PeopleCert
Software Development Skills

JavaScript Stream

Lesson 15

Study Guide

Copyright Details

The contents of this workshop are protected by copyright and can be reproduced under the Terms of Use agreed between PeopleCert and the ATO using this material only.

Material in this presentation has been sourced from the bibliography listed in the certification's Syllabus. All software-related images are used for educational purposes only and may differ across time.

No part of this document may be reproduced in any form without the written permission of PeopleCert International Ltd. Permission can be requested at www.peoplecert.org.

e-mail: info@peoplecert.org, www.peoplecert.org

- Document Version: PC-SDS_SG 1.0 | February 2020

PeopleCert
All talents, certified.

PeopleCert: A Global Leader in Certification

- ✓ **Web** & **Paper based** exams in **25** languages
- ✓ **2,500 Accredited Training Organisations** worldwide
- ✓ Delivering exams across **200 countries every year**
- ✓ Comprehensive Portfolio of **500+ Exams** and Growing

AXELOS GLOBAL BEST PRACTICE · PeopleCert DevOps. · digital marketing institute · IASSC · ISACA · Language Cert · LEAN IT ASSOCIATION · Quality Software Development · ECDL

PeopleCert
All talents, certified.

How to Use This Document

This document is your **PeopleCert Software Developer Skills Study Guide** to help you prepare for the **PeopleCert Software Developer Skills Foundation & Advanced examination**.

It is meant to provide you with a clear outline of everything covered in the course presentation by your instructor that will be on the PeopleCert Software Developer Skills Foundation & Advanced exams.

Your exams will be closed book. You will be given 120 minutes to complete it. It contains 100 multiple choice questions and to pass the exam you must achieve a grade of 65% or higher, or a minimum of 65/100 correct responses. For further details on your exam, including more information on question types and learning objectives, please refer to your course syllabus.

As you follow along, you may see that some material here is not replicated in the trainer presentation. This study guide includes questions, activities, knowledge checks, or other material in the presentation that are facilitated verbally by the instructor. It also does not contain content that is not examinable, but instead is designed to reinforce learning or add value to your course experience. It also provides valuable links and references, throughout the slides, which you can explore further to enhance your learning and understanding of the material provided in the study guide.

PeopleCert
All talents, certified.

Coding Bootcamp

Lesson 15

Web Design and Development Fundamentals (Front End)

Objectives

- JavaScript Functions
- Object Prototypes
- Exception Handling
- Document Object Model
- JavaScript Events
- JSON
- XML Syntax and Validation
- Data Validation

Syllabus Items:

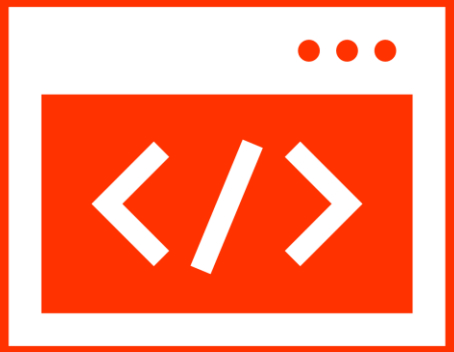
- 4.6.1 Code in JavaScript, variables, functions
- 4.6.2 Code using advance java script: if conditions, loops

Syllabus

Category	Topic	Task
FSD_4 Web Design and Development Fundamentals (Front-End)	**4.6 JavaScript/ jQuery**	4.6.1 Code in JavaScript, variables, functions
		4.6.2 Code using advance java script: if conditions, loops
		4.6.3 Improve usability of forms, validate data from the user

Contents | Learning Objectives

- ✓ Introduction to JavaScript
- ✓ Client-Side Scripting
- ✓ Language Syntax
- ✓ Variables
- ✓ Strings
- ✓ Numbers
- ✓ Booleans
- ✓ Operators
- ✓ Conditionals
- ✓ Loops
- ✓ Arrays
- ✓ Objects
- ✓ Understand the JavaScript functions and the different ways that they can be defined and called
- ✓ Understand the object prototypes
- ✓ Familiarize with the exception handling
- ✓ Understand the Document Object Model and the selection process
- ✓ Familiarize with the JavaScript events
- ✓ Understand the JSON and XML model
- ✓ Learn how to validate the form data



JavaScript Events

JavaScript Events

Definition: A JavaScript **event** is an action that can be detected by JavaScript

- many of them are initiated by user actions
- some are generated by the browser itself

- We say that an event is triggered and then it is handled by JavaScript functions
- JavaScript's interaction with HTML is handled through events that occur when the user or the browser manipulates a page.
- When the page loads, it is called an event. When the user clicks a button, that click too is an event. Other examples include events like pressing any key, closing a window, resizing a window, etc.
- Developers can use these events to execute JavaScript coded responses, which cause buttons to close windows, messages to be displayed to users, data to be validated, and virtually any other type of response imaginable.
- Events are a part of the Document Object Model (DOM) Level 3 and every HTML element contains a set of events which can trigger JavaScript Code.

Source: https://www.tutorialspoint.com/javascript/javascript_events.htm
Source: https://www.w3schools.com/js/js_events.asp

Event Handlers

- **Event handlers** can be used to handle events. They can verify user input, user actions, and browser actions:
  - Things that should be done every time a page loads
  - Things that should be done when the page is closed
  - Action that should be performed when a user clicks a button
  - Content that should be verified when a user inputs data
- Many different **methods** can be used to let JavaScript work with events:
  - HTML event attributes can execute JavaScript code directly
  - HTML event attributes can call JavaScript functions
  - You can assign your own event handler functions to HTML elements
  - You can prevent events from being sent or being handled

Source: https://www.w3schools.com/js/js_events.asp

Event-Handling Approaches (1)

- **Inline Hook**

```
…
<script type="text/javascript" src="inline.js"></script>
…
<form name='mainform' onsubmit="validate(this);">
<input name="name" type="text"
   onchange="check(this);"
   onfocus="highlight(this, true);"
   onblur="highlight(this, false);">
…
```

inline.js

```
function validate(node) {
   …
}
function check(node) {
   …
}
function highlight(node,
state) {
   …
}
```

- **Event Property Approach**

```
var myButton = document.getElementById('example');
myButton.onclick = alert('some message');
```

Source: https://www.w3schools.com/js/js_events.asp

Event-Handling Approaches (2)

- **Event Listener Approach**

```
var myButton = document.getElementById('example');
myButton.addEventListener('click', alert('some message'));
myButton.addEventListener('mouseout', funcName);
```

- **Event Listener Approach (anonymous function)**

```
myButton.addEventListener('click', function() {
        var d = new Date();
        alert("You clicked this on "+ d.toString());
});
```

Source: https://www.w3schools.com/js/js_events.asp

removeEventListener() method

- removeEventListener() method removes an event handler that has been attached with the addEventListener() method.
- To remove event handlers, the function specified with the addEventListener() method must be an external function.
- Anonymous functions, like "element.removeEventListener("event", function(){ myScript });" will not work.
- Example

```
var x = document.getElementById("myButton");
myButton.addEventListener('mouseout', funcName);

if (x.removeEventListener) {
  x.removeEventListener("mouseout", funcName);
}
```

Source: https://www.w3schools.com/jsref/met_element_removeeventlistener.asp

multiple handlers

- We can add many event handlers to one element.
- We can add many event handlers of the same type to one element, i.e two "click" events.
- We can add event listeners to any DOM object not only HTML elements. i.e the window object.
- Example

```
<input id="elem" type="button" value="Click me"/>

<script>
 function handler1() {
  alert('Thanks!');
 };
 function handler2() {
  alert('Thanks again!');
 }
 elem.onclick = () => alert("Hello");
 elem.addEventListener("click", handler1); // Thanks!
 elem.addEventListener("click", handler2); // Thanks again!
</script>
```

Source: https://javascript.info/introduction-browser-events

Class Exercise/Discussion #1

- Create a simple toggle button in javascript. The button will turn "OFF" and will turn back "ON"

Event Object

- When an event is triggered, the browser will construct an event object that contains information about the event.

```
div.addEventListener('click', function(e) {
        // find out where the user clicked
        var x = e.clientX;
        …
```

- bubbles
  - Indicates whether the event bubbles up through the DOM
- cancelable
  - Indicates whether the event can be cancelled
- target
  - The object that generated (or dispatched) the event
- type
  - The type of the event

Event Object Properties (full list):
https://developer.mozilla.org/en-US/docs/Web/API/Event#Properties

preventDefault() method

- The preventDefault() method cancels the event if it is cancelable, meaning that the default action that belongs to the event will not occur.
- Clicking on a "Submit" button, prevent it from submitting a form:

```
<form action="https://yahoo.com">
      <input type="submit">
   </form>

 const form = document.querySelector("form");
      form.addEventListener("submit", function(e){
         e.preventDefault();
         alert("stay here!!")
      })
```

Source: https://www.w3schools.com/jsreF/obj_event.asp

Event Types

- Mouse Events
- Click Events
- Touch Events
- Form Events
- Frame Events

Full list of Events at MDN:
https://developer.mozilla.org/en-US/docs/Web/Events

Mouse Event Types

- **click**
  - The mouse was clicked on an element
- **dblclick**
  - The mouse was double clicked on an element
- **mousedown**
  - The mouse was pressed down over an element
- **mouseup**
  - The mouse was released over an element
- **mouseover**
  - The mouse was moved (not clicked) over an element
- **mouseout**
  - The mouse was moved off of an element
- **mousemove**
  - The mouse was moved while over an element

Source: https://www.w3schools.com/js/js_events.asp

Click Event Types

- keydown
  - The user is pressing a key (this happens first)
- keyup
  - The user releases a key that was down (this happens last)
- Keypress (deprecated)
  - The user presses a key (this happens after keydown)

Source: https://www.w3schools.com/js/js_events.asp

Class Exercise/Discussion #2

- Create a h1 element that will change color on mouseover event

Touch Event Types

- Touch events are a new category of events that can be triggered by devices with touch screens
- The different events (e.g., touchstart, touchmove, and touchend) are analogous to some of the mouse events (mousedown, mousemove, and mouseup)

Form Event Types

- **blur**
  - When an element has lost focus
- **change**
  - When a value in an element has changed by the user
- **focus**
  - When an element has received focus
- **reset**
  - When a form is reset
- **select**
  - When some text is being selected
- **input**
  - fires when the value of an <input>, <select>, or <textarea> element has been changed (change vs input events)
- **submit**
  - When the form is submitted

Source: https://www.w3schools.com/js/js_events.asp

Class Exercise/Discussion #3

- Validate an email by using the following criteria:
  - email id must contain the @ and . character
  - There must be at least one character before and after the @.
  - There must be at least two characters after . (dot).

Frame Event Types

- abort
  - An object was stopped from loading
- error
  - An object or image did not properly load
- load
  - When a document or object has been loaded
- resize
  - The document view was resized
- scroll
  - The document view was scrolled
- unload
  - The document has unloaded

Source: https://www.w3schools.com/js/js_events.asp

Class Exercise/Discussion #4

- Create a div with scrollbar and show popup message when is being scrolled

What is JSON?

- JSON: JavaScript Object Notation.
- JSON is a syntax for storing and exchanging data.
- JSON is text, written with JavaScript object notation.
- Definition:
- JSON is text, written with JavaScript object notation
  - When exchanging data between a browser and a server, the data can only be text
  - We can convert any JavaScript object into JSON, and send JSON to the server, also convert any JSON received from the server into JavaScript objects
  - This way we can work with the data as JavaScript objects, with no complicated parsing and translations

```
{
    "name":"John",
    "age":30,
    "cars": {
        "car1":"Ford",
        "car2":"BMW",
        "car3":"Fiat"
    }
}
```

- JSON syntax is a subset of the JavaScript syntax
- JSON data is written as name/value pairs
  - Keys (names) must be strings, written with double quotes
  - Values must be one of the following data types: string, number, object (JSON object), array, boolean, null
  - In JavaScript, keys can be strings, numbers, or identifier names
  - In JavaScript values can be all of the above, plus any other valid JavaScript expression, including function, date and undefined
- The file type for JSON files is ".json"
- The MIME type for JSON text is "application/json“
- Both JSON and XML can be used to receive data from a web server

Source: https://www.w3schools.com/js/js_json_intro.asp

JSON Sending Data

- If you have data stored in a JavaScript object, you can convert the object into JSON, and send it to a server:

```
var myObj = { "name":"John", "age":31, "city":"New York" };
var myJSON = JSON.stringify(myObj);
window.location = "demo_json.php?x=" + myJSON;
```

JSON Syntax Rules

- The JSON syntax is a subset of the JavaScript syntax
- JSON syntax is derived from JavaScript object notation syntax:
  - Data is in name/value pairs
  - Data is separated by commas
  - Curly braces hold objects
  - Square brackets hold arrays
- In JSON, values must be one of the following data types:
  - a string
  - a number
  - an object (JSON object)
  - an array
  - a boolean
  - null

Source: https://www.w3schools.com/js/js_json_intro.asp

Class Exercise/Discussion #5

- From link (https://restcountries.eu/rest/v2/name/) we will get data using json and parse it into page

XML

Definition:

- XML stands for eXtensible Markup Language.
  - Designed to store and transport data.
  - Designed to be both human- and machine-readable.
  - Does not carry any information about how to be displayed.
  - Used in many aspects of web development.
  - Used to separate data from presentation.
- XML documents form a tree structure that starts at "the root" and branches to "the leaves".

```xml
<?xml version="1.0" encoding="UTF-8"?>
<note>
  <to>Tove</to>
  <from>Jani</from>
  <heading>Reminder</heading>
  <body>Don't forget me this weekend!</body>
</note>
```

Source: https://www.w3schools.com/xml/default.asp

XML Syntax Rules

- XML can contain the **prolog** line, that often defines XML document version and encoding.

```
<?xml version="1.0" encoding="UTF-8"?>
```

- XML documents must contain one **root** element that is the **parent** of all other elements.
  - All XML Elements Must Have a Closing Tag
  - XML Elements Must be Properly Nested
  - XML Tags are Case Sensitive
- XML elements can have attributes, just like HTML.
  - Attributes are designed to contain data related to a specific element.

```
<person gender="female">
  <firstname>Anna</firstname>
  <lastname>Smith</lastname>
</person>
```

- XML Namespaces provide a method to avoid element name conflicts.

```
<root>
  <h:table xmlns:h="http://www.w3.org/TR/html4/">
  </h:table>

  <f:table xmlns:f="https://www.w3schools.com/furniture">
  </f:table>
</root>
```

Source: https://www.w3schools.com/xml/default.asp

XML Validation

- An XML document with correct syntax is called "Well Formed".
- An XML document validated against a DTD is both "Well Formed" and "Valid".
  - The purpose of a DTD is to define the structure of an XML document.
  - It defines the structure with a list of legal elements.

```
<!DOCTYPE note
[
<!ELEMENT note (to,from,heading,body)>
<!ELEMENT to (#PCDATA)>
<!ELEMENT from (#PCDATA)>
<!ELEMENT heading (#PCDATA)>
<!ELEMENT body (#PCDATA)>
]>
```

- An XML Schema (XSD) describes the structure of an XML document, just like a DTD.
  - XML Schema is an XML-based alternative to DTD.

```
<xs:element name="note">

<xs:complexType>
  <xs:sequence>
    <xs:element name="to" type="xs:string"/>
    <xs:element name="from" type="xs:string"/>
    <xs:element name="heading" type="xs:string"/>
    <xs:element name="body" type="xs:string"/>
  </xs:sequence>
</xs:complexType>

</xs:element>
```

Source: https://www.w3schools.com/xml/default.asp

XML: XSLT

- XSL (eXtensible Stylesheet Language) is a styling language for XML.
- XSLT stands for XSL Transformations
- XSLT is the most important part of XSL
- XSLT transforms an XML document into another XML document
- XSLT uses XPath to navigate in XML documents
- XSLT is a W3C Recommendation
- An XSL style sheet consists of one or more set of rules that are called templates. A template contains rules to apply when a specified node is matched.

```
<?xml version="1.0" encoding="UTF-8"?>

<xsl:stylesheet version="1.0"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform">

<xsl:template match="/">
  <html>
  <body>
  <h2>My CD Collection</h2>
  <table border="1">
    <tr bgcolor="#9acd32">
      <th>Title</th>
      <th>Artist</th>
    </tr>
    <xsl:for-each select="catalog/cd">
    <tr>
      <td><xsl:value-of select="title"/></td>
      <td><xsl:value-of select="artist"/></td>
    </tr>
    </xsl:for-each>
  </table>
  </body>
  </html>
</xsl:template>

</xsl:stylesheet>
```

Source: https://www.w3schools.com/xml/default.asp

XML Parser

- The XML DOM (Document Object Model) defines the properties and methods for accessing and editing XML.
- However, before an XML document can be accessed, it must be loaded into an XML DOM object.
- All modern browsers have a built-in XML parser that can convert text into an XML DOM object.
- Example

```
<p id="demo"></p>
<script>
var text, parser, xmlDoc;
text = "<bookstore><book>" +
"<title>Everyday Italian</title>" +
"<author>Giada De Laurentiis</author>" +
"<year>2005</year>" +
"</book></bookstore>";

parser = new DOMParser();
xmlDoc = parser.parseFromString(text,"text/xml");
document.getElementById("demo").innerHTML =
xmlDoc.getElementsByTagName("title")[0].childNodes[0].nodeValue;
</script>
```

Source: https://www.w3schools.com/xml/xml_parser.asp

Form Validation

- Sending data is not enough — we also need to make sure that the data users fill out in forms is in the correct format we need to process it successfully, and that it won't break our applications.
- We also want to help our users to fill out our forms correctly and not get frustrated when trying to use our apps.
- Form validation helps us achieve these goals.

Definition:

Form validation *normally used to occur at the server, after the client had entered all the necessary data and then pressed the Submit button.*

- Go to any popular site with a registration form, and you will notice that they give you feedback when you don't enter your data in the format they are expecting.
- You'll get messages like:
  - "This field is required" (you can't leave this field blank)
  - "Please enter your phone number in the format xxx-xxxx" (it wants three numbers followed by a dash, followed by four numbers)
  - "Please enter a valid e-mail address" (the thing you've entered doesn't look like a valid e-mail address)
  - "Your password needs to be between 8 and 30 characters long, and contain one uppercase letter, one symbol, and a number" (seriously?)

Sources: https://www.tutorialspoint.com/javascript/javascript_form_validations.htm
https://developer.mozilla.org/en-US/docs/Learn/HTML/Forms/Form_validation

Form Validation

- **Form Validation** is when you enter data and the web application checks it to see if it is correct.
- If so, it allows it to be submitted to the server and (usually) saved in a database; if not, it gives you error messages to explain what you've done wrong (provided you've done it right).
- Form validation can be implemented in a number of different ways.
- We want to make filling out web forms as non-horrible as possible, so why do we insist on blocking our users at every turn? There are **three** main reasons:
  - Get the right data, in the right format — our applications won't work properly if our user's data is stored in any old format they like, or if they don't enter the correct information in the correct places, or omit it altogether.
  - Protect our users — if they entered really easy passwords, or no password at all, then malicious users could easily get into their accounts and steal their data.
  - Protect ourselves — there are many ways that malicious users can misuse unprotected forms to damage the application they are part of.

There are **different types of form validation** that you'll encounter on the web:

- **Client-side validation** is validation that occurs in the browser, before the data has been submitted to the server. This is more user-friendly than server-side validation as it gives an instant response. This can be further subdivided:
  - **JavaScript** validation is coded using JavaScript. It is completely customizable.
  - **Built in form validation** is done with HTML5 form validation features, and generally doesn't require JavaScript. This has better performance, but it is not as customizable.
- **Server-side validation** is validation that occurs on the server, after the data has been submitted — server-side code is used to validate the data before it is put into the database, and if it is wrong a response is sent back to the client to tell the user what went wrong. Server-side validation is not as user-friendly as client-side validation, as it requires a round trip to the server, but it is essential — it is your application's last line of defense against bad (meaning incorrect, or even malicious) data. All popular server-side frameworks have features for validating and sanitizing data (making it safe).
- In the real world, developers tend to use a combination of client-side and server-side validation, to be on the safe side.

Source: https://developer.mozilla.org/en-US/docs/Learn/HTML/Forms/Form_validation

Data Validation

Definition:

Data validation is the process of ensuring that user input is clean, correct, and useful.

Typical cases:

- has the user filled in all required fields?
- has the user entered a valid date?
- has the user entered text in a numeric field?

- Data validation can occur on both the Server and the Client.
- **Server-side validation** is performed by a web server, after input has been sent to the server.
- **Client-side validation** is performed by a web browser, before input is sent to a web server.

✓ NOTE: both validations should take place, for security purposes.

In many cases, data are sent to the server via HTML Forms.

- However, validation can be crucial on data.
- Validation can (and should) occur both on the server and the client.

HTML form validation can be done by JavaScript

- JavaScript Can Validate Text Input
- JavaScript Can Validate Numeric Input
- HTML form validation can also be performed automatically by the browser
  - If a form field is empty, the **required** attribute prevents this form from being submitted:

```
function validateForm() {
    var x = document.forms["myForm"]["fname"].value;
    if (x == "") {
        alert("Name must be filled out");
        return false;
    }
}
```

```
<form name="myForm" action="/action_page.php" onsubmit="return validateForm()" method="post">
Name: <input type="text" name="fname">
<input type="submit" value="Submit">
</form>
```

Source: https://www.w3schools.com/js/js_validation.asp

Class Activity | HTML built-in validation Example

- https://developer.mozilla.org/en-US/docs/Learn/HTML/Forms/Form_validation

Validating forms using JavaScript (1)

CONSTRAINT VALIDATION API PROPERTIES

Property	Description
validationMessage	A localized message describing the validation constraints that the control does not satisfy (if any), or the empty string if the control is not a candidate for constraint validation (willValidate is false), or the element's value satisfies its constraints.
validity	A **ValidityState** object describing the validity state of the element.
validity.customError	Returns true if the element has a custom error; false otherwise.
validity.patternMismatch	Returns true if the element's value doesn't match the provided pattern; false otherwise. If it returns true, the element will match the **:invalid** CSS pseudo-class.
validity.rangeOverflow	Returns true if the element's value is higher than the provided maximum; false otherwise. If it returns true, the element will match the **:invalid** and **:out-of-range** and CSS pseudo-class.
validity.rangeUnderflow	Returns true if the element's value is lower than the provided minimum; false otherwise. If it returns true, the element will match the **:invalid** and **:out-of-range** CSS pseudo-class.
validity.stepMismatch	Returns true if the element's value doesn't fit the rules provided by the step attribute; otherwise false . If it returns true, the element will match the **:invalid** and **:out-of-range** CSS pseudo-class.

Source: https://developer.mozilla.org/en-US/docs/Learn/HTML/Forms/Form_validation

Validating forms using JavaScript (2)

CONSTRAINT VALIDATION API PROPERTIES

Property	Description
validity.tooLong	Returns true if the element's value is longer than the provided maximum length; else it will be false If it returns true, the element will match the **:invalid** and **:out-of-range** CSS pseudo-class.
validity.typeMismatch	Returns true if the element's value is not in the correct syntax;otherwise false. If it returns true, the element will match the **:invalid** css pseudo-class.
validity.valid	Returns true if the element's value has no validity problems; false otherwise. If it returns true, the element will match the **:valid** css pseudo-class; the **:invalid** CSS pseudo-class otherwise.
validity.valueMissing	Returns true if the element has no value but is a required field; false otherwise. If it returns true, the element will match the **:invalid** CSS pseudo-class.
willValidate	Returns true if the element will be validated when the form is submitted; false otherwise.

Source: https://developer.mozilla.org/en-US/docs/Learn/HTML/Forms/Form_validation

Validating forms using JavaScript (3)

CONSTRAINT VALIDATION API METHODS

Method	Description
checkValidity()	Returns true if the element's value has no validity problems; false otherwise. If the element is invalid, this method also causes an **invalid** event at the element.
setCustomValidity(message)	Adds a custom error message to the element; if you set a custom error message, the element is considered to be invalid, and the specified error is displayed. This lets you use JavaScript code to establish a validation failure other than those offered by the standard constraint validation API. The message is shown to the user when reporting the problem. If the argument is the empty string, the custom error is cleared.

Source: https://developer.mozilla.org/en-US/docs/Learn/HTML/Forms/Form_validation

Class Example – Using the Constraint Validation API

- HTML

```
<form novalidate>
  <p>
    <label for="mail">
      <span>Please enter an email address:</span>
      <input type="email" id="mail" name="mail">
      <span class="error" aria-live="polite"></span>
    </label>
  </p>
  <button>Submit</button>
</form>
```

- CSS

JS-L2_CSS_ClassExercise.cs

Class Example – Using the Constraint Validation API

- JavaScript code handles the custom error validation

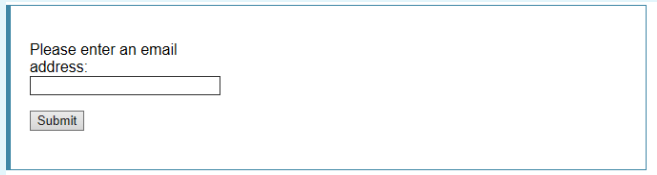
```
// There are many ways to pick a DOM node; here we get the form itself and the email
// input box, as well as the span element into which we will place the error message.

var form  = document.getElementsByTagName('form')[0];
var email = document.getElementById('mail');
var error = document.querySelector('.error');

email.addEventListener("input", function (event) {
  // Each time the user types something, we check if the
  // email field is valid.
  if (email.validity.valid) {
    // In case there is an error message visible, if the field
    // is valid, we remove the error message.
    error.innerHTML = ""; // Reset the content of the message
    error.className = "error"; // Reset the visual state of the message
  }
}, false);
form.addEventListener("submit", function (event) {
  // Each time the user tries to send the data, we check
  // if the email field is valid.
  if (!email.validity.valid) {

    // If the field is not valid, we display a custom
    // error message.
    error.innerHTML = "I expect an e-mail, darling!";
    error.className = "error active";
    // And we prevent the form from being sent by canceling the event
    event.preventDefault();
  }
}, false);
```

- Here is the live result



- Apply the Class example above to one of your pages (or create a new simple form page)

Summary on Form validation

- Form validation does not require complex JavaScript, but it does require thinking carefully about the user.
- Always remember to help your user to correct the data he provides. To that end, be sure to:
  - Display explicit error messages.
  - Be permissive about the input format.
  - Point out exactly where the error occurs (especially on large forms).

Drag'n'Drop

- Drag’n’Drop is a great interface solution. Taking something and dragging and dropping it is a clear and simple way to do many things, from copying and moving documents (as in file managers) to ordering (dropping items into a cart).
- In the modern HTML standard there’s a section about Drag and Drop with special events such as dragstart, dragend, and so on.
- These events allow us to support special kinds of drag’n’drop, such as handling dragging a file from OS file-manager and dropping it into the browser window. Then JavaScript can access the contents of such files.
- But native Drag Events also have limitations. For instance, we can’t prevent dragging from a certain area. Also we can’t make the dragging “horizontal” or “vertical” only. And there are many other drag’n’drop tasks that can’t be done using them. Also, mobile device support for such events is very weak.

Drag'n'Drop algorithm

- The basic Drag’n’Drop algorithm looks like this:
  - On mousedown – prepare the element for moving, if needed (maybe create a clone of it, add a class to it or whatever).
  - Then on mousemove move it by changing left/top with position:absolute.
  - On mouseup – perform all actions related to finishing the drag’n’drop.

Source: https://javascript.info/mouse-drag-and-drop

Drag'n'Drop – Example(1)

```
<p>Drag the ball.</p>
<img src="https://en.js.cx/clipart/soccer-gate.svg" id="gate" class="droppable">
<img src="https://en.js.cx/clipart/ball.svg" id="ball">

<script>
  let currentDroppable = null;

  ball.onmousedown = function(event) {

    let shiftX = event.clientX - ball.getBoundingClientRect().left;
    let shiftY = event.clientY - ball.getBoundingClientRect().top;

    ball.style.position = 'absolute';
    ball.style.zIndex = 1000;
    document.body.append(ball);

    moveAt(event.pageX, event.pageY);

    function moveAt(pageX, pageY) {
      ball.style.left = pageX - shiftX + 'px';
      ball.style.top = pageY - shiftY + 'px';
    }
```

Source: https://javascript.info/mouse-drag-and-drop

Drag'n'Drop – Example(2)

```
  function onMouseMove(event) {
    moveAt(event.pageX, event.pageY);

    ball.hidden = true;
    let elemBelow = document.elementFromPoint(event.clientX, event.clientY);
    ball.hidden = false;

    if (!elemBelow) return;

    let droppableBelow = elemBelow.closest('.droppable');
    if (currentDroppable != droppableBelow) {
      if (currentDroppable) { // null when we were not over a droppable before this event
        leaveDroppable(currentDroppable);
      }
      currentDroppable = droppableBelow;
      if (currentDroppable) { // null if we're not coming over a droppable now
        // (maybe just left the droppable)
        enterDroppable(currentDroppable);
      }
    }
  }

  document.addEventListener('mousemove', onMouseMove);

  ball.onmouseup = function() {
    document.removeEventListener('mousemove', onMouseMove);
    ball.onmouseup = null;
  };

};

function enterDroppable(elem) {
  elem.style.background = 'pink';
}

function leaveDroppable(elem) {
  elem.style.background = '';
}

ball.ondragstart = function() {
  return false;
};
```

Source: https://javascript.info/mouse-drag-and-drop

It is time for a Knowledge Check!

Sample Questions

#1 The type that specifies what kind of event occurred is _____.

A. event type
B. even target
C. both event type and even target
D. Interface

#2 In general, event handler is nothing but _____.

A. function
B. interface
C. event
D. handler

#3 When will the browser invoke the handler?

A. Program begins
B. Any event occurs
C. Specified event occurs
D. Webpage loads

#4 Which event occurs when the user clicks on an HTML element?

A. onclick
B. onchange
C. onmouseover
D. onmouseclick

Any Questions?

Self Study

- https://developer.mozilla.org/en-US/docs/Web/JavaScript/Guide/Details_of_the_Object_Model
- https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Statements/try...catch#Nested_try-blocks
- https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Statements/try...catch
- https://developer.mozilla.org/en-US/docs/Web/Events
- http://www.javascriptkit.com/jsref/events.shtml
- https://www.w3schools.com/js/js_htmldom_events.asp
- https://developer.mozilla.org/en-US/docs/Learn/HTML/Forms/Form_validation

References / Further Reading

- https://www.w3schools.com/js/
- https://www.w3schools.com/xml/default.asp
- https://www.w3schools.com/xml/xsl_intro.asp
- https://developer.mozilla.org/en-US/docs/Web/JavaScript/Guide/Functions
- https://developer.mozilla.org/en-US/docs/Web/JavaScript/Guide/Control_flow_and_error_handling
- https://www.tutorialspoint.com/javascript/javascript_events.htm
- https://www.w3schools.com/js/js_events.asp
- https://www.w3schools.com/js/js_json_syntax.asp
- https://developer.mozilla.org/en-US/docs/Learn/HTML/Forms/Form_validation
- https://developer.mozilla.org/en-US/docs/Web/JavaScript/Guide/Working_with_Objects
- https://docs.oracle.com/cd/E19957-01/816-6409-10/obj2.htm
- https://developer.mozilla.org/en-US/docs/Learn/Tools_and_testing/Cross_browser_testing/JavaScript
- Randy Connolly and Ricardo Hoar: Fundamentals of Web Development
- https://developer.mozilla.org/en-US/docs/Web/JavaScript/Guide/Details_of_the_Object_Model
- https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Statements/try...catch#Nested_try-blocks
- https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Statements/try...catch
- https://developer.mozilla.org/en-US/docs/Web/Events
- http://www.javascriptkit.com/jsref/events.shtml
- https://www.w3schools.com/js/js_htmldom_events.asp
- https://developer.mozilla.org/en-US/docs/Learn/HTML/Forms/Form_validation

PeopleCert Values Your Feedback

Like the course?
Have something to say?

Send us your comments at:
academic@peoplecert.org

Thank You!

For latest news and updates follow us.

linkedin.com/company/peoplecert-group/
twitter.com/peoplecert
youtube.com/channel/UCLBidKZS9Xk08f_PM5Edtfg
facebook.com/peoplecert.org