



Final:

This final is comprised of 3 files plus the jQuery library. Name the 3 files final.html, final.css, and final.js and store them all in one folder. The main purpose of this final to build a simple registration html form, perform some JavaScript validation, and include some jQuery functionality.

Part 1: Create the form and implement current input element focus visually

Build the html form as follows, the exact structure is shown below:

The screenshot shows a web browser window with the title 'Final'. The address bar displays 'file:///G:/Documents/Course'. The browser's toolbar includes a search bar and various navigation icons. Below the toolbar, a row of folder icons is visible: 'Getting Started', 'Latest Headlines', 'Finance', 'Databases', 'Community', 'IT Technical', and 'Wisdom, Quotes, etc.'. The main content area of the browser displays a registration form titled 'Form Validation with Javascript/jQuery'. The form is contained within a dark gray box and features the following elements:

- Login:
- Password:
- Confirm Password:
- Email:
- Confirm Email:
- Register:



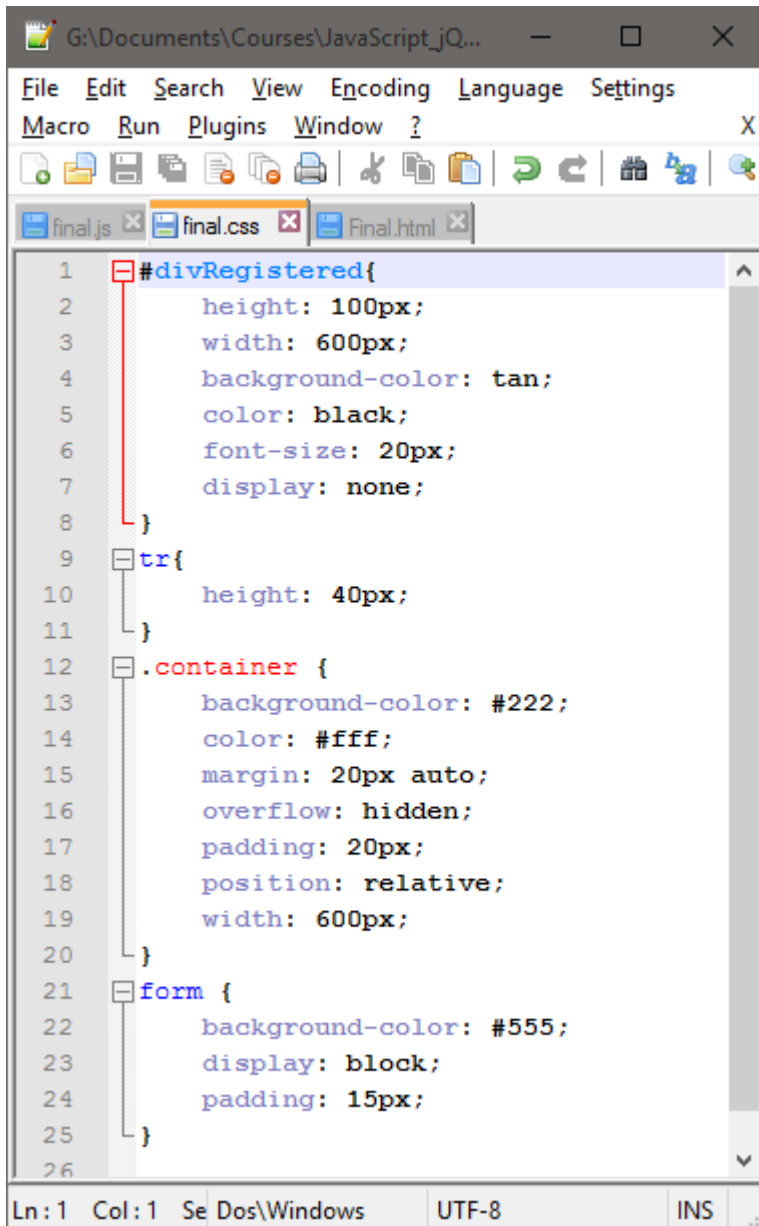
To clarify the form structure the main html tags and their nesting are shown below. Furthermore, some tags that are not visible are also shown here:

```
1 <!DOCTYPE html>
2 <head>
3   <title>Final</title>
4   <link href="final.css" rel="stylesheet" type="text/css" />
5   <script type="text/javascript" src="jquery-2.1.4.min.js"></script>
6   <script type="text/javascript" src="final.js"></script>
7 </head>
8 <body>
9   <h2 class="container">Form Validation with Javascript/jquery</h2>
10  <div class="container" id="divRegistered"></div>
11  <div class="container" id="divMain">
12    <form action="final.html" method="get" name = "frmRegister" id="frmRegister" >
13      <table>
14        <tr>
15          <td>Login</td>
16          <td><input type="text" name="Login" id="txtLogin" />&nbsp;<span id="spnLogin"></span></td>
17        </tr>
18        <tr>
19          <td>Password</td>
20          <td><input type="password" id="txtPassword">&nbsp;<span id="spnPasswordComplexity"></span></td>
21        </tr>
22        <tr>
23          <td>Confirm Password</td>
24          <td><input type="password" id="txtConfirmPassword">&nbsp;<span id="spnPasswordCompare"></span></td>
25        </tr>
26        <tr>
27          <td>Email</td>
28          <td><input type="text" id="txtEmail" >&nbsp;<span id="spnEmail"></span></td>
29        </tr>
30        <tr>
31          <td>Confirm Email</td>
32          <td><input type="text" id="txtConfirmEmail">&nbsp;<span id="spnEmailCompare"></span></td>
33        </tr>
34        <tr>
35          <td colspan="2"><input type="submit" id="btnRegister" value="Register"></td>
36        </tr>
37      </table>
38    </form>
39  </div>
40 </body>
41 </html>
```

Hyper Text Markup Language file length: 1499 lines: 41 Ln: 1 Col: 1 Sel: 0 | 0 Dos\Windows UTF-8 INS



The CSS file is shown below:



```
1 #divRegistered{
2     height: 100px;
3     width: 600px;
4     background-color: tan;
5     color: black;
6     font-size: 20px;
7     display: none;
8 }
9 tr{
10     height: 40px;
11 }
12 .container {
13     background-color: #222;
14     color: #fff;
15     margin: 20px auto;
16     overflow: hidden;
17     padding: 20px;
18     position: relative;
19     width: 600px;
20 }
21 form {
22     background-color: #555;
23     display: block;
24     padding: 15px;
25 }
26
```

- The divRegistered element will be used at the end when the user clicks on the Register button.
- The tr element simply specifies the height of each table row.
- The class container is used for the h2 header and for the two div elements.
- The form element specifies some form styling features.



Now to the JavaScript functionality. Create a text file named final.js.

Steps:

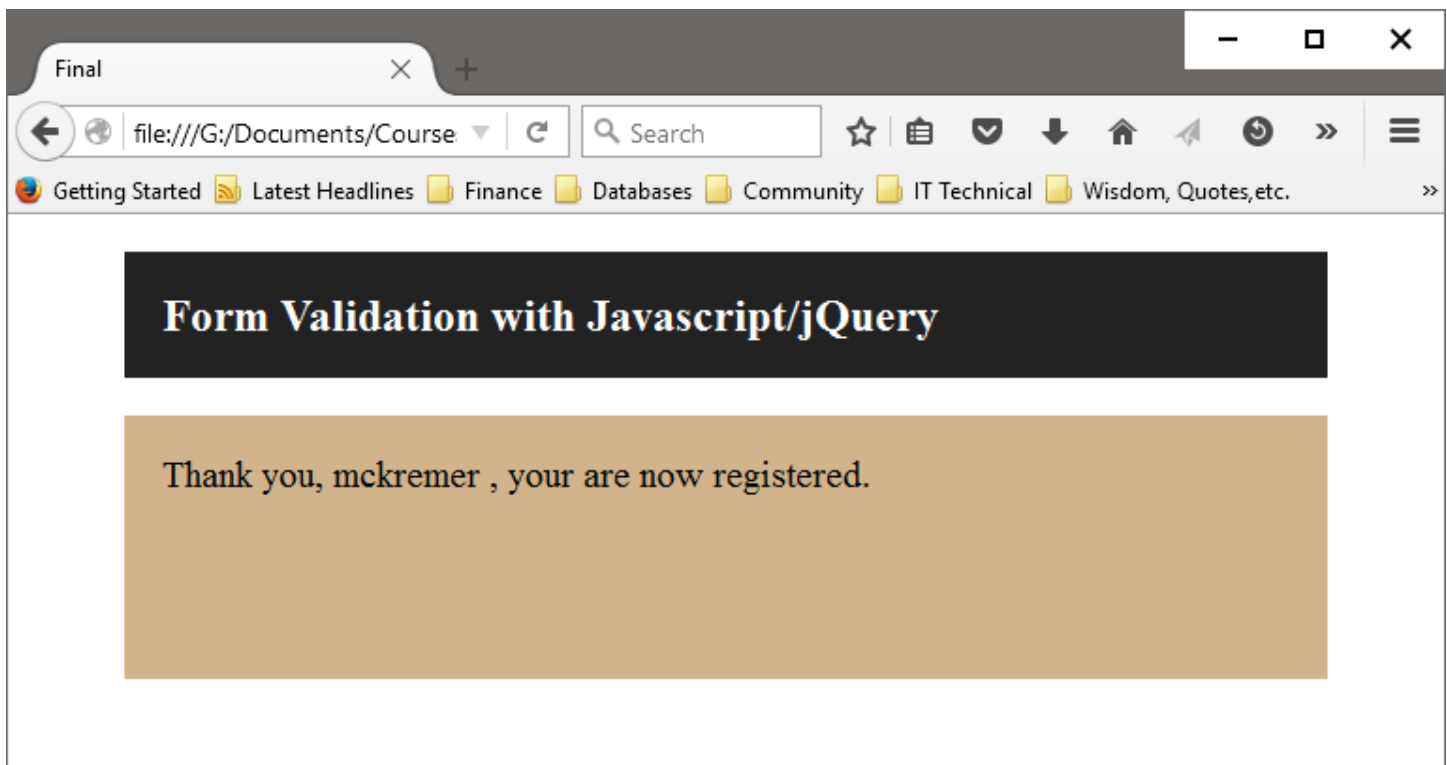
- Create two global variables named divReg and divMain
- Create an DOMContentLoaded event listener using the addEventListener method:
`addEventListener("DOMContentLoaded", function(){ //code })`
- Inside this listener assign the variable divReg the reference to the element divRegistered
- Also assign the variable divMain the reference to the element divMain
- Setup two event listeners at the divMain element using the addEventListener method:
 - One for the focus event, assign the function fHandleEnter, and set the third argument to true to capture the events for all descendants (in our case all input elements).
 - The other one is for the blur event, assign the function fHandleExit, and also set the third argument to true.
- Now setup the two functions outside the DOMContentLoaded event listener:
 - fHandleEnter: Use the function parameter name e or some other name for the event object. This will be used to dynamically identify the actual input element. Refer to the current input element by using e.target. So when an input element gets the focus, change the background color to yellow.
 - fHandleExit: Use the same function parameter name e to remove the background color yellow from the current input element.
- Save everything and test this functionality. Click inside all the input elements and ensure that the background is yellow for the element that has focus.

Part 2: Form processing and displaying registration message

- First perform the following test: Enter something into the Login input and click on the Register button. Notice the URL, it contains a querystring ?Login=..., this is due to the fact that the txtLogin contains a name attribute Login and the button is a submit button. This is the built-in HTML form processing. We will now built some more JavaScript logic around this querystring functionality.
- Inside the DOMContentLoaded event handler invoke the function fProcessForm with no arguments.
- Create a function named fProcessForm outside the DOMContentLoaded event handler. This function will be used to hide the form when the user clicks on the button Register and to slowly display the divRegistered element using jQuery with some text in it.
- Create a variable named strQueryString and assign it the location.search property. This is the same logic as in HW4, and you may use the same code.



- Use the replace method and a regular expression to find everything up to the equal sign and replace it with nothing (same logic as in HW4). Now check whether the length of this string is greater than 0 or not.
 - If it is greater than 0, the Register button was clicked and the form was processed.
 - Create a variable named login and assign it the querystring after the equal sign (using replace logic from above).
 - Using the variable divReg assign the following string to the inner HTML property of divReg: Thank you, {login variable} , you are now registered
 - Then hide the divMain by setting the display style property to none. Use the jQuery method fadeIn on the divRegistered element to slowly show this element. Use any numbers of seconds.
 - If the length is 0, then the form was initially loaded.
 - Hide the divReg element
 - Show the divMain element.
- Save everything and test this new functionality. Enter a login name and then click on the Register button. Your html page should look like this (after the fade in):





Part 3: Compare input of two input elements (Password, Email)

Inside the event listener for the DOMContentLoaded we are now creating two arrays holding all input and associated span elements using the querySelectorAll method:

- Declare a variable inputElements (which will be an array) and use the document method querySelectorAll to include all text input and text password elements within the form frmRegister. Use CSS selector syntax, using the id for the form, then the tag syntax for input elements and further using attribute selectors to specify type=text and type=password.
- This array will hold now all the pertinent input elements in one array variable: Index 0 refers to txtLogin, 1 refers to txtPassword, 2 refers to txtConfirmPassword, 3 refers to txtEmail, and 4 refers to txtConfirmEmail.
- Create a similar array named spanElements and refer to all the span elements within the form using the document method querySelectorAll.
- For testing purposes, create a loop to loop over each array and displaying the element's id in an alert. This is optional and simply for testing purposes of the querySelectorAll method.
- Now we want to write a function that allows us to compare two inputs whether they are exactly the same or not and display a message in the span element next to the second input element (for password and email).
- Create a function outside the event handler for DOMContentLoaded and name it fCompareInput having three parameters:
 - value1: The first value to be compared
 - value2: The second value to be compared against the first.
 - display: The span element to display a message whether the two entirely match or not.
- Check whether one of the values parameter is empty (length ==0), if so, set the inner HTML and the style attribute of the display parameter to an empty string.
- Check whether both values are exactly the same, if so, set the inner HTML attribute to 'Entries match' and set the background color to green.
- If the values are not empty and do not equal, set the inner HTML attribute to 'Entries do not match' and the background color to red.
- Now we need to invoke this function, and we will be using the blur event of the second input control (txtComparePassword and txtCompareEmail) to trigger this function.
- In the event listener DOMContentLoaded set up two addEventListener methods for each of the second input elements (using the array of inputElements[] and the corresponding index) and passing the first and the second value (use inputElements[].value) and the appropriate span element (using spanElements[]). Since you need pass arguments, use the following construct:
`inputElements[index].addEventListener('blur',function(){ fCompareInput(arguments); });`
- Test this new functionality by entering a simple password and retyping the password, once correctly and the other time incorrectly. Do the same for the e-mail inputs.



Part 4:

This is a creative part. Here I want you to come up with your own solution/implementation. Effort as well as uniqueness and creativity will be rewarded. There are quite a few more validation tasks to tackle. Select **one** of the following **or** use your own imagination to design a task on this form. Come up with your own solution:

Login:

- Enforce a minimum and maximum length for the login username. You could use the input event (triggered for every keystroke) and display the current length in span element next to the input element. You may choose a red and green background color in the span element to visually convey to the user whether the length is good or not. And you could also show the number of characters entered.
- Enforce that only certain characters are used, such as alphanumeric and the underscore. The input event is probably a good choice for tracking what the user entered. You may use the span element to display successful or failed login names. You could use regular expression functionality to verify the characters entered.

Password:

- Enforce a password complexity rule of your own choice, such as at least one uppercase letter, one numeric character, one special character, etc. You may use regular expression for this verification. You may display the status of the password complexity in the span element next to the password input. And you could use different background colors for visually showing the password complexity (such as weak, medium, strong, very strong, etc.)

E-mail:

- Verify the validity of the e-mail address. Come up with your own rules here, you may use regular expression functionality. You could use the span element next to the e-mail input to display whether the e-mail is valid or invalid.

All these are suggestions, you may deviate as much as want. Keep it simple, I am not looking for perfect solution here. But I want you to bike on your own without training wheels.

Uploading the final:

The final is due just before the last class, Zip up all your files and name the zip file FE_{your name} and upload it at the homework website.