Checkpoint 9

Scope

Do you remember scoping? Turns out, you've already learned the basics of scoping in JavaScript. In this checkpoint, you will revisit the concept of scope and explore it further. You'll also learn how to diagram scope.

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

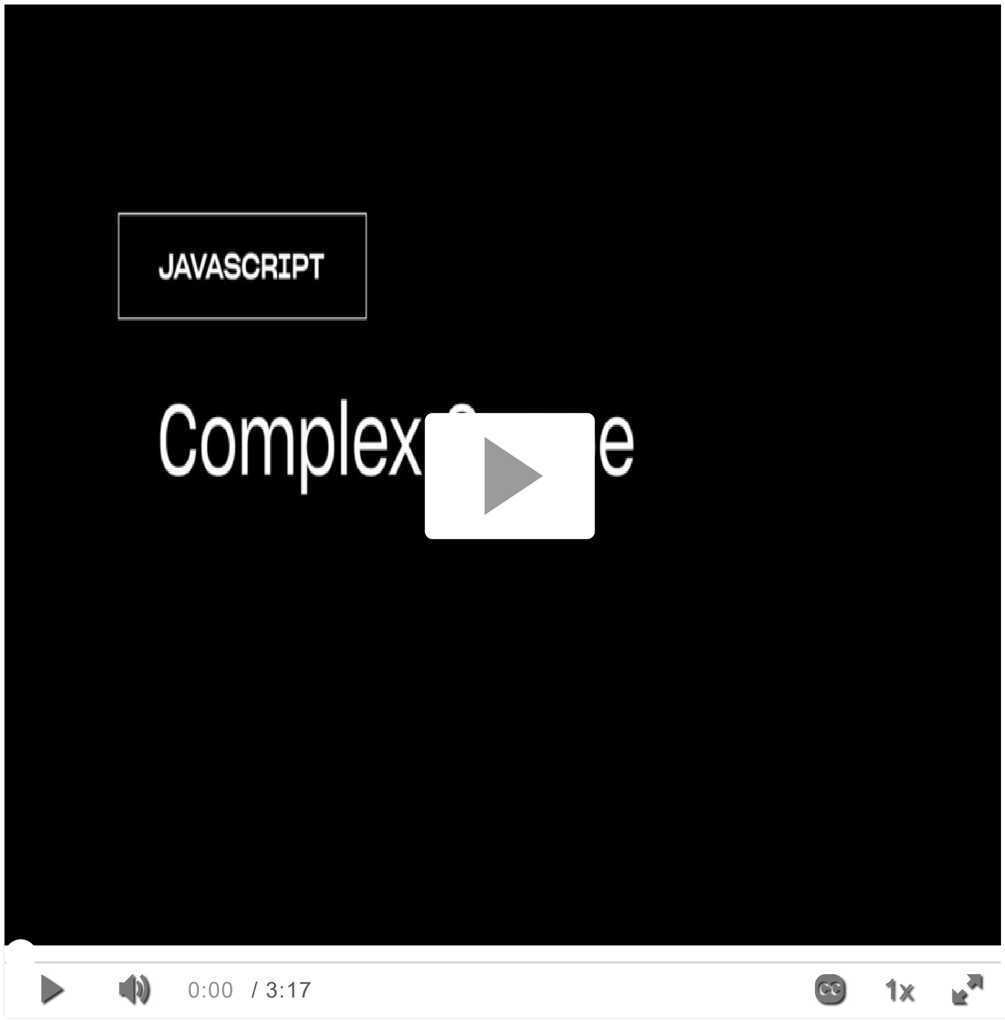
- Evaluate functions with a complicated scope
- Diagram the scope of programs

Complex scope

The following video breaks down how to evaluate functions with complicated scope and diagram the scope of programs. Start by watching the video, and then read thr

Feeling stuck?
Chat live with an expert now. Beta

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



Outline

When you use `let` and `const`, you can easily separate *scope* by simply looking for the curly brackets `{}`. Each pair of curly brackets gives you a new level of scope.

Additionally, anything inside of a set of curly brackets can access reference variables that are set outside of it. Take a look:

```
const DISCOUNT_PERCENTAGE = 0.15;
function discountPricesInCents(products) {
  const result = [];

  for (let i = 0; i < produ
    const product = products[i];
    let price = product.priceInCents;
    if (DISCOUNT_PERCENTAGE > 0) {
      const multiplier = 1 - DISCOUNT_PERCENTAGE;
      price = product.priceInCents * multiplier;
```

Feeling stuck?
Chat live with an expert now. Beta

```
      }
      result.push(price);
    }

    return result;
  }
```

The above code relies on different rules for *block scope* and *global scope*. Here are the details:

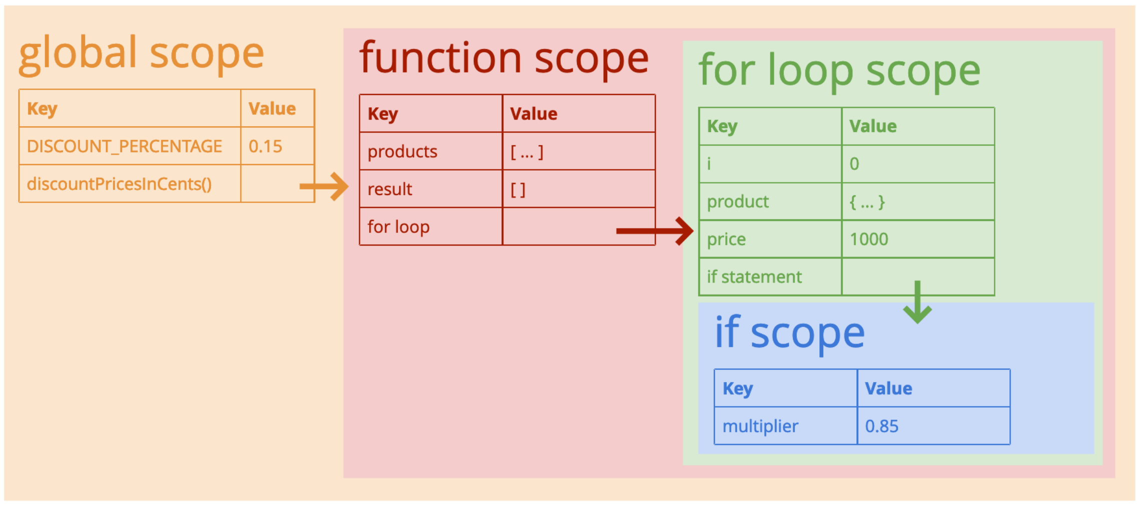
- In the `discountPricesInCents()` function, you can access `DISCOUNT_PERCENTAGE` because it's part of the global scope.
- Later on, if you were to call `discountPricesInCents()`, you would be accessing it from the global scope.
- In the `for` loop statement, you're able to access the `products` parameter because it is shared inside the function scope. If you tried to log `products` outside of the function, you would get an error.
- Inside of the `for` loop, you're able to access `products` and `DISCOUNT_PERCENTAGE` because they are both in outer scopes.
- However, `i`, `product` and `price` are accessible *only* within the `for` loop. You can't access those variables outside of the curly brackets.
- Finally, `multiplier` is accessible *only* inside of the `if` statement. It can't be accessed

Outline

Feeling stuck?
Chat live with an expert now. Beta

Diagramming scope

As you can see above, a complex scope can be hard to understand. To gain some clarity, developers often find it useful to diagram out the scope for a particular function or program. The following is one way to diagram scope.



Key	Value
DISCOUNT_PERCENTAGE	0.15
discountPricesInCents()	

Key	Value
products	[...]
result	[]
for loop	

Key	Value
i	0
product	{ ... }
price	1000
if statement	

Key	Value
multiplier	0.85

Outline

The above diagram demonstrates the following:

1. Each distinct background color represents a level of scope. For example, the `if` statement scope is enclosed inside of the `for` loop scope.
2. At each arrow, a new scope is introduced. For example, inside of the function scope is a `for` loop scope.
3. Each scope has access to the variables of its containers. For example, the function scope has access to the global scope, and the `if` statement scope has ac

Feeling stuck?
Chat live with an expert now. Beta

Reminder: Variable declaration and scope

As you know, variables declared with `let` and `const` can only ever be declared once. But there's one exception to this rule: you may declare variables that are in sibling scopes. Take a look at this example.

```
const DISCOUNT_PERCENTAGE = 0.15;
const product = {
  name: "Black Longline T-Shirt",
  priceInCents: 1500,
  availableSizes: ["XS", "S", "XL", "XXL"],
};

if (DISCOUNT_PERCENTAGE > 0) {
  const multiplier = 1 - DISCOUNT_PERCENTAGE;
  const price = product.priceInCents * multiplier;
  console.log(price);
} else {
  const price = product.priceInCents;
  console.log(price);
}
```

In the case above, `price` is declared twice as a `const` variable, but the scopes are separate. That means that they can both be declared successfully on their own.

The assignment below includes a quiz to test your knowledge. When you're finished, you can use this answer key to check your answers.

Feeling stuck?
Chat live with an expert now. Beta

Checkpoint

This checkpoint will be autograded. Please click the link below pen 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

03.25.21	**Approved**

Completed Next checkpoint

Outline

How would you rate this content?

Report a typo or other issue

Go to Overview

Feeling stuck?
Chat live with an expert now. Beta