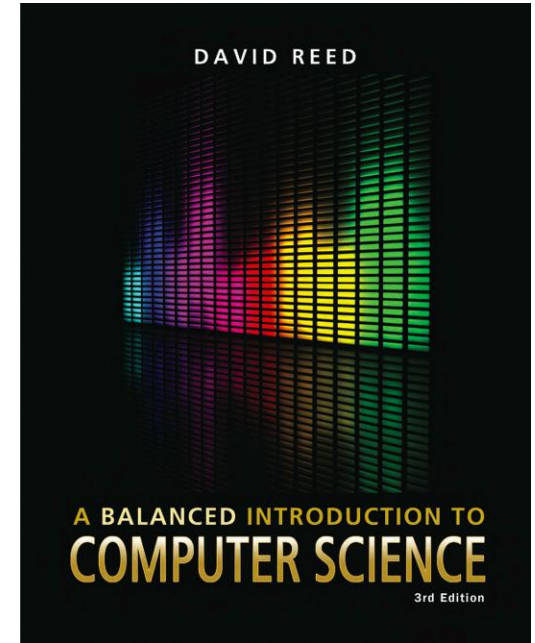


A Balanced Introduction to Computer Science, 3/E

David Reed, Creighton University

**©2011 Pearson Prentice Hall
ISBN 978-0-13-216675-1**



Chapter 11 Conditional Execution

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

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  
}
```

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

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
<code>==</code>	equal to
<code>!=</code>	not equal to
<code><</code>	less than
<code><=</code>	less than or equal to
<code>></code>	greater than
<code>>=</code>	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

If Statement Examples



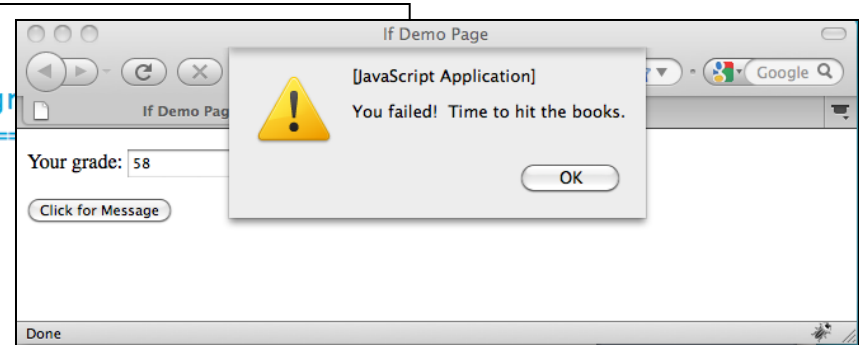
<pre>if (grade < 60) { alert('You failed! Time to hit the books.');</pre>	}	code executed if grade < 60
<pre>} if (grade < 60) { diff = 60 - grade; alert('You failed! If only you could have ' + 'earned ' + diff + ' more points,');</pre>	}	code executed if grade < 60
<pre>} if (grade < 60) { diff = 60 - grade; alert('You failed! If only you could have ' + 'earned ' + diff + ' more points.');</pre>	}	code executed if grade < 60
<pre>} else { alert('Congratulations, you passed,');</pre>	}	code executed otherwise (grade >= 60)
<pre>}</pre>		

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

Example within a Page



```
1. <!doctype html>
2. <!-- ifdemo.html
3. <!-- This program warns a student of a failing grade
4. <!-- =====
5.
6. <html>
7. <head>
8.   <title> If Demo Page </title>
9.   <script type="text/javascript">
10.     function ShowMessage()
11.     // Assumes: gradeBox contains a grade (non-negative number)
12.     // Results: displays a warning in response to a failing grade
13.     {
14.       var grade;
15.
16.       grade = parseFloat(document.getElementById('gradeBox').value);
17.
18.       if (grade < 60) {
19.         alert('You failed! Time to hit the books.');
```



here, the if statement is executed when the button is clicked

- what happens if the text box contains a number ≥ 60 ?

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.');
```

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.');
```

will behave as expected

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: three different grade levels

- failing ($\text{grade} < 60$), acceptable ($60 \leq \text{grade} < 90$), A-level ($\text{grade} \geq 90$)
- the outer if-else distinguishes failing from passing grades
- the nested if-else further separates passing grades into acceptable and A-level

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

}	executed if grade < 60
else {	
if (grade < 90) {	
alert('You passed, but could do better.');	executed if grade < 90
}	
else {	
alert('Congratulations! You got an A.');	executed if grade >= 90
}	
}	executed if grade >= 60

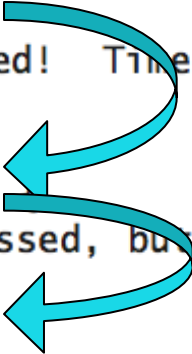
Cascading If-else Statements




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


```
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 >= 90
executed if grade >= 60

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
- can simplify by removing some unnecessary curly braces & aligning each case to the left

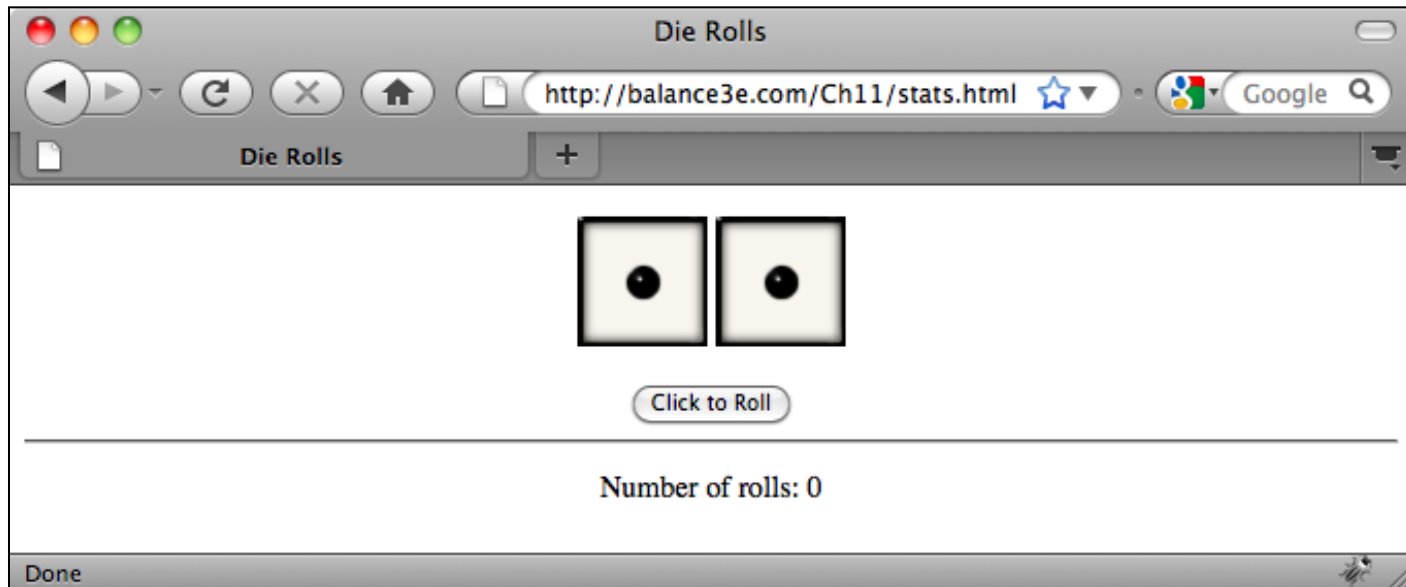
nested if statements	vs. more readable else-if
example: <pre>if (grade < 60) { letterGrade = 'F'; } else { if (grade < 70) { letterGrade = 'D'; } else { if (grade < 80) { letterGrade = 'C'; } else { if (grade < 90) { letterGrade = 'B'; } else { letterGrade = 'A'; } } } }</pre>	<pre>if (grade < 60) { letterGrade = 'F'; } else if (grade < 70) { letterGrade = 'D'; } else if (grade < 80) { letterGrade = 'C'; } else if (grade < 90) { letterGrade = 'B'; } else { letterGrade = 'A'; }</pre>

Dice Stats Example



consider a Web page that simulates the roll of two dice

- will use image to display the dice
 - will use a button to initiate the die rolls
 - will keep track and display the number of rolls
-
- when the user clicks the button, two random die rolls are selected, the corresponding images are displayed, and the number of rolls incremented





```
1. <!doctype html>
2. <!-- stats.html                                Dave Reed -->
3. <!-- This page simulates dice rolls and keeps a roll count. -->
4. <!-- ===== -->
5.
6. <html>
7.   <head>
8.     <title> Die Rolls </title>
9.     <script type="text/javascript" src="http://balance3e.com/random.js">
10.    </script>
11.    <script type="text/javascript">
12.      function RollDice()
13.        // Assumes: die images are in http://balance3e.com/Images
14.        // Results: displays 2 random die rolls & keeps a count in rollSpan
15.        {
16.          var roll1, roll2;
17.
18.          roll1 = RandomInt(1, 6);
19.          roll2 = RandomInt(1, 6);
20.
21.          document.getElementById('die1Img').src =
22.            'http://balance3e.com/Images/die' + roll1 + '.gif';
23.          document.getElementById('die2Img').src =
24.            'http://balance3e.com/Images/die' + roll2 + '.gif';
25.
26.          document.getElementById('rollSpan').innerHTML =
27.            parseFloat(document.getElementById('rollSpan').innerHTML) + 1;
28.        }
29.    </script>
30.  </head>
31.
32.  <body>
33.    <div style="text-align:center">
34.      <p>
35.        
37.        
39.      </p>
40.      <input type="button" value="Click to Roll" onclick="RollDice();">
41.      <hr>
42.      <p>
43.        Number of rolls: <span id="rollSpan">0</span>
44.      </p>
45.    </div>
46.  </body>
47. </html>
```

Stats Page

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

since each die image is stored as die#.gif, can assign each image source with one assignment

the number of rolls appears in a span

- initially 0
- incremented in RollDice each time the button is clicked

Counters



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('rollSpan').innerHTML =  
    parseFloat(document.getElementById('rollSpan').innerHTML)+1;
```

in software applications, counters are often conditional

- 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

conditional counters must be controlled by if statements

- if the desired event occurs, then you increment the counter

```
if (roll1 == roll2) {  
    // CODE TO BE EXECUTED WHEN DOUBLES ARE ROLLED  
}
```

Boolean Expressions



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 DOUBT 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  
}
```



```
1. <!doctype html>
2. <!-- slots.html                                Dave Reed -->
3. <!-- This page simulates a simple slot machine with 3 slots. -->
4. <!-- ===== -->
5.
6. <html>
7.   <head>
8.     <title> Slot Machine </title>
9.     <script type="text/javascript" src="http://balance3e.com/random.js">
10.    </script>
11.     <script type="text/javascript">
12.       function SpinSlots()
13.       // Assumes: slot images are in http://balance3e.com/Images
14.       // Results: displays 3 random slot images
15.       {
16.         var slot1, slot2, slot3;
17.
18.         slot1 = RandomOneOf(['lemon', 'cherry', 'bar', 'donut']);
19.         slot2 = RandomOneOf(['lemon', 'cherry', 'bar', 'donut']);
20.         slot3 = RandomOneOf(['lemon', 'cherry', 'bar', 'donut']);
21.
22.         document.getElementById('slot1Img').src =
23.           'http://balance3e.com/Images/' + slot1 + '.jpg';
24.         document.getElementById('slot2Img').src =
25.           'http://balance3e.com/Images/' + slot2 + '.jpg';
26.         document.getElementById('slot3Img').src =
27.           'http://balance3e.com/Images/' + slot3 + '.jpg';
28.       }
29.     </script>
30.   </head>
31.
32.   <body>
33.     <div style="text-align:center">
34.       <p>
35.         
37.         
39.         
41.       </p>
42.       <input type="button" value="Click to Spin" onclick="SpinSlots();">
43.     </div>
44.   </body>
45. </html>
```

Slot Machine

initially, displays three random slot images at the click of a button

need to display player's credits

- player starts with 20 credits
- each spin costs 1 credit, three matching slots earns 13 credits
- disallow play if no credits
- possibly give a loan when broke

