Univerzitet Singidunum

# Web Programming

www.singidunum.ac.rs

# JavaScript Regular expressions

- A regular expression is a sequence of characters that forms a search pattern.
- When you search for data in a text, you can use this search pattern to describe what you are searching for.
- A regular expression can be a single character, or a more complicated pattern.
- Regular expressions can be used to perform all types of text search and text replace operations.
- Syntax:

```
/pattern/modifiers;
```

## JavaScript Regular expressions

- Example
- var pattern = /Singidunum/i;

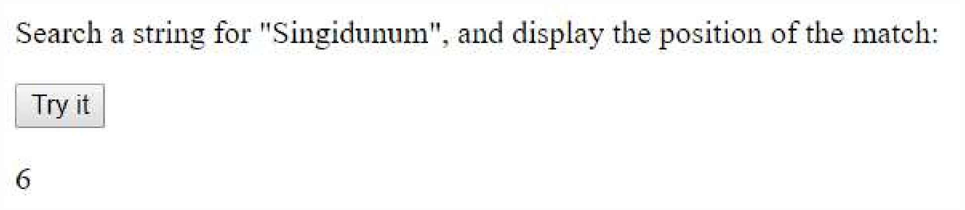
- explained:
- /Singidunum/i  is a regular expression.
- Singidunum is a pattern (to be used in a search).
- i  is a modifier (modifies the search to be case-insensitive).

## JavaScript Regular expressions

- Using String Methods
- In JavaScript, regular expressions are often used with the two string methods: search() and replace().
- The search() method uses an expression to search for a match, and returns the position of the match.
- The replace() method returns a modified string where the pattern is replaced.

# JavaScript Regular expressions

```html
<!DOCTYPE html>
<html>
<body>

<p>Search a string for "Singidunum", and display the position of the match:</p>

<button onclick="myFunction()">Try it</button>

<p id="demo"></p>

<script>
function myFunction() {
    var str = "Visit Singidunum!";
    var n = str.search(/Singidunum/i);
    document.getElementById("demo").innerHTML = n;
}
</script>

</body>
</html>
```

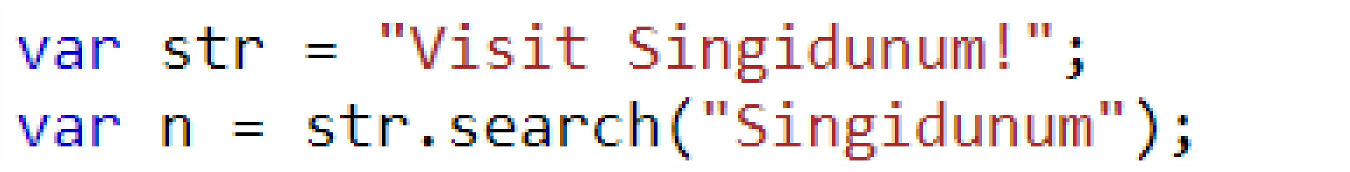


Search a string for "Singidunum", and display the position of the match:

```
6
```

## JavaScript Regular expressions

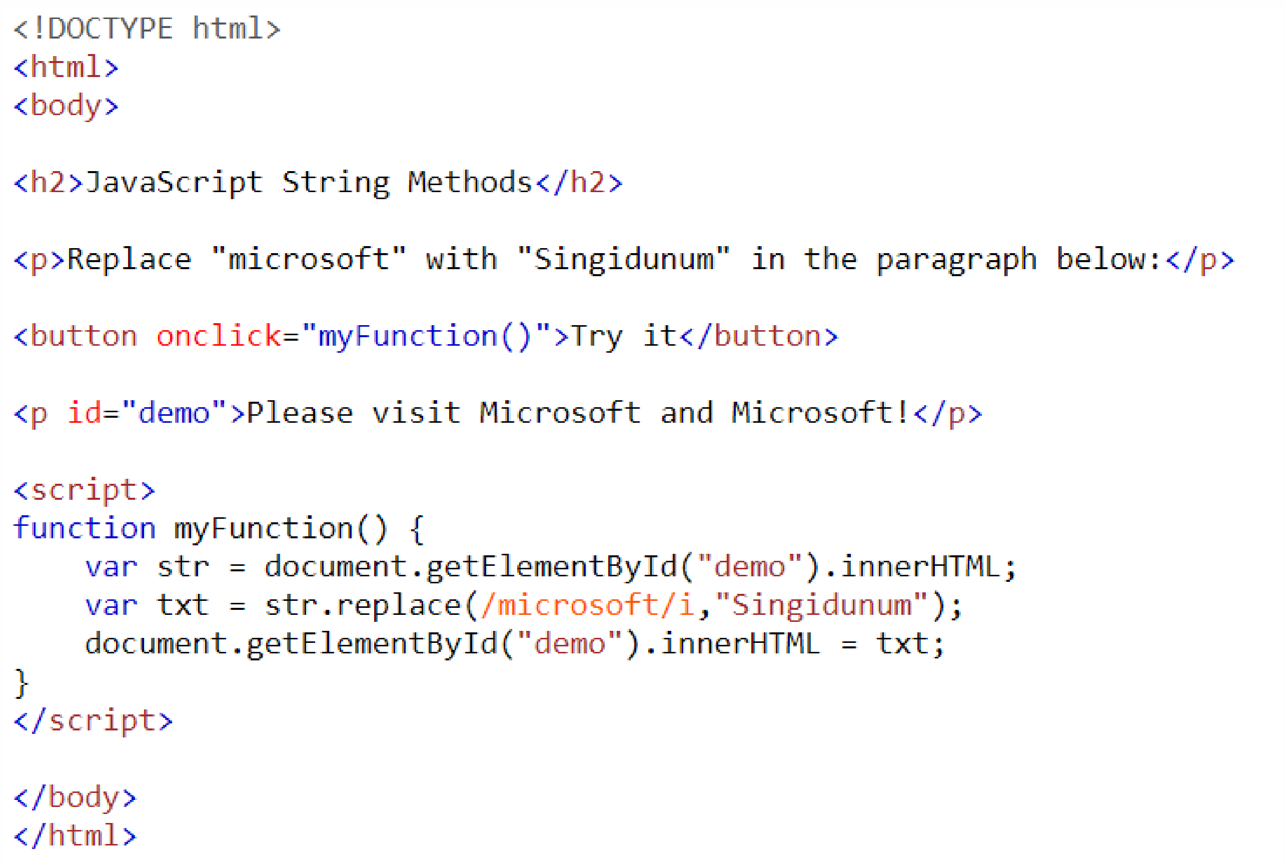
- Using String search() With String
- The search method will also accept a string as search argument. The string argument will be converted to a regular expression:



```
var str = "Visit Singidunum!";
var n = str.search("Singidunum");
```

# JavaScript Regular expressions

- Use String replace() With a Regular Expression



```
<!DOCTYPE html>
<html>
<body>

<h2>JavaScript String Methods</h2>

<p>Replace "microsoft" with "Singidunum" in the paragraph below:</p>

<button onclick="myFunction()">Try it</button>

<p id="demo">Please visit Microsoft and Microsoft!</p>

<script>
function myFunction() {
    var str = document.getElementById("demo").innerHTML;
    var txt = str.replace(/microsoft/i,"Singidunum");
    document.getElementById("demo").innerHTML = txt;
}
</script>

</body>
</html>
```

Univerzitet Singidunum

## JavaScript Regular expressions

- Using String replace() With a String
- The replace() method will also accept a string as search argument:

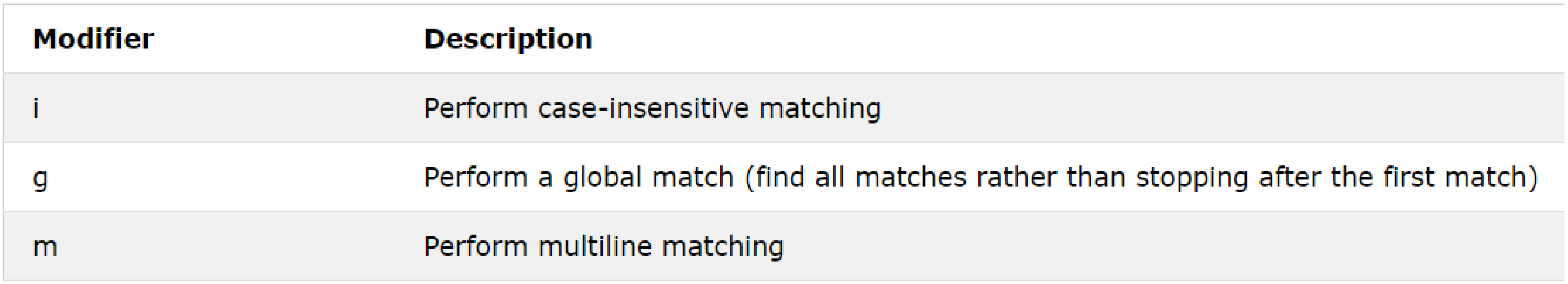
```
var str = "Visit Microsoft!";
var res = str.replace("Microsoft", “Singidunum");
```

## JavaScript Regular expressions

- Regular expression arguments (instead of string arguments) can be used in the methods above.
- Regular expressions can make your search much more powerful (case insensitive for example).

## JavaScript Regular expressions

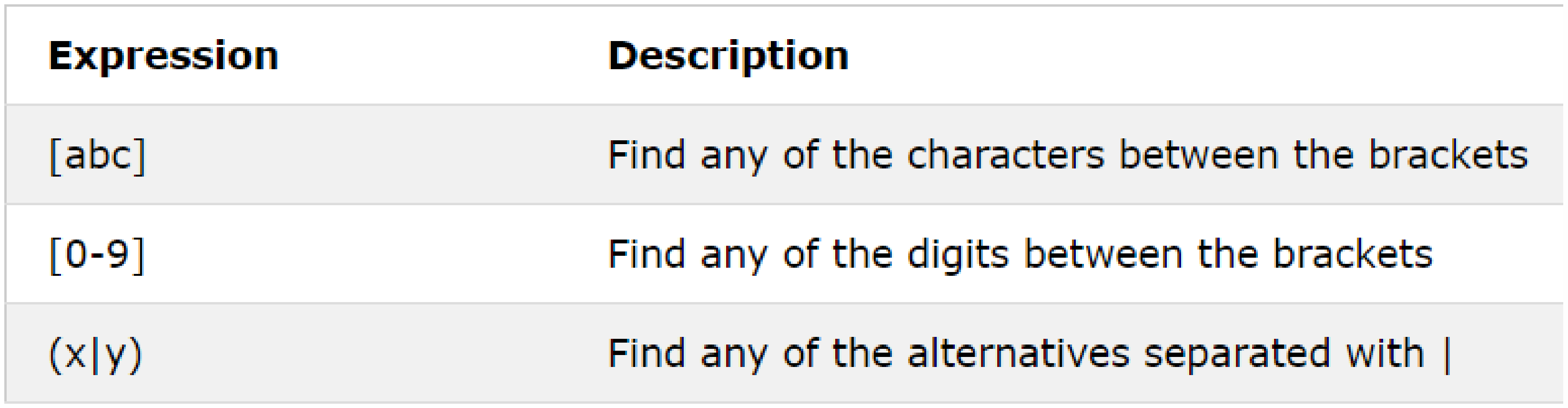
- Regular Expression Modifiers
- Modifiers can be used to perform case-insensitive more global searches:



| Modifier | Description |
|---|---|
| i | Perform case-insensitive matching |
| g | Perform a global match (find all matches rather than stopping after the first match) |
| m | Perform multiline matching |

# JavaScript Regular expressions

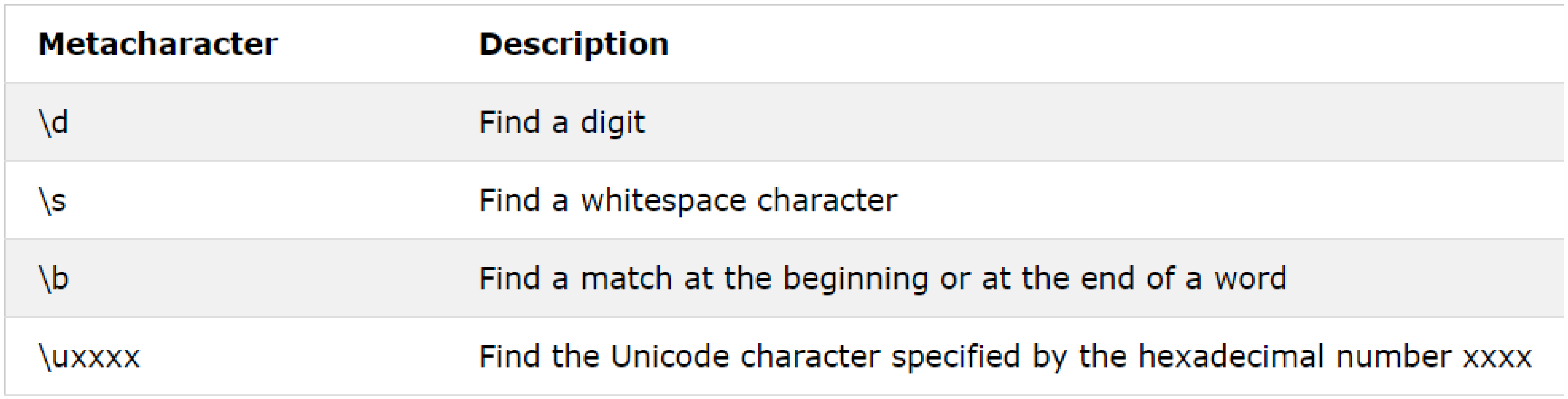
- Regular Expression Patterns
- Brackets are used to find a range of characters:



| Expression | Description |
|---|---|
| [abc] | Find any of the characters between the brackets |
| [0-9] | Find any of the digits between the brackets |
| (x\|y) | Find any of the alternatives separated with \| |

## JavaScript Regular expressions

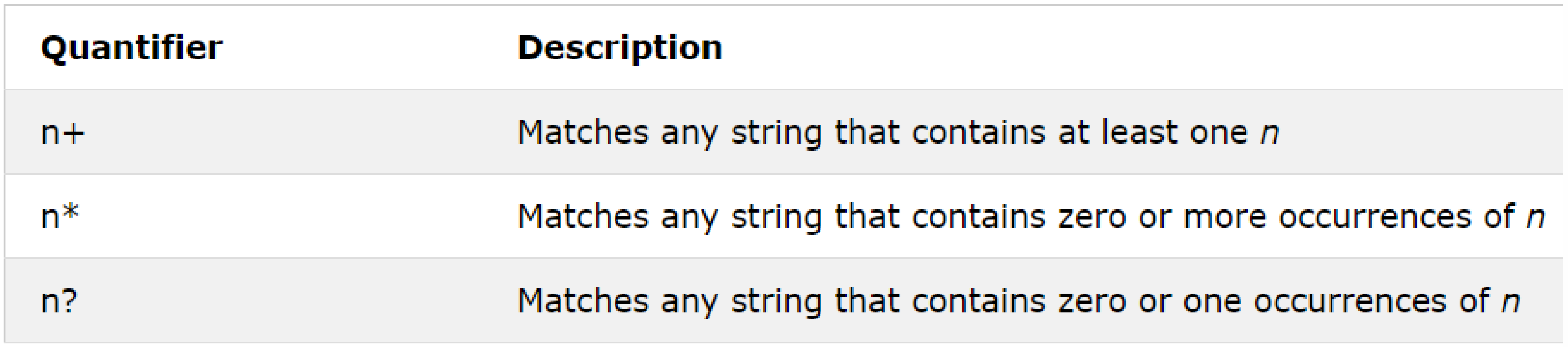
- Metacharacters are characters with a special meaning:



| Metacharacter | Description |
|---|---|
| \d | Find a digit |
| \s | Find a whitespace character |
| \b | Find a match at the beginning or at the end of a word |
| \uxxxx | Find the Unicode character specified by the hexadecimal number xxxx |

# JavaScript Regular expressions

- Quantifiers define quantities:



| Quantifier | Description |
|---|---|
| n+ | Matches any string that contains at least one *n* |
| n* | Matches any string that contains zero or more occurrences of *n* |
| n? | Matches any string that contains zero or one occurrences of *n* |

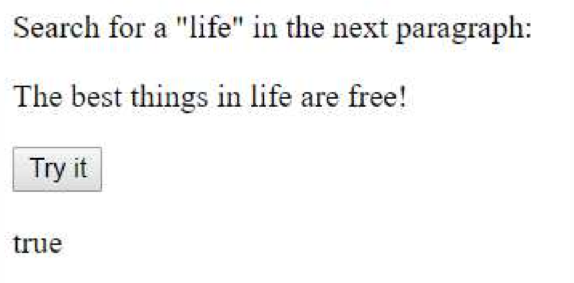
# JavaScript Regular expressions

- **Using the RegExp Object**
- In JavaScript, the RegExp object is a regular expression object with predefined properties and methods.
- The test() method is a RegExp expression method.
- It searches a string for a pattern, and returns true or false, depending on the result.
- The following example searches a string for the character "e":

```
var patt = /e/;
patt.test("The best things in life are free!");
```

# JavaScript Regular expressions

```
<!DOCTYPE html>
<html>
<body>

<p>Search for a "life" in the next paragraph:</p>

<p id="p01">The best things in life are free!</p>

<button onclick="myFunction()">Try it</button>

<p id="demo"></p>

<script>
function myFunction() {
    text = document.getElementById("p01").innerHTML;
    document.getElementById("demo").innerHTML = /life/.test(text);
}
</script>

</body>
</html>
```


Search for a "life" in the next paragraph:

The best things in life are free!

```
true
```

## JavaScript Regular expressions

- You don't have to put the regular expression in a variable first. The two lines above can be shortened to one:

  /life/.test("The best things in life are free!");

## JavaScript Regular expressions

- Using exec()
- The exec() method is a RegExp expression method.
- It searches a string for a specified pattern, and returns the found text.
- If no match is found, it returns null.
- Example :

```
/life/.exec("The best things in life are free!");
```

- Since there is a “life" in the string, the output of the code above will be:

```
life
```

## JavaScript Regular expressions

- Some examples:
- Index (9/09, 22/07, 44/08, 1001/09)

  re=/^(\d){1,4}\/(\d){2}$/
- URL for image (winter.jpg, summer2012.png, img112.gif)

  /^\S+\.(gif|jpg|jpeg|png)$/
- Password (ZimA99x)

  re=/^[A-Za-z\d]{5,12}$/
- Mail (nenad.kojic@ict.edu.rs)

  re=/^(\w+[\-\.])*\w+@(\w+\.)+[A-Za-z]+$/;

  re=/^\w+([\.-]?\w+)*@\w+([\.-]?\w+)*(\.\w{2,3})+$/

## JavaScript Regular expressions

- Date (m/d/y) i.e. 1/5/2011

  /^([\d]|1[0,1,2])/([0-9]|[0,1,2][0-9]|3[0,1])/\d{4}$/

12/21/2005

- Decimal number, i.e. 876.450

  /^\d*[0-9](\.\d*[0-9])?$/

- Name of the document, i.e. example-1.doc

  /^[a-zA-Z0-9-_\.]+\.(pdf|txt|doc|csv)$/

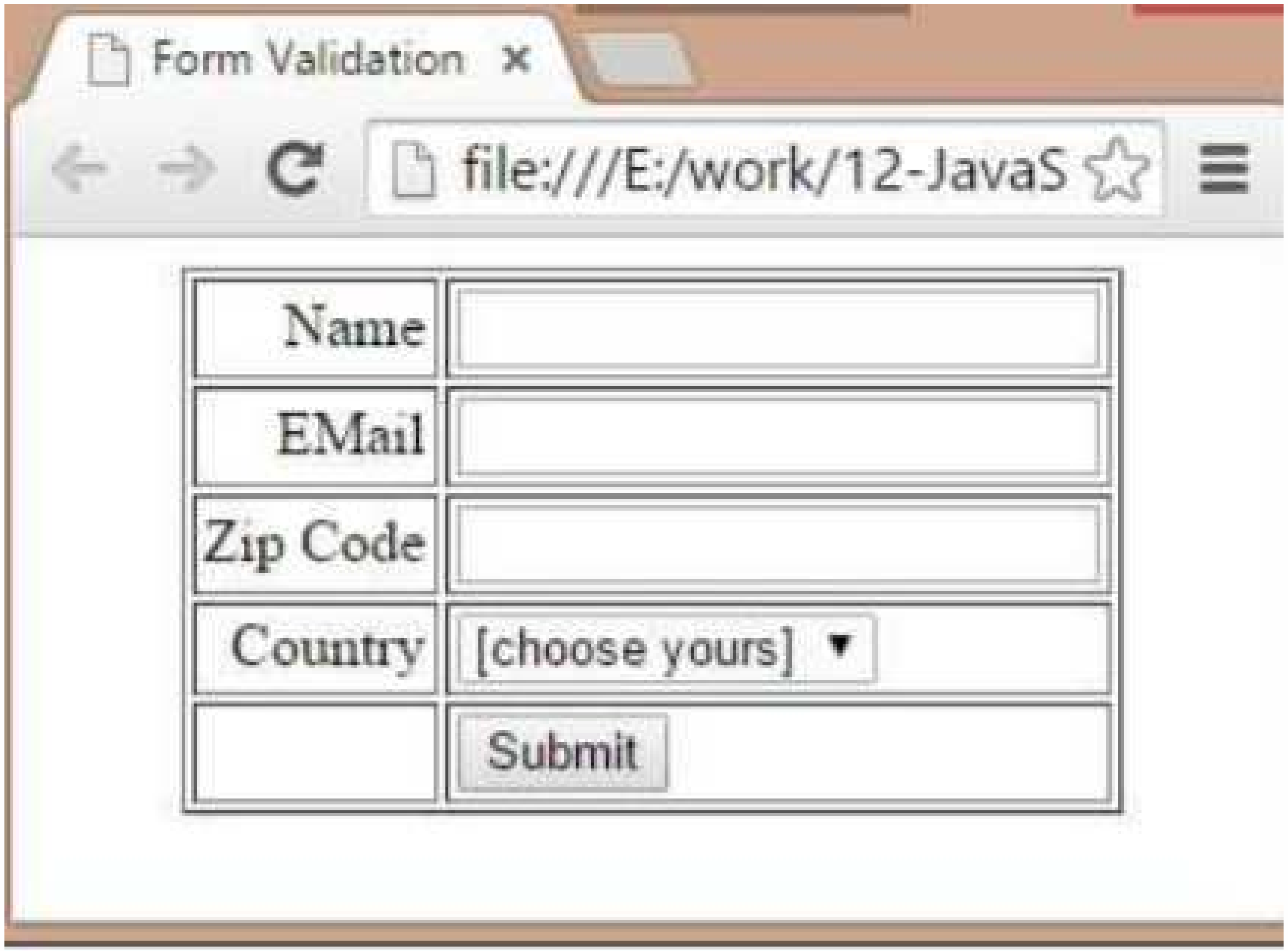
## JavaScript Validation

- Form validation normally used to occur at the server, after the client had entered all the necessary data and then pressed the Submit button.
- If the data entered by a client was incorrect or was simply missing, the server would have to send all the data back to the client and request that the form be resubmitted with correct information.
- This was really a lengthy process which used to put a lot of burden on the server.

# JavaScript Validation

- JavaScript provides a way to validate form's data on the client's computer before sending it to the web server. Form validation generally performs two functions.
  - Basic Validation - First of all, the form must be checked to make sure all the mandatory fields are filled in. It would require just a loop through each field in the form and check for data.
  - Data Format Validation - Secondly, the data that is entered must be checked for correct form and value. Your code must include appropriate logic to test correctness of data

# JavaScript Validation



Form Validation ×

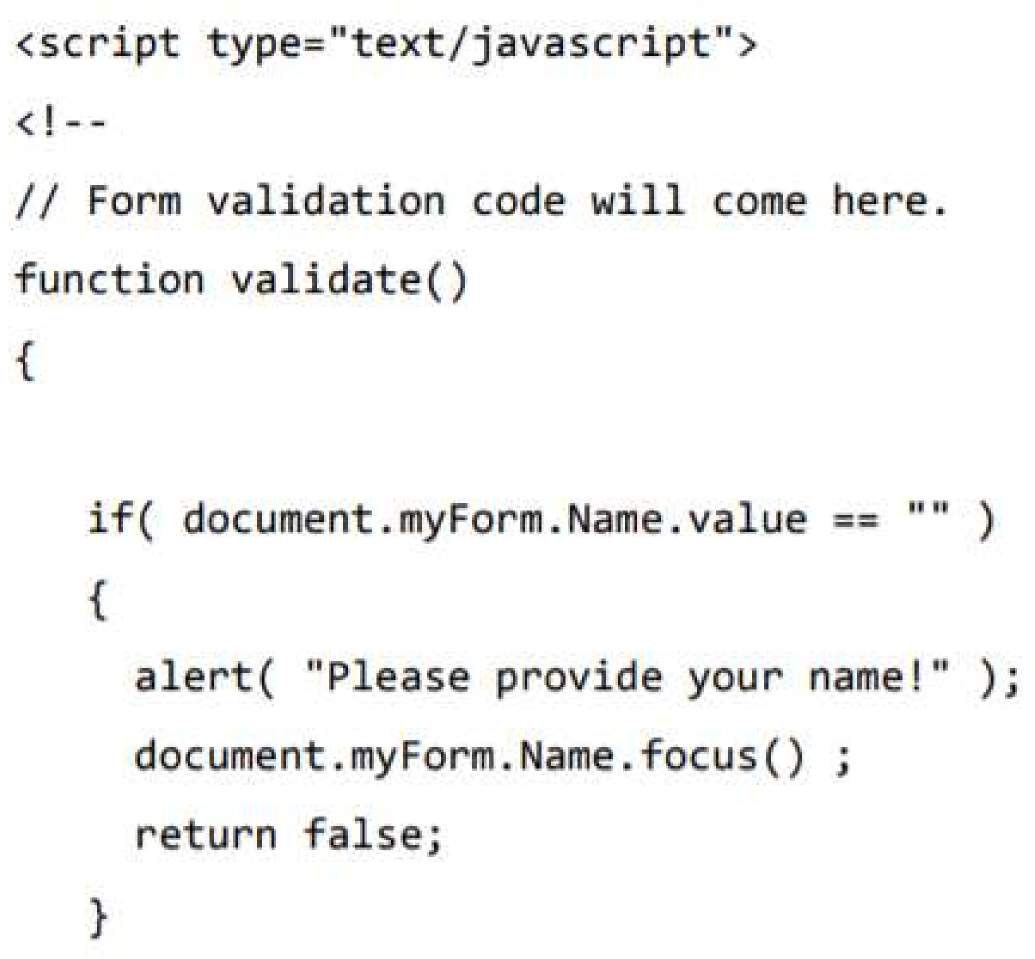
file:///E:/work/12-JavaS

|  |  |
|---|---|
| Name |  |
| EMail |  |
| Zip Code |  |
| Country | [choose yours] ▼ |
|  | Submit |

## JavaScript Validation

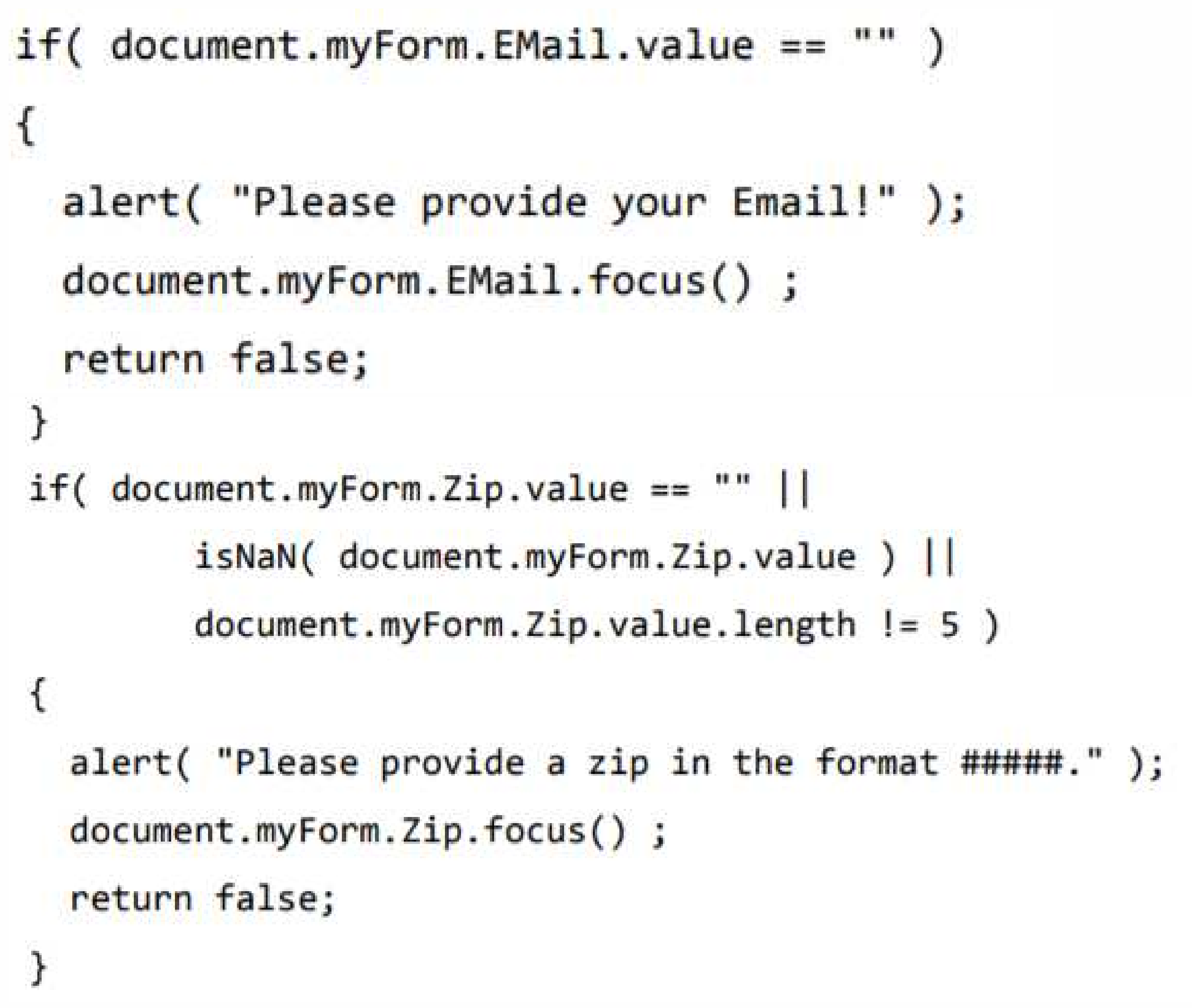
- First let us see how to do a basic form validation.
- In the above form, we are calling validate() to validate data when onsubmit event is occurring.
- The following code shows the implementation of this validate() function.

# JavaScript Validation



```
<script type="text/javascript">
<!--
// Form validation code will come here.
function validate()
{

   if( document.myForm.Name.value == "" )
   {
     alert( "Please provide your name!" );
     document.myForm.Name.focus() ;
     return false;
   }
```

# JavaScript Validation



```
if( document.myForm.EMail.value == "" )
{
  alert( "Please provide your Email!" );
  document.myForm.EMail.focus() ;
  return false;
 }
 if( document.myForm.Zip.value == "" ||
          isNaN( document.myForm.Zip.value ) ||
          document.myForm.Zip.value.length != 5 )
 {
   alert( "Please provide a zip in the format #####." );
   document.myForm.Zip.focus() ;
   return false;
 }
```

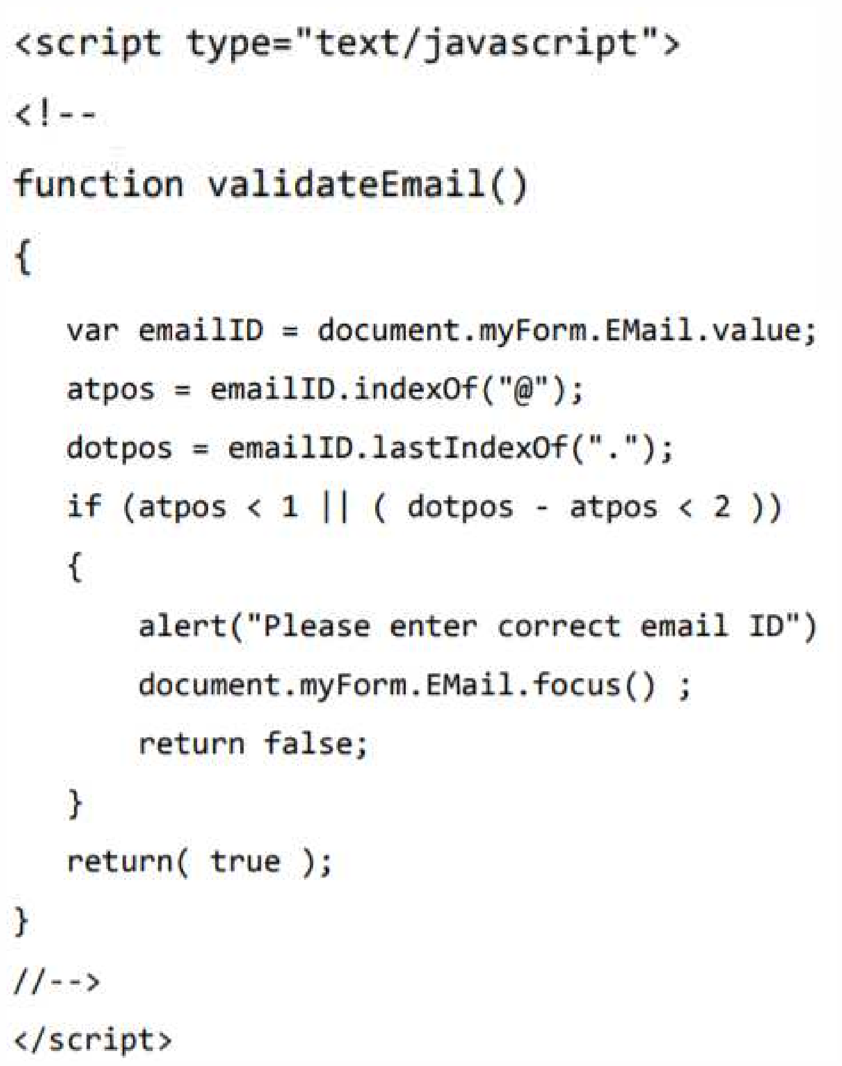
## JavaScript Validation

```
if( document.myForm.Country.value == "-1" )
{
  alert( "Please provide your country!" );
  return false;
}
return( true );
```

## JavaScript Validation

- We can validate our entered form data before submitting it to the web server.
- The following example shows how to validate an entered email address.
- An email address must contain at least a ‘@’ sign and a dot (.).
- Also, the ‘@’ must not be the first character of the email address, and the last dot must at least be one character after the ‘@’ sign.

# JavaScript Validation



```
<script type="text/javascript">
<!--
function validateEmail()
{
   var emailID = document.myForm.EMail.value;
   atpos = emailID.indexOf("@");
   dotpos = emailID.lastIndexOf(".");
   if (atpos < 1 || ( dotpos - atpos < 2 ))
   {
       alert("Please enter correct email ID")
       document.myForm.EMail.focus() ;
       return false;
   }
   return( true );
}
//-->
</script>
```