
Software testing

Dimitri Merejkowsky

2026-08-21

TABLE OF CONTENTS

1	Welcome!	1
---	----------	---

WELCOME!

This online course is meant to be a gentle introduction to software testing, designed for people who already know how to code.

Note that the sources of this website are available [on my forge](#), and that the contents are also available as a [pdf document](#)

Finally, note that the contents is licensed CC BY 4.0 which means you are free to share and adapt this course anyway you want, at the sole condition to give me appropriate credit. (More details on the [Creative Commons website](#))

1.1 Introduction

Let's dig in right away !

1.1.1 A silly bug

FizzBuzz

Rules

Count from 1 to 100

- If divisible by 3, print "Fizz"
- If divisible by 5, print "Buzz"
- If divisible by 15, print "FizzBuzz"
- Otherwise, print the number

First implementation

```
const fizzBuzz = (i: number): string => {  
  if (i % 15 === 0) {  
    return "fizzbuzz";  
  }  
  if (i % 3 === 0) {  
    return "fizz";  
  }  
  if (i % 5 === 0) {  
    return "buzz";  
  }  
  return i.toString(10);  
};
```

Quick! Where is the bug?

Well, let's write a *main* and run it:

```
const main = () => {  
  for (let i = 1; i <= 30; i++) {  
    console.log(fizzBuzz(i));  
  }  
};  
  
main();
```

This gives:

```
1  
2  
fizz  
4  
buzz  
fizz  
7  
8  
fizz  
buzz  
11  
fizz  
13  
14  
fizz  
16  
17  
fizz  
19  
buzz  
fizz  
22  
23  
fizz  
buzz  
26  
fizz  
28  
29  
fizz
```

Surely, something is wrong : we never see “fizzbuzz” in the output !

Fixing the bug

By reading the code carefully, we can see that the bug occurs if the number is divisible by both 3 and 5.

We get through the first *if* first, and return early.

There are many ways to fix the code, one solution is to move the `if (number % 15 === 0)` first:

```
export const fizzBuzz = (i: number): string => {
  if (i % 15 === 0) {
    return "fizzbuzz";
  }
  if (i % 3 === 0) {
    return "fizz";
  }
  if (i % 5 === 0) {
    return "buzz";
  }
  return i.toString(10);
};
```

Writing some tests

Writing a main and checking the output by hand is not fun, so let's write some tests instead:

```
import { fizzBuzz } from "./fizzbuzz.ts";
import { assertEquals } from "@std/assert";

Deno.test("Print 'fizz' when divisible by 3", () => {
  assertEquals(fizzBuzz(3), "fizz");
  assertEquals(fizzBuzz(6), "fizz");
  assertEquals(fizzBuzz(9), "fizz");
});

Deno.test("Print 'buzz' when divisible by 5", () => {
  assertEquals(fizzBuzz(5), "buzz");
  assertEquals(fizzBuzz(10), "buzz");
  assertEquals(fizzBuzz(20), "buzz");
});

Deno.test("Print 'fizzbuzz' when divisible by 15", () => {
  assertEquals(fizzBuzz(15), "fizzbuzz");
  assertEquals(fizzBuzz(30), "fizzbuzz");
});

Deno.test("Print number otherwise", () => {
  assertEquals(fizzBuzz(2), "2");
  assertEquals(fizzBuzz(4), "4");
});
```

Running the tests

```
$ deno test fizzbuzz_test.ts

Check fizzbuzz_test.ts
running 4 tests from ./fizzbuzz_test.ts
Print 'fizz' when divisible by 3 ... ok
Print 'buzz' when divisible by 5 ... ok
Print 'fizzbuzz' when divisible by 15 ... ok
Print number otherwise ... ok
```

(continues on next page)

(continued from previous page)

ok | 4 passed | 0 failed (4ms)

1.1.2 Our first tests

As a reminder, here's our final implementation:

```
export const fizzBuzz = (i: number): string => {
  if (i % 15 === 0) {
    return "fizzbuzz";
  }
  if (i % 3 === 0) {
    return "fizz";
  }
  if (i % 5 === 0) {
    return "buzz";
  }
  return i.toString(10);
};
```

Note that we *export* the *fizzbuzz* function, so we can import it later on.

Writing some tests

Writing a main and checking the output by hand is not fun, so let's write some tests instead:

```
import { fizzBuzz } from "./fizzbuzz.ts";
import { assertEquals } from "@std/assert";

Deno.test("Print 'fizz' when divisible by 3", () => {
  assertEquals(fizzBuzz(3), "fizz");
  assertEquals(fizzBuzz(6), "fizz");
  assertEquals(fizzBuzz(9), "fizz");
});

Deno.test("Print 'buzz' when divisible by 5", () => {
  assertEquals(fizzBuzz(5), "buzz");
  assertEquals(fizzBuzz(10), "buzz");
  assertEquals(fizzBuzz(20), "buzz");
});

Deno.test("Print 'fizzbuzz' when divisible by 15", () => {
  assertEquals(fizzBuzz(15), "fizzbuzz");
  assertEquals(fizzBuzz(30), "fizzbuzz");
});

Deno.test("Print number otherwise", () => {
  assertEquals(fizzBuzz(2), "2");
  assertEquals(fizzBuzz(4), "4");
});
```

Running the tests

```
$ deno test fizzbuzz_test.ts
```

```
Check fizzbuzz_test.ts
running 4 tests from ./fizzbuzz_test.ts
Print 'fizz' when divisible by 3 ... ok
Print 'buzz' when divisible by 5 ... ok
Print 'fizzbuzz' when divisible by 15 ... ok
Print number otherwise ... ok

ok | 4 passed | 0 failed (4ms)
```

1.2 Specifications, requirements, and bugs