Frameworks

Purpose of a framework

- Basic functionality already available
  - Python can create network listeners, manipulate strings etc.
  - JS can extend elements, use APIs to manipulate documents
- Problem:
  - Lots of code repetition - boilerplate
  - Reinventing the wheel - different coding styles, techniques
- Solution:
  - Standard techniques for common problems - design patterns
  - Frameworks: Flask for Python web apps, React for JS components
- SPA: Single Page Application
  - Many JS front-end frameworks focus on enabling this - also useful for mobile

Example - React

Library for building user interfaces

- Declarative
  - Opposed to imperative - specify what is needed, not how to do it
- Components
  - Different from WebComponents - similar ideas, different techniques
  - Webcomponents are more imperative: functions that specify behaviour
  - React is declarative: focus on UI, but allow composing views
- Examples: https://reactjs.org/

Frameworks

- React - numerically most popular at present
- Angular - origins from Google - well supported
- EmberJS - component + service framework
- Vue

https://developer.mozilla.org/en-US/docs/Learn/Tools_and_testing/Client-side_JavaScript_frameworks/Introduction

Summary

- HTML5 is a living standard - no major changes, but continuous adaptation
- JS provides the adaptation layer
- HTML + CSS + JS = rule the world!
- But difficult to code
- Frameworks fill in the gaps

Front-end development for dynamic applications