Concise Arrow Function Syntax

Arrow functions are already less verbose than function declarations and function expressions, but you can write them in a more concise way.

Arrow Functions with One Parameter

If your arrow function accepts a single argument, like the `square` function below, you can omit the parentheses:

```
const square = x => {
  return x * x;
}

square(10); // 100
```

Arrow Functions with Multiple Parameters

JavaScript does not allow you to remove the parentheses from arrow functions with no or multiple parameters; only functions with a single parameter.

The following arrow function expression (without parentheses) throws a syntax error:

```
const getArea = width, length, unit => {
  const area = width * length;
  return `${area} ${unit}`;
}

getArea(10, 20, 'sq ft');
// Uncaught SyntaxError: Missing initializer in const declaration
```

You need to include the parentheses when working with multiple (or no) parameters:

```
const getArea = (width, length, unit) => {
  const area = width * length;
  return `${area} ${unit}`;
}

getArea(10, 20, 'sq ft');  // 200 sq ft
```

Arrow Functions as One-line Statements

If your function body (the part between the curly braces) is only one line of code, you can make the function more concise by omitting the `return` keyword and placing everything on one line:

```
const square = x => { x * x };
```

When you omit the `return` keyword in a single-line function, the value is returned automatically (or implicitly). This is called an **implicit return**.

In fact, when using an implicit return, you don't even need the curly braces (they are optional). You can reduce the arrow syntax further by writing the function like this:

```
const square = x => x * x;

const multiply = (x, y) => x * y;
```

```
const add = (a, b) => a + b;

const subtract = (a, b) => a - b;
```

Single-line Functions with No Parameters

Finally, a single-line arrow function with no parameters requires parentheses before the arrow (=>) token, otherwise JavaScript produces a syntax error. For example:

```
const name = 'Jesse';
const greeting = => alert(`Greetings, ${name}!`);
greeting(); // Uncaught SyntaxError: Unexpected token '=>'
```

The following is the correct way:

```
const name = 'Jesse';
const greeting = () => alert(`Greetings, ${name}!`);
```

This shorter syntax might seem strange at first, and take a little getting used to, but compared to the following expression:

```
const greeting = function() {
  alert(`Greetings, ${name}!`);
};
```

it can make parts of your code more concise and easier to read. However, even if your function meets the criteria for concise syntax, it's still perfectly acceptable to use the standard arrow function syntax.

Ok, I got it

- Questions?2
- 

conflictingcode

11,222 Points

Shouldn't the final code example on this stage end with a semicolon (...compared to the following function:)?

Posted on Jul 27, 2020 by conflictingcode

conflictingcode

11,222 Points

- JavaScript
- JavaScript Functions