Multimedia im Netz (Online Multimedia)

Wintersemester 2014/15

Übung 06 (Nebenfach)

Today‘s Agenda

- Flashback! 5th tutorial
- Introduction to JavaScript
- Assignment 5 - discussion

Flashback!

Explain a concept from last week‘s tutorial to your grandmother!

JavaScript: Events (I)

```
<!DOCTYPE html>
<html>
<head lang="en">
    <meta charset="UTF-8">
    <title>Call Me!</title>
</head>
<body>
<!-- Variant 1 -->
<input type="button" name="text" value="First Button" onclick="callMe()"/>
<!-- Variant 2 -->
<input type="button" name="text" value="Second Button" onclick="callMe2(this)"/>

<script>
    //Variant 1
    function callMe(){
        alert("Hi");
    }

    //Variant 2
    function callMe2(element){
        alert("Button Value: " + element.value);
    }
</script>

</body>
</html>
```

JavaScript: Events (II) - addEventListener

```
<!DOCTYPE html>
<html>
<head lang="en">
    <meta charset="UTF-8">
    <title>EventListener Call Me</title>
</head>
<body>
<!-- Variant 3 -->
<input id="button3" type="button" name="text" value="Click 3"/>
<script>
    //Variant 3: adding an EventListener.
    // Also known as handler or callback function.
    var button3 = document.getElementById("button3");
    button3.addEventListener("click", function(){
        alert("Button Value: " + this.value);
    });
</script>
</body>
</html>
```

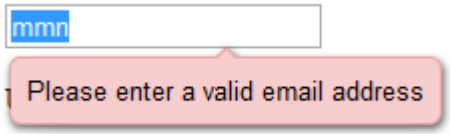
Event Types

- Mouse Events
  - onclick
  - ondbclick
  - onmousedown
  - onmouseover
  - …
- Keyboard Events
  - onkeydown
  - onkeyup
  - …
- …

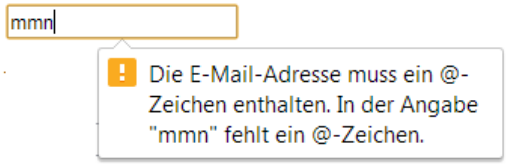
HTML5 Form Validation

- Additional semantic markup for forms in HTML allow you to validate user input before the values are sent.

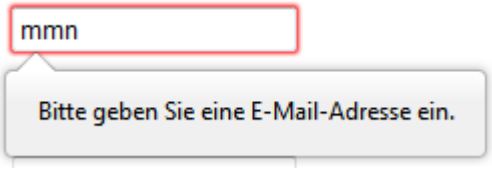
- Opera



- Chrome



- Firefox



- Safari

HTML 5: new input types

- **E-Mail**

```
<input type="email" id="userEmail" name="userEmail" />
```

- **URL**

```
<input type="url" id="someURL" name="testURL" />
```

- **…**

HTML 5: new attributes

- Prevent for validation (also works on input-elements)
  ```
  <form novalidate="novalidate“>
  ```
- Place holder
  ```
  <input type="email" placeholder="Your Email" />
  ```
- Required fields
  ```
  <input type="email" required="true" />
  ```
- Autofocus on a specific field
  ```
  <input type="text" autofocus="true" />
  ```
- Pattern
  ```
  <input type="text" pattern="[a-zA-Z]" />
  ```

Patterns & Regular Expressions

- Regular expressions define requirements for strings
- The process of validating is often called „pattern matching“

- From the previous slide:

```
<input type="text" pattern="[a-zA-Z]" />
```

  Here you can only enter latin letters in lower or upper case, but no numbers or special characters.

Regular Expressions: Characters

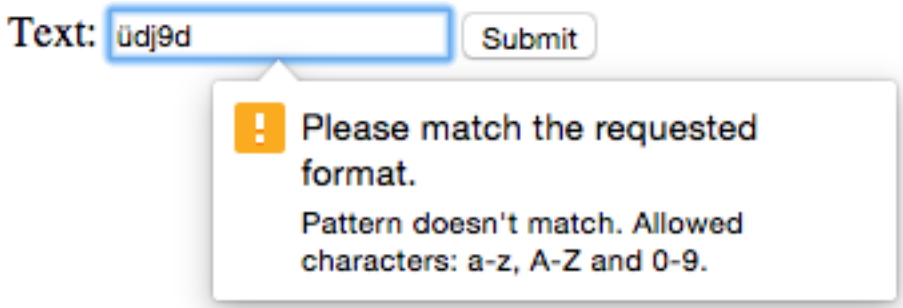
[0-6]	A digit from 0 to 6
[A-Za-z]	A character included in A-Z or a-z
[A-Za-z0-9]	A character included in A-Z or a-z or a numeric character (= alphanumeric characters)
[^z]	Any character except z
\d	Shortcode for „digit“ → [0-9]
\D	Any character that‘s not a digit
\w	A letter, a digit or an underscore
\W	Opposite of \w
\s	whitespace
\S	Any character that‘s not whitespace

Regular Expressions: Quantifiers

- How often can a character be used?

?	Preceding expression is optional: It can be used once or not at all
+	Preceding expression must be used at least once.
*	Preceding expression can be used any number of times
{*n*}	Preceding expression can be used exactly *n* times
{*min*, }	Preceding expression must be used at least *min* times
{*min*, *max*}	Preceding expression must be used at least *min* times but at most *max* times

Adjust Error Messages for the User




```
<form>
    <label for="myText">Text: </label>
    <input id="myText" type="text" name="someText"
           pattern="[a-zA-Z0-9]+"
           title="Pattern doesn't match. Allowed
                  characters: a-z, A-Z and 0-9."
           required/>
    <input type="submit" id="submit"/>
</form>
```

Constraint Validation API

- JavaScript API to handle form validation with DOM nodes
- `validity` attribute represents a `ValidityState` object. It can have different states, e.g.
  - valueMissing
  - typeMismatch
  - patternMismatch

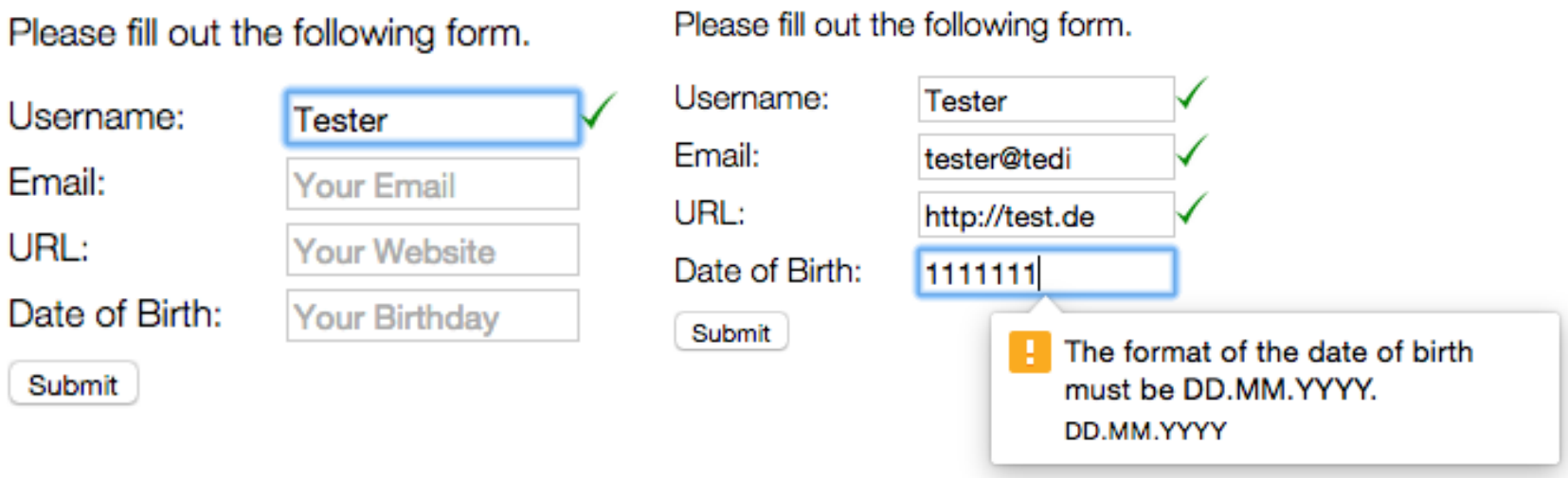
- checkValidity() allows you to check if all ValidityState objects inside a form are **`true`**
- setCustomValidity() is a function to set a custom error message (alternative to method on previous slide)

Customizing Error Messages

```
<!DOCTYPE html>
<html>
<head lang="en">
    <meta charset="UTF-8"><title>Constraint Validation API</title>
</head>
<body>
<form>
    <label for="myText">Text: </label>
    <input id="myText" type="text" name="text"
           pattern="[a-zA-Z0-9]+" required />
    <input type="submit" id="submit" />
</form>
<script>
    var text = document.getElementById("myText");
    text.addEventListener("keyup", function(){
        if(this.validity.patternMismatch){
            this.setCustomValidity("Pattern doesn't match.");
        } else {
            this.setCustomValidity("");
        }
    });
</script>
</body></html>
```

Assignment 6

- **Topic: Form validation with JavaScript an HTML5**
- Due in: 2 Weeks
- Due date: 01.12.2014 16:00h



Thanks!

What are your questions?