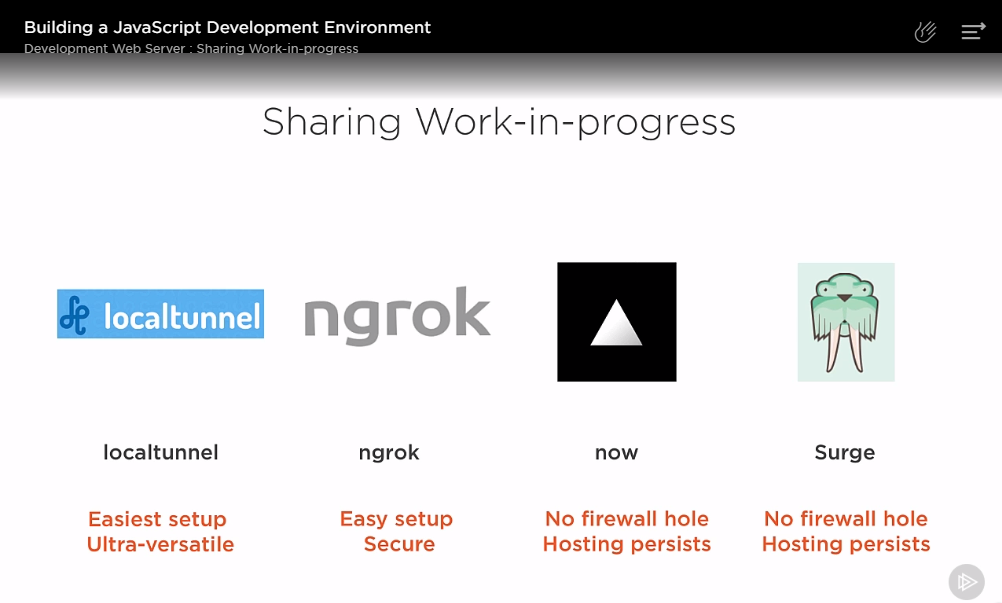
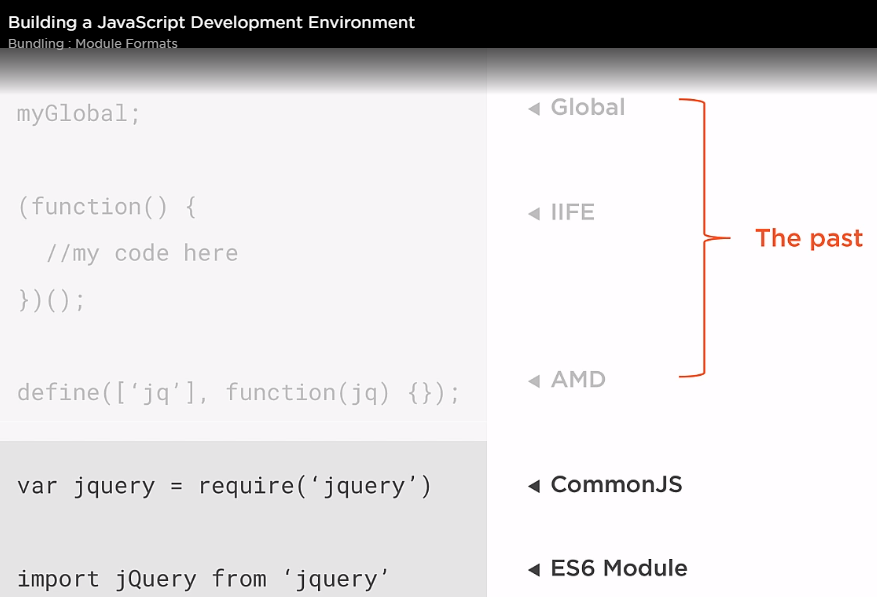
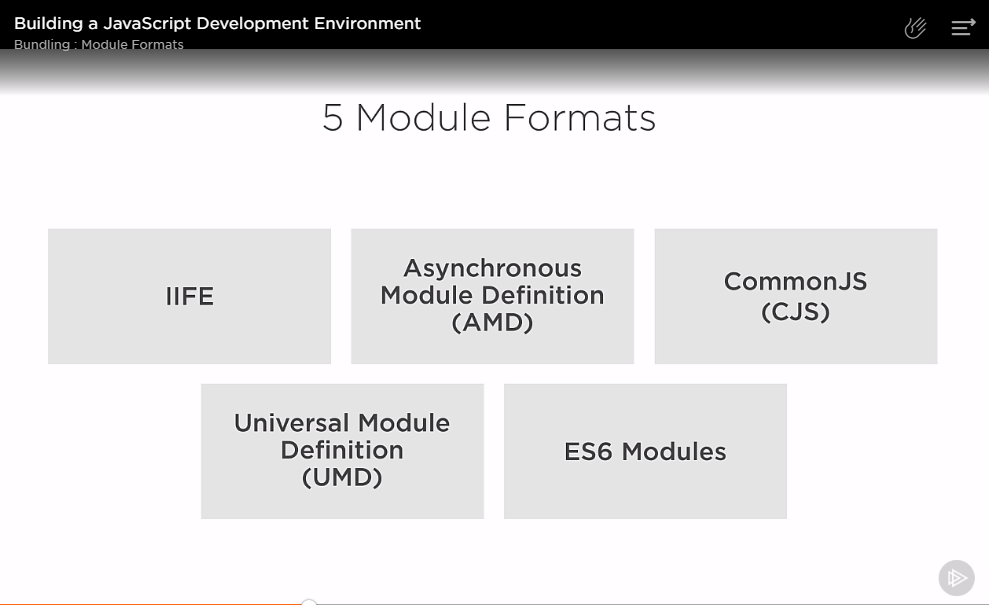
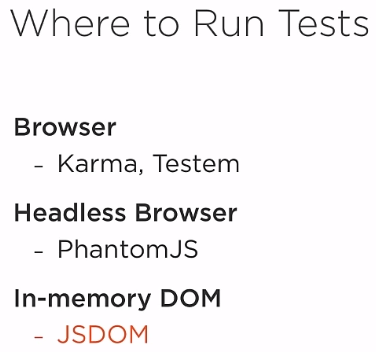
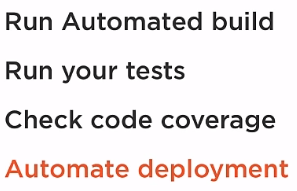
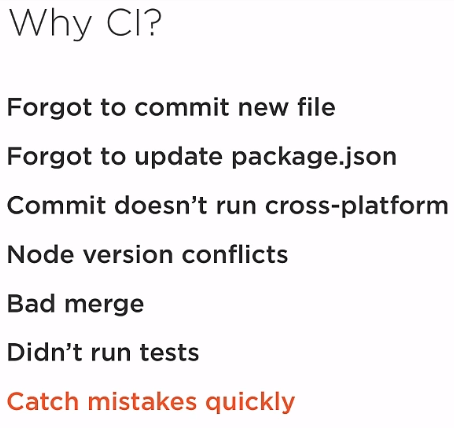
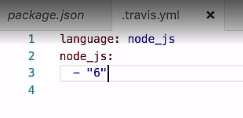
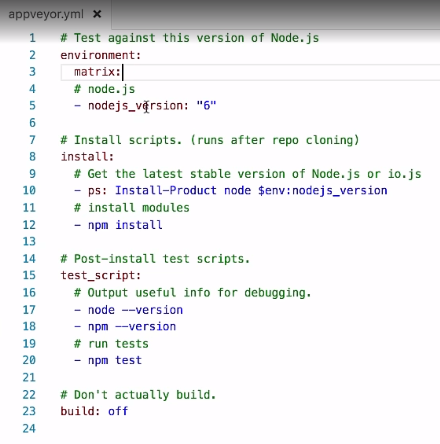
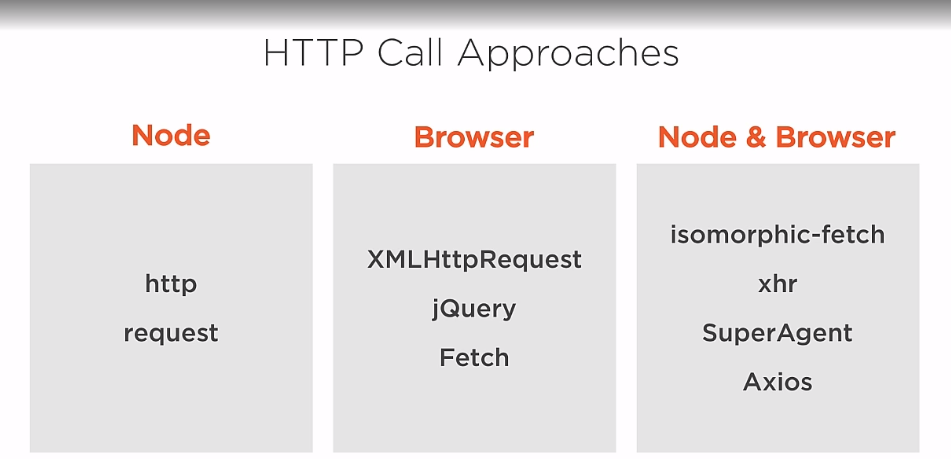
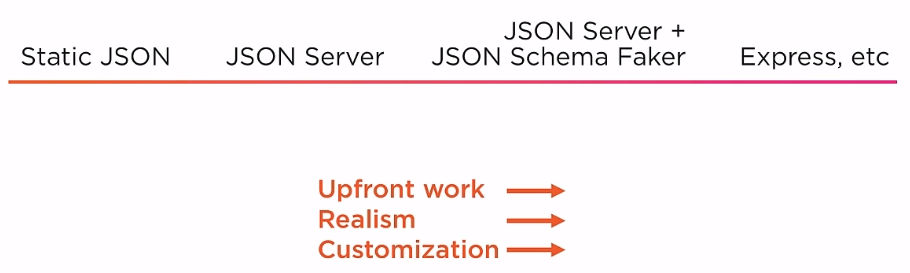
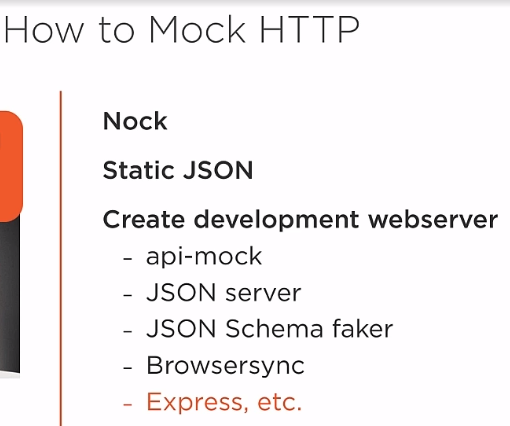
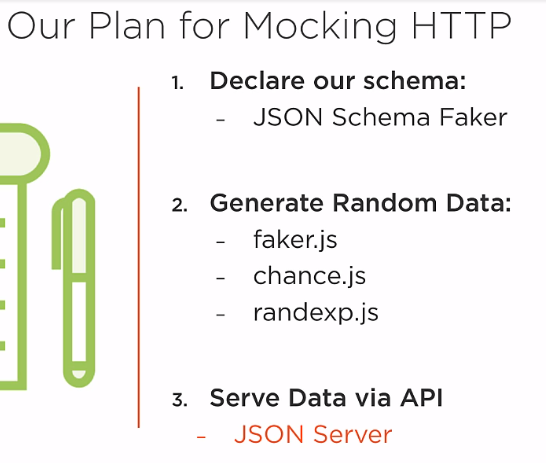
Notes: Create Javascript Development Environment

* Setup Versioning
  + Create git repository and clone it down to local
* Setup IDE
  + Create .editorconfig and install plugin for Visual Studio Code
* Setup Package Manager - Install Node and NPM
  + Do npm init and npm install with starting packages
* Setup Security for packages
  + Install Node Security Platform (alternative for retire.js) and do nsp check to check secure packages
    - Remember to uninstall globally
* Setup Development Web Server
  + Browserify and Webpack are also bundlers
  + Express is part of MEAN framework
  + BrowserSync – Run your app in sync between browsers (helpful for testing)
  + 
    - Install localtunnel – remember to uninstall globally
* Setup Task Automation
  + Grunt, Gulp or NPM scripts
    - Bit.ly/npmvsgulp
* Transpiling
  + TypeScript, Babel and Elm
  + Babel
    - Eperimental Features: Stage-X Plugins – Install appropriate plugins
    - Setup
      * Create .babelrc
      * Insert preset object
* Bundlers
  + Module Formats



* + Bundlers – Browserify, Webpack, Rollup, JSPM
  + Sourcemaps to debug our work
    - Devtool setting in webpack
    - <https://webpack.js.org>
    - Opciones de sourcemap
      * <https://webpack.js.org/configuration/devtool/>
* Linting
  + ESLint
  + Take rules from linter documentation
  + Define warnings vs errors
  + Choose plugins to add extra checks if you are using a framework
    - <https://github.com/dustinspecker/awesome-eslint>
  + OR you can use a preset like AirBnb or StandardJS
  + To watch files with ESLint
    - Eslint-loader, tied to webpack watch
    - Eslint-watch, npm package that wraps a watcher to eslint
  + To lint ES experimental features use babel-eslint
* Unit Tests
  + Framework, Assertion Library, Helper Libraries, Where to run them, Where to place them files, When to run them
  + Mocha
    - Highly Configurable
    - Will couple with Chai as an Assertion Library
    - A reporter is what writes in the console, we will use Progress
  + Jasmine
    - Includes an Assertion Library (a.k.a. the expects) but not that configurable
    - Jest is a wrapper for it – popular for React
  + Helper Libraries
    - JSDOM to run tests like in the browser without the browser – faster
    - Cheerio can query the virtual DOM just like jQuery on a regular DOM
  + Where to run tests
    - 
    - JSDOM, being browserless, runs on NodeJS
  + Where to place the files
    - Centralized or Alongside?
  + When to run the tests
    - On save
  + Not to forget Mocking and Code Coverage reports
* Continuous Integration
  + 
    - Travis CI runs in Linux
      * 
    - Appveyor runs in Windows
      * 
    - Jenkins
  + Just login with GitHub and add project repo to the CI you are using
  + Remember to include your .yml file
* HTTP Calls
  + 
  + Centralize API Calls – Intercept every call so that you can centralize it and do the same thing for every call
* Pollyfill
  + You can polyfill for every browser or
  + You can polyfill only for browsers that need it with polyfill.io
* Mocked Calls
  + 
  + 

Other Javascript Development frameworks:

* Mobile Development
  + <https://www.nativescript.org/>
  + <https://phonegap.com/>
  + <https://facebook.github.io/react-native/>
* <https://nwjs.io/>
* Desktop
  + <https://electronjs.org/>