Arrays

INFO/CSE 100, Spring 2006
Fluency in Information Technology

http://www.cs.washington.edu/100
HTML Form Controls

<form name="sample_form">
    First Name: <input type="text" name="fName" id="fName" size="20" maxlength="20" />

    Gender: <input type="radio" name="gender" id="male" checked /> <input type="radio" name="gender" id="female" />

    Yankees: <input type="checkbox" id="yankees" />
    Mets: <input type="checkbox" id="mets" />

    Age: <select>
        <option value="0-18">0-18</option>
        <option value="19-62">19-62</option>
        <option value="63+">63 +</option>
    </select>

    <button type="reset">Reset</button>
    <button type="submit">Submit</button>
</form>
Sample_form.html
Collections in the Real World

• Think about:
  » words in a dictionary
  » list of pets in your household
  » deck of cards
  » books in a library
  » songs on a CD
  » controls in an HTML form

• These things are all *collections* of objects
A Collection of Form Elements

• Lets check the gender radio buttons:

```javascript
var elements = new Array();
elements = document.getElementById("gender");

for (var i=0; i < elements.length; i++) {
    var element = elements[i];
    if (element.checked) {
        document.writeln("Your gender is: " + element.value);
    }
}
```
How can we manage lists of objects?

• We'd like to be able to ...
  » add things to the list
  » look at the elements of the list one by one
  » find out how many things have been put in the list
  » remove things from the list
  » … among other things
iCCC example

• Consider the iCCC example program
  » There are 4 radio buttons for shot count, 3 radio buttons for cup size, and 4 radio buttons for drink
  » We could give each radio button an id and check it individually to see if it is currently selected
  » But it's much cleaner to treat the buttons in each group the same way, and just look at them in turn

• Looping over the elements of a group is often simpler and more flexible than treating them individually
for (var i=0; i<document.getElementById("shotForm").elements.length; i++) {
    element = document.getElementById("shotForm").elements[i];
    if (element.checked) {
        shotCount = parseInt(element.value, 10);
    }
}
Arrays

• JavaScript (and most other languages) includes *arrays* as the most basic kind of collection.
  » Simple, ordered collections
  » Special syntax for accessing elements by position

• JavaScript arrays can be created
  » by the programmer in the script
  » by the system and provided to the script
    • for example, the elements array in the iCCC program
Array Example

arraysA.html
Array Example

Variable
petNames

Array
length: 5

String
"Jaba"

String
"Bingo"

String
"Jessica"
JavaScript Indexed Arrays

- An indexed array is a data type that stores a collection of values, accessible by number
  - the values in the array are called the *elements* of the array
  - the elements (or values) are accessed by *index*
    - the index of the first value is 0
  - the values in the array can be any type
    - usually all the values are the same type
    - but they can be different from one another if necessary
Array Declaration and Creation

- Arrays can be created several different ways
  - `var petNames = new Array();`
    - 0-length array with no elements in it yet
  - `var studentNames = new Array(102);`
    - 102-element array, all of which have the value `undefined`
  - `var myList = ["Sally", "Splat", "Google"];`
    - 3-element array initialized with an array literal

- Arrays have a property that stores the length
  - `<array name>.length`
    - you can lengthen or shorten an array by setting the length to a new value
Array Element Access

- Access an array element using the array name and position: `<array name> [ <position> ]`

- Details:
  » `<position>` is an integer expression.
  » Positions count from zero

- Update an array element by assigning to it:
  ```
  <array name> [ <position> ] = <new element value> ;
  ```

  ```
  myCurrentCarNo = carList.length-1; 
  myCurrentCar = carList[myCurrentCarNo];
  ```
<html>
<head>
<title>Arrays Example B</title>
<script type="text/javascript">
var petNames = new Array();
var studentNames = new Array(102);
var myList = ["Sally", "Splat", "Google"];
</script>
</head>

<body>
<script type="text/javascript">
document.write("<br>petNames has " + petNames.length + " elements.");
document.write("<br><br>studentNames has " + studentNames.length + " elements.");
if (studentNames.length > 0) {
    document.write("<br>The first student name is " + studentNames[0] + ".");
}
document.write("<br><br>myList has " + myList.length + " elements.");
if (myList.length > 0) {
    document.write("<br>" + myList.join("", ") + ".");
}
</script>
</html>

arraysB.html
Array References

```javascript
var dwarf = new Array(7);
var deux = 2;
dwarf[0] = "Happy";
dwarf[1] = "Sleepy";
dwarf[deux] = "Dopey";
dwarf[deux+1] = "Sneezy";
```
Looping Over Array Contents

- The length attribute makes it easy to loop over all the elements of an Array:

```javascript
document.write("<br>Unsorted list of pet names.<br>");
for (var i=0; i<petNames.length; i++) {
    if (i != 0) {
        document.write("", ");
    }
    document.write(petNames[i]);
}
```
deleting elements

• Change the length property to change the number of elements in the array
  » names.length = 4;

• Use the delete operator to set a particular entry to the value undefined
  » delete names[0];
<body>
<script type="text/javascript">
// 2-element array literal
var names = ["alex","brenda"];
document.write("length: "+names.length);

// extend it to 4 elements
names.length = 4;
document.write("<br><br>length: "+names.length);
for (var i =0; i<names.length; i++) {
    document.write("<br>"+names[i]);
}

// delete, assign, and shorten
delete names[0];
names[2] = "carrie";
names.length = 3;
document.write("<br><br>length: "+names.length);
for (var i =0; i<names.length; i++) {
    document.write("<br>"+names[i]);
}
</script>
</body>
interesting functions

• There are several predefined functions available for working with arrays
  » join() ← join all the elements in one long string
  » reverse() ← reverse the order of the elements
  » sort() ← sort the elements in the array
  » concat(...) ← add elements to the array
  » etc

```javascript
document.write("<br><br>Sorted list of pet names.<br>");
petNames.sort();
...
```
Sort function

```javascript
var petNames = ("Jaba", "Bingo", "Jessica", "Sumi", "Jennifer");

petNames.sort();
document.write(petNames);

Output:
Bingo, Jaba, Jennifer, Jessica, Sumi
```
Array Summary

- Arrays are a collection data type built in to the JavaScript language.
  » Also found in essentially all programming languages
- Indexed access to elements
  » remember, it's 0-based, the first element is element 0
- Elements can be added to an array by specifying the index value in the assignment statement

```
petNames[5] = "Eleanor";
```
- There are useful functions available for manipulating arrays
Some Built-in Javascript Functions

• Getting a document object
  
  var element = document.getElementById("shotforms");

• Getting today's date

  var today = new Date();
  document.write("Today is: "+ today.toString());

• Random Numbers

  math.random();  \(<\!\<\text{ produces a random number between 0 and 1}\!\>
  
  aRandomNumber = 75 * math.random();  \(<\!\<\text{ produces a random number between 0 and 75}.\!\>\)