

# CSIT128 / CSIT828

## More on JavaScript

Joseph Tonien

# Form validation


```
<form action="myService" method="get"  
  onSubmit="return validateForm()">
```

```
... your form goes here ...
```

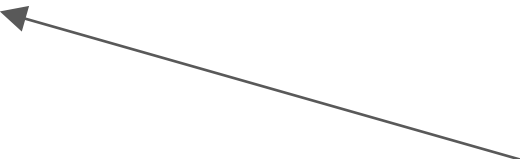
```
</form>
```

```
<script>  
function validateForm() {  
  if (... something wrong ...) {  
    return false;  
  }  
  return true;  
}  
</script>
```

Use form attribute  
`onSubmit` to check input  
before form submission



When function returns  
`false`, form will not be  
submitted



# Form validation

**Example 1:** we want user to fill out the email, if email is not filled out then we will alert the user

```
<form action="myService" method="get" onSubmit="return  
validateForm()" ">
```

```
Email: <input id="email" type="text" name="email">
```

```
<br /><br />
```

```
<input type="submit" value="Submit">
```

```
</form>
```



Email:

Submit

# Form validation

**Example 1:** we want user to fill out the email, if email is not filled out then we will alert the user

```
<script>
function validateForm() {
    var email = document.getElementById("email").value;

    if (email == null || email == "") {
        alert("Email must be filled out");
        return false;
    }

    return true;
}
</script>
```

Email: <input **id="email"** type="text" name="email">

# Form validation

## Example 2:

<http://www.uow.edu.au/~dong/w3/example/js/validation2.html>

What if user enter only whitespaces?

The `trim()` method removes whitespace from both sides of a string.

```
<script>
function validateForm() {
    var email = document.getElementById("email").value;

    if (email == null || email.trim() == "") {
        alert("Email must be filled out");
        return false;
    }

    return true;
}
</script>
```

Email:

# Form validation

## Example 3:

<http://www.uow.edu.au/~dong/w3/example/js/validation3.html>

If user didn't fill out the email, we want to display an error message.  
We use a **span** element as a placeholder for the error message.

```
<form action="myService" method="get" onSubmit="return  
validateForm()">
```

```
Email: <input id="email" type="text" name="email">
```

```
<span id="emailError"></span>
```

```
<br /> <br />
```

```
<input type="submit" value="Submit">
```

```
</form>
```

# Form validation

If user didn't fill out the email, we want to display an error message.  
We use a **span** element as a placeholder for the error message.

```
function validateForm() {  
    var email = document.getElementById("email").value;  
  
    if (email == null || email.trim() == "") {  
        document.getElementById("emailError").innerHTML =  
            "Email must be filled out";  
  
        return false;  
    }  
  
    return true;  
}
```

```
<span id="emailError"></span>
```

# Form validation

We want the error message has the color red.

```
<style>
#emailError{
  color: red;
}
</style>
```

```
<span id="emailError"></span>
```



# Form validation

**Example 4:** We want to have two input fields. One for email and another one for email confirmation. User has to fill in the same email for both input fields.

```
<form action="myService" method="get" onSubmit="return  
validateForm()" ">
```

```
Email: <input id="email" type="text" name="email">
```

```
<span id="emailError"></span>
```

```
<br />
```

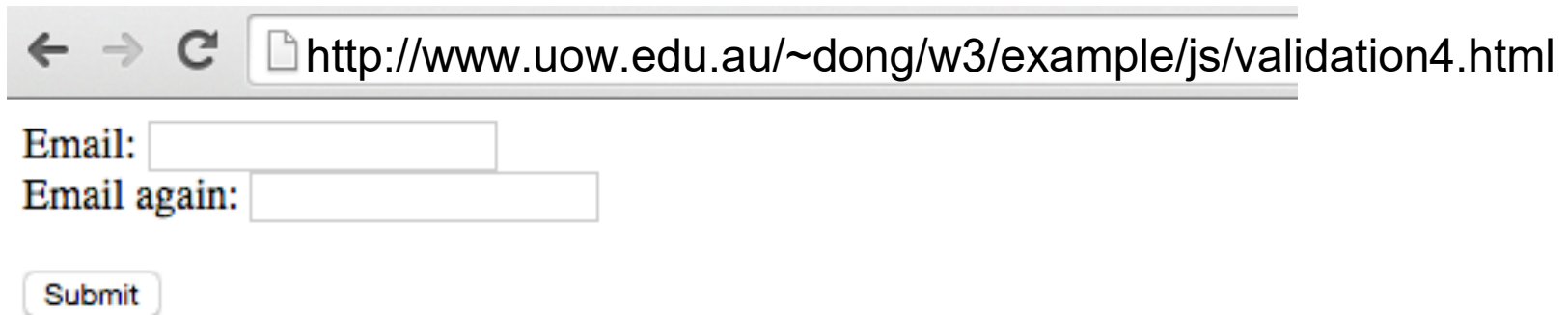
```
Email again: <input id="email2" type="text" name="email2">
```

```
<span id="emailError2"></span>
```

```
<br /> <br />
```

```
<input type="submit" value="Submit">
```

```
</form>
```



The screenshot shows a web browser window with the address bar displaying `http://www.uow.edu.au/~dong/w3/example/js/validation4.html`. Below the address bar, the form is rendered. It consists of two text input fields. The first field is preceded by the label "Email:" and the second by "Email again:". Below these fields is a "Submit" button.

# Form validation

```
function validateForm() {  
    var email = document.getElementById("email").value;  
    if (email == null || email.trim() == ""){  
        document.getElementById("emailError").innerHTML  
            = "Email must be filled out";  
        return false;  
    }  
  
    var email2 = document.getElementById("email2").value;  
    if (email2 == null || email2.trim() == ""){  
        document.getElementById("emailError2").innerHTML =  
            "Email must be filled out";  
        return false;  
    }  
  
    if(email.trim() != email2.trim()) {  
        document.getElementById("emailError2").innerHTML =  
            "Email does not matched";  
        return false;  
    }  
    return true;  
}
```

# Form validation

We want all the error messages have the color **red**.

```
<style>
#emailError{
  color: red;
}

#emailError2{
  color: red;
}
</style>
```

```
<span id="emailError"></span>
<span id="emailError2"></span>
```

# Form validation

Better solution: using **class**

```
<style>
.errorMessage{
  color: red;
}
</style>
```

```
<span id="emailError" class="errorMessage"></span>
<span id="emailError2" class="errorMessage"></span>
```

# Form validation

Now suppose that user didn't fill out the email and click Submit, there will be a red error message next to the first input field.

Suppose that user fixed the error by filling out the first email, but leaving the second email field blank.

When the user clicks Submit, we will see that the error message next to the first input field still shows.

← → ↻ http://www.uow.edu.au/~dong/w3/example/js/validation4.html

Email:  Email must be filled out ← We don't want this

Email again:  Email must be filled out

# Form validation

<http://www.uow.edu.au/~dong/w3/example/js/validation4b.html>

We will fix the javascript code

```
function validateForm() {  
    var email = document.getElementById("email").value;  
  
    if (email == null || email.trim() == "") {  
        document.getElementById("emailError").innerHTML =  
            "Email must be filled out";  
        return false;  
    } else {  
        document.getElementById("emailError").innerHTML = "";  
    }  
    ...  
}
```

# Form validation

We will fix the javascript code

```
function validateForm() {  
    ...  
    var email2 = document.getElementById("email2").value;  
  
    if (email2 == null || email2.trim() == "") {  
        document.getElementById("emailError2").innerHTML =  
            "Email must be filled out";  
        return false;  
    }else{  
        document.getElementById("emailError2").innerHTML = "";  
    }  
    ...  
}
```

# Form validation

We will fix the javascript code

```
function validateForm() {  
    ...  
    if(email.trim() != email2.trim()){  
        document.getElementById("emailError2").innerHTML =  
            "Email does not matched";  
        return false;  
    }else{  
        document.getElementById("emailError2").innerHTML = "";  
    }  
    ...  
}
```



# Form validation

Final touch: we want to remove all whitespaces in the two input fields before submit

```
function validateForm() {  
    var email = document.getElementById("email").value;  
  
    if (email == null || email.trim() == "") {  
        document.getElementById("emailError").innerHTML =  
            "Email must be filled out";  
        return false;  
    } else {  
        document.getElementById("emailError").innerHTML = "";  
        document.getElementById("email").value = email.trim();  
    }  
    ...  
}
```

# Form validation

Final touch: we want to remove all whitespaces in the two input fields before submit

```
function validateForm() {  
    ...  
    var email2 = document.getElementById("email2").value;  
  
    if (email2 == null || email2.trim() == "") {  
        document.getElementById("emailError2").innerHTML =  
            "Email must be filled out";  
        return false;  
    } else {  
        document.getElementById("emailError2").innerHTML = "";  
        document.getElementById("email2").value = email2.trim();  
    }  
    ...  
}
```

# Form validation

## Example 5:

<http://www.uow.edu.au/~dong/w3/example/js/validation5.html>

Ask user a simple math problem, only submit the form if user answers correctly

```
function validateForm() {  
    ...  
    var answer = prompt("What is 1+2 ?");  
    if(answer == null || answer != 3){  
        return false;  
    }  
    ...  
}
```

# Form validation

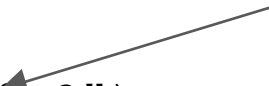
## Example 5:

<http://www.uow.edu.au/~dong/w3/example/js/validation5.html>

Ask user a simple math problem, only submit the form if user answers correctly

```
function validateForm() {  
    ...  
    var answer = prompt("What is 1+2 ?");  
    if(answer == null || answer != 3){  
        return false;  
    }  
    ...  
}
```

Can we generate random question?



# Form validation

`Math.random()` :


returns a random number between 0 (inclusive) and 1 (exclusive),  
for example, `.753`

`Math.floor(x)` :

returns the greatest integer below `x`  
for example, `Math.floor(4.6) = 4`

To get a random number between 0 and 9:

```
Math.floor(Math.random() * 10);
```



`.753 * 10 = 7.53`  
`Math.floor(7.53) = 7`

To get a random number between 1 and 10:

```
Math.floor(Math.random() * 10) + 1;
```

# Form validation

**Example 5:** Ask user a simple math problem, only submit the form if user answers correctly

<http://www.uow.edu.au/~dong/w3/example/js/validation5b.html>

Generate random question



```
function validateForm() {  
    ...  
    var x = Math.floor(Math.random() * 10) + 1;  
    var y = Math.floor(Math.random() * 10) + 1;  
    var correctAnswer = x + y;  
  
    var answer = prompt("What is " + x + " + " + y + " ?");  
    if(answer == null || answer != correctAnswer) {  
        return false;  
    }  
    ...  
}
```

# Form validation

We can use input **required** attribute

Email: `<input id="email" type="text" name="email" required>`

```
<form action="myService" method="get">
```

```
Email: <input id="email" type="text" name="email"
title="Please enter email." required>
```

```
<br />
```

```
Email again: <input id="email2" type="text" name="email2"
title="Please enter email again." required>
```

```
<br /> <br />
```

```
<input type="submit" value="Submit">
```

```
</form>
```

However, it still allows user to enter whitespace only

To fix this, we need to use **regular expression**

# Input attribute

Title: just a small hints

Placeholder: inside textbox

Oninvalid: customise error message

```
<html>
  <head></head>
  <body>
    <form action="myService" method="get">
      Email:
      <input type="text" id="email" name="email"
        title="Email should not be empty"
        placeholder = "error if you see me!"
        oninvalid = "this.setCustomValidity( 'hello already tell u see me is error, still submit!');"
        required>
      <input type="submit" onsubmit="return validateForm()">
    </form>
  </body>
</html>
```

Email:



hello already tell u see me is error, still submit!



# Regular expression

A regular expression describes a pattern of characters

`/pattern/`



`^...$` Starts and ends

`[abc]` Only a, b, or c

`[^abc]` Not a, b, nor c

`[a-z]` Characters a to z

`[0-9]` Numbers 0 to 9

`\d` Any Digit

`\D` Any Non-digit character

`\w` Any Alphanumeric character

`\W` Any Non-alphanumeric character

`\s` Any Whitespace

`\S` Any Non-whitespace character

# Regular expression

A regular expression describes a pattern of characters

`/pattern/`



`.` Any Character  
`\.` Period

`{m}` m Repetitions  
`{m,n}` m to n Repetitions

`*` Zero or more repetitions  
`+` One or more repetitions  
`?` Optional character

`(...)` Capture Group  
`(abc|def)` Matches abc or def

# Regular expression

Some characters which have special meaning need to be **escaped** before put into the regular expression

.	Any Character
\.	Period

Here is a list of special characters that need to be **escaped**

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

# Regular expression

## Example 1:

```
<form action="myService" method="get">
```

Enter some text with pattern

```
<input type="text" name="t1"
```

```
  pattern="^[A-Z][a-z], [A-Z][a-z]$" required>
```

```
<br /> <br />
```

```
<input type="submit" value="Submit">
```

```
</form>
```

Bt, Ca	match
Bt Ca	not match
Bt,Ca	not match
BT, Ca	not match
Bt, CA	not match
Da, Te	match

<code>^[A-Z]</code>	start with a letter in the range A-Z
<code>[a-z]</code>	follow by a letter in the range a-z
<code>,</code>	follow by a comma and a space
<code>[A-Z]</code>	follow by a letter in the range A-Z
<code>[a-z]\$</code>	end with a letter in the range a-z

<http://www.uow.edu.au/~dong/w3/example/js/regex1.html>

# Regular expression

We can use javascript to check regular expression

```
<script>
function validateForm(){
    var input = document.getElementById("t1").value;
    if(/^[A-Z][a-z], [A-Z][a-z]$/.test(input) == false){
        alert("input does not match the pattern");
        return false;
    }
    return true;
}
</script>
```

```
<form action="myService" method="get"
    onsubmit="return validateForm()">
```

Enter some text with pattern

```
<input id="t1" type="text" name="t1" required>
```

...

<http://www.uow.edu.au/~dong/w3/example/js/regex1b.html>

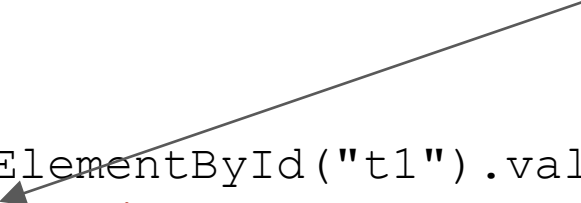
# Regular expression

Using javascript, we need to write regular expression like this

`/pattern/`

## Comparing the two methods

```
<script>
function validateForm(){
    var input = document.getElementById("t1").value;
    if(/^[A-Z][a-z], [A-Z][a-z]$/.test(input) == false){
        alert("input does not match the pattern");
        return false;
    }
    return true;
}
</script>
```



```
<form action="myService" method="get"
    onSubmit="return validateForm()">
```

Enter some text with pattern

```
<input id="t1" type="text" name="t1" required>
...
```

# Regular expression

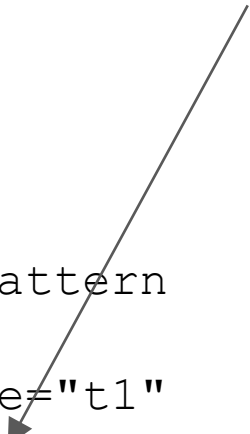
## Comparing the two methods

Using *pattern attribute*, we need to write regular expression like this

```
<input pattern="just-the-pattern" >
```

Enter some text with pattern

```
<input type="text" name="t1"  
  pattern="^[A-Z][a-z], [A-Z][a-z]$" required>  
...
```



# Regular expression

## Example 2:

```
<form action="myService" method="get">
```

Enter some text with pattern

```
<input id="t1" type="text" name="t1" pattern="^[0-9]{5}$"
required>
```

```
<br /> <br />
```

```
<input type="submit" value="Submit">
```

```
</form>
```

12345	match
1234	not match
123456	not match
12 45	not match
03527	match
23492	match

<code>^[0-9]</code>	start with a letter in the range 0-9
<code>{5}</code>	repeated 5 times
<code>\$</code>	till the end

<http://www.uow.edu.au/~dong/w3/example/js/regex2.html>



# Regular expression

## Example 3:

```
<form action="myService" method="get">
```

Enter some text with pattern

```
<input id="t1" type="text" name="t1" pattern="^[0-9]{2,5}$"
required>
```

```
<br /> <br />
```

```
<input type="submit" value="Submit">
```

```
</form>
```

1	not match
12	match
123	match
1234	match
12345	match
123456	not match

<code>^[0-9]</code>	start with a letter in the range 0-9
<code>{2,5}</code>	repeated 2 to 5 times
<code>\$</code>	till the end end

<http://www.uow.edu.au/~dong/w3/example/js/regex3.html>

# Regular expression

## Example 4:

```
pattern="^[0-9]{2,}$"
```

2 or more digits, the same as

```
pattern="^\d{2,}$"
```

# Regular expression

## Example 5:

```
pattern="^{5,}$"
```

5 or more characters

## Example 6:

```
pattern="^{4,10}$"
```

4 to 10 characters

## Example 7:

```
pattern="^[0-9]{1,3}\.[0-9]{1,3}$"
```

1 to 3 digits, follow by a period, and then 1 to 3 digits

# Regular expression

## Example 8:

```
pattern="^[^0-9][0-9]$"
```

Not a digit, follow by a digit, the same as

```
pattern="^\D\d$"
```

## Example 9:

```
pattern="^[0-9]+[a-z]*$"
```

One or more digit, follow by zero or more letters a-z

## Example 10:

```
pattern="^http(s?):\/\/.+ $"
```

http:// (or https://) follow by one or more characters

# Regular expression

## Example 11:

```
pattern="^[0-9]{4}-[0-9]{4}$"
```

4 digits, follow by -, follow by 4 digits

## Example 12:

```
pattern="^[0-9]{2}:[0-9]{2}:[0-9]{4}$"
```

2 digits, follow by :, follow by 2 digits, follow by :, follow by 4 digits

## Example 13:

```
pattern="^(NSW|ACT|NT|QLD|SA|TAS|VIC|WA) [0-9]{4}$"
```

NSW or ACT, ..., or WA, follow by a space, follow by 4 digits

# Regular expression

## Example 14:

```
pattern="(0[1-9]|1[0-2])-20(1[0-9]|2[0-5])$"
```

01, or 02, ..., or 09, or 10, or 11, or 12, follow by -, follow  
by 20, follow by 10, or 11, ..., or 19, or 20, or 21, ..., or 25

00-2010 not match

01-2010 match

12-2021 match

12-2026 not match

# References

`http://www.w3schools.com/js`

`http://www.w3schools.com/jsref/jsref\_obj\_regexp.asp`

Robert W. Sebesta, *Programming the World Wide Web*, Pearson.