Programming

FIT100

- Why is programming fun?
  - Third is the fascination of fashioning complex puzzle-like objects of interlocking moving parts and watching them work in subtle cycles, playing out the consequences of principles built in from the beginning. The programmed computer has all the fascination of the pinball machine or the jukebox mechanism, carried to the ultimate.

Source: Frederick P. Brooks, Jr. *The Mythical Man-Month Essays on Software Engineering.*

Schedule Changes

FIT100

Labs 7 and 8:

- Both due *next* week
  - Tuesday, February 19, at 5pm
- Rubric for Lab 8 will be available before lab tomorrow

Announcements

FIT100

- This week's quiz:
  - Chapters 20, 21, and 22 of *Fluency*
  - Review
    - Questions at ends of chapters 20 and 21
    - Answers at back of book

Announcements

FIT100

- This week's quiz topics
  - Variables—global and local
  - Functions—syntax, names, declaring, calling, arguments, parameters,
  - Loops—iteration variables, counters, step increase,
  - Arrays—syntax, declaration, indexes, elements, using with the World-Famous Iteration

Announcements

FIT100

- This week's quiz topics (continued)
  - Opening windows
  - Dates
  - Event handlers—onclick, onchange, onsubmit, onload, etc.
  - Concatenation

Announcements

FIT100

- Project and lab turn-ins
  - Catalyst Collect It shows date and time
  - Your html files show date and time
  - Do NOT keep working on your html files after the due date or they will be marked late!

FIT100

Announcements

- Project turn-ins
  - 1-1-1 Rule (see Syllabus online)
    - *One* project part, such as Project 1A, can be *one* day late one time during the quarter
- If you have used up your 1-1-1 rule for the quarter, turn in as much as you have finished so you get at least partial credit—rather than no credit!

FIT100

Announcements

- At end of quarter,
  - We will drop your
    - Lowest quiz score
    - Lowest lab score

FIT100

Thinking Big: Programming Functions

A function is a package for an algorithm; once written, it can be used over and over.

FIT100

Anatomy of a Function

- Functions are packages for algorithms
- 3 parts
  - Name
  - Parameters
  - Definition
- These parts are the *function declaration*

20-10

FIT100

Pick a Name

- Name is the identifier for the function
  - Commonly used to describe what the function does
- Function declaration form:

function <name> (<*parameter list*>)
{
  <*statement list*>
}

20-11

FIT100

Parameters

- Parameters are the values the function will compute on, the input values
- They are given names
- Listed parameters are separated by commas
- Parameter names follow usual rules for identifiers

```
function convertC2F ( tempInC )
{
   <statement list>
```

20-12

Definition

FIT100

- Definition is the algorithm written in a programming language
- To say what the answer/result is, JavaScript uses the statement: **return** <expression>

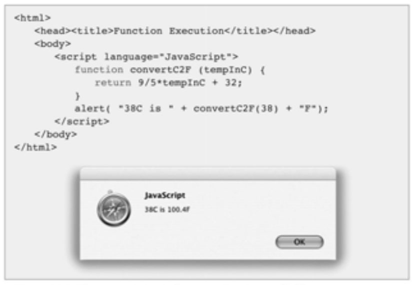
```
function convertC2F ( tempInC )
{
   return 9.0 / 5.0 * tempInC + 32;
}
```

- "Calling" a function is to run or execute it
  - Write the function's name, put the input values (*arguments*) in the parentheses

```
convertC2F( 38 )
```

20-13

FIT100



```
<html>
   <head><title>Function Execution</title></head>
   <body>
      <script language="JavaScript">
         function convertC2F (tempInC) {
            return 9/5*tempInC + 32;
         }
         alert( "38C is " + convertC2F(38) + "F");
      </script>
   </body>
</html>
```

Figure 20.1. The **convertC2F()** *function in JavaScript called from an* **alert()**.

20-14

Declaration versus Call

FIT100

- A function's declaration is different from its call (use)
- Functions are declared once
- Functions can be called as many times as their answers are needed

20-15

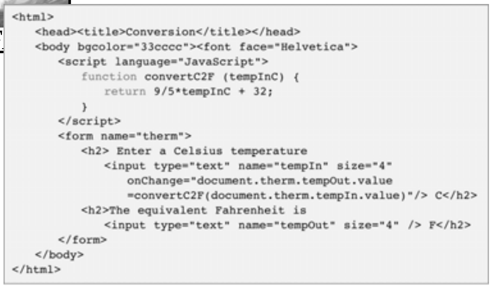
Forms and Functions

FIT100

- Construct a web page in which to run a function
- Recall <form> and <input /> tags and event handlers in HTML
  - Event handlers usually implemented as functions
- Using an input window, the value in that window can be used as an argument to a function

20-16

FIT100



```
<html>
   <head><title>Conversion</title></head>
   <body bgcolor="33cccc"><font face="Helvetica">
      <script language="JavaScript">
         function convertC2F (tempInC) {
            return 9/5*tempInC + 32;
         }
      </script>
      <form name="therm">
         <h2> Enter a Celsius temperature
            <input type="text" name="tempIn" size="4"
               onChange="document.therm.tempOut.value
               =convertC2F(document.therm.tempIn.value)"/> C</h2>
         <h2>The equivalent Fahrenheit is
            <input type="text" name="tempOut" size="4" /> F</h2>
      </form>
   </body>
</html>
```

Figure 20.2 The HTML/JavaScript source for the Conversion application.

20-17

Calling to Customize a Page

FIT100

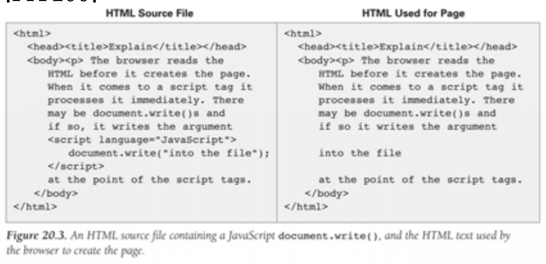
- Three ways to get the result of a function call to print on the monitor
  1) Before the page is created
     For example, with the alert() call (Fig. 20.1)
  2) Interactively after the page is displayed
     For example, the Conversion application (Fig. 20.2)
  3) While the page is being loaded
     For example, document.write() built-in function
- Calling functions while the browser is creating the page allows us to customize pages on-the-fly

20-18

Calling to Customize a Page

- How a browser builds a page:
  - Reads through HTML file, figuring out all tags and preparing to build page
  - Removes JavaScript tags and all text between them, and does whatever the JavaScript tells it to do
    - It could tell the browser to put some text back in the file, as in *document.write()*

20-19



HTML Source File

```
<html>
  <head><title>Explain</title></head>
  <body><p> The browser reads the
     HTML before it creates the page.
     When it comes to a script tag it
     processes it immediately. There
     may be document.write()s and
     if so, it writes the argument
     <script language="JavaScript">
        document.write("into the file");
     </script>
     at the point of the script tags.
   </body>
</html>
```

HTML Used for Page

```
<html>
  <head><title>Explain</title></head>
  <body><p> The browser reads the
     HTML before it creates the page.
     When it comes to a script tag it
     processes it immediately. There
     may be document.write()s and
     if so it writes the argument

     into the file

     at the point of the script tags.
   </body>
</html>
```

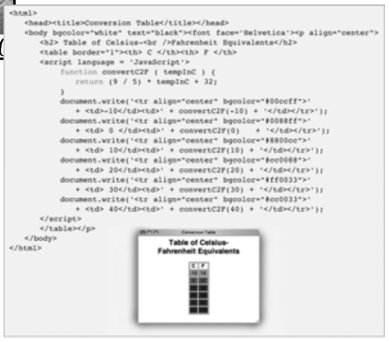
Figure 20.3. *An HTML source file containing a JavaScript* `document.write()`*, and the HTML text used by the browser to create the page.*

20-20

Calling to Customize a Page

- Suppose we want a table of temperature conversions for a web page with a column for Celsius and a column for Fahrenheit
- Put *document.write()* within the <script> </script> tags to create the rows of the table
- Put Celsius values in first column cells, second column cells can call conversion function

20-21



```
<html>
  <head><title>Conversion Table</title></head>
  <body bgcolor="white" text="black"><font face='Helvetica'><p align="center">
     <h2> Table of Celsius-<br />Fahrenheit Equivalents</h2>
     <table border="1"><th> C </th><th> F </th>
     <script language = 'JavaScript'>
         function convertC2F ( tempInC ) {
            return (9 / 5) * tempInC + 32;
         }
         document.write('<tr align="center" bgcolor="#00ccff">'
            + <td>-10</td><td>' + convertC2F(-10) + '</td></tr>');
         document.write('<tr align="center" bgcolor="#0088ff">'
            + <td> 0 </td><td>' + convertC2F(0)    + '</td></tr>');
         document.write('<tr align="center" bgcolor="#8800cc">'
            + <td> 10</td><td>' + convertC2F(10) + '</td></tr>');
         document.write('<tr align="center" bgcolor="#cc0088">'
            + <td> 20</td><td>' + convertC2F(20) + '</td></tr>');
         document.write('<tr align="center" bgcolor="#ff0033">'
            + <td> 30</td><td>' + convertC2F(30) + '</td></tr>');
         document.write('<tr align="center" bgcolor="#cc0033">'
            + <td> 40</td><td>' + convertC2F(40) + '</td></tr>');
     </script>
     </table></p>
  </body>
</html>
```

Figure 20.4 Source text and image for the Conversion Table computation.

20-22

Writing Functions, Using Functions

- Flipping Electronic Coins
  - A coin flip is an unpredictable event whose two outcomes are "equally probable"
  - Computers can generate pseudo-random numbers
    - An algorithm that produces a sequence of numbers that passes the statistical tests for randomness
    - We can just call them random numbers

20-23

Flipping Electronic Coins

- *Math.random()* is JavaScript's built-in function for generating random numbers
  - Each time it is called, it generates a random number between 0 (inclusive) and 1 (exclusive)
- A function to flip electronic coins:

```
function coinFlip() {
   return Math.round( Math.random() );
}
```

20-24

Flipping Electronic Coins (cont’d)

FIT100

- coinFlip() returns with equal probability a 0 or a 1
- Next improvement is to return the text Heads or Tails rather than numbers

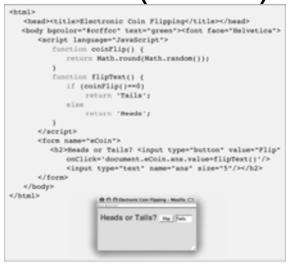
```
function flipText() {
   if ( coinFlip() == 0 )
      return 'Tails';
    else
      return 'Heads';
}
```

20-25

Flipping Electronic Coins (cont’d)

FIT100

- Even more useful to give outcome in response to pressing a button on a web page



```
<html>
   <head><title>Electronic Coin Flipping</title></head>
   <body bgcolor="#ccffcc" text="green"><font face="Helvetica">
      <script language="JavaScript">
         function coinFlip() {
            return Math.round(Math.random());
         }
         function flipText() {
            if (coinFlip()==0)
               return 'Tails';
            else
               return 'Heads';
         }
      </script>
      <form name="eCoin">
         <h2>Heads or Tails? <input type="button" value="Flip"
            onClick='document.eCoin.ans.value=flipText()'/>
            <input type="text" name="ans" size="5"/></h2>
      </form>
   </body>
</html>
```

Figure 20.5 The JavaScript and image for the Electronic Coin-Flipping page.

20-26

The Body Mass Index Computation

FIT100

- BMI is a standard measure of weight in proportion to height
- Formula (in metric units):
  - Index = weight/height^2

  Two parameters for this function, weight and height

```
function bmiM ( weightKg, heightCm ) {   // Compute BMI in metric
   var heightM = heightCm/100;             // Change cm to meters
   return weightKg / (heightM * heightM);
}
```

20-27

The Body Mass Index Computation (cont'd)

FIT100

- Formula (in English units):
- Index = 4.89 weight / height^2
- Function:

```
function bmiE ( weightLbs, heightIn ) {     // Compute BMI in English
   var heightFt = heightIn / 12;                // Change inches to feet
   return 4.89 * weightLbs / (heightFt * heightFt);
}
```

20-28

The Body Mass Index Computation (cont'd)

FIT100

- Function that could calculate BMI in type of units specified by user would need 3 inputs (kind of unit, weight, height)

```
function BMI ( units, weight, height ) {
   if (units == ‘E’)
      return bmiE (weight, height);   // Answer in English
   else
      return bmiM (weight, height); // Answer in Metric
}
```

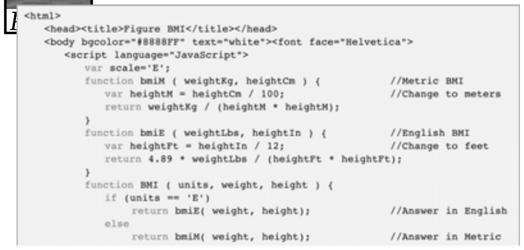
20-29

The Body Mass Index Computation (cont'd)

FIT100

- To put this function in a web page, we add radio buttons to select type of units
- Two new features of radio buttons:
  - All related buttons share same name (clicking one on turns the other off)
  - Can be preset using *checked='true'*
- Add event handlers for the radio buttons

20-30

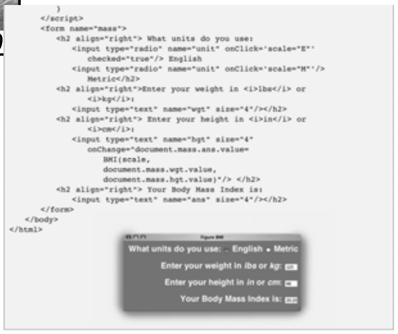


```
<html>
   <head><title>Figure BMI</title></head>
   <body bgcolor="#8888FF" text="white"><font face="Helvetica">
      <script language="JavaScript">
         var scale='E';
         function bmiM ( weightKg, heightCm ) {          //Metric BMI
            var heightM = heightCm / 100;                //Change to meters
            return weightKg / (heightM * heightM);
         }
         function bmiE ( weightLbs, heightIn ) {         //English BMI
            var heightFt = heightIn / 12;                //Change to feet
            return 4.89 * weightLbs / (heightFt * heightFt);
         }
         function BMI ( units, weight, height ) {
            if (units == 'E')
               return bmiE( weight, height);             //Answer in English
            else
               return bmiM( weight, height);             //Answer in Metric
```

Figure 20.6 The image and source for the Figure BMI page.

20-31

FIT10



```
         }
      </script>
      <form name="mass">
         <h2 align="right"> What units do you use:
            <input type="radio" name="unit" onClick='scale="E"'
               checked="true"/> English
            <input type="radio" name="unit" onClick='scale="M"'/>
               Metric</h2>
         <h2 align="right">Enter your weight in <i>lbs</i> or
               <i>kg</i>:
            <input type="text" name="wgt" size="4"/></h2>
         <h2 align="right"> Enter your height in <i>in</i> or
               <i>cm</i>:
            <input type="text" name="hgt" size="4"
               onChange="document.mass.ans.value=
                  BMI(scale,
                  document.mass.wgt.value,
                  document.mass.hgt.value)"/> </h2>
         <h2 align="right"> Your Body Mass Index is:
            <input type="text" name="ans" size="4"/></h2>
      </form>
   </body>
</html>
```

Figure BMI
What units do you use: English Metric
Enter your weight in lbs or kg:
Enter your height in in or cm:
Your Body Mass Index is:

Figure 20.6 The image and source for the Figure BMI page.

20-32

FIT100

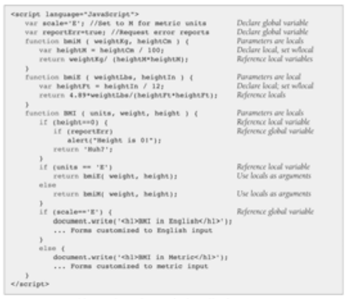
Scoping: When to Use Names

- Scope of a name defines how "far" from its declarations it can be used
- General rule for scoping:
  - Variable names declared in a function can be used only within that function (they are *local to the function*)
    - Parameters are considered local variables
  - Variable names declared outside any function can be used throughout the program (*global to the function*)

20-33

FIT100

An Annotated Example



```
<script language="JavaScript">
   var scale='E'; //Set to M for metric units          Declare global variable
   var reportErr=true; //Request error reports         Declare global variable
   function bmiM ( weightKg, heightCm ) {               Parameters are locals
      var heightM = heightCm / 100;                     Declare local, set w/local
      return weightKg/ (heightM*heightM);               Reference local variables
   }
   function bmiE ( weightLbs, heightIn ) {              Parameters are local
      var heightFt = heightIn / 12;                     Declare local; set w/local
      return 4.89*weightLbs/(heightFt*heightFt);        Reference locals
   }
   function BMI ( units, weight, height ) {             Parameters are locals
      if (height==0) {                                  Reference local variable
         if (reportErr)                                 Reference global variable
            alert("Height is 0!");
         return 'Huh?';
      }
      if (units == 'E')                                 Reference local variable
         return bmiE( weight, height);                  Use locals as arguments
      else
         return bmiM( weight, height);                  Use locals as arguments
      }
      if (scale=='E') {                                 Reference global variable
         document.write('<h1>BMI in English</h1>');
         ... Forms customized to English input
      }
      else {
         document.write('<h1>BMI in Metric</h1>');
         ... Forms customized to metric input
   }
</script>
```

Figure 20.7. Annotated functions showing the scope of each variable reference.

20-34

FIT100

Scoping

- *Local variables* come into existence when a function begins, and when it ends, they vanish
- *Global variables* are around all the time
- If information must be saved from one function call to the next, it must be in a *global variable*

20-35

FIT100

Global/Local Scope Interaction

- Where a global variable and a local variable have the same name:

```
var y=0;
…
function tricky (x) {
   var y;
     y = x;
   …
}
```

20-36

FIT100

Global/Local Scope Interaction (cont'd)

- y is globally declared and can be referenced anywhere
- y is also declared as a local variable in the tricky() function
- They are two different variables
- Which y is assigned the parameter x?
  - The local y, because it is declared in the function's scope, making it the "closest" declaration and hiding the global y

20-37

FIT100

The Memory Bank Web Page

- Create a web page for remembering useful computations and storing them in an interactive form
- Practice programming with functions

20-38

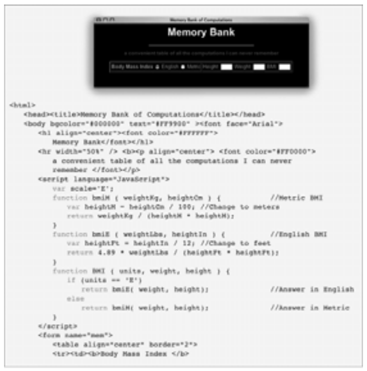
FIT100

Plan the Memory Bank Web Page

- Each table row presents a computation
- Each text box except the last is an input to the computation
- The last text box is for the output
- Start with the row from the BMI computation page

20-39

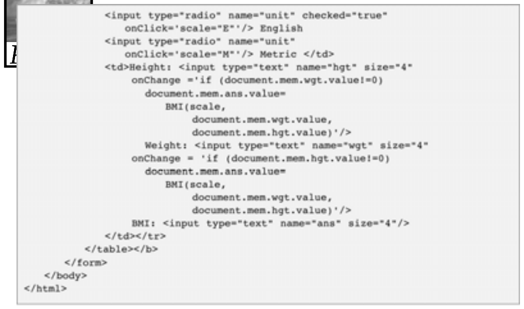
FIT100


Memory Bank of Computations
Memory Bank
a convenient table of all the computations I can never remember
Body Mass Index
English
Metric
Height
Weight
BMI

```
<html>
  <head><title>Memory Bank of Computations</title></head>
  <body bgcolor="#000000" text="#FF9900" ><font face="Arial">
    <h1 align="center"><font color="#FFFFFF">
      Memory Bank</font></h1>
    <hr width="50%" /> <b><p align="center"> <font color="#FF0000">
      a convenient table of all the computations I can never
      remember </font></p>
    <script language="JavaScript">
      var scale='E';
      function bmiM ( weightKg, heightCm ) {            //Metric BMI
        var heightM = heightCm / 100; //Change to meters
        return weightKg / (heightM * heightM);
      }
      function bmiE ( weightLbs, heightIn ) {           //English BMI
        var heightFt = heightIn / 12; //Change to feet
        return 4.89 * weightLbs / (heightFt * heightFt);
      }
      function BMI ( units, weight, height ) {
        if (units == 'E')
          return bmiE( weight, height);                 //Answer in English
        else
          return bmiM( weight, height);                 //Answer in Metric
      }
    </script>
    <form name="mem">
      <table align="center" border="2">
      <tr><td><b>Body Mass Index </b>
```

Figure 20.8 The initial Memory Bank interface and its source. This program can be downloaded from www.aw.com/snyder/ (continues next page)

20-40



```
          <input type="radio" name="unit" checked="true"
            onClick='scale="E"'/> English
          <input type="radio" name="unit"
            onClick='scale="M"'/> Metric </td>
          <td>Height: <input type="text" name="hgt" size="4"
              onChange ='if (document.mem.wgt.value!=0)
                document.mem.ans.value=
                   BMI(scale,
                       document.mem.wgt.value,
                       document.mem.hgt.value)'/>
                Weight: <input type="text" name="wgt" size="4"
              onChange = 'if (document.mem.hgt.value!=0)
                document.mem.ans.value=
                   BMI(scale,
                       document.mem.wgt.value,
                       document.mem.hgt.value)'/>
              BMI: <input type="text" name="ans" size="4"/>
          </td></tr>
        </table></b>
      </form>
    </body>
</html>
```

Figure 20.8 (*continued*) The initial Memory Bank interface and its source. This program can be downloaded from www.aw.com/snyder/

20-41

FIT100

Random Additions

- Add the row from the coin-flipping page
- Program event handler to keep track of the number of heads and tails flipped
- Use global variables so they keep their values across function calls

20-42

Revising Random Choice Function

- Write a function that chooses random whole numbers in a range from 0 to n, not including n

```
function randNum ( range ) {
   return Math.floor( range * Math.random() );
}
```

- For coin-flipping, the range will be 2: 0 and 1

randNum(2)

20-43

The Coin-Flipping Row

- Flip button and textboxes for current flip Outcome, Heads total, and Tails total
- Use global variables to keep track of the number of heads and tails flipped
  - Increment appropriate variable with each flip
- Update/display current flip outcome and total number of heads or total number of tails with each flip

20-44

The "I'm Thinking of a Number" Row

- Guessing game – choose a number from 1 to *n*
- Use randNum() function, but shift the range by 1
  - randNum(n)+1;
- This table row is similar to coin-flipping row, but has a text box to set the upper end of the range
  - Declare global variable (topEnd) to say what the limit of the range is
  - When the user clicks button, the randNum() function is called with topEnd as the argument, and the result is incremented to shift its range. The result is displayed.

20-45

Improving the Memory Bank Web Page

- Needs to be fancier and include more features
- Program the memory bank to splash new pages onto the screen
- Unlike a link, this allows both pages to display at the same time

20-46

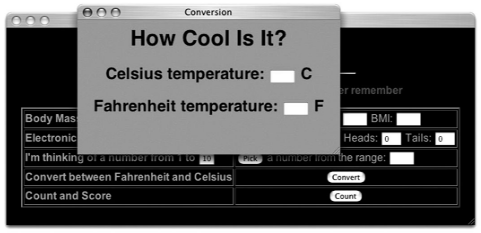
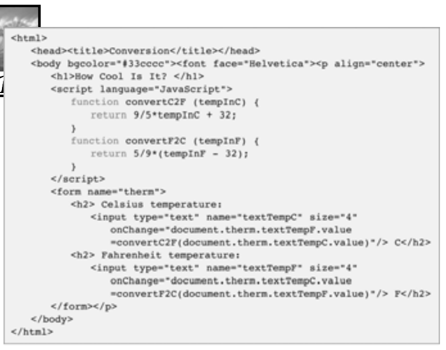

Conversion
How Cool Is It?
Celsius temperature: C
Fahrenheit temperature: F
er remember
Body Mas
BMI:
Electronic
Heads: 0
Tails: 0
I'm thinking of a number from 1 to 10
Pick a number from the range:
Convert between Fahrenheit and Celsius
Convert
Count and Score
Count

Figure 20.9. *The revised Memory Bank page and the Conversion page that displays when* **Convert** *is clicked.*

20-47



```
<html>
   <head><title>Conversion</title></head>
   <body bgcolor="#33cccc"><font face="Helvetica"><p align="center">
      <h1>How Cool Is It? </h1>
      <script language="JavaScript">
         function convertC2F (tempInC) {
            return 9/5*tempInC + 32;
         }
         function convertF2C (tempInF) {
            return 5/9*(tempInF - 32);
         }
      </script>
      <form name="therm">
         <h2> Celsius temperature:
            <input type="text" name="textTempC" size="4"
               onChange="document.therm.textTempF.value
               =convertC2F(document.therm.textTempC.value)"/> C</h2>
         <h2> Fahrenheit temperature:
            <input type="text" name="textTempF" size="4"
               onChange="document.therm.textTempC.value
               =convertF2C(document.therm.textTempF.value)"/> F</h2>
      </form></p>
   </body>
</html>
```

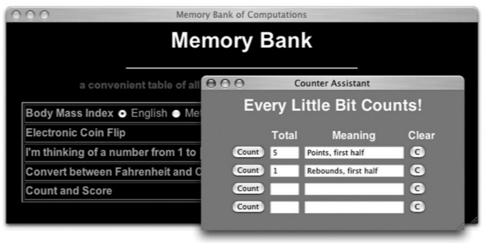
Figure 20.10 The file `temperature.html` for the new Conversion page in Figure 20.9.

20-48

A Counting Page

- To keep track of counts of things
- Counter Assistant application:
  - **Count** button increments **Total** field
  - **Meaning** field can be filled in with any text to remind us what the counter is
  - **C** clears all the fields for that row

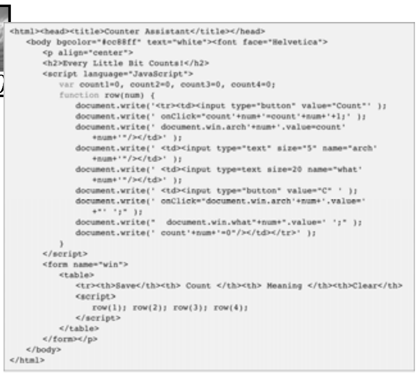
20-49



Memory Bank of Computations
Memory Bank
a convenient table of all
Body Mass Index
English
Me
Electronic Coin Flip
I'm thinking of a number from 1 to
Convert between Fahrenheit and C
Count and Score
Counter Assistant
Every Little Bit Counts!
Count
C

	Total	Meaning	Clear
Count	5	Points, first half	C
Count	1	Rebounds, first half	C
Count			C
Count			C

20-50



```
<html><head><title>Counter Assistant</title></head>
   <body bgcolor="#cc88ff" text="white"><font face="Helvetica">
      <p align="center">
      <h2>Every Little Bit Counts!</h2>
      <script language="JavaScript">
         var count1=0, count2=0, count3=0, count4=0;
         function row(num) {
            document.write('<tr><td><input type="button" value="Count"' );
            document.write(' onClick="count'+num+'=count'+num+'+1;' );
            document.write(' document.win.arch'+num+'.value=count'
               +num+'"/></td>' );
            document.write(' <td><input type="text" size="5" name="arch'
               +num+'"/></td>' );
            document.write(' <td><input type=text size=20 name="what'
               +num+'"/></td>' );
            document.write(' <td><input type="button" value="C" ' );
            document.write(' onClick="document.win.arch'+num+'.value='
               +"' ';" );
            document.write("  document.win.what"+num+".value=' ';" );
            document.write(' count'+num+'=0"/></td></tr>' );
         }
      </script>
      <form name="win">
         <table>
            <tr><th>Save</th><th> Count </th><th> Meaning </th><th>Clear</th>
            <script>
               row(1); row(2); row(3); row(4);
            </script>
         </table>
      </form></p>
   </body>
</html>
```

Figure 20.11 The Counter Assistant application, saved in a file counter.html.

20-51

Recap: Two Reasons to Write Functions

- Packaging algorithms into functions
- Reuse
  - Building blocks of future programming
  - Make them as general as possible
- Complexity management
  - Help us keep our sanity while we're solving problems

20-52

Add Final Touches to Memory Bank

- Add a date
  - JavaScript Date().toString()
    - References the date object, which contains the current date and time in numeric form, and converts to a printable form
- Add web links
  - Add any useful links (online dictionary, etc) in their own column or in a row at the bottom of the table

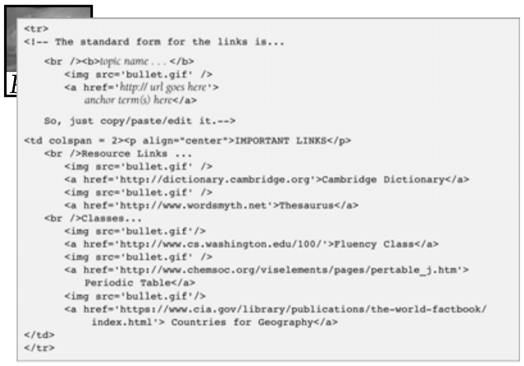
20-53

Assess the Web Page Design

- Table data which spans two columns using colspan=2 attribute in <td> tag
- Links are grouped by topic
- Red bullet is used to separate entries
- Link area has a neat structure; adding new links is easy

20-54

F

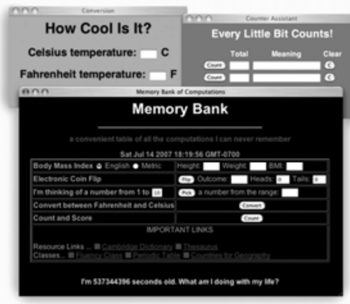


```
<tr>
<!-- The standard form for the links is...

   <br /><b>topic name . . . </b>
      <img src='bullet.gif' />
      <a href='http:// url goes here'>
         anchor term(s) here</a>

   So, just copy/paste/edit it.-->

<td colspan = 2><p align="center">IMPORTANT LINKS</p>
   <br />Resource Links ...
      <img src='bullet.gif' />
      <a href='http://dictionary.cambridge.org'>Cambridge Dictionary</a>
      <img src='bullet.gif' />
      <a href='http://www.wordsmyth.net'>Thesaurus</a>
   <br />Classes...
      <img src='bullet.gif'/>
      <a href='http://www.cs.washington.edu/100/'>Fluency Class</a>
      <img src='bullet.gif' />
      <a href='http://www.chemsoc.org/viselements/pages/pertable_j.htm'>
         Periodic Table</a>
      <img src='bullet.gif'/>
      <a href='https://www.cia.gov/library/publications/the-world-factbook/
          index.html'> Countries for Geography</a>
</td>
</tr>
```

Figure 20.12 HTML for the link area of the Memory Bank Web page.

20-55

FIT10



Conversion

How Cool Is It?

Celsius temperature: C

Fahrenheit temperature: F

Counter Assistant

Every Little Bit Counts!

Total Meaning Clear

Memory Bank of Computations

Memory Bank

a convenient table of all the computations I can never remember

Sat Jul 14 2007 18:19:56 GMT-0700

Body Mass Index ○ English ● Metric	Height: Weight: BMI:
Electronic Coin Flip	Flip Outcome: Heads: 0 Tails: 0
I'm thinking of a number from 1 to 10	Pick a number from the range:
Convert between Fahrenheit and Celsius	Convert
Count and Score	Count

IMPORTANT LINKS

Resource Links ... ■ Cambridge Dictionary ■ Thesaurus
Classes... ■ Fluency Class ■ Periodic Table ■ Countries for Geography

I'm 537344396 seconds old. What am I doing with my life?

Figure 20.13 Final version of the Memory Bank, Conversion, and Counter Assistant pages.

20-56