# JavaScript

The aim of this exercise is to prepare an HTML form using JavaScript. The form will verify the correctness of its contents prior to it being sent to the server. The form will also be dynamic, i.e., some of its elements will change their state based on the user input. To complete this exercise, you will need a text editor and a web browser.

1. Create two text files in the same directory: *form.html* and *form_check.js*.
2. The form will be used to complete the data required to register a new user in a hypothetical website. Paste the following code into your *form.html* file:

```
<!doctype html>
<html>
<head>
    <meta charset="utf-8" />
    <title>JavaScript</title>
</head>
<body>
    <form name="data">
        <table>
            <tr><td>First name</td><td><input type="text" name="f_fname"></td></tr>
            <tr><td>Last Name</td><td><input type="text" name="f_lname"></td></tr>
            <tr>
                <td>Gender</td>
                <td>
                    <input name="f_gender" value="f_f" checked type="radio" />female<br />
                    <input name="f_gender" value="f_m" type="radio" />male
                </td>
            </tr>
            <tr><td>Maiden name</td><td><input type="text" name="f_mname"></td></tr>
            <tr><td>E-mail</td><td><input type="text" name="f_email"></td></tr>
            <tr><td>Zip code</td><td><input type="text" name="f_zip"></td></tr>
            <tr><td>Street</td><td><input type="text" name="f_street"></td></tr>
            <tr><td>City</td><td><input type="text" name="f_city"></td></tr>
            <tr><td>Remarks</td><td><textarea rows="5" cols="15" name="remarks"></textarea>
            </td></tr>
            <tr><td colspan="2"><input type="submit" value="Submit"></td></tr>
        </table>
    </form>
</body>
</html>
```

3. In the *form_check.js* file we will put all function definitions, which will be used to check the elementary conditions of the values in the form. We will start by writing a function which checks whether a given field is empty. Declare a function named `isEmpty` which accepts one parameter and return `true` if the value passed as the parameter is empty and `false`, otherwise. You can use the `length` field which holds the length of a given string.
4. Now, we will use the function declared in the previous step to check if the user provided his first name. Declare another function, called `validate`, which we will pass the form into for verification. For now, the function should only call the `isEmpty` function, passing the value of the `f_fname` field (`formParameter.elements["f_fname"].value`). If the field is empty, the function should display an alert with an appropriate message (`alert("First name cannot be empty!");`) and return `false`. Otherwise, the function should return `true`.
5. Reference the external JavaScript file in the head of your HTML file using the `script` tag.
6. Add the `onclick` event to your form. As an action to this event, call the `validate` method, passing the form as a parameter (`validate(this.form);`).

7. Run your form in the browser and test it's behaviour by trying to submit a form without first name. Pay attention to the website when closing the alert window.
8. Change the `onclick` event handler to `return validate(this.form);`. Test the website again. Can you see the difference?
9. The form can be easily fooled by providing a name consisting of white spaces. Add a function which checks if a given string is entirely composed of white spaces.

```
function isWhiteSpace(str) {
    var ws = "\t\n\r ";
    for (var i = 0; i < str.length; i++) {
        var c = str.charAt(i);
        if (ws.indexOf(c) == -1) {
            return false;
        }
    }
    return true;
}
```

10. Modify the `validate` function co that it also calls the function `isWhiteSpace`. Test the website again.
11. We will now modify the code in our scripts o that it is easier to add validation to other fields. Add a new function, called `checkString`, to your script file. The function should accept two parameters: the string to check and a message to display if the string turns out to be empty or containing only white space symbols. Furthermore, the function should return `false` in case of an incorrect string and `true`, otherwise.
12. Modify the `validate` function, so that it uses the `checkString` function and add last name, zip code, steet, and city validation. Test your website.
13. In this step, we will add a function which performs a basic e-mail verification. Add the following code to your script file.

```
function checkEmail(str) {
    if (isWhiteSpace(str)) {
        alert("Incorrect e-mail");
        return false;
    }
    else {
        var at = str.indexOf("@");
        if (at < 1) {
            alert("Incorrect e-mail");
            return false;
        }
        else {
            var l = -1;
            for (var i = 0; i < str.length; i++) {
                var c = str.charAt(i);
                if (c == ".") {
                    l = i;
                }
            }
            if ((l < (at + 2)) || (l == str.length - 1)) {
                alert("Incorrect e-mail");
                return false;
            }
        }
        return true;
    }
}
```

14. Implement a necessary modification in the `validate` function to add e-mail validation.
15. The solution achieved so far is not very convenient, as it requires two additional clicks from a user when he/she makes a mistake (one to close the alert window and another one to focus back

to the invalid field). In this next example, we will show how to solve this issue. Modify the form by adding a field for displaying errors after the first name field.

```
<span class="err" id="e_fname" />
```

16. Create a new stylesheet, link it to your website and add a rule which will format the error message. The rule should select all elements `<span>` with class `err`.

```
color: red;
font-weight: bold;
padding-left: 5px;
```

17. Add a new field verification function to your script and perform a necessary modification in the `validate` function.

```
function checkStringAndFocus(obj, msg) {
    var str = obj.value;
    var errorFieldName = "e_" + obj.name.substr(2, obj.name.length);
    if (isWhiteSpace(str) || isEmpty(str)) {
        document.getElementById(errorFieldName).innerHTML = msg;
        obj.focus();
        return false;
    }
    else {
        return true;
    }
}
```

18. Unfortunately, the error message still appears even after the user provided a correct value, what is somewhat confusing. Let us use a timer to automatically remove the message after 5 seconds. Add the following functions to your script file.

```
var errorField = "";

function startTimer(fName) {
    errorField = fName;
    window.setTimeout("clearError(errorField)", 5000);
}

function clearError(objName) {
    document.getElementById(objName).innerHTML = "";
}
```

19. The last step of this part of the exercise is to call the `startTimer` function. Add the following line in the appropriate place within the `checkStringAndFocus` function.

```
startTimer(errorFieldName);
```

20. Test your website.

21. In this step, we will modify the form so that maiden name is accessible only to women. Add the following code to your script.

```
function showElement(e) {
    document.getElementById(e).style.visibility = 'visible';
}

function hideElement(e) {
    document.getElementById(e).style.visibility = 'hidden';
}
```

22. Surround the maiden name field with a `<span>` element with id equal to `MaidenName`.
23. Add an event handler which will be triggered after selecting one of the available gender options. Depending on the selection, the maiden name should be shown or hidden. In the first `f_gender` element set the `onclick` attribute to an appropriate call of the `showElement` function. Do the same for the second element but this time call the `hideElement` function.
24. Test your website.
25. Now, we will change the e-mail validation to a one using regular expressions. Add the following code to your script file and call the `checkEmailRegEx` function in the `validate` function. Test your website.

```
function checkEmailRegEx(str) {
    var email = /[a-zA-Z_0-9\.]+@[a-zA-Z_0-9\.]+\.[a-zA-Z][a-zA-Z]+/;
    if (email.test(str))
        return true;
    else {
        alert("Wrong e-mail address");
        return false;
    }
}
```

26. The next enhancement we will introduce is a live validation of the zip code. Surround the `f_zip` field with a `<span>` element with a `zip` identifier.
27. Create a new function, called `checkZIPCodeRegEx`, which accepts one parameter. Use regular expression to check if the passed value is a proper zip code. If yes:
    - set the content of `kod` element (`innerHTML`) to `"OK"` and class (`className`) to `green`,
    - return `false`.

    If not:
    - set the content of `kod` element to `"WRONG"` and class to `red`,
    - return `true`.
28. On the `onkeyup` event of the zip code field, add an appropriate call to the `checkZIPCodeRegEx` function.
29. Add two CSS rules for classes `green` and `red`, which will set the color to green and red, respectively.
30. In the body of the `validate` function, change the zip code validation call to your new function.
31. JavaScript allows for an easy form field enumeration. Add another CSS rule which adds a red, dotted border to all elements with class `wrong`. Modify the `validate` function so that it assigns the `wrong` class to all invalid fields.
32. Test your website.

33. Insert the following table above the form.

```
<table>
    <tbody>
        <tr><td>Row 1</td></tr>
        <tr><td>Row 2</td></tr>
        <tr><td>Row 3</td></tr>
        <tr><td>Row 4</td></tr>
        <tr><td>Row 5</td></tr>
        <tr><td>Row 6</td></tr>
        <tr><td>Row 7</td></tr>
    </tbody>
</table>
```

34. Add the following function to your script file. Parameter i is a row counter and parameter e is a row.

```
function alterRows(i, e) {
    if (e) {
        if (i % 2 == 1) {
            e.setAttribute("style", "background-color: Aqua;");
        }
        e = e.nextSibling;
        while (e && e.nodeType != 1) {
            e = e.nextSibling;
        }
        alterRows(++i, e);
    }
}
```

35. Call the above function, passing 1 as the first value and the first row from the table (use `getElementsByTagName`), as the second parameter. Test your website. Do you see any change? Fix this issue.
36. Now, we will show how to manipulate the contents of a website using JavaScript. Place the following code in your script file.

```
function nextNode(e) {
    while (e && e.nodeType != 1) {
        e = e.nextSibling;
    }
    return e;
}

function prevNode(e) {
    while (e && e.nodeType != 1) {
        e = e.previousSibling;
    }
    return e;
}

function swapRows(b) {
    var tab = prevNode(b.previousSibling);
    var tBody = nextNode(tab.firstChild);
    var lastNode = prevNode(tBody.lastChild);
    tBody.removeChild(lastNode);
    var firstNode = nextNode(tBody.firstChild);
    tBody.insertBefore(lastNode, firstNode);
}
```

37. Insert a button immediately after the table in your document and call the `swapRows` function on the click event.
38. This next example will illustrate how to control the length of the user input. Replace the following code:

```
<tr>
    <td>Remarks</td>
    <td>
        <textarea rows="5" cols="15" name="remarks"></textarea>
    </td>
</tr>
```

with:

```
<tr>
    <td>Remarks</td>
    <td>
        <textarea rows="5" cols="15" name="remarks"
            onkeyup="cnt(this.form.remarks, document.getElementById('left'), 150)">
</textarea>

    </td>
</tr>
<tr>
    <td><span id="left">150</span> letters left.</td>
</tr>
```

39. Add the following function to your script file and test your website.

```
function cnt(form, msg, maxSize) {
    if (form.value.length > maxSize)
        form.value = form.value.substring(0, maxSize);
    else
        msg.innerHTML = maxSize - form.value.length;
}
```

40. Create a net file *form2.html* and fill it with the following content. Validate the form using your validation library of choice (e.g., http://parsleyjs.org/).

```
<!doctype html>
<html>
<head>
    <meta charset="utf-8" /><title>JavaScript</title>
</head>
<body>
    <form name="data"><table>
            <tr><td>Name</td><td><input type="text" name="f_name"></td></tr>
            <tr><td>E-mail</td><td><input type="text" name="f_email"></td></tr>
            <tr><td>Zip code</td><td><input type="text" name="f_zip"></td></tr>
            <tr><td>City </td><td><input type="text" name="f_city"></td></tr>
            <tr><td colspan="2"><input type="submit" value="Submit"></td></tr>
    </table></form>
</body>
</html>
```