Checkpoint 5

Control flow

By this point, you already know how to use `if/else` statements to control the flow of code. And while you can always effectively control the flow of the code with the `if/else` statements you've learned before, there are some additional tools that can make your code simpler. In this checkpoint, you will explore different ways of writing `if/else` statements. You'll also learn about `switch` statements, which offer another way of controlling the flow of programs.

Outline

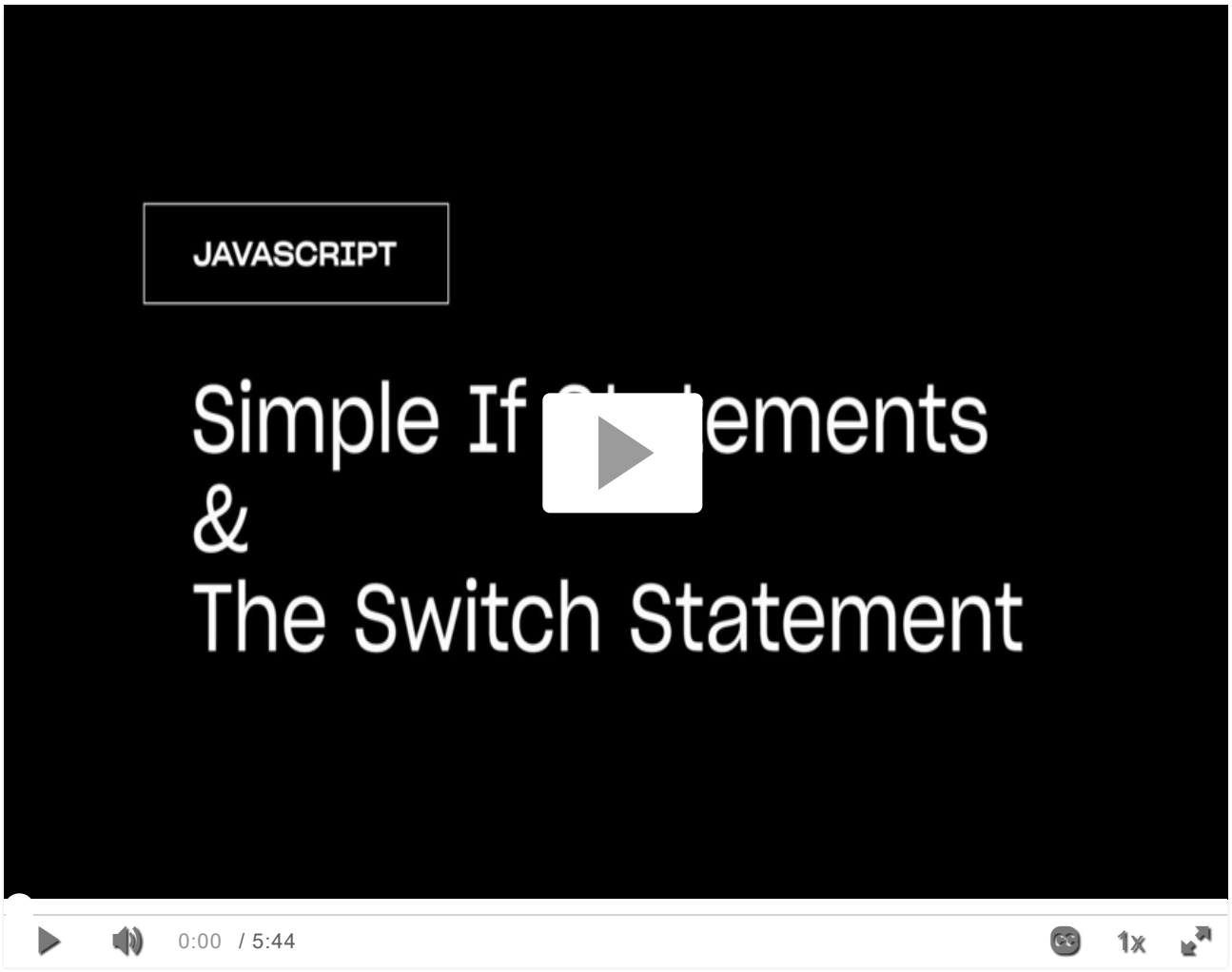
It's important to know the various tools that you have available as a developer. Ultimately, the tool that you'll use in specific cases will likely come down to your personal preferences.

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

- Write `if/else` statements concisely
- Use `switch` statements to control the flow of programs

Simple `if` statements and the `switch` statement

Start by watching the video below, which provides a brief introduction to this topic. Then, read through the rest of the checkpoint and complete the practice work required. This will give you a full understanding of these concepts.



Outline

Single-line `if` statements

There's a lot of talk about `if/else` statements. And with good reason—they're pretty useful. But sometimes you don't need an `else` statement as part of your conditional. Take a look.

```
function getPrice(product) {
  let price = product.priceInCents;
  if (product.onSale) {
    price = price * 0.9;
  }

  return price;
}

const product = {
  priceInCents: 2100,
  name: "Red Beanie",
  size: "L",
  onSale: true,
};
getPrice(product); //> 1890
```

Here, the `getPrice()` function applies a 10% discount to any item that has `onSale` set to a `true` (or *truthy*) value. And although the above code sample is perfectly fine, you may also see the following:

Outline

```
function getPrice(product) {
  let price = product.priceInCents;
  if (product.onSale) price = price * 0.9;

  return price;
}
```

This function definition, by contrast, has removed that set of curly brackets `{}` and has instead added the remaining line, `price = price * 0.9`, to the line above it. However, you should *only* consolidate lines like this in your own code if the content inside the block is *very short*. Otherwise, it will be too difficult to read.

It's important to note that there is no substantive difference between these two options. They simply look different.

The conditional operator

The *conditional operator*, also commonly called the *ternary operator*, is a way to write a short `if/else` statement. It should *only* be used if the overall `if/else` statement is very short. The syntax works like this:

```
(conditional expression) ? (expression if true) : (ex
```

Take a look at the following rewrite of the `getPrice()` function. As you can see, this function is now much more concise, which makes it cleaner and easier to read.

Outline

```
function getPrice(product) {
  return product.onSale ? product.price * 0.9 : produ
}
```

To make sure you understand what is happening here, take a moment to break this down. The expression *before* the question mark `?` is the conditional. If it evaluates to a truthy value, the expression *after* the `?` will be returned. If the conditional evaluates to a falsy value, the expression *after the colon* `:` will be returned.

When it comes to comparing the options discussed here, there's no one best approach. The single-line `if` statement, the conditional operator, and a full `if/else` statement can all work for you. O ime, you'll develop preferences based on which method you find to be

clearest and easiest to read. However, because you're still learning the ropes, you should default to writing out the full statement for the time being. This will help you become comfortable with the process before you start abbreviating it.

Check out MDN to learn more about the conditional operator.

The `switch` statement

Sometimes, like in the cases noted above, your `if/else` statements can be quite short. But sometimes, they can be very long.

Take a look at this example. What do you think this code does?

```
function getStateSalesTax(stateAbbreviation) {
  let result;
  if (stateAbbreviation === "CA") {
    result = 0.0725;
  } else if (stateAbbreviation === "CO") {
    result = 0.029;
  } else if (stateAbbreviation === "GA") {
    result = 0.04;
  } else if (stateAbbreviation === "VT") {
    result = 0.06;
  } else {
    result = 0;
  }

  return result;
}
```

Outline

The code above is actually pretty straightforward—depending on the state that is inputted, a different value (in this case, a sales tax amount) is returned. But this might feel a bit clunky. Fortunately, there's another way you could write the function above: with the `switch` statement.

```
function getStateSalesTax(stateAbbreviation) {
  let result;
  switch (stateAbbreviation) {
    case "CA":
      result = 0.0725;
      break;
    case "CO":
      result = 0.029;
      break;
    case "GA":
      result = 0.04;
      break;
    case "VT":
      result = 0.06;
      break;
    default:
      result = 0;
  }

  return result;
}
```

Outline

So, what's happening here? Right after the `switch` keyword is introduced, the given expression is evaluated. Then, a matching `case` is searched for using strict equality. If anything matches, it runs the code in that `case`. The `break` keyword stops the current `switch` statement, preventing the code from running for any more cases. If no matching value is found, the `default` runs.

Usually, `case` statements will use `break` between each case. This can help you avoid getting behavior or results that you don't expect.

Sometimes, the fallthrough behavior is helpful. See below:

```
function getStateSalesTax(stateAbbreviation) {
  let result;
```

```
  switch (stateAbbreviation) {
    case "CA":
      result = 0.0725;
      break;
    case "CO":
      result = 0.029;
      break;
    case "GA":
      result = 0.04;
      break;
    case "MD":
    case "VT":
    case "WV":
      result = 0.06;
      break;
    default:
      result = 0;
  }

  return result;
}
```

Outline

In this example, `"MD"`, `"VT"`, and `"WV"` all have the same sales tax. Entering any one of those values will set the result to the same value.

The two functions discussed above will work in similar ways. Ultimately, it'll be up to you when you want to use a `switch` statement instead of a longer `if/else` statement.

Check out MDN to learn more about the switch statement.

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

03.25.21 **Approved**

Completed Next checkpoint

Outline

How would you rate this content?

Report a typo or other issue

Go to Overview