Digital Information

INFO/CSE 100, Spring 2005

Fluency in Information Technology

http://www.cs.washington.edu/100

Readings and References

- Reading
  - » *Fluency with Information Technology*
    - Chapters 9, 11 18-21

Variables In Real Life

- A variable is a "container" for information you want to store
  - » The name of the variable stays the same, but the value associated with that name can change

That's why it's called a "variable"!

Variable **Name**	Current **Value**	Previous **Value**
#1 Single	My Boo, Usher And Alicia Keys	Goodies, Ciara
AL Champion	Boston Red Sox	New York Yankees
#1 Box Office	Shark Tale	Shark Tale
Day Of The Week	Monday	Sunday
Husky Card Balance	$52	$60

Variables In Programming

- Program variables have names and values
  - » Names (also called identifiers)
    - generally start with a letter and can contain letters, numbers, and underscore characters "_"
    - Names are *case sensitive*
  - » Values
    - can be numbers, strings, boolean, etc
    - change as the program executes

Variable **Name**	Current **Value**	Previous **Value**
No_1_Single	My Boo, Usher And Alicia Keys	Goodies, Ciara
ALChampion	Boston Red Sox	New York Yankees
No_1_Box_Office	Shark Tale	Shark Tale
dayOfTheWeek	Monday	Sunday
huskyCardBalance	$52	$60

Variable Declarations

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

var eyeColor;  <<< undefined!

var eyeColor = "green";  <<< initialized

var eyeColor = ""; <<< initilized, empty

var eyeColor = "green", hairColor="blonde";

hairColor = "carmel";
</script>
```

Basic Data Types in Javascript

Numbers:

```
var gasPrice = 2.55;
```

Strings

```
var eyeColor = "hazel green";
```

Boolean

```
var isFriday = true;
var isWeekend = 0;
```

Expressions

- The right-hand side of an assignment statement can be any valid *expression*
- Expressions are "formulas" saying how to manipulate existing values to compute new values

```
balance = balance - transaction;
seconds = 60*minutes;
message = "Status code is " + codeValue;
isFreezing = (temp < 32);
```

Operators

Use operators to build expressions

- Numeric operators

  + - * / *mean* add, subtract, multiply, divide

  3 + 3 = 6

- String operator

  + *means* concatenate strings

  "3" + "3" = "33"

- Relational operators

  < <= == != >= > *mean* less than, less than or equal to, equal to, not equal to, greater than or equal to, greater than

- Boolean operators

  && || ! *mean* and, or, not

Functions

A *function* is a way to bundle a set of instructions and give them a name so that you can reuse them easily

Functions have a specific layout

- » *<name>* ← the function name is an identifier
- » *<parameter list>* ← list of input variables for the function
- » *<statements>* ← the statements do the work

```
function <name> ( <parameter list> ) {
    <statements>
}
```

Example Function

template

```
function <name> ( <parameter list> ) {
   <statements>
}
```

Write a simple function to compute the Body Mass Index when the inputs are in English units (ie, US units)

example

```
// Calculate Body Mass Index in English units
// weight in pounds
// height in inches
// returns body mass index

function bmiE(weightLBS, heightIN)  {
  var heightFt = heightIn / 12; // convert to feet
  return 4.89 * weightLBS / (heightFt * heightFt);
}
```

Calling a Function

parameters

```
// Calculate Body Mass Index in English units
// weight in pounds
// height in inches
// returns body mass index

function bmiE(weightLBS, heightIN)  {
  var heightFt = heightIn / 12; // convert to feet
  return 4.89 * weightLBS / (heightFt * heightFt);
}
```

function calls

```
// call the bmiE function
var bmi = bmiE(162, 51);

// another function call
document.write(bmiE(162, 51));
```

arguments

Global or Local?!?

- Scope of a variable describes where and when it can be referenced
  - » Local variables are only known inside of a function (curly braces)
  - » Global variables are know by all the Javascript inside of <script> </script> pairs

```
// Calculate Percentage of Study Hours/Week
// time in hours
// returns hours
var days = 7;
function calculateStudyHrs(time)  {
  var totalHrs = 24 * days;
  return time/totalHrs;
}
```

Layout of the GUI

- The layout of the page is controlled with HTML in the body of the page

```
<body>
  HTML form layout and specification
</body>
</html>
```

- The layout and controls are provided using new tags
  - » <form id="buttonForm">
  - » <button type="button" ...
  - » <input type="text" …
  - » <input type="radio" …
  - » <button type="reset" …

A simple example

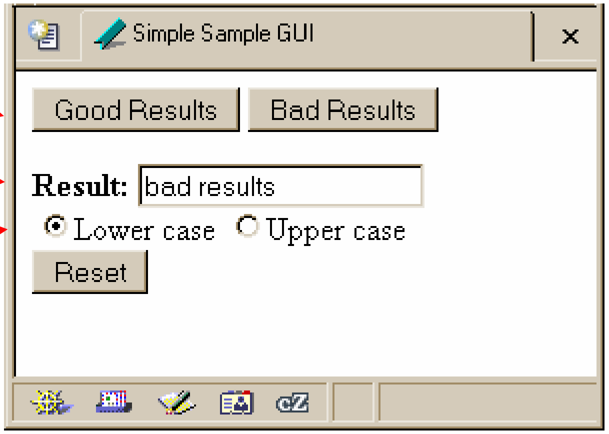
This GUI has several simple controls.

Two buttons to control the results

One text field to display the results

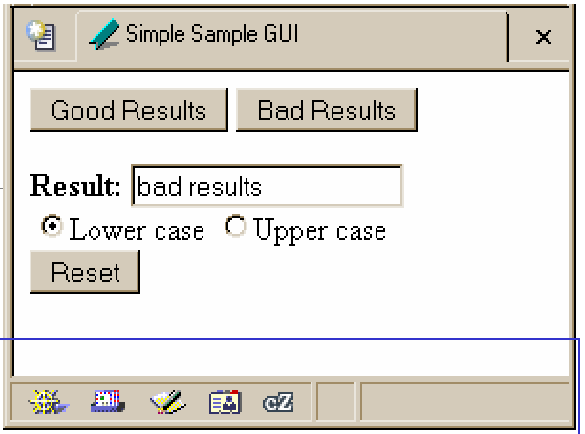
One pair of radio buttons to control the display

One button to reinitialize



http://www.cs.washington.edu/education/courses/100/04au/slides/13-gui/gui.html

Form Controls



Result: bad results

[✓ Lower case] Upper case

```
<form>
<button type="button"
  onclick="setResults('good results')">Good Results</button>
<button type="button"
  onclick="setResults('bad results')">Bad Results</button>
<b>Result:</b>
<input type="text" value="nada" readonly id="resultField">
<br>
<input type="radio" name="case" id="radioLC" checked

onclick="setResults(document.getElementById('resultField').value)
">Lowercase
<input type="radio" name="case" id="radioUC"

onclick="setResults(document.getElementById('resultField').value)
">Uppercase
<br><button type="reset">Reset</button>
</form>
```

Events Cause Processing

- After drawing a page, the browser sits idle waiting for something to happen … when we give input, we cause *events*
- Processing events is the task of a block of code called an event handler
  - » The code to execute is identified in the tag using the appropriate attribute
  - » There are many event types
    - onClick, onChange, onMouseOver ...

setResults(resultString)

```
<script type="text/javascript">
function setResults(resultString) {
  var tempString = resultString;
  if (document.getElementById("radioLC").checked) {
    tempString = tempString.toLowerCase();
  } else if (document.getElementById("radioUC").checked) {
    tempString = tempString.toUpperCase();
  }
  document.getElementById("resultField").value = tempString;
}
</script>
```

parameter variable, local variable, if/else statement, field reference, call to toLowerCase() function

The `if` / `else` statement

The `if` statement is a *conditional statement*

- » a conditional expression is evaluated as being `true` or `false`
  - the expression is a *boolean expression* (ie, returns `true` or `false`)
- » if the condition is `true`, then one set of statements is executed
- » if the statement is `false`, then a different set of statements is executed

condition

```
if (<boolean expression>) {
   <statements>
} else {
   <statements>
}
```

Examples

```
if (count == 0) {
  ready = false;
} else {
  ready = true;
  count = count-1;
}
```

What is the conditional expression?

What statements are part of the true block?

Which statements are part of the false block?

What happens when count is 21?   0? -1?

```
if (pageCount >= 100) {
  alert("This may take a few minutes.");
}
```

What is the conditional expression?

What statements are part of the true block?

Which statements are part of the false block?

What happens when pageCount is 21?   100? 200?

More if/else Statements

```
if (temp < 32) {
  if (sky == "cloudy) {
     alert("Snow is forecast!");
   }
}

if (temp < 32 && sky == "cloudy") {
  alert("Snow is forecast!");
}
```

The **`for`** loop

A counting loop is usually implemented with **`for`**

```
var count = 10;
```

initialize

check for limit

update loop control index
shorthand for **`i=i+1`**

```
for (var i=0; i < count; i++) {
    document.writeln("<br>index value is : "+i);
}
```

one or more statements in the loop body

`i++` is a shortcut

- `for (i=0; i < count; i++)`
- at the end of every pass through the **`for`** loop body, do the following:
  - » get the value of i
  - » increment i
  - » store the incremented value
- Used as it is here, this is the same as writing
  - » **`i = i + 1`**

body of loop may not execute at all

- Notice that depending on the values of the control variables, it is quite possible that the body of the loop will not execute at all

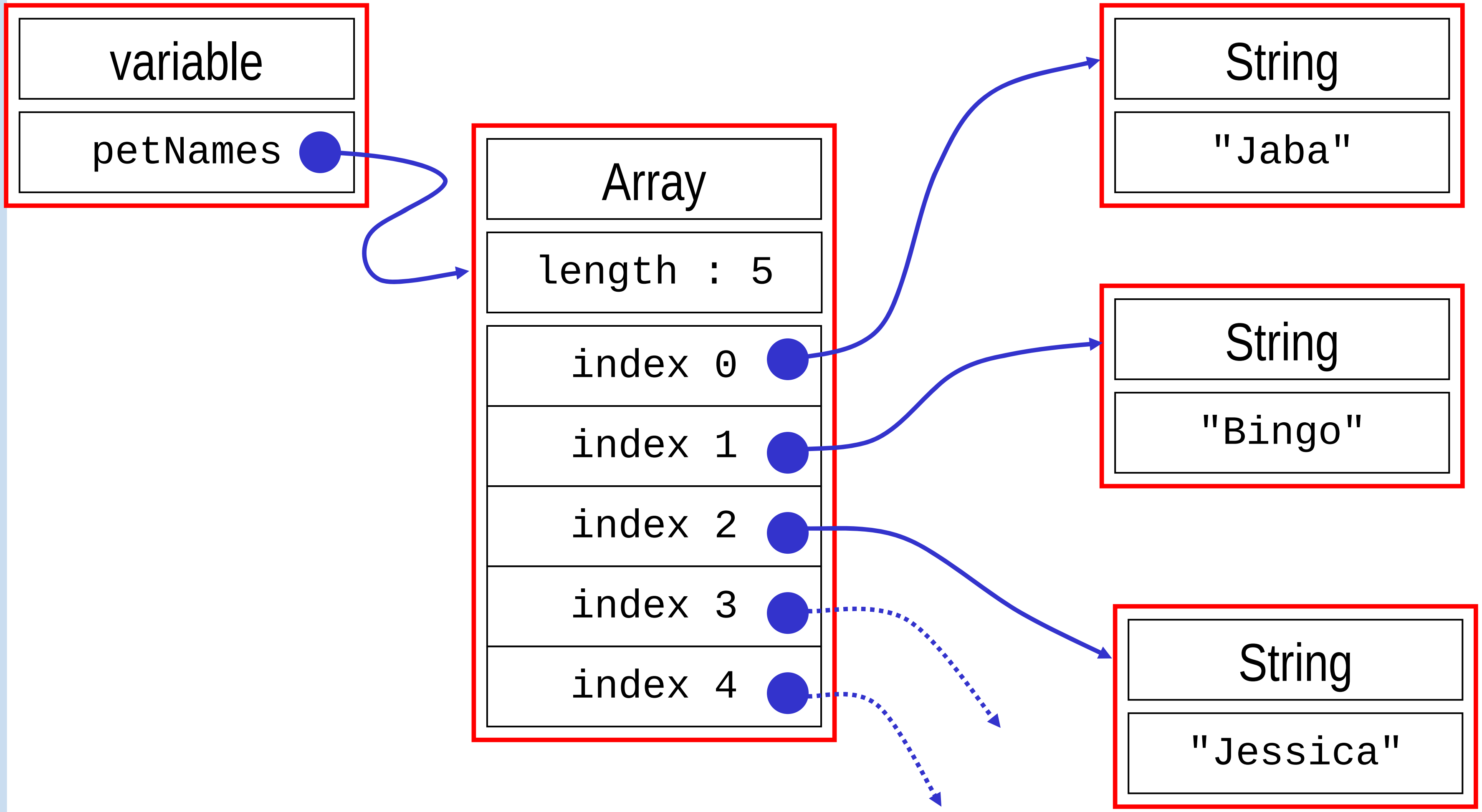
check for limit condition
`itemCount` is 0 when we get here, so **`i<itemCount`** is immediately false and the loop body is skipped completely

```
var itemCount = 0;
...
for (var i=0; i < itemCount; i++) {
    document.writeln("<br>..processing item "+i);
}
```

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



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];
```