Enterprise Web Development

Building Modern Web Applications & Services using Node.js

Waterford Institute *of* Technology

INSTITIÚID TEICNEOLAÍOCHTA PHORT LÁIRGE

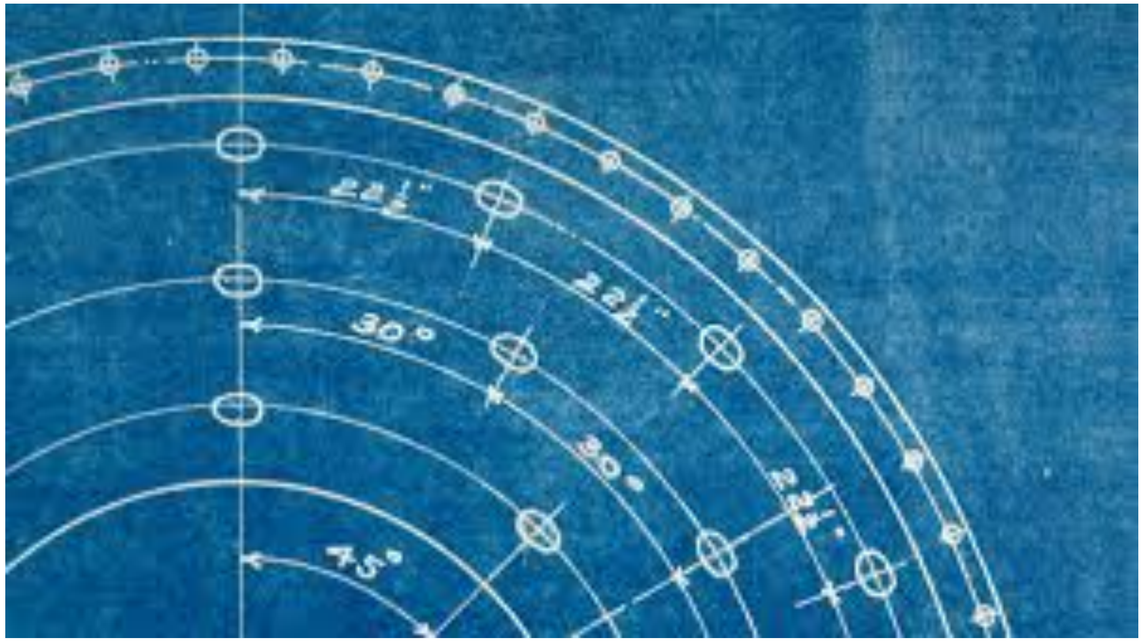
Eamonn de Leastar
edeleastar@wit.ie
Jimmy McGibney
jmcgibney@wit.ie

Course Mission

Transfer a set of foundation skills to enable you to design, build, secure, test and deploy a modern web application + API.

Agenda

- Prerequisites
- Schedule & Assessment
- Module Overview
- Lab Requirements

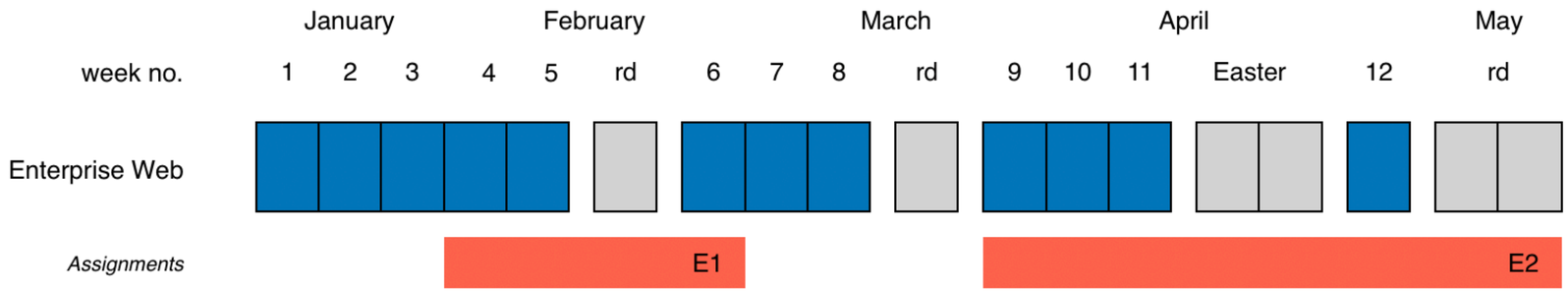


Perquisites

- Intermediate level skills in:
  - HTML: Ability to effectively structure the html content of a small to medium static site, including the use of templates
  - CSS: Understand the fundamentals of CSS, and be able to realise simple layouts and designs
  - Javascript: Be familiar with the building blocks of the language and be able to compose realise algorithms to accomplish simple tasks.

Schedule & Assessment

Semester 1		S	M	T	W
January	Week	6	7	8	9
	1	13	14	15	16
	2	20	21	22	23
	3	27	28	29	30
February	*4*	3	4	5	6
	5	10	11	12	13
	reading-week	17	18	19	20
	6	24	25	26	27
March	*7*	3	4	5	6
	8	10	11	12	13
	reading-week	17	18	19	20
	9	24	25	26	27
April	*10*	31	1	2	3
	11	7	8	9	10
	Easter-break	14	15	16	17
		21	22	23	24
May	12	28	29	30	1



Weekly Topics

01: Javascript Client Review

Javascript in the browser. The DOM, Jqquery & Ajax to access APIs.

02: Introducing Node

The fundmendals of node. Using node as a api client. A first node application.

03: Hapi Applications

HAPI application structure: controllers, views & sessions.

04: MongoDB & Validation

Mongoose Schema, mongo db configuration, Joi validation

05: Seeding & Deployment

Deployment to the cloud. Seeding the database.

06: Security I & APIs

Security threats and attacks. Designing an API.

07: Security II & TDD

Cryptography & encryption. Test Driven Development.

08: Digital Certificates & REST

Digital signatures, Certificates and TLS. Rest principles & patterns.

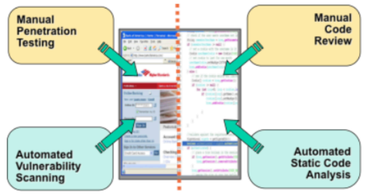
09: Security Threats & API Clients

Web Application Vulnerabilities. Java REST API Clients

10: Web App Authentication & Aurelia I

Authentication & OAuth. Introducing Aurelia

11: Penetration Testing & Aurelia 2



Ethical Hacking. Aurelia core concepts.

12: Aurelia 3

Aurelia Application Architecture

Web Development

- Simple Client Side
- Web Application
- NoSQL Databases
- Rest API
- Test Driven Development
- Front End (SPA)

Security

- Context & Crypto Basics
- Encryption & Digital Certificates
- Threats & Threat Modelling
- Penetration Testing

01: Javascript Client Review

Javascript in the browser. The DOM, Jqquery & Ajax to access APIs.

02: Introducing Node

The fundmendals of node. Using node as a api client. A first node application.

03: Hapi Applications

HAPI application structure: controllers, views & sessions.

04: MongoDB & Validation

Mongoose Schema, mongo db configuration, Joi validation

05: Seeding & Deployment

Deployment to the cloud. Seeding the database.

06: Security I & APIs

Security threats and attacks. Designing an API.

Web Development

- Simple Client Side
- Web Application
- NoSQL Databases
- Rest API

Security

- Context & Crypto Basics

07: Security II & TDD

Cryptography & encryption. Test Driven Development.

08: Digital Certificates & REST

Digital signatures, Certificates and TLS. Rest principles & patterns.

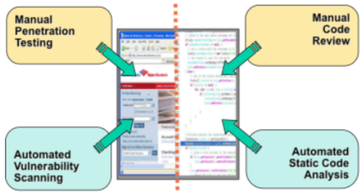
09: Security Threats & Aurelia 1

Web Application Vulnerabilities. Java REST API Clients

10: Web App Authentication & Aurelia 2

Authentication & OAuth. Introducing Aurelia

11: Penetration Testing & Aurelia 2



Ethical Hacking. Aurelia core concepts.

12: Aurelia 3

Aurelia Application Architecture

Web Development

- Test Driven Development
- Front End (SPA)

Security

- Context & Crypto Basics
- Encryption & Digital Certificates
- Threats & Threat Modelling
- Penetration Testing

01: Javascript Client Review

Javascript in the browser. The DOM, Jqquery & Ajax to access APIs.

JS Browser Tools

Simple JavaScript client-side applications. Introducing Chrome Developer tools.

DOM

The DOM: Javascript manipulates the contents of the web page through a standard abstraction called the Document Object Model (DOM), hence the reason for this brief introduction.

JQuery

jQuery: although not as popular as it once may have been, rumours of its death have likely been greatly exaggerated. Undoubtedly losing ground to modern frameworks. it will still probably be in widespread use for quite some time yet.

Lab 01a JQuery

Use both native JavaScript and JQuery to access and manipulate the DOM.

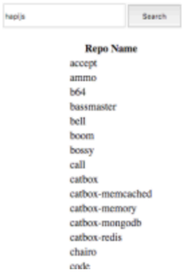
Ajax Introduction

Concluding our introduction to JavaScript with a brief discussion about Ajax.

APIs in Ajax

APIs in Ajax

Lab 01b Github API

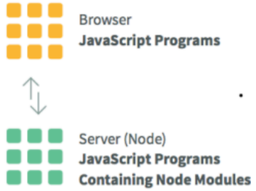


Explore the github API

02: Introducing Node

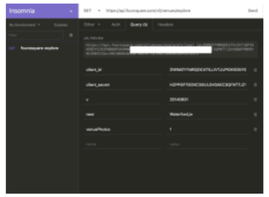
The fundmendals of node. Using node as a api client. A first node application.

Node Essence



What is node? What is its role in modern development? We look at some key characteristics of the platform

APIs in Node

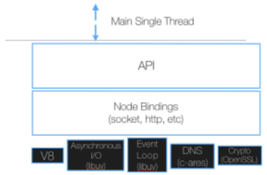


Review the FoureSquare api and the venues end point. Show how the browser, a dedicated rest tool and a node programme can access this API

Lab 02a FoureSquare API

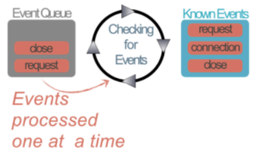
Connect to foursquare API and retrieve list of venues based on a simple location and venue keyboard. Log the venues to the console. Use an npm package to access the api.

Node Context



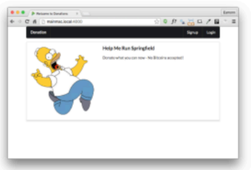
A look at the components of node, its versioning systems, and key advantages

Programming Node



Examine how node is programmed. Explore callback styles, modules and node program structure.

Lab 02b Applications

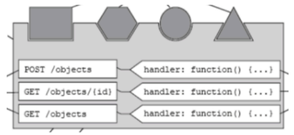


Create a new simple node/HAPI project to serve a simple set of static views. Structure the project into controller, views, public assets and routes. The app offers no user interaction, apart from simple navigation.

03: Hapi Applications

HAPI application structure: controllers, views & sessions.

Hapi Building Blocks



Enumerate the core building blocks of hapi and explain how these are assembled into a simple application.

Static Views

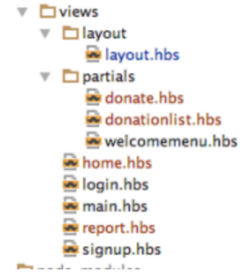
```
<body>
  <section class="ui container">
    <section class="ui stacked segment">
      <div class="ui grid">
        <aside class="six wide column">
          <img src="images/homer.png" class="
        </aside>
        <article class="ten wide column">
          <header class="ui  header"> Help Me
          <p> Donate what you can now - No Bi
        </article>
      </div>
    </section>
  </section>
</body>
</html>
```

Explore how to server simple views using the Hapi Inert plugin

Templates

Extend the static view with more dynamic capabilities with the vision plugin and the handlebars templating engine.

Lab 03a Views



Extend the application to employ layouts, templates, and partials. Include simple interaction features to track users and donations.

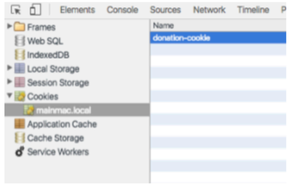
HAPI Philosophy

Introduce Hapi, positioning it within the spectrum for Node Frameworks. Identify Hapis' unique features.

Sessions

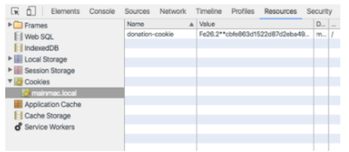
Keeping track of the currently logged in user is a challenge - as HTTP is, by definition 'stateless'. Hidden form fields, url rewriting and cookies are three common techniques for implementing sessions.

Sessions in Hapi



The hapi-auth-cookie provides cookie management for Hapi applications, facilitating creating, read and delete of secure browser based cookies.

Lab 03b Sessions



Incorporate sessions tracking into the app, defining a session strategy, protected and unprotected routes and cookie paramaters.

04: MongoDB & Validation

Mongoose Schema, mongo db configuration, Joi validation

JS Arrays

Brief discussion on arrays and global variables.

Promises in Javascript

then Promises/A+

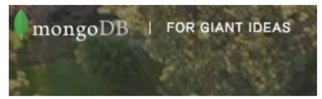
A short introduction to Promises, a new feature is ES6, to simplify asynchronous programmming.

Introducing NoSql DBs

Redis Cassandra CouchDB mongoDB membase riak

A review of the NoSQL movement and some of its key characteristics

Setting Up Mongo

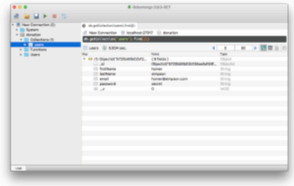


Installing, connecting to and initialising Mongoose/MOngo db from a node application.

Accessing Mongo

Creating and using Objects in Mongo, using its promise based API

Lab 04a Models



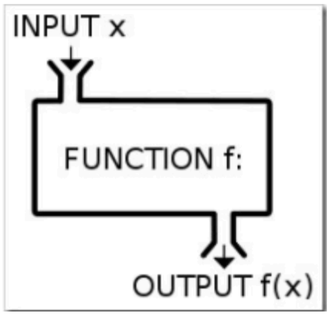
Replace the simple user and donation tracking with a database implementation - Mongo DB. Use the Mongoose ODM to create User + Donation models.

Validation

04: MongoDB & Validation

Mongoose Schema, mongo db configuration, Joi validation

JS Functions



Exploring some of the expressive features of the language, such as functions as first class objects. Brief encounter with some of the newly introduced functionaliy as part of ECMAScript 2015 (ES6).

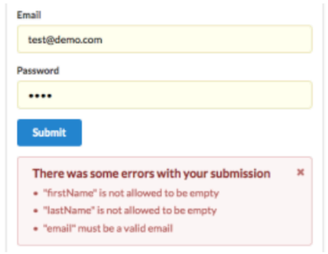
Joi Validation

Joi is a node validation module providing general purpose schema based validation.

Hapi Validation with Joi

Hapi & Joi can work together to deliver easy to configure declarative validation for handlers.

Lab 04b Validation

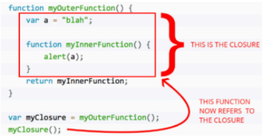


Include an object relationship from donor to user, and render a donors full name on the report view. Incorporate validation into registration view.

05: Seeding & Deployment

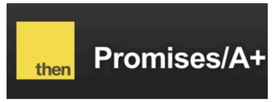
Deployment to the cloud.
Seeding the database.

JS Scope



Examining closure and scope

Callbacks & Promises

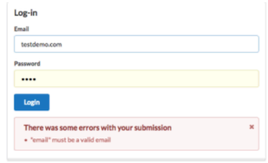


Callbacks are gradually being replaced with Promises in some modules and components. We review callbacks again here and look at how promises approach the same tasks.

Node on Heroku

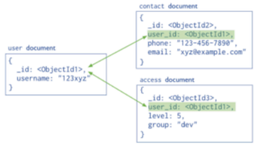
Heroku is a PaaS, supporting node and a variety of other application types. We review here the steps involved in deploying a node app.

Lab 05a Deployment



Deploy the application + the mongo database, to an application server.

Relationships between Mongo Documents

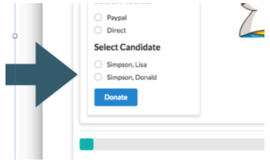


Creating and maintaining relatopnships between mongo documents enable powerful models to be constructed and queried.

Mongoose Seeding

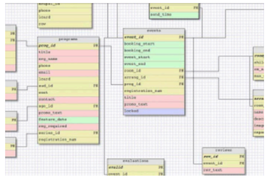
Seeding the database can simplify exploratory development, prepopulating the database with simple test data during development.

Candidate Model



Users should be able to donate to different candidates. We extend the model to include Candidates, incorporating candidate references into donations.

Lab 05b Seeding



Include a mongoose seeder component in the application. Use this to validate a new Candidate model, preloading it with a json specified object graph.

06: Security I & APIs

Security threats and attacks. Designing an API.

Security context

Setting the scene

Crypto basics

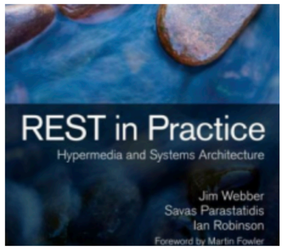
Crash course in cryptography for first lab exercise

Lab 06a GPG

GnuPG

Use a cryptographic application to manage keys and carry out encryption and authentication exercises

APIs



An Application Programmer Interface is the published set of http endpoints and messages that a service can support. API design and implementation is a rich field of study - here we take a general overview.

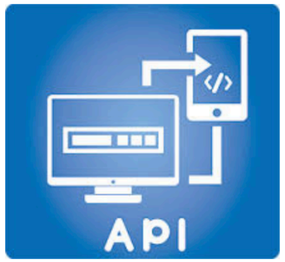
Endpoints

Expose access to the Candidates model as a REST endpoint. This involved defining new routes and handlers, which respond simple JSON representations.

Testing Endpoints

Tools like Postman and Insomnia usefully exercise endpoints. However, we can also exercise them problematically, which offers some significant advantages.

Lab 06b Apis



Start the development of an API for the donation service, focusing initially on providing access to the Candidates model. Implement the API using simple REST principles.

07: Security II & TDD

Cryptography & encryption.
Test Driven Development.

Standard security services

SECURITY SERVICES

Very short talk on standard security services

Encryption

Data encryption using symmetric and public-key techniques

Lab 07a OpenSSL and Certificates

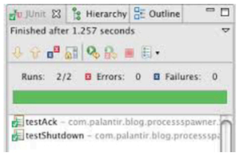
OpenSSL

Use OpenSSL for key generation, encryption, authentication and digital certificate issuance

TDD Introduction

Test Driven Development has been among the most influential approaches in recent software engineering history. Here we look at its origins, principles and some of the important benefits of the approach.

First Tests



The essential elements of TDD are fairly easy to grasp. The support libraries (xUnit) are relatively straightforward, and we can expect our IDE to provide direct assistance to using these libraries. Here we look at the facilities Eclipse provides to the JUnit library.

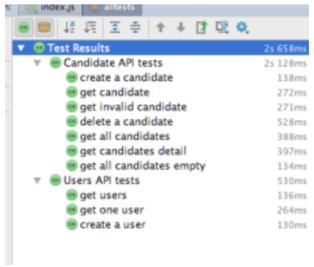
Pragmatic Testing Stack Example

One way of becoming familiar with TDD is to explore some simple examples of various strategies that might be employed in some simple examples. Here we look the evolution of a simple Stack class.

TDD & APIs

Accessing http and the donations api can be encapsulated in purpose built classes. We can then simplify the tests significantly, and expand them to be a more comprehensive exercise of the API.

Lab 07b Tdd



Expand the rudimentary test from the last lab into a more comprehensive suit of tests, exercising the api in depth.

08: Digital Certificates & REST

Digital signatures, Certificates and TLS. Rest principles & patterns.

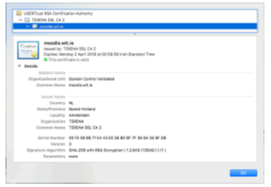
Authentication and digital certificates

Hash functions; message authentication codes; digital certificates

Transport Layer Security (TLS)

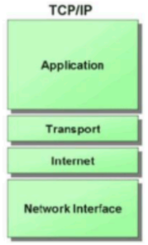
Transport Layer Security (TLS), formerly SSL

Lab 08a TLS Configuration



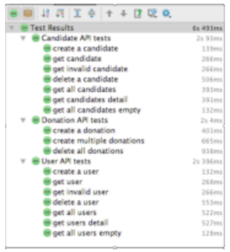
The objective of this lab is to learn about configuring TLS (formerly SSL) on web servers.

HTTP Protocol



HTTP is the heartbeat of the Web, and the key protocol through with both applications and services communicate. A general understanding of it is a key part of a developers knowledge base.

REST Endpoints



Completing the Donation API involves constructing more complex so called 'Restful' endpoints, which serve to introduce references between entities in our model.

Lab 08b Rest

Extend the api to support creating and deleting donations. Donations are associated with candidates, so we utilise the url to establish this relationship.

09: Security Threats & Aurelia 1

Web Application Vulnerabilities.
Java REST API Clients

OWASP Top 10

OWASP Open Web Application Security Project

OWASP's "Ten Most Critical Web Application Security Risks"

Threat Modelling

Security requirements analysis using misuse cases

Security Assignment

This assignment requires you to prepare a report based mainly on your practical experience with web application security.

Lab 09a Misuse Cases

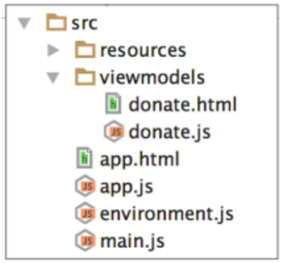
The focus of this lab is this first part of the security assignment, which is broken down into the following steps:

Aurelia Introduction

aurelia.io Next Generation Web Apps

A review of the fundamental features of the Aurelia framework.

Aurelia First Steps



Building a first aurelial app using the aurelia-cli + webstorm. Explore the basics of viewmodels, views + binding.

Lab 10b Aurelia 1

aurelia

Commence an exploration of the Aurelia library by building small app that demonstrates so basic features.

10: Web App Authentication & Aurelia 2

Authentication & OAuth. Introducing Aurelia

Authentication techniques

Authentication techniques

OAuth

Delegated access technique

Lab 10a Authentication techniques

In this lab, we will walk through some examples in class of how to add various authentication-related features to a web application. You may adapt these for the purposes of your web application security assignment report. In particular we will examine:

Aurelia Introduction

PLAY VIDEO

A review of the fundamental features of the Aurelia framework.

Aurelia First Steps

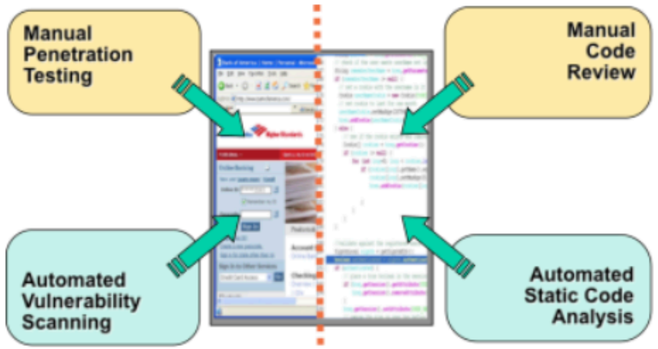
PLAY VIDEO

Building a first aurelial app using the aurelia-cli + webstorm. Explore the basics of viewmodels, views + binding.

Lab 10b Aurelia 1

Commence an exploration of the Aurelia library by building small app that demonstrates so basic features.

11: Penetration Testing & Aurelia 2



Ethical Hacking. Aurelia core concepts.

Vulnerability Testing

Testing for vulnerabilities; attack techniques

Validation & Sanitisation

Cleaning and filtering input

Lab 11a Burp-Proxy

In this lab, we will explore the proxy feature of a popular penetration testing tool.

Lab 11b Regular Expressions

This is just a short demonstration of some basic regular expressions.

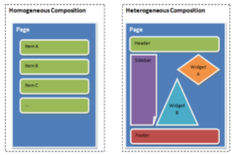
Lab 11c XSS tips

Some tips on demonstrating XSS in a Hapi application

EventAggregator

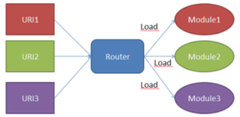
A simple Pub/Sub mechanism for Aurelia

Composition



View-Models can be composed from other view-models, creating a reusable ecosystem of pluggable building blocks.,

Routers

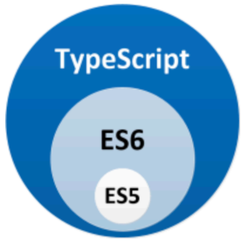


Client side routing in Aurelia

Lab 12a Aurelia 3

Incorporate aurelia routers into donation-client. This will enable a more orderly evolution of the application, allowing new viewmodels to be introduced in a consistent manner.

Lab 12a Aurelia Lab 3 (TS)



Redevelop a version of donation-client in Typescript. This version should match the features in the Aurelia Lab 3 application.

12: Aurelia 4

Aurelia Application Architecture

aurelia-client-http

```
GET JSON with Basic HTTP

import {HttpClient} from 'aurelia-http-client';

let client = new HttpClient();

client.get('package.json')
  .then(data => {
    console.log(data.description)
  });
```

This is a component in the Aurelia framework which delivers http client access. We will need some work on the server for it to be accessed from an aurelia client.

DonationService

The class can now be reimplemented to use the aureli-http-client. Existing data binding relationships should not need to be changed.

Lab 12b Aurelia 4

Rework the donation-service class to communicate with the donation-web application. This will involve incorporating http components into the app and refactoring DonationService to use them.

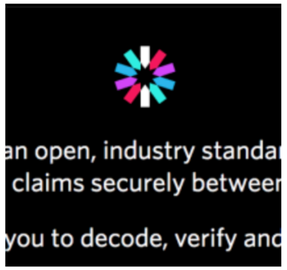
Lab 12b Aurelia 4 (TS)

Rework the donation-service class to communicate with the donation-web application. This will involve incorporating http components into the app and refactoring DonationService to use them.

12: Aurelia 4

Aurelia Application Architecture

JWT

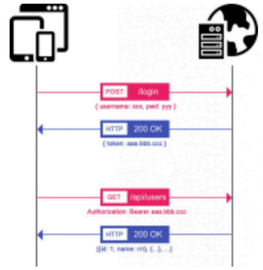


Json Web Tokens is a prominent authentication mechanism securing APIs. A review of its general principles.

Jwt in HAPI

Integrating JWT into an API built using HAPI.

Aurleia JWT



Incorporating JWT authentication into the Aurelia clients

Lab 12c Hapi-JWT

Incorporate JWT security strategy into the application, securing the api routes. Rework the test infrastructure to fully exercises the secure API.

Lab 12d Aurelia 5

Assuming our donation-web app api has been secured with JWT tokens, revised donation-service classes to authenticate against and use these secure routes.

Lab 12d Aurelia 5 (TS)

Assuming our donation-web app api has been secured with JWT tokens, revised donation-service classes to authenticate against and use these secure routes.

Lab Requirements

- WebStorm 2018
- VSCode
- Node.js
- Mongo DB
- Robo 3T
- Heroku-cli
- + additional libraries & tools as needed in labs