College of Information Studies

University of Maryland Hornbake Library Building College Park, MD 20742-4345

Functions and Data Structures

Week 4

INFM 603

Programming in Four Parts

- Structured Programming
- Data Structures
- Modular Programming
- Object-Oriented Programming

Arrays

- A set of elements
  - For example, the number of days in each month
- Each element is assigned an index
  - A number used to refer to that element
    - For example, x[4] is the fifth element (count from zero!)
  - Arrays and iteration work naturally together
- "Constructor" allocates space
  - var myArray = new Array(5); // all unitialized
  - var myArray = [42, 17, , 22, 1]; // partially initialized

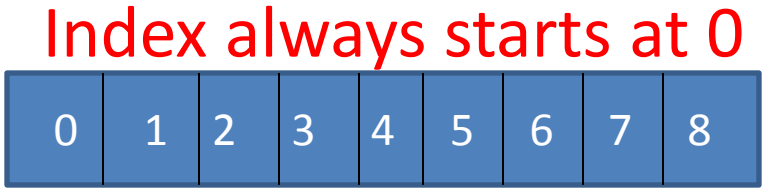
Indexing Values

expenses [0]= “25”;

expenses[1] = “30”;

…

expenses[365] =“100”;



OR

var expenses = [“25”,”30”,…”100”];

var expenses = new Array (“25”,”30”,…”100”);

Array Example

```
// allocate five-element Array (indexed 0..4)
var n = new Array(5);

// assign values to each element of Array n
for (var i=0; i<n.length; i++) {
  n[i] = i;
}

// output index and value of each element
for (var i=0; i<n.length; i++) {
  document.writeln(i + ": " + n[i]);
}
```

Data Structures

- Constant
  - Names given to unchanging values (for readability)
- Scalar
  - Single-valued item (int, float, boolean)
- Object
  - Multi-valued item, mixed data types [+methods]
- Array
  - Integer-indexed set of objects (usually of one type)
- Associative array (“hash table”)
  - Object-index set of objects (usually of one type)

Associative Arrays in JavaScript

```
var myArray = new Array();
myArray[‘Monday'] = 1;
myArray[‘Tuesday'] = 2;
myArray[‘Wednesday'] = 3;

// show the values stored
for (var i in myArray) { // skips unitialized
  alert('key is: ' + i +
        ', value is: ' + myArray[i]);
}
```

Common Uses of Arrays

- Iterative computation
- Queue (FIFO)
- Stack (LIFO)

Hands On

- Write a JavaScript program that asks you for three numbers and outputs the last one

Use arrays instead of three different variables

```
<!DOCTYPE html>
<html>
<body>


<script>
var numbers = new Array();
var c,input;

for (c=0;c<3;c++) {
  input = prompt(“Type a number");
  numbers[c] = input;
}
document.writeln("The last number is" + numbers[2]);

</script>

</body>
</html>
```

Array Methods

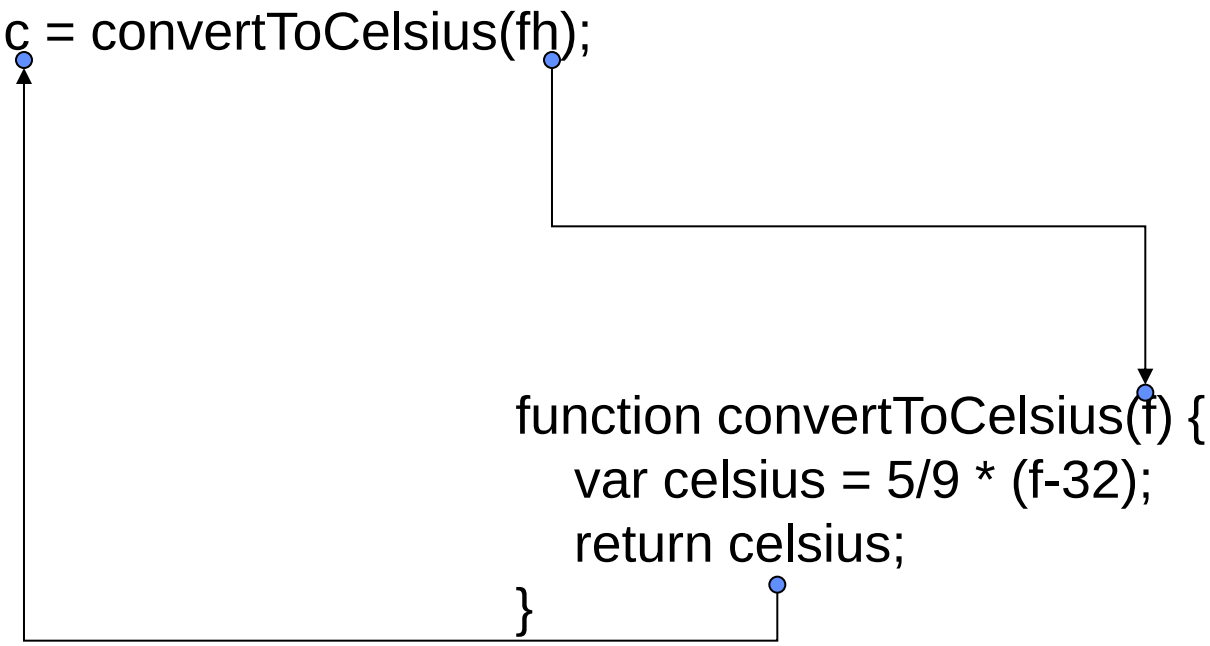
- array.join()
  - Put together all elements as a string
- array.sort()
  - Order alphabetically
- array.reverse()
  - Reverse order of array
- array1.concat(array2,array3,…)
  - Concatenates two or more arrays

Functions

- Reusable code for complex code blocks
  - Takes zero or more values as “parameters”
  - Returns at most one value as the “result”

```
function convertToCelsius(f) {
    var celsius = 5/9 * (f-32);
    return celsius;
}
```

```
var fh = 60;
c = convertToCelsius(fh);
```



More Examples

```
var r = weirdAddition(2, 4);

var a = 2;
var b = 3;
var s = weirdAddition(a, b);
```

```
function weirdAddition(a, b) {
    var result = a + b - 0.5;
    return result;
}
```

Writing JavaScript Functions

- Convenient to put in the <head> section
  - Use <!-- … //--> to prevent display of code

```
…
<head>
<script language="JavaScript" type="text/javascript">
<!--
function calculate() {
  var num = eval(document.input.number.value);
…
  document.output.number.value = total;
}
//-->
</script>
</head>
…
```

Uses of Functions

- Compactness
  - Minimizing duplicative code
- Modular programming
  - Abstraction
  - Reusability
  - Avoid “side effects”

Function Format

```
function Name(parameter1, parameter2,…) {
    statements
}
```

Example

```
<html>
  <head>
    <meta charset=UTF-8" />
     <title>JavaScript Program Template</title>
  </head>
  <body>
    <script>
      /*ask for user input*/
      var day = prompt("Enter day");
      var kind = prompt("Enter kind");
       /*call function*/
      menu(day, kind);
      /* Function definition */
      function menu(day, kind) {
        var actualChoices;
        if (day === "Monday" || kind === "healthy") {
          actualChoices = "<h2>chicken</h2>";
        } else {
          actualChoices = "<h2>steak, cheesecake, whip cream, lard</h2>";
        }
        document.writeln("The " + kind + " Menu for " + day + " ");
        document.writeln(actualChoices);
      }
    </script>
  </body>
</html>
```

Parameter Passing

- Scalars are copied
  - "Pass by value"
- Arrays (and all objects) pass "by reference"
  - The values in the array are **not** copied
    - Be careful to make "side effects" explicit
  - No need to return the **same** reference
- Functions can also be passed as parameters
  - Unchangable, so "by reference" = "by value"
- Returned values work the same way

Returning a Function's Result

```
function Name(parameter1, parameter2,…) {
    statements;
    return value;
}
```

```
<html>  <head>
      <meta charset="utf-8" />
      <title>Functions</title>
   </head>
   <body>
         <script>
         /* calling main */
         main();
         function main() {
            /* Where we call the other functions */
            var firstValue = Number(prompt("First Value"));
            var secondValue = Number(prompt("Second Value"));
            var max;
            max = maxValue(firstValue, secondValue);
            alert("Maximum value is: " + max);
         }
         function maxValue(value1, value2) {
            var maximum;
            if (value1 > value2) {
               maximum = value1;
            } else {
               maximum = value2;
            }
            return maximum;  // what happens if we don't return a value?
         }
         </script>   </body>  </html>
```

Hands On

Write a JavaScript function that asks for two names and returns the one with the shortest length

```
<html>
    <body>
        <script>

        /* calling main */
            var firstName = prompt("First Name");
            var secondName = prompt("Second Name");

            var max;
            max = maxValue(firstValue, secondValue);
            alert("Longest string is: " + max);

            function maxValue(value1, value2) {
                var maximum;

                if (value1.length > value2.length) {
                    maximum = value1;
                } else {
                    maximum = value2;
                }

            return maximum;  // what happens if we don't return a value?
        }

        </script>
   </body>
</html>
```

Scope of a Variable

- In JavaScript, *var* "declares" a variable

  var mystery; create a variable without defining its type

  var b = true; create a boolean *b* and set it to true

  var n = 1; create an integer *n* and set it to 1

  var s = "hello"; create a string *s* and set it to "hello"

- Variables declared in a function are "local"
  - Function parameters are implicitly declared (local)
  - Same name outside function refers to **different** variable

- All other variables are "global"

Global vs Local Variables

```
<html>
  <head>
    <meta charset="utf-8" />
    <title>Pass by value</title>
  </head>

  <body>
    <script type="text/javascript">
      /* calling main */
      main();

      var m = 10, t = 20;
      document.writeln("Before m value is: " + m + ", t value is: " + t + "<br />");
       wrongSwap(m, t);
      document.writeln("After m value is: " + m + ", t value is: " + t);

      /* What is wrong with this function? */
      function wrongSwap(x, y) {
        var temp = x;
        x = y;
        y = temp;
      }
       </script>
  </body>
</html>
```

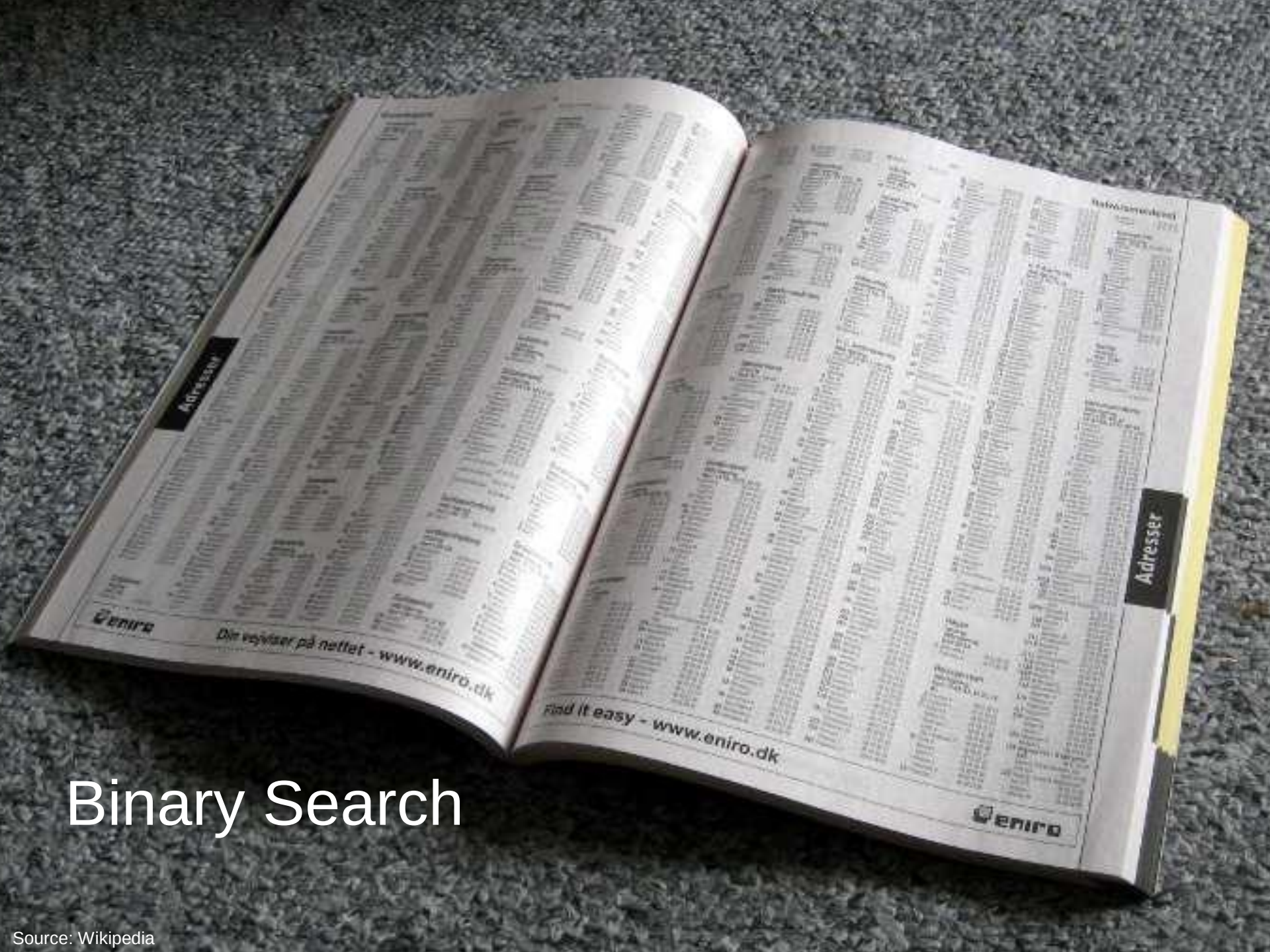
What does this program do?
Try it!

Modify Global Variables

```
<html>

  <body>
    <script type="text/javascript">


      var m = 10, t = 20;
      document.writeln("Before m value is: " + m + ", t value is: " + t + "<br />");
      wrongSwap(m, t);
      document.writeln("After m value is: " + m + ", t value is: " + t);


      /* What is wrong with this function? */
      function wrongSwap(x, y) {
        var temp = x;
        var temp2 = y;
        m = temp2;
        t = temp;

      }
      </script>
  </body>
</html>
```

Sequential Search

Source: Wikipedia



Binary Search

Source: Wikipedia

Recursion

Source: Wikipedia

Recursion

- A function can call itself
  - Local variables are different each time
- Every invocation of the function must end
  - There must be a path that ends the recursion
  - That path must eventually be taken
  - The usual way to do this is an initial **if** statement
- Never essential
  - But sometimes more elegant than iteration

Binary Search with Recursion

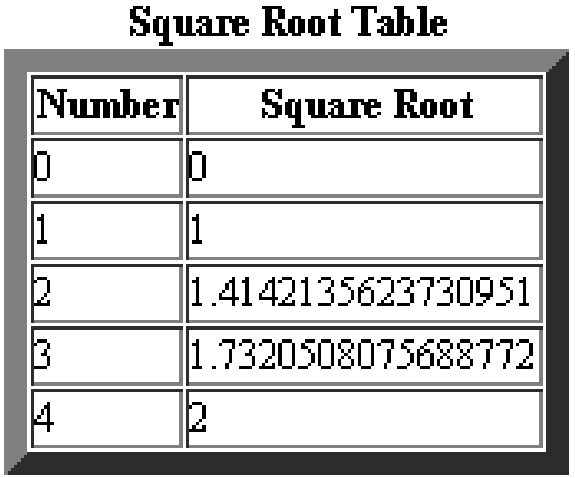
```
function binarySearch(theArray, key, low, high) {
  var middle;
  if (low>=high) {                          // Safety check!
    if (key==theArray[low]) {
      return low;
    } else {
      return -1;
    }
  } else {
    middle = Math.floor((low+high)/2); // Explicit!
    buildOutput( theArray, low, middle, high );
    if (key<=theArray[middle]) {          // Equality!
      return binarySearch(theArray, key, low, middle);
    } else {
      return binarySearch(theArray, key, middle+1, high);
    }
  }
}
```

Some Math Functions

- Math.abs() – Absolute value
  - Example: Math.abs(-10)
- Math.max() – Maximum of two values
  - Example: Math.max(10, 20)
- Math.sqrt() – Square root
  - Example: Math.sqrt(4)
- Math.random() – Random value between 0 and less than 1
  - Example: Math.random()
- Constants
  - Math.PI – Mathematical constant pi

One More Example

- Write a JavaScript program that asks for a number (n) and writes an HTML table with two columns:
  - Column1: numbers 0 to n
  - Column2: square root of number



Square Root Table

Number	Square Root
0	0
1	1
2	1.4142135623730951
3	1.7320508075688772
4	2

For n=4

```
<!doctype html>
<html>
   <head>
      <meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
            <title>Square Root Table</title>
   </head>
   <body>
      <script type="text/javascript">
         var currValue = 0;
         var maximumValue;
         maximumValue = Number(prompt("Enter maximum value"));

         document.writeln("<table border=\"10\">");
         document.writeln("<caption> Table</caption>");
         document.writeln("<tr><th>Number</th><th>2*Number</th></tr>");

         while (currValue <= maximumValue) {
            document.writeln("<tr><td>" + currValue + "</td><td>" + currValue*2 +"</td></tr>");
            currValue = currValue + 1;
         }

         document.writeln("</table>");
      </script>
   </body>
</html>
```

Documentation Tips

- Reflect your pseudocode in your code
  - Use meaningful variable names
  - Use functions for abstractable concepts
    - And name those functions well
  - Use comments to fill remaining gaps
- Add a comment to identify each revision
  - Give author, date, nature of the change
- Waste space effectively
  - Use indentation and blank lines to guide the eye

Using JavaScript with Forms

HTML:

```
<form name="input" action=" ">
   Please enter a number:
   <input size="10" value=" " name="number"/>
</form>
<form name="output" action=" ">
   The sum of all numbers up to the number above is
   <input size="10" value=" " name="number" readonly="true"/>
</form>
```

Reads in a value from the first form
(*eval* method turns it into a number)

JavaScript:

```
var num = eval(document.input.number.value);
document.output.number.value = 10;
```

Changes the value in the second form

HTML Form Element Types

- Textarea (multiple lines)
- Input
  - Text (single line)
  - Password (like text, but masked)
  - Hidden (like text, but not displayed at all)
  - Button
  - Checkbox (multiple selection)
  - Radio (single selection)
- Select (dropdown list)

see `http://www.w3schools.com/html/html_forms.asp` for examples

Linking Forms to Functions

- Events:
  - Actions that users perform

- An “event handler” is triggered by an event
  - onClick: the user clicked on the item
  - onMouseover: the mouse moved onto the item
  - onMouseout: the mouse moved off of the item

Referring to Form Content

```
<form name=years>
  <b>Please enter your age</b>
  <input type=text name=box />
</form>
…
var age = document.years.box.value;
```

```
<form action = " ">
   <p>Enter integer search key<br />
   <input id = "inputVal" type = "text" />
</form>
…
var inputVal   = document.getElementById("inputVal");
var searchKey  = inputVal.value;
```

Looking Ahead

- Objects
- Events

Events

- Events allow an HTML webpage to react to a users' behavior (event handler)
  - Click of a mouse
  - Mouse over an element
  - User strokes a key
  - Image has been loaded
  - Input field is changed (forms)

More complex events…Forms

- Click on button to select an action
- Text box for entering a line of text
- Radio buttons for making one selection among a group of options
- Check boxes for selecting or deselecting a single, independent option
- Lists of things to select one

Push Button

```
<!DOCTYPE html>
<html>
<body>

<form>
   <input type="button" value="This is a button">
</form>

</body>
</html>
```

Text Box

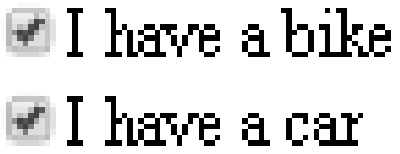
```
<!DOCTYPE html>
<html>
<body>

<textarea rows="3" cols="30">
    This is a text area.
</textarea>

</body>
</html>
```

Radio Button

```
<!DOCTYPE html>
<html>
<body>

<form>
  <input type="radio" name=“fruit" value="apples">Apples<br>
  <input type="radio" name=“fruit" value="oranges">Oranges<br>
  <input type="submit" value="Send">
</form>

</body>
</html>
```

Check Box

```
<!DOCTYPE html>
<html>
<body>

<form >
  <input type="checkbox" name="vehicle" value="Bike">I have a
    bike<br>
  <input type="checkbox" name="vehicle" value="Car">I have a car
</form>

</body>
</html>
```



List

```
<!DOCTYPE html>
<html>
<body>

<form >
  <select name="fruits">
    <option value="oranges">Oranges</option>
    <option value="apples">Apples</option>
    <option value="lemons">Lemons</option>
    <option value="bananas">Bananas</option>
  </select>
</form>

</body>
</html>
```

Handling Events…JavaScript!

- JavaScript is responsible for “doing things” when buttons are clicked

```
<input type="button" value="This is a button">

<input type="button" value="This is a button"
onclick="myFunction()">
```

Handling Events…JavaScript!

```
<input type="button" value="This is a button"
onclick="myJSFunction()">

<script>
function myJSFunction()
{ do something}
</script>
```

Example with Push Button

Try this

```
<!DOCTYPE html>
<html>
<body>
<script>
function myJSFunction()
{
document.getElementById("demo").innerHTML="Hello World";
}
</script>
<form>
   <input type="button" onclick="myJSFunction()" value="This is a button">
</form>

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

Text Box: access text

```
<p><input type="text"
   id="textInput"><br/></p>


var textInput =
   document.getElementById("textInput");
var textValue = textInput.value;
alert(textValue);
```

Text Box

```
<!doctype html>
<html lang="en">
<body
<title>Forms</title>
<meta charset="utf-8">

<script>
function handleClick() {
  var textInput = document.getElementById("textInput");
  var textValue = textInput.value;
  alert("The input text is: " + textValue);
}
</script>
</head>
<body>

<form>
<p><input type="text" id="textInput" size="40"><br/></p>

<p><input onclick="handleClick()" type="button" id="button" value="Click me!"></p>
</form>

</body>
</html>
```

USE THE DOM

Radio Button

```
<input id="radioone" type="radio" name=“select"
   value=“one“>One
<input id="radiotwo" type="radio" name=“select"
   value=“two“>Two</p>

var onechecked =
   document.getElementById("radioone").checked;
alert(“button one checked: " + onechecked);
```

```
<!doctype html>
<html lang="en">
<body
<title>Forms</title>
<meta charset="utf-8">

<script>
function handleClick() {
  var onechecked = document.getElementById("radioone").checked;
  var twochecked = document.getElementById("radiotwo").checked;
  alert("one checked: " +  onechecked+" two checked "+twochecked);

}
</script>
</head>
<body>

<form>
   <input id="radioone" type="radio" name=“select" value=“one“>One
   <input id="radiotwo" type="radio" name=“select" value=“two“>Two</p>

   <p><input onclick="handleClick()" type="button" id="button" value="Click me!"></p>
</form>

</body>
</html>
```

Check Box

```
<input id="checkDog" type="checkbox"
   name="pet" value="dog">I have a dog<br>
```

```
var hasDog =
   document.getElementById("checkDog").checked;
```

```
<!doctype html>
<html lang="en">
<body
<title>Forms</title>
<meta charset="utf-8">

<script>
function handleClick() {
  var hasDog = document.getElementById("checkDog").checked;
  var hasCat = document.getElementById("checkCat").checked;
  alert("dog? " + hasDog+ " cat? " +  hasCat);

}
</script>
</head>
<body>

<form>
<p>
<input id="checkDog" type="checkbox" name="pet" value="dog">I have a dog<br>
<input id="checkCat" type="checkbox" name="pet" value="cat">I have a cat
</p>

<p><input onclick="handleClick()" type="button" id="button" value="Click me!"></p>
</form>

</body>
</html>
```

List

```
<select id="selectInput">
  <option value="oranges">Oranges</option>
</select>

<script>
var selectInput =
    document.getElementById("selectInput");
var selectedIndex = selectInput.selectedIndex;
 var selectedValue =
    selectInput.options[selectedIndex].value;
```

```
<!doctype html>
<html lang="en">
<body
<title>Forms</title>
<meta charset="utf-8">

<script>
function handleClick() {
  var selectInput = document.getElementById("selectInput");
  var selectedIndex = selectInput.selectedIndex;
  var selectedValue = selectInput.options[selectedIndex].value;
  alert("Which fruit do you prefer? " + selectedValue);

}
</script>
</head>
<body>

<form>
<p>
<select id="selectInput">
  <option value="oranges">Oranges</option>
  <option value="lemons">Lemons </option>
  <option value="apples">Apples</option>
  <option value="bananas">Bananas</option>
</select>
</p>

<p><input onclick="handleClick()" type="button" id="button" value="Click me!"></p>
</form>

</body>
</html>
```

Before You Go

On a sheet of paper, answer the following (ungraded) question (no names, please):

What was the muddiest point in today's class?