# Desempenho de JavaScript

[❮ Anterior](https://translate.googleusercontent.com/translate_c?depth=1&hl=pt-BR&ie=UTF8&prev=_t&rurl=translate.google.com.br&sl=en&sp=nmt4&tl=pt-BR&u=https://www.w3schools.com/js/js_mistakes.asp&usg=ALkJrhhOZIxtbxVKU10X5aVbrzmfsmvPiw)[Próximo ❯](https://translate.googleusercontent.com/translate_c?depth=1&hl=pt-BR&ie=UTF8&prev=_t&rurl=translate.google.com.br&sl=en&sp=nmt4&tl=pt-BR&u=https://www.w3schools.com/js/js_reserved.asp&usg=ALkJrhiLjQCSkO0tfiN_M6XDm3CJ4KVS1Q)

Como acelerar seu código JavaScript.

## Reduza a atividade em Loops

Loops são freqüentemente usados ​​na programação.

Cada declaração em um loop, incluindo a instrução for, é executada para cada iteração do loop.

Declarações ou atribuições que podem ser colocadas fora do loop farão o loop executar mais rápido.

### Mau:

var i;  
for (i = 0; i < arr.length; i++) {

### Código melhor:

var i;  
var l = arr.length;  
for (i = 0; i < l; i++) {

O código incorreto acessa a propriedade de comprimento de uma matriz cada vez que o ciclo é iterado.

O código melhor acessa a propriedade de comprimento fora do loop e faz o loop executar mais rápido.

## Reduzir o acesso DOM

Acessar o HTML DOM é muito lento, em comparação com outras instruções do JavaScript.

Se você espera acessar um elemento DOM várias vezes, acesse-o uma vez e use-o como uma variável local:

### Exemplo

var obj;  
obj = document.getElementById("demo");  
obj.innerHTML = "Hello";

[Tente você mesmo "](https://translate.googleusercontent.com/translate_c?depth=1&hl=pt-BR&ie=UTF8&prev=_t&rurl=translate.google.com.br&sl=en&sp=nmt4&tl=pt-BR&u=https://www.w3schools.com/js/tryit.asp%3Ffilename%3Dtryjs_element_reference&usg=ALkJrhh82M2wYvgGuXqQE6cBodCQsmt9kw)

## Reduzir o tamanho do DOM

Mantenha o número de elementos no HTML DOM pequeno.

Isso sempre melhorará o carregamento da página e acelerará a renderização (exibição da página), especialmente em dispositivos menores.

Toda tentativa de pesquisar o DOM (como getElementsByTagName) se beneficiará de um DOM menor.

## Evite Variáveis ​​Desnecessárias

Não crie novas variáveis ​​se você não planeja salvar valores.

Muitas vezes você pode substituir um código como este:

var fullName = firstName + " " + lastName;  
document.getElementById("demo").innerHTML = fullName;

Com isso:

document.getElementById("demo").innerHTML = firstName + " " + lastName

## Atrasar JavaScript Carregando

Colocar seus scripts na parte inferior do corpo da página permite que o navegador carregue a página primeiro.

Enquanto um script está sendo baixado, o navegador não iniciará nenhum outro download. Além disso, todas as atividades de análise e renderização podem ser bloqueadas.

A especificação HTTP define que os navegadores não devem baixar mais de dois componentes em paralelo.

Uma alternativa é usar **defer = "true"** na etiqueta do script. O atributo de diferimento especifica que o script deve ser executado depois que a página terminar de analisar, mas funciona apenas para scripts externos.

Se possível, você pode adicionar seu script à página por código, após a página ter carregado:

### Exemplo

<script>  
window.onload = function() {  
    var element = document.createElement("script");  
    element.src = "myScript.js";  
    document.body.appendChild(element);  
};  
</script>

## Evite usar com

Evite usar o **com a palavra-chave** . Tem um efeito negativo sobre a velocidade. Também escava os escopos JavaScript.

A palavra-chave **não** é **permitida** no modo estrito.