8.8 Strict mode

Applying strict mode

The flexible nature of JavaScript can lead to programming errors that are difficult to find.

PARTICIPATION ACTIVITY 8.8.1: Misspelling a variable name can be difficult to detect.

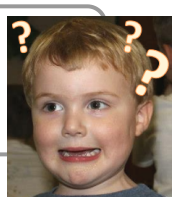
1 2 3 2x speed

```
let humanLivesRemaining = 3;

// Set to 1 if greater than 0
if (humanLivesRemaining > 0) {
   humanLivesRemaing = 1;
}

console.log(humanLivesRemaining);
```

humanLivesRemaining 3

humanLivesRemaing 1

3



The programmer may wonder "Why is humanLivesRemaining still 3?"

Captions

1. Variable humanLivesRemaining is declared and initialized to 3.
2. A new variable humanLivesRemaing is created and assigned 1 unintentionally, because the programmer misspelled "humanLivesRemaining".
3. The programmer may wonder "Why is humanLivesRemaining still 3?"

Feedback?

A programmer can make the JavaScript interpreter catch mistakes like mistyped variable names using strict mode. ***Strict mode*** makes a JavaScript interpreter apply a set of restrictive syntax rules to JavaScript code.

To enable strict mode for an entire script, the statement `"use strict"` must be placed before any other statements.

PARTICIPATION ACTIVITY 8.8.2: Strict mode causes misspelled variable assignment to throw an exception.

1 2 3 4 2x speed

humanLivesRemaining 3

```
"use strict";

let humanLivesRemaining = 3;

// Set to 1 if greater than 0
if (humanLivesRemaining > 0) {
   humanLivesRemaing = 1;
}

console.log(humanLivesRemaining);
```

ReferenceError exception!

Exceptions are displayed in the browser's developer console.

Captions

1. "use strict" enables strict mode.
2. Variable humanLivesRemaining is declared and initialized to 3.
3. Assignment to non-existing variable humanLivesRemaing causes the JavaScript interpreter to throw a ReferenceError.
4. Exceptions are displayed in the browser's developer console.

Feedback?

PARTICIPATION ACTIVITY 8.8.3: Strict errors.

Which scripts have strict errors?

1)
```
"use strict";
x = 10;
```

- (•) Error
- ( ) No error

Correct

Strict mode requires all variables to be declared, and `x` is not declared.

2)
```
"use strict";
let z;
if (z > 0) {

console.log("true");
}
```

- ( ) Error
- (•) No error

Correct

Although `z` is not initialized to a value, strict mode does not restrict comparison of `undefined` variables.

3)
```
"use strict";
let p = { x: 5,
x: 10};
```

- ( ) Error
- (•) No error

Correct

The property `x` is assigned twice. Strict mode in ECMAScript 5 does not allow `x` to be re-initialized to 10, but strict mode in ECMAScript 6 and above allows duplicate property names.

4)
```
"use strict";
function test(a,
b, a) {
   return a - b;
}
```

- (●) Error
- ( ) No error

Correct

Strict mode does not allow duplicate parameter names, and `a` appears twice in the parameter list.

5)
```
"use strict";
let x = 2 + 04 +
8;
```

- (●) Error
- ( ) No error

Correct

Strict mode does not allow octal literals (numbers that begin with a zero), and `04` starts with a zero.

6)
```
"use strict";
let p = {
   get x() {
return 0; }
};

p.x = 1;
```

- (●) Error
- ( ) No error

Correct

Strict mode does not allow assignment to getter-only properties, and `x` only has a getter.

Feedback?

Applying strict mode to functions

Strict mode may apply to only a function by placing `"use strict"` at the beginning of the function.

Figure 8.8.1: A strict function and a regular function.

```
function strict() {
   "use strict";
   // All code in this function is
strict
}

function notStrict() {
   // Not in strict mode
}
```

Feedback?

PARTICIPATION ACTIVITY 8.8.4: Strict errors in functions.

Which scripts have strict errors?

1)
```
x = 5;
function test() {
   "use strict";
   let y = 1;
}
```

- ○ Error
- ◉ No error

Correct

Although `x` is initialized without being declared, only the function `test()` uses strict mode.

2)
```
function test(x)
{
   "use strict";
   let x = 1;
}
```

- ○ Error
- ◉ No error

Correct

Although declaring a local variable `x` with the same name as the parameter `x` is confusing, strict mode does not prohibit the action.

3)
```
function test() {
   "use strict";
   let interface
= 1;
}
```

- ◉ Error
- ○ No error

Correct

ECMAScript 6 reserved words like `interface`, `implements`, and `private` may not be used as variable or function names in strict mode.

4)
```
"use strict";
if (true) {
   function
test() {
      let x = 1;
   }
}
```

- ◉ Error
- ○ No error

Correct

Functions must be declared at the top level of a script or function in strict mode.

5)
```
"use strict";
function zig() {
   function zag()
{
      let x = 1;
   }
}
```

- ○ Error
- ◉ No error

Correct

Functions may be declared inside a function in regular and strict mode.

Feedback?

Exploring further:

- Strict mode (MDN)

How was this section?

Provide section feedback