Java Script-Validation using Regular Expression

```
<form action="" onsubmit="validate()">
  <input type="text"id="phone-number" placeholder="phone number"/> <br>
  <input type="text" id="postal-code" placeholder="postal code" /><br>
  <input type="submit" />
</form>
```

Here, I have a simple form. It has an onsubmit event attached to a function called validate() that we are going to create later inside this form there are three inputs one for our phone number another one is for postal codes and the third and the last one is a submit button that is going to submit the form after we are done filling it.

Now let’s jump to the javascript code which will contain all the regular expression parts .

```
function validate()
{
  var phoneNumber = document.getElementById('phone-number').value;
  var phoneRGEX = /^[(]{0,1}[0-9]{3}[)]{0,1}[-\s\.]{0,1}[0-9]{3}[-\s\.]{0,1}[0-9]{4}$/;
  var phoneResult = phoneRGEX.test(phoneNumber);
  alert("phone:"+phoneResult );
}
```

This is our validate() function for now. It contains only the code that validates the phone number. In the first line, I am using document.getElementById() to grab the phone number from the input element. Then I created my regular expression and attached it to a variable called phoneRGEX notice that I did not need to wrap the regular expression in quotes this is because javascript natively recognizes regular expressions so there is no need to create them as strings then convert them.

Then I created a variable that contains the result of running the test() function on the phoneNumber string which will be a boolean that contain either true if the string matches our regular expression or false if it does not.

Now let’s jump into the fun part which is discussing the regular expression

```
/^[(]{0,1}[0-9]{3}[)]{0,1}[-\s\.]{0,1}[0-9]{3}[-\s\.]{0,1}[0-9]{4}$/
```

First we have the starting and ending slashes “/” , the expression then starts with a “^” sign to match with the beginning of the string. Notice the [-\s\.] this part matches a hyphen(-) space or a dot (.).[0-9]{3} means 3 digits. So basically the expression tries to match with a phone number like this (541) 754-3010 or like this 541-754-3010 or with spaces

Now let’s move on to the Postal code part.

```
function validate(){

  var phoneNumber = document.getElementById('phone-number').value;

  var postalCode = document.getElementById('postal-code').value;

  var phoneRGEX = /^[(]{0,1}[0-9]{3}[)]{0,1}[-\s\.]{0,1}[0-9]{3}[-\s\.]{0,1}[0-9]{4}$/;

  var postalRGEX = /^[A-Z]{1,2}[0-9]{1,2} ?[0-9][A-Z]{2}$/i;

  var phoneResult = phoneRGEX.test(phoneNumber);

  var postalResult = postalRGEX.test(postalCode);

  alert("phone:"+phoneResult + ", postal code: "+postalResult);

}
```

Here I have added a new variable called postalCode to fetch the postal code from the form then created another variable to store the postal code regular expression which we will be discussing in a second. Then I used the test function to test my string against the regular expression and print the result in an alert.

Now let’s jump to the regular expression itself:

UK postal code examples : EC1A 1BB W1A 0AX M1 1AE B33 8TH CR2 6XH DN55 1PT So, 1 or 2 alphabetic characters, followed by 1 or 2 digits, then a space and one digit and exactly two alphabetic characters. So here is the Regular expression pattern:

```
/^[A-Z]{1,2}[0-9]{1,2} ?[0-9][A-Z]{2}$/i
```

Now that we have got the patterns right, we can go ahead and add this code to the validation function of the form:

```
function validate(){

  var phoneNumber = document.getElementById('phone-number').value;

  var postalCode = document.getElementById('postal-code').value;

  var phoneRGEX = /^[(]{0,1}[0-9]{3}[)]{0,1}[-\s\.]{0,1}[0-9]{3}[-\s\.]{0,1}[0-9]{4}$/;

  var postalRGEX = /^[A-Z]{1,2}[0-9]{1,2} ?[0-9][A-Z]{2}$/i;

  var phoneResult = phoneRGEX.test(phoneNumber);

  var postalResult = postalRGEX.test(postalCode);

  if(phoneResult == false)

  {

    alert('Please enter a valid phone number');

    return false;

  }
```

```
 if(postalResult == false)

 {

  alert('Please enter a valid postal number');

  return false;

 }

 return true;

}
```

More Regular Expression Checks

1. Date
   ```
   /^\d{2}\/\d{2}\/\d{4}$/
   ```
   This simple regular expression just checks for 2 digits / 2 digits / 4 digits date format
   If you want more complex, tight validation for mm/dd/yyyy format, here is the format
   ```
   /^(0[1-9]|1[0-2])\/(0[1-9]|1\d|2\d|3[01])\/(19|20)\d{2}$/
   ```
   source

2. URL A URL of the format http(s)://website/page can be validated with this regular expression:
   ```
   https?:\/\/(www\.)?[-a-zA-Z0-9@:%._\+~#=]{2,256}\.[a-z]{2,6}\b([-a-zA-Z0-
   9@:%_\+.~#?&//=]*)
   ```
   source

3. Any Alpha Numeric String If you want to allow only alphanumeric characters, use this regular expression:
   ```
   /^[a-zA-Z0-9]*$/
   ```

4. Decimal Numbers For decimal numbers with one decimal point, the regular expression is:
   ```
   /^[0-9]+\.?[0-9]*$/
   ```