6
CONDITIONALS AND LOOPS

Conditionals and loops are two of the most important concepts in JavaScript. A *conditional* says, "If something is `true`, do this. Otherwise, do that." For example, if you do your homework, you can have ice cream, but if you don't do your homework, you don't get the ice cream. A *loop* says, "As long as something is `true`, keep doing this." For example, as long as you are thirsty, keep drinking water.

Conditionals and loops are powerful concepts that are key to any sophisticated program. They are called *control structures* because they allow you to control which parts of your code are executed when and how often they're executed, based on certain conditions you define.

We first need to go over how to embed JavaScript in our HTML file so we can start creating longer programs than we've looked at so far.

EMBEDDING JAVASCRIPT IN HTML

Here is the HTML file we created in Chapter 5, with additions in color and the existing text in gray. (To make this example a little simpler, I've also deleted the link to xkcd.)

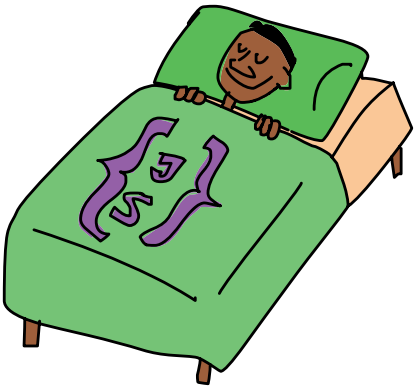
```
<!DOCTYPE html>
<html>
<head>
    <title>My first proper HTML page</title>
</head>

<body>
    <h1>Hello world!</h1>
    <p>My first web page.</p>
    <script>
    var message = "Hello world!";
    console.log(message);
    </script>
</body>
</html>
```

Here we've added a new element, called `script`. This is a special element in HTML. With most HTML elements, the content between the opening and closing tags is displayed on the page. With `script`, on the other hand, everything between the tags is treated as JavaScript and run by the JavaScript interpreter.

Now let's look at the code inside the `script` element:

```
var message = "Hello world!";
❶ console.log(message);
```



Running JavaScript in an HTML file is quite different from running it in the console. When you're using the JavaScript console, each line you type is run as soon as you press ENTER, and the value of that line is printed out to the console. In a web page, the JavaScript is all run from top to bottom at one time, and nothing is automatically printed to the console, unless we tell the browser otherwise. We can use `console.log` to print things out, which will make it easier to see what's going on as we run our programs. The `console.log` method takes any value and prints out, or *logs*, that value to the console. For example, if you load the HTML file from the beginning of this section with the JavaScript console open, you'll see this:

```
Hello world!
```

Calling `console.log(message)` at ❶ caused the string `"Hello world!"` to be printed to the console.

Now that you know how to write longer programs with JavaScript, you can start learning about conditionals.

CONDITIONALS

There are two forms of conditional statements in JavaScript: `if` statements and `if...else` statements. An `if` statement is used to execute a piece of code if something is `true`. For example, *if* you've been good, you get a treat. An `if...else` statement executes one piece of code if something is `true` and another if not. For example, *if* you've been good, you get a treat; *else*, you get grounded.

IF STATEMENTS

The `if` statement is the simplest of JavaScript's control structures. It's used to run code only if a condition is `true`. Return to your

HTML file and replace the two lines inside the `script` element with this:

```
❶ var name = "Nicholas";
❷ console.log("Hello " + name);
❸ if (name.length > 7) {
❹   console.log("Wow, you have a REALLY long name!");
}
```

First, at ❶ we create a variable called `name` and set its value to the string `"Nicholas"`. Then we use `console.log` to log the string `"Hello Nicholas"` at ❷.

At ❸ we use an `if` statement to check whether the length of `name` is greater than 7. If it is, the console will display `"Wow, you have a REALLY long name!"`, using `console.log` at ❹.

As Figure 6-1 shows, an `if` statement has two main parts: the condition and the body. The *condition* should be a Boolean value. The *body* is one or more lines of JavaScript code, which are executed if the condition is `true`.

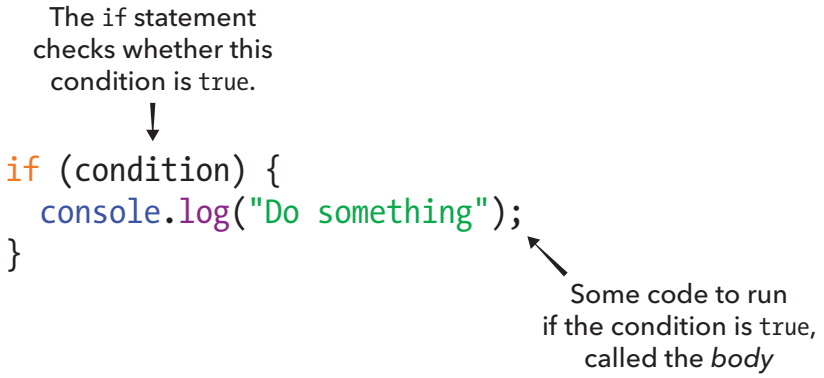


Figure 6-1: The general structure of an `if` statement

When you load your HTML page with this JavaScript in it, you should see the following in the console:

```
Hello Nicholas
Wow, you have a REALLY long name!
```

Because the name *Nicholas* has eight characters, `name.length` returns `8`. Therefore, the condition `name.length > 7` is `true`, which causes the body of the `if` statement to be run, resulting in this

somewhat startling message being logged. To avoid triggering the `if` condition, change the name *Nicholas* to *Nick* (leaving the rest of the code as is):

```
var name = "Nick";
```

Now save the file and reload the page. This time, the condition `name.length > 7` is not `true`, because `name.length` is 4. That means that the body of the `if` statement is not run and all that gets printed to the console is this:

```
Hello Nick
```

The body of an `if` statement is executed only if the condition is `true`. When the condition is `false`, the interpreter simply skips over the `if` statement and moves on to the next line.

IF...ELSE STATEMENTS

As I said before, an `if` statement will execute its body only if the condition is `true`. If you want something else to happen when the condition is `false`, you need to use an `if...else` statement.

Let's extend the example from earlier:

```
var name = "Nicholas";
console.log("Hello " + name);
if (name.length > 7) {
  console.log("Wow, you have a REALLY long name!");
} else {
  console.log("Your name isn't very long.");
}
```

This does the same thing as before, except that if the name *isn't* longer than seven characters, it prints out an alternative message.

As Figure 6-2 shows, `if...else` statements look like `if` statements, but with two bodies. The keyword `else` is placed between the two bodies. In an `if...else` statement, the first body is run if the condition is `true`; otherwise, the second body is run.

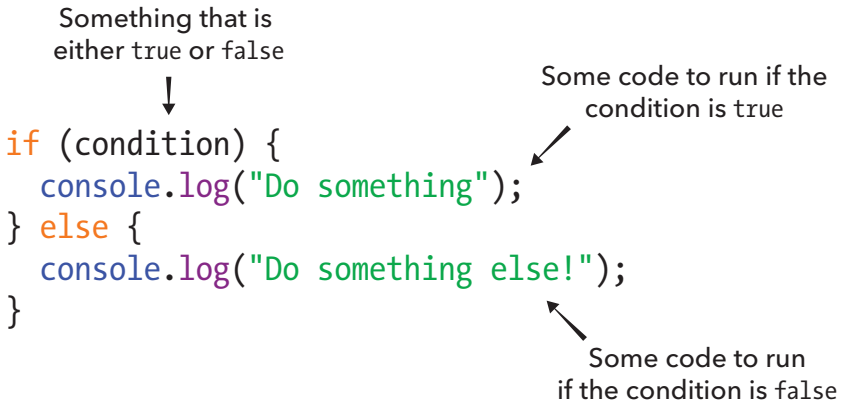


Figure 6-2: The general structure of an `if...else` statement

CHAINING IF...ELSE STATEMENTS

Often we need to check a sequence of conditions and do something when one of them is `true`. For example, say you're ordering Chinese food and you're choosing what to eat. Your favorite Chinese dish is lemon chicken, so you'll have that if it's on the menu. If it's not, you'll have beef with black bean sauce. If *that*'s not on the menu, you'll have sweet and sour pork. In the rare case that none of those options is available, you'll have egg fried rice, because you know all the Chinese restaurants you go to will have that.

```
var lemonChicken = false;
var beefWithBlackBean = true;
var sweetAndSourPork = true;

if (lemonChicken) {
  console.log("Great! I'm having lemon chicken!");
} else if (beefWithBlackBean) {
  console.log("I'm having the beef.");
} else if (sweetAndSourPork) {
  console.log("OK, I'll have the pork.");
} else {
  console.log("Well, I guess I'll have rice then.");
}
```

To create a chain of `if...else` statements, start with a normal `if` statement and, after the closing brace of its body, enter the keywords `else if`, followed by another condition and another body. You can keep doing this until you run out of conditions; there's no

limit to the number of conditions. The final `else` section will run if none of the conditions is `true`. Figure 6-3 shows a generic chain of `if...else` statements.

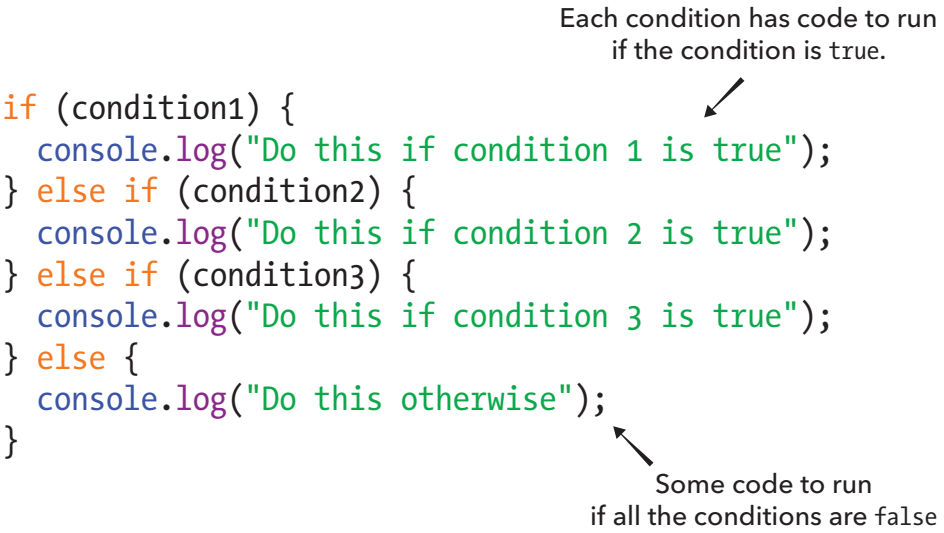
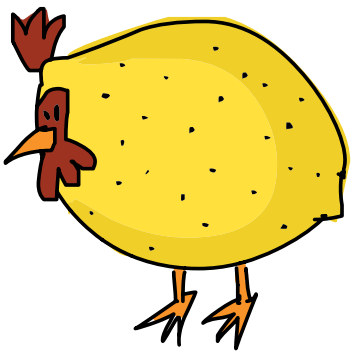


Figure 6-3: Chaining multiple `if...else` statements

You can read this as follows:

1. If the first condition is `true`, execute the first body.
2. Otherwise, if the second condition is `true`, execute the second body.
3. Otherwise, if the third condition is `true`, execute the third body.
4. Otherwise, execute the `else` body.

When you have a chain of `if...else` statements like this with a final `else` section, you can be sure that one (and only one) of the bodies will be run. As soon as a `true` condition is found, its associated body is run, and none of the other conditions is checked. If we run the code in the previous example, `I'm having the beef` will be printed to the console, because `beefWithBlackBean` is the first condition that's found to be `true` in the `if...else` chain. If none of the conditions is `true`, the `else` body is run.



There's one other thing to note: you don't necessarily have to include the final `else`. If you don't, though, and none of the conditions is `true`, then nothing inside the `if...else` chain will be executed.

```
var lemonChicken = false;
var beefWithBlackBean = false;
var sweetAndSourPork = false;

if (lemonChicken) {
  console.log("Great! I'm having lemon chicken!");
} else if (beefWithBlackBean) {
  console.log("I'm having the beef.");
} else if (sweetAndSourPork) {
  console.log("OK, I'll have the pork.");
}
```

In this example, we've left out the final `else` section. Because none of your favorite foods is available, nothing gets printed out (and it looks like you're not going to have anything to eat!).

TRY IT OUT!

Write a program with a `name` variable. If `name` is your name, print out `Hello me!`; otherwise, print `Hello stranger!`. (Hint: Use `===` to compare `name` to your name.)

Next, rewrite the program so it'll say hi to your dad if `name` is set to your dad's name or hi to your mom if `name` is your mom's name. If it's neither of them, say `Hello stranger!` as before.