

# AUTOMAÇÃO FRONT END

---

Linha de Comando

## Linha de Comando (CLI)

- Interagir com o Computador através de texto

Comando começando com \$ (Unix) ou > (Win)

- UNIX (Mac e Linux) vs Windows

Bash e Zsh (Unix) | CMD e PowerShell (Windows)

- <https://ss64.com>

## Movimentar (Bash)

- **\$ cd sites**

Move para a pasta sites

- **\$ cd ..**

Move para a pasta anterior

- **\$ cd ~/Desktop/sites**

Move para a pasta sites dentro de Desktop

- **\$ cd ~**

Move para o diretório principal do sistema / home

- **\$ clear**

Limpa a tela

## Listar e Criar (Bash)

---

- `$ ls`

Lista diretórios e arquivos

- `$ ls -all`

Lista diretórios, arquivos, invisíveis e detalhes

- `$ mkdir site`

Cria o diretório site

- `$ touch index.html`

Cria o arquivo index.html

## Remover (Bash)

- `$ rm index.html`

Remove o arquivo index.html

- `$ rm -r site`

Remove o diretório site e todos os arquivos dentro dele

- `seta para cima / baixo`

Acessa o comando anterior

- `tab`

Auto-completa o comando

# AUTOMAÇÃO FRONT END

---

NPM

## NPM (Node Package Manager)

- Gerenciador de Pacotes

Feito para Node.js

- Linha de comando

```
$ npm install lodash
```

- <https://www.npmjs.com/>

## Instalar Pacotes

---

- **\$ npm install lodash**

Instala o pacote lodash no diretório atual da linha de comando

- **\$ npm install eslint -g**

Instala o eslint globalmente (No mac é necessário o sudo para instalar globalmente.)

- **\$ npm update lodash**

Atualiza o pacote lodash

- **\$ npm uninstall lodash**

Desinstala o pacote lodash

## Package.json

---

- **package.json**

Arquivo local com as configurações e dependências de pacotes NPM

- **\$ npm init**

Inicia uma nova configuração local do npm.

- **\$ npm install**

Instala todas as dependências listadas no arquivo package.json

## Arquivos Invisíveis

---

- Windows

Exibir > Itens Ocultos

- Mac

```
$ defaults write com.apple.Finder AppleShowAllFiles true  
$ killall Finder
```

# AUTOMAÇÃO FRONT END

---

ESLint

## ESLint

---

- Evitar problemas

Indica a existência de possíveis padrões problemáticos no código

- Definir padrões

Padronizar e manter consistência entre diferentes códigos JavaScript

- <https://eslint.org>

## Instalar ESLint

- `$ npm install eslint -g`

Instalar o eslint globalmente.

- `$ npm init`

Iniciar repositório NPM no local.

- `$ eslint --init`

Use Popular > Airbnb > (N) Enter > JSON > (Y) Enter

- Instalar extensão ESLint do VS Code

# AUTOMAÇÃO FRONT END

---

Webpack

## Webpack

---

- **Bundler**

Agrupa / processa diversos arquivos e otimiza os mesmos.

- Altamente configurável

Por isso é complexo.

- <https://webpack.js.org/>

## Webpack

---

- `$ npm install --save-dev webpack  
webpack-cli`

Instala o webpack e a cli do mesmo. Ter package.json antes.

- `$ npx webpack ./js/script.js --  
output ./main.js`

Irá agrupar todo o código, otimizar e mais. Utilizar `npx` é a mesma coisa que utilizar `node_modules/.bin/webpack`. Facilita para utilizarmos cli's instaladas localmente ao invés de globalmente.

## NPM Scripts

---

Permite definirmos uma linha de comando inteira, que será ativada com `npm run nomeScript`. Não precisamos utilizar o npx aqui.

```
"scripts": {  
    "dev": "webpack --mode development --watch ./js/script.js --  
output ./main.js",  
    "build": "webpack --mode production ./js/script.js --output  
./main.js"  
},  
  
// --mode define o modo da compilação  
// --watch define se deve ficar observando
```

## Scripts Externos

Podemos facilmente importar scripts externos instalando os mesmos através do NPM e utilizando o Webpack para fazer o bundler final.

```
$ npm install jquery  
$ npm install lodash
```

```
import $ from 'jquery';  
import _ from 'lodash';
```

```
$( 'nav' ).hide();  
  
_.difference(['Banana', 'Morango', 'Uva'], ['Banana',  
'Morango', 'Pêra']);  
// Uva
```

# AUTOMAÇÃO FRONT END

---

Babel

## Babel

---

- Compilador

Transforma código novo em código antigo. Ex: `const nome = 'Andre';`  
vira `var nome = 'Andre';`.

- Browser Suporte

Para que um browser possa suportar algo novo de JavaScript é preciso que ele esteja atualizado, mas nem todo usuário possui a última versão do browser instalada.

- Can I Use

O site <https://caniuse.com/> mostra em quais browsers a novidade está disponível ou não.

## Polyfill vs Transpiler

- **Polyfill**

Cria métodos / funções com o mesmo nome das atuais, porém utilizando código antigo para permitir o uso em browsers que não possuem a API.

- **Transpiler**

Transforma código novo em código antigo. Ou seja, transforma `const` em `var`.

## Instalar Babel

---

- <https://babeljs.io/docs/en/usage>

```
$ npm install --save-dev  
  @babel/core @babel/cli  
  @babel/preset-env
```

Instala o Babel, a CLI, e configurações pré definidas

- `$ npm install @babel/polyfill`

Instala os polyfill's

# webpack.config.js

Precisamos configurar o webpack, para utilizarmos o @babel/polyfill.

```
const path = require('path');

module.exports = {
  entry: ['@babel/polyfill', './js/script.js'],
  output: {
    path: path.resolve(__dirname, './'),
    filename: 'main.js'
  }
};
```

```
"scripts": {
  "dev": "webpack --mode development --watch",
  "build": "webpack --mode production"
},
```

## Fetch Polyfill

---

Nem todo browser suporta a Fetch API e por padrão o Babel não possui um polyfill para o Fetch. Podemos resolver isso instalando um polyfill externo.

```
$ npm install whatwg-fetch
```

```
const path = require('path');

module.exports = {
  entry: ['@babel/polyfill', 'whatwg-fetch', './js/script.js'],
  output: {
    path: path.resolve(__dirname, './'),
    filename: 'main.js'
  }
};
```

# AUTOMAÇÃO FRONT END

---

Git

## Git

---

- **Git**

Sistema de controle de versões. Facilita o trabalho em equipe e o controle de mudanças entre arquivos e diretórios.

- **Github**

Plataforma online de hospedagem para repositórios Git. Existem outras como GitLab e Bitbucket.

## Git Setup

---

- Instalar o Git

<https://git-scm.com/>

- Configurar Nome

```
$ git config --global user.name "Seu Nome"
```

- Configurar Email

```
$ git config --global user.email "email@gmail.com"
```

## Git Comandos

---

- `$ git init`

Inicia um repositório

- `$ git add style.css`

Adiciona o arquivo style.css ao index do git. Com o `$ git add -A`, adicionamos todos os arquivos.

- `$ git status`

Mostra os arquivos que tiveram mudanças.

- `$ git commit -m 'Descrição'`

Irá fazer fazer o commit do código adicionado com uma mensagem.

## Criar Repositório no Github

- Github

Criar conta: <https://github.com/>

- Novo Re却t髍io

<https://github.com/new>

- Adicionar diretório remoto

```
$ git remote add origin  
https://github.com/seuusuario/seurepositorio.git
```

- Push do primeiro commit

```
$ git push -u origin master
```

- Se for a sua primeira vez

## Branching

- **Branch**

Uma das principais vantagens do git é a possibilidade de criarmos 'ramificações'. Assim podemos trabalhar em funcionalidades adicionais para um projeto, sem modificarmos o 'ramificação principal', o master.

- **\$ git branch nomebranch**

Toda vez que formos adicionar uma nova funcionalidade, devemos iniciar criando um novo branch ao invés de fazermos alterações direto no master. O que for modificado no branch não afetara o master.

- **\$ git checkout nomebranch**

Irá mudar de branch. Podemos usar o atalho

`$git checkout -b novobranch`, assim ele cria e muda de branch ao mesmo tempo.

- **\$ git branch**

ORIGAMID

## Workflow

---

- Sempre crie um branch

Toda funcionalidade nova, crie um branch para desenvolver a mesma.

```
$ git checkout -b feature1
```

- Após o desenvolvimento e commit, vá até o master e veja se existem mudanças

```
$ git checkout master e $ git pull
```

- Volte para o branch e dê um merge com o master

```
$ git checkout feature1 e depois $ git merge master
```

- Conflitos

Se existirem conflitos você será avisado e deverá lidar com os mesmos

Após lidar com os conflitos faça o push do branch: `$ git push` e  
`$ git push --set-upstream origin feature1`.

## Lidando com Pull Request

---

- No Github

Agora você possui um novo branch no github e pode fazer o pull request (juntar ao master).

- Compare e Pull Request

Pode adicionar comentários. Create Pull Request.

- Merge Pull Request

Geralmente é o líder do projeto / responsável por fazer o review do seu código. Pode deletar o branch após o merge com o master.

## Mais Git

---

- **.gitignore**

Lista de arquivos que não devem ser manipulados pelo git. `node_modules` é um bom exemplo.

- **Commit sem texto**

Ao usar o `$ git commit` você entra no modo completo de comentário, com um editor de texto direto na linha de comando. Utilize `esc + :wq` para sair do mesmo.

- **Bitbucket**

Permite repositórios privados e gratuitos. <https://bitbucket.org/product>

## Github Pages

- Criar repositório

O nome deve ser `seuusuario.github.io`

- HTML Simples

O site só funcionará em html/css/js simples, sem linguagem de servidor

- Qualquer projeto

Qualquer projeto poderá ter uma página para o mesmo. Vá em Settings > GitHub Pages > selecione master branch e salve. E acesso `seuusuario.github.io/repositorio/`