CHT2520 Advanced Web Programming

Matthew Mantle m.e.mantle@hud.ac.uk

No Classes Next Week

- All classes next week (w/c 2nd December) have been cancelled
- Replacement sessions will be run in w/c 6th January
  - Hand-in the assignment on 10th January
  - These sessions are optional (i.e. not attendance monitored)

Today's Session - JavaScript Frameworks and Build Tools

JavaScript

- HTML was not designed for dynamic content
- JavaScript applications are largely about making HTML dynamic
  - Using JavaScript we hide/show/add/remove DOM elements, change the content of elements etc.
- When building complex front-end applications we get problems
  - Tight coupling of HTML and JavaScript
  - Empty HTML files
  - Nested functions and lots of callbacks

JavaScript - Frameworks

- JavaScript frameworks were developed as a solution to this problem
- There are lots of JavaScript Frameworks
  - React, Vue.js, Svelte.js, Angular.js etc.
- They don't all work in the same way but do share common principles
  - Component driven
  - HTML is more declarative
  - Handle changes to the HTML automatically
- We will focus on React

Frameworks are often Component Driven

- Components group the HTML and JS for a UI element
  - React does this using a markup syntax called JSX

```
function FilmLink({ title, id }) {
  return <a href="/films/{id}">{title}</a>;
}
```

- We can then use the components like HTML elements

```
export default function MyApp() {
  return (
    <div>
      <FilmLink title="Winter's Bone" id="3" />
      <FilmLink title="Gravity" id="10" />
      <FilmLink title="Arrival" id="11" />
    </div>
  );
}
```

The HTML is more Declarative

- In React, using JSX, we specify what the final element will be like

```
function FilmsHeading({ decade }) {
  return <h1>Films from the {decade}s</h1>;
}
```

- Much clearer than

```
const filmsHeading = document.querySelector("#filmsHeading");
filmsHeading.innerHTML = "Films from the " + decade + "s";
```

- Frameworks take care of data binding.
  - If a variable changes value, the HTML updates automatically (we don't have to manually use `innerHTML`)

JavaScript - Build Tools

- We browsers don't understand JSX

```
function FilmsHeading({ decade }) {
  return <h1>Films from the {decade}s</h1>;
}
```

- We need to 'transpile' the JSX code into plain JavaScript
- To do this(and other tasks) we use we use a build tool

JavaScript - Build Tools

- There are lots of front-end build tools (sometime called bundlers)
  - Vite, Webpack, Rollup, Parcel
  - We will use Vite
- Typically search through code and
  - Discard unused/unneeded code (tree shaking)
  - Transpile into plain JavaScript
  - Minify files (JS and CSS) to produce optimised
- Installed as Node.js modules (like we did with TailwindCSS)
- Use them from the command line

Using JavaScript Frameworks

- React can be used in a number of different ways:
  - To add a feature as part of a larger web app (this is how I expect you to use it)
  - To build Single Page Apps (SPA)
    - Load a single page from the server
    - All user actions update the contents of this one page
    - Usually built using a dedicated full-stack framework e.g. Next.js (outside the scope of this module)
  - React Native
    - Using React to Build Native Mobile apps (outside the scope of this module)

Integration with Laravel

- Laravel is set-up ready to use Vite
  - See: https://laravel.com/docs/11.x/vite
- Inertia is Laravel's approach to complex (SPA) type apps
  - Returns an component (doesn't return a Blade view)
  - I'd advise against using Inertia unless you are really confident about what you are doing