JavaScript Use Strict

❮ Previous Next ❯

`"use strict";` Defines that JavaScript code should be executed in "strict mode".

The "use strict" Directive

The "use strict" directive is new in JavaScript 1.8.5 (ECMAScript version 5).

It is not a statement, but a literal expression, ignored by earlier versions of JavaScript.

The purpose of "use strict" is to indicate that the code should be executed in "strict mode".

With strict mode, you can not, for example, use undeclared variables.

> Strict mode is supported in:
> IE from version 10. Firefox from version 4.
> Chrome from version 13. Safari from version 5.1.
> Opera from version 12.

Declaring Strict Mode

Strict mode is declared by adding `"use strict";` to the beginning of a script or a function.

Declared at the beginning of a script, it has global scope (all code in the script will execute in strict mode):

Example

```
"use strict";
x = 3.14;       // This will cause an error because x is not declared
```

Try it Yourself »

Example

```
"use strict";
myFunction();

function myFunction() {
    y = 3.14;   // This will also cause an error because y is not declared
}
```

Try it Yourself »

Declared inside a function, it has local scope (only the code inside the function is in strict mode):

```
x = 3.14;       // This will not cause an error.
myFunction();

function myFunction() {
   "use strict";
    y = 3.14;   // This will cause an error
}
```

Try it Yourself »

The "use strict"; Syntax

The syntax, for declaring strict mode, was designed to be compatible with older versions of JavaScript.

Compiling a numeric literal (4 + 5;) or a string literal ("John Doe";) in a JavaScript program has no side effects. It simply compiles to a non existing variable and dies.

So "use strict"; only matters to new compilers that "understand" the meaning of it.

Why Strict Mode?

Strict mode makes it easier to write "secure" JavaScript.

Strict mode changes previously accepted "bad syntax" into real errors.

As an example, in normal JavaScript, mistyping a variable name creates a new global variable. In strict mode, this will throw an error, making it impossible to accidentally create a global variable.

In normal JavaScript, a developer will not receive any error feedback assigning values to non-writable properties.

In strict mode, any assignment to a non-writable property, a getter-only property, a non-existing property, a non-existing variable, or a non-existing object, will throw an error.

Not Allowed in Strict Mode

Using a variable, without declaring it, is not allowed:

```
"use strict";
x = 3.14;                // This will cause an error
```

Try it Yourself »

Objects are variables too.

Using an object, without declaring it, is not allowed:

```
"use strict";
x = {p1:10, p2:20};      // This will cause an error
```

Try it Yourself »

Deleting a variable (or object) is not allowed.

```
"use strict";
var x = 3.14;
delete x;                // This will cause an error
```

Try it Yourself »

Deleting a function is not allowed.

```
"use strict";
function x(p1, p2) {};
delete x;                // This will cause an error
```

Try it Yourself »

Duplicating a parameter name is not allowed:

```
"use strict";
function x(p1, p1) {};   // This will cause an error
```

Try it Yourself »

Octal numeric literals are not allowed:

```
"use strict";
var x = 010;              // This will cause an error
```

Try it Yourself »

Octal escape characters are not allowed:

```
"use strict";
var x = "\010";            // This will cause an error
```

Try it Yourself »

Writing to a read-only property is not allowed:

```
"use strict";
var obj = {};
Object.defineProperty(obj, "x", {value:0, writable:false});

obj.x = 3.14;             // This will cause an error
```

Try it Yourself »

Writing to a get-only property is not allowed:

```
"use strict";
var obj = {get x() {return 0} };

obj.x = 3.14;             // This will cause an error
```

Try it Yourself »

Deleting an undeletable property is not allowed:

```
"use strict";
delete Object.prototype; // This will cause an error
```

Try it Yourself »

The string "eval" cannot be used as a variable:

```
"use strict";
var eval = 3.14;          // This will cause an error
```

Try it Yourself »

The string "arguments" cannot be used as a variable:

```
"use strict";
var arguments = 3.14;     // This will cause an error
```

Try it Yourself »

The with statement is not allowed:

```
"use strict";
with (Math){x = cos(2)}; // This will cause an error
```

Try it Yourself »

For security reasons, eval() is not allowed to create variables in the scope from which it was called:

```
"use strict";
eval ("var x = 2");
alert (x);                // This will cause an error
```

Try it Yourself »

In function calls like f(), the this value was the global object. In strict mode, it is now undefined.

Future Proof!

Future reserved keywords are not allowed in strict mode. These are:

- implements
- interface
- let
- package
- private
- protected
- public
- static
- yield

```
"use strict";
var public = 1500;      // This will cause an error
```

Try it Yourself »

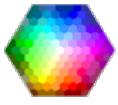
Watch Out!

The "use strict" directive is only recognized at the **beginning** of a script or a function.

❮ Previous

Next ❯

COLOR PICKER



HOW TO

Tabs
Dropdowns