



JavaScript debugging

Ángel Tornero Hernández
Gabriel García Jaubert

JavaScript debugging

What is debugging? **01**

Types of errors **02**

Debugging strategies **03**

04 **Debugging with Node**

05 **Debugging with VSCode**

06 **Debugging with Chrome**



What is debugging?

In terms of software:

- Identifying bugs
- Removing errors

Attention!

Investing time on learning how to debug will save you time solving errors on the future.



What is a debugger?

Computer program for examining the state of your program while running.

How does the debugging process work?

1. Reproduce the problem
2. Describe the bug
3. Capture the program snapshot when the problem appears.
4. Analyse the snapshot.
5. Fix the existing bug.

How does the debugging process work?

1. Reproduce the problem
2. Describe the bug
3. Capture the program snapshot
when the problem appears.
4. Analyse the snapshot.
5. Fix the existing bug.

Careful!

**Try to not create more
bugs when fixing
errors.** { ☹️ 😊 }

Types of errors



SYNTAX ERRORS



RUNTIME ERRORS



LOGIC ERRORS

Syntax errors



These are errors where the interpreter finds something wrong with your program, and you can't even try to execute it.

Example:

```
1  const score = (coorX, coorY) => {  
2    let module = Math.sqrt(coorX * coorX + coorY * coorY);  
3    if (module <= 1 { ←  
4      return 10;  
5    }  
6    if (module <= 5) {  
7      return 5;  
8    }  
9    if (module <= 10) {  
10     return 1;  
11   }  
12   return 0;  
13 };
```

Syntax errors



These are errors where the interpreter finds something wrong with your program, and you can't even try to execute it.

Example:

```
1  const score = (coorX, coorY) => {  
2    let module = Math.sqrt(coorX * coorX + coorY * coorY);  
3    if (module <= 1 {   
4      return 10;  
5    }  
6    if (module <= 5) {  
7      return 5;  
8    }  
9    if (module <= 10) {  
10     return 1;  
11   }  
12   return 0;  
13 };
```

Error:

```
if (module <= 1 {  
                  ^  
SyntaxError: Unexpected token '{'
```

Runtime errors



Also called exceptions, occur during execution

Example:

```
1  const product = (firstFactor, secondFactor) => {  
2    |  return firstFactor * secondFactor;  
3  };  
4  
5  console.log(taradiddle(4, 5));
```

Runtime errors



Also called exceptions, occur during execution

Example:

```
1  const product = (firstFactor, secondFactor) => {  
2    |  return firstFactor * secondFactor;  
3  };  
4  
5  console.log(taradiddle(4, 5));
```

Error:

```
ReferenceError: taradiddle is not defined
```

Logic errors



They occur when you make a mistake in the logic that drives your script and you do not get the result you expected.

Example:

```
18 ✓ const multiplicationTables = () => {  
19   ✓   for (let i = 1; i <= 10; i++) {  
20       console.log(`Tabla del ${i}`);  
21   ✓   for (let j = 1; j <= 10; i++) {  
22       |   console.log(`${i} * ${j} = ${i * j}`);  
23       |   }  
24       }  
25   }  
26   multiplicationTables();
```

Logic errors



They occur when you make a mistake in the logic that drives your script and you do not get the result you expected.

Example:

```
18 ✓ const multiplicationTables = () => {  
19   for (let i = 1; i <= 10; i++) {  
20     console.log(`Tabla del ${i}`);  
21   for (let j = 1; j <= 10; i++) {  
22     console.log(`${i} * ${j} = ${i * j}`);  
23   }  
24 }  
25 }  
26 multiplicationTables();
```

Output:

```
20000 1 = 20000  
26869 * 1 = 26869  
26870 * 1 = 26870  
26871 * 1 = 26871  
26872 * 1 = 26872  
26873 * 1 = 26873  
26874 * 1 = 26874  
26875 * 1 = 26875  
26876 * 1 = 26876
```



Do not worry!

Bugs are frequent even
for senior programmers

Do not worry!

Bugs are frequent even
for senior programmers

This is why testing was invented

Ariane 5

4th june, 1996
French Guiana



Ariane 5

4th june, 1996.
French Guiana

~\$370 millions



Some debugging strategies



**Incremental program
development**



Error-detection tools



Backtracking



Bug clustering



Binary search



Use a debugger

Strict mode

'use strict'; -> Defines that JavaScript code should be executed in "strict mode".

Strict mode

'use strict'; -> Defines that JavaScript code should be executed in "strict mode".

For example:

Strict mode

1. Assign values to non-writable properties.

Strict mode

1. Assign values to non-writable properties.
2. Mistyping a variable name creates a new global variable.

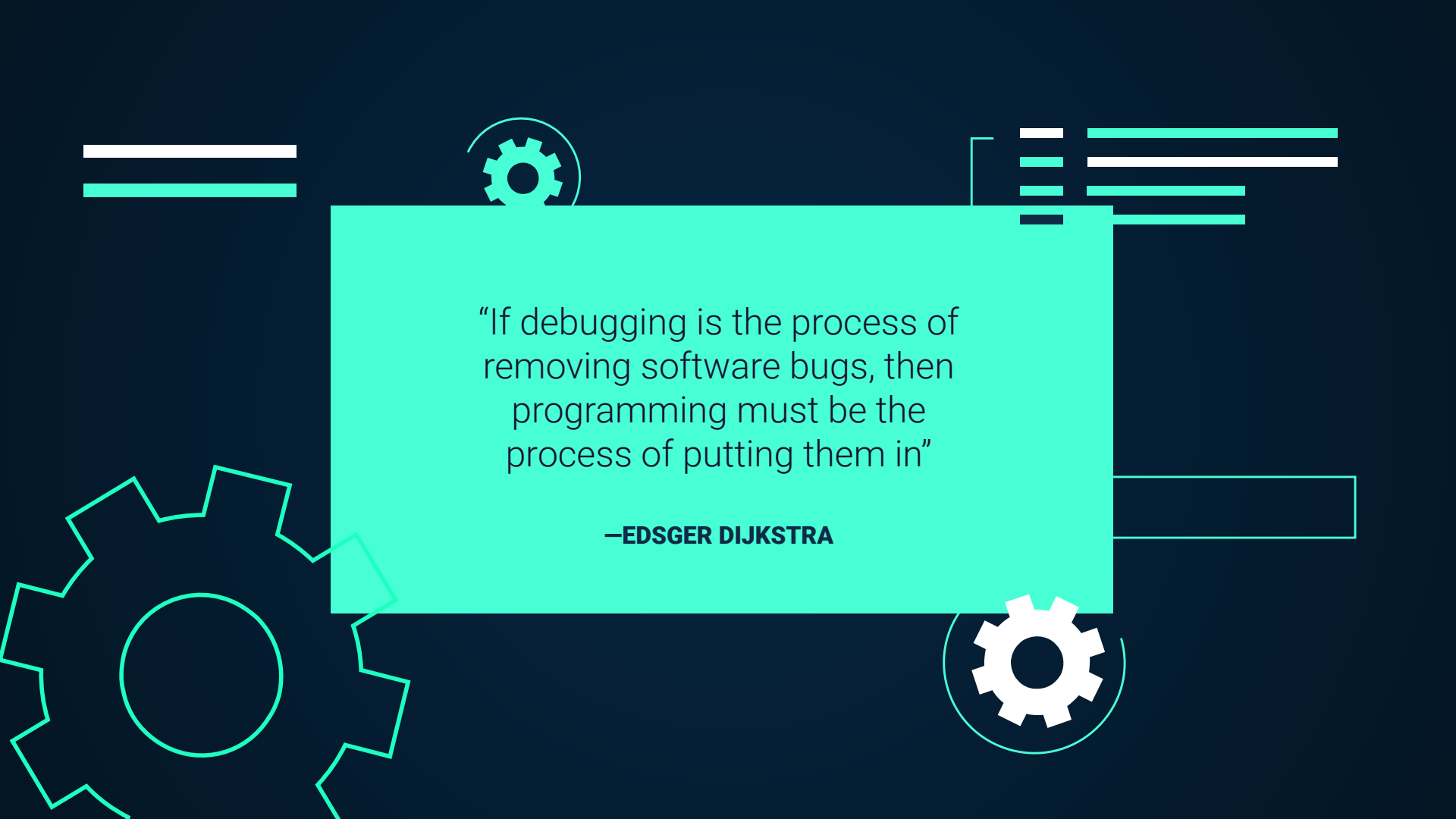
Strict mode

1. Assign values to non-writable properties.
2. Mistyping a variable name creates a new global variable.
3. Any assignment to a non-writable property, a getter-only property, a non-existing property...

Breakpoints

It is an intentional stopping or pausing place in a program.



The background is a dark navy blue. It features several decorative elements: a large, light blue gear outline in the bottom left corner; a smaller, solid light blue gear with a circular outline above the central text box; a solid light blue gear with a circular outline in the bottom right corner; and various horizontal and vertical lines in white and light blue scattered across the top and right sides of the image.

“If debugging is the process of
removing software bugs, then
programming must be the
process of putting them in”

—EDSGER DIJKSTRA

Debugging with Node



Debugging with VSCode



Debugging with Chrome



Other tools



About us

Ángel Tornero Hernández

- angel.tornero.27@ull.edu.es

Gabriel García Jaubert:

- gabriel.jaubert.38@ull.edu.es

Bibliography

- "What Is Debugging?" BBC Bitesize. BBC, September 5, 2019.
<https://www.bbc.co.uk/bitesize/topics/zkcqn39/articles/ztkx6sq>
- Quick, James "Jamesqquick." GitHub.
<https://github.com/jamesqquick>
- "RisingStack." GitHub.
<https://github.com/risingstack>
- Matuszek, David. Director of MCIT, 2001 - 2015. "Errors."
<https://www.cis.upenn.edu/~matuszek/General/JavaSyntax/errors.html>

Repository

<https://github.com/PAI-ULL/2020-2021-pai-trabajo-debugging-debugging-presentation.git>

Thanks!
Questions?