| Sprint 10, Assignment 3.10 Please also update the doc name with correct numbers. | |
| --- | --- |
| Assignment type: JS Interactive | |
| Assignment name  Can remain the same as the assignment topic name, or…  can be phrased in the active/imperative voice (depending on the task), e.g. “Create your own X” | “Create a Form with Listeners” |
| Description  The essential tasks a student must perform in order to complete the assignment  ?+ a general, colorful introduction to the assignment? | After checking out an online job board, you've taken the following contract: the customer would like for you to create the front-end submission form for their site. They say they'll need all the standard pieces of shipping info for their customers. In order to tackle this problem, you will need to break the project into sections (spread over the following three assignments). In this assignment, create a basic form - and don't worry about buttons etc.; just   * Create the basic html layout, * Give it the ability to take in and pass the user's entries, and * Create some basic listeners to verify that what you've made is working as expected.   Also, note that the client would like your code file to be detailed in its comments and log messages (so that they can make changes down the line quickly). Try annotating and logging your code as clearly and succinctly as possible - imagine that you are adding helpful labels so that the client can operate your code “machine” on their own! |
| Link to full code file (Answer)  Push a full code file to our [GitHub repo](https://github.com/bitdegree-foundation/academy-assignments-code-files) for this sprint’s assignment and paste a link to that file here. Please write the file in a concise and clear format, according to the “[example-js-code-file](https://github.com/bitdegree-foundation/academy-assignments-code-files/blob/master/example-js-code-file.js)” found in our repo. | (Upload) |

| # | Step  Write each small step of the task. These assignments should be doable by students on their own (without BitDegree-instructor help). It can be any number of steps, but keep them pretty clear and separate (don’t combine 2 steps into 1 step). | 1x hint  Write brief text or partial code that will help the student figure out how to either move forward or completely accomplish the given step. | Output Expected code that our platform will be able to run & check against student submissions |
| --- | --- | --- | --- |
| 1 | Create an html document and paste the following "starter" text into it:  <!DOCTYPE *html*>  <html *lang*="en">  <head>  <meta *charset*="UTF-8" />  <meta *http-equiv*="X-UA-Compatible" *content*="IE=edge" />  <meta *name*="viewport" *content*="width=device-width, initial-scale=1.0" />  <title>Sprint 10, Assignment 3.10 ("Create a form with listeners")</title>  </head>  <body>  <script *src*="./3.10.Assignment.script.js"></script>  </body>  </html> |  | <!DOCTYPE *html*>  <html *lang*="en">  <head>  <meta *charset*="UTF-8" />  <meta *http-equiv*="X-UA-Compatible" *content*="IE=edge" />  <meta *name*="viewport" *content*="width=device-width, initial-scale=1.0" />  <title>Sprint 10, Assignment 3.10 ("Create a form with listeners")</title>  </head>  <body>  <script *src*="./3.10.Assignment.script.js"></script>  </body>  </html> |
| 2 | In the body of your HTML document, add a (parent) "form" element. Give it an ID, such as "signUpForm", and add some temporary text (like "...A form will go here...") to make sure that your html is visible & working. | Hint: "form" is one of many html elements, and is written just like <div>, <li> and other elements! | <!DOCTYPE *html*>  <html *lang*="en">  <head>  <meta *charset*="UTF-8" />  <meta *http-equiv*="X-UA-Compatible" *content*="IE=edge" />  <meta *name*="viewport" *content*="width=device-width, initial-scale=1.0" />  <title>Sprint 10, Assignment 3.10 ("Create a form with listeners")</title>  </head>  <body>  <form *id*="signUpForm">... A Form will go here ...</form>  <script *src*="./3.10.DEV.script.js"></script>  </body>  </html> |
| 3 | Create fields for the user to input data during their sign-up (as nested elements of the form element). You should have fields for the following seven pieces of data: First Name, Last Name, Address, Apt #, City, State/Country, and Phone #.  Note: Make sure to label each field with comments, so that the html is more read-able | Hint: In Lesson 3.8, we used "label" & "input" elements to create the fields, and "break" elements to seperate them | <!DOCTYPE *html*>  <html *lang*="en">  <head>  <meta *charset*="UTF-8" />  <meta *http-equiv*="X-UA-Compatible" *content*="IE=edge" />  <meta *name*="viewport" *content*="width=device-width, initial-scale=1.0" />  <title>Sprint 10, Assignment 3.10 ("Create a form with listeners")</title>  </head>  <body>  <form *id*="signUpForm">  ... A Form will go here ...  <br />  *<!-- First Name -->*  <label *for*="fname">First Name:</label><br />  <input *type*="text" *id*="fname" *name*="fname"></input><br />  *<!-- Last Name -->*  <label *for*="lname">Last Name:</label><br />  <input *type*="text" *id*="lname" *name*="lname"></input><br />  *<!-- Address 1 -->*  <label *for*="address1">Address:</label><br />  <input *type*="text" *id*="address1" *name*="address1"></input><br />  *<!-- Address 2 -->*  <label *for*="address2">Apartment etc.:</label><br />  <input *type*="text" *id*="address2" *name*="address2"></input><br />  *<!-- City -->*  <label *for*="userCity">City:</label><br />  <input *type*="text" *id*="userCity" *name*="userCity"></input><br />  *<!-- State/Country -->*  <label *for*="userState">State / Country:</label><br />  <input *type*="text" *id*="userState" *name*="userState"></input><br />  *<!-- Phone # -->*  <label *for*="userPhone">Phone #:</label><br />  <input *type*="text" *id*="userPhone" *name*="userPhone"></input><br />  </form>  <script *src*="./3.10.Assignment.script.js"></script>  </body>  </html> |
| 4 | Create a JavaScript file and name it 3.10.Assignment.script.js |  |  |
| 5 | Store & log the (parent) "signUpForm" element in a variable | Hint: use querySelector | const form = document.querySelector("#signUpForm");  console.log(form); |
| 6 | Store & log all of "signUpForm"s children | Hint: what elements are available that can tell us all of "form's" children? In your web browser console, try entering "form.ch" and see what is suggested ...  Hint: use .children | const formElements = form.children;  console.log("Here's the form.children elements:");  console.log(formElements); |
| 7 | Store \*only\* the children that users will use to input their personal information into individual variables | Hint: there are multiple ways to do this. For now, try declaring variables equal to form.children.fName, .lName, .address1 etc | const fName = form.children.fname;  const lName = form.children.lName;  const address1 = form.children.address1;  const address2 = form.children.address2;  const userCity = form.children.userCity;  const userState = form.children.userState;  const userPhone = form.children.userPhone; |
| 8 | Add an event listener that, when the "form" parent element is moused-over, log's a message like "User considering sign-up ..." | Hint: use mouseenter | form.addEventListener("mouseenter", function(){console.log("User considering sign-up ...");}); |
| 9 | Add an event listener that, when the user's mouse exits the "form" parent element, logs a message like "User LEFT sign-up ..." | Hint: use mouseleave | form.addEventListener("mouseleave", function(){console.log("User LEFT sign-up ...");}); |
| ... |  |  |  |