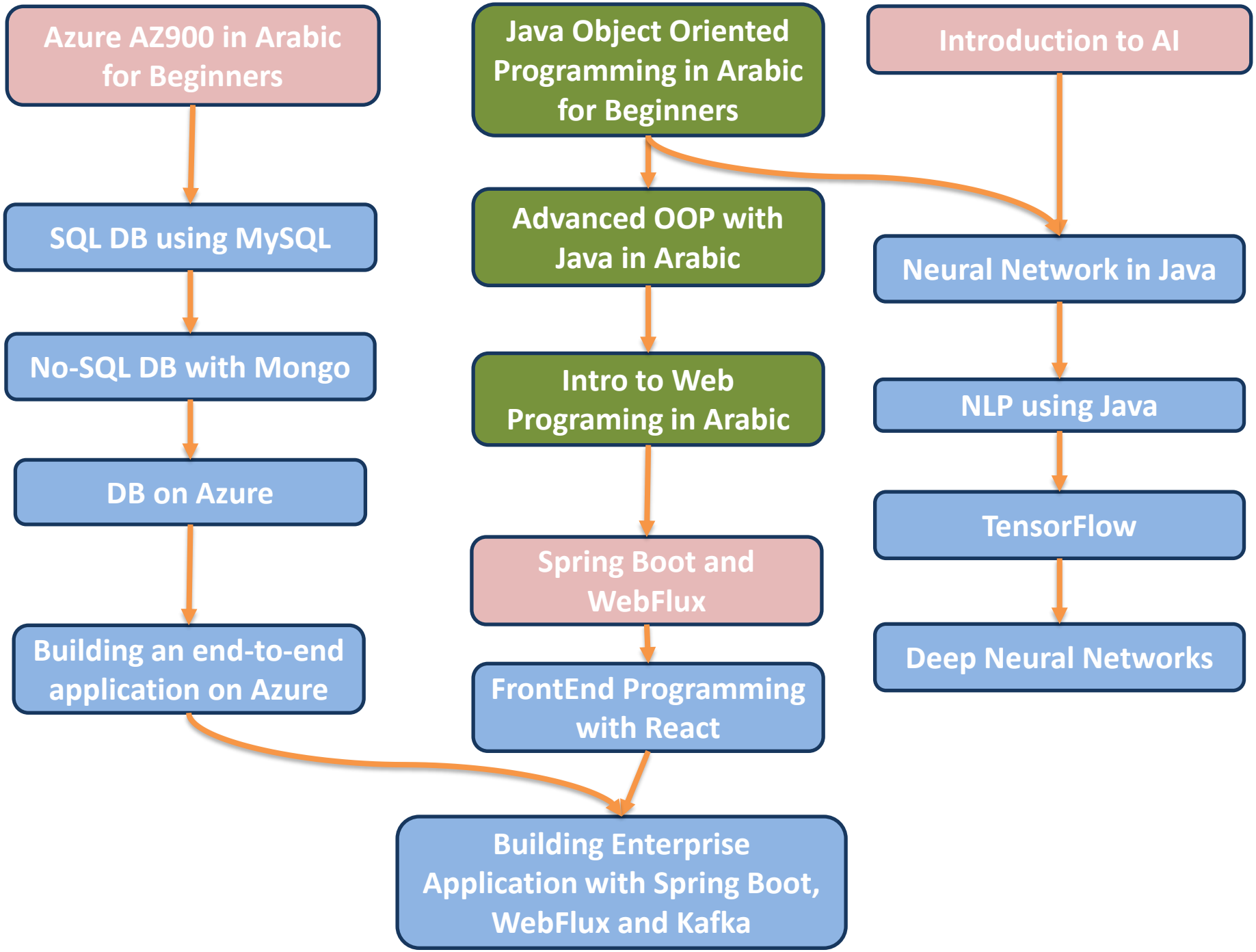
Azure AZ900 in Arabic for Beginners
SQL DB using MySQL
No-SQL DB with Mongo
DB on Azure
Building an end-to-end application on Azure
Java Object Oriented Programming in Arabic for Beginners
Advanced OOP with Java in Arabic
Intro to Web Programing in Arabic
Spring Boot and WebFlux
FrontEnd Programming with React
Introduction to AI
Neural Network in Java
NLP using Java
TensorFlow
Deep Neural Networks
Building Enterprise Application with Spring Boot, WebFlux and Kafka

DOM and Validation

Dr. Magdi AMER

Document Object Model

Basic DOM manipulation

- To get an object by ID
  - <a **id=“tom" href="/name/nm0000297/">Tom Jerry**
  - var target = **document.getElementById("tom");**
  - **Target will be null if no such element**
- To get the name of an element:
  - target**.nodeName➔ (In this case: a)**
- To get an object by tag
  - var listItems = **document.getElementsByTagName("li");**
  - var numItems = listItems**.length; ➔ get the length of the list**
  - **Returns empty list if no element (length = 0)**
- To restrict search tree
  - **obj.getElement…. ➔ looks only in element sub-tree.**

- To get all the elements in a document:

```
var elementArray = [];
if (typeof document.all != "undefined")
{ elementArray = document.all; }  // for IE
else
{ elementArray = document.getElementsByTagName("*"); } //IE 5.x does not support this
```

- To change the style of an element

```
var target = document.getElementById('a1');
var display= target.style.display;
if(display=="none"){
   target.style.display="";
}else {
   target.style.display="none";
}
```

Navigating DOM tree

- Finding a parent
  - var paragraph = element**.parentNode;**
- Finding children
  - var list= element**.childNodes;**
  - var first= element**.firstChild;**
  - var last= element**.lastChild;**
- Finding Siblings
  - var next= element**.nextSibling;**
  - var previous= element**.previousSibling;**

Getting Attributes

Let's all hug Koko

var koko = document.getElementById("koko");

var kokoHref = koko**.getAttribute("href");**

- **This has an unpredictable behavior in many browsers.**

  Use this instead

var kokoHref = koko**.href;**

- **Works for all attribute except class and for, which are accessed using** *element.className* ***And*** *label_element.htmlFor*
- **To set an attribute**
  - koko**.setAttribute("href", "/koko/");**

Handling Styles

- Changing color
  - var scarlet = document.getElementById("scarlet");
  - scarlet**.style.color = "#FF0000";**
  - scarlet**.style.backgroundColor = "#000066";**

- **Note:** To validate your HTML use:
  - **https://validator.w3.org/**

Handling Events

- ```
  <form name="myForm2">
  <input
     type="button"
     name="myButton"
     value="Press Me 2"
     onClick="alert('myButton was pressed')">
  </form>
  ```

- ```
  <!DOCTYPE html>
  <html lang="en">
  <head>
    <meta charset="UTF-8">
    <title>Title</title>
  </head>
  <body>
  <form name="myForm">
    <input type="button" name="myButton" value="Press Me">
  </form>
  <script type="text/javascript">
    document.myForm.myButton.onclick = function() { alert('myButton was pressed'); };
  </script>

  </body>
  </html>
  ```

Abort	onAbort
Blur	onBlur
Change	onChange
Click	onClick
DblClick	onDblClick
DragDrop	onDragDrop
Error	onError
Focus	onFocus
KeyDown	onKeyDown
KeyPress	onKeyPress
KeyUp	onKeyUp
Load	onLoad
MouseDown	onMouseDown
MouseMove	onMouseMove
MouseOut	onMouseOut
MouseOver	onMouseOver
MouseUp	onMouseUp
Move	onMove
Reset	onReset
Resize	onResize
Select	onSelect
Submit	onSubmit
Unload	onUnload

Text Properties and Methods

Table 8.37 Event Handlers, Methods, and Properties Used by the `Text` *Object*

Type	Item	Description
Event Handler	`onBlur`	Executes code when the text box loses the focus.
	`onChange`	Executes code when the text box loses the focus and has had its value modified.
	`onFocus`	Executes code when the text box receives the focus.
	`onSelect`	Executes code when a user selects some of the text within the text box.
Method	`blur()`	Removes the focus from the text box.
	`focus()`	Gives the focus to the text box.
	`handleEvent()`	Invokes the handler for the event specified and was added in JavaScript 1.2.
	`select()`	Selects the text in the text box.
	`unwatch()`	Used to turn off the watch for a particular property.
	`watch()`	Used to turn on the watch for a particular property.
Property	`defaultValue`	Returns the value of the text box specified by the `value` attribute. Note that this property is not supported by the Opera browsers.
	`form`	Returns the entire form the text box is in.
	`name`	Returns the name of the text box specified by the `name` attribute.
	`type`	Returns the type of the text box specified by the `type` attribute. Note that this is always `text` and was added in JavaScript 1.1.
	`value`	Returns the value that is actually displayed in the text box.

Password Properties and Methods

Table 8.25 Properties, Methods and Events of the `Password` *Object*

Type	Item	Description
Property	`defaultValue`	Refers to the `value` attribute of the HTML password box
	`form`	Refers to the form that contains the password box
	`name`	Refers to the `name` attribute of the HTML password box
	`type`	Refers to the `type` attribute of the HTML password box
	`value`	Refers to the current contents of the password box
Method	`blur()`	Removes focus from the password box
	`focus()`	Sets focus to the password box
	`handleEvent()`	Invokes the event handler
	`select()`	Selects the text entered in the password box
	`unwatch()`	Removes a watchpoint on a `Password` property
	`watch()`	Sets a watchpoint on a `Password` property
Event	`onBlur`	Event handler used when the focus is removed from the password box
	`onFocus`	Event handler used when the focus is put on the password box

Hidden Properties and Methods

Table 8.16 Properties and Methods of the `Hidden` *Object*

Type	Item	Description
Property	`form`	Specifies the form containing the `Hidden` object.
	`name`	Refers to the name of the `Hidden` object.
	`type`	Refers to the HTML `type` attribute of the `Hidden` object.
	`value`	Refers to the HTML `value` attribute of the `Hidden` object.
Method	`unwatch()`	Removes a watchpoint on a `Hidden` property.
	`watch()`	Sets a watchpoint on a `Hidden` property.

Radio Properties and Methods

Table 8.27 Properties, Methods, and Events of the `Radio` *Object*

Type	Item	Description
Property	`checked`	Specifies whether a button is checked or unchecked
	`defaultChecked`	Refers to the `checked` attribute of the HTML `<input>` tag
	`form`	Refers to the `Form` object that contains the `Radio` object
	`name`	Refers to the `name` attribute of the HTML `<input>` tag
	`type`	Refers to the `type` attribute of the HTML `<input>` tag
	`value`	Refers to the `value` attribute of the HTML `<input>` tag
Method	`blur()`	Removes focus from the `Radio` object
	`click()`	Simulates a mouse click on the button
	`focus()`	Sets the focus to a button
	`handleEvent()`	Invokes the default handler for the specified event
	`unwatch()`	Removes a watchpoint on a `Radio` property
	`watch()`	Sets a watchpoint on a `Radio` property
Event	`onBlur`	Event handler for the `Blur` event
	`onClick`	Event handler for the `Click` event
	`onFocus`	Event handler for the `Focus` event

Checkbox Properties and Methods

Table 8.5 Arguments, Properties, Methods, and Event Handlers Associated with the `Checkbox` ***Object***

Type	Item	Description
Argument	`num1`	An index number that allows access to check boxes through a form's element list.
	`num2`	An index number that allows access to individual check boxes that are grouped together under the same name.
Property	`checked`	A boolean value that determines whether the check box is checked.
	`defaultChecked`	A boolean value that holds the initial state of the check box. This value is set with the `checked` attribute.
	`form`	This property returns the `Form` object of the check box.
	`name`	The string that is specified in the `name` attribute of the HTML `<input>` tag.
	`type`	The string that is specified in the `type` attribute of the HTML `<input>` tag. This string is always `"checkbox"` for the `Checkbox` object.
	`value`	The value returned when the form is submitted.
Method	`blur()`	This method removes focus from the check box.
	`click()`	This method calls the check box's `onClick` event handler.
	`focus()`	This method applies focus to this check box.
	`handleEvent()`	This method passes an event to the appropriate event handler associated with the check box.
	`unwatch()`	This method removes a watch point.
	`watch()`	This method sets a watch point.
Event Handler	`onBlur`	The handler invoked when focus is removed from the check box.
	`onClick`	The handler invoked when the check box is selected.
	`onFocus`	The handler invoked when focus is applied to the check box.

Submit Properties and Methods

Table 8.36 Event Handlers, Methods, and Properties Used by the `Submit` *Object*

Type	Item	Description
Event Handler	`onBlur`	Executes code when the submit button loses focus. This event handler was added in JavaScript 1.1.
	`onClick`	Executes code when the submit button is clicked.
	`onFocus`	Executes code when the submit button receives the focus. This event handler was added in JavaScript 1.1.
Method	`blur()`	Removes focus from the submit button. This method was added in JavaScript 1.1.
	`click()`	Simulates a mouse click on the submit button.
	`focus()`	Gives the focus to the submit button. This method was added in JavaScript 1.1.
	`handleEvent()`	Invokes the handler for the event specified and was added in JavaScript 1.2.
	`unwatch()`	Used to turn off the watch for a particular property.
	`watch()`	Used to turn on the watch for a particular property.
Property	`form`	Returns the entire form that the submit button is in.
	`name`	Returns the name of the submit button specified by the `name` attribute.
	`type`	Returns the type of the submit button specified by the `type` attribute. This property always returns `submit`. This property was added in JavaScript 1.1.
	`value`	Returns the value of the submit button specified by the `value` attribute.

Reset Properties and Methods

Table 8.28 Properties, Methods, and Events of the `Reset` *Object*

Type	Item	Description
Property	`form`	Specifies the form name that contains the Reset button
	`name`	HTML **`name`** attribute for the Reset button
	`type`	HTML **`type`** attribute for the Reset button
	`value`	HTML **`value`** attribute for the Reset button
Method	`blur()`	Removes focus from the Reset button
	`click()`	Simulates a mouse click on a Reset button
	`focus()`	Sets the focus to the Reset button
	`handleEvent()`	Invokes the event handler
	`unwatch()`	Removes a watchpoint on a **`Reset`** property
	`watch()`	Sets a watchpoint on a **`Reset`** property
Event	`onBlur`	Event handler for the **`Blur`** event
	`onClick`	Event handler for the **`Click`** event
	`onFocus`	Event handler for the **`Focus`** event

Select Properties and Methods

Table 8.30 Event Handlers, Methods, and Properties Used by the `Select` ***Object***

Type	Item	Description
Event Handler	`onBlur`	Executes code when the select box loses the focus.
	`onChange`	Executes code when the select box has had its value modified.
	`onFocus`	Executes code when the select box receives the focus.
Method	`blur()`	Removes the focus from the select box.
	`focus()`	Gives the focus to the select box.
	`handleEvent()`	Invokes the handler for the event specified and was added in JavaScript 1.2.
	`unwatch()`	Used to turn off the watch for a particular property.
	`watch()`	Used to turn on the watch for a particular property.
Property	`form`	Returns the entire form that the select box is in.
	`length`	Returns the number of options in the select box.
	`name`	Returns the name of this select box specified by the `name` attribute.
	`options`	Returns an array containing each of the items in the select box. These items are created using the `<option>` HTML tag. There is also a `length` and `selectedIndex` subproperty of this property.
	`selectedIndex`	Returns an integer specifying the indexed location of the selected option in the select box.
	`type`	Returns the type of this select box specified by the `type` attribute. For `<select>` instances that contain the `multiple` attribute, this property returns `select-multiple`. Instances without this attribute return `select-one`. Note that this property was added in JavaScript 1.1.

Option Properties and Methods

Table 8.24 Properties and Methods of the `Option` *Object*

Type	Item	Description
Property	`defaultSelected`	Refers to the option that is selected by default from the select box
	`selected`	Refers to the selected value of the select box
	`text`	Refers to the text for the option
	`value`	Refers to the value that is returned to when the option is selected
Method	`unwatch()`	Removes a watchpoint on an `Option` property
	`watch()`	Sets a watchpoint on an `Option` property

Option Properties and Methods

```
<html>
<body>
<script type="text/javascript" language="JavaScript">
<!--
// function checks the form to see what is the default selected.
function check(myForm)
{
        for (var i = 0; i < document.form1.myList.length; i++)
         {
                if (document.form1.myList.options[i].defaultSelected == true)
                 {  alert("The default value is: "+ i);  }
        }
}
// -->
</script>
<form name=form1>
<select name="myList" multiple>
<option value=1>One
<option value=2 selected>Two
<option value=3>Three
<option value=4>Four
</select><p>
<input type="button" value="Get Default Value"
onClick="check(this.form)">
</form>
</body>
</html>
```

Option Properties and Methods

```
<form name="myForm2">
<select name="mySelect"
onchange="alert(mySelect.options.selectedIndex) ">
  <option value="">Please Select</option>
  <option value="HOCK">Hockey</option>
  <option value="RUG" selected>Rugby</option>
  <option value="GOLF">Golf</option>
  <option value="TENNIS">Tennis</option>
  </select>
</form>
```

TextArea Properties and Methods

Table 8.38 Event Handlers, Methods, and Properties Used by the `Textarea` *Object*

Type	Item	Description
Event Handler	`onBlur`	Executes code when the text area loses the focus.
	`onChange`	Executes code when the text area loses the focus and has had its value modified.
	`onFocus`	Executes code when the text area receives the focus.
	`onKeyDown`	Executes code when a key is pressed down. This occurs before an `onKeyPress` event handler is triggered and was added in JavaScript 1.2.
	`onKeyPress`	Executes code when a key is pressed down immediately after an `onKeyDown` event handler is triggered. This event handler was added in JavaScript 1.2.
	`onKeyUp`	Executes code when a key is released. This was added in JavaScript 1.2.
	`onSelect`	Executes code when a user selects some of the text within the text area.
Method	`blur()`	Removes the focus from the text area.
	`focus()`	Gives the focus to the text area.
	`handleEvent()`	Invokes the handler for the event specified and was added in JavaScript 1.2.
	`select()`	Selects the text in the text area.
	`unwatch()`	Used to turn off the watch for a particular property.
	`watch()`	Used to turn on the watch for a particular property.

TextArea Properties and Methods

Table 8.38 Continued

Type	Item	Description
Property	`defaultValue`	Returns the value of the text area defined between the beginning and ending `<textarea>` tags. Note that this property is not supported by the Opera browsers.
	`form`	Returns the entire form the text area is in.
	`name`	Returns the name of this text area specified by the `name` attribute.
	`type`	Returns the type of this text area. Note that this is always `textarea` and was added in JavaScript 1.1.
	`value`	Returns the value that is actually displayed in the text area.

Validation

```
<body>
<form method="POST" action="formSubmittedSuccessfully.html" name="myForm" onsubmit="return check();">
  <table >
    <tr>
      <td>Name</td>
      <td>
        <input type="text" name="name" size="20"/>
      </td>
    </tr>
    <tr>
      <td colspan="2"><span id="nameError" class="error"></span></td>
    </tr>
    <tr>
      <td>Pass</td>
      <td>
        <input type="password" name="pass" size="20"/>
      </td>
    </tr>
    <tr>
      <td colspan="2"><span id="passError" class="error"></span></td>
    </tr>
    <tr>
      <td>Gender</td>
      <td>
        <input type="radio" name="gender" value="M"/>Male
        <input type="radio" name="gender" value="F"/>Female
      </td>
    </tr>
    <tr>
      <td colspan="2"><span id="genderError" class="error"></span></td>
    </tr>
```

```
    <tr>
      <td>City</td>
      <td>
        <select size="1" name="city">
          <option value="">Please Select</option>
          <option value="cairo">Cairo</option>
          <option value="paris">Paris</option>
        </select>
      </td>
    </tr>
    <tr>
      <td colspan="2"><span id="cityError" class="error"></span></td>
    </tr>
    <tr>
      <td>Languages</td>
      <td>
        <input type="checkbox" name="lang" value="ar"/>ar
        <input type="checkbox" name="lang" value="en"/>en
      </td>
    </tr>
    <tr>
      <td colspan="2"><span id="langError" class="error"></span></td>
    </tr>
    <tr>
      <td colspan="2" align="center">
        <input type="submit" value="OK"/>
      </td>
    </tr>
  </table>
</form>
</body>
</html>
```

```javascript
<script type="text/javascript">
1 usage
function check() {
  var isValid = true; // Assume form is valid initially
  var errorMessage = [];
  errorMessage["name"] = "Name is required";
  errorMessage["pass"] = "Password is required";
  errorMessage["pass"]= "Gender is required";
  errorMessage["city"] = "City is required";
  errorMessage["lang"] = "At least one Language checkbox should be checked";
  var fieldsToCheck = ["name", "pass", "gender", "city", "lang"];
  // Reset previous errors
  fieldsToCheck.forEach(function(field) {
    document.getElementById(field + "Error").innerHTML = "";
  });
  // Check name
  if (document.myForm.name.value.trim() === "") {
    document.getElementById("nameError").innerHTML = errorMessage["name"];
    isValid = false;
  }
  // Check password
  if (document.myForm.pass.value.trim() === "") {
    document.getElementById("passError").innerHTML = errorMessage["pass"];
    isValid = false;
  }
  // Check gender
  if (!document.myForm.gender.value) {
    document.getElementById("genderError").innerHTML = errorMessage["pass"];
    isValid = false;
  }
  // Check city
  if (document.myForm.city.value === "") {
    document.getElementById("cityError").innerHTML = errorMessage["city"];
    isValid = false;
  }
  // Check languages (at least one checkbox should be checked)
  var checkboxes = document.querySelectorAll('input[name="lang"]:checked');
```

```
    // Check languages (at least one checkbox should be checked)
    var checkboxes = document.querySelectorAll('input[name="lang"]:checked');
    if (checkboxes.length === 0) {
      document.getElementById("langError").innerHTML = errorMessage["lang"] ;
      isValid = false;
    }
    return isValid; // Return the overall validity of the form
  }
</script>
<style>
  .error {
    color: red;
  }
</style>
</head>
```

Adding Edit Validation

```
<form method="POST" action="formSubmittedSuccessfully.html" name="myForm" onsubmit="return check();">
  <table >
    <tr>
      <td>Name</td>
      <td>
        <input type="text" name="name" size="20"  onblur="validateField('name')"/>
      </td>
    </tr>
    <tr>
      <td colspan="2"><span id="nameError" class="error"></span></td>
    </tr>
    <tr>
      <td>Pass</td>
      <td>
        <input type="password" name="pass" size="20"  onblur="validateField('pass')"/>
      </td>
    </tr>
    <tr>
      <td colspan="2"><span id="passError" class="error"></span></td>
    </tr>
    <tr>
      <td>Gender</td>
      <td>
        <input type="radio" name="gender" value="M" onblur="validateField('gender')"/>Male
        <input type="radio" name="gender" value="F" onblur="validateField('gender')"/>Female
      </td>
    </tr>
    <tr>
      <td colspan="2"><span id="genderError" class="error"></span></td>
    </tr>
    <tr>
      <td>City</td>
      <td>
        <select size="1" name="city"  onblur="validateField('city')">
          <option value="">Please Select</option>
          <option value="cairo">Cairo</option>
          <option value="paris">Paris</option>
        </select>
      </td>
    </tr>
```

```
    <tr>
      <td>Languages</td>
      <td onclick="validateField('lang')">
        <input type="checkbox" name="lang" value="ar"/>ar
        <input type="checkbox" name="lang" value="en"/>en
      </td>
    </tr>
    <tr>
      <td colspan="2"><span id="langError" class="error" ></span></td>
    </tr>
    <tr>
      <td colspan="2" align="center">
        <input type="submit" value="OK"/>
      </td>
    </tr>
  </table>


</form>
</body>
</html>
```

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>Form Validation</title>
  <script type="text/javascript">
    var errorMessage = {
      "name": "Name is required",
      "pass": "Password is required",
      "gender": "Gender is required",
      "city": "City is required",
      "lang": "At least one Language checkbox should be checked"
    };

    5+ usages
    function validateField(fieldId) {
      var value = document.myForm[fieldId].value.trim();
      var errorSpan = document.getElementById(fieldId + "Error");
      if (fieldId === "lang") {
        // Special handling for checkboxes
        var checkboxes = document.querySelectorAll('input[name="lang"]:checked');
        if (checkboxes.length === 0) {
          errorSpan.innerHTML = errorMessage[fieldId];
        } else {
          errorSpan.innerHTML = "";
        }
      } else if (fieldId === "gender") {
        // Special handling for radio buttons
        var selectedGender = document.querySelector('input[name="gender"]:checked');
        if (!selectedGender) {
          errorSpan.innerHTML = errorMessage[fieldId];
        } else {
          errorSpan.innerHTML = "";
        }
      } else if (value === "") {
        errorSpan.innerHTML = errorMessage[fieldId];
      } else {
        errorSpan.innerHTML = "";
      }
    }
```

```
    function check() {
      var isValid = true;
      ["name", "pass", "gender", "city", "lang"].forEach(function(fieldId) {
        validateField(fieldId);
        if (document.getElementById(fieldId + "Error").innerHTML !== "") {
          isValid = false;
        }
      });
      return isValid;
    }
  </script>
  <style>
    .error {
      color: red;
    }
  </style>
</head>
```