

CSC 121

Computers and Scientific Thinking

David Reed
Creighton University

Conditional Execution

1

Conditional Execution



so far, all of the code you have written has been *unconditionally executed*

- the browser carried out statements in the same set order

in contrast, many programming tasks require code that reacts differently under varying circumstances or conditions

- e.g., a student's course grade depends upon his/her average
- e.g., an ESP test requires recognizing when a subject guessed right
- e.g., the outcome of a game depends upon die rolls or player moves

conditional execution refers to a program's ability to execute a statement or sequence of statements only if some condition holds true

2

If Statements

in JavaScript, the simplest form of conditional statement is the *if statement*

- one action is taken if some condition is true, but a different action is taken if the condition is not true (called the *else case*)
- the else case is optional

general form of the if statement:

```
if (BOOLEAN_TEST) {  
    STATEMENTS_EXECUTED_IF_TRUE  
}  
else {  
    STATEMENTS_EXECUTED_IF_FALSE  
}
```

3

Braces in If Statements

some people prefer braces on separate lines formatted like this:

```
if (BOOLEAN_TEST)  
{  
    STATEMENTS_EXECUTED_IF_TRUE  
}  
else  
{  
    STATEMENTS_EXECUTED_IF_FALSE  
}
```

either style is acceptable, but be consistent!

- properly aligning the code (with if-else lining up and statements indented) is central in producing code that is easy to read and modify

technically, you can omit the braces if there is only one statement

- however, THIS IS STRONGLY DISCOURAGED!
- can lead to tricky errors if the code is ever modified

4

Boolean Tests

the test that controls an if statement can be any *boolean expression* (i.e., an expression that evaluates to either true or false)

- boolean tests are formed using *relational operators* because they test the relationships between values

Relational Operator	Comparison Defined by the Operator
==	equal to
!=	not equal to
<	less than
<=	less than or equal to
>	greater than
>=	greater than or equal to

NOTE:

== is for comparisons

= is for assignments

the boolean test in an if statement determines the code that will be executed

- if the test is true, then the code inside the subsequent curly braces will execute
- if the test is false, then the code inside the curly braces following the else will execute
- note that if the test is false and there is no else case, the program moves on to the statement directly after the if

5

If Statement Examples

```
if (grade < 60) {
    alert("You failed! Time to hit the books.");
}
```

} code executed if grade < 60

```
if (grade < 60) {
    diff = 60 - grade;
    alert("You failed! If only you could have " +
        "earned " + diff + " more points.");
}
```

} code executed if grade < 60

```
if (grade < 60) {
    diff = 60 - grade;
    alert("You failed! If only you could have " +
        "earned " + diff + " more points.");
}
else {
    alert("Congratulations, you passed.");
}
```

} code executed if grade < 60
} code executed otherwise (grade >= 60)

an if statement is known as a *control statement*, since its purpose is to control the execution of other statements

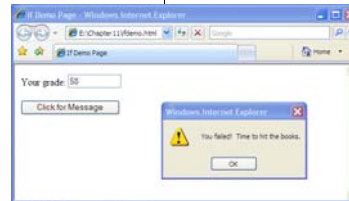
6

Example within a Page

```

1. <html>
2. <!-- ifdemo.html           Dave Reed -->
3. <!-- This program warns a student of a failing grade. -->
4. <!-- ===== -->
5.
6. <head>
7. <title> If Demo Page </title>
8. <script type="text/javascript">
9.   function ShowMessage()
10.    // Assumes: gradeBox contains a grade
11.    // Results: displays a warning in response to a failing grade
12.    {
13.      var grade;
14.
15.      grade = document.getElementById("gradeBox").value;
16.      grade = parseFloat(grade);
17.
18.      if (grade < 60) {
19.        alert("You failed! Time to hit the books.");
20.      }
21.    }
22. </script>
23. </head>
24.
25. <body>
26. <p>
27.   Your grade: <input type="text" id="gradeBox" size="10" value="" />
28. </p>
29. <p>
30.   <input type="button" value="Click for Message" onclick="ShowMessage();" />
31. </p>
32. </body>
33. </html>

```



7

Accessing Text Fields

recall that values entered via text boxes/areas are always returned as strings

```

if (document.getElementById('age').value >= 18) {
  alert("You are old enough to vote.");
}
else {
  alert("Sorry. You are too young to vote.");
}

```

will say that a 2-year old can vote, but a 102-year old can't!
WHY?

if you wish to treat a value obtained from a text box or text area as a number, you must use the `parseFloat` function to convert it

```

age = parseFloat(document.getElementById('age').value);
if (age >= 18) {
  alert("You are old enough to vote.");
}
else {
  alert("Sorry. You are too young to vote.");
}

```

will behave as expected

8

Nested If Statements

programming tasks often require code that responds to more than one condition

- this can be accomplished by nesting one if statement inside of another

example: determining wind-chill

- wind-chill is only defined for temperatures less than or equal to 50 degrees
- the initial if test is to determine if it is a valid temperature to calculate wind-chill
- the nested if statement only executes if the outer test is true

```

if (temperature <= 50) {
  if (windSpeed <= 3) {
    windChill = temperature;
  }
  else {
    windChill = 35.74 + 0.6215*temperature +
      (0.4275*temperature - 35.75)*Math.pow(windSpeed, 0.16);
  }
}
else {
  alert("Wind-chill is defined only if temperature <= 50.");
  windChill = NaN;
}

```

9

Cascading If-else Statements

nested if-else structures are known as *cascading if-else statements* because control cascades down the branches

- the topmost level is evaluated first
- if the test succeeds, then the corresponding statements are executed and control moves to the next statement following the cascading if
- if the test fails, then control cascades down to the next if test
- in general, control cascades down the statement from one test to another until one succeeds or the end of the statement is reached

example: nested if-else structure

```

if (grade < 60) {
  alert("You failed! Time to hit the books.");
}
else {
  if (grade < 90) {
    alert("You passed, but could do better.");
  }
  else {
    alert("Congratulations! You got an A.");
  }
}

```

} executed if grade < 60
 } executed if grade < 90
 } executed if grade >= 60

10

A Cleaner Notation

when it is necessary to handle a large number of alternatives, nested if-else statements can become cumbersome and unwieldy

- multiple levels of indentation and curly braces cause the code to look cluttered make it harder to read/understand

example: nested if statements vs. a more readable else-if

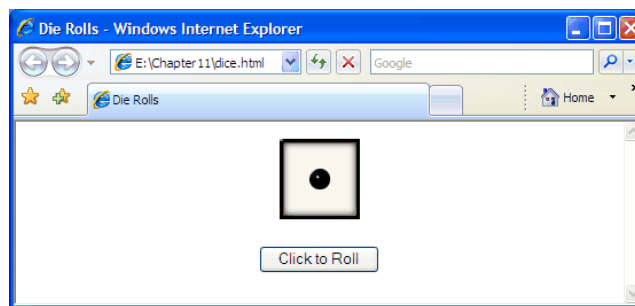
<pre> if (grade < 60) { letterGrade = "F"; } else { if (grade < 70) { letterGrade = "D"; } else { if (grade < 80) { letterGrade = "C"; } else { if (grade < 90) { letterGrade = "B"; } else { letterGrade = "A"; } } } } </pre>	<pre> if (grade < 60) { letterGrade = "F"; } else if (grade < 70) { letterGrade = "D"; } else if (grade < 80) { letterGrade = "C"; } else if (grade < 90) { letterGrade = "B"; } else { letterGrade = "A"; } </pre>
-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------	-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

11

Die Roll Example

consider a Web page that simulates the roll of a single die

- will use an image to display the die
- will use a button to initiate the die roll
- when the user clicks the button, a random die roll is selected and the corresponding image is displayed



12

```

1. <html>
2. <!-- dice.html                                     Dave Reed -->
3. <!-- This page simulates and displays the roll of a die. -->
4. <!--
5.
6. <-head>
7. <title> Die Rolls </title>
8. <script type="text/javascript"
9.     src="http://dave-reed.com/book/random.js">
10. </script>
11.
12. <script type="text/javascript">
13.     function Roll()
14.     // Assumes: die images are in dave-reed.com/book/Images
15.     // Results: displays a randomly selected image of a 6-sided die
16.     {
17.         var roll;
18.
19.         roll = RandomInt(1, 6);
20.
21.         if (roll == 1) {
22.             document.getElementById("die").src =
23.                 "http://dave-reed.com/book/Images/die1.gif";
24.         }
25.         else if (roll == 2) {
26.             document.getElementById("die").src =
27.                 "http://dave-reed.com/book/Images/die2.gif";
28.         }
29.         else if (roll == 3) {
30.             document.getElementById("die").src =
31.                 "http://dave-reed.com/book/Images/die3.gif";
32.         }
33.         else if (roll == 4) {
34.             document.getElementById("die").src =
35.                 "http://dave-reed.com/book/Images/die4.gif";
36.         }
37.         else if (roll == 5) {
38.             document.getElementById("die").src =
39.                 "http://dave-reed.com/book/Images/die5.gif";
40.         }
41.         else {
42.             document.getElementById("die").src =
43.                 "http://dave-reed.com/book/Images/die6.gif";
44.         }
45.     }
46. </script>
47. </head>
48.
49. <-body>
50. <div style="text-align:center">
51. <p>
52. 
54. </p>
55. <p>
56. <input type="button" value="Click to Roll" onclick="Roll();" />
57. </p>
58. </div>
59. </body>
60. </html>

```

Die Roll Page

the RandomInt function from random.js is used to select the random roll

depending on the roll value, the correct image is displayed

since more than two possibilities, a cascading if-else is needed

13

Generalizing Code

note that each case in the cascading if-else follows the same pattern

```

if (roll == 1) {
    document.getElementById('die').src = "http://dave-reed.com/book/Images/die1.gif";
}
else if (roll == 2) {
    document.getElementById('die').src = "http://dave-reed.com/book/Images/die2.gif";
}
else if (roll == 3) {
    document.getElementById('die').src = "http://dave-reed.com/book/Images/die3.gif";
}
else if (roll == 4) {
    document.getElementById('die').src = "http://dave-reed.com/book/Images/die4.gif";
}
else if (roll == 5) {
    document.getElementById('die').src = "http://dave-reed.com/book/Images/die5.gif";
}
else {
    document.getElementById('die').src = "http://dave-reed.com/book/Images/die6.gif";
}

```

this entire cascading if-else structure could be replaced by the following:

```

document.getElementById('die').src =
    "http://dave-reed.com/book/Images/die" + roll + ".gif";

```

14

Counters

in software applications, if statements are often used to count occurrences of conditional or user-initiated events

- e.g., count the number of times dice rolls come up doubles
- e.g., count the number of times the user guesses a number correctly

any variable that is used to record occurrences of an event is known as a *counter*

- initially, the counter is set to zero
- each time the specified action occurs, the counter is incremented
- after a given time period, the value stored in the counter will tell you the number of times the desired event took place

```
document.getElementById("numRolls").value =  
    parseFloat(document.getElementById("numRolls").value) + 1;
```

15

Logical Connectives

sometimes, simple comparisons between two values may not be adequate to express the conditions under which code should execute

JavaScript provides operators for expressing multipart tests

- *logical AND* (&&): represents the conjunction of two things
 - (TEST1 && TEST2) is true if both TEST1 and TEST2 are true

```
if (roll1 == 4 && roll2 == 4) {  
    // code to be executed when double fours are rolled  
}
```

- *logical OR* (||): represents the disjunction of two things
 - (TEST1 || TEST2) is true if either TEST1 or TEST2 are true

```
if (roll1 == 4 || roll2 == 4) {  
    // code to be executed when at least one four is rolled  
}
```

- *logical NOT* (!): represents negation
 - (!TEST1) is true only if TEST1 is false

```
if (!(roll1 == 4 || roll2 == 4)) {  
    // code to be executed when neither roll is a four  
}
```

16