Strict mode

Contents

Overview

Strict mode is a literal expression introduced in ECMAScript 5 which is used to indicate that the code should be executed in “strict mode”.

Tutorial

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

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

How to declare strict

The way to declare a JavaScript document to use strict is by declaring `use strict` as the start of the document.

```
'use strict';
```

When using strict mode there are certain things that we are **not allowed to do**:

- Using a variable without declaring it. E.g. x=10;
- Using an object with out declaring it. E.g. x={a:10,b:20};
- Deleting a variable or object.
- Deleting a function.
- Duplicating a parameter name.
- Octal numeric literal
- Octal escape characters
- Writing to a read-only properly.
- Writing to a get-only property.

There are also certain words that we cannot use for naming variables:

- eval
- Arguments
- With

For security reasons `eval()` is not allowed to create a variables in the scope from which it is called.

- The `this` keyword in functions behaves differently in strict mode.
- The `this` keyword refers to the object that called the function.
- If the object is not specified, functions in strict mode will return `undefined` and functions in normal mode will return the global object (window):

Finally there is also a list of keywords that are reserved for future javascript versions which can **not** be used as variables names in **strict** mode.

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

Exercises

There are no exercises for this module.