Chapter 6, Conditional Execution

John M. Morrison

September 27, 2017

Contents

0 Introduction

So far, your JavaScript code runs line-by-line until a function call occurs, then the function runs, and you resume where you left off. There is no facility for making decisions. This is the next order of business. You will be able to make decisions based on what is in the visible symbol tables and have your code react to them.

1 An Aperitif: JavaScript Dialog ("popup") Boxes

There are three major types of popups that are used in JavaScript to obtain textual information from a user.

- The function alert creates a popup that conveys a message. Its argument is a string, which is the message to be conveyed.

- The function prompt creates a popup that asks the user for text. It requires two arguments. The first is the message on the box, the second is the default text if the user enters nothing. The text typed in by the user is returned to the caller.
- The function confirm produces a box with two buttons marked "OK and "Cancel." It accepts one argument, a message from the page. It returns true if the user hits the OK button and false if the user hits the cancel button.

Create this page named `demo.html`.

```
<!doctype html>
<html>
<head>
    <meta charset="utf-8"/>
    <title>JavaScript Dialog Demo</title>
</head>
<body>

<h2> JavaScript Dialog Demo</h2>

<p>There are three major types of popups that are used in JavaScript to obtain
textual information from a user.  You can click on the box types to see an
example of each in the list below.  Notice how we can apply an
onclick attribute to any element in the body of a page.</p>

<ul>
        <li>The function <code onclick="alert('I am an alert box.\nHear me
roar');"> alert</code> creates a popup that conveys a message.  Its argument is
a string, which is the message to be conveyed.</li>
    <li>The function <code
        onclick="alert('I am a prompt box.', 'foo');">prompt</code>
    creates a popup that asks the user for text.  It requires two arguments.
    The first is the message on the box, the second is the default text if the
    user enters nothing.  The text typed in by the user is returned to the
    caller.</li>
    <li>The function <code onclick="confirm('Don\'t ask me what I am.');">
    confirm</code> produces a box with two buttons
    marked "OK and "Cancel."  It accepts one argument, a message from the page.
    It returns <code>true</code> if the user hits the OK button and
    <code>false</code> if the user hits the cancel button.</li>
</ul>
</body>
</html>
```

We will use these boxes to react to obtain user input. We can then use conditional logic, which we are about to discuss, to carry out different actions based on the user's input.

2 Welcome to Dusty's Pub

In this section you will meet some new keywords, `if`, `else if`, and `else`. You will help Dusty to decide if you are of the proper age to imbibe of his appealing brands of rotgut.

To get started, make this file and name it `dusty.html`.

```
<!DOCTYPE html>
<html>
<head>
<title>Welcome to Dusty's Pub!</title>
<meta charset="utf-8"/>
<script type = "text/javascript" src = "pubActions.js"></script>
</head>
<body>
<h2>Dusty's Pub</h2>

<p>Welcome to Dusty's Pub, where no under-age people get served and the
rest get pure attitude!</p>

<button onclick="showAge(getAge())">Click to Enter Age</button>

<p id = "reply"></p>
</body>
</html>
```

We will place our JavaScript in the file `pubActions.js`. To begin, we shall create a button labeled `"Click to enter your age"`. When the button is pushed, a poppy dialog will ask for your age. When you enter it, this page will display your age in the paragraph with id `"reply"`.

To start the process, let us now create the file `pubActions.js` and place this code in it.

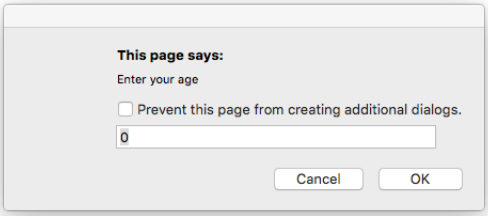
```
function getAge()
{
    var age = prompt("Enter your age", "0");
    return age;
}
```

This function, when called will pop up a dialog box with a text field in it labeled `"Enter your age"`. This function will then return the string entered by the user containing the user's age. If the user fails to enter an age, a default value of `"0"` is used.

Next, we need to create a button with `onclick` attribute that calls `getAge()`. Add this HTML to `dusty.html` right after the first paragraph.

```
<button onclick="getAge();">Click to Enter Age</button>
```

Save everything; once you do this, clicking on the button will bring up a little dialog box with a text area. The user types in the text area and whatever is entered is returned as a JavaScript string.



As of now, the function is called but its return value is never used. Let us now use the function and place the value on the page. To do so, we create a new function in our JavaScript program.

```
function getAge()
{
    var age = prompt("Enter your age", "0");
    return age;
}
function showAge(age)
{
    document.getElementById("reply").innerHTML = "You are " + age + "years old". ;
}
```

This puts a report on the age entered by the user into the paragraph element with id `"reply"`. So far there is nothing really new here.

Now we need to have our JavaScript program decide if a customer is of age. To do this we will use the `if` statement. We demonstrate this by adding a message that sends a customer packing if he is under age.

Now add this function to our JavaScript program.

```
function greetCustomer(age)
{
    if(age < 21)
    {
        document.getElementById("greet").innerHTML
            = "Now get lost, punk!";
    }
}
```

Next, add a new paragraph element with an ID of `"greet"`. Now we get the whole thing to work by modifying the `onclick` attribute for the button.

```
<!DOCTYPE html>
<html>
<head>
<title>Welcome to Dusty's Pub!</title>
<meta charset="utf-8"/>
<script type = "text/javascript" src = "pubActions.js"></script>
</head>
<body>
<h2>Dusty's Pub</h2>

<p>Welcome to Dusty's Pub, where no under-age people get served and the
rest get pure attitude!</p>

<button
    onclick="var input = getAge();showAge(input); greetCustomer(input);">
    Click to Enter Age</button>

<p id = "reply"></p>
<p id = "greet"></p>
</body>
</html>
```

The script in `onclick` does three things.

1. We get the user to enter the age and save it under the name `input`.
2. We reply to the user about his age.
3. We then greet the customer with a message dismissing his request if he us under age.

Next let us test it. Try these cases. Enter 18, then enter 2. When you enter 18 and 2, the application performs as expected. Now enter 35. You will see that no message prints about being under-age. So, let's back up and take a look at the `if` statement in general. Its general form is

```
if(predicate)
{
    blockOfCode;
}
```

The expression `predicate` is a boolean-valued (true/false) expression. As a matter of terminology, we use the term *predicate* for any boolean-valued expression.

The `blockOfCode` can consist of zero or more statements. If the predicate is false, the `blockOfCode` does not execute. The predicate can be any boolean-valued expression involving literals and variables that are in some visible symbol table. The simple `if` allows you to omit (jump over) code you want ignored under certain circumstances, which you specify in the `if` statement's predicate.

Notice that the phrase "if predicate" is a subordinating clause and it is not a complete sentence. This is in indication that a simple `if` is a boss statement. Do not put a semicolon right after any boss statement or it "decapitates" your code, leaving the block right after it to be executed unconditionally. You should do this and observe the effect. You should use the developer tools to see if you get any nastygrams.

3 Using `else`

You can attach an `else` clause to any simple `if` statement as follows.

```
if(predicate)
{
    //code executes if predicate evaluates
    //to true
}
else
{
    //code executes if predicate evaluates
    //to false
}
```

You should now be able to see how to improve Dusty's response. Notice how we have attached an `else` clause to the `if` statement in the function `greetCustomer`. Now try this out, using an age below 21, 21 exactly, and an age above 21. You will see how Dusty responds in each case.

```
function getAge()
{
```

```
    var age = prompt("Enter your age", "0");
    return age;
}
function showAge(age)
{
    document.getElementById("reply").innerHTML = "You are " + age + " years old." ;
}
function greetCustomer(age)
{
    if(age < 21)
    {
        document.getElementById("greet").innerHTML = "Now get lost, punk!";
    }
    else
    {
        document.getElementById("greet").innerHTML = "Name yer poizon, baw!";
    }
}
```

4 Using `else if`

Dusty decides he wants to ask his customers over 65 if they wish to cash their Social Security check. Now we wish to have three different responses based on the customer's action. This is accomplished using the `else if` construct. This is another boss statement. Its usage is as follows

```
if(predicate)
{
    //This code executes if predicate is true.
    //If so, we are done.
}
else if(predicate1)
{
    //This code executes if predicate1 is true.
    //If so, we are done.
}
else if(predicate2)
{
    //This code executes if predicate1 is true.
    //If so, we are done.
}
    .
    .
    .
```

```
else
{
    //This only executes if all of the preceding
    //predicates are false.
}
```

Here are some notable features of this construct and some usage rules.

1. You can have as many `else if` clauses as you wish.
2. The `else` clause at the end is optional. We recommend, however, you always use it. You can use it to print out an error message if your program enters some illegal state.
3. In an `if-else if` progression, only one block will ever execute. Once a block executes, control passes beyond the code in the progression.
4. If you have an `else` clause at then end, exactly one block will execute.
5. You may *not* have code between an `if` statement and an `else` statement.
6. You may *not* have code between an `if` statement and an `else if` statement.
7. You may *not* have code between an `else if` statement and an `else` statement.

Now let us return to Dusty's `greetCustomer` function. We will add in an `else if` clause to our code.

```
function getAge()
{
    var age = prompt("Enter your age", "0");
    return age;
}
function showAge(age)
{
    document.getElementById("reply").innerHTML = "You are " + age + " years old." ;
}
function greetCustomer(age)
{
    if(age < 21)
    {
        document.getElementById("greet").innerHTML = "Now get lost, punk!";
        alert("I'm callin' yer Mama and yer gettin' a whoopin'!!");
    }
    else if(age < 65)
    {
```

```
        document.getElementById("greet").innerHTML = "Name yer poizon, baw!";
    }
    else
    {
        document.getElementById("greet").innerHTML = "Shall I cash yer soshal, geezer?";
    }
}
```

JavaScript has one other conditional construct, the *ternary operator*. Its usage is as follows.

```
predicate? exprTrue: exprFalse;
```

The expression `predicate` is any predicate, or boolean-valued expression. If the predicate evaluates to `true`, then the expression `exprTrue` is evaluated; if the predicate evaluates to `false` the expression `exprFalse` is evaluated.

For example the expression

```
grade >= 70? "PASS": "FAIL";
```

evaluates to `"PASS"` if the predicate `grade >= 70` is true and `"FAIL"` otherwise. The ternary operator is just another way to do a very short, simple `if-else`.

Programming Exercises

1. Use the ternary operator to write an absolute value function named `abs`.
2. We have all seen piecewise functions defined in a mathematics class. Here is one
$$f(x) = \begin{cases} 4 - x, & x \leq 4 \\ 0, & 4 < x \leq 6 \\ x - 6, & x \geq 6. \end{cases}$$
Make a JavaScript function with this function's action. Test all of the branches and the border cases.