Lecture Material - Week 4

Simple Forms

- JavaScript used in forms for form validation
- **forms**: used to pass data
  - adds new layer to request-response cycle
  - have different element types
  - must be associated w/ a server to really work (back-end)

- **front-end web development** = what happens on the browser/client side (this program lol)
  HTML, CSS, JS

- **back-end web development** = what the server is handling
  Python, Java, Ruby, PHP, Perl

- HTML form tags: <form>
  <label>
  <input>

- **<form>** = tells a browser there's gonna be a form & will be processing some data

- **<label>** = what matches different text on the screen w/ the actual input
  can also activate the input field by being clicked (not just the input field box)

- **<input>** = inserts a input field (different types of input based on type attribute)

- form element attributes: type
  name
  id
  value
  placeholder

  (type, name, id: most common)

- **input types**: textfield
  email
  password
  radiobutton
  checkbox
  submit
  number
  range
  color
  date
  url

- **name**: should be included in almost all input types

- **id**: used for labels

used for by JavaScript

- **value**: does different things depending on input type
  - → (eg. button): value becomes text inside the button
    permanent, can't change value
  - → (eg. textfield): default value passed to the server if not changed
    changeable

- **placeholder** = provides a suggestion to let user know what kind of input/format you're expecting
  nonpermanent → gone as soon as input field is clicked
  not an official value

Simple Validation

- what to validate: type of input
  - ✓ number ✗ string
  format of input
  - is email address valid? url?
  - does phone # have spaces/parentheses?

  value of input
  - should it be required?
  - do email values match?

- how to validate: use new HTML5 input types (email, number, url)
  use HTML5 attributes (required, placeholder, min/max)
  use JavaScript functions (write custom code)
  ] support not guaranteed

- validation via input types: requires browsers to validate format
  if supported & non-valid input inputted → halts submit process & highlights non-valid input
  if unsupported → input type is just text → non-valid input won't get caught

- validation via input attributes: required
  pattern
  min/max/step - place limits on # inputs

- **required**: halts submit process if required field is empty
- **novalidate**: overwrites required attribute → doesn't require required to be filled
  useful while testing
  - format: <form novalidate>

- **pattern**: works w/ input = text
  requires input to have a specific form set by regular expressions
  best used w/ placeholder & supporting text

- find patterns: http://html5pattern.com/

  → (eg. [0-9]{5}): "can only enter #s & there has to 5 of them"
  pattern for zipcodes

  → (eg. [0-9]{13-16}): "can only enter #s & there has to be 13-16 of them"
  pattern for credit cards

  → (eg. [a-zA-Z]+): "can only enter lowercase & uppercase letters"
  possible pattern for usernames

- min = input ≥ min
- max = input ≤ max
- step = can only enter #s in increments of given value

- validation via JavaScript: to ensure correctness since browsers may not support input type/pattern attribute
  events + custom JS functions = extra validation

Comparing Inputs

- email comparison: what?
  - compare 2 emails are the same

  how?
  - HTML: input=email
    required attribute
  - JavaScript

  when?
  - as soon as second email is entered
  - on submit

  event?
  - onchange = checks once input field is exited
  - oninput = checks every single time a character is typed

  compare?
  - inputs? or some attribute of the inputs

  output?
  - how to communicate comparison result?

- return: functions can return values

- if submit type sees a false → halts submit process

Checkboxes & Radio Buttons

- checkboxes: typically shaped as squares
  allows user to select option(s) w/ a single click
  (in JS code) all options share the same name

  radio buttons: typically shaped as circles
  can only select 1 option/group w/ a single click

- in JavaScript, boolean expression via checked attribute
- checked = how JavaScript communicates w/ checkboxes & radio buttons
  typically used to handle interactivity

  → (eg. checked = "true"): preselects option

- purpose of forms: give user an easy way to interact w/ the page
- you can't fully implement forms w/o back end programming
- free form builders: http://www.wufoo.com/
  google forms

PASSED!