

End papers...

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*.
FIT100, D.A. Clements, UW iSchool

Announcements

- QuickClicks
 - None today
 - Will resume on Wednesday
- Tutoring on Wednesdays 6:30-8pm
 - Moved to OUGL 102

FIT100, D.A. Clements, UW iSchool

Fluency with Information Technology

Functions and Methods

Good Things Come in Small Packages

D.A. Clements

FIT100, D.A. Clements, UW iSchool

Functions

- A function is a package for an algorithm; once written, it can be used over and over.
- Professional developers have their own script libraries they bring to each job they work on.

FIT100, D.A. Clements, UW iSchool

Anatomy of a Function

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

FIT100, D.A. Clements, UW iSchool

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>
}
```

FIT100, D.A. Clements, UW iSchool

Parameters

- Parameters are the input values the function will compute on
- Parameters are given names
- If more than one, they are separated by commas
- Parameter names follow usual rules for identifiers

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

FIT100, D.A. Clements, UW ISchool

7

Definition

- 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 );
```

FIT100, D.A. Clements, UW ISchool

8



```
<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().

FIT100, D.A. Clements, UW ISchool

9

Declaration vs. Call

- 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

Declare once.
Call many times.

FIT100, D.A. Clements, UW ISchool

10

Forms and Functions

- Construct a web page in which to run a function
- Event handlers usually call functions
- The value in an input window, or text box, can be used as an *argument* to a function

FIT100, D.A. Clements, UW ISchool

11



```
<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.

FIT100, D.A. Clements, UW ISchool

12

Calling to Customize a Page

- 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 (slide 7)
 - 2) Interactively after the page is displayed
For example, the Conversion application (slide 10)
 - 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

FIT100, D.A. Clements, UW iSchool 13

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()

FIT100, D.A. Clements, UW iSchool 14

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.

FIT100, D.A. Clements, UW iSchool 15

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

FIT100, D.A. Clements, UW iSchool 16

```
<html>
<head><title>Conversion Table</title></head>
<body bgcolor="white" text="black"><p align="center">
<b2> Table of Celsius-<br />Fahrenheit Equivalents</b2>
<table border="1"><thead>
<tr>
<th align="center">C </th><th align="center">F </th>
</tr>
</thead>
<tbody>
<tr>
<td align="center">0 </td><td align="center">convertC2F(0) </td>
</tr>
<tr>
<td align="center">10 </td><td align="center">convertC2F(10) </td>
</tr>
<tr>
<td align="center">20 </td><td align="center">convertC2F(20) </td>
</tr>
<tr>
<td align="center">30 </td><td align="center">convertC2F(30) </td>
</tr>
<tr>
<td align="center">40 </td><td align="center">convertC2F(40) </td>
</tr>
</tbody>
</table>
</body>
</html>
```

Figure 20.4. Source text and image for the Conversion Table, conversion.

FIT100, D.A. Clements, UW iSchool 17

Writing and 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

FIT100, D.A. Clements, UW iSchool 18

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() );
}
```

FIT100, D.A. Clements, UW iSchool 19

Flipping Electronic Coins (cont'd)

- `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';
}
```

FIT100, D.A. Clements, UW iSchool 20

Flipping Electronic Coins (cont'd)

- 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><script><!--Header--><script><script language="JavaScript">
function coinFlip() {
  return Math.round(Math.random());
}
function flipText() {
  if (coinFlip()==0)
    return "Tails";
  else
    return "Heads";
}
</script>
<form name="Coin">
<td>Heads or Tails? <input type="button" value="Flip"
onClick="document.write(coinFlipText())">
</td></tr>
</form>
</body>
</html>
```

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

FIT100, D.A. Clements, UW iSchool 21

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*)

FIT100, D.A. Clements, UW iSchool 22

An Annotated Example

```
<script language="JavaScript">
var scale="M"; //Set to M for metric units          Declare global variable
var reportError=true; //Report error reports       Declare global variable
function BMI ( weight, height ) {                 Parameters are local
  var heightM = heightCm / 100;                  Declare local, set w/global
  return weight/( heightM*heightM);               Reference local variables
}
function BMI2 ( weight, height ) {                 Parameters are local
  var heightM = heightC / 12;                    Declare local, set w/global
  return 4.89*weight/(height*height);             Reference local
}
function BMI3 ( units, weight, height ) {          Parameters are local
  if (height==0) {                                Reference local variable
    if (reportError)                              Reference global variable
      alert("Height is 0!");
    return "NaN";
  }
  if (units == "M") {                             Reference local variable
    return BMI2( weight, height);                 Use locals as arguments
  }
  else {                                           Use locals as arguments
    return BMI2( weight, height);
  }
}
if (scale=="M") {                                 Reference global variable
  document.write("<M>BMI in English</M>");
  ... Forms customized to English input
}
else {
  document.write("<M>BMI in Metric</M>");
  ... Forms customized to metric input
}
</script>
```

Figure 20.7 FIT100, D.A. Clements, UW iSchool's variable reference.

23

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*

FIT100, D.A. Clements, UW iSchool 24

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;
  ...
}

```

FIT100, D.A. Clements, UW iSchool 25

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

FIT100, D.A. Clements, UW iSchool 26

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

FIT100, D.A. Clements, UW iSchool 27

Return

- A means of returning the result of the function:

```

var y=0;
function tricky (x) {
  y = x * 2;
  return;
}
...
results = tricky(16);

```

FIT100, D.A. Clements, UW iSchool 28

Built-in JavaScript functions

METHODS

FIT100, D.A. Clements, UW iSchool 29

Methods

- Methods are built-in JavaScript for commonly used code:
 - window.open()
 - alert();
 - prompt();
 - confirm();
 - document.write();

FIT100, D.A. Clements, UW iSchool 30



Methods—Write your own

- You can even write your own methods and “attach” them to objects