Loops

Topics
- The while Loop
- Program Versatility
  - Sentinel Values and Priming Reads
  - Checking User Input Using a while Loop
- Counter-Controlled (Definite) Repetition
- Event-Controlled (Indefinite) Repetition
- for Loops
- do-while Loops
- Choosing an Appropriate Loop
- Break and Continue Statements

Review: Repetition Structure
- A repetition structure allows the programmer to specify that an action is to be repeated while some condition remains true.
- There are three repetition structures in JavaScript, the while loop, the for loop, and the do-while loop.

The while Repetition Structure
while (condition) {
  statement(s)
}

- The braces are not required if the loop body contains only a single statement. However, they are a good idea and are required by the 104 Coding Standards.

Example
while (children > 0) {
  children = children - 1;
  cookies = cookies * 2;
}

Good Programming Practice
- Always place braces around the body of a while loop.
- Advantages:
  - Easier to read
  - Will not forget to add the braces if you go back and add a second statement to the loop body
  - Less likely to make a semantic error
- Indent the body of a while loop 2 to 3 spaces -- be consistent!
Another while Loop Example

- **Problem:** Write a program that calculates the average exam grade for a class of 10 students.
- What are the program inputs?
  - the exam grades
- What are the program outputs?
  - the average exam grade

The Pseudocode

```
<total> = 0
<grade_counter> = 1
While (<grade_counter> <= 10)
    Display "Enter a grade:"
    Read <grade>
    <total> = <total> + <grade>
    <grade_counter> = <grade_counter> + 1
End_while
<average> = <total> / 10
Display "Class average is: ", <average>
```

The Code

```
1. var counter, grade, total, average;
2. total = 0;
3. counter = 1;
4. while (counter <= 10)
5. {
6.   grade = prompt("Enter a grade: ");
7.   grade = parseInt(grade);
8.   total = total + grade;
9.   counter = counter + 1;
10. }
11. average = total / 10;
12. alert("Class average is ", average);
```

Versatile?

- How versatile is this program?
- It only works with class sizes of 10.
- We would like it to work with any class size.
- A better way:
  - Ask the user how many students are in the class.
  - Use that number in the condition of the while loop and when computing the average.

New Pseudocode

```
<total> = 0
<grade_counter> = 1
Display "Enter the number of students:"
Read <num_students>
While (<grade_counter> <= <num_students>)
    Display "Enter a grade:"
    Read <grade>
    <total> = <total> + <grade>
    <grade_counter> = <grade_counter> + 1
End_while
<average> = <total> / <num_students>
Display "Class average is: ", <average>
```

New Code

```
1. var numStudents, counter, grade, total, average;
2. total = 0;
3. counter = 1;
4. numStudents = prompt("Enter number of students: ");
5. numStudents = parseInt(numStudents);
6. while (counter <= numStudents)
7. {
8.   grade = prompt("Enter a grade: ");
9.   grade = parseInt(grade);
10. total = total + grade;
11. counter = counter + 1;
12. }
13. average = total / numStudents;
14. alert("Class average is: ", average);
```
Why Bother to Make It Easier?

- Why do we write programs?
  - So the user can perform some task
- The more versatile the program, the more difficult it is to write. BUT it is more useable.
- The more complex the task, the more difficult it is to write. But that is often what a user needs.
- Always consider the user first.

Using a Sentinel Value

- We could let the user keep entering grades and when he’s done enter some special value that signals us that he’s done.
- This special signal value is called a sentinel value.
- We have to make sure that the value we choose as the sentinel isn’t a legal value. For example, we can’t use 0 as the sentinel in our example as it is a legal value for an exam score.

The Priming Read

- When we use a sentinel value to control a while loop, we have to get the first value from the user before we encounter the loop so that it will be tested and the loop can be entered.
- This is known as a priming read.
- We have to give significant thought to the initialization of variables, the sentinel value, and getting into the loop.

New Pseudocode

```plaintext
<total> = 0
<grade_counter> = 1
Display "Enter a grade: "
Read <grade>
While (<grade> != -1)
    <total> = <total> + <grade>
    <grade_counter> = <grade_counter> + 1
    Display "Enter another grade: "
    Read <grade>
End_while
<average> = <total> / <grade_counter>
Display "Class average is: ", <average>
```

New Code

```javascript
var counter, grade, total, average;
var counter = 1;
grade = prompt("Enter a grade: ");
grade = parseInt(grade);
while (grade != -1)
{
    total = total + grade;
    counter = counter + 1;
    grade = prompt("Enter another grade: ");
    grade = parseInt(grade);
}
average = total / counter;
alert ("Class average is: "+ average);
```

Final Clean* code

```javascript
var counter; /* counts number of grades entered */
var grade; /* individual grade */
var total; /* total of all grades */
var average; /* average grade */
/* Initializations */
total = 0;
counter = 1;
grade = prompt("Enter a grade: ");
grade = parseInt(grade);
while (grade != -1)
{
    total = total + grade;
    counter = counter + 1;
    grade = prompt("Enter another grade: ");
    grade = parseInt(grade);
}
average = total / counter;
alert ("Class average is: "+ average);
```

*Follows course coding standards (continued)
**Final Clean Code**

```java
17. /* Get grades until user enters -1. Compute
18.    grade total and grade count */
19. while (grade != -1)
20. {
21.    total = total + grade;
22.    counter = counter + 1;
23.    grade = prompt("Enter another grade: ");
24. }
25. /* Compute and display the average grade */
26.    average = total / counter;
27.    alert("Class average is: " + average + ".");
```

**Using a while Loop to Check User Input**

```javascript
1. var number;
2. 3. number = prompt("Enter a positive number: ");
4. 5. number = parseFloat(number);
6. 7. while (number <= 0)
8.   {
9.    alert("That's incorrect. Try again.
10.   number = prompt("Enter a positive number: ");
11. 12. } 13. alert ("You entered: " + number);
```

**Counter-Controlled Repetition (Definite Repetition)**

- If it is known in advance exactly how many times a loop will execute, it is known as a **counter-controlled loop**.

```javascript
var i = 1;
while (i <= 10)
{
    alert ("i is " + i);
    i = i + 1;
}
```

**Event-Controlled Repetition (Indefinite Repetition)**

- If it is NOT known in advance exactly how many times a loop will execute, it is known as an **event-controlled loop**.

```javascript
sum = 0;
value = prompt("Enter a value: ");
while (value != -1)
{
    sum = sum + value;
    value = prompt("Enter a value: ");
}
```

**Event-Controlled Repetition**

- An event-controlled loop will terminate when some **event** occurs.
- The event may be the occurrence of a sentinel value, as in the previous example.
- There are other types of events that may occur, such as reaching the end of a data file.

**The 3 Parts of a Loop**

```javascript
var i = 1;
//count from 1 to 100
while (i < 101)
{
    alert("i is " + i);
    i = i + 1;
}
```
The for Loop Repetition Structure

- The for loop handles details of the counter-controlled loop "automatically".
- The initialization of the loop control variable, the termination condition test, and control variable modification are handled in the for loop structure.

```
for (i = 1; i < 101; i = i + 1)
{
    // initialization
    // modification
    // test
}
```

When Does a for Loop Initialize, Test and Modify?

- Just as with a while loop, a for loop initializes the loop control variable before beginning the first loop iteration,
- modifies the loop control variable at the very end of each iteration of the loop, and
- performs the loop termination test before each iteration of the loop.

- The for loop is easier to write and read for counter-controlled loops.

A for Loop That Counts From 0 to 9

```
for(i = 0; i < 10; i = i + 1)
{
    alert("i is " + i);
}
```

We Can Count Backwards, Too

```
for(i = 9; i >= 0; i = i - 1)
{
    alert("i is " + i);
}
```

We Can Count By 2’s ... or 7’s ... or Whatever

```
for(i = 0; i < 10; i = i + 2)
{
    alert("i is " + i);
}
```

The do-while Repetition Structure

```
do
{
    statement(s)
}
while (condition);
```

- The body of a do-while is ALWAYS executed at least once. Is this true of a while loop? What about a for loop?
Example

do {
  num = prompt("Enter a positive number: ");
  num = parseInt(num);
  if (num <= 0) {
    alert("That is not positive. Try again.");
  }
} while (num <= 0);

An Equivalent while Loop

num = prompt("Enter a positive number: ");
num = parseInt(num);
while (num <= 0) {
  alert("That is not positive. Try again.");
  num = prompt("Enter a positive number: ");
  num = parseInt(num);
}

Notice that using a while loop in this case requires a priming read.

An Equivalent for Loop

num = prompt("Enter a positive number: ");
num = parseInt(num);
for ( ; num <= 0; ) {
  alert("That is not positive. Try again.");
  num = prompt("Enter a positive number: ");
  num = parseInt(num);
} • A for loop is a very awkward choice here because the loop is event-controlled.

So, Which Type of Loop Should I Use?

• Use a for loop for counter-controlled repetition.
• Use a while or do-while loop for event-controlled repetition.
  • Use a do-while loop when the loop must execute at least one time.
  • Use a while loop when it is possible that the loop may never execute.

Nested Loops

• Loops may be nested (embedded) inside of each other.
• Actually, any control structure (sequence, selection, or repetition) may be nested inside of any other control structure.
• It is common to see nested for loops.

Nested for Loops

for (i = 1; i < 5; i = i + 1) {
  for (j = 1; j < 3; j = j + 1) {
    if (j % 2 == 0) {
      document.write("O");
    } else {
      document.write("X");
    }
  }
  document.write("\n");
}
The break Statement

- The break statement can be used in while, do-while, and for loops to cause premature exit of the loop.
- THIS IS NOT A RECOMMENDED CODING TECHNIQUE.

Example break in a for Loop

```javascript
var i;
for(i = 1; i < 10; i = i + 1)
{
  if(i == 5)
  {
    break;
  }
  document.write(i + " ");
}
document.write("Broke out of loop at i = " + i);
```

OUTPUT:

```
1 2 3 4
Broke out of loop at i = 5.
```

The continue Statement

- The continue statement can be used in while, do-while, and for loops.
- It causes the remaining statements in the body of the loop to be skipped for the current iteration of the loop.
- THIS IS NOT A RECOMMENDED CODING TECHNIQUE.

Example continue in a for Loop

```javascript
var i;
for(i = 1; i < 10; i = i + 1)
{
  if(i == 5)
  {
    continue;
  }
  document.write(i + " ");
}
document.write("Done.");
```

OUTPUT:

```
1 2 3 4 6 7 8 9
Done.
```