**Exercise 1: Working with the JavaScript DOM**

**Scenario**

In this exercise you will be required to complete and analyze the following code segments from the index.html page.

**index.html   
…**

<script>

***MISSING\_CODE1***

***MISSING\_CODE2***

</script>

</head>

…

<div id="section">

<p onmouseover="changeBgColor1(this)" onmouseout="changeBgColor2(this)">  
Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua.  
</p>

</div>

**…**

**Exercise 1**

| **Task list** | |
| --- | --- |
| **Number** | **Instructions** |
| **1** | In the <script> tags add a JavaScript function that changes the background color of an element to blue. |
| **2** | In the <script> tags add a JavaScript function that makes the background of an element transparent. |
| **3** | You want to change the color of the <p> tag when a user puts the mouse over it and change it back to normal when the mouse leaves. Add event handlers to the <p> tag to call each function accordingly. |

**Exercise 1: Working with JavaScript**

**Solution**

| **Checkpoints** | |
| --- | --- |
| **Number** | **Details** |
| **1** | To change the background color of an element you can use the following function  function changeBgColor1 ( el ) {  el.style.backgroundColor = 'blue';  } |
| **2** | To change the background element to transparent you can use the following function  function changeBgColor1 ( el ) {  el.style.backgroundColor = 'transparent';  }  Note that 'transparent' is the default. |
| **3** | The complete code to add the event handler to the <p> tag is :  <p onmouseover="changeBgColor1(this)" onmouseout="changeBgColor2(this)">  Note we want to capture both when the mouse enters the element and when it leaves. |

**Exercise 2: Working with JavaScript and Forms**

**Scenario**

In this exercise you will be required to complete and analyze the following JavaScript code snippets to validate a form.

**input.html   
…**

<script>

function validateForm(){

let empName = ***MISSING\_CODE1 ;***

if (***MISSING\_CODE2***) {

***MISSING\_CODE3***="Name cannot be empty!";

empName.focus();

return false;

}

}

</script>

…

…

<form name="frmCollectWeights" action="handler.html" ***MISSING\_CODE4*** method="post">

<div>

<label for="empName">Employee Name</label>

<input id="empName" type="text" /><span id="emptyName"></span>

</div>

<div>

<button>Submit</button>

</div>

</form>

**…**

**Exercise 2: Working with JavaScript and Forms**

| **Task list** | |
| --- | --- |
| **Number** | **Instructions** |
| **1** | Complete the code to assign a reference to the form input element "empName" to a JavaScript variable. |
| **2** | Complete the code to check if the input value is empty. |
| **3** | Complete the code to display an error message if the name field is empty. Use the <span> named " emptyName" to display your message. |
| **4** | Complete the code in the <form> tag to check for an empty name when the form is submitted. |

**Exercise 2: Working with JavaScript**

**Solution**

| **Checkpoints** | |
| --- | --- |
| **Number** | **Details** |
| **1** | The code required to assign the element reference to a variable is:  document.forms["frmCollectWeights"]["empName"]  Note we make sure to refer to both the element and the form that contains it. |
| **2** | We can use the variable empName and the comparison operator to check for an empty string. So in this case the code would be:  empName.value == "" |
| **3** | To complete the code to display the error message we can use:  document.getElementById("emptyWeight").innerHTML=  So long as we know the name of the element, we can use the document object to refer to it using "getElementbyId". |
| **4** | To check the form we use:  onsubmit="return validateForm()"  This will call the JavaScript function "validateForm()", which checks for empty values. If it returns false to the call, this cancels the form action, which is to submit the form to the "handler.html" page. |

**Exercise 3: Working with JavaScript and AJAX**

**Scenario**

In this exercise you will be required to complete and analyze the following JavaScript code to correctly implement the use of AJAX.

**script\_ajax.js**

let xmlhttp = ***MISSING\_CODE1*** ;

let url = "http://localhost:8000/getweights";

function getData() {

***MISSING\_CODE2*** = function() {

if (this.readyState == 4 && this.status == 200) {

let myArr = ***MISSING\_CODE3*** (this.responseText);   
displayData(myArr);

}

};

***MISSING\_CODE4 ;***

xmlhttp.setRequestHeader("Content-Type", "application/x-www-form-urlencoded");

xmlhttp.send();

}   
function displayData(arr) {

let out = "";

let i;

for(i = 0; i < arr.length; i++) {

out += arr[i].empName + ' weighs ' + arr[i].empWeight + 'Kg</br>';

}

document.getElementById("id01").innerHTML = out;

}

**Exercise 3: Working with JavaScript and AJAX**

| **Task list** | |
| --- | --- |
| **Number** | **Instructions** |
| **1** | Complete the code to create an XMLHttpRequest object so that we can utilize AJAX. |
| **2** | Complete the code so the function will be called when the readyState property changes. |
| **3** | Correctly parse the responseText so it can be used with JavaScript. |
| **4** | Complete the code to initialize a new request. |

**Exercise 3: Working with JavaScript and AJAX**

**Solution**

| **Checkpoints** | |
| --- | --- |
| **Number** | **Details** |
| **1** | The code to create an XMLHttpRequest object so that we can utilize AJAX is:  *new XMLHttpRequest()*  We need to make sure we use the "new" operator to properly instantiate the object to use in our code. |
| **2** | To call the function whenever the readyState attribute changes we use the following code:  xmlhttp.onreadystatechange  The handler will be called any time the readystatechange event is fired. |
| **3** | Correctly parse the responseText so it can be used with JavaScript. |
| **4** | To initialize a new request we use the following:  xmlhttp.open("GET", url, true);  This creates a new request, specifying the method to use, the url to send it to , and whether or not the operation is asynchronous. |