

INSTITUTO FEDERAL

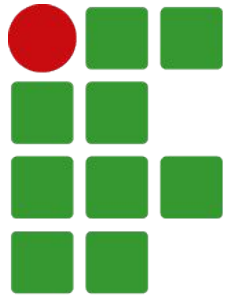
Paraná

Campus Paranaguá

Node.js e Electron

Desenvolvimento de Aplicativo para Desktop

Prof. Diego Stiehl



INSTITUTO FEDERAL

Paraná

Campus Paranaguá

Executando JavaScript fora do browser

Node.js

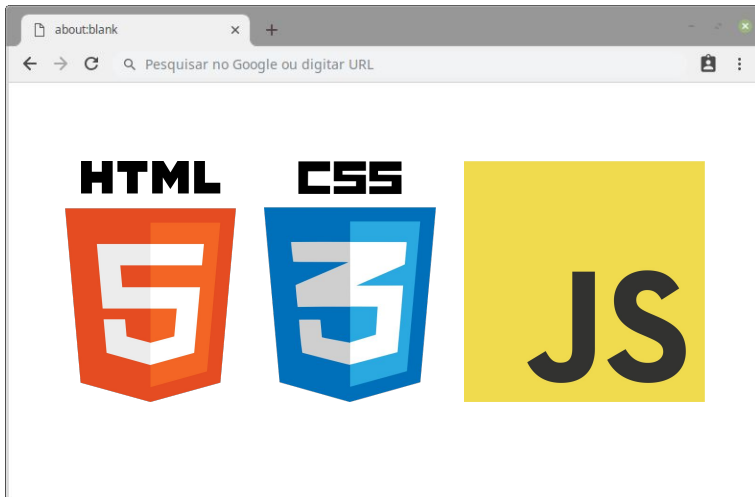
- Definição:
 - Node.js® é um runtime JavaScript desenvolvido com o Chrome's V8 JavaScript engine.



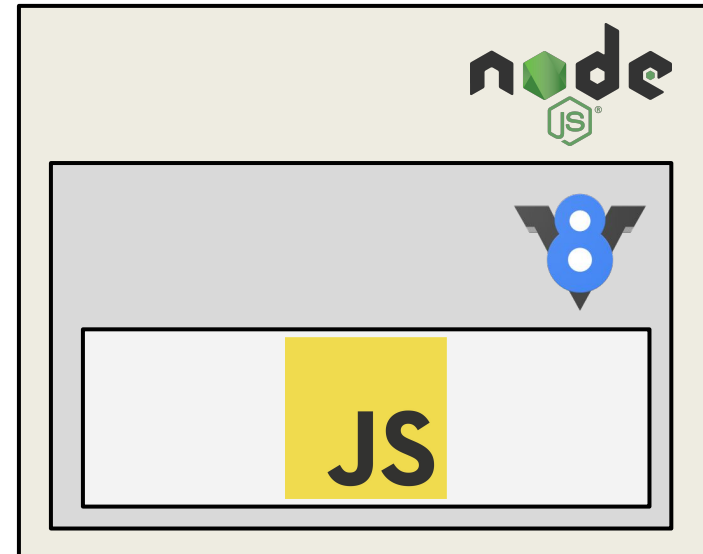
- Site: <https://nodejs.org>
- Criado em 2009

JavaScript fora do Browser

Browser do usuário



Qualquer computador
(ex: meu servidor)



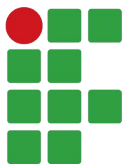
JavaScript no computador

- Com Node.js podemos executar JavaScript em qualquer computador
 - Antes ele era restrito ao browser
- Node.js estende as capacidades do JavaScript
 - Acessar sistema de arquivos
 - Melhoria nas capacidades de rede
 - Detalhes do sistema operacional
 - Módulos
 - ...

Possibilidades

- Node.js === novas possibilidades
- Principais:
 - Desenvolvimento para dispositivos móveis
 - React Native
 - Desenvolvimento para a web (server-side)
 - Desenvolvimento para desktop
 - Electron



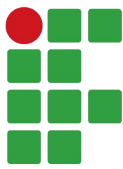


Por quê?

- Comunidade de desenvolvedores muito ativa
 - Muitas bibliotecas (NPM)
- Muitas empresas importantes usando
 - Netflix
 - Trello
 - Paypal
 - LinkedIn
 - Walmart
 - Uber
 - NASA

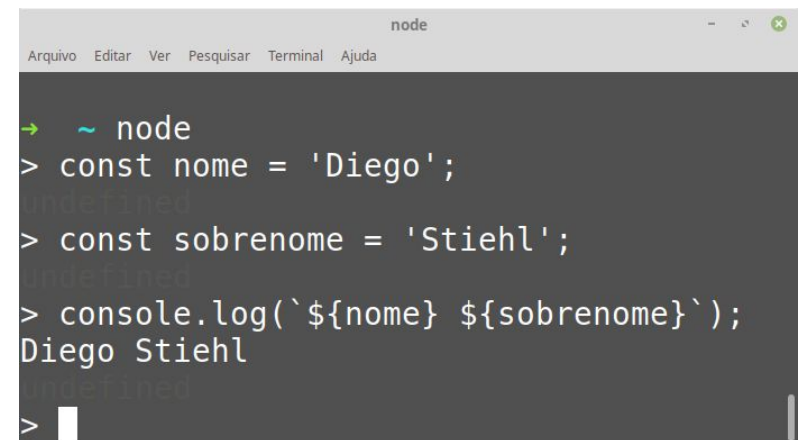
Instalação

- Acessar <https://nodejs.org>
- Baixar a última versão estável
 - Hoje 12.16.1 LTS
 - Cuidar com sistema operacional e 32/64 bits
- Instalar no seu sistema
 - Talvez precise reiniciar
- Observação
 - No laboratório já está instalado

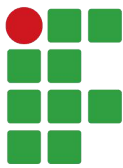


REPL

- O Node.js vem com um REPL
 - *Read-eval-print loop*
- Semelhante ao Console do browser
- Podemos executar código JavaScript
- Para abrir:
 - `node` ← No terminal
- Para sair:
 - Ctrl + D

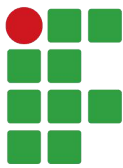


```
node
Arquivo  Editar  Ver  Pesquisar  Terminal  Ajuda
> ~ node
> const nome = 'Diego';
> const sobrenome = 'Stiehl';
> console.log(`${nome} ${sobrenome}`);
Diego Stiehl
>
```



Arquivo .js

- O comando **node** também é usado para executar nossos arquivos **.js**
- Executar:
 - `node arquivo.js`



Módulos

- Node.js provê muita modularidade de código
 - Bibliotecas
- Módulo é uma forma de exportar objetos JavaScript para outros usarem
- Podemos usar módulos:
 - Do próprio Node.js
 - Criados por nós
 - De terceiros

Usando Módulos

- Para usar um módulo usamos a sintaxe:

```
const modulo = require( 'nome-modulo' );
```

- O módulo da string precisa existir no meu ambiente
- A variável criada dá acesso a tudo que o módulo exporta
 - Funções, propriedades, objetos, ...



- **npm** (*Node Package Manager*)
- É um serviço de registro e distribuição de pacotes JavaScript
- Consiste em três componentes
 - Site (<https://www.npmjs.com>)
 - Command Line Interface (CLI)
 - Repositório de pacotes
- Mais informações:
 - <https://docs.npmjs.com/about-npm>



npm

- O **npm** permite o uso de módulos criados por terceiros
 - Busca, verifica versão, faz download, vincula ao nosso projeto, atualiza, ...
- Facilidade para instalar pacotes
- Gestão das dependências do nosso projeto
 - Facilita trabalho em equipe
- Possibilita a configuração e execução de scripts

Usando o npm

- Na pasta do projeto, digitar no terminal:
 - `npm init`
- Irá solicitar algumas configurações
 - Apenas confirme
- Ao final irá gerar um arquivo
 - `package.json`
 - Será utilizado para todas as configurações de dependências do projeto

package.json

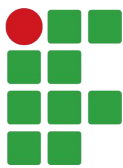
Arquivo gerado:

```
{  
  "name": "projeto",  
  "version": "1.0.0",  
  "description": "",  
  "main": "index.js",  
  "scripts": {  
    "test": "echo \"Error: no test specified\" && exit 1"  
  },  
  "author": "",  
  "license": "ISC"  
}
```


Instalando um pacote do npm

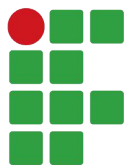
- Para instalar um pacote no nosso projeto:
 - 1. Busque no site: <https://www.npmjs.com>
 - 2. Execute o comando na pasta do projeto:
 - `npm install nome-do-pacote --save`
- O npm irá verificar, baixar e vincular o pacote
 - Ele irá modificar o arquivo package.json

```
"dependencies": {  
  "nome-do-pacote": "versão",  
  "outro-pacote": "versão",  
}
```



Pacote **cpf**

- Vamos instalar o pacote **cpf**
 - <https://www.npmjs.com/package/cpf>
 - Valida, gera e formata CPFs
- Executar
 - `npm install cpf --save`
- Ver package.json
- Ver diretório criado → `node_modules`
 - Todas nossas dependências ficam nele



Dicas

- npm install pode ser abreviado
 - npm i nome-do-pacote --save
- Ao baixar um projeto feito por outra pessoa
 - Executar:
 - npm install
 - Irá ler o package.json e baixar todos pacotes

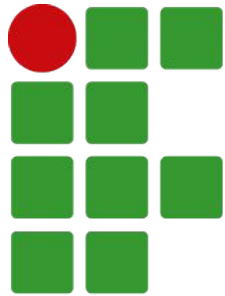
Usando pacote no projeto

- Pacotes normalmente exportam módulos
- Então:

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

```
const cpfGerado = cpf.generate(true);  
console.log(cpfGerado);
```

- Sempre leia a documentação do pacote



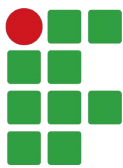
INSTITUTO FEDERAL

Paraná

Campus Paranaguá

Build cross-platform desktop apps with JavaScript, HTML, and CSS

Electron



Electron

- Framework para criação de aplicações Desktop utilizando apenas **HTML, CSS e JavaScript**
- Lançado em 2013
 - Nome original: Atom Shell
- GitHub
- Originado do projeto do editor Atom
- Open Source
- <https://electronjs.org>

Resumo da História

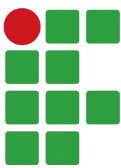
- Pessoas usam o GitHub para gerenciamento de código
- GitHub quer se envolver na escrita destes códigos
 - Vamos fazer um Editor!
- A maioria das pessoas no GitHub usa JavaScript
 - <https://madnight.github.io/githut>
- Vamos criar um editor em JavaScript
 - Facilmente extensível através de ainda mais JavaScript
- GitHub viu os benefícios de utilizar tecnologias da web no Desktop
 - Let's open source!!
 - Criado projeto Atom Shell (“core” do atom)
- Logo várias companhias adotaram
 - Muitos usuários contribuindo para o projeto

Multiplataforma

- Uma das maiores vantagens
- Electron executa em:
 - Linux
 - Windows
 - Mac
- Mesmo código pode ser reaproveitado

Como Funciona?

- Electron = Node.js + Chromium
- O **Node.js** permite a execução de código JavaScript fora do navegador
- O **Chromium** permite a renderização de interfaces gráficas ricas usando HTML e CSS



Chromium

- Versão open source do Google Chrome
 - Serve como “beta” de atualizações
- O Electron inclui o Chromium Content Module (CCM)
 - É o “core” do navegador
 - Foca no essencial para a renderização de páginas
 - Permite a requisição e renderização de HTML, carregamento e aplicação de CSS e também execução de JavaScript

O que eu preciso saber?

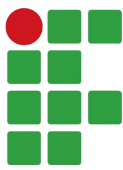
- Principal
 - HTML
 - CSS
 - JavaScript
 - EcmaScript 6 (e superior)
 - Node.js
- Secundário (para enriquecer a aplicação)
 - Frameworks CSS e JavaScript
 - React, Vue.js, Angular...

O que eu vou construir?

- Aplicações **desktop** multiplataforma que executarão (atualmente) em:
 - Linux
 - Windows
 - Mac

Atenção!!

- O que faremos é o desenvolvimento de **aplicações híbridas**
- Diferente de aplicações nativas criadas especificamente para cada plataforma
- Pode haver dificuldades
 - Menor desempenho (tempo de execução)
 - Acesso a recursos nativos
 - Adaptação da Experiência do Usuário (UX)
 - ...

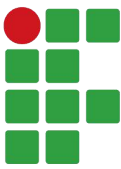


Chrome

- Abra o Monitor de Processos do sistema
- Olhe a quantidade de processos do Chrome
- Inicie mais abas
 - O que aconteceu?
- Aplicações em processos separados
 - Não se “conhecem”
- O Electron (Chromium) aplica este conceito
 - Cada janela é um processo separado

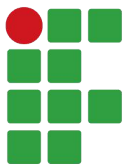
Criando Projeto com Electron

- Criar e entrar diretório “hello-electron”
- `npm init --yes`
- `npm install electron --save`
- Criar `index.js`
 - Principal script da aplicação (Main Process)
- Configurar script no `package.json`
 - Para iniciar aplicação
- `npm run electron`
 - Um app Electron é um app Node comum



package.json

```
{
  "name": "hello-electron",
  "version": "1.0.0",
  "description": "",
  "main": "index.js",
  "scripts": {
    "electron": "electron ."
  },
  "keywords": [],
  "author": "",
  "license": "ISC",
  "devDependencies": {
    "electron": "^8.0.2"
  }
}
```

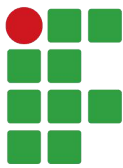



index.js

```
const { app, BrowserWindow } = require('electron');  
  
app.whenReady().then(() => {  
  console.log('Hello Electron!');  
});
```

Cadê a App?

- Por enquanto, fizemos a parte **Node** da nossa aplicação Electron
 - JavaScript no terminal (fora de um browser)
- Falta a interface gráfica
- Vamos chamar a parte **Chromium**
 - Instanciar uma nova BrowserWindow



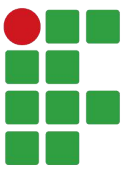
index.js

```
const { app, BrowserWindow } = require('electron');
```

```
const createWindow = () => {  
  let window = new BrowserWindow({  
    width: 800,  
    height: 600  
  });
```

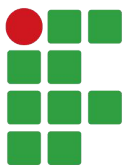
```
  window.loadFile('index.html');  
};
```

```
app.whenReady().then(createWindow);
```



index.html

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Electron</title>
  </head>
  <body>
    <h1>Hello Electron</h1>
    <p>Build cross-platform desktop apps with JavaScript, HTML, and CSS.</p>
  </body>
</html>
```



E Agora?

- Executar a aplicação novamente
 - `npm run electron`
- Viu a tela?
 - Lembra um Google Chrome?
- Navegue pelos menus disponibilizados



index.js

```
const electron = require('electron');
const { app, BrowserWindow } = electron;

app.on('ready', () => {
  console.log('Hello Electron!');
  const mainWindow = new BrowserWindow({});
  mainWindow.loadURL(
    `file://${__dirname}/index.html`);
});
```

Atividade

- Baixe o jogo da força disponível no Moodle
- Faça-o funcionar como uma app Electron
- Ao abrir, adapte as dimensões da janela para o jogador
- Testar desconectado de qualquer rede
 - A aplicação deve funcionar offline