Modo Estrito ('use strict')

1. INTRODUÇÃO AO MODO ESTRITO

O Modo Estrito (`Strict Mode`) é uma ferramenta estratégica e fundamental no desenvolvimento JavaScript moderno. Ele não introduz novas funcionalidades à linguagem, mas atua como um mecanismo de segurança que ajuda a escrever um código mais limpo, robusto e menos propenso a erros. Adotar essa prática desde o início é um passo essencial para qualquer desenvolvedor que busca criar aplicações elegantes e funcionais.

O `'use strict'` é um recurso opcional do JavaScript que, quando ativado, introduz uma série de restrições ao código. Sua função é transformar erros "silenciosos" — que normalmente passariam despercebidos pelo interpretador da linguagem — em erros explícitos, forçando uma correção imediata e evitando comportamentos inesperados na aplicação.

O objetivo principal do Modo Estrito é compelir os programadores a serem mais cuidadosos, especialmente com a declaração e o uso de variáveis. Ao impor regras mais rígidas, ele garante que os desenvolvedores sejam "obrigados a criar um código mais limpo". O resultado é um "código Limpo, um código elegante, um código funcional", alinhado com as melhores práticas do desenvolvimento contemporâneo. Para entender seu valor, vamos primeiro analisar o comportamento padrão do JavaScript quando o Modo Estrito não está ativo.

1.1. O Comportamento Padrão: Código Sem Modo Estrito

Por padrão, o JavaScript opera de maneira permissiva. Uma de suas flexibilidades mais notáveis é a capacidade de atribuir um valor a uma variável que não foi formalmente declarada. Embora isso possa parecer conveniente, essa leniência pode esconder bugs, levar à criação de variáveis globais acidentais e tornar o código imprevisível. Considere o seguinte trecho de código, que demonstra essa característica e diferentes usos do `console.log`:

```
// Atribuição de valor a uma variável não declarada
nome = 'Bruno';

// Demonstração do console.log
console.log('cfb cursos'); // Imprimindo uma string
console.log(nome); // Imprimindo o valor da variável
console.log('Canal: ' + nome); // Concatenando string com variável
```

- **`nome = 'Bruno';`**: Um valor é atribuído a `nome` sem que a variável tenha sido declarada previamente com `let`, `const` ou `var`.
- **`console.log('cfb cursos');`**: Exibe uma string de texto literal no console.
- **`console.log(nome);`**: Exibe o valor contido na variável `nome`.
- **`console.log('Canal: ' + nome);`**: Concatena uma string com o valor da variável e exibe o resultado.

Ao executar este script em um ambiente JavaScript padrão, o resultado é que "não deu erro nenhum o código rodou normalmente". O interpretador cria uma variável global `nome` implicitamente, uma prática que pode causar conflitos e dificultar a manutenção do código em projetos maiores. Para evitar esse tipo de problema silencioso, a solução é ativar o Modo Estrito e forçar a detecção de tais erros.

1.2. Ativando e Aplicando o Modo Estrito

O JavaScript oferece um mecanismo simples e integrado para impor uma análise mais rigorosa e um tratamento de erros mais estrito. A ativação desse modo — também conhecido como Modo Restrito — é uma prática comum e recomendada em ecossistemas modernos como Node.js e React, onde a qualidade e a previsibilidade do código são essenciais. Para habilitar o Modo Estrito, basta adicionar a seguinte diretiva como a primeira linha do seu arquivo de script:

```
'use strict';
```

Ao colocar esta única linha no topo do arquivo, você instrui o motor JavaScript a interpretar todo o código subsequente sob um conjunto de regras mais rígido. Qualquer violação dessas regras, que antes seria ignorada, agora gerará um erro. Vamos aplicar essa diretiva ao nosso exemplo anterior para observar a mudança de comportamento:

```
'use strict'; // Modo Estrito ativado

// Tentativa de atribuição a uma variável não declarada
nome = 'Bruno';

console.log(nome);
```

Agora, vamos analisar o impacto direto e imediato que essa única linha de código tem na execução do programa.

1.3. O Impacto na Prática: Identificação e Correção de Erros

É neste ponto que o valor prático do Modo Estrito se torna evidente. Ele transforma um problema silencioso e potencialmente perigoso em um erro claro e acionável, promovendo uma disciplina de codificação muito mais rigorosa. Ao executar o código da seção 3.4, o programa não roda mais silenciosamente. Em vez disso, ele para e exibe um erro crítico no console. Os detalhes desse erro são:

- **Tipo de Erro:** `ReferenceError`
- **Mensagem de Erro:** `nome is not defined`
- **Causa:** O erro ocorre porque "eu tô tentando atribuir alguma coisa a uma variável que não foi declarada".

É fundamental notar que "isso só é esse erro só é alertado agora porque eu indiquei aqui o modo estrito". Sem essa diretiva, o problema passaria completamente despercebido. Esse `ReferenceError` não é apenas uma mensagem; é o mecanismo que impede ativamente o motor JavaScript de criar uma variável global acidental `nome`, evitando assim possíveis bugs e conflitos de namespace.

A solução para o erro é direta e alinhada com as boas práticas de programação: declarar formalmente a variável usando `let` antes de atribuir um valor a ela. O código corrigido e totalmente funcional fica assim:

```
'use strict';

// A variável agora é devidamente declarada com 'let'
let nome = 'Bruno';

console.log('Canal: ' + nome);
```

Com a declaração explícita, o `ReferenceError` desaparece e o programa executa conforme o esperado, imprimindo o resultado no console. Esse comportamento de verificação de erros é consistente, ocorrendo tanto em ambientes de servidor (Node.js) quanto no navegador (Browser).

2. POR QUE ADOTAR O MODO ESTRITO?

A lição principal é clara: o Modo Estrito atua como uma rede de segurança indispensável para o desenvolvedor JavaScript. Ele altera o comportamento padrão da linguagem para converter certos erros silenciosos em exceções explícitas. Essa mudança, por si só, representa uma vantagem significativa na construção de software confiável. Vale ressaltar que, enquanto em JavaScript é uma opção, "algumas linguagens vão obrigar inclusive o uso de strict mode", o que reforça sua importância como uma prática moderna. Os principais benefícios de usar `'use strict'` podem ser resumidos nos seguintes pontos:

- **Prevenção de Erros:** Converte erros silenciosos, como a atribuição a variáveis não declaradas, em erros explícitos do tipo `ReferenceError`. Isso **força a declaração correta de variáveis** e previne a criação de variáveis globais acidentais.
- **Código Mais Limpo:** Ajuda a evitar "sujeira" e a criar um **código mais elegante e funcional**, eliminando práticas ambíguas e propensas a falhas.
- **Melhores Hábitos de Programação: Obriga os programadores** a terem mais cuidado, incentivando a disciplina e a adesão às melhores práticas desde o início.
- **Consistência:** Garante que o código se comporte de forma mais previsível e consistente em diferentes ambientes de execução, como **Node.js e o Navegador (Browser)**.

Adotar o Modo Estrito é um passo simples, mas de alto impacto, na jornada para escrever um código JavaScript profissional, moderno e de fácil manutenção.