Hoisting no Javascript

Luan Alves
09/02/2022

COMPARTILHE

Confira neste artigo:

Autores(as): Luan Alves e Mônica Hillman

[Banner promocional da Alura, com um design futurista em tons de azul, apresentando o texto]

Quando uma equipe trabalha em algum projeto, muito provavelmente o código será lido e alterado por diversas pessoas, dessa maneira, é necessário entender diferentes formas possíveis de escrever o mesmo código, obtendo uma boa produtividade na sua performance de desenvolvimento.

Contudo, é importante para uma **pessoa programadora** entender o **que é Hoisting** e como se comporta em diferentes casos nesta linguagem.

Vamos analisar este **código de function declaration**, e o que retorna:

```
console.log(soma(2, 5))
function soma(a, b) {
return a + b
}
```

Este código retorna o valor: `7`

Repare que a função consegue ser chamada antes mesmo de ter sido declarada. **Hoisting** é o termo que explica essa situação, em português ele significa "içamento", ou "elevação" e foi citado pela primeira vez no **ECMAScript® 2015 Language Specification**. O Hoisting permite que você execute funções antes das suas declarações. Na prática, inicialmente as **declarações de funções** são colocadas na memória durante a fase de compilação e, mesmo assim, permanecem no mesmo lugar que estão digitadas.

A razão pela qual o código anterior funciona é que os mecanismos JavaScript movem a função soma para o início do escopo, como ilustra o código abaixo:

```
function soma(a,b){
    return a + b
}
console.log(soma(2,5))
```

Entretanto, será que Hoisting funciona em outros tipos de código? Vamos conferir!

```
console.log(alura)
var alura = 'cursos';
```

Este código retorna:

```
undefined
```

A utilização de **Hoisting em var não é indicada**, pois a variável criada é elevada para o escopo, mas sem o seu valor, e com isso, retorna valor `undefined`.

O comportamento é parecido utilizando também var function:

```
function testaHoisting() {
    console.log('testaHoisting', alura)
    var alura = 'cursos'
}
```

Este código retorna: `undefined`.

Hoisting também não é indicado utilizando let, pois acontece um outro tipo de **comportamento não desejado:**

```
console.log(alura)
let alura = 'cursos';
```

Este código retorna:

```
script.js:1 Uncaught ReferenceError: Cannot access 'alura' before initializati
```

Esta mensagem em português significa "Não é possível acessar 'alura' antes de sua inicialização", ou seja, a linguagem JavaScript reconhece que 'let = alura' existe, porém não consegue acessar sua declaração. Esse comportamento de Hoisting com `let` é o mesmo ao utilizar `const`.

Para saber mais

Após a leitura desse artigo você conheceu uma característica da **linguagem de programação Javascript.** Para aprofundar seus conhecimentos, **a Alura pode te ajudar nisso, aprenda mais com nossos cursos**:

- JavaScript: primeiros passos com a linguagem
- JavaScript: Programando na linguagem da web
- JavaScript para Web: Crie páginas dinâmicas
- JS na Web: Manipule o DOM com JavaScript

Luan Alves

Sou Luan Alves, estudante de Análise e Desenvolvimento de Sistemas, instrutor de Desenvolvimento Front-End no Grupo Alura. Estou aqui para ajudar a tirar suas dúvidas, aprender e compartilhar conhecimento. :)

← Artigo Anterior
Navegação com Next.Js utilizando rotas dinâmicas

Próximo Artigo →
Position CSS: entenda essa propriedade

Veja outros artigos sobre Front-end

Quer mergulhar em tecnologia e aprendizagem?

Receba conteúdos, dicas, notícias, inovações e tendências sobre o mercado tech diretamente na sua caixa de entrada.

Email* ENVIAR