What is a form?

- adds interaction to a website
- user can enter data (text, checkbox, select)
- data is sent to server, server returns new page
    - OR JS submits data and gets a response without a new page

Form Element

Foundation of a form:

```html
<form action="/some/url/" method="POST">
  <button type="submit">Submit</button>
</form>
```

- Sends any data from form to `/some/url`
- kind of similar to a link (navigates)
- No inherent UI (block)

Form Method

```
<form action="/some/url/" method="POST">
```

- Sends data using POST method
- GET is what normally loads a page
    - gets url (navigates)
    - might have data in url query paramaters
        - should NEVER changes the server state
    - never sends any body in request
    - hint: `<a>` tags cause a GET request
- POST also gets a response (navigates)
    - can send data in query parameters AND/OR body of request

URL Encoding

- A conversion to make text safe for a url (query-params)
- Also used in POST form request bodies
- Forms CAN use other encodings
    - particularly for file uploads
    - url encoding is appropriate for simple text data

How to URL Encode

- Happens automatically from forms
- Manual process:
    - spaces become `+` (or `%20`)
    - key-value pairs are `key=value` (no spaces)
    - multiple key-value pairs separated by `&`
    - Special characters replaced
        - `%` then 2 digit hex ASCII
        - `%20` is a space
        - `+` is `%2b`
        - `?` is `%3f`
        - `&` is `%26`
        - `%` is `%25`
        - `#` is `%23`

Input element

```
<input name="demo1" type="text"/
<input name="demo2" type="checkbox"/>
<input name="demo3" value="cat" type="radio"/>
<input name="demo3" value="cats" type="radio"/>
```

When data is sent, it is sent as key/value pairs

```
demo1=whatwastyped
demo2=on
demo3=cats
```

url-encoded:

```
demo1=whatwastyped&demo2=on&demo3=cats
```

Input text field

```
<input name="demo1" type="text"/>
```

Notable attributes

- `type` (text is default)
- `name`
- `value`
- `placeholder`
- `disabled`
- `readonly`
- (validation covered later)
- (a11y covered later)

Other text-like inputs

Change `type` for related text-like inputs

- `password` (hides characters from view)
- `hidden` (hides field, passes value)

Recent(ish) additions:

- `color` (graphical input, textual value)
- `date` (text or cal input, textual value)
- `email`
- `number`
- `search`
- `tel`
- `time`
- `url`

Checkbox

- Sends value (default "on") when checked
- Doesn't even send field when not checked
- `checked` attribute to pre-select

```
<input type="checkbox" name="it-is"/>
<input type="checkbox" name="already" checked/>
```

☐ ☑

Radio buttons

- Only one of same name can be selected
- Name didn't age well
- uses `checked` as well
- no unselecting

```
<input type="radio" name="favorite" value="maru">
<input type="radio" name="favorite" value="nyan">
<input type="radio" name="favorite" value="grumpy">
<input type="radio" name="meh" value="labrador" checked>
<input type="radio" name="meh" value="poodle">
<input type="radio" name="meh" value="retriever">
```

○ ○ ○ ◉ ○ ○

Select dropdowns

- `<option>` tags inside a `<select>` element
- `name` of `<select>`, `value` of `<option>`
- `selected` attribute on `<option>`
    - or first one (always a selection)
- Note: `value` is sent, not content
    - unless no value

```
<select name="cats">
  <option value="rule">Rule the World</option>
  <option value="awesome">Are Awesome</option>
  <option value="inspire">Inspire Me</option>
</select>
```

Rule the World

Textarea element

- NOT an `<input>`
    - mostly same attributes
- content is value, not `value` attribute
- multiline input
- default resizable (CSS `resize` can change)
- is a natural `inline-block`!
- `wrap` attribute (either "soft" or "hard") sets text wrapping

```
<textarea name="blahblah"></textarea>
```

Label element

Forms should have text labels describing them

- don't use placeholder for this

`<label>` element for that text

Two ways to use:

- with a `for` element w/value of input id
- as a parent of the input/textarea/radio group
  - no `for` needed (and no `id` needed)

Not only text, but selecting label selects the field

- great for accessibility and/or mobile!

Fieldset and legend

- A `<fieldset>` element groups 1+ labels and fields
- A child `<legend>` element labels the fieldset

Styling these in different browsers can be challenging

- Investigate before committing to it

What to consider with a form

- Communication
    - Telling the user what to do
- Validation
    - Having the user fix input
- Accessibility
    - Making sure everyone can interact

Form Communication

What am I filling in?

- Do I even notice the fields?

What is required?

- The more we ask, the more often they give up
- `*` convention (backed with hint)

What is the expected value?

- Syntax?
- Data type in general

Form Validation

Ensuring data is acceptable

- May be done before data is sent
    - HTML validation
    - JS-based validation
- May be done after data is sent
    - Server returns a form requesting changes

Make sure they know what to fix and how to fix!

- Per field hints is best
- Often a top-level indication that fixes are needed

Form Accessibility

Forms are often most important part for usability

- often the worst accessibility

Great visuals may not translate well!

- don't fall in love with effect until you are sure

Screen readers read text

- Do they know what to read?
- Does it have the necessary context?

More on a11y later

Form Layout: 2 Column

Labels on one side, fields on other

Pros/Cons:

- Everyone argues, studies are..."thin"
- Arguably better for longer forms
- Disliked in the current designer meta
- Easy to layout in a few ways

Important details:

- Whitespace makes it easier for users
- Align text with edges
- Avoid big gaps between label and field

Form Layout: 1 Column

Labels above (or below - ick!) fields

Pros/Cons:

- Some tests show users are faster to fill out
    - Better "conversion rates"
    - Fewer leave before finishing

Other Form variations

- Multi-step form
    - Break form (in any layout) into multiple forms
    - Can "breadcrumb" or "step" navigation
    - Lulls the user (or lulz the user, ha!)
- Accordion or Folding form
    - Form in one page, but broken among collapsible sections.

HTML Form Validation

Making sure the data entered is correct

- Server side
    - MUST HAPPEN!
    - Can be slow
    - Backend devs are less UI oriented
- Client side
    - For convenience, not security
    - Faster
    - Can be HTML-based or JS-based

HTML-based Form validation

- Enforced by browser
    - Some may dislike default behavior
    - limited intelligence/combinations/UI
- CSS can alter/extend behavior

Example: `required` attribute

- What if you have multiple?

regex pattern validation

Many inputs can validate against a "Regular Expression" (Regex)

- A lot of hate in the world against Regex
  - But it is very powerful
- Many like various "tool" websites
  - I prefer to learn the actual items
  - But I also coded Perl, so....

CSS for validation

The `:required` pseudo-selector matches

- e.g. `input:required`

The `:invalid` pseudo-selector matches flagged input

- e.g. `input:invalid`
- Immediate, before you can ever type
    - This limits the usefulness without JS

JS can add/remove classes as values change

- offers more detailed styling
- more later

Summary - Forms

- Add interaction options
- load a page
- can send data with request
- GET (params in URL) or POST (params in body)
- name=value pairs URL encoded
- form element (action and method)
- input elements (various 'types')
- select elements (with option elements)
- button elements
- label elements

Summary - Form Validation

- Frontend: HTML-based or JS-based
- Backend: can return different pages based on data
- HTML-based has limited options
    - required
    - regex pattern + invalid
- Make sure messages are visible
    - including a11y concerns

Summary - Form Layout

- 1 column, 2 column, multi-column
    - be deliberate
- avoid too much space between labels and fields
- consider alignment of fields and labels