Form Validation

Validation with JavaScript

- Validating form input with JavaScript is more widely supported than the HTML validation.
- However, validation with JavaScript still requires JavaScript, which is not a given in every browser.
- Data is still under the user's control even after JavaScript has validated it.
- Therefore, you must validate form data on the server.

Where Are We?

- We've seen a good deal about JavaScript in the past lectures:
  - Basic syntax, including if/else conditions
  - Functions and Built-in Objects
  - The Document Object Model, including style changes
  - JavaScript Objects
- These four things make form validation rather trivial:
  - Add an event handler to the form submit
  - Compare values entered by user against your version of the truth.
  - React appropriately

The Default Action

- One important concept around form validation is around the default action.
- When the submit button is clicked, the default action of the form is to submit to the server or target of the "action" attribute.
- Part of form validation is preventing this default action from happening until we have validated the input and then preventing it from happening if validation fails.

Preventing Default

- The event.preventDefault() method stops the form from actually submitting.
- Returning false is also typically used so that the event stops propagating through the DOM.
- As you might expect, older IE's don't have preventDefault().

Handling Default Cross-Browser

- For most cases you can simply return false; to prevent default action, which will work for most browsers.
- If you need the preventDefault() method, then add this for old IE:

```
if (!evt.preventDefault) {
    evt.preventDefault = function() {
        window.event.returnValue = false;
    };
}
```

The Opposite is True

- Using return false; prevents the form from submitting to the server.
- Use return true; to allow the form action to continue – such as would be desired when the validation "passes".

Nested Function for Validation

- See D2L for an example of an event listener for a submit on a form.
- Demo

RegExp

- We introduced regular expressions previously.
- JavaScript has a RegExp object that can be used for building complex expressions.
- This is especially useful for form validation.
- Instantiate with:
- var myRegex = /pattern/; // No Quotes!
- //OR
- var myRegex = new RegExp('pattern');

Same Rules But Different

- JavaScript regexes are generally more powerful than the browser based ones in HTML5 (right now).
- You can use anchoring and grouping.
- Methods that are relevant:
  - test
  - exec

Test and Exec

- The test method tests the regular expression against the given value and returns true or false based on whether the value matches.
- The exec method tests the regular expression and returns information from the match (if you used grouping) or null if no match.
- Use test when you want to test and use exec when you want to process the value.

Try it out: RegExp

- Remove the existing JavaScript from your test page.
- In its place, let's build a regex that tests for a zip code.

String Match

- Another way to validate is to use the match() method of a string.
- Whereas the RegExp object syntax is:
- RegexpObject.test("string")
- The match syntax is the other way around:
- String.match(RegExp)
- Unlike the test() method, the match() method returns the matches.
- Can use !== to simply compare two strings.

Put It Together

- Now we've seen how to add a form handler for submit.
- We've encountered keypress/blur events in a previous lecture.
- We've seen how to change styles and classes in a previous lecture.
- We've seen test/match for comparing strings and we've seen comparison of numbers.

Reacting with JavaScript

1. Setup a form handler
2. Test conditions
3. Alert the user

- Demo

Best Practices

- Always give the user an idea of what is expected before they fill it out wrong.
- Clear invalid fields on blur after the initial validation. Doing so gives the user a signal that they completed the task correctly.
- Provide messaging near the problem, near the form field that is invalid, when possible.
- Provide visual but not obnoxious indication of fields that are invalid.
- Gather all inputs that are invalid before returning.

Summary

- JavaScript forms validation is a collection of techniques from the core JavaScript language along with tools and methods related to the browser.

1. Connect the submit handler
2. Add test conditions
3. React and alert the user