# O V8 JavaScript Engine

V8 é o nome da engine JavaScript que roda no Google Chrome. É a coisa que pega nosso JavaScript e o executa enquanto navegamos com o Chrome.

V8 provê o ambiente de execução em que o JavaScript executa. O DOM, e outras APIs web são fornecidas pelo browser.

O legal é que a engine JavaScript é independente do browser que ela está hospedada. Essa funcionalidade chave possibilitou a ascensão do Node.js. A V8 foi escolhida como engine por trás do Node.js em 2009, e com a explosão de popularidade do Node.js, a V8 se tornou a engine que agora possibilita uma quantidade incrível de código sever-side escrito em JavaScript.

O ecossistema Node.js é enorme e isso graças à V8 que também possibilitou aplicações desktop, com projetos como o Electron.

## Outras engines JS

Outros browsers têm suas próprias engines:

- Firefox tem a **SpiderMonkey**
- Safari tem a **JavaScriptCore** (também chamada de Nitro)
- Edge foi originalmente basedo na **Chakra** mas recentemente foi refeito utilizando Chromium e a V8.

e existem muitas outras também.

Todas essas engines implementam o padrão ECMA ES-262, também chamado de ECMAScript, o padrão utilizado pelo JavaScript

chamado de ECMAScript, o padrão utilizado pelo JavaScript.

## A busca por performance

V8 foi escrita em C++, e é continuamente melhorada. É portável e roda no Mac, Windows, Linux e diversos outros sistemas.

Nessa introdução à V8, vamos ignorar os detalhes de implementação: eles podem ser encontrados em sites mais apropriados (por exemplo, o site oficial da V8), e eles mudam com o passar do tempo, frequentemente.

V8 está sempre evoluindo, assim como as outras engines JavaScript ao seu redor, para agilizar a Web e o ecossistema Node.js.

Na web, há uma corrida por performance que vem sendo travada por anos, e nós (como usuários e desenvolvedores) nos beneficiamos muito por essa competição, porque nós possuímos máquinas mais rápidas e otimizadas ano a ano.

## Compilação

JavaScript é geralmente considerado como uma linguagem interpretada, mas engines modernas de JavaScript não o interpretam apenas, elas o compilam.

Isso vem acontecendo desde 2009, quando o compilador JavaScript SpiderMonkey foi adicionado no Firefox 3.5, e todo mundo seguiu essa ideia.

JavaScript é internamente compilado pela v8 com **compilação just-in-time** (JIT) para acelerar a execução.

Isso pode parecer contra-intuitivo, mas desde a introdução do Google Maps em 2004, o JavaScript evoluiu de uma linguagem que geralmente executava poucas dúzias de linhas de código, para aplicações completas com centenas de milhares de linhas de código executando no browser.

Nossas aplicações agora podem rodar por horas dentro do browser, em vez de uma simples validação de regras de formulários ou scripts banais.

Nesse *novo mundo*, compilar JavaScript faz total sentido porque, embora possa demorar um pouco mais para termos o código JavaScript *pronto*, uma vez concluída a compilação temos muito mais desempenho do que código puramente interpretado.


Previous
**Diferenças entre Node.js e o navegador**

Next
**Execute scripts Node.js na linha de comando**