



COMP5347: Web Application Development

Week 3 Tutorial: Client-Side and JavaScript

Learning Objectives

- Get to know a few more HTML elements
- Understand DOM and Event in JavaScript
- Trace and debug JavaScript code with Developer Tool
- Practice client-side JavaScript to manipulate HTML and CSS properties

Start-up Files

Download week3.zip from Canvas and extract it to a collection of start files. This should include a base HTML file: week3.html, a simple CSS file: week3.css and a simple JavaScript file: week3.js. week3.html describes part of a photo uploading form allowing end users to input the title, description and other information of the photo.

Tasks

Your task in this week's lab is to implement some simple pre-validation logic in week3.js to check if all required information is filled in the photo uploading form. The pre-validation is activated when an end user clicks the submit button. It does the check and highlights the ones that needs to be filled. Figure 1 shows an example of pre-validation result, with three fields highlighted. Given the provided details, complete the following tasks.

Task 1: Inspecting HTML

Open week3.html in Microsoft Expression Web to inspect the actual HTML mark-ups. The main content of this HTML file is a form with various input related elements. The elements are organized as a table to position them nicely. Most of the elements are different types of input such as text, number, checkbox, color, data, time and so on. Make sure you understand the elements and their respective representations in a browser window.

The screenshot shows a web form titled "Photo Details". The form contains several input fields and sections:

- Title:** A text input field with a red background and red text, indicating a validation error.
- Description:** A large text area with a red background and red text, indicating a validation error.
- Continent:** A dropdown menu with the text "Choose continent ▾".
- Country:** A dropdown menu with the text "Choose country ▾".
- City:** A text input field.
- Copyright?:** A section with two radio buttons: "All rights reserved" (unselected) and "Creative Commons" (selected).
- Creative Commons Types:** A section with four checkboxes: "Attribution" (unselected), "Noncommercial" (unselected), "No Derivative Works" (unselected), and "Share Alike" (unselected).
- I accept the software license:** A checkbox that is currently unchecked.
- Rate this photo:** A text input field.
- Color Collection:** A color selection box showing a black color.
- Date Taken:** A text input field with the placeholder "dd/mm/yyyy".
- Time Taken:** A text input field with the placeholder "--:-- --".
- Buttons:** "Submit" and "Clear Form" buttons at the bottom.

Figure 1: Screenshot pre-validation results

Task 2: JavaScript Tracing and Debugging

- Open week3.html in Google chrome. Click the title field, you will notice that the background turns into green when the focus is inside the field. When you type in the field, the font is bold. After you finish typing, you can move the focus out of the title field by pressing the tab key or by clicking on a different area. You will notice that the title field's background changes back to white and the font style changes to normal. This simple interactive feature is enabled through JavaScript.
- Load the JavaScript file week3.js in Microsoft Expression Web to inspect the content. The script defines an anonymous function to be executed on the

window.onload event. This function finds an element with id mainFrame (see line 3). This is the element representing the form. It then finds all elements belonging to a class required. For every element in class required: one function is registered on the onfocus event, another function is registered on the onblur event. They are the event handler of the onfocus event and the onblur event. The onfocus event handler (line 9-10) changes the backgroundColor of the **current element** (as indicated by this key- word) to “green” and the fontWeight to “bold”. The onblur event handler (line 15-16) changes the fontWeight back to “normal” and backgroundcolor to “#FFFFFF”, the color code representing “white”. It also register an onsubmit event handler displaying a prompt window (line 21-24).

Switch back to week3.html and make sure you can identify the elements belonging to required class.

- c) Turn on Chrome’s Developer Tool (DevTool) as show in figure 2

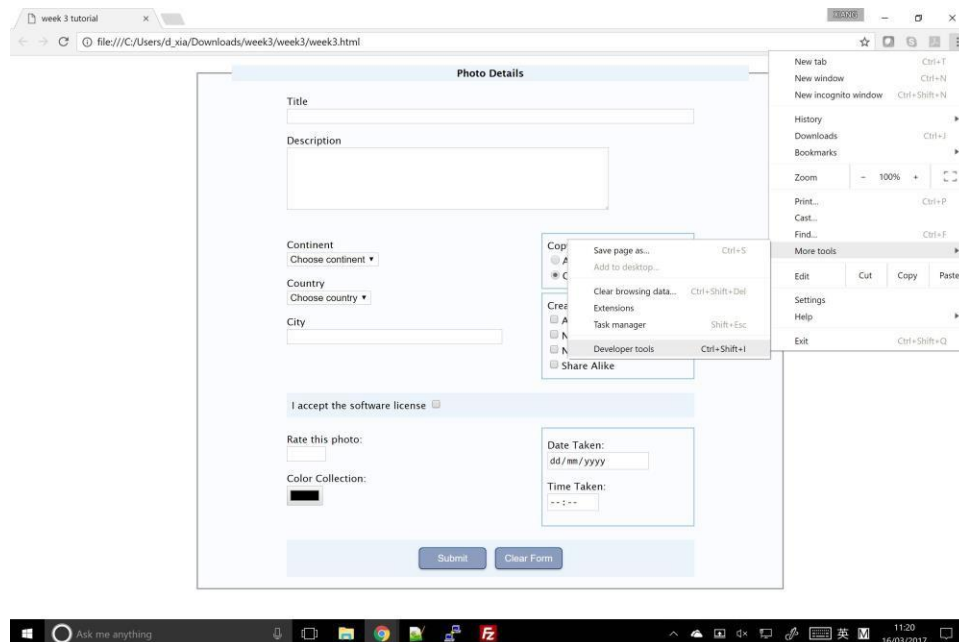


Figure 2: Start Developer Tool

- d) Load the JavaScript file in DevTool by clicking Sources menu item on the DevTool's navigation bar and week3.js on the left panel (See Figure 3). You can scroll to see the whole JavaScript file. Now set a breakpoint on line 8 by clicking the line no. If the breakpoint is successfully set, you will see line no 8 highlighted with a blue arrow, similar to that on Figure 3. This would force the JavaScript execution to pause at line 8, which is the first statement inside the for- loop of the windows.onload event handler.

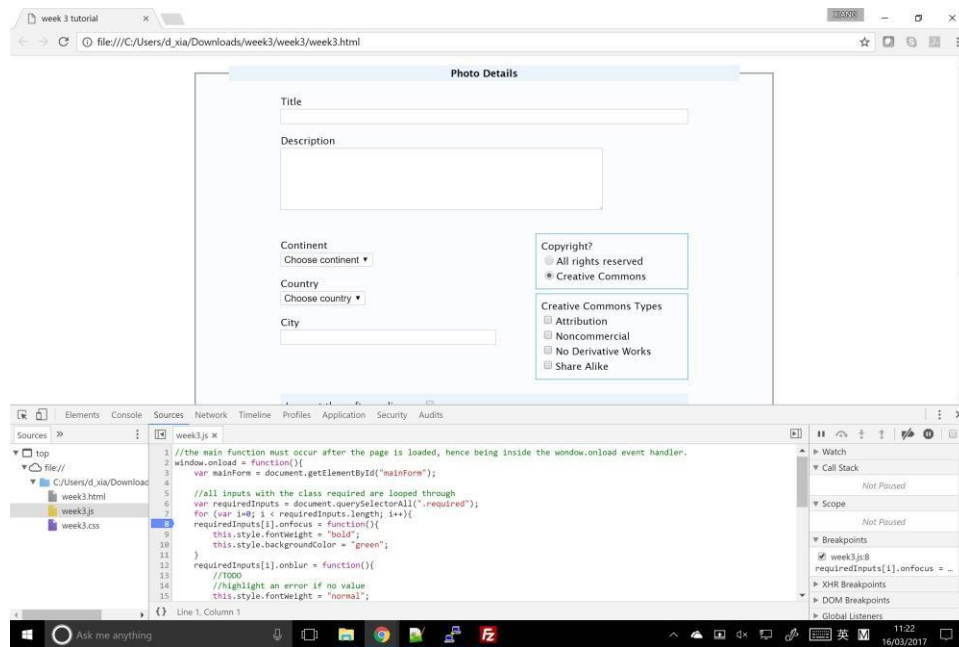


Figure 3: Load JavaScript source and set breakpoint inside function registered on `onwindow.focus`

- e) Now reload the page, you will notice that the browser displays the whole form as before but is paused by JavaScript execution. You will not be able to interact with that browser window at this moment (see Figure 4). This implies that the `window.onload` event happens after the browser starts to display the HTML content on the screen.

When the browser pauses in the middle of JavaScript execution, it enters the debug mode, which allows you to inspect variable values. The DevTools by default displays values right next to the variable. You can also check the values on the right panel. Figure 4 is a screen shot of browser pausing at the statement on line 8. You are able to see the value of the following variables:

- The `mainFrame` variable points to a form element with id “mainFrame”
- The variable `requiredInputs` points to an array with two values, both are of input type and belong to class “required”
- the local variable `i` has a value of 0.

Press F8 or the play icon representing “Resume script execution”, the browser will execute the script until next breakpoint. It will stop at the same statement on line 8 because we need to loop on all values of `requiredInputs`. At the second stop, the local variable `i` changes its value to 1.

Press the play icon again, the JavaScript will finish execution and you are able to interact with the browser as usual. This registered the two event handlers for switching styling when focus is on an off. In the next question, we will see when those handlers are called.

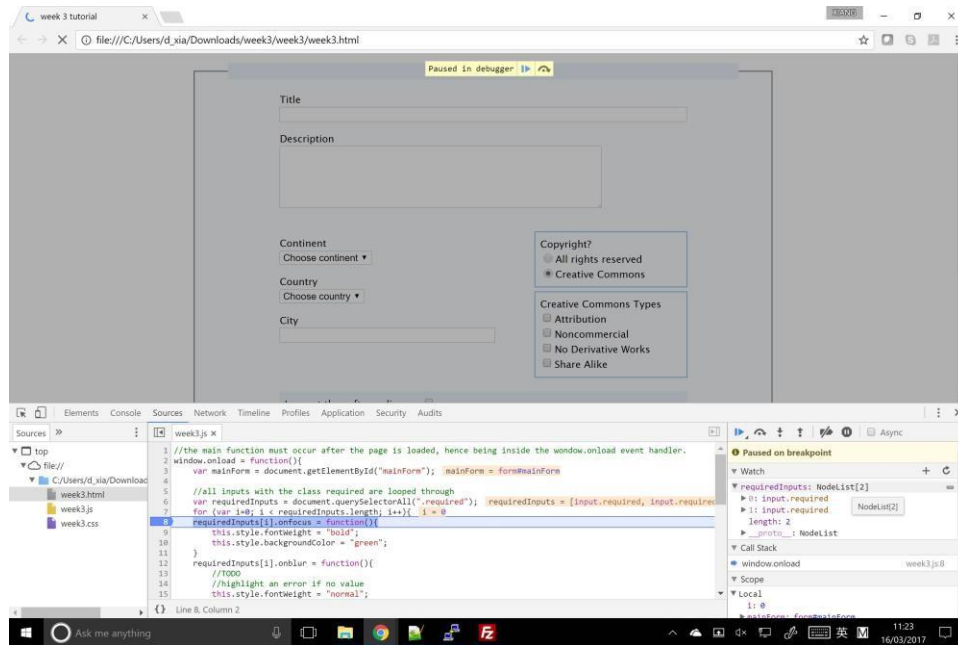


Figure 4: Reload week3.html

Task 3: JavaScript Event Handler

- Scroll down to statement on line 15 and create a breakpoint there (see Figure 5). This statement is inside the onblur event handler of any element with class label “required”. It sets the fontWeight back to “normal”.
- Now type something in the title field (as show in Figure 6) then move the focus out. After you move the focus out, the onblur event handler on the title input field is triggered, the browser starts to execute the JavaScript and pauses on statement on line 15 (as show in figure 7) in debug mode.
- In debug mode, you can control how frequent browser pauses JavaScript execution. For instance, you can instruct the browser to proceed to next breakpoint using F8 as we did in the previous question. You can also instruct the browser to execute one statement a time and observe the effect of that particular statement. Press F10 or click the icon representing “step over next function call”, this will cause browser to execute the first statement inside the event handler (statement on line 15). After executing this statement, you will notice the font weight becomes normal in the title input field (see Figure 8).
- Press F10 or click the “step over” icon a second time to execute the statement on line 16. After executing the statement, you will find the background color of the title input field is changed back to white (see Figure 9).

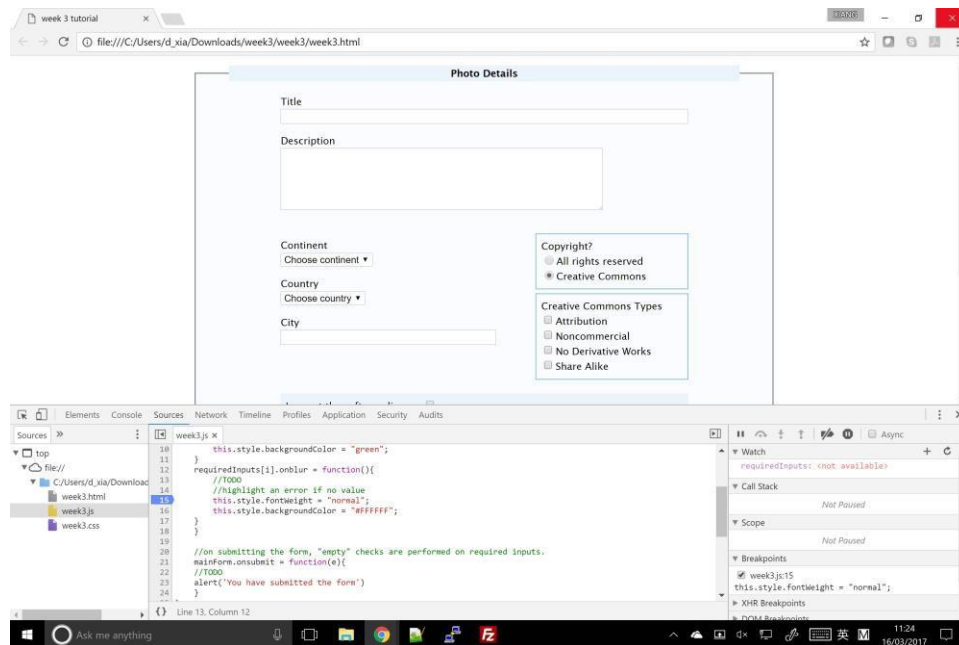


Figure 5: Breakpoint inside onBlur event handler

Task 4: Write and test your own pre-validation code

When you click the submit button, you will see a prompt window saying “You have submitted the form”. This is the current implementation of the onsubmit event handler of the form. Now modify the onsubmit event handler to implement the simple pre-validation logic:

- loop through all required element to see if any is left blank
- for any required element that is left blank, change the background color of the parent element to red. We use the parent element to ensure that any label, such as the “Title” text is included and highlighted.
- for any required element that is not left blank, change the background color of the parent element back to normal.

The last item is to ensure that for a second submission of the form, all styles defined previously would be reset. For instance, if a highlighted input is filled and resubmitted, the highlight should be removed.

If you cannot finish the coding part, you can continue working on it at your own time.

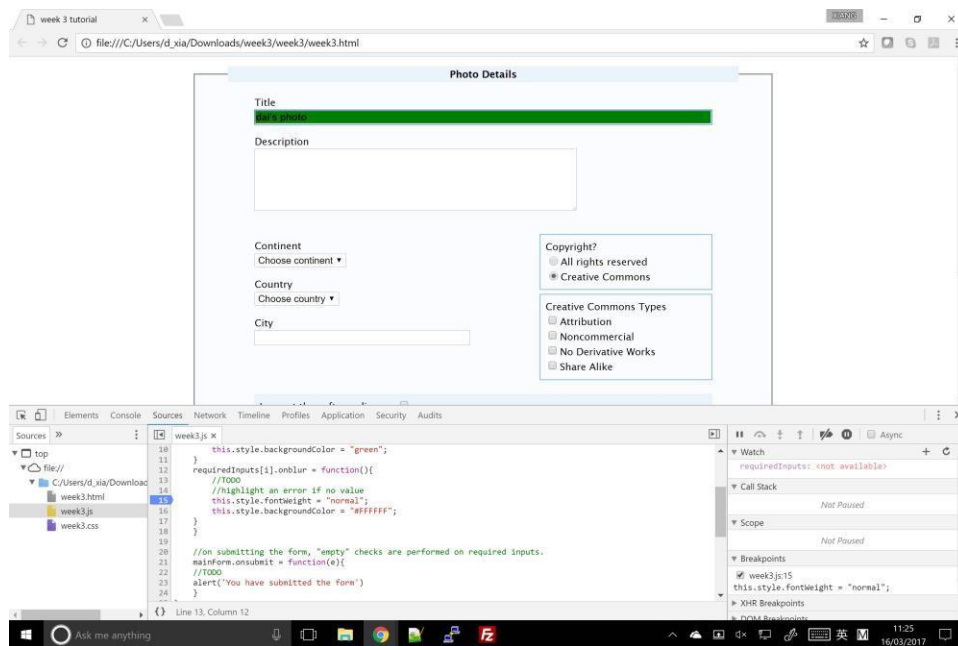


Figure 6: Typing in title input field

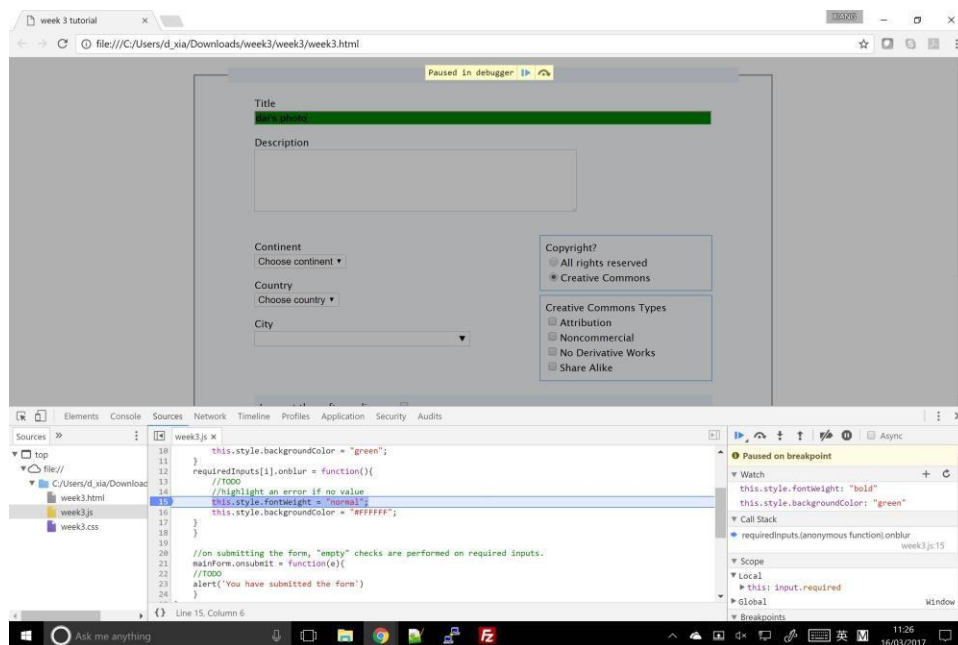


Figure 7: Browser pause when focus moves out

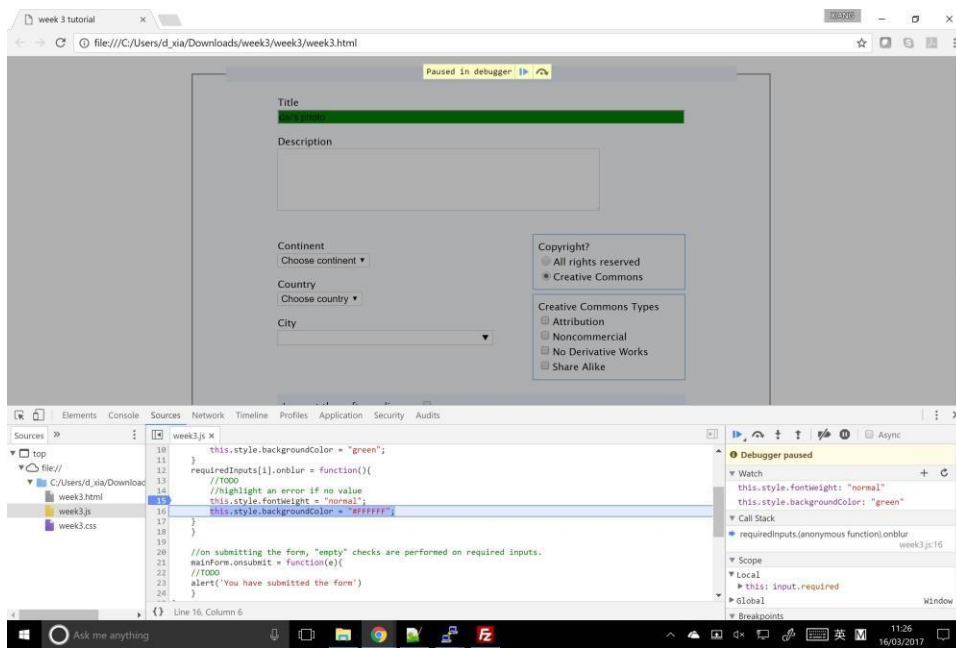


Figure 8: Browser pause when focus moves out

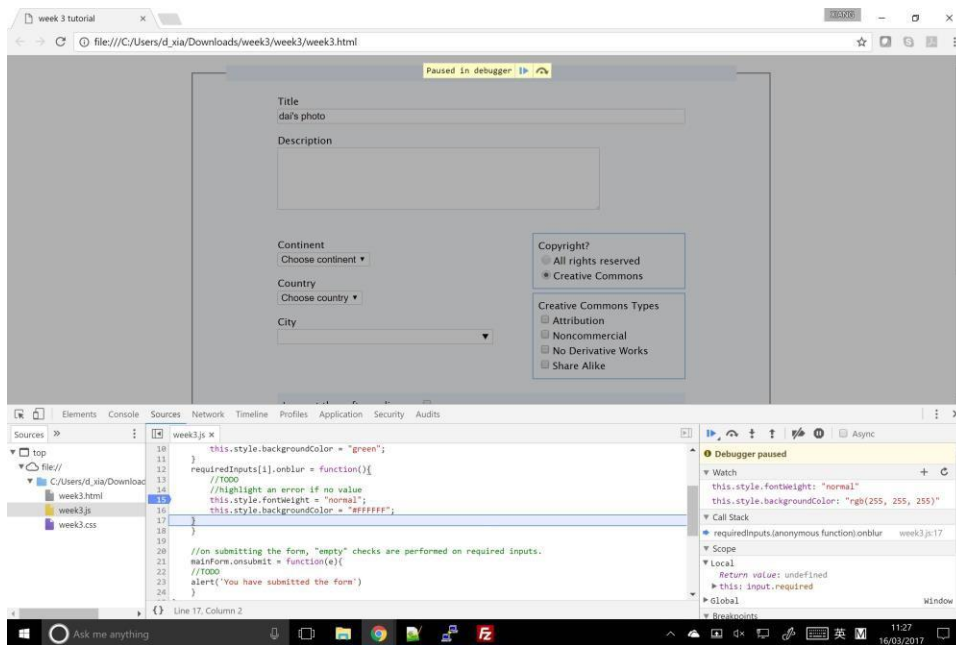


Figure 9: Browser pause when focus moves out