In this section, let's build something useful, based on what we have learned so far.

Let's create a drop-down list controls and use them for decision-making in any JavaScript code.

You can add this to any HTML page you already have so to allow users to type their search words and do the search.

All steps are described as in the following:

1- Start a new html boilerplate
2- In the <body> </body> add:

```
<body>
   <p>
     Search this site:
     <input type="text" id="txtlookfor" autofocus />
     <input type="button" onclick="search()" value="Go" />
   </p>
</body>
```

3- Save the file as SiteSearch.htm in your Intro JavaScript folder.
4- View the page:

The controls don't do anything yet, because we need JavaScript code to actually perform the search.

5- Add <script> </script> tag to the head section.

So now, the code in the head section of your SiteSearch.htm page should look something like this:

```
<head>
  :
  <title>Document</title>
  <script>

  </script>
</head>
```

6- Add a function:

```
<script>
     //Search a specific site rather than the entire Web.
     function search() {}
   </script>
```

Before we write that code, note the following steps:

- Specify the domain name of the site to be searched (your website).
- Get whatever the user typed into the textbox.
- If the length of that text is greater than zero (meaning the user didn't leave the box blank), create a URL to perform the search, and put it in the Address bar.
- Otherwise, if the user left the textbox blank, show an alert asking the user to type some text in the box.

Let's start by specifying the domain to search. In real life, this would be the domain name of your own site or whatever site you want to search. If you don't have a large site out on the public Web right now, you can use Wikipedia.org or some other large site to practice with.

7- Make sure the cursor is between the opening and closing curly braces for the search() function.
8- Feel free to replace wikipedia.org with your own domain name or the domain name of some other site to search.

```
<script>
  //Search a specific site rather than the entire Web.
  function search() {
    //Replace sample domain name below with your own domain name.
    var site = 'wikipedia.org';
  }
</script>
```

9- Then add:

```
    var lookfor = document.getElementById("txtlookfor").value;
```

This line creates a variable named look for which will store whatever text the user types into the Search textbox.

Remember
In JavaScript terminology, the ***value*** of a textbox is whatever text the user typed into the textbox.

Doing the Search

10- Add a conditional statement below

```
    if () {}
    else {}
```

JavaScript includes a **.length** property that you can use with strings to determine their length. That length is the number of characters in the string. If the user typed nothing into the box, then the length of the lookfor string will be 0 (zero). Otherwise, it will be some number greater than zero. So to determine if the **lookfor** variable contains any text, we just have to check for **lookfor.length > 0** (greater than zero). So add the following code:

```
        if (lookfor.length > 0) {
```

If the length of **lookfor** is greater than zero, we want to do the search. To use Google to do the search, the code will need to send to Google a URL that looks something like this:

http://www.google.com/search?q=searchwords site:domain

The code will have to substitute whatever the user typed in the textbox for search words. And it will also have to substitute the domain to be searched for domain. Thankfully, we have that information stored in variables named **lookfor** and site. So we just have to use a common programming practice called string concatenation to build a URL.

String Concatenation

If we have two strings, Hello and World, and concatenate them together, we end up with one string containing HelloWorld. Not really something to be afraid of.

To concatenate strings in JavaScript, we use the JavaScript **+** operator. When used with strings (rather than numbers), the + operator joins multiple strings into a single string. For

instance **"Hello"+"World"** creates the string **"HelloWorld"**. So our script needs to use some **+** operators to string together a part that's always the same with the information from the variables in our script.

11- Now add the following code

```
var query="http://www.google.com/search?q=" + encodeURIComponent(lookfor) + "
site:" + site;
```

The **encodeURIComponent()** function is actually just an optional convenience that makes the text that the user typed more palatable as a URL. You may have noticed that URLs never contain certain characters, such as spaces. The lack of blank spaces and some other characters makes the URLs more compatible with older computers that had limited character sets or didn't treat spaces the same as more modern computers do.

The **encodeURIComponent()** function is built into JavaScript. It takes whatever string you hand it and hands back the same text modified (if necessary) to work better as part of a URL. Since we can't control exactly what the user types into the search textbox, we use the encodeURIComponent() to conveniently convert that to something that we know will work in a URL.

After adding the massaged **lookfor** variable, the code adds the literal text "site:". We need to tack this onto the end so Google will search only a single site rather than the entire Web. And after that, we need to provide the domain name of the site to search. We've already stored that in a variable named site. So

all we have to do is tack that variable's value after the literal "**site**:" text, as indicated by the last term, site;.

So, let's say the user types hello world into the textbox. And for the sake of example, let's say you left the site variable pointing to Wikipedia.org. After the line of code that creates the query variable is executed, the query variable contains this:

http://www.google.com/search?q=hello%20world site:wikipedia.org

That's what should be to sent to Google to have it search the Wikipedia.org site for hello world. The %20 represents a space. The encodeURIComponent() function automatically (and correctly) converted the space to %20 for us.

12- Add the following code

```
            location.href = query;
```

The **location** object in JavaScript represents the Address bar. And the **href** property is its http reference (basically, the URL that's showing in it). By setting the value of the property to the contents of the query variable, we change the URL to the contents of the variable, and that's what performs the search.

However, we're not completely done yet. Earlier, we put in an if statement that says if the user typed something into the search box, then do the search. We haven't yet said what to do if the user didn't type some text into the search box. In this case, the variable **lookfor** has a length of zero.

To deal with that situation, we'll need to use our else portion of the if statement, which we've already added to our code. So let's just fill in that part with an alert message.

13- So add the following code to the else { } section.

```
    } else {
        //If textbox was empty, show an alert.
        alert('Please type the word or words for which you want to search.');
    }
```

14- Now try the program.

The whole program is described below:

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <title>Document Search</title>
    <script>
      //Search a specific site rather than the entire Web.
```

```
      function search() {
        //Replace sample domain name below with your own domain name.
        var site = 'wikipedia.org';
        var lookfor = document.getElementById('txtlookfor').value;

        if (lookfor.length > 0) {
          var query =
            'http://www.google.com/search?q=' +
            encodeURIComponent(lookfor) +
            ' site:' +
            site;
          location.href = query;
        } else {
          //If textbox was empty, show an alert.
          alert('Please type the word or words for which you want to search.');
        }
      }
    </script>
  </head>
  <body>
    <p>
      Search this site:
      <input type="text" id="txtlookfor" autofocus />
      <input type="button" onclick="search()" value="Go" />
    </p>
  </body>
</html>
```

Adding a Drop-Down List

In the Input Box (#txtlookfor) add:

```
  <body>
    <p>
      Search this site:
      <input type="text" id="txtlookfor" autofocus />
      <!-- Show drop-down list -->
       Choose engine:
      <select id="dropdown">
        <option value="1">Bing</option>
        <option value="2">Google</option>
      </select>
      <input type="button" onclick="search()" value="Go" />
    </p>
  </body>
```

If the code is right, refreshing your page you should see:

By adding "**selected"** to the option, you make it default:

```
        <option value="2" selected>Google</option>
```

Test the program again, and see the effect.

Searching the Chosen Engine

15- This task requires another conditional statement just before **location.href** as below:

```
if (lookfor.length > 0) {
          var query =
            'http://www.google.com/search?q=' +
            encodeURIComponent(lookfor) +
            ' site:' +
            site;
          var ddchoice = document.getElementById('dropdown').value;
          if (ddchoice == 1) {
            var query =
              'http://www.bing.com/search?q=' +
              encodeURIComponent(lookfor) +
              ' site:' +
              site;
          } else {
            var query =
              'http://www.google.com/search?q=' +
              encodeURIComponent(lookfor) +
              ' site:' +
              site;
          }
          location.href = query;
        } else {
```

The program is complete and is time to test it: