Web Interface Elements (i)

The aim of this exercise is to explore the use of client-side scripts to add interactivity to web-pages, and to become familiar with the set of interface widgets supported by web-browsers.

Using a text-editor, create a web-page that includes both a `<head>` and a `<body>` section, then add the following elements.

- In the `<body>` of the page add a pair of form tags (`<form>` and `</form>`).

  In the opening `<form>` tag specify a name (e.g., `name='myForm'`).
- Inside the `<form>` tags, create a text-box, e.g.:

```
<input type='text' name='firstName'>
```

- OUTSIDE of the form (i.e., after the closing `</form>` tag), create a button, e.g.:

```
<input type='button' value='Test' onclick='processForm()'>
```

- In the `<head>` of the web-page, add a pair of `<script>` tags, e.g.:

```
<script type='text/javascript'>

</script>
```

- Within the `<script>` tags, write a function that will be called when the button is clicked, e.g.:

```
function processForm() {
    alert(document.myForm.firstName.value);
}
```

  (assuming `myForm` is the name of your form, and `textBox` is the name of your text-box).
- Save the page and view it in a browser. Type some text into the text-box and click the button - your text should be displayed in a dialog-box.

When this is working correctly, add a **label** to the text-box. This will contain a text string which appears on the web-page next to the text-box to indicate its purpose.

- First give your text-box an `id`, e.g.:

```
<input type='text' name='firstName' id='name1'>
```

- Next, add a pair of `<label>` tags before the text-box, and assign the `id` of your text-box to the `for` property, e.g.:

```
<label for='name1'>First Name</label>
```

- Modify the `alert()` statement in your function as follows:

```
alert('Your first name is: ' + document.myForm.firstName.value);
```

  Note the use of the `+` operator in the `alert()` dialog to concatenate the value of a variable onto a literal string.

- Open your web-page in a browser and test it. Note that there is no default formatting associated with the `<label>` tag, but you can add suitable formatting using CSS if you wish.

Add another text-box into which the user can enter his/her surname. Give it a suitable label, and extend the `alert()` statement so that it reports both the first-name and surname entered by the user. Make sure you give each text-box a unique `name` and `id`.

Add a section to the form in which the user can list his/her hobbies and interests. This can be done using ***check-boxes***.

- Add a check-box to your form, e.g.:

```
<input type='checkbox' name='cinema' id='hobby1'>
```

  Give it a suitable label, just as you did with the text-boxes.

- Check-boxes have an attribute called `checked`, a Boolean value which indicates whether the user has checked the box or not. Thus you can find out whether the check-box is checked by adding the following line to your function:

```
alert(document.myForm.cinema.checked);
```

- Test your code in a browser, and make sure the function correctly reports whether the check-box is checked or not.

When this is working correctly, add further check-boxes for other hobbies and interests.

- Rather than use a separate `alert()` statement for each one, create a message and add to it as you check each box, e.g.:

```
var message = 'Your hobbies are: ';
if (document.myForm.cinema.checked) message += 'cinema; ';
if (...

alert(message);
```

When this is working correctly, add a section to the form in which the user can indicate his/her age-group. This can be done using ***radio-buttons***.

Radio-buttons are normally used in groups. If each button in the group has the same name, it will only be possible to select one button at a time. For example:

```
<input type='radio' name='ageGroup' value='25-34'>
```

If there are several such radio buttons within a form, each having the same name but different values, it will only be possible to select one at a time.

- Create a set of radio buttons in your form. Give each one a `label` that reflects its `value`, indicating an age-group.

- Add code to your function to indicate which radio-button has been checked. Like check-boxes, radio-buttons have a `checked` attribute that will be set to `true` if the button is checked. You can find out which radio-button is checked in various ways:

- Groups of radio-buttons can be accessed as an array, so you can loop though the array to find out which one has been selected, e.g.:

```
for (var i = 0; i < document.myForm.ageGroup.length; i++){
   if(document.myForm.ageGroup[i].checked)
         alert('You checked option ' + i);
}
```

  Note the use of the `length` property: in JavaScript, the number of elements in an array is stored in the `length` property, and this can be used to determine the number of times the loop executes.

  This code will return the ***index*** of the selected button, but it can easily be modified to display the `value`.

- Alternatively, you can use the `querySelector()` method to find out which radio-button in the group has been selected, e.g.:

```
document.querySelector('input[name=ageGroup]:checked').value
```

- Test your code in a browser, and make sure the function correctly reports which radio-button is checked.

When you have all the above working, modify the code so that it checks the contents of ALL the form elements (text-boxes, check-boxes, and radio-buttons) and displays a single dialog message that contains all the form information, e.g., 'Your name is Fred Bloggs, your age is 42 and your hobbies are: cinema, line-dancing'.