Strict Mode

JavaScript strict mode is a feature introduced in ECMAScript 5 (ES5) that allows developers to opt into a restricted version of JavaScript, where certain actions that might have been ignored or failed silently in normal mode will instead generate errors. This mode helps developers write cleaner, safer, and more reliable code by catching common coding mistakes and discouraging the use of problematic language features.

1. Enabling strict mode:

Strict mode is enabled at the beginning of a script or a function by adding the string 'use strict'; (including the quotes) at the beginning of the script or function.

```
// Enabling strict mode in a script
'use strict';
// Enabling strict mode in a function
function myFunction() {
    'use strict';
    // Function code
}
```

2. **Examples of strict mode effects:**

   Undefined variables: Using a variable without declaring it first will result in a reference error.

```
'use strict';
x = 10; // ReferenceError: x is not defined
```

3. **Assignment to read-only properties:**

   Attempting to assign a value to a read-only property, such as NaN, Infinity, or undefined, will throw a type error.

```
'use strict';
NaN = 5; // TypeError: Cannot assign to read only property 'NaN' of object '#<Global>'
```

4. **Deletion of undeletable properties:**

   Deleting undeletable properties will throw a type error.

```
'use strict';
delete Object.prototype; // TypeError: Cannot delete property 'prototype' of function Object() { [native code] }
```

5. **Function parameter name duplication:**

Defining multiple function parameters with the same name will throw a syntax error.

```
'use strict';
function myFunction(a, a, b) {} // SyntaxError: Duplicate parameter name not allowed in this context
```

6. **Octal syntax:**

Octal syntax is not allowed in strict mode, and using it will throw a syntax error.

```
'use strict';
var num = 0123; // SyntaxError: Octal literals are not allowed in strict mode.
```

These are just a few examples of how strict mode helps catch errors and enforce best practices in JavaScript development. By using strict mode, developers can write more robust and maintainable code.