use strict

to apply this mode, write in double quotes to top of the JavaScript file:

"use strict";

JavaScript's strict mode is a way to opt in to a restricted variant of JavaScript, thereby implicitly opting-out of "sloppy mode". Strict mode isn't just a subset: it intentionally has different semantics from normal code. Browsers not supporting strict mode will run strict mode code with different behavior from browsers that do, so don't rely on strict mode without feature-testing for support for the relevant aspects of strict mode. Strict mode code and non-strict mode code can coexist, so scripts can opt into strict mode incrementally.

1. Strict mode makes several changes to normal JavaScript semantics:
2. Eliminates some JavaScript silent errors by changing them to throw errors.
3. Fixes mistakes that make it difficult for JavaScript engines to perform optimizations: strict mode code can sometimes be made to run faster than identical code that's not strict mode.
4. Prohibits some syntax likely to be defined in future versions of ECMAScript.

**Invoking strict mode**

Strict mode applies to entire scripts or to individual functions. It doesn't apply to block statements enclosed in {} braces; attempting to apply it to such contexts does nothing. eval code, Function code, event handler attributes, strings passed to setTimeout(), and related functions are either function bodies or entire scripts, and invoking strict mode in them works as expected.

Strict mode for scripts

To invoke strict mode for an entire script, put the exact statement "use strict"; (or 'use strict';) before any other statements.

JS

// Whole-script strict mode syntax

"use strict";

const v = "Hi! I'm a strict mode script!";

**Usage**

You can enable strict mode in two different ways, globally and locally.

Enable strict mode globally by adding the string "use strict" as the first statement in your file. All subsequent code in the script will execute in strict mode.

// First line inside a .js file

'use strict';

// rest of the script

Enable strict mode locally by adding the string "use strict" as the first statement inside a function. Using strict mode locally inside the function will enforce stricter parsing only within the context of that function.

// Inside a function

function strict\_function() {

'use strict'; // rest of the function

}

**It is important to note that JavaScript modules are in strict mode by default.**

**Benefits of Using Strict Mode**

Enabling strict mode in your code has [many benefits](https://johnresig.com/blog/ecmascript-5-strict-mode-json-and-more/).

Here are some examples:

**Prevent accidental creation of global variables**

Strict mode is useful in situations where you can accidentally create a global binding. Accidental global variables can create bugs in the code. For example:

'use strict';

x = 9; *// ReferenceError: x is not defined*

function strict\_function() {

'use strict';

x = 'I am a strict function';

console.log(x);

}

strict\_function(); *// ReferenceError: x is not defined*

A common scenario in which you can accidentally create global variables is within for loops. With strict mode enabled, JavaScript will throw an error if you forget to add the let keyword before the loop counter variable (counter in the following example):

function catchTheProblem() {

'use strict';

for (counter = 0; counter < 10; counter++) {

console.log('Catch me if you can!');

}

}

catchTheProblem(); *// ReferenceError: counter is not defined*

If you don’t use strict mode here, JavaScript will create a global variable with the name counter.

**Catch typing mistakes in variable names**

If you don’t use strict mode, a typing error can create a new variable, which can cause bugs down the line. For example:

'use strict';

let misspelledVariable;

misspelleVariable = 9; *// This line will throw en error due to // misspelling of "misspelledVariable"*

**Prevent accidental deleting**

Deleting a variable, a function, or an argument will result in an error:

'use strict';

let aVariable = 9;

delete aVariable; *//This will cause an error*

function testFunction() {}

delete testFunction; *// This will cause an error*

function testWithArgs(arg) {

delete arg; *// This will cause an error*

}

**Prevent duplicating parameter names in a function**

Duplicating the parameter’s names will result in an error:

function test(arg1, arg1) {} *// This will cause an error*

**Prevent writing to read-only properties**

If you set an object’s writable value to false, and then try to assign a new value to the object, strict mode will throw an error:

'use strict';

const obj = {};

Object.defineProperty(obj, 'x', { value: 0, writable: false });

obj.x = 9; *// This will cause an error*

You can see more examples and benefits of using strict mode on [w3schools](https://www.w3schools.com/js/js_strict.asp) and [MDN documentation](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Strict_mode).