CURSO DE

{ JAVASCRIPT FUNCIONAL }

GRATUITO

O Curso

- AO VIVO
- Gratuitamente gratuito
- Provavelmente aos Sábados a noite
- via Hangout ON AIR
- **http://webschool.io/jsfuncional/**
- **https://www.facebook.com/funcionaljavascript**
- **https://github.com/Webschool-io/workshop-js-funcional-free**

O Curso

Depois de alguns módulos o Curso só terá sequência caso ainda haja interesse pelos alunos.

O Professor

O Professor

Olá para quem não me conhece eu sou mais conhecido como Suissa e escrevo em um blog maladrinho, o nomadev.com.br. Ele tem esse nome por eu ser um nômade, sim aquele cara que não tem casa fixa, quase um hippie. 😋

O Professor

Faz dois anos que dou o único Workshop de MEAN presencial e ao vivo do Brasil. Basicamente é com isso que consigo sobreviver e fazer projetos gratuitos como esse, o JS4Girls e outros.

Doações

Como um pessoal já me perguntou como doar para esse projeto eu coloquei um botão no site mas quem preferir transferir ou depositar diretamente na Caixa ou Banco do Brasil só me avisar via Facebook ou pelo email webschool.cursos@gmail.com .

Doações - Acima de R$99

Quem doar acima de **R$99** ganhará uma camiseta do JS Funcional, porém precisa pagar o envio.

Doações - Acima de R$199

Quem doa acima de R$199 ganhará uma camiseta do JS Funcional, porém precisa pagar o envio.

E ainda poderá fazer ao vivo o Workshop de Geolocalização com MongoDB dia 29 de Julho as 19 horas!!!

O que é programação funcional?

Possuímos 2 grandes paradigmas de programação:

- Funcional
- Imperativo.

O que é programação funcional?

Programação funcional é um paradigma de programação que trata a computação como uma avaliação de funções matemáticas e evita estados ou dados mutáveis.

Utiliza a aplicação de funções, em contraste da programação imperativa, que enfatiza mudanças no estado do programa.

O que é programação funcional?

Assim como na orientação a objetos a menor parte de um sistema é um objeto, você pode atribuir objetos a variáveis, pode passá-los por parâmetro e ter métodos retornando objetos, na programação funcional, a menor parte do seu sistema é uma função.

O que é programação funcional?

Por exemplo, a função f(x) = x^2 + 5 é definida utilizando funções de exponenciação e adição. Do mesmo modo, a linguagem deve oferecer funções básicas que não requerem definições adicionais.

Por que usar PF?

Temos 4 grandes motivos para usar programação funcional, são eles:

Concorrência

Não temos deadlocks ou race conditions simplesmente porque não precisamos de locks - o dado é imutável.

Testes

Criar testes unitários sem se preocupar com o estado simplesmente porque não existe estado. Devemos preocupar apenas com os argumentos das funções que nós testamos.

Debugging

Rastrear algum valor no stack trace é bem simples.

Base teórica

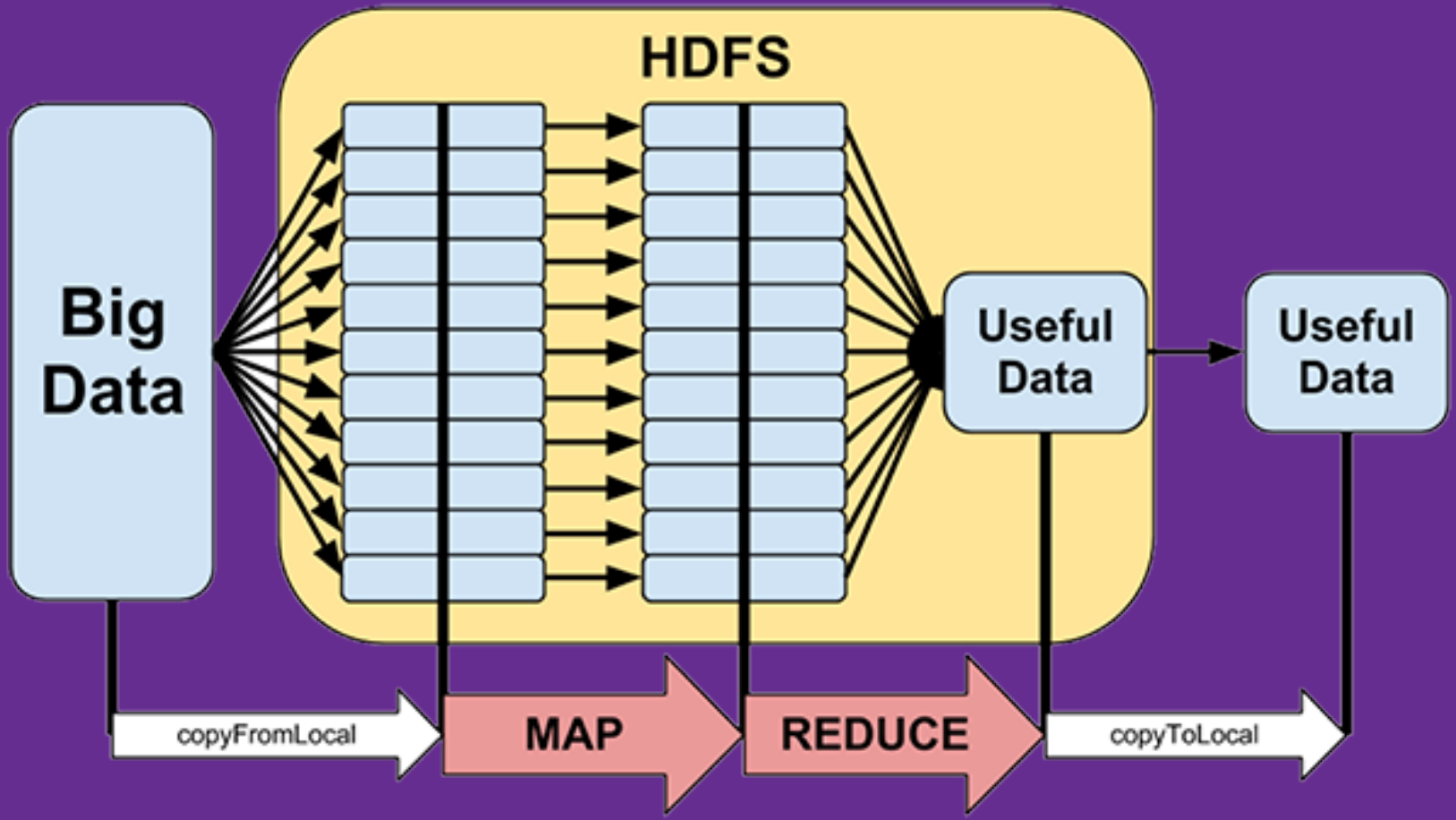
Linguagens funcionais são baseados no cálculo lambda, que é um sistema formal. Esta fundamentação teórica faz a prova para correção dos programas seja muito simples (por exemplo, usando indução).

Onde usar?

BI, Sistemas concorrentes.

Onde usar?

Por exemplo quase todo o Yahoo utiliza fortemente Hadoop que utiliza fortemente map/reduce.


HDFS
Big Data
Useful Data
Useful Data
copyFromLocal
MAP
REDUCE
copyToLocal

Quem está usando?

Spark, Netflix, Google, Facebook, [Amazon (Amazon Lambda), sistemas de avião como da família Airbus A340.

Linguagens funcionais

Linguagens mais conhecidas:

- Erlang;
- F#;
- Haskell;
- Lisp;
- OCaml;
- R;
- Scala;
- Scheme.

Linguagens funcionais

LISP introduziu a maioria das características hoje encontradas nas modernas linguagens de programação funcional.

Linguagens funcionais

Scheme foi uma tentativa posterior de simplificar e melhorar LISP.

Linguagens funcionais

Haskell foi lançada no fim dos anos 1980 numa tentativa de juntar muitas ideias na pesquisa de programação funcional.

Linguagens funcionais - Erlang

Além da Ericsson, é lógico, há algumas outras grandes empresas e projetos estão usando Erlang, como por exemplo:

- Facebook, no backend de seu sistema de chat, lidando com 100 milhões de usuários ativos;
- Delicious, que tem mais de 5 milhões de usuários e mais de 150 milhões de bookmarks;
- Amazon SimpleDB, o serviço de dados do poderoso Amazon EC2;
- GitHub, no seu sistema de backend, lidando com milhares de transações concorrentes;
- Motorola;
- CouchDB.

Linguagens funcionais - Elixir

Como a sintaxe de Erlang pode não ser convidativa para desenvolvedores "modernos", por isso José Valim desenvolveu o Elixir, linguagem com sintaxe moderna que roda dentro da madura VM do Erlang.

Por que JavaScript é funcional?

Vamos ver algumas features que fazem do JavaScript uma linguagem co conceitos funcionais.

Funções

No JavaScript uma função nada mais é que um objeto que possui atributos como:

- arguments
- name
- length

Funções

E funções importantes como:

- apply
- bind
- call

A diferença entre elas é que o apply aceita como parâmetro um array de argumentos, enquanto que, o call aceita uma lista, ou seja, passando um-a-um

Funções

Para criarmos uma função no JavaScript é muito simples, como já vimos anteriormente, precisamos apenas utilizar a palavra function.

Funções

```
function HelloMotherfucker(motherfuckersname) {
    alert('Oie: ' + motherfuckersname + ' como vai sua mãe?');
}
```

Funções

Percebi que o jeito mais fácil de entender programação funcional é algo que sempre falei e que sempre tentei seguir:

TODA FUNÇÃO PRECISA RETORNAR UM VALOR!

Funções

Sabendo dessa premissa como faríamos um simples atribuição de valor como:

var idade = 30;

Funções

Fácil!!!

```
function setIdade(idade) { return idade; }

var idade = setIdade(30);

function maioridade(idade) {
    return idade >= 18;
}
```

Funções

E chamamos ela da seguinte forma:

```
maioridade(setIdade(30));
```

Atomic Design Funcional

Dessa forma podemos pensar que ele se assemelha muito ao **Atomic Design** onde criamos pequenos átomos independentes, nesse caso as funções e com elas vamos compondo funções maiores, **exatamente** como visto aqui no artigo do Brad Frost.

Funções

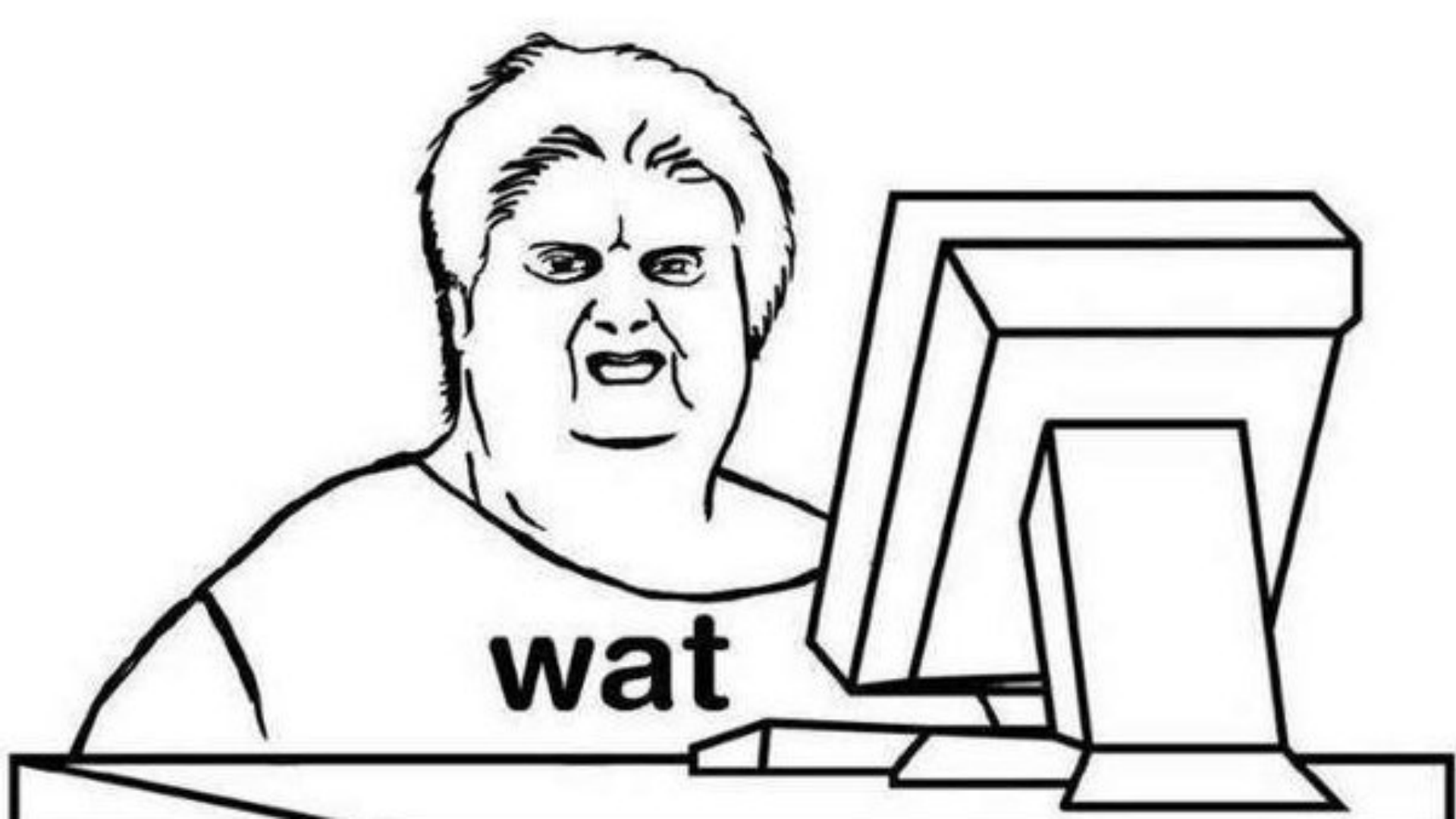
Agora vamos usar um exemplo mais simples ainda, uma função que duplica **Strings**:

```
var repeat = function(s) {
    return s + s;
};

repeat('Na');
// NaNa
```

Funções

Então se chamamos apenas a função repeat dessa forma, passando String então estará correta, porém se não fizermos isso teremos um resultado indesejado.


wat

Funções

Aí encontramos o problema!

Nesse caso ela não está mais repetindo a *String* como desejado inicialmente, agora ela está multiplicando o valor por 2 caso seja um Number. Isso porque não temos um contrato com uma função que retorne apenas Strings. Para resolver esse problema é fácil, criamos essa função abaixo:

Funções

```
var str = function(s) {
    if(typeof s !== 'string') {
        throw new TypeError('Expected a string');
    }
    else {
        return s;
    }
}
```

Funções

Agora você passa uma String para a função, como valor de entrada, e espera-se que seu retorno também seja uma String, como valor de saída.

Refatorando nossa função repeat:

Funções

```
var repeat = function(s) {
    var s = str(s)
    return s + s;
};

repeat('Na');
// NaNa
repeat(1)
// TypeError: Expected a string
```