Extracted from:

JavaScript Brain Teasers

Exercise Your Mind

The Pragmatic Bookshelf

Dallas, Texas

The Pragmatic Programmers

JavaScript Brain Teasers

Exercise Your Mind

Faraz K. Kelhini
edited by Margaret Eldridge

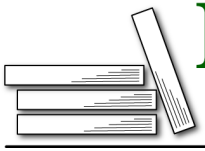
JavaScript Brain Teasers

Exercise Your Mind

Faraz Kelhini

The Pragmatic Bookshelf

Dallas, Texas



For our complete catalog of hands-on, practical, and Pragmatic content for software developers, please visit *https://pragprog.com*.

For sales, volume licensing, and support, please contact *support@pragprog.com*.

For international rights, please contact *rights@pragprog.com*.

ISBN-13: 979-8-88865-050-9
Encoded using the finest acid-free high-entropy binary digits.
Book version: B1.0—October 26, 2023

Puzzle 2

The Usurper

the_usurper/the_usurper.js

```
// Calculate the area of a rectangle
function calculateArea(length, width) {
  return length * width;
}

// Calculate the area of a square
function calculateArea(length) {
  return length * length;
}

console.log(calculateArea(4, 6));
```

Guess the Output

Try to guess what the output is before moving to the next page.

You might have expected the first function, which accepts two arguments, to be executed, but the second function is executed, resulting in the following output:

```
16
```

Discussion

JavaScript doesn't support *function overloading* like you might be used to in other languages. Function overloading means having multiple functions with the same name but different parameter lists, and the right function gets called based on the arguments you provide.

In JavaScript, if you define multiple functions with the same name, the last one you define will replace the previous functions. But there's a workaround. You just need to be a little creative.

Instead of having separate functions, you can write a single function and then check the types and number of arguments inside it. Based on what you find, you can make your function behave differently. It's a bit different from traditional function overloading, but it gets the job done in JavaScript:

the_usurper/the_usurper_ex1.js

```
function calculateArea(length, width) {
  if (width === undefined) {
    // Calculate area of a square
    return length * length;
  } else {
    // Calculate area of a rectangle
    return length * width;
  }
}

console.log(calculateArea(5));       // → 25 (Area of a square)
console.log(calculateArea(4, 6));    // → 24 (Area of a rectangle)
```

In this example, we have a function that can calculate the area of both squares and rectangles. Here's how it works: If we call the function with just one argument, it knows that we want to calculate the area of a square. So, it squares the `length` value and gives us the area we want.

But, if we decide to pass two arguments, the function realizes that we're dealing with a rectangle. So, it multiplies the `length` and `width` values together and we get the area of that rectangle.

This trick allows us to have a single function that does all the work. Now, it's worth mentioning that with the introduction of ECMAScript 2015 (ES6), JavaScript got some other new tricks up its sleeve. Default function parameters and the rest parameter syntax can help us handle different numbers of arguments with more flexibility.

It's good to keep in mind that true function overloading based on parameter types isn't something JavaScript has built-in. But hey, we can still achieve some pretty cool stuff with what JavaScript offers!

Further Reading

Function overloading on Wikipedia

https://en.wikipedia.org/wiki/Function_overloading

Default Parameters on MDN

https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Functions/Default_parameters

Rest Parameters on MDN

https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Functions/rest_parameters