DESENVOLVIMENTO WEB 2

Jair C Leite

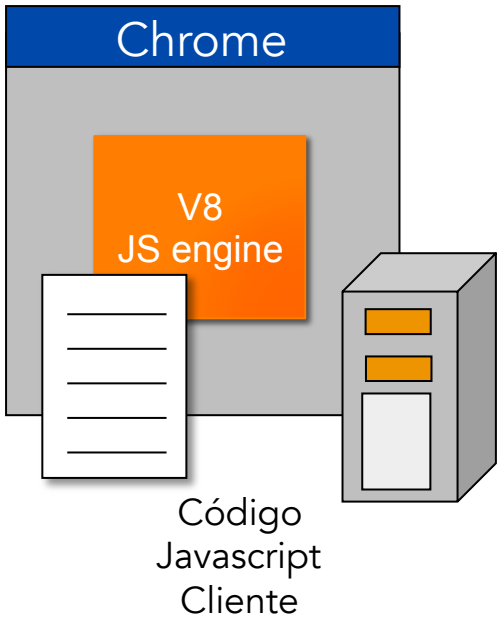
NODE JS

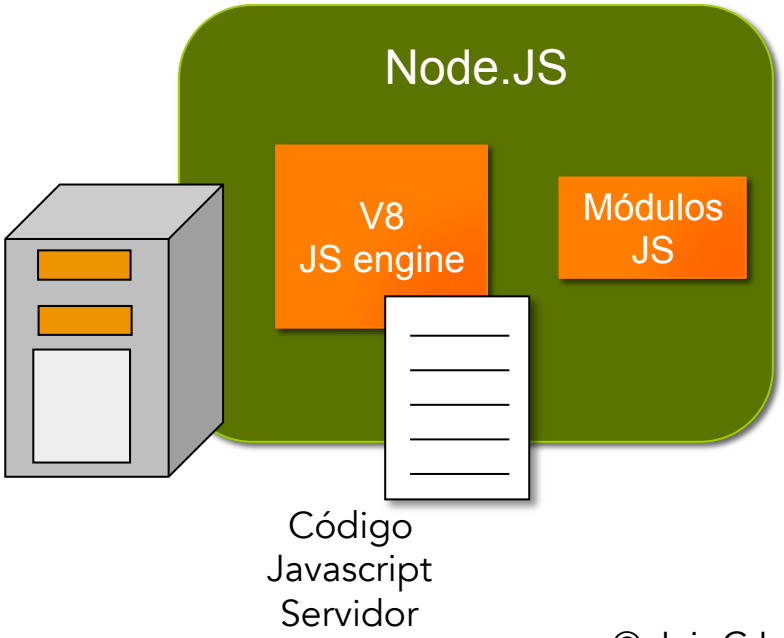
O Que é o Node.JS?

- Um ambiente de tempo de execução para JavaScript
- Usa o engine V8 do Chrome
- Executa na linha de comando ou a partir de requisições do browser

Node.JS

- Node.JS é um ambiente de execução de JS que utiliza o motor V8 do Chrome
- Pode ser usado como ambiente isolado ou na Web





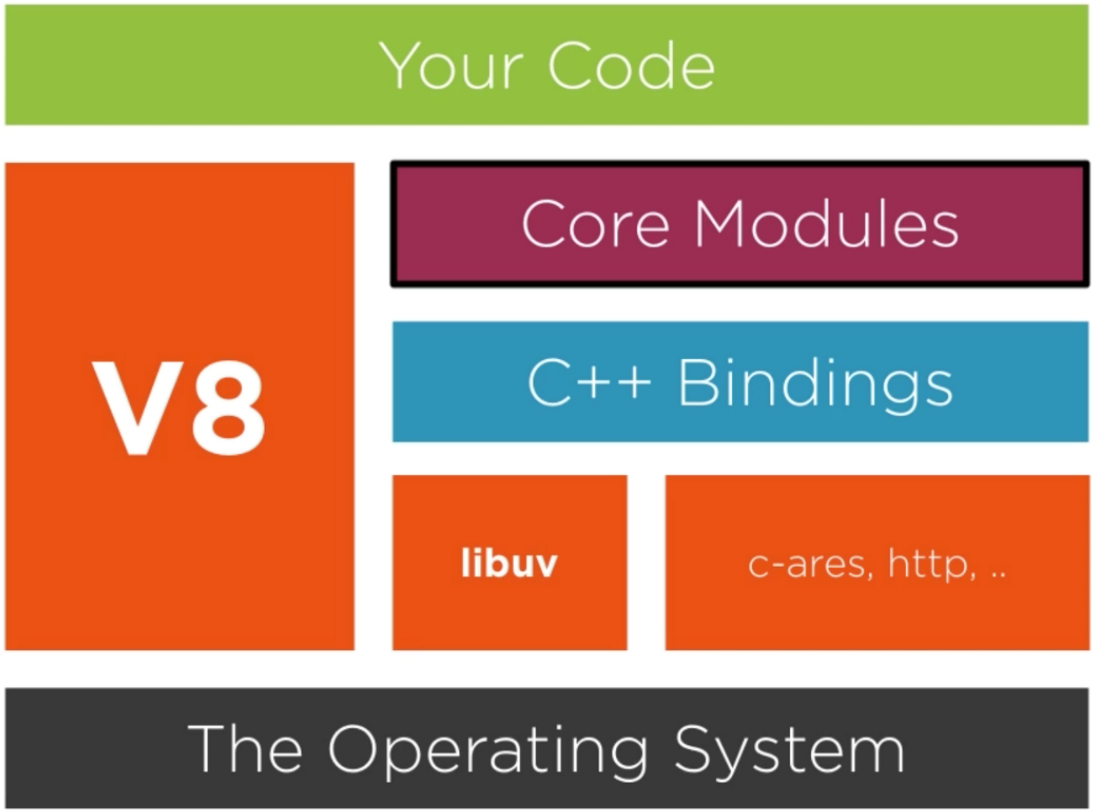
Instalando e rodando

- Baixe em nodejs.org
- Instale
- Verifique instalação
  - $ node –v
- Crie o programa nodeapp.js
- Rode no Node
  - $ node nodeapp.js

```
function ola(name) {
        console.log("Ola " + name);
}

ola("Pessoal");
```

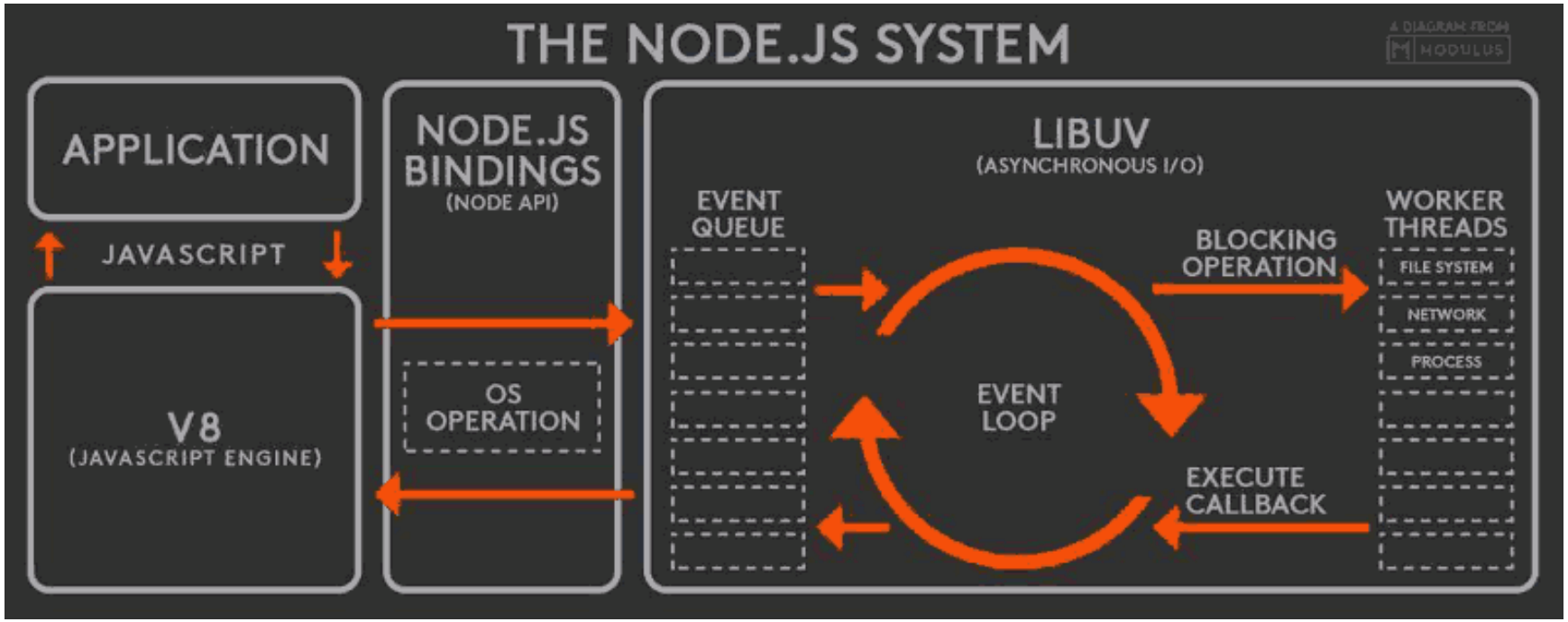
Arquitetura do Node.JS



Arquitetura dirigida por eventos

- Executa aplicações baseada em evento
  - Um loop fica esperando a ocorrência de eventos (rodar a aplicação)
  - Quando o evento ocorre ele delega para o sistema operacional ou outro recurso
  - O SO ou um recurso chamam de volta (callback) a aplicação

Arquitetura e o loop de eventos



Quando usar e não usar

- Use Node.JS para aplicações
  - com muitos acessos a E/S
  - com APIs JSON
  - com uso intensivo de dados em tempo-real
  - com stream de dados
  - arquiteturas página única (SPA)
- Não use em aplicações com uso intensivo de processamento:
  - processamento de imagens
  - deep learning

Módulos

- Cada arquivo é um modulo
- Pode conter várias funções
- Há modulos pre-definidos (built-in)
- Podem ser carregados com a função require()

```
var http = require('http');
```

- Podem ser criados e exportados com export

```
exports.myDateTime = function () {
    return Date();
};
```

Exemplo criando, exportando e utilizando

- Crie o programa e salve modulo1.js

```
exports.dataAtual = function ( ) {
    return Date();
}
```

- E crie outro programa

```
var modulo = require('./modulo1.js');
modulo.dataAtual();
```

Gerenciando módulos com npm

- npm – Node Packager Manager
  - Gerenciador de módulos (pacotes ou bibliotecas)
  - Instalado junto com Node
- Instalando pacotes
  - Na linha de comando

    $ npm install pacote-util
- Usando um modulo

    var modulo1 = require('pacote-util');

Gerenciando arquivos com Node.JS

- O módulo fs oferecer funções para gerenciar arquivor
  - Open
  - Read
  - Write
  - Append
  - Rename

Lendo um arquivo

- Usa o módulo fs

```
var fs = require('fs');

fs.readFile('../json/tarefas.json', 'utf8', function(err, dados)
  {
    console.log(dados);
  });

console.log('depois de ler...');
```

Adicionando dados em arquivos

- Usando o método appendFile( )

```
var fs = require('fs');

fs.appendFile('testenode.txt', 'nome1:valor1\n', function
  (err) {
  if (err) throw err;
  console.log('Arquivo salvo');
});
```

Salvando dados em arquivos

- Usando fs.writeFile( )

```
var fs = require('fs');

fs.writeFile('testenode.txt', 'nome1:valor1\n', function
  (err) {
  if (err) throw err;
  console.log('Arquivo substituido');
});
```

Exercícios

- Utilize as funções abaixo para:
- Apagar um arquivo

  fs.unlink('novoarquivo.txt', function (err) {...});
- Renomear um arquivo

  fs.rename('novonome.txt', 'myrenamedfile.txt', function (err) {...});
- Abrir um arquivo

  fs.open('meuarquivo.txt', 'w', function (err, file) {...});

SERVIDORES EM NODE.JS

Criando um servidor simples

- Crie um programa JS como abaixo:

```
var http = require('http');
http.createServer(function (req, res) {
    res.writeHead(200, {'Content-Type': 'text/html'});
    res.end('Esta página carrega a saída do programa executando NODEJS');
}).listen(8080);
```

- No browser chame o seu app com
  - localhost:8080

Recuperando a URL

- Usando a propriedade req.url

```
var http = require('http');
http.createServer(function (req, res) {
    res.writeHead(200, {'Content-Type': 'text/html'});
    res.write(req.url);
    res.end();
}).listen(8080);
```

- No browser coloque

http://localhost:8080/teste

Analisando a query string

- Use modulos http e url

```
var http = require('http');
var url = require('url');
http.createServer(function (req, res) {
  res.writeHead(200, {'Content-Type': 'text/html'});
  var q = url.parse(req.url, true).query;
  var txt = q.primnome + " " + q.sobrenome;
  res.end(txt);
}).listen(8080);
```

- No browser coloque

```
http://localhost:8080/?primnome=jair&sobrenome=leite
```

Analisando a URL completa

```
var url = require('url');
var adr = 'http://localhost:8080/teste?
  primnome=jair&sobrenome=leite';
var q = url.parse(adr, true);

console.log(q.host); // 'localhost:8080'
console.log(q.pathname); // 'teste'
console.log(q.search); // query string

var qdata = q.query; //json
console.log(qdata.sobrenome); //elemento da QS
```

Exercícios

- Crie um servidor que faça a autenticação de login de um usuários
- Crie um servidor que gerencie o arquivo de tarefas a fazer

Eventos

- Programas podem emitir e gerenciar eventos

```
// Importa modulos de eventos

var eventos = require('events');

// Cria um objeto emissor e um gestor de eventos
var emissorEventos = new eventos.EventEmitter(),
    gestorEventos = function conectado() {
            console.log('conexão estabelecida');
            //envia um evento
            emissorEventos.emit('recebido');
    };
```

Eventos (cont)

- Programas podem emitir e gerenciar eventos

```
// Associa o evnto conectado ao gestor de eventos
emissorEventos.on('conexao', gestorEventos);

// Associa o recebido com uma função
emissorEventos.on('recebido', function() {
    console.log('Dados recebidos com sucesso.');
});
// emite o evento
emissorEventos.emit('conexao');
console.log("Finalizado.");
```

Exercicio

- Adapte o código que manipula dois eventos para gerir outros eventos