o módulo local / global
npm uninstall module-name npm uninstall -g module-name	Remove o módulo local / global
npm install module-name --save	Salva as dependências no arquivo “package.json”

# Comandos Úteis NPM

<i>Comando</i>	<i>Descrição</i>
npm list npm list -g	Lista módulos local / global
npm -v module-name	Identifica a versão do módulo instalado
npm init	Inicia a criação do arquivo “package.json”
npm ls	Lista os módulos do projeto local

# package.json

- Arquivo deve estar na raiz do projeto
- Possui inúmeras propriedades, sendo a requerida apenas a “dependencies”
  - Demais como, autores, emails, etc, são opcionais
  - A propriedade “dependencies” receberá o nome do módulo como nome da propriedade e seu valor a versão do módulo

<https://docs.npmjs.com/files/package.json>

# package.json

- Arquivo que contém informações sobre o projeto
  - Dependências de produção e desenvolvimento
  - Automatiza a manutenção do projeto
- O comando “npm init” cria o arquivo e define as suas configurações e dependências
  - Múltiplas perguntas serão feitas no terminal, ao término o arquivo “package.json” é criado

# package.json

```
{ "name": "MyApp",
 "version": "1.0.0",
 "dependencies": {
 "express": "4.2.x",
 "cordova": "*",
 "mongoose": ">=3.8.0 <3.8.25"
 }
}
```

## NPM vs Dependências

- O NPM consegue gerenciar e identificar as dependências conforme as informações contidas no arquivo “package.json”
  - Qualquer módulo pode utilizar este recurso, inclusive os criados por você em seu projeto
  - Uma vez definido “package.json” basta executar o comando na pasta

```
npm install
```

# NPM vs Dependências

- Caso uma nova dependência seja adicionada após a execução inicial do projeto o arquivo “package.json” deve ser atualizado
  - O parâmetro “--save” armazena as informações no arquivo
    - Arquivo “package.json” deve existir antes da execução do comando

```
npm install package-name --save
```

# Express

- Framework open source
- API simplifica a criação de serviços Web
  - Facilita o desenvolvimento de sites e aplicações Web
  - Possui diversas funcionalidades prontas, reduzindo a codificação e facilitando o desenvolvimento
- Foco principal é lidar com as requisições HTTP

```
npm install express
```

# Rotas

- Mapeia uma requisição HTTP
  - A requisição pode ser feita usando qualquer método (GET, POST, PUT, DELETE, etc)
- Vincula um endereço URL à uma função JS
  - Função lida com a requisição e executa as ações necessárias

`http://mysite.com/index`

```
app.get('/index', function(req, resp){...})
```

## EP: Rotas

1. Crie uma nova pasta de nome “express” no seu projeto
2. Navegue com o terminal até a nova pasta criada
3. Execute o comando

```
npm init
```

# EP: Rotas

4. Responda as perguntas de configuração com (algumas configurações ficarão em branco):

- a. name: (express) router
- b. version: (1.0.0) 0.0.1
- c. description: router tutorial
- d. entry point: (index.js) router.js
- e. keywords: router
- f. author: me

```
name: (express) router
version: (1.0.0) 0.0.1
description: router tutorial
entry point: (index.js) router.js
test command:
git repository:
keywords: router
author: me
license: (ISC)
```

# EP: Rotas

5. Confirme as alterações

```
x bout to write to /Users/mac/Desktop/temp_project/node/express/package.json
{
 "name": "router",
 "version": "0.0.1",
 "description": "router tutorial",
 "main": "router.js",
 "scripts": {
 "test": "echo \"Error: no test specified\" && exit 1"
 },
 "keywords": [],
 "author": "me",
 "license": "ISC"
}

Is this ok? (yes) y
```

## EP: Rotas

6. Abra o arquivo “package.json” no seu editor de códigos (WebStorm, Sublime Text, etc)
  - a. Visualize as informações configuradas
  - b. O sub-item “test” de “scripts” tem uma mensagem de erro pois nenhum teste foi especificado
  - c. Perceba que não existe o valor “dependencies” neste arquivo

## EP: Rotas

7. Volte ao terminal e instale o “express” com a opção “--save”

```
npm install express --save
```
8. Abra novamente o arquivo “package.json” no seu editor de códigos
  - a. Identifique o valor “dependencies” configurado para o “express”

## EP: Rotas

9. Crie um arquivo de nome “route.js” e adicione:

```
var express = require('express') ,
 app = express();
app.get('/', function (req, res) {
 res.send('<h1>Hello from NodeJS</h1>');
});
app.listen(8080);
```

## EP: Rotas

8. Salve, execute o arquivo JS e abra o endereço:

- a. Lembre-se sempre de interromper a execução anterior com “CTRL + C” antes de executar o arquivo alterado

```
node route.js
```

```
http://localhost:8080
```

# EP: Rotas

9. Crie mais uma rota:

- a. Adicione antes da instrução “app.listen(8080)”

```
app.get('/service', function (req, res) {
 res.json({data: 'Some data'});
});
```

10. Salve, execute o arquivo JS e abra o endereço:

<http://localhost:8080/service>

## Parâmetros GET

- O método “get( )” pode receber parâmetros enviados pela URL
  - Caminho + “/” + nome do parâmetro

```
get('/service/:name',function(req,resp){..})
```

# Parâmetros GET

- Múltiplos parâmetros podem ser enviados acrescentando os valores:

```
get('/service/:name/:last_name'...)
```

## EP: Parâmetros GET

- Abra o arquivo “route.js” criado no EP anterior
- Adicione o seguinte código:

a. Antes do código “app.listen(8080)”

```
app.get('/service/:name', function (req, res){
 res.send('<h1>Hello ' + req.params.name + '</h1>');
});
```

## EP: Parâmetros GET

3. Salve, execute o arquivo e teste as URLs:

- a. Perceba que o serviço JSON continua a funcionar quando nenhum parâmetro é enviado pela URL

`http://127.0.0.1:8080/service/Raul`

`http://127.0.0.1:8080/service`

## EP: Parâmetros GET

4. Altere o código para receber mais de um parâmetro:

```
app.get('/service/:name/:last_name', function (req, res) {
 res.send('<h1>Hello ' + req.params.name + ' ' +
 req.params.last_name + '</h1>');
});
```

5. Salve, execute o arquivo e teste a URL:

`http://127.0.0.1:8080/service/Raul/Gil`

# Arquivos Estáticos

- São os arquivos sem manipulação do servidor
  - HTML não é manipulado pelo NodeJS para gerar a página dinâmica
    - Ex: páginas de um website tradicional são arquivos estáticos
    - Páginas de um blog são criadas dinamicamente com base nas informações enviadas ao servidor (ou páginas de e-commerce, de mídias sociais, etc)
- O servidor atua apenas como “ponte” de acesso
  - Geralmente estão localizados num caminho URL público do servidor
  - Comum ser criada a pasta “public” no projeto para estes arquivos

# Arquivos Estáticos

- Os arquivos estáticos mais comuns são do tipo:
  - HTML → \*.html
  - CSS → \*.css
  - JS → \*.js (para o “front-end”)
  - Imagens → \*.png, \*.jpg, etc
  - Mídia → \*.ogg, \*.mp3, etc
  - Fontes → \*.woff, \*.ttf, etc

# Arquivos Estáticos

- O método “use( )” adiciona os serviços NodeJS
- Já o método “static( )” define o diretórios com os arquivos estáticos

```
app.use(express.static('public'));
```

## EP: Arquivos Estáticos

1. Dentro da pasta do arquivo “route.js” crie uma nova pasta aninhada de nome “public”
2. Dentro da pasta “public”
  - a. Crie um arquivo HTML de nome “index.html”
  - b. Crie uma pasta de nome CSS e um arquivo de nome “style.css”
  - c. Vincule os arquivos “index.html” e “style.css”
  - d. Adicione algum conteúdo no arquivo HTML e aplique estilo no CSS

# EP: Arquivos Estáticos

- Sua estrutura de arquivos deve ficar similar à:

```
router
├── node_modules
└── public
 ├── css
 │ └── style.css
 └── index.html
├── package.json
└── route.js
```

# EP: Arquivos Estáticos

- Abra o arquivo “route.js” e comente o código:

```
/*app.get('/', function (req, res) {
 res.send('<h1>Hello from NodeJS</h1>');
}); */
```

- No mesmo lugar do código comentado adicione:

```
app.use(express.static('public'));
```

# EP: Arquivos Estáticos

6. Salve, execute o arquivo e teste a URL:

`http://127.0.0.1:8080/`

## Simple REST

- O método “express( )” retorna um objeto “Application”
- “Application” possui métodos específicos para REST:
  - `get()`
  - `post()`
  - `put()`
  - `delete()`
- Todos recebem como parâmetro a função “callback” com os parâmetros “request” e “response”

# Simple REST

```
app.get(function(req, res) {...})
app.post(function(req, res) {...})
app.put(function(req, res) {...})
app.delete(function(req, res) {...});
```

# Simple REST

- Os dados são recebidos na função “callback” através da propriedade “body” do parâmetro “request”
  - As propriedades do JSON enviado são transformadas em propriedades de “body”

```
app.post('post', function(req,res){
 console.log(req.body.property);
});
```

# Simple REST

- Para interpretar os dados enviados como JSON é necessário o módulo “body-parser”

- Deve ser instalado via NPM

```
npm install body-parser
```

- O módulo tem de ser requerido e utilizado no código JS

```
var bodyParser = require('body-parser');
app.use(bodyParser.urlencoded({extended: false}));
app.use(bodyParser.json());
```

## EP: Simple REST

- Crie um novo arquivo de nome “rest.js”
  - Coloque dentro de uma pasta do arquivo “route.js”
- Instale o “body-parser”

```
npm install body-parser --save
```

- No arquivo “rest.js” adicione o código para o “express”:

```
var express = require('express');
var app = express();
```

# EP: Simple REST

4. Adicione o código para os arquivos estáticos e para o “body-parser”:

```
var bodyParser = require('body-parser');

app.use(express.static('public'));

app.use(bodyParser.urlencoded({extended: false}));

app.use(bodyParser.json());
```

# EP: Simple REST

5. Crie uma função para devolver o objeto aos métodos:

- a. Cada método retornará as informações recebidas em um objeto JSON

```
function getData(req) {

 return {name: req.body.name, method: req.method};

}
```

# EP: Simple REST

6. Adicione o código para o serviço REST:

```
app.get('/service/:name', function(req,res){
 res.send('<h1>GET ' + req.params.name + '</h1>');});
app.post('/service', function(req,res){
 res.send(getData(req));});
app.put('/service', function(req,res){
 res.send(getData(req));});
app.delete('/service', function(req,res){
 res.send(getData(req));});
app.listen(8080);
```

# EP: Simple REST

7. Crie o código para alternar os métodos nos arquivos da pasta pública:

- a. HTML:

```
<button value="post">POST</button>
<button value="put">PUT</button>
<button value="delete">DELETE</button>
```

# EP: Arquivos Estáticos

## 7. (cont.):

- b. Dentro da pasta “public” crie uma pasta de nome “js”
- c. Dentro da pasta “js” crie um arquivo de nome “app.js”
  - i. Este arquivo refere-se à página HTML (“front-end”) e não ao NodeJS
- d. Vincule o arquivo “index.html” com o arquivo “app.js”
- e. No arquivo “app.js” adicione o código:

```
var buttons = document.querySelectorAll('button');
```

# EP: Simple REST

## e. (cont.):

```
for (var i = 0; i < buttons.length; i++) {
 buttons[i].onclick = function(event){ajax(event.target.value);};;}

function ajax(method){

 var xhr = new XMLHttpRequest();

 xhr.open(method, 'http://127.0.0.1:8080/service',true);

 xhr.setRequestHeader("Content-Type", "application/json;charset=UTF-8");

 xhr.onload = function(){

 var response = JSON.parse(xhr.responseText);

 document.body.innerHTML = '<h2>' + response.method + ': ' + response.name+ '</h2>';
 };
};
```

# EP: Arquivos Estáticos

## 8. Salve e teste

- a. Abra o endereço abaixo e verifique a informação obtida do servidor e adicionada na página
- b. Perceba que não é necessário interromper o serviço após alterar os arquivos estáticos

`http://127.0.0.1:8080/`

# LESS

## CSS Pre Processor

# LESS

- LESS é um pré--processador CSS
  - Processa seu código e gera um arquivo final CSS
- Permite programar de forma mais estruturada o CSS
  - Não é considerada uma nova linguagem, apenas modifica a forma de programar instruções CSS
  - Reduz a quantidade de código CSS
  - Menor repetição de código
  - Facilita a manutenção

# LESS



# Nova Linguagem ?

- Não é considerada uma nova linguagem de programação
- Apenas uma forma diferente de programar CSS
  - Basicamente a mesma sintaxe CSS é utilizada
  - Novos recursos são disponibilizados elevando a capacidade do CSS

# Introdução a Variáveis

- Permitem declarar um valor e reutilizá-lo em diversas partes do código
  - Evita repetição e facilita a alteração do valor
  - Possível ainda efetuar operações matemáticas ou mesmo concatenação com o valor da variável
- Possui a seguinte assinatura:

```
@variable: value;
```

# Introdução a Variáveis

```
@color: #ff0;
@font: 36px;
.myClass {
 color: @color;
 border-color: @color;
 font-size: @font;
}
```

## Pre Processor

- O conceito refere-se ao pré-processamento de um código para gerar o código final
- Necessário utilizar algum recurso adicional para que a compilação ocorra:
  - Biblioteca JavaScript: tempo de execução, na página visitada
  - Aplicativos Web: on-line
  - Linha de comando: máquina do usuário
  - Softwares (diversos): máquina do usuário

# Pre Processor

- Independente da forma como o arquivo é compilado, o resultado final sempre será um arquivo CSS sem qualquer código LESS
  - Fundamental para que qualquer navegador consiga interpretá-lo

## Recursos LESS

- Funções
- Variáveis
- Mixins
- Condicionais
- Escopo
- Aninhamento de blocos

# LESS == CSS

## CSS != LESS

- LESS é CSS, mas CSS não é LESS
- Os recursos do LESS são todos opcionais
  - Se um arquivo LESS contiver apenas instruções CSS será compilado igualmente para CSS
  - Isso permite migrar um arquivo atual CSS para LESS em etapas
- LESS é uma extensão do CSS
  - Adiciona novos recursos e possibilidades à instruções CSS

# Compilação

# Compilando

- O arquivo no qual o código LESS é criado tem a extensão ".less"
  - Este arquivo não funciona no navegador de forma nativa
- Arquivo LESS deve ser convertido (compilado) em CSS
  - Antes de ser implementado em uma página HTML
  - Uma biblioteca JS pode ser usada para converter o arquivo LESS em CSS, mas esta prática é pouco recomendável

# Compilando

- Existem diversas formas e ferramentas disponíveis para a compilação do LESS
  - On-line / Desktop
  - Comando no terminal
  - Integrado com o editor do código
- A escolha impactará no processo de desenvolvimento
  - Idealmente a ferramenta deve estar integrada com o editor do código para se obter uma melhor produtividade de desenvolvimento

# On-line / Desktop

- Existem alguns sites que permitem compilar o código LESS
  - Geralmente não utilizam o arquivo ".less", código deve ser copiado e colado
  - Recebe o código LESS e exibe o resultado em CSS

<http://lesscss.org/usage/#online-less-compilers>

# EP: Pre Processor

1. Abra o endereço:  
<http://less2css.org/>
2. Identifique as duas áreas
  - a. Primeira para código LESS
  - b. Segunda com o código LESS compilado para CSS
3. Apague o código LESS existente

# EP: Pre Processor

4. Digite o código abaixo:

```
@color: red;

.myClass {

 color: @color;

 border-color: @color;

}
```

# EP: Pre Processor

5. Crie uma nova variável:

```
@font: 36px;
```

6. Adicione na classe ".myClass"

7. Visualize o resultado CSS

# Comando no Terminal

- Processo que antecede a integração com o editor de código
- Uso é essencialmente através do terminal do sistema operacional
  - Windows → cmd
  - Mac → Terminal
  - Linux → Terminal

# Comando no Terminal

- Necessário baixar algum software que interpreta o código do arquivo LESS e gera o arquivo CSS
  - Mais comum é o uso do servidor NodeJS
- Configurações podem ser feitas, como:
  - Pasta na qual o arquivo CSS compilado será gerado
  - Nome do arquivo CSS compilado
  - Minificação do arquivo CSS

# Comandos

- O comando para compilação é através do "lessc"
  - Representa o executável do compilador
- O comando abaixo compila o arquivo LESS para CSS:

```
lessc style.less style.css
```

- Pode receber diversos parâmetros de configuração
  - Como para minificar o resultado:

```
lessc -x style.less style.css
```

# Comandos

- Lista completa pode ser encontrada no endereço:

<http://lesscss.org/usage/#command-line-usage>

# Compilação LESS

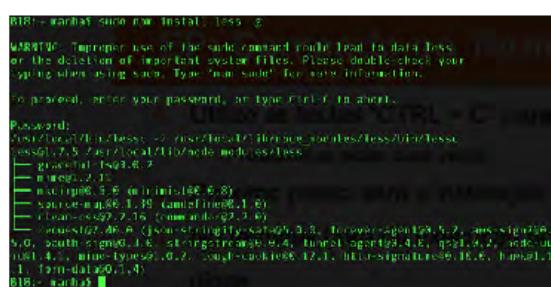
- O NodeJS pode ser usado para compilar LESS
  - Necessário instalar o módulo

```
npm install less -g
```

## EP: Compilação LESS

1. No terminal digite:

```
sudo npm install less -g
```



```
Bruno-Mazurek:~ Bruno$ sudo npm install less -g
WARNING: Proper use of the sudo command could lead to data loss
or the deletion of important system files. Please double-check your
writing when using sudo. Type "man sudo" for more information.

[password]
/usr/local/lib/node_modules/.bin/less@1.3.3
└── less@1.3.3
 ├── autoprefixer@6.5.0 (node-sass@0.10.0)
 ├── source-map@0.1.19 (punycode@2.1.0)
 ├── clean-css@3.2.16 (commander@2.3.0)
 ├── lesscss@0.1.0 (css-stringify@0.3.0, browser-sync@2.9.5, less@0.12.0, node-sass@0.10.0, stringtree@0.4.1, browser-supertab@3.4.1, postcss@4.0.1, node-sass@0.11.0, minify@0.2.2, less-cache@0.7.1, browser-signatures@0.10.0, browser-s...)
 └── Bruno-Mazurek:~ Bruno$
```

# EP: Compilação LESS

2. A instalação do LESS foi concluída
3. Teste o compilador utilizando o comando no terminal:

```
lessc -version
```

```
B18:~ manha$ lessc -version
lessc 1.7.5 (Less Compiler) [JavaScript]
```

# EP: Compilação LESS

4. Crie um arquivo de nome "test.less" no seu desktop
5. Adicione (e salve) o seguinte código LESS:

```
@color: red;
.myClass {
 color: @color;
 border-color: @color;
}
```

# EP: Compilação LESS

6. No terminal, navegue até o desktop:

```
cd ~/Desktop/
```

7. Digite o seguinte comando:

```
lessc test.less test.css
```

8. Abra e verifique o conteúdo do arquivo "test.css"

## Integração com Editor

- A instalação dos recursos de comandos no terminal permitem a integração com o editor de código
  - Essencial ter instalado os recursos por comandos no terminal para integrar com o editor
  - Sem estes recursos previamente instalados não é possível integrar com o editor
  - Editor cria os comandos de terminal dinamicamente

# EP: Pre Processor

1. Abra o Sublime Text e instale os seguintes pacotes
  - a. LESS
  - b. LESS-Build
2. Crie um arquivo com código e extensão LESS

```
@cor: red;
div{ color: @cor; }
```

# EP: Pre Processor

3. Salve e compile o arquivo
  - a. Menu clique em “Tools/Build System/LESS”
  - b. CMD / CTRL + B
4. Windows:
  - a. C:\Users\{USER}\AppData\Roaming\npm\lessc.cmd

# Aninhamento

## Aninhamento

- Permite aninhar declarações CSS
  - Facilita a utilização e interpretação da árvore hierárquica
  - Reduz o número de instruções digitadas (instruções de seletores não necessitam ser repetidas - menor possibilidade de erro de digitação)
  - Como tudo no LESS, seu uso é opcional
- Pode ser usada com qualquer seletor válido CSS
  - Lembre-se LESS continua sendo CSS, apenas adicionou mais recursos a linguagem

# Aninhamento

- No caso do seletor "Descendant Selector" do CSS, basta colocar o seletor "pai" com o seletor "filho" aninhada

# Aninhamento

```
/* LESS */
.parent{
 .child{
 .granChild{
 color: green;
 }
 }
}
```

```
/* CSS */
.parent .child .grandChild{
 color: green;
}
```

# Aninhamento

```
/* LESS */ /* CSS */
div{ div{
 color: red; color: red;
 h1{ }
 color: blue; div h1{
 } color: blue;
} }
```

## EP: Aninhamento

1. Crie os arquivos HTML e LESS de nome "nesting"
  - a. Vincule o arquivo CSS ao HTML
2. Gere o código através da instrução do Emmet:

```
div.parent>h1.child{Header}+p>lorem*3
```

## EP: Aninhamento

3. Adicione o seguinte código no arquivo LESS:

```
.parent {
 color: red;
}
.parent .child{
 color: blue;
}
```

4. Salve, compile e teste

## EP: Aninhamento

5. Converta a instrução anterior para o formato LESS:

```
.parent {
 color: red;
 .child{
 color: blue;
 }
}
```

6. Salve, compile e teste

# EP: Aninhamento

7. Utilizando a técnica de aninhamento LESS e seletor hierárquico CSS, modifique a cor do segundo elemento <span>

## Demais Seletores

- Os demais seletores avançados funcionam da mesma forma que o anterior
  - A instrução será dividida entre "pai" e "filho"
  - Ex: Adjacent Sibling Selector, Sibling Selector, Child Selector, etc

# Demais Seletores

```
/* LESS */
div{
 > div{color: red;}
 + div{color: blue;}
}

/* CSS */
div > div{
 color: red;
}
div + div{
 color: blue;
}
```

## &

- Repete o "parent selector"
  - Muitas vezes não é necessário, a repetição ocorrerá como esperado
  - Eventualmente será necessário utilizar o caractere "&" para compor o seletor desejado
- Uso principal é para aninhamento de
  - pseudo-class → `div:hover`
  - pseudo-elements → `p::first-line`

# Aninhamento

```
/* LESS */
.parent{
 & .child{
 color: red;
 }
}
```

```
/* CSS */
.parent .child{
 color: red;
}
```

# Aninhamento

```
/* LESS */
div{
 &:hover{
 background-color: red;
 }
}
```

```
/* CSS */
div:hover{
 background-color: red;
}
```

## EP: Aninhamento

1. Volte ao arquivo "nesting.html"
2. Adicione o seguinte HTML usando Emmet:

```
(div.product>img[src="http://lorempixel.com/200/200/cats/$"])*10
```

## EP: Aninhamento

3. Adicione o seguinte código no arquivo "nesting.less":

```
.product{
 background-color: #333;
 margin: 10px;
 padding: 20px;
 width: 200px;
 height: 200px;
 float: left;
}
```

# EP: Aninhamento

4. Dentro da classe ".product" adicione o aninhamento:

```
.product{
 ...
 &:hover{ background-color: red; }
}
```

5. Salve, compile e teste
  - a. Analise o CSS resultante
6. (Opcional) Utilize variáveis LESS

## Variáveis

# Variáveis

- Como visto anteriormente, variáveis podem ser criadas para facilitar a manutenção do código
  - Seu uso deve ser considerado sempre que houver a repetição de valores
- Dependendo do seu valor podem ser usadas em cálculos e concatenações

# Variáveis

```
/* LESS */
@gap: 5px;

div{
 margin: @gap;
 padding: @gap * 2;
}

/* CSS */
div{
 margin: 5px;
 padding: 10px;
}
```

## EP: Variáveis

1. Crie e vincule os arquivos HTML, LESS e CSS de nome "variables"
2. Adicione o seguinte HTML usando Emmet:

a. A instrução do Emmet deve ser feita em uma única linha  
`(div.item>img[src="http://lorempixel.com/100/150/people/$"]+h3{People $})*10`

## EP: Variáveis

3. No arquivo LESS, adicione o seguinte código:

```
@gap: 5px;
@background: #999;
```

# EP: Variáveis

4. No arquivo LESS, adicione o seguinte código:

```
.item{
 background-color: @background;
 margin: @gap;
 padding: @gap * 2;
 float: left;
}
```

5. Salve, compile e teste

# Escopo

- Como no JavaScript, possuem escopo
  - Global → fora das chaves de qualquer seletor
  - Local → dentro das chaves do seletor

# Escopo

```
/* LESS */
@color: red;

div{
 h1 {
 color: @color;
 }
}
```

```
/* CSS */
div h1{
 color: red;
}
```

# Escopo

```
/* LESS */
@color: red;

div{
 @color: blue;
 h1 {
 color: @color;
 }
}
```

```
/* CSS */
div h1{
 color: blue;
}
```

# EP: Escopo

1. Crie e vincule os arquivos HTML, LESS e CSS de nome "scope"
2. Adicione o seguinte HTML usando Emmet:

```
div*2
```

# EP: Escopo

3. Adicione as instruções LESS:

```
@color: red;
div{
 margin: 10px;
 width: 300px; height: 200px;
 background-color: @color;
}
```

4. Salve, compile e teste

# EP: Escopo

5. Altere o código LESS para:

```
div{
 ... background-color: @color;
 &:last-of-type{
 @color: blue;
 background-color: @color;
 }
}
```

6. Salve, compile e teste

## Interpolação

- Muitas vezes será necessário utilizar o valor de uma variável para compor um resultado final
- LESS permite interpolação de valores como:
  - Seletor
  - Nome de uma propriedade CSS
  - Parte do nome de uma propriedade
  - Parte do "string" usado em uma URL

# Interpolação (LESS)

```
@fade: opacity;
@classname: myClass;
@path: "../images";
@property: color;

.@{classname} {
 @{fade}: 0.5;
 border-@{property}: red;
 background: url('@{path}/back.png');
}
```

# Interpolação (CSS)

```
.myClass{
 opacity: 0.5;
 border-color: red;
 background: url('../images/back.png');
}
```

# EP: Interpolação

1. Crie e vincule os arquivos HTML, LESS e CSS de nome "interpolation"
2. Adicione o seguinte HTML usando Emmet:

```
div.item
```

# EP: Interpolação

3. Adicione as seguintes variáveis LESS:

```
@path: 'http://lorempixel.com';
@size: 100;
@category: 'fashion';
```

# EP: Interpolação

4. Após as variáveis, adicione a instrução LESS:

```
div.item {
 width: 100px; height: 100px;
 background-color: #ccc;
 background-image: url('@{path}/{size}/{size}/{category}');
}
```

5. Salve, compile e teste

## Concatenação

- A interpolação gera uma concatenação do valor da variável com o texto seguinte
  - Desde que esteja dentro de aspas como texto
  - O resultado será um valor concatenado dentro de um texto
- Alguns casos o resultado como texto não atenderá as necessidades das instruções
  - Necessário remover as aspas, para que o valor seja considerado

# Concatenação

```
/* LESS */
@size: 100;

div{
 width: @size}px;
 height: @size}px;

}
```

```
/* CSS */
Error !!!
```

# Concatenação

```
/* LESS */
@size: 100;

div{
 width:'@size}px';
 height:'@size}px';

}
```

```
/* CSS */
div{
 width: '100px';
 height: '100px';
 /* Invalid value */
}
```

# Evaluate

- LESS permite interpretar um texto como valor
  - Efetua a conversão do valor em texto para valor CSS
  - Consequentemente remove as aspas do valor
- O caractere ~ interpreta o texto e converte para valor:  
`~'@{variable}'`

# Evaluate

```
/* LESS */ /* CSS */
@size: 100;

div{ width: 100px;
 width:~'@{size}px'; height: 100px;
 height:~'@{size}px'; }
}
```

# EP: Evaluate

1. Abra o arquivo "interpolation.less" do EP anterior
2. Altere a atual instrução de largura e altura para:

```
width:~'@{size}px';
height:~'@{size}px';
```

3. Salve, compile e teste

## :extend( )

# :extend( )

- Pseudo-classe exclusiva do LESS
- Permite "estender" uma declaração CSS
  - Copia o conjunto de regras de uma declaração para outra
- Evita a repetição de seletores
  - Adiciona uma vírgula ao seletor estendido com a nova classe

# LESS

```
.gap { margin: 10px; padding: 5px; }
```

```
.item:extend(.gap) {
 color: blue;
}
```

# CSS

```
.gap, .item { margin: 10px; padding: 5px; }
```

```
.item {
 color: blue;
}
```

## Múltipla Extensão

- Pode ser aplicada a vários seletores separando por vírgula:

```
.myClass:extend(.gap,.font,.theme) { ... }
```

## EP: extend( )

1. Crie e vincule os arquivos HTML, LESS e CSS de nome "extend"
2. Adicione o seguinte HTML usando Emmet:

```
div.item-$*3>lorem
```

## EP: extend( )

3. No arquivo LESS adicione:

```
.gap { margin: 10px; padding: 5px; }
.item-1:extend(.gap) { color:red; }
.item-2:extend(.gap) { color:green; }
.item-3:extend(.gap) { color:blue; }
```

4. Salve, compile e teste

## EP: extend( )

5. Crie uma nova classe:

```
.font {font-size: 24px; font-family: Arial}
```

6. Aplique nas classes dos itens:

- a. Aplique à todos conforme o exemplo abaixo

```
.item-1:extend(.gap, .font) { ... }
```

7. Salve, compile e teste

## &:extend( )

- Permite aplicar à múltiplos seletores utilizando aninhamento:

```
.item-1, item-2, item-3 {
 &:extend(.gap,.font){ ... }
}
```

# Pseudo-Class CSS

- Possível ainda utilizar a extensão com pseudo-class do CSS
  - A instrução de estender deve ser sempre a última

```
.item-1:hover:extend(.gap) { ... }
```

## EP: Pseudo-Class CSS

1. Abra o arquivo do EP anterior
2. Crie um estado "on":

```
.on {
 background-color: tomato;
}
```

# EP: Pseudo-Class CSS

3. Adicione o aninhamento e a "pseudo-class"

```
item-1, item-2, item-3{
 &:hover:extend(.on) { color: white; }
}
```

4. Salve, compile e teste

# Mixins

## .mixin( )

- Permite incorporar a declaração CSS de uma classe em outro seletor
  - Difere do "extend( )" que utiliza uma vírgula para implementar o código CSS
- Tem funcionamento análogo à uma função ou método
  - Pode ser invocado quantas vezes forem necessárias
  - Pode receber parâmetros

## .mixin( )

- A classe de origem CSS receberá um parenteses:

```
.myClass() { ... }
```

- O seletor que recebe deve invocar o "mixin":

```
.otherClass {
 .myClass();
}
```

# LESS

```
.gap() { margin: 10px; padding: 5px; }
```

```
.item {
 .gap();
 color: blue;
}
```

# CSS

```
.item {
 margin: 10px;
 padding: 5px;
 color: blue;
}
```

## EP: Mixin

1. Crie e vincule os arquivos HTML, LESS e CSS de nome "mixin"
2. Adicione o seguinte HTML usando Emmet:

```
div.item-$*3>lorem
```

## EP: Mixin

3. Adicione o "mixin":

```
.gap() { margin: 10px; padding: 5px; }
```

4. Invoque no seletor:

```
.item-1 { .gap(); color: red; }
.item-2 { .gap(); color: green; }
.item-3 { .gap(); color: blue; }
```

5. Salve, compile e teste

# Aninhamentos

- Mixins podem conter blocos aninhados:
  - Cram seletores CSS como "descendant selectors"  
`selector selector`
  - Se utilizado o caractere "&" cram seletores combinados  
`selector.selector`

## LESS (descendant)

```
.product() {
 .dvd { color: gold; }
 .book { color: #ccc; }
}
.item { .product(); }
```

## CSS (descendant)

```
.item .dvd {
 color: gold;
}

.item .book {
 color: #ccc;
}
```

## LESS (combo)

```
.product() {
 &.dvd { color: gold; }
 &.book { color: #ccc; }
}

.item { .product(); }
```

## CSS (combo)

```
.item.dvd {
 color: gold;
}

.item.book {
 color: #ccc;
}
```

## EP: Mixins com Aninhamentos

1. Crie e vincule os arquivos HTML, LESS e CSS de nome "mixin\_nesting"
2. Adicione o seguinte HTML usando Emmet:

```
div.item>div.dvd{DVD}+div.book{BOOK}
```

# EP: Aninhamentos

3. Adicione o "mixin":

```
.product() {
 .dvd { color: gold; }
 .book { color: #ccc; }
}
```

# EP: Aninhamentos

4. Adicione o "mixin":

```
.item {
 .product();
}
```

5. Salve, compile e teste

# Parâmetros

- Informações podem ser enviadas aos "mixins" utilizando parâmetros
  - Similar aos argumentos passados à uma função de uma linguagem de programação tradicional
- Permitem maior reutilização e configuração de código
  - Resultado pode ser controlado e obtido com base no parâmetro enviado

# Parâmetros

- O "mixim" deve receber o parâmetro com a declaração similar ao de uma variável LESS

```
.mixin(@param) { ... }
```

# Parâmetros

- Podem ser obrigatórios ou opcionais
  - Dependerá se um valor padrão foi atribuído (opcional) ou não (obrigatório)

```
.mixin(@param: 10px) { ... }
```

# Parâmetros

- Mais de um parâmetro pode ser passado, separando por ponto e vírgula
  - Quando existirem parâmetros obrigatórios e opcionais, os opcionais devem ser sempre os últimos

```
.mixin(@param1; @param2; @param3: 5) { ... }
```

```
.myClass{.mixin(12px; blue; 10)}
```

# LESS

```
.gap (@value) {
 margin: @value;
 padding: @value / 2;
}
.font (@size: 2em) {
 font-size: @size;
}
```

# LESS

```
.item {
 .gap(12px);
 .font();
}
```

# CSS

```
.item {
 margin: 12px;
 padding: 6px;
 font-size: 2em;
}
```

## EP: Parâmetros

1. Crie e vincule os arquivos HTML, LESS e CSS de nome "mixin\_param"
2. Adicione o seguinte HTML usando Emmet:

```
div.item*5>p>lorem
```

# EP: Parâmetros

- Crie os "mixins":

```
.gap(@value) {
 margin: @value;
 padding: @value / 2;
}
.font(@size: 2em) {
 font-size: @size;
}
```

# EP: Parâmetros

- Use os "mixins" com os seguintes parâmetros:

```
.item {
 .gap(12px);
 .font();
}
```

- Salve, compile e teste

# Sobrecarga

- O LESS permite a sobrecarga de "mixins"
  - Muitas linguagens de programação robustas também possuem este recurso
- Um "mixin" pode ser declarado mais de uma vez com "assinaturas" diferentes
  - Mesmo nome deve ser usado
  - Quantidade de parâmetros deve ser diferente

# LESS

```
.gap(@value) {
 margin: @value;
 padding: @value / 2;
}
.gap(@margin; @padding){
 margin: @margin;
 padding: @padding;
}
```

# LESS

```
.item-1 {
 @gap(10px);
}

.item-2 {
 @gap(13px, 8px);
}
```

# CSS

```
.item-1 {
 margin: 10px;
 padding: 5px;
}

.item-2 {
 margin: 13px;
 padding: 8px;
}
```

# EP: Sobrecarga

1. Crie e vincule os arquivos HTML, LESS e CSS de nome "mixin\_overload"
2. Adicione o seginte HTML usando Emmet:

```
div.item-$*2>p>lorem
```

# EP: Sobrecarga

3. Adicione os "mixins":

```
.gap(@value) {
 margin: @value;
 padding: @value / 2;
}
.gap(@margin; @padding) {
 margin: @margin;
 padding: @padding; }
```

# EP: Sobrecarga

4. Utilize os "mixins":

```
.item-1 { .gap(10px); }
.item-2 { .gap(13px; 8px); }
```

5. Salve, compile e teste

# Funções

# Sobre

- Recursos do LESS que permitem realizar operações:
  - Matemáticas
  - Transformação
  - Operações condicionais
  - Loops

## Matemáticas

- Operações seguem as regras tradicionais de cálculos matemáticos
  - Ex: parenteses determinam ordem, multiplicação antes de soma, etc

# Matemáticas

```
@border: 10px;
@padding: 20px;
@gap: (@border + @padding) / 2;
```

## Cálculos com Unidades

- Se um valores contiver unidade de medida
  - Tudo será convertido para esta unidade
  - Se nenhuma unidade de medida for usada o valor será numérico
- Se os números tiverem mais de uma unidade diferentes
  - Elas poderão ser convertidas automaticamente
  - Desde que compatíveis e as operações forem de soma ou subtração

# Cálculos com Unidades

- Unidades compatíveis:
  - A primeira unidade usada na expressão será a unidade resultante

<i>Tipo</i>	<i>Unidades</i>
Comprimentos	px, m, cm, mm, in, pt, pc
Tempo	s, ms
Ângulos	deg, rad, grad, turn

# Cálculos com Unidades

```
@number: 1 + 2 + 3 * 4;
@pixel: 2 + 4px + 5;
@cm: 2cm + 4px + 5mm;
@pt: 2pt + 4px + 5mm;
```

# Funções Built-in

- Funções definidas pelo LESS para:
  - Arredondar
  - Percentual
  - Conversão de Unidades
  - Trigonométricas
  - Transformações
  - Formatação de Texto

## Arredondar

<i>Tipo</i>	<i>Exemplo</i>	<i>Resultado</i>	<i>Descrição</i>
ceil(number)	ceil(5.1)	6	Arredonda o valor para cima
floor(number)	floor(5.9)	5	Arredonda o valor para baixo
round(number)	round(5.5) round(5.4)	6 5	Arredonda o valor para cima se maior ou igual à x.5 e para baixo se menor ou igual à x.4

# Porcentagem

<i>Tipo</i>	<i>Exemplo</i>	<i>Resultado</i>	<i>Descrição</i>
percentage(number)	percentage(0.5)	50%	Multiplica o valor por 100 e acrescenta a unidade %

# Conversão de Unidades

- Nem sempre é possível ou recomendado usar transformação automática
  - Muitas vezes é desejável acrescentar uma unidade em um número
  - Ou ainda trocar a unidade sem fazer conversão
- O LESS possui funções específicas para a conversão de uma unidade para outra
  - Devem ser compatíveis

# Conversão de Unidades

<i>Tipo</i>	<i>Exemplo</i>	<i>Resultado</i>	<i>Descrição</i>
convert(n, u)	convert(1in, cm)	2.54cm	Converte o valor, calculando seu valor, para a unidade especificada
[OBJ]unit(n, u)	[OBJ][OBJ][OBJ]unit (1in, cm)	1cm	Converte o valor, sem efetuar nenhum cálculo, para a unidade especificada

*n = número (com ou sem unidade)*

*u = unidade compatível*

## convert( )

- Se as unidades passadas para não forem compatíveis ele funcionará igual a "unit( )" [OBJ]
  - Simplesmente copiando o mesmo valor e trocando a unidade
- A função não suporta a conversão de pixels (px) para:
  - em
  - rem

# Coleções

- Funções que operam sobre listas
- Uma lista é um conjunto de valores separada por vírgulas ou espaços
  - Similar à um "array" de uma linguagem de programação tradicional
  - Muitas propriedades do CSS e seletores podem ser tratados como listas (podem ser usadas em loops)

# Coleções

<i>Tipo</i>	<i>Exemplo</i>	<i>Resultado</i>	<i>Descrição</i>
length( <i>l</i> )	@list: 10px 5px 8px length(@list)	3	Retorna a quantidade de elementos existentes na lista
extract( <i>l</i> , <i>p</i> )	@sizes: 36pt 52pt 18pt; extract(@sizes, 1)	36pt	Extrai valor da lista contido na posição informada
min( <i>l</i> )	min(@sizes)	18pt	Extrai o menor valor da lista
max( <i>l</i> )	max(@sizes)	52pt	Extrai o maior valor da lista

*l* = lista com itens  
*p* = posição

# Unidades

<i>Tipo</i>	<i>Exemplo</i>	<i>Verificação do Valor</i>
iscolor(v)	iscolor(red)	Cor (nome, rgb, rgba, hsl, hexadecimal, etc)
iskeyword(v)	@param: .item; iskeyword(@param)	Qualquer nome diferente de cor, número ou URL, retornará "true"
isnumber(v)	isnumber(123)	Número

v = valor

# Unidades

<i>Tipo</i>	<i>Exemplo</i>	<i>Verificação do Valor</i>
isstring(v)	isstring('text')	Texto (deve estar entre aspa ou apóstrofe)
isurl(v)	@param: url("images/logo.png"); isurl(@param)	Valor é URL

v = valor

# Unidades

<i>Exemplo</i>	<i>Resultado</i>
isstring('12345')	true
isstring(12345)	false
iskeyword(.item)	true
iskeyword(red)	false
isnumber(12.77)	true
isnumber(~'123')	false

# Unidades

<i>Exemplo</i>	<i>Resultado</i>
iscolor(red)	true
iscolor(~'rgba(100%,100%,100%,1)')	false
isurl(url(image.jpg))	true
isurl(~'/'image.jpg')	false

# Unidades Específicas

Tipo	Exemplo	Verificação do Valor
ispixel(v)	ispixel(12px)	Valor "pixels"
isem(v)	isem(2em)	Valor "em"
ispercentage(v)	ispercentage(50%)	Valor em porcentagem
isunit(v, u)	@param: 12px; isunit(@param, px)	Valor equivale ao especificado pelo segundo parâmetro com a unidade

v = valor  
u = unidade

# Mixins Guards

# Sobre

- Permite iterar ("loop")
  - Similar à um "for" em linguagens de programação tradicionais
- Criar condições
  - Similar à um "if/else" em linguagens de programação tradicionais

# Condisional

- Acrescenta-se à um "mixin" LESS a palavra chave "when"
  - Avaliará a condição definida
  - A condição deve resultar em "true" ou "false"

```
.mixin(@param) when (@param = red) { ... }
```

# Operadores Lógicos

<i>Operador</i>	<i>Uso</i>
Igualdade	(value = value)
Maior	(value > value)
Menor	(value < value)
Maior Igual	(value >= value)
Menor Igual	(value <= value)

# Operadores Lógicos

<i>Operador</i>	<i>Uso</i>
IF	<b>when</b> (condition)
ELSE	<b>when (default( ))</b>
Negando	<b>when not</b> (condition)
Ou	(condition) , (condition)
E	(condition) <b>and</b> (condition)

# Condicional

- Condição pode ser usada com operadores lógicos
  - And → and
  - Or → vírgula
  - Not → not

```
.mixin(@param) when not (@param = red) { ... }
```

# Condicional

```
.page(@bg) when (@bg = black) {
 background: @bg;
 color: white;
}
.page(@bg) when (@bg = white) {
 background: @bg;
 color: black;
}
```

# Condicional

```
.page(@bg) when (@bg = red) , (@bg = blue) {
 background: @bg;
 color: white;
}
```

## EP: Condicional

1. Crie e vincule os arquivos HTML, LESS e CSS de nome "mixin\_conditional"
2. Adicione o seguinte HTML usando Emmet:

```
div.item-$*4>p>lorem
```

# EP: Condicional

3. Adicione o seguinte "mixin":

```
.page(@bg) when (@bg = black) {
 background-color: @bg;
 color: white;
}
```

# EP: Condicional

4. Adicione outro "mixin" usando sobrecarga:

```
.page(@bg) when (@bg = white) {
 background: @bg;
 color: black;
}
```

# EP: Condicional

5. Adicione as seguintes classes:

```
.item-1 { .page(black) }
```

```
.item-2 { .page(white) }
```

6. Salve, compile e teste

# EP: Condicional

7. Adicione um novo "mixins":

```
.page(@bg) when (@bg = red), (@bg = blue){
 background: @bg;
 color: white;
}
```

# EP: Condicional

8. Adicione um novo "mixins":

```
.item-3 { .page(red) }
```

9. Salve, compile e teste

# EP: Condicional

10. Adicione um "mixin" padrão:

```
.page(@bg) when (default()) {
 background: @bg;
 color: black;
}
```

# EP: Condicional

11. Adicione um novo "mixins":

```
.item-4 { .page(gold) }
```

12. Salve, compile e teste

## Loops

- São chamadas recursivas de mixins
  - Um mixin invoca ele mesmo
- Para que um loop não seja infinito, é necessário que:
  - Cada repetição uma condição seja testada
  - Valor testado mude de forma que em algum momento a condição seja falsa (e a recursividade tenha fim)

# Loops

```
.loop(@i) when (@i > 0) {
 .loop(@i - 1);
}

.item {
 .loop(10);
}
```

## EP: Loop

1. Crie e vincule os arquivos HTML, LESS e CSS de nome "mixin\_loop"
2. Adicione o seguinte HTML usando Emmet:

- a. Adicione um tamanho de 200 pixels para todas as <div>

```
div.item-$*10
```

# EP: Loop

3. Adicione o seguinte "mixin":

```
.getSelector(@index) when (@index > 0){
 .getSelector(@index - 1);
 .item-@{index}{
 background-image:
 url('http://lorempixel.com/200/200/nature/@{index}');
 }
}
```

# EP: Loop

4. Invoque o "mixin":

```
.getSelector(10);
```

5. Salve, compile e teste

# Import

## Sobre

- Importar arquivos permitem uma manutenção mais clara e concisa
  - Quanto menor for o arquivo, menor será o trabalho para entender o código
  - Arquivos com estilos específicos podem ser criados, sem misturar códigos para funcionalidades diferentes (Ex: tema, layout, responsividade, etc)

# @import CSS

- A instrução “@import” do CSS tradicional tem seu funcionamento prejudicado pelo carregamento do navegador
  - Tempo de carregamento do arquivo principal CSS é diferente dos importados
  - Página pode “piscar” sem estilo até que o(s) arquivo(s) importado(s) seja(m) carregado(s)

# @import LESS

- No LESS não existe risco da página ter tempo de carregamento diferente
  - Por padrão o LESS gera um arquivo único CSS que conterá todos os arquivos importados
  - Por ser um arquivo único CSS, não ocorrerá o problema do tempo de carregamento

# @import LESS

- No Importante é importar arquivos com a extensão “.less” (ou ainda omiti-la)
  - Se utilizado arquivos com a extensão “.css”, a instrução será usada de forma tradicional e o problema de carregamento persistirá

```
@import "theme";
@import "responsive.less";
```

# Configurando Parâmetros

- A instrução “@import” do LESS permitem alguns parâmetros e configurações
  - Parâmetros são adicionados entre parenteses logo após a declaração do comando e antes do nome do arquivo

```
@import (param) "file";
```

# Parâmetros

Parâmetro	Descrição
css	Trata o arquivo como CSS (independente de sua extensão)
inline	Inclui o conteúdo do arquivo no CSS final, mas não o processa
less	Trata o arquivo como LESS (independente de sua extensão)
multiple	Aceita que um arquivo seja importado mais de uma vez
once	Permite importar o arquivo uma vez e ignora outras tentativas
reference	Inclui e usa o arquivo LESS mas não utiliza o conteúdo na saída

## EP: @import

1. Crie os seguintes arquivos:
  - a. style.less
  - b. theme.less
  - c. utils.less
  - d. reset.css
2. No arquivo “reset.css” adicione o código “css reset”

<http://meyerweb.com/eric/tools/css/reset/reset.css>

# EP: @import

3. Adicione a seguinte instrução no arquivo “style.less”:

- a. Verifique o resultado
- b. Perceba que a instrução foi matida para um CSS tradicional

```
@import "reset.css"
```

4. Utilize o parâmetro no comando e observe o resultado

- a. Desta vez o conteúdo foi adicionado ao arquivo “style.css”

```
@import (less) "reset.css"
```

# EP: @import

5. Abra o arquivo “theme.less” e adicione o código:

```
@color: #333;
@background: tomato;

body {
 background-color: @background;
 color: @color;
}
```

## EP: @import

6. Abra o arquivo “style.less” e importe o arquivo “theme.less” do passo anterior
  - a. Perceba que desta vez não foi necessário o parâmetro “(less)” para que o código fosse adicionado

```
@import "theme.less"
```

## EP: @import

7. Abra o arquivo “utils.less” e adicione o código:

```
.getPrefix(@prorperty;@value) {
 -webkit-@{prorperty}: @value;
 -moz-@{prorperty}: @value;
 -ms-@{prorperty}: @value;
 -o-@{prorperty}: @value;
 @{prorperty}: @value;
}
```

## EP: @import

8. Abra o arquivo “style.less” e importe o arquivo “utils.less” do passo anterior

- a. Uma vez que o arquivo “utils.less” contém apenas “mixin” não existe real necessidade de colocar o parâmetro “(reference)”

```
@import (reference) "utils.less"
```

## EP: @import

9. Utilize o “mixin” no arquivo “style.less”:

```
.rotate {
 .getPrefix(transform;rotate(180deg));
}
.transition{
 .getPrefix(transition; width 2s)
}
```

# Less vs Sass

<https://gist.github.com/chriseppstein/674726>

