Experiment No. 2

Problem Definition: 2. a) Develop and demonstrate, using Javascript , an XHTML document that collects the USN (the valid format is: A digit from 1 to 4 followed by two upper-case characters followed by two digits followed by two upper-case characters followed by three digits; no embedded spaces allowed) of the user. Event handler must be included for the form element that collects this information to validate the input. Messages in the alert windows must be produced when errors are detected.

b) Modify the above program to get the current semester also (restricted to be a number from 1 to 8)

CSE@GIT

Objectives of the Experiment:

To demonstrate the use of JavaScript and XHTML

To develop an understanding of JavaScript syntax

CSE@GIT

Need of the Experiment

- JavaScript allows you to build interactive websites.
- JavaScript has become an essential web technology along with HTML and CSS, as most browsers implement JavaScript.
- For web development, as a front-end developer or on using JavaScript for backend development.
- JavaScript usage has now extended to mobile app development, desktop app development, and game development.

Theoretical Background of the Experiment

JavaScript

- JavaScript is an interpreted, scripting language, designed to add interactivity to HTML pages
- It is used in Web pages to improve the design, validate forms, detect browsers, create cookies
- Most popular scripting language on the internet, and works on all major browsers like Firefox, Chrome, Opera, IE
- It is usually embedded directly into HTML pages
- Can be used without purchasing a license

Theoretical Background of the Experiment

JavaScript

- JavaScript created by Netscape
- Joint Development with Sun Microsystems in 1995
- JScript created by Microsoft
- IE and Netscape renderings are slightly different
- Standardized by European Computer Manufacturers Association (ECMA) - http://www.ecma-international. org/publications /standards/Ecma-262.htm

Theoretical Background of the Experiment

Uses of JavaScript

- Provide alternative to server-side programming , Servers - often overloaded , Client processing - quicker reaction time
- JavaScript can work with forms
- Event-Driven Computation - a JavaScript program could validate data in a form before it is submitted to a server
- JavaScript can interact with the internal model of the web page (Document Object Model)
- JavaScript is used to provide more complex user interface than plain forms with HTML/CSS can provide

JavaScript – General Format - Directly embedded

```
<!doctype ...>
<html>
 <head>
  <title> Name of web page </title>
  <script type="text/javascript">
  ...script goes here
  </script>
 </head>
 <body>
 ...page body here: text, forms, tables
 ...more JavaScript if needed
 </body>
</html>
```

JavaScript – General Format
Indirect reference

```
<!doctype ...>
<html>
 <head>
  <title> Name of web page </title>
  <script type="text/javascript" src="tst_number.js"/ />
 </head>
 <body>
 ...page body here: text, forms, tables
 ...more JavaScript if needed
 </body>
</html>
```

JavaScript Basic Examples

```
<script>
  document.write("Hello World!")
</script>
```

⇐ format text with HTML code - heading

Example

```
<script>
 x="Hello World!"
 document.write(x)
</script>

<script>
 x="World"
 document.write("Hello " +x)
</script>
```

JavaScript Popup Boxes - alert

- Alert Box
  - An alert box is used if you want to make sure information comes through to the user.
  - When an alert box pops up, the user will have to click "OK" to proceed.

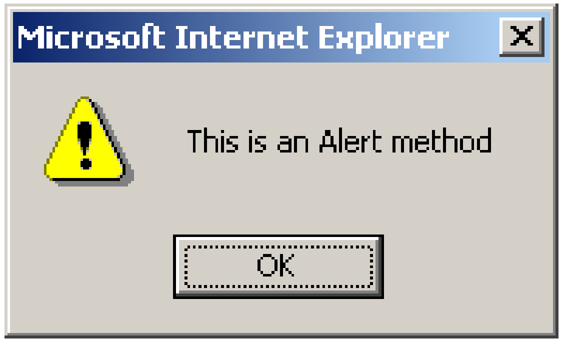
```
<script>
  alert("Hello World!")
</script>
```

JavaScript Popup Boxes - confirm

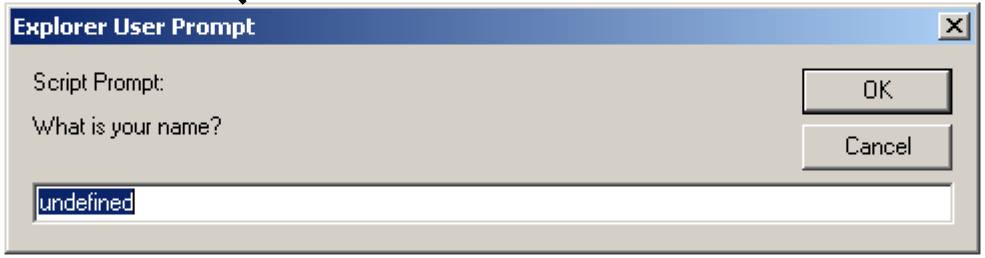
- Confirm Box
  - A confirm box is used if you want the user to verify or accept something.
  - When a confirm box pops up, the user will have to click either "OK" or "Cancel" to proceed.
  - If the user clicks "OK", the box returns true. If the user clicks "Cancel", the box returns false.

alert(), confirm(), and prompt()

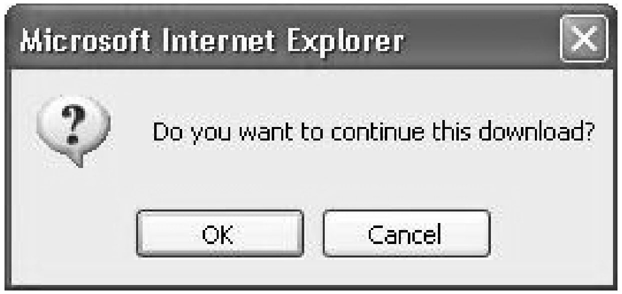
```
<script type="text/javascript">
   alert("This is an Alert method");
   confirm("Do you want to continue
this download?");
   prompt("What is your name?");
   prompt("Who am I?","Jackie Chan");
</script>
```



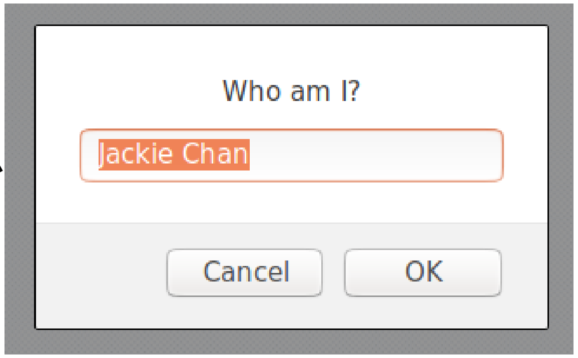
This is an Alert method



Script Prompt:
What is your name?
undefined



Do you want to continue this download?



Who am I?
Jackie Chan

promt()

- Return valut of promt() is the value enter in text area
- On cliciking **OK** or pressing Enter , the value can be collected in a vairable

```
returnValue = promt("Enter a number");
```

JS Examples -1

y =20 x + 12 ,
if x=3 then, what will be the value of y ?

```
<script>
  x=3
  y=20*x+12
  alert(y)
</script>
```

Examples -2

```
<script>
  s1=12
  s2=28
  total=s1+s2
  document.write("sum of two nos.: "+total)
</script>
```

Statements and Primitive Types

- Statements can be terminated with a semicolon
- But, the interpreter will insert the semicolon if missing at the end of a line
- Can be a problem:

```
return
x;
```

- If a statement must be continued to a new line, make sure that the first line does not make a complete statement by itself
- Five primitive types – Number , String , Boolean , Undefined , Null
- Date object

Declaring variable

- Dynamically typed

```
var counter,
index,
pi = 3.14159265,
quarterback = "Elway",
stop_flag = true;
```

String Catenation +

+
- Types automatically converted to string
- Implicit , Explicit type conversion

```
document.write( "<br/> counter = " + counter );
document.write( "<br/> index = " + index )
document.write( "<br/> pi = " + pi );
document.write( "<hr/> quarterback = " + quarterback );
document.write( " stop_flag = " + stop_flag)
```

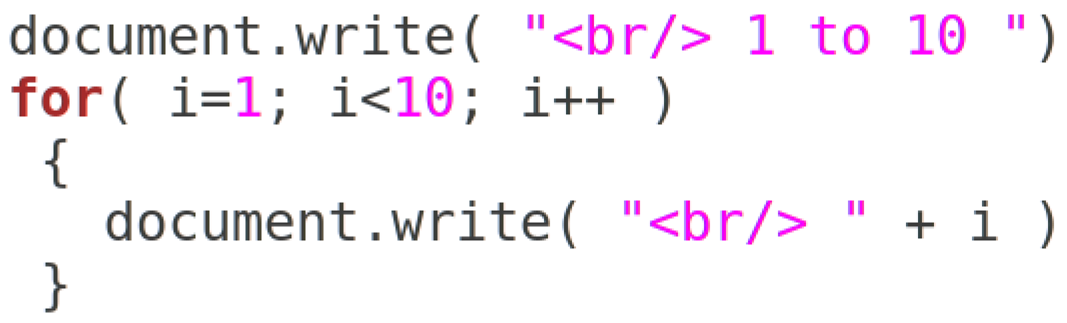
Conditional Statements

- Similar to C/C++/Java
- if statement
- if…else statement
- if…else if….else statement
- switch statement

```
x=3
if( x < 0 )
 {
     alert (x + " is negative")
 }
else
 {
     alert (x + " is positive")
 }
```

Looping Statements

- Loop statements in JavaScript are similar to those in C/C++/Java
- while
- for
- do while



```
document.write( "<br/> 1 to 10 ")
for( i=1; i<10; i++ )
 {
   document.write( "<br/> " + i )
 }
```

Looping Statements – Print prime numbers

- Prime Number - A number that is divisible only by itself and 1
- Using looping statement find and print prime numbers from 2 to 100

Function and Function call

- Functions are objects in JavaScript

```
function function_name(optional-formal-parameters)
 {
   return value;
 }



rvalue = function_name(parmeters);
```

Function and Function call

- Functions are objects in JavaScript

```
function fun()
 {
    document.write( " <hr/> <br/> This surely is fun! <br/>");
 }

ref_fun = fun; // Now, ref_fun refers to the fun object

fun(); // A call to fun

ref_fun(); // Also a call to fun
```

Regular Expressions

- Regular expressions are used to specify patterns in strings
- JavaScript provides two methods to use regular expressions in pattern matching
    - String methods
    - RegExp objects
- A literal regular expression pattern is indicated by enclosing the pattern in slashes
- The search method **return**s the position of a match, if found,
  or
  -1 if no match was found
- stringVaraible.search()

Regular Expressions

- What would be the output?

```
var str = "Rabbits are furry";
var position = str.search(/bits/);
if (position > 0)
  {
    document.write("'bits' appears in position", position );
    document.write("<br />");
  }
else
  document.write("'bits' does not appear in str <br />");
```

Regular Expressions

- What would be the output?

```
var str = "Rabbits are furry";
var position = str.search(/Fibonacci/);
if (position > 0)
  {
    document.write("'Fibonacci' appears in position", position );
    document.write("<br />");
  }
else
  {
    document.write("Position = ", position, " <br/>");
    document.write("'Fibonacci' does not appear in str <br />");
  }
```

Regular Expressions - Metacharacters

Metacharacters have special meaning in regular expressions

\ | ( ) [ ] { } ^ $ * + ? .

- These characters may be used literally by escaping them with \
- A period matches any single character
- /f.r/ matches **for** and **far** and **fir** but not **fr**

Regular Expressions - Metacharacters

- A character class matches one of a specified set of characters
- **[character set]**
- List characters individually: **[abcdef]**
- Give a range of characters: **[a-z]**
- Beware of **[A-z]**
- **^** at the beginning negates the class

Regular Expressions - Predefined character classes

Name	Equivalent Pattern	Matches
\d	[0-9]	A digit
\D	[^0-9]	Not a digit
\w	[A-Za-z_0-9]	A word character (alphanumeric)
\W	[^A-Za-z_0-9]	Not a word character
\s	[\r\t\n\f]	A whitespace character
\S	[^ \r\t\n\f]	Not a whitespace character

Regular Expressions - Repeated Matches

- A pattern can be repeated a fixed number of times by following it with a pair of curly braces enclosing a count
- A pattern can be repeated by following it with one of the following special characters
- * indicates zero or more repetitions of the previous pattern
- **+** indicates one or more of the previous pattern
- **?** indicates zero or one of the previous pattern
- **/\(\d{3}\)\-\d{3}\-\d{4}/** might represent a telephone number (Thats for USA)
- **/[$_a-zA-Z][$_a-zA-Z0-9]***/ matches identifiers

Regular Expressions - Anchors

- Anchors in regular expressions match positions rather than characters
- Anchoring to the end of a string
- **^** at the beginning of a pattern matches the beginning of a string
- **$** at the end of a pattern matches the end of a string

value and match methods, length property

- **object.value** - returns object's own enumerable property
- match() -
- method retrieves the matches when matching a string against a regular expression

```
str.match(regexp)
```

- Returns - An Array containing the entire match result and any parentheses-captured matched results;
  null if there were no matches.

value and match methods, length property

- length property represents the length of a string

```
str.length
```

- For an empty string, length is 0.

HTML Form

- A form is the usual way information is gathered from a browser
- When the Submit button of a form is clicked, the form’s values are sent to the server
- Components of a form are defined in the content of a **<form>** tag
- Attributes – action , method
- **action**, specifies the **URL** that is to be called when the Submit button is clicked

```
<form action ="http://www.cs.ucp.edu/cgi-bin/survey.pl">
```

HTML Form

- Attributes – action , method
- **method** , specifies one of the two possible techniques of transferring the form data to the server, **get** and **post**
- **input** tag
- The **type** attribute of <input> specifies the kind of widget being created
- **text** - Creates a horizontal box for text input

```
<input type = "text" name = "Phone"size = "12" >

Username:<input type="text" name="username">

Password:<input type="password" name="sem">
```

- Check box , Radio button , Select menu

HTML Form

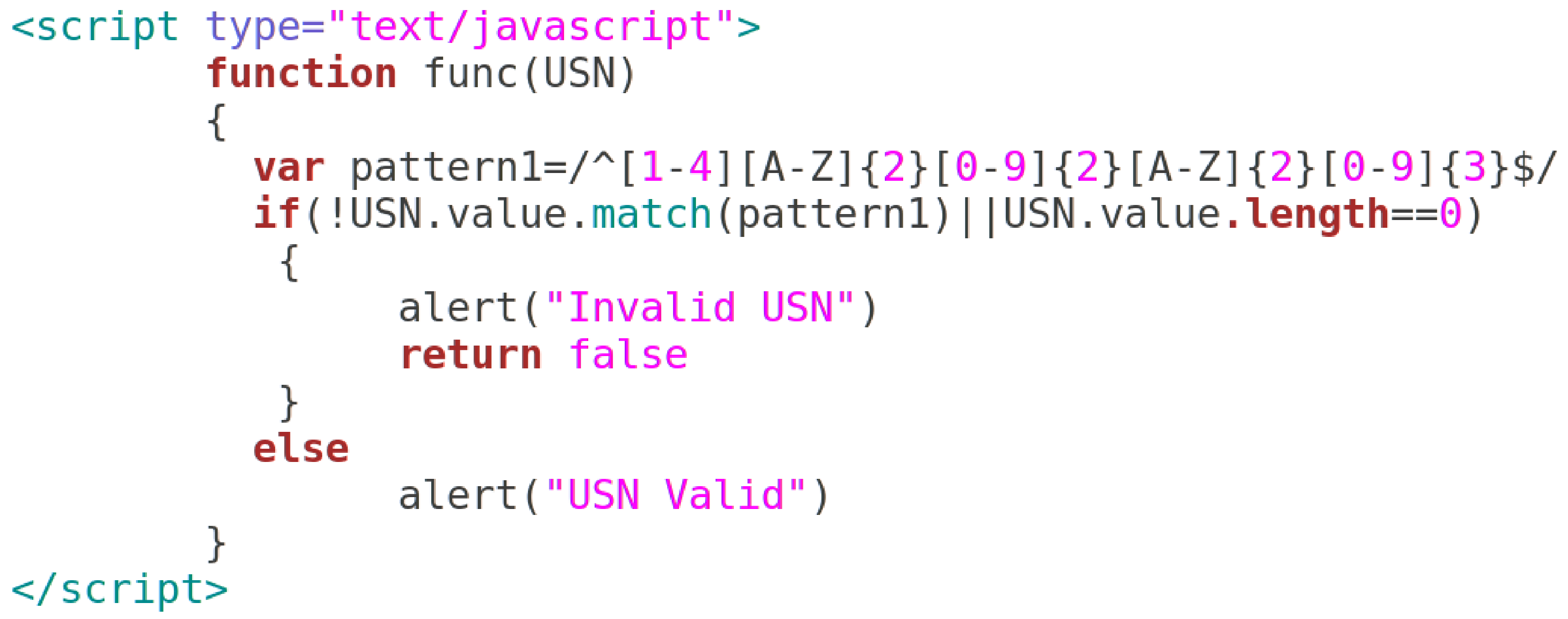
- **button** - Creates a button
- onClick attribute

```
Username:<input type="text" name="username">

Password:<input type="password" name="password">

<input type="button" value="Validate"
onclick="validateUsernameAndPassword(username,password)">
```

2a. Pseudo Code / Outline of the Algorithm

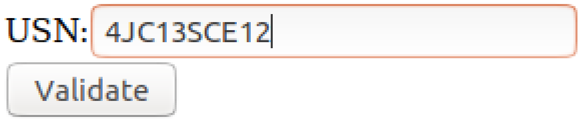
```html
<form action="">
  <p>
     USN:<input type="text" name="USN"><br/>
     <input type="button" value="Validate" onclick="func(USN)">
  </p>
</form>
```

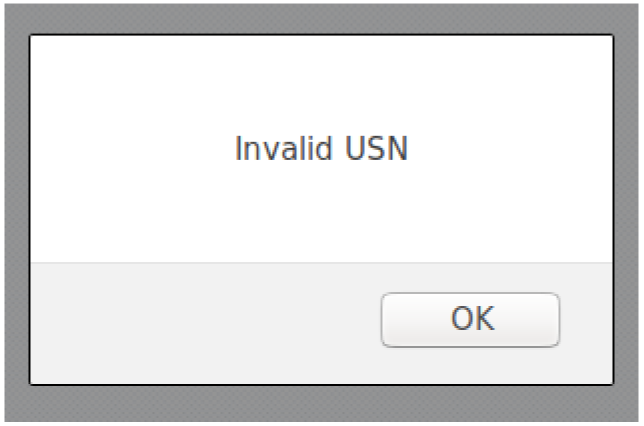
2a. Pseudo Code / Outline of the Algorithm

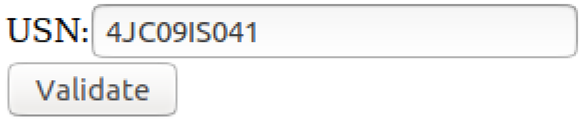


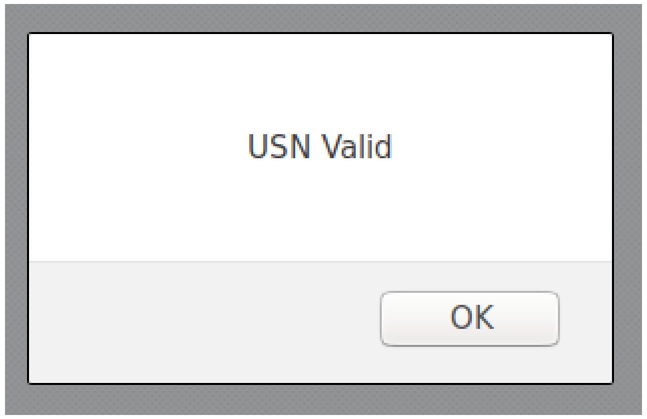
```
<script type="text/javascript">
        function func(USN)
        {
          var pattern1=/^[1-4][A-Z]{2}[0-9]{2}[A-Z]{2}[0-9]{3}$/
          if(!USN.value.match(pattern1)||USN.value.length==0)
           {
                alert("Invalid USN")
                return false
           }
          else
                alert("USN Valid")
        }
</script>
```

Sample Run

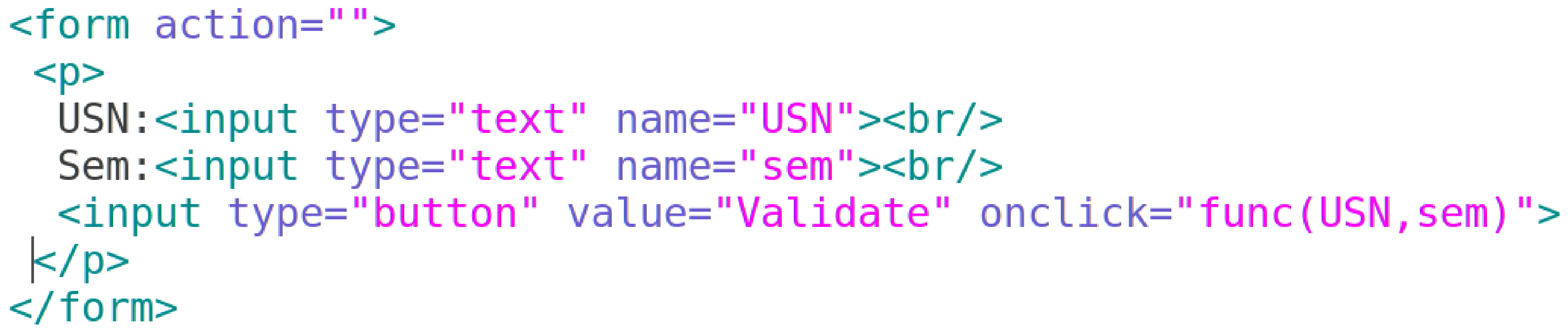

USN: 4JC13SCE12


Invalid USN


USN: 4JC09IS041


USN Valid

2b. Pseudo Code / Outline of the Algorithm

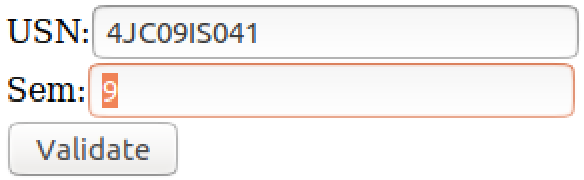


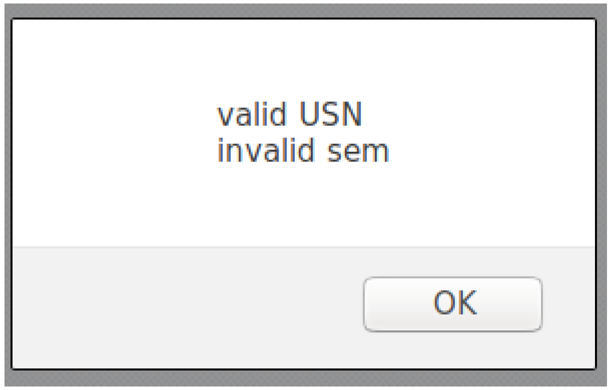
```
<form action="">
 <p>
  USN:<input type="text" name="USN"><br/>
  Sem:<input type="text" name="sem"><br/>
  <input type="button" value="Validate" onclick="func(USN,sem)">
 </p>
</form>
```

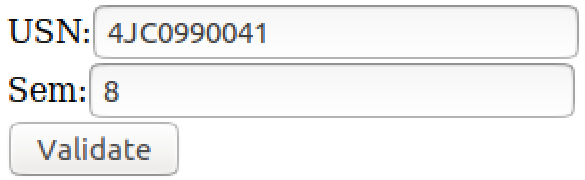
2b. Pseudo Code / Outline of the Algorithm

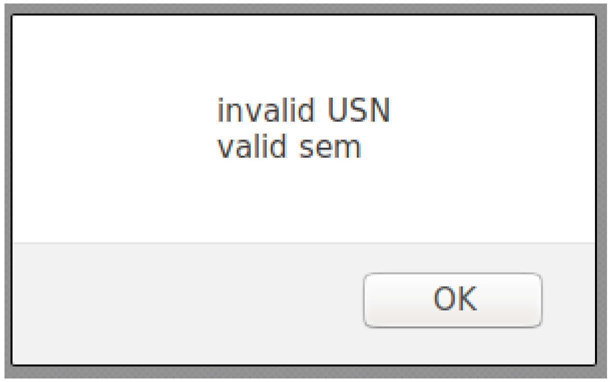
```
<script type="text/javascript">
 function func(USN, sem)
  {
    var pattern1=/^[1-4][A-Z]{2}[0-9]{2}[A-Z]{2}[0-9]{3}$/
    var pattern2=/^[1-8]$/
    var c=""

    if(!USN.value.match(pattern1)||USN.value.length==0)
     {
        c=c+"invalid USN\n"
     }
    else
        c=c+"valid USN\n"

    if(!sem.value.match(pattern2)||sem.value.length==0)
     {
        c=c+"invalid sem\n"
     }
    else
        c=c+"valid sem\n"

    alert(c)
  }
</script>
```

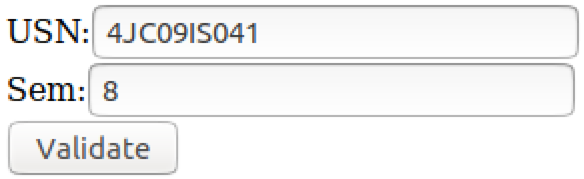
Sample Run

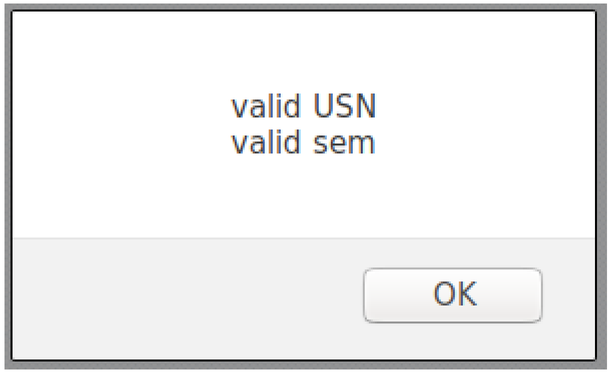

USN: 4JC09IS041
Sem: 9


valid USN
invalid sem


USN: 4JC0990041
Sem: 8


invalid USN
valid sem


USN: 4JC09IS041
Sem: 8


valid USN
valid sem

Learning Outcomes of the Experiment

At the end of the session, students should be able to :

1) Explain the features of Javascript. [L2]

2) Experiment with the usage of basic programming concepts like variables, data types and conditional statements [L3]

Firebug

[http://getfirebug.com/img/firebug-large.png]