
Facens

JavaScript não é Linguagem Java

Prof.º Abimael Oliveira

Facens

Conteúdo

- Breve contexto histórico | Java e JavaScript
- Notações de Tipagem e Sintaxe
- Diferenças nas linguagens

Facens

História do Java

Java foi criado em 1991 pela equipe de engenheiros da **Sun Microsystems**, liderada por **James Gosling**. Inicialmente chamado de "Oak," foi renomeado para Java em 1995. Java foi projetado para ser uma linguagem de programação orientada a objetos, com o objetivo de ser portátil, ou seja, "escreva uma vez, execute em qualquer lugar." A linguagem rapidamente ganhou popularidade por seu desempenho e segurança, e se tornou uma escolha popular para desenvolvimento de software corporativo, sistemas embarcados e aplicativos Android.

Facens

História do JavaScript

JavaScript foi desenvolvido por **Brendan Eich** na Netscape Communications em 1995, em apenas 10 dias. Inicialmente chamado de "Mocha," foi renomeado para "LiveScript" e depois para "JavaScript" em uma estratégia de marketing que buscava associá-lo à popularidade de Java, ainda que as linguagens sejam bem diferentes. A linguagem foi criada para tornar as páginas web interativas e, ao longo do tempo, evoluiu de um simples script de cliente para uma linguagem poderosa para desenvolvimento full-stack, com o advento do Node.js e diversas ferramentas de front-end.

Facens

Diferenças entre Java e JavaScript

1. **Tipagem e Sintaxe: Java** é uma linguagem compilada e fortemente tipada, enquanto **JavaScript** é uma linguagem interpretada e fracamente tipada.

2. **Execução: Java** é executado em uma Máquina Virtual Java (JVM) para garantir portabilidade, enquanto **JavaScript** originalmente só era executado no navegador e hoje também no servidor com Node.js.

Facens

Diferenças entre Java e JavaScript

3. **Paradigma de Programação:** Java é uma linguagem orientada a objetos pura, enquanto JavaScript suporta programação orientada a objetos, funcional e procedural.
4. **Uso:** Java é amplamente utilizado para desenvolvimento de aplicativos corporativos, dispositivos Android e sistemas embarcados. JavaScript é popular no desenvolvimento web e para criar interatividade em sites, além de ser usado no backend com Node.js.

Facens

1. Declaração de Variáveis e Tipagem

- **Java (tipagem forte)**
- Em Java, você precisa declarar o tipo da variável, e ela não pode mudar de tipo ao longo do programa.

java

```java
public class Main {
    public static void main(String[] args) {
        int numero = 10; //Declaração de uma variável inteira
        System.out.println(numero);
        // Não é possível fazer isso: número = "dez"; // Erro, pois "numero" é do tipo int
    }
}
```

Facens

1. Declaração de Variáveis e Tipagem

- **JavaScript (tipagem fraca)**
- Em JavaScript, as variáveis podem mudar de tipo dinamicamente.

javascript

```javascript
let numero = 10; // Variável inicializada como número
console.log(numero);
numero = "dez"; // Agora a variável armazena uma string
console.log(numero);
```

Facens

2. Sintaxe e Estruturas de Controle

- **Java (Sintaxe Orientada a Objetos)**
- Java requer a definição de classes e métodos, mesmo para executar um simples "Olá, Mundo!".

java

```java
public class Main {
public static void main(String[] args) {
 System.out.println("Olá, Mundo!"); // Saída no console
 }
}
```

Facens

2. Sintaxe e Estruturas de Controle

- **JavaScript (Sintaxe Mais Flexível)**
- JavaScript não exige a criação de classes para executar código simples.

javascript

```javascript
console.log("Olá, Mundo!"); // Saída no console
```

Facens

3. Funções e Métodos

- **Java (funções dentro de classes)**
- Em Java, todas as funções (métodos) devem estar dentro de uma classe.

java

```java
public class Main {
public static void saudacao() {
System.out.println("Olá de uma função em Java!");
}
public static void main(String[] args) {
 saudacao(); // Chamando o método
}
}
```

Facens

3. Funções e Métodos

- **JavaScript (Funções Independentes)**
- Em JavaScript, as funções podem existir de forma independente, fora de qualquer estrutura de classe.

javascript

```javascript
function saudacao() {
console.log("Olá de uma função em JavaScript!");
}
 saudacao(); //Chamando a função
```

Facens

4. Orientação a Objetos

- **Java (Classes e Objetos)**
- Java usa classes e objetos para orientação a objetos de forma mais rígida.

java

```java
public class Pessoa {
String nome; public Pessoa(String nome) {
 this.nome = nome;
}
public void saudacao() {
 System.out.println("Olá, meu nome é " + nome);
}
public static void main(String[] args) {
 Pessoa pessoa = new Pessoa("Carlos");
 pessoa.saudacao();
}
}
```

Facens

4. Orientação a Objetos

- **JavaScript (Prototipagem e Flexibilidade)**
- JavaScript permite criar objetos sem classes, usando protótipos ou a palavra-chave **`class`**.

javascript

```
class Pessoa { constructor(nome) {
 this.nome = nome; } saudacao() {
 console.log("Olá,meu nome é " + this.nome);
 }
}


const pessoa = new Pessoa("Carlos");
pessoa.saudacao();
```

Facens

5. Execução Assíncrona

- **Java (Threads)**
- Java lida com tarefas assíncronas principalmente através de threads.

java

```java
public class Main extends Thread {
 public void run() {
 System.out.println("Tarefa emsegundo plano em Java!");
}
 public static void main(String[] args) {
 Main tarefa = new Main();
 tarefa.start(); // Inicia a thread
  }
}
```

Facens

5. Execução Assíncrona

- **JavaScript (Promises e async/await)**
- JavaScript usa Promises e async/await para lidar com operações assíncronas.

javascript

```
function tarefa() {
    return new Promise(resolve => {
    setTimeout(() => {
        resolve("Tarefa em segundo plano em
    JavaScript!");
      }, 1000);
     });
    }
    async function executarTarefa() {
        const resultado = await tarefa();
            console.log(resultado);
     }
    executarTarefa();
```

Facens

4. Orientação a Objetos

- **Java (Classes e Objetos)**
- Java usa classes e objetos para orientação a objetos de forma mais rígida.

javascript

```javascript
function saudacao() {
console.log("Olá de uma função em JavaScript!");
}
 saudacao(); //Chamando a função
```

Facens

Concluindo...

Esses exemplos ilustram as diferenças de tipagem, estrutura de código, orientação a objetos e abordagem para tarefas assíncronas entre **Java** e **JavaScript**.

Facens

Muito agradecido!

Professor Abimael Oliveira

Facens