PROG8020

Week 4b

JavaScript Source Files

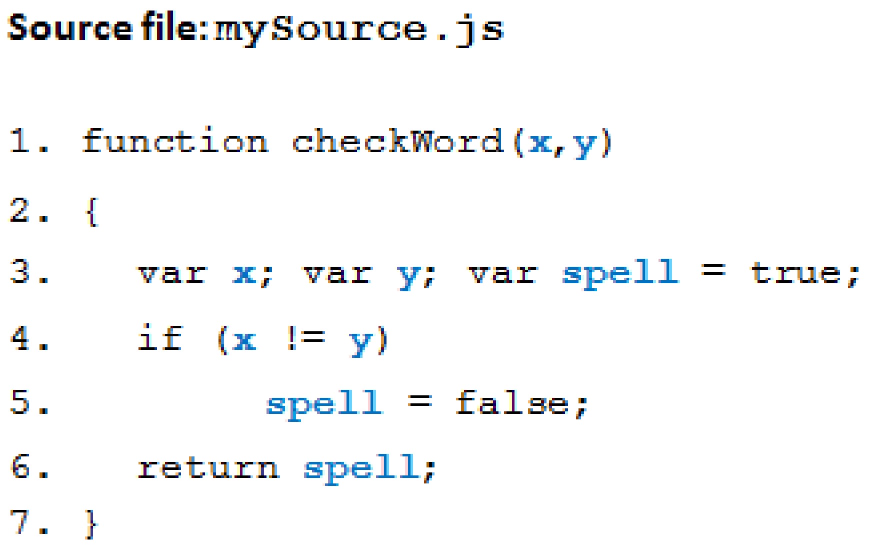
Creating a Source File

- Comparable to an external css file
- Is a simple text file containing JavaScript functions
- Use the `.js` extension
- Use to create a library of functions
- Link to the file in either `<head>` or `<body>`:

```
<script type="text/javascript" src="filename.js"></script>
```

Creating a Source File (2)

This example creates a source file which we will call `mySource.js`. It contains one function that will check whether a word is spelled correctly.



Source file: `mySource.js`

```
function checkWord(x,y)
{
  var x; var y; var spell = true;
  if (x != y)
        spell = false;
  return spell;
}
```

Using a Source File

- Checking if user enters a valid code:

Web page file:

```
<html><head>
<title>Example 7.15</title>
<script type="text/javascript" src="mySource.js"></script>
<script>
  function shipIt()
  {
       var shipCode = "FREEBIE";
       var userCode = prompt("Enter your code:");
       if (checkWord(shipCode, userCode))
            document.getElementById('result').innerHTML = ↵
                            ("Shipping is free!");
       else
            document.getElementById('result').innerHTML = ↵
                            ("Sorry, your code is not valid");
  }
  </script></head>
  <body>
       <input type ="button" onclick="shipIt()" value = "Enter ↵
                            free shipping code"><br />
       <h3><span id = "result">&nbsp;</span></h3>
  </body></html>
```

What happens?

- `shipIt()` is called when the button is clicked.
  - it initializes a variable with the value of the free shipping code (`"FREEBIE"`)
  - then prompts the user to enter his or her code
  - then it calls a function named `checkWord()`.
- If the `checkWord()` function returns `true`, line 10 is executed.
- If `checkWord()` returns `false`, the `else` statement on line 12 is executed.
- When `checkWord()` is called, the computer will look through the `<head>` section for a function named `checkWord()`.
- Since none is found, it goes to the `.js` file and looks for `checkWord()` there.
- The variables, **`shipCode`** and **`userCode`**, are passed into the parameters of `checkWord()` which are **`x`** and **`y`**.
- The function evaluates **`x`** and **`y`**, checking to see if they match.
  - If they don't, the variable **`spell`** is changed to `false`
  - if they do, **`spell`** remains as it was initialized, `true`.
  - This is returned to the `shipIt()` function and the program continues.

Regular Expressions

Regular Expressions

Regular expressions are patterns used to match character combinations in strings also, called objects

- Similar to Input mask within Access and VB
- Controls validation
- Expression always start with and end with:

/^...$/

\ ..... toggle between literal and special characters
^ ..... beginning of a string
$ ..... ending of a string

Regular Expressions (2)

\d ..... any digit 0 through 9 (same as [0-9])
\D ..... any non-digit
\f ..... form feed
\n.....new line
\r.... Carriage return
\s ..... any single white space character (same as [\f\n\r\t\v])
\S.... .. any single ***non***-white space character
[abcde] ..... a character set that matches any one of the enclosed characters
[ABCDE] ..... Same for upperCase
[^abcde] ..... one that does not match the enclosed characters
[a-e] ..... one that matches the range of enclosed characters
[A-E] ..... matches the range of enclosed characters
x|y.......either x or y
( ) ......a grouping which is also stored for later use

Regular Expressions (3)

*.......zero or more times
+......one or more times
?.......zero or one time
{n} ..... exactly *n* occurrences of the previous character
{n,}.....at least *n* occurrences of the previous character
{n,m}...between *n* and *m* occurrences of the previous character

Regular expression Modifiers

i.... Search without case-sensitivity

More examples:

http://www.devguru.com/Technologies/ecmaScript/quickref/regexp_special_characters.html

Regular Expressions (4)

To format and validate a postal code:

var rePostal = /^[A-Z][0-9][A-Z]\s[0-9][A-Z][0-9]$/

This regular expression looks for a string that has:

- beginning /^
- [A-Z] uppercase A-Z
- [0-9] digit 0-9
- [A-Z] uppercase A-Z
- \s a single white space character
- [0-9] digit 0-9
- [A-Z] uppercase A-Z
- [0-9] digit 0-9
- ending $/

Regular Expressions (5)

To validate a phone number:

var rePhone = /^\(?(\d{3})\)?[\.\-\/\s]?(\d{3})[\.\-\/\s]?(\d{4})$/ ;

This regular expression looks for a string that has:

- beginning /^
- an optional left parenthesis \(?
- 3 digits (\d{3})
- an optional right parenthesis \)?
- an optional period, dash, forward slash, or space [\.\-\/\s]?
- 3 digits (\d{3})
- an optional period, dash, forward slash, or space [\.\-\/\s]?
- 4 digits (\d{4})
- ending $/

Regular Expressions (6)

validPhone = rePhone.test(document.myForm.phone.value);

The test() method performs the regular expression stored in rePhone on document.myForm.phone.value

If the pattern isn’t found validPhone will be set to null (false)

Otherwise, validPhone will be the contents of phone number value therefore (true)

Regular Expressions (7)

The use of ! (not) in the condition is like saying

“ if editmask test on phone is not True”

```
if (!rePhone.test(document.myForm.phone.value))
{
   code for error handling
}
```

Regular Expressions (8)

To test the expression only if the user enters data in the field (i.e. field is optional, but if user enters any data it must match the expression)

```
if  (document.myForm.phone.value != “” &&
    !rePhone.test(document.myForm.phone.value)
{
   code for error handling
}
```

Regular Expressions (9)

To ignore case in expressions you may either:

a) Tack the regular expression modifier to ignore case onto the end of the expression

```
rePostal = /^[A-Z][0-9][A-Z]\s[0-9][A-Z][0-9]$/i
```

b) Use the toUpperCase() function to convert what is being tested to upper case and check for uppercase alpha

```
rePostal.test(document.myForm.postalCode.value.toUpperCase());
```

Working with Form properties

- Use Form properties to specify where to send the form data and how to send it

- The ACTION property identifies the CGI script that will process your form

```
<form action="URL">
```

Example

```
<form action="#">
```

Working with Form properties

- The METHOD property controls how the web browser will send the data

  <form method="GET">

  or

  <form method="POST">

- We'll use POST
  - generally considered more flexible and safer

Thank You!