Homework Questions Part 2:

1. What ***IS*** form validation?

- Form validation is used to occur at the server, after the client enters all the required data in the correct format and presses the submit button. Clients are provided with feedback regarding the data that they have entered in the various form inputs.

2. What are the two types of client-side form validation?

- The two types of client-side form validation is using html5 built in form validation, which doesn't use Javascript and the constraint validation API, which is another form using Javascript.

3. What exactly is html5 built-in form validation? Describe the one ***attribute*** that needs to be **added** to the ***form input element*** in order for **built in form validation** to ***work*** at a **minimum**?

- The html5 built-in form validation is the ability which validates most user data without using Javascript. It's achieved using validation attributes on form elements.
- One example of an attribute that needs to be added to the form input element is the input boolean attribute.

4. Use the different input elements (username, password, email) we have talked about and demonstrated in class as examples of how to implement html5 built-in form validation. You can and SHOULD take from the form which you have to build for your next project as examples. That's the idea here. A number of you used code from the Arithmetic Forms to provide examples in answers to the questions for the week 3 homework. That was really excellent! I would like to see the same happen here.

```
<input type="text" name="username" placeholder="create
username" pattern="^(?=.*\d)(?=.*[a-zA-Z]).{7,15}$"
required>
<input type="email" name="mail" placeholder="enter your
email"
pattern="(\b[a-z0-9._%+-]+@[a-z0-9.-]+\.[a-z]{2,}\b)"
required>
<input type="password" name="password"
placeholder="create a password"
pattern="^(?=.*\d)(?=.*[a-z])(?=.*[A-Z])\w{13,24}$"
required>
```

5. What is the main difference between HTML5 form validation and the JavaScript Constraint Validation API?
   - The main difference is that HTML5 form validation does not us JavaScript and the constraint validation API uses JavaScript.

6. What is the input attribute you would NOT want to use when creating a form with the JavaScript Constraint Validation API?
   - The input attribute that we wouldn't use is the "title attribute" because it overrides any messages set up via JavaScript using the ".setCustomValidity( )" method.

7. Is client-side form validation enough by itself? Why ***not***?
   - The client-form validation is not enough by itself because the user inputs are not secure. Security checks can only be done server-side and client side validation is easily manipulated or disabled. The purpose of website security is to prevent users' information from being leaked or exposed.

8. Why would we want to use form validation when at the same time we want them to have the best UX possible on our site? Give ***three reasons*** why we would want to use **form validation** on our users.
   - We want to use form validation at the same time to have the best UX on our site because

- we want to get all the client's data in the right format.
- we want to protect all the users' data by forcing the users to input secure passwords, which helps protect the information they have shared with us.
- we want to protect ourselves from malicious hackers.

9. What is one of the most significant features of HTML5 form controls? How is this achieved?

- The most significant features of HTML5 form controls is the ability to validate most user data without Javascript. This is achieved by using validation attributes on form elements.

10. List **seven attributes** used in **built-in form validation**, and ***describe*** what they **do**.

- Seven attributes used in a built-in form validation are
    1. **required**: specifies whether a form field needs to be filled in before the form can be submitted
    2. **minlength** and **maxlength**: specifies the minimum and maximum length of textual data (strings)
    3. **min** and **max**: specifies the minimum and maximum values of numerical input types
    4. **type**: specifies whether the data needs to be a number, an email address, or some other specific preset type
    5. **pattern**: specifies a regular expression that defines a pattern the entered data needs to follow