Checkpoint 2

Function expressions and arrow functions

Outline

In JavaScript, you can write anonymous functions that don't have names but can still be called. One way to create these functions is by using arrow syntax. In this checkpoint, you'll learn how to write functions using arrow syntax.

Arrow functions are shorter than the functions that you've written so far, and they can make it easier to see what's happening when there are lots of functions. They're also used frequently in higher-order functions, asynchronous code, and React. These are topics that you'll learn about later on.

By the end of this checkpoint, you will be able to do the following:

- Distinguish between function declarations and function expressions
- Write functions using the arrow syntax

Different types of functions

Read through this checkpoint and complete the practice work required. This will give you a full understanding of these concepts.

In JavaScript, there are two different ways to create functions: function declarations and function expressions. A *function declaration*, also called a *function definition* or *function statement*, is JavaScript code that creates a new function with a name. This is the most common way to create functions, and it's the approach that you've seen so far in this program. A *function expression* is a function created in a place where there would otherwise be a value. Functions created this way may or may not have a name.

Suppose that you wish to create a function that accepts a `park` object in the form `{ name: "Biscayne", rating: 4.2 }` and returns `"Excellent!"` if the rating was greater than 4 and `"Good"` otherwise. Here are three different ways to implement that same function:

Outline

```
// Function declaration
function ratingAsText(park) {
  console.log("This is a function declaration.");
  return park.rating > 4 ? "Excellent!" : "Good";
}

// Function expression
const ratingAsTextNamed = function ratingToWord(park)
  console.log("This is a function expression that has
  return park.rating > 4 ? "Excellent!" : "Good";
};

// Anonymous function expression
const ratingAsTextAnonymous = function (park) {
  console.log("This is an anonymous function that doe
```

```
  return park.rating > 4 ? "Excellent!" : "Good";
};
```

The first function is a function declaration. It is the typical way to define functions. If you wanted to call this function, you would use the name of the function along with the argument list. For example, you could call `ratingAsText(park)`.

The second function consists of two parts: the constant (`ratingAsTextNamed`) and the function expression (`ratingToWord(park)`). Notice that the `ratingToWord(park)` function expression uses exactly the same syntax as a function declaration. The only difference is that the function is being assigned to a variable. You can call this function with `ratingAsTextNamed(park)`. However, you cannot call the function using `ratingToWord(park)` unless the function is calling itself (You will learn more about this later.)

Outline

The last function does not have a name, meaning that it is an *anonymous function*. It is an expression that is assigned to the variable `ratingAsTextAnonymous`, and it can be called with `ratingAsTextAnonymous(park)`.

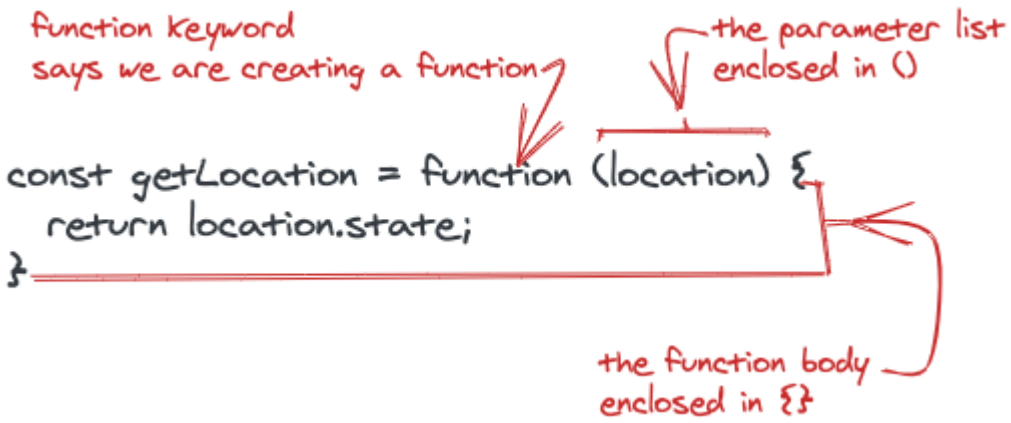
Arrow functions

Now, take a look at the following function expression:

```
const location = {
  name: "Arches",
  state: "Utah",
  geo: {
    lat: 38.68,
```

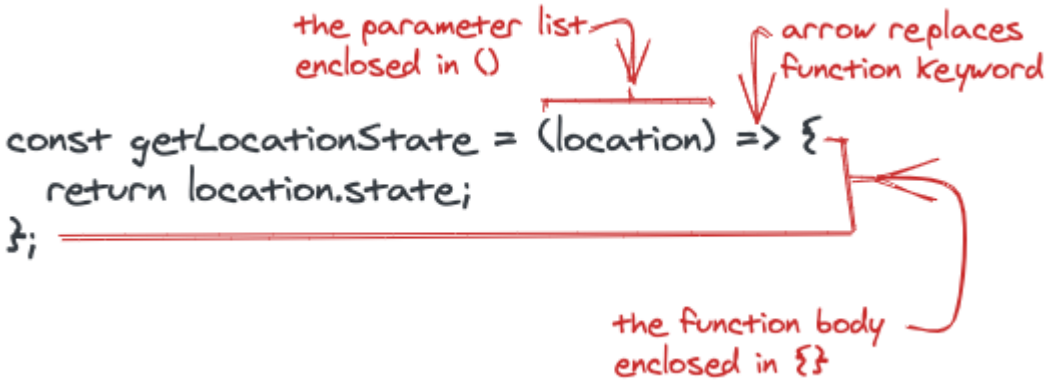
```
    lon: -109.57,
  },
};

const getLocationState = function (location) {
  return location.state;
};
```

The above function will return the `state` value of a given `location` object. Recall the different parts of the function syntax:



Outline

You can rewrite this function using *arrow syntax*, like this:



Instead of using the `function` keyword, this syntax uses an *arrow* `=>` to define the function. Notice that the rest of the syntax is esse y the same.

```
const getLocationState = (location) => {
  return location.state;
};
```

There are a couple of important facts to keep in mind when using an arrow function:

- If there is only one parameter, the parentheses `()` around the parameter are optional. For example, the function above accepts just one parameter, `location`, so you can leave out the parentheses around the parameter and write it like this:

```
const getLocationState = location => {
  return location.state;
};
```

- If the function consists of just a single `return` statement, you can omit the curly brackets `{}` and just have the arrow point to the value being returned. For example, the body of the above function consists of just the statement `return location.state`, you can drop the `return` keyword and the `{}`. This means that you can write the function in an even more concise way:

```
const getLocationState = location => location.s
```

Next, take a look at some more examples of arrow functions and their equivalent function declarations.

Get location name

```
// Arrow function
const getLocationName = (location) => location.name;

// Function declaration
function getLocationName(location) {
  return location.name;
}
```

Get Google Map URL

Note: The return value in this function is quite long. If you still want to avoid using the `return` keyword, you can use parentheses.

```
// Arrow function
const getGoogleMapURL = ({ geo: { lat, lon } }, zoom
  `https://www.google.com/maps/@${lat},${lon},${zoom}
);

// Function declaration
function getLocationCoordinates({ geo: { lat, lon } }
  return `https://www.google.com/maps/@${lat},${lon},
}
```

Outline

Which is better?

At the moment, there is no "right" answer as to whether you should use a function declaration or a function expression. Arrow functions are a newer feature of JavaScript, and you will see them often in upcoming checkpoints.

Note: Later on, you'll learn another reason to use arrow functions. This reason will have to do with the `this` keyword and something called *execution context*.

Checkpoint

This checkpoint will be autograded. Please click the link below to open your assignment in a new tab. Once you complete the assignment, you will see a button allowing you to submit your answers and move on to the next checkpoint.

Your work

Outline

03.29.21 **Approved**

Completed Next checkpoint

How would you rate this content?

Report a typo or other issue

Go to Overview

Outline