



Control Flow

INFO/CSE 100, Spring 2006
Fluency in Information Technology

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

Readings and References

- Reading
 - » *Fluency with Information Technology*
 - Chapter 21, Iteration Principles

<input type="radio" ...>

```

<form>
<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>

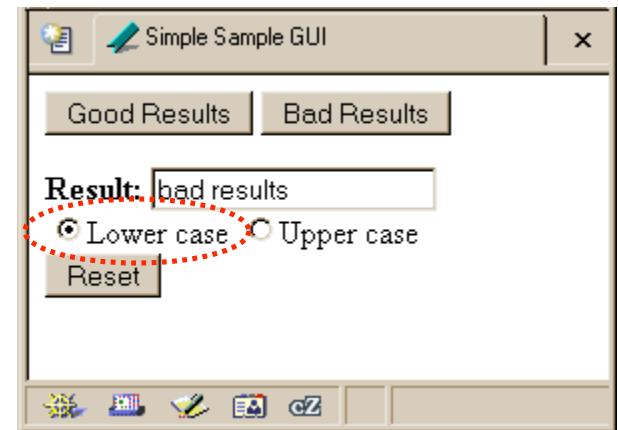
```

an `<input>` with `type="radio"` allows the user to select one of several choices

`name="case"` identifies all the buttons in the same group (only one will be selected at a time)

`onclick` attribute gives the JavaScript to execute when the user clicks this button

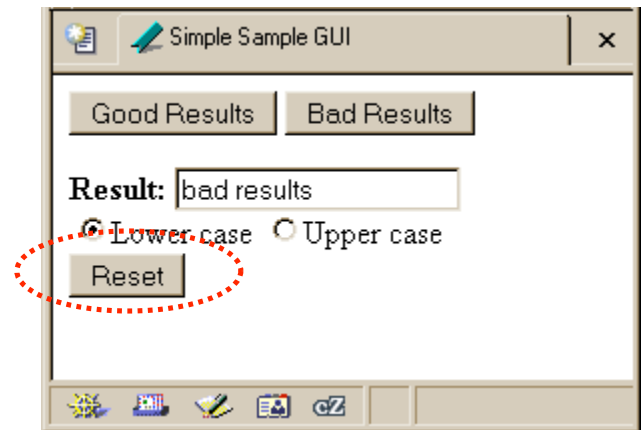
`id="radioLC"` gives us a way to identify this particular control in our JavaScript



<button type="reset" ...>

```
<form>
<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>
```

- a `<button type="reset">` resets all the other controls in the same form to their original values



Events Cause Processing

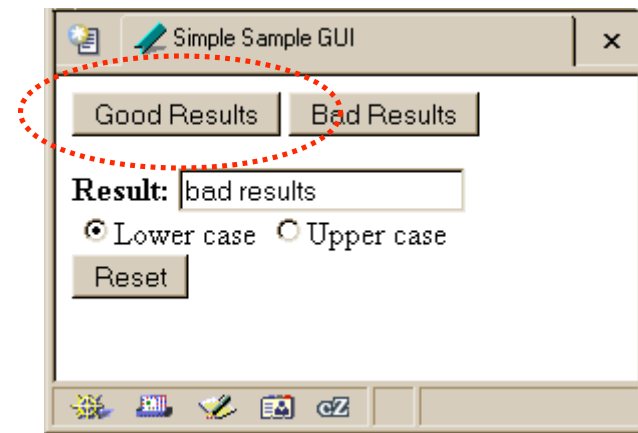
- After drawing a page, the browser sits idle waiting for something to happen ... when we give input, we cause *events*
- Processing events is the task of a block of code called an **event handler**
 - » The code to execute is identified in the tag using the appropriate attribute
 - » There are many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



request processing of an event

```
<form>
<button type="button"
  onclick="setResults('good results')">Good Results</button>
<button type="button"
  onclick="setResults('bad results')">Bad Results</button>
</form>
```

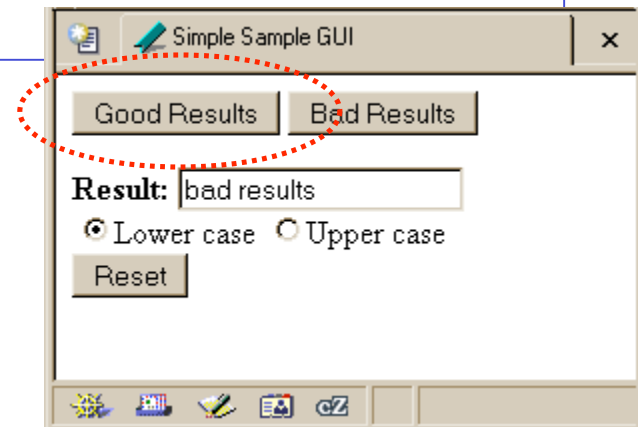
- the `onclick` attribute defines some JavaScript to call when the button is clicked
- in this case, the code is a call to the `setResults(string)` function defined in the page `<head>`
- the appropriate string value is supplied to the `setResults(string)` function and then the function executes



process a button's onclick event

```
<script type="text/javascript">  
function setResultString(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>
```

- the `setResults(string)` function is called by several event processors
- in every case, it takes the string that it is given, decides if upper or lower case is desired, and sets the `resultField` accordingly



setResults(resultString)

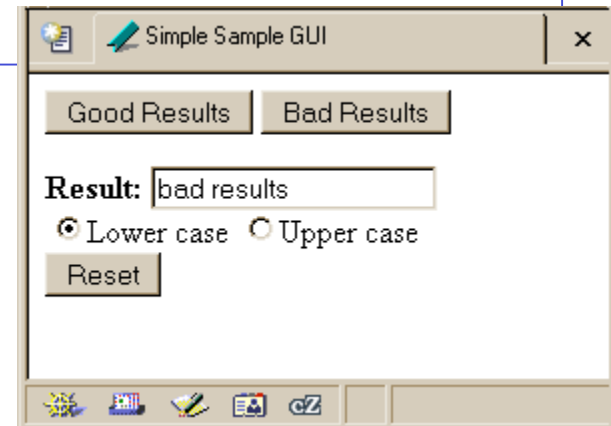
```
<script type="text/javascript">  
function setResults(resultString) {  
    var tempString = resultString;  
    if (document.getElementById("radioLC").checked) {  
        tempString = tempString.toLowerCase();  
    } else if (document.getElementById("radioUC").checked) {  
        tempString =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

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

if statement in Simple Sample GUI

```
<script type="text/javascript">
function setResultString(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>
```

- the `setResults(string)` function is called by several event processors
- in every case, it takes the string that it is given, **decides if upper or lower case is desired**, and sets the `resultField` accordingly



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

```
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?

Which statements are part of the false block?

What statements are part of the true block?

What happens when pageCount is 21? 100? 200?

scratch.html

The screenshot shows a Mozilla browser window with the title "Scratch file for testing - Mozilla". The address bar displays the URL "http://www.cs.washington.edu/education/courses/100/04". The main content area shows the text "This page is a simple place to try out JavaScript." and a "Source of:" panel on the right displaying the HTML source code. A "JavaScript Application" dialog box is open in the foreground, displaying a warning icon and the text "Ready: false, Count: 0" with an "OK" button.

Scratch file for testing

This page is a simple place to try out JavaScript.

[JavaScript Application]

Ready: false, Count: 0

OK

Source of: <http://www.cs.washington.edu/education/courses/100/04au/slides/14-control/scratch.html>

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
  "http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<title>Scratch file for testing</title>
<script type="text/javascript">
var count = 0;
</script>
</head>
<body>
This page is a simple place to try out JavaScript.
<script type="text/javascript">
if (count == 0) {
  ready = false;
} else {
  ready = true;
  count = count-1;
}
alert("Ready: "+ready+", Count: "+count);
</script>
</body>
</html>
```

W3Schools TryIt Editor

The screenshot shows the W3Schools TryIt Editor interface. The browser window title is "Tryit Editor v1.4 - Mozilla". The address bar shows the URL "http://www.w3schools.com/js/tryit.asp?filename=tryjs_text". The main content area is split into two panes. The left pane contains HTML and JavaScript code, and the right pane shows the rendered output. A JavaScript alert dialog is displayed in the center, with the title "[JavaScript Application]" and the message "This may take a few minutes." with an "OK" button.

Tryit Editor v1.4 - Mozilla

File Edit View Go Bookmarks Tools Window Help

Back Forward Reload Stop http://www.w3schools.com/js/tryit.asp?filename=tryjs_text Search Print

Loading...

Edit the text and click me

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01  
Transitional//EN"  
  "http://www.w3.org/TR/html4/loose.dtd">  
<html>  
<head>  
<title>Scratch file for testing</title>  
<script type="text/javascript">  
var pageCount = 100;  
</script>  
</head>  
  
<body>  
This page is a simple place to try out JavaScript.  
<script type="text/javascript">  
if (pageCount >= 100) {  
  alert("This may take a few minutes.");  
}  
</script>  
</body>  
</html>
```

This page is a simple place to try out JavaScript.

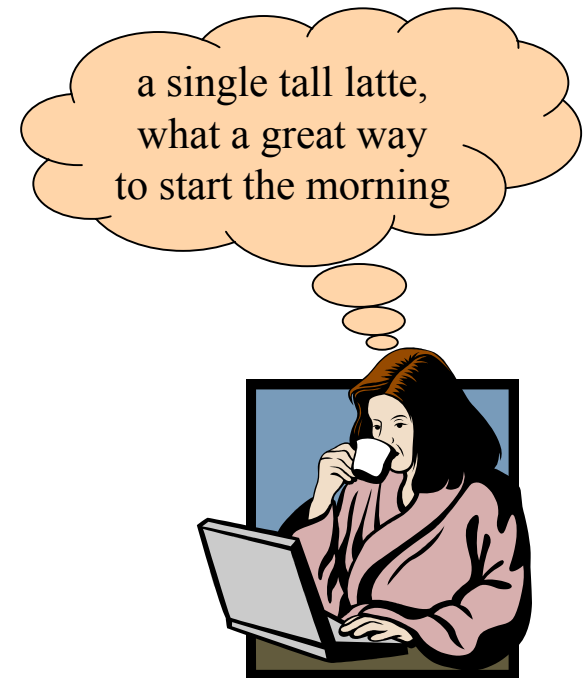
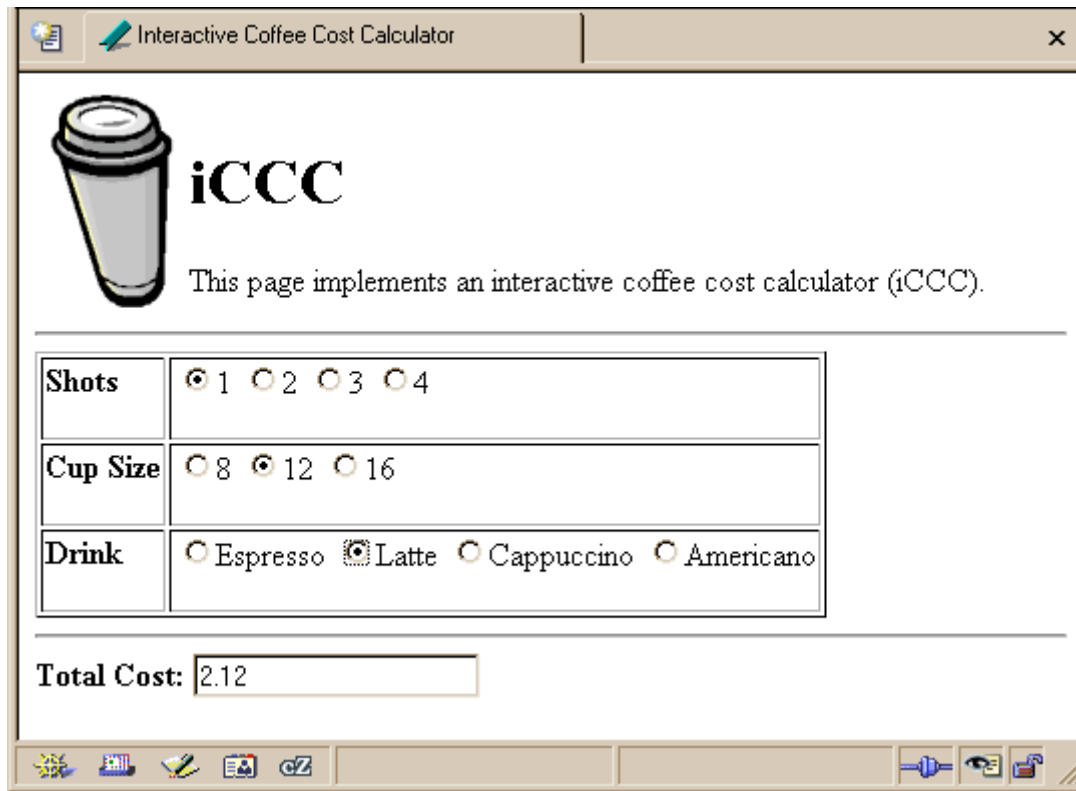
[JavaScript Application] This may take a few minutes. OK

Edit the text above, and click on the button to see the result.

Transferring data from www.w3schools.com...



A Fancier Example of a GUI program



An `if` statement from `bean.html`

```
<html>
<head>
<title>Interactive Coffee Cost Calculator</title>
<script type="text/javascript">
function refresh() {

    var shotCount;           // number of espresso shots
    var cupSize;             // size of the cup in ounces
    var drink;               // name of the requested drink

    var price;               // calculated price of the drink
    var taxRate = 0.087;     // Seattle retail tax

    var element;            // the current gui element (radio button)

    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);
        }
    }
    ...
}
```

Nested if/else Statements

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

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




Iteration

- Iteration or looping is a way to execute a block of program statements more than once
- we will use the **for** statement to create loops
 - » The **for** loop is generally controlled by counting
 - » There is an index variable that you increment or decrement each time through the loop
 - » When the index reaches some limit condition, then the looping is done and we continue on in the code



Why do we want loops in our code?

- Do something for a given number of times or for every object in a collection of objects
 - » for every radio button in a form, see if it is checked
 - » for every month of the year, charge \$100 against the balance
 - » calculate the sum of all the numbers in a list
 - » etc.
- Many loops are counting loops
 - » they do something a certain number of times

The **for** loop

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

```
var count = 10;
```

initialize

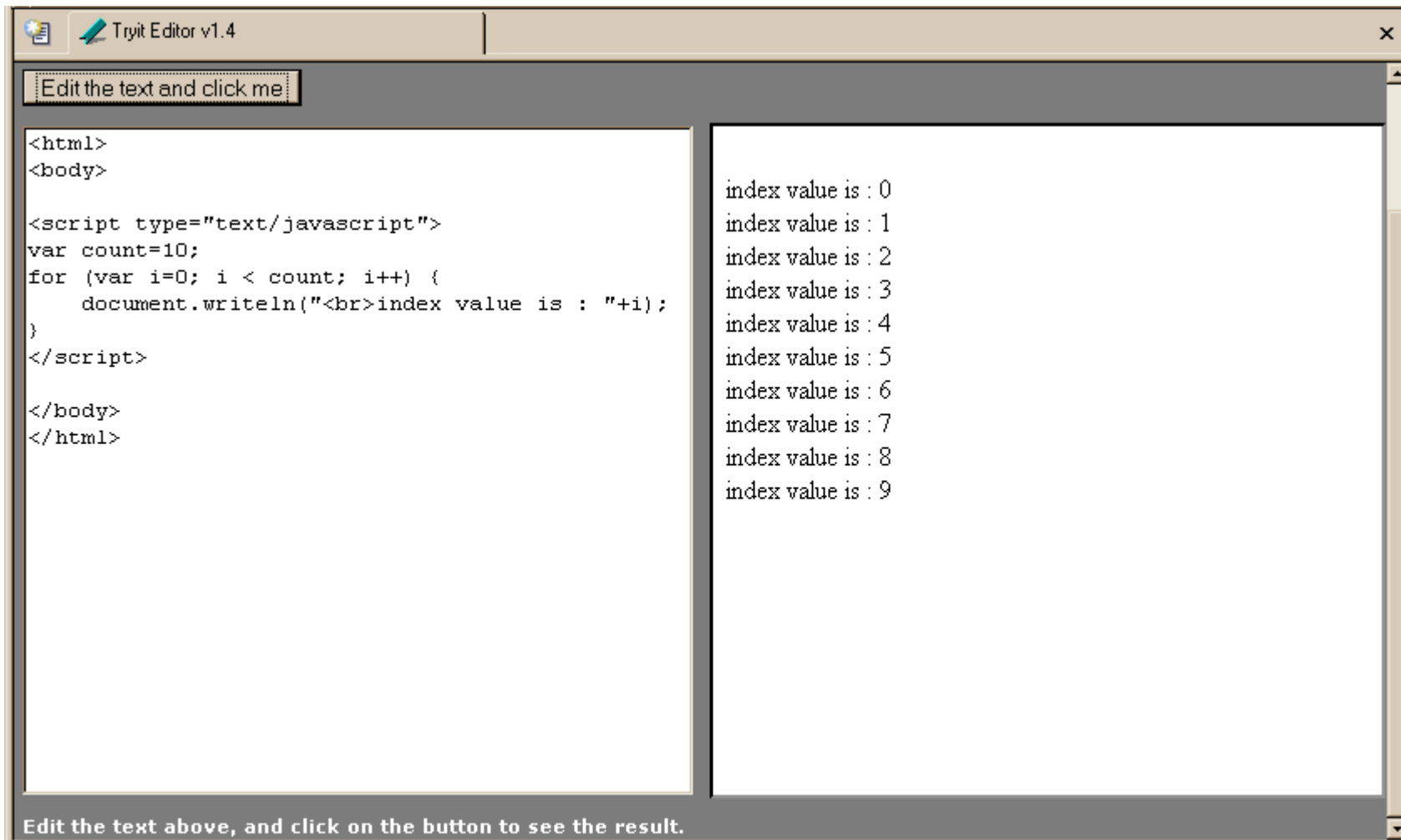
check for limit

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

for example



The image shows a screenshot of the Tryit Editor v1.4 interface. The editor is split into two panes. The left pane contains the following HTML and JavaScript code:

```
<html>
<body>

<script type="text/javascript">
var count=10;
for (var i=0; i < count; i++) {
    document.writeln("<br>index value is : "+i);
}
</script>

</body>
</html>
```

The right pane displays the output of the code, which is a list of ten lines, each containing the text "index value is : " followed by a number from 0 to 9. The output is as follows:

```
index value is : 0
index value is : 1
index value is : 2
index value is : 3
index value is : 4
index value is : 5
index value is : 6
index value is : 7
index value is : 8
index value is : 9
```

At the top of the editor, there is a button labeled "Edit the text and click me". At the bottom of the editor, there is a message that says "Edit the text above, and click on the button to see the result."



`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);  
}
```





loop body skip

Edit the text and click me

```
<html>
<body>
Begin processing.
<script type="text/javascript">
var itemCount=0;
//...
for (var i=0; i < itemCount; i++) {
    document.writeln("<br>..processing item "+i);
}
</script>
<br>End processing.
</body>
</html>
```

Begin processing.
End processing.

Edit the text above, and click on the button to see the result.

“Off By 1” Error

- The most common error when working with iterations is to miscount by 1
 - » *Everyone* makes this mistake
 - » A common place where the “off by 1” error matters is in how many times a loop loops
 - » One advantage of a simple loop control statement is that it's easier to tell how many loops there will be

```
for ( var i=0; i<n; i++) {  
    <statement list>  
}
```

Number of iterations





Avoid Infinite Loops

```
var count = 10;  
  
for ( var i=0; i<count; j++) {  
    document.write("All work and no play, makes Jack a dull boy.");  
}
```



Another Example from the iCCC

The screenshot shows a web browser window titled "Interactive Coffee Cost Calculator". The page features a coffee cup icon and the text "iCCC" followed by "This page implements an interactive coffee cost calculator (iCCC)". Below this is a form with three sections: "Shots" with radio buttons for 1, 2, 3, and 4; "Cup Size" with radio buttons for 8, 12, and 16; and "Drink" with radio buttons for Espresso, Latte, Cappuccino, and Americano. At the bottom, a "Total Cost:" label is followed by a text input field containing the value "2.88". The browser's taskbar at the bottom shows various system icons.

a double tall
Cappuccino, what a
great way to start the
afternoon





A for loop from bean.html

```
<html>
<head>
<title>Interactive Coffee Cost Calculator</title>
<script type="text/javascript">
function refresh() {

    var shotCount;           // number of espresso shots
    var cupSize;             // size of the cup in ounces
    var drink;               // name of the requested drink

    var price;               // calculated price of the drink
    var taxRate = 0.087;     // Seattle retail tax

    var element;            // the current gui element (radio button)

    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

- On the previous page, we are selecting one element from a collection of elements
- this collection is an array named **elements**
 - » one entry for each radio button in the shotForm
 - » the length of this array is available

```
document.getElementById("shotForm").elements.length
```
 - » we retrieve an individual element using the index variable

```
element =  
document.getElementById("shotForm").elements[i];
```
 - » The index of the first element is 0