Looping

Dr. Stephen Blythe
CSC10022, Fall 2019

The `while` Statement

- Like an "if", but go "back" to the test after the dependent statement(s).

- For example (this is in a .js file, but could be part of a <script> instead):

```
var input = Number( prompt("Enter a positive value") );

while (input<0) // note: replace "while" with "if", and what do you get?

  input = Number( prompt("Please be more careful, and enter a positive value") );
```

- Take care *not* to put an "empty" semicolon immediately after the while!

- You may *not* "attach" an `else` to a `while`!

- Need more than one statement inside a while loop?

  - surround the statements with curly braces (just like with "if").

The `while` Statement ...

- `while` is an example of a "loop" in JavaScript

    - allows for repetitive execution of code

    - usually called "iterative" execution or just "*iteration*"

- Most common use of iteration: counting

    - for example:

    ```
    var counter=0;        // initialize counter
    while (counter<=10)   // test condition
    {
      document.write(counter+"<br>"); // loop "body"
      counter++;   // increment counter
    }
    ```

The `while` Statement ...

- Consider this while loop:

```
var loopStop = Number( prompt("How high should we go?") );
document.write("<table border=1><tr><th>i</th><th>i<sup>2</sup></th></tr>");
var index=1;        // loop control variable & initializer
while (index <=  loopStop)   // loop test condition
{
 //loop body
 document.write("<tr><td>" + index + "</td><td>" + index*index + "</td></tr>");
 index++;   // loop updater
}
document.write("</table>");
```

Loop Components

- **<u>EVERY</u>** loop should have:

  - at least one *loop control variable*, used in each of the components

  - a *loop initialization*, gives initial value(s) to loop control variable(s)

    - may be more than one statement

    - usually found before the loop itself

  - a *loop test condition*, which dictates wether or not to stay in the loop

    - true means do the loop body again, false means do not do so

  - a *loop body,* which does the "work" each time through the loop

  - a *loop updater*, which changes the value of the control variable(s)

- Note the comments in the prior slides show some of these

Loop Components

- If you can't name _each_ of the components, your loop is not likely to work

- What happens if you skip/miss one?

  - no _loop body_:
    - the loop will (appear to) do nothing (although it still "iterates")
  - no _loop control variable_ :
    - no way to stop the loop
  - no _loop initialization_ :
    - loop may not start at right point, and may not stop
  - no _loop test condition_ :
    - likely will not load in browser (error)
  - no _loop updater_ :
    - loop will likely not stop

Infinite Loops

- Consider this while loop:

```
var loopStop = Number( prompt("How high should we go?") );
document.write("<table border=1><tr><th>i</th><th>i<sup>2</sup></th></tr>");
var index=1;         // loop control variable & initializer
while (index <=  loopStop)  // loop test condition
{
 //loop body
 document.write("<tr><td>" + index + "</td><td>" + index*index + "</td></tr>");
 index--;   // loop updater
}
document.write("</table>");
```

The for Loop

- Another loop statement is the `for` statement

- General form:

```
for ( <initializer>; <test condition>; <updater>)

    <loop body>
```

- Basic idea:

  1. do the code in the `<initializer>`

  2. check the `<test condition>` … if true, do the `<loop body>`

  3. do the code in the `<updater>`

  4. go back to step (2)

for Loop Example

- Consider this for loop (similar to previous `while` loop):

```
var loopStop = Number( prompt("How high should we go?") );
document.write("<table border=1><tr><th>i</th><th>i<sup>2</sup></th></tr>");
var index;
for (index=1; index <=  loopStop; index++)
{ // do not actually need these {}'s in this example. Why not?
  document.write("<tr><td>" + index + "</td><td>" + index*index + "</td></tr>");
}
document.write("</table>");
```

Loop Options

- Any `for` loop can be re-written as a `while` loop

- Any `while` loop can be re-written as a `for` loop

- Which one should you use?

    - entirely up to you. Experienced programmers gain an intuition as to which is easier to work with in different scenarios.

    - translation: practice for a **_LONG_** time, and you'll get the idea.

- You can put a loop inside of an if statement!

- You can put an if statement inside of a loop body!

- You can put a loop inside of another loop (inside of another loop …)

- basically, anywhere you can put a statement, you can put a loop