Title: Arrow Functions

Answer:

So far, we have gone through how to define functions using the `function` keyword. However, there is a newer, more concise method of defining a function known as **arrow function expressions** as of ECMAScript 6 (https://www.ecma-international.org/ecma-262/6.0/). Arrow functions, as they are commonly known, are represented by an equals sign followed by a greater than sign: `=>` .

Arrow functions are always anonymous functions and a type of function expression. We can create a basic example to find the product of two numbers.

arrowFunction.js

```
// Define multiply function
const multiply = (x, y) => {
    return x * y;
}

// Invoke function to find product
multiply(30, 4);
```

```
Output
120
```

Instead of writing out the keyword `function` , we use the `=>` arrow to indicate a function. Otherwise, it works similarly to a regular function expression, with some advanced differences which you can read about under Arrow Functions on the Mozilla Developer Network (https://developer.mozilla.org/en-US/docs/Web/JavaScript/Guide/Functions#Arrow_functions).

In the case of only one parameter, the parentheses can be excluded. In this example, we're squaring `x` , which only requires one number to be passed as an argument. The parentheses have been omitted.

```
// Define square function
const square = x => {
    return x * x;
}

// Invoke function to find product
square(8);
```

```
Output
64
```

Note: In the case of no parameters, an empty set of parentheses `()` is required in the arrow functions.

With these particular examples that only consist of a `return` statement, arrow functions allow the syntax to be reduced even further. If the function is only a single line `return` , both the curly brackets and the `return` statement can be omitted, as seen in the example below.

```
// Define square function
const square = x => x * x;

// Invoke function to find product
square(10);
```

```
Output
100
```

All three of these types of syntax result in the same output. It is generally a matter of preference or company coding standards to decide how you will structure your own functions.

Tags: functions / methods, javascript