Chapter 17: Introduction to JavaScript

JavaScript is a programming language that allows one to design complex, interactive Web pages. The language is one of several scripting languages that permit a Web page to interact with the user. This book will include enough JavaScript to allow you to add considerable power to your experiments. However, a complete treatment is beyond the scope of this book. There are many books available that teach JavaScript, and if you like programming and enjoy these chapters, you will want to study these books. There are also links to on-line tutorials and books about JavaScript, which are included in the examples to this chapter.

Even if you do not plan to become a programmer, there are useful things that you can add to your experiments with a little JavaScript. For example, this chapter will develop a small script that you can use in your HTML experiments to accomplish the useful task of random assignment to conditions. In previous chapters, participants were assigned to conditions by clicking on their birthday or birth month, and horoscope was counterbalanced by a Latin Square. In this chapter, you will learn use a random number generator to assign participants to conditions.

JavaScript is a scripting language that should not be confused with the Java language, which is a distinct language. The intended goal of both languages is to support programs that will run on anyone's computer, whether they have a Windows PC or Mac.

JavaScript programs can be included in the HTML document, and the server delivers them to the remote computer along with the HTML. Because JavaScript runs on the client's (the participant's) machine, it frees the server from having to make computations. It also frees the participant from waiting for a remote computer to

compute something and send an answer. Thus, when JavaScript is delivered with the Web page, it runs on the participant's computer, not your server.

JavaScript is fairly new and still growing, but it is the oldest and most widely used scripting language of its type on the Web, so it is well worth learning. Ideally, JavaScript would run on Netscape and Explorer, and on Mac or PC in the same way. In practice, however, not all of the differences have yet been ironed out, so you must test your JavaScript program on several different browsers to make sure that your script behaves properly on them all; often you can find a way to make the program work on the major browsers. Because JavaScript was developed by Netscape, it tends to run better on Netscape than on early versions of Microsoft Internet Explorer.

Another consideration in the use of JavaScript (or Java, Shockwave, Active-X, or other advanced techniques) is that not all people are using browsers that support it (and some have it turned off). Therefore, any experiment that uses these techniques will lose some of its potential audience. Of these “extras,” JavaScript is the one available to more users than any other, followed closely by Java.

A. A Simple JavaScript

The HTML page in Figure 17.1, *Ch17_ex1.htm*, illustrates how to include a JavaScript program in a Web page.

Insert Figure 17.1 about here.

Figure 17.1. How to incorporate JavaScript in an HTML page. This is a listing of *Ch17_ex1.htm*. Load this example in your browser.

```
<HTML><HEAD><TITLE>First JavaScript Example</TITLE>
</HEAD><BODY>
<H3>The Following is from JavaScript</H3>
<P><PRE>

<SCRIPT LANGUAGE="JavaScript">
<!-- this comment hides the script from older browsers
// this is a comment. It has no effect.
/* This is also a comment. This one can
    stretch over several lines.  It also has no effect */

document.write("Hi! ")
document.write("<A HREF='Ch5_exp7.htm'>Click here</A>")

// JavaScript comment hides the end of HTML comment  -->
</SCRIPT>

</PRE></BODY></HTML>
```

There are four things in this example that you will likely use in all of your JavaScripts. First, note the `<SCRIPT LANGUAGE="JavaScript"> </SCRIPT>` tags. These identify that the material between them is JavaScript. Second, notice the two lines that have the HTML comment tag, i.e., `<! . . .>`. All of the JavaScript is hidden from older browsers, because it is within this giant HTML comment. It will soon become unnecessary to hide the JavaScript, as more people shift to modern browsers. Third, anything that follows `//` on a line is a JavaScript comment, and does not affect JavaScript. Note that a JavaScript comment hides the close of the HTML comment from JavaScript. Fourth, to include a comment that covers several lines, begin it with `/*` and end it with `*/`.

The `//` comment can begin on a line, or it can follow an actual JavaScript statement, as on the line that writes *Hi!* The material that precedes the `//` is not a comment. It is a good idea to include comments in your scripts to remind yourself how your script works or to help others understand what you are doing. The computer, of course, does not read or interpret your comments.

There is only one JavaScript statement illustrated in the example, `document.write()`. As you might guess, this expression (called a *method*) writes the contents within the quotes in the parentheses to the document. In the first case it writes *Hi!,* and in the second instance, it places a link in the document. Because the document is HTML, you can put HTML tags inside the quotes in the `document.write()` command, and the browser will interpret and display them correctly. The HTML will not be displayed literally, but will become part of the Web page itself. As you will see in Chapter 18, this allows one to dynamically change the contents of a Web page.

In JavaScript, each statement is placed on a separate line. Multiple statements can be placed on the same line if they are separated by semicolons. JavaScript commands and key words are case sensitive. Try changing `document.write()` to `Document.Write()`. You will get an error message. Because HTML is unaffected by capitalization, it may be difficult to get used to JavaScript's sensitivity to case.

B. Illustration of Random Numbers

The script in *Ch17_ex2.htm*, listed in Figure 17.2, illustrates the use of variables, mathematical computations, and random numbers.

Insert Figure 17.2 about here.

Load the page, *Ch17_ex2.htm*, in your browser. Then push the *Reload* button (or *Reload* from the **View** menu). You will see that the values of *x* and *y* are not only different from each other, they are also different every time you reload the page. JavaScript has a package of mathematical functions available. These are collectively known as the *Math Object*, and the functions are called *methods*. `Math.random()` produces a random number that is uniformly distributed between 0 and 1.

The first three lines of the program establish that *x*, *y*, and *z* are numerical variables and set their initial values to zero. In JavaScript, unlike BASIC, the symbols *x*, *y*, and *z* might refer to numbers or strings (sequences of letters and numbers) such as "Hi", "2-10-97", or "137.157.14". When strings are added, they are concatenated. For example `"HiThere" = "Hi" + "There"`. When numbers are added, their numerical values are added, 4 = 2 + 2; however if the variables were strings, "2" + "2" is "22". If you see such arithmetic in one of your scripts, and you did not intend it, you should probably declare the variable with the `var` expression.

Figure 17.2. Load this example, *Ch17_ex2 .htm*, in your browser. Each time you reload, the values of *x* and *y* will change, but *z* will always be the sum of *x* and *y*.

```
<HTML><HEAD><TITLE>Illustrates math.random()</TITLE></HEAD><BODY>
<H3>Random Numbers. </H3>
<H4>Reload this program--it will be different each time.</H4>
<P><PRE>

<SCRIPT LANGUAGE="JavaScript">
<!-- this HTML comment hides the script

var x=0   // these lines establish the variables
var y=0
var z =0
// The following code illustrates use of Math.random()
// Math.random() produces a pseudo random number, uniform (0,1)

x = Math.random()
y = Math.random()
z = x + y     // adds x and y and puts sum in z

document.writeln("x = "+x+" y = "+y+" z = "+z)

//  JavaScript comment hides end of HTML comment  -->
</SCRIPT>
</PRE></BODY></HTML>
```

If you are unfamiliar with computer programming, it helps to read this statement,

```
z = x + y
```

from right to left. The statement is sometimes called an "assignment" statement, because it adds *x* to *y* and assigns the resulting value to the variable, *z*. The assignment statement should not be misunderstood as an algebraic equality--it is not! For example, consider the statement `y = y + 2`. If `y = 1` before the statement, then after the statement, y is equal to 3; if y had been zero, then it is now 2. The computer statement `y = y + 1` means to add one to the value of *y* and store the result in *y*. In algebra, the statement is certainly a puzzle. Unlike algebra, the JavaScript statement `z = x + y` is *not* the same as `y = z - x`. In the former case, z is changed; in the latter case, y is changed.

The expression, `y = y + 2`, can also be written `y += 2`; the expression `y = y + 1` can also be written `y++`.

Unlike some programming languages and unlike HTML, JavaScript variables are case-sensitive. That means that `x` is different from `X`, and `cow` is different from `Cow`.

The arithmetic operations of addition, subtraction, multiplication and division are represented by the symbols, `+, -, *,` and `/`. Try replacing the `+` with `*` or `/` to see different computations in this example. Operations that are enclosed in parentheses are performed first. For example, `x + y/z` will produce a different result from `(x + y)/z`. To raise *x* to the *y* power requires the power function in the Math Object. The expression is `z = Math.pow(x,y)`, which computes x^y. You will learn more about the Math methods in Chapter 19.

C. Random Assignment to Conditions using JavaScript

This section develops a simple JavaScript to assign participants to one of two or more conditions. This assignment will occur randomly in a way that no one can predict.

Suppose there are two experimental groups and the experiments are in files *numbersA.htm* and *numbersB.htm*. Suppose you want half of the subjects to complete *numbersA.htm* and half to complete *numbersB.htm*. The script in Figure 17.3, *Ch17_ex3.htm*, will accomplish that. Insert Figure 17.3 about here.

Figure 17.3. JavaScript to randomly assign participants to two conditions, *Ch17_ex3.htm*. Reload this example several times and click on the link to verify that it behaves differently on different occasions of being loaded.

```
<HTML><HEAD><TITLE>RANDOM ASSIGNMENT TO CONDITIONS</TITLE>
</HEAD><BODY>
<H3>RANDOM ASSIGNMENT TO TWO CONDITIONS</H3>
<H4>Reload this program. Reload each time to see different actions.</H4>
<P>
<PRE>

<SCRIPT LANGUAGE="JavaScript">
<!-- this HTML comment hides the script from older browsers

var x=0
//                if x > .5, subjects get A, otherwise B
x = Math.random()
if (x > .5)
     {document.write("<A HREF='numbersA.htm'>Click to continue</A>")}
else {document.write("<A HREF='numbersB.htm'>Click to continue</A>")}

//  JavaScript comment hides the end of HTML comment  -->
</SCRIPT>
</PRE></BODY></HTML>
```

The script in *Ch17_ex3.htm* uses a new idea, the *if* branch. The general expression is as follows:

```
If (Boolean expression)
     {statements; statements }
else {statements;
        statements}
```

If the Boolean (logical) expression is `true`, then the first block of statements in braces will be executed; if the statement is `false`, then the second block of statements will be executed. In the example of *Ch17_ex3.htm*, there is only one statement in each block. The only difference is that the `<A HREF>` tag points to a different file when `x > .5` from that when `x <= .5`. If the random number generator works properly, then subjects will be assigned equally likely in either direction.

Because instances of a random sequence rarely come out evenly, do not expect an exactly equal number of people in each condition. The example, *Ch17_ex4.htm*, shows how to assign to three conditions, which are the three conditions for obtaining responses for the value of the St. Petersburg gamble. If there were four between-subjects conditions, you could use two random variables, *x* and *y*, to assign people to conditions. Alternatively, you could assign to group 1 if `x < .25`, to group 2 if `.25 <= x < .5`, to group 3 if `.5 <= x < .75`, and to group 4 otherwise.

Note also in the example that there are nested quote marks. When quotes are nested, use single quotes to distinguish matching quotes. In this case, the outer quotes indicate that the material inside is a string literal to be printed in the document. The inner quotes show the filenames of the `<A HREF="URL">` tag.

A Boolean expression is either `true` or `false`. Here is a list of the Boolean relations: `(<, >, >=, <=, ==, !=);` these refer to less than, greater than, greater than

or equal, less than or equal, equal, and not equal, respectively. For example, if `a == 3` and `b == 4`, it is `true` that `a < b`, and it is `true` that `a != b`. However, it is `false` that `a >= b` or that `a == b`. Note the use of double equal (`==`) for logical equal. The Boolean operators of `AND`, `OR`, and `NEGATION` are represented by `&`, `|`, and `!`. For example,

```
if (x < .5 & y < .5) {document.write("x and y are both less than .5")}
```

The above statement will write its string only when both conditions are true.

D. Selecting a Random Number from 1 to *n*

Consider the expression,

```
y = Math.floor(n*Math.random()) +1
```
(17.1)

The `Math.floor()` method truncates a number to the next lower integer. For example, `Math.floor(3.1)` is 3, and `Math.floor(3.99)` is also 3. If *n* is an integer, `n*Math.random()` will be uniformly distributed on (0, *n*). Therefore, `Math.floor(n*Math.random())`, yields integers from 0 to *n* - 1, with equal probability. Adding 1, the expression gives integers from 1 to *n* with equal likelihood. This expression is at the heart of *Ch17_ex6.htm*. In *Ch17_ex6.htm*, there are six conditions, but it should be clear that it could be changed to any number. Expression 17.1 is also used in Examples 7 and 8 of Chapter 17 to randomly select the outcome of a rolled die. Examples 7 and 8 also illustrate how to manipulate which of several images will be displayed from JavaScript.

E. Review of JavaScript Fundamentals

`<SCRIPT></SCRIPT>`	HTML tags indicating script. `<SCRIPT LANGUAGE="JavaScript">`
`//` `/*  */`	JavaScript one line comment. JavaScript multiple line comment
`<!  anything...>`	HTML comment, used to hide JavaScript from older browsers
`document.write()`	Writes to the document
`Var x = 1` `Var y = "goodBye"`	Variable initialization and initial numerical assignment. y initialized to a string variable and assigned a string.
`Math.random()`	Random number, uniformly distributed between 0 and 1
`Math.pow(x,y)`	Computes *x* to the power of *y*.
`+,-,*,/`	Addition, subtraction, multiplication, division
`z = x + y`	Computation and assignment (right to left)

`if (`*`Boolean`*`)`
`{`*`statements`*`}`
`else {`*`statements`*`}`
	Logical branching. If the Boolean condition is true, executes the first block of statements, otherwise executes the second.

`Math.floor(x)`	Truncates x to the largest integer less than x. `Math.floor(2.9)` is 2.

F. Exercises

1. Try the following JavaScript, and see what it does:

```
var x = "2"
var y = "3"
```

```
z = x + y
document.write("x= "+x+" y = "+ y+ "z = "+z)
```

2. Now remove the quotes in the above script; next try changing the "`+`" to a "`-`", `*`, `or /`. Try adding parentheses to evaluate the following:

```
z = x + y/x
z = (x + y)/x
z = x/y + y/x
```

3. To get power functions, you can use the following:

`z = Math.pow(x, y)` (this assigns, $z = x^y$)

4. Try the following to calculate the area (A) of a circle with radius, r.

```
A = Math.PI*Math.pow(r, 2)
```

(`Math.PI` is the value of π; note that `PI` is in ALL CAPS.)

Develop a script that will calculate the area of a pizza whose diameter is 12 inches. It should print the result.

5. Make the JavaScript for a page that assigns subjects to one of four conditions, based on two random variables x and y. It should assign to condition 4 if x > .5 and y >.5. Also do the same, using one random variable, x.
6. Alter *Ch17_ex6.htm* so that it randomly assigns people to one of five conditions. Delete the printing of the random integer. Also, change the filenames to five variations of the Heider experiment, heider1.htm to heider5.htm. If you understand how to modify this example, you can handle random assignment to conditions.