

**Caio**

Posted on Jul 24, 2023



3



1



1



1



1

Porque estão evitando de usar o require e usando o import no JavaScript

Diferenças

Uma das diferenças entre require e import é que require é usado para carregar módulos no Node.js, enquanto import é usado para importar módulos no JavaScript.

Outra diferença importante é que o require retorna um objeto, enquanto o import retorna uma referência para o módulo.

Isso significa que, quando você usa require, você pode atribuir o retorno à uma variável e usar essa variável para acessar as propriedades e métodos do módulo.

Já com o import você precisa acessar diretamente as propriedades e métodos do módulo importado, resumindo, o require é usado para carregar módulos no Node.js e é uma função built-in, enquanto o import é usado para importar módulos no JavaScript e é uma palavra-chave do ECMAScript 6, e não é suportado nativamente pelo Node.js.

Require

O require é uma função built-in do Node.js e é usado para carregar módulos de arquivos externos e pacotes instalados globalmente. Ele também pode ser usado para carregar módulos internos do Node.js, como http e fs.

Exemplo de importação com require:

```
// módulo "myModule.js"
const myVariable = 'Hello World';
function myFunction() {
  console.log('This is my function');
}
module.exports = { myVariable, myFunction }

// arquivo "main.js"
const myModule = require('./myModule');
console.log(myModule.myVariable); // imprime "Hello World"
myModule.myFunction(); // imprime "This is my function"
```

Import

Já o import é uma palavra-chave do JavaScript, ela foi introduzida a partir da versão ECMAScript 6 (ES6) e não é suportada pelo Node.js, para usar essa funcionalidade é necessário usar algum transpiler que possa transpilar o código para uma versão que o Node.js entenda.

Exemplo de importação com import :

```
// módulo "myModule.js"
export const myVariable = 'Hello World';
export function myFunction() {
  console.log('This is my function');
}

// arquivo "main.js"
import { myVariable, myFunction } from './myModule';
console.log(myVariable); // imprime "Hello World"
myFunction(); // imprime "This is my function"
```

No entanto, é importante observar que o JavaScript evoluiu bastante nos últimos anos com a introdução do ECMAScript 6 (ES6) e versões posteriores. Uma das

principais adições foi a introdução dos módulos do ES6, que fornecem uma sintaxe mais clara e poderosa para importar e exportar módulos.

Com a introdução dos módulos do ES6, muitos desenvolvedores têm adotado essa nova sintaxe como uma alternativa ao `require`, especialmente em projetos modernos. Os módulos do ES6 oferecem recursos avançados, como importações nomeadas, importações assíncronas e importações dinâmicas, que podem trazer benefícios em termos de legibilidade, manutenção e desempenho do código.

Portanto, em vez de evitar o `require`, é mais preciso dizer que os desenvolvedores estão preferindo usar os módulos do ES6 sempre que possível, especialmente em projetos JavaScript modernos e em ambientes que suportam essa sintaxe. No entanto, em ambientes mais antigos ou em casos específicos, o `require` ainda é amplamente utilizado e continua sendo uma opção válida para inclusão de módulos em JavaScript.

Usar o `import` em vez do `require` no JavaScript traz algumas vantagens, especialmente quando se trata de projetos modernos que fazem uso dos recursos do ECMAScript 6 (ES6) e ambientes de desenvolvimento que suportam módulos. Algumas razões pelas quais o `import` é preferido em muitos casos:

- Sintaxe mais clara e concisa.
- Escopo de importação controlado.
- Suporte nativo para módulos.
- Importações assíncronas e dinâmicas.
- Ferramentas de construção e bundling.

No entanto, é importante mencionar que, em certos contextos, como em projetos mais antigos ou ambientes que não suportam nativamente os módulos ES6, ainda pode ser necessário usar o `require`. O `require` é uma função comumente usada em ambientes como o Node.js e é suportada por sistemas de gerenciamento de pacotes, como o npm. Portanto, a escolha entre `import` e `require` depende do contexto do projeto, das versões do ECMAScript suportadas e dos requisitos específicos do ambiente de desenvolvimento.

Fonte: <https://horadecodar.com.br/qual-a-diferenca-entre-require-e-import-no-node-js/>

Top comments (1)

Caio • Jul 26 '23



Usamos isso diariamente .

[Code of Conduct](#) • [Report abuse](#)[arm](#) Arm **PROMOTED**

The advertisement features a woman with long brown hair, wearing a dark blue zip-up jacket and a gold headband, standing with her hands on her hips. To her left is a white robot character with a circular body, a single eye, and a small antenna. The background is a green and blue grid pattern. The text "arm" is written in a large, bold, black font. Below it, the text "Bring Your Boldest Ideas to Life with Help from Ruth Amos." is written in a bold, black font. At the bottom, there is a white robot character with a circular body, a single eye, and a small antenna, standing on two wheels.

[Learn How Ruth Amos Supercharges Developer Creativity.](#)

- Tap into your child-inspired creativity.
- Embrace fearless problem-solving.
- Create ground-breaking solutions.



Level up your design and development skills.



Learn from an award-winning engineer and innovator.

[Register Now](#)



Caio

Desenvolvedor / Programador

EDUCATION

Ti

WORK

Autonomo

JOINED

Apr 4, 2023

More from **Caio**

Use @use e não o @import no SASS

Por que usar ponto e vírgula no Javascript?

Como escrever um README profissional no seu Github



DEV Sponsors

PROMOTED





Monetize your audience: Fund an OSS project or website with EthicalAds, **a privacy-first ad network**

Ads by EthicalAds