Aurelia Routers

Agenda

- The Back Button
- Fragment Identifiers
- Hash Based Routing
- Routing in Aurelia

The Back Button

- For and SPA, the Back button can have the effect of terminating the app
- Back goes to the previous page - but the user may be already screens into the app.
- The previous page may not concur with the users perception

Fragment Identifiers:

- In URIs a hashmark # introduces the optional fragment near the end of the URL
- The fragment identifier functions differently than the rest of the URI:
  - namely, its processing is exclusively client-side with no participation from the web serve
  - Eg:

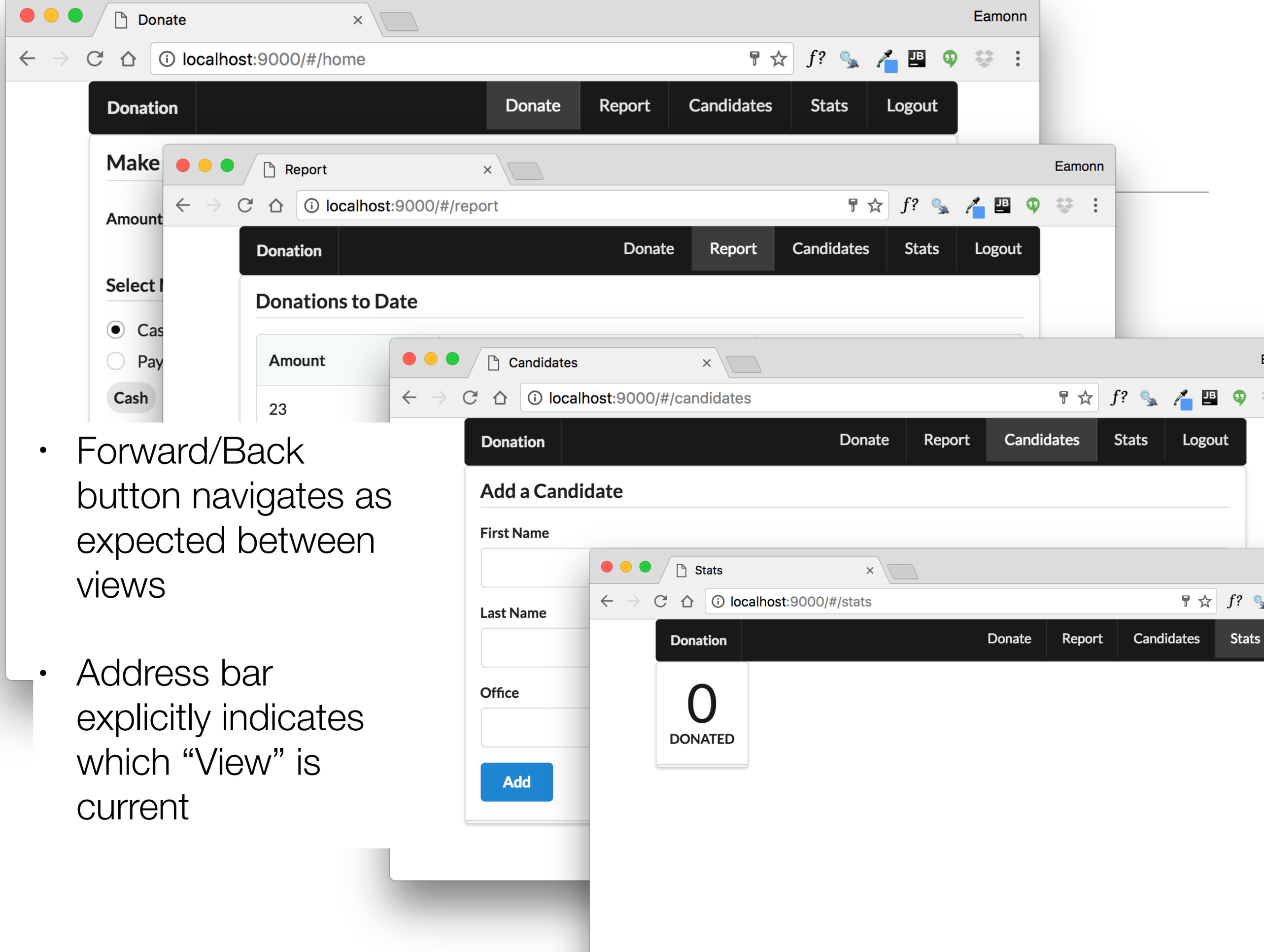
https://en.wikipedia.org/wiki/Fragment_identifier#References

Hash-based routing

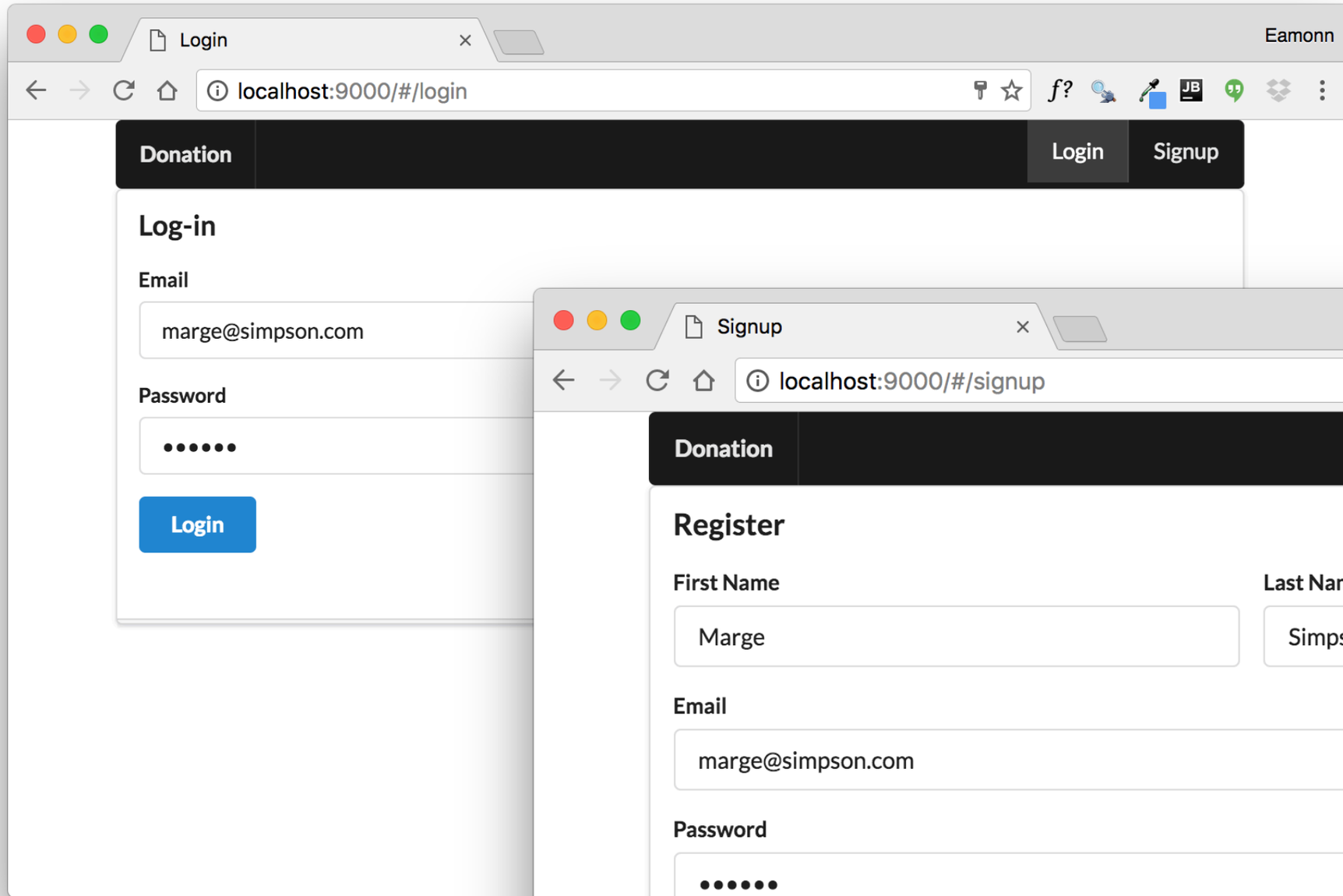
- Use the fragment part of the URL to simulate different content.
  - For example http://site.com/#/products/list leads to displaying a list of products.
  - “#/products/list” is never sent to the server and is completely processed on the client side
- Javascript in the client can collaborate with the browser to maintain the illusion of page navigation
- Requires ‘pushstate’ api, which permits Javascript to explicity manipulate the address bar, specifically back button interventions.

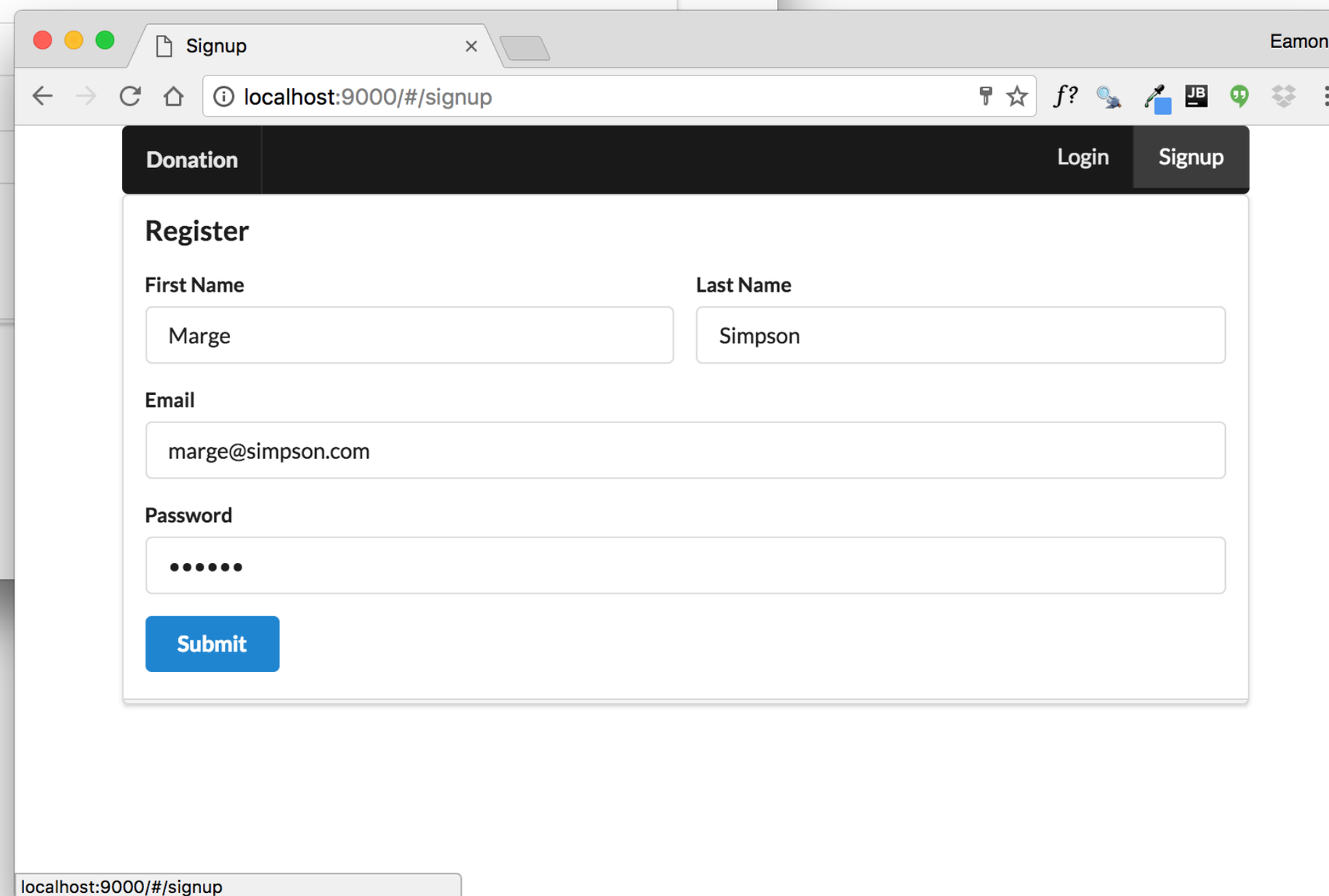
Routing in Aurelia

- Aurelia, link most SPA frameworks, provides a comprehensive client side routing support.
- This mirrors the type of routing familiar from server side development
- As the user navigates a client side app, the routes are inserted into the address bar (after ‘#’ character).
- Back button rolls back to a previous route.



- Forward/Back button navigates as expected between views
- Address bar explicitly indicates which “View” is current



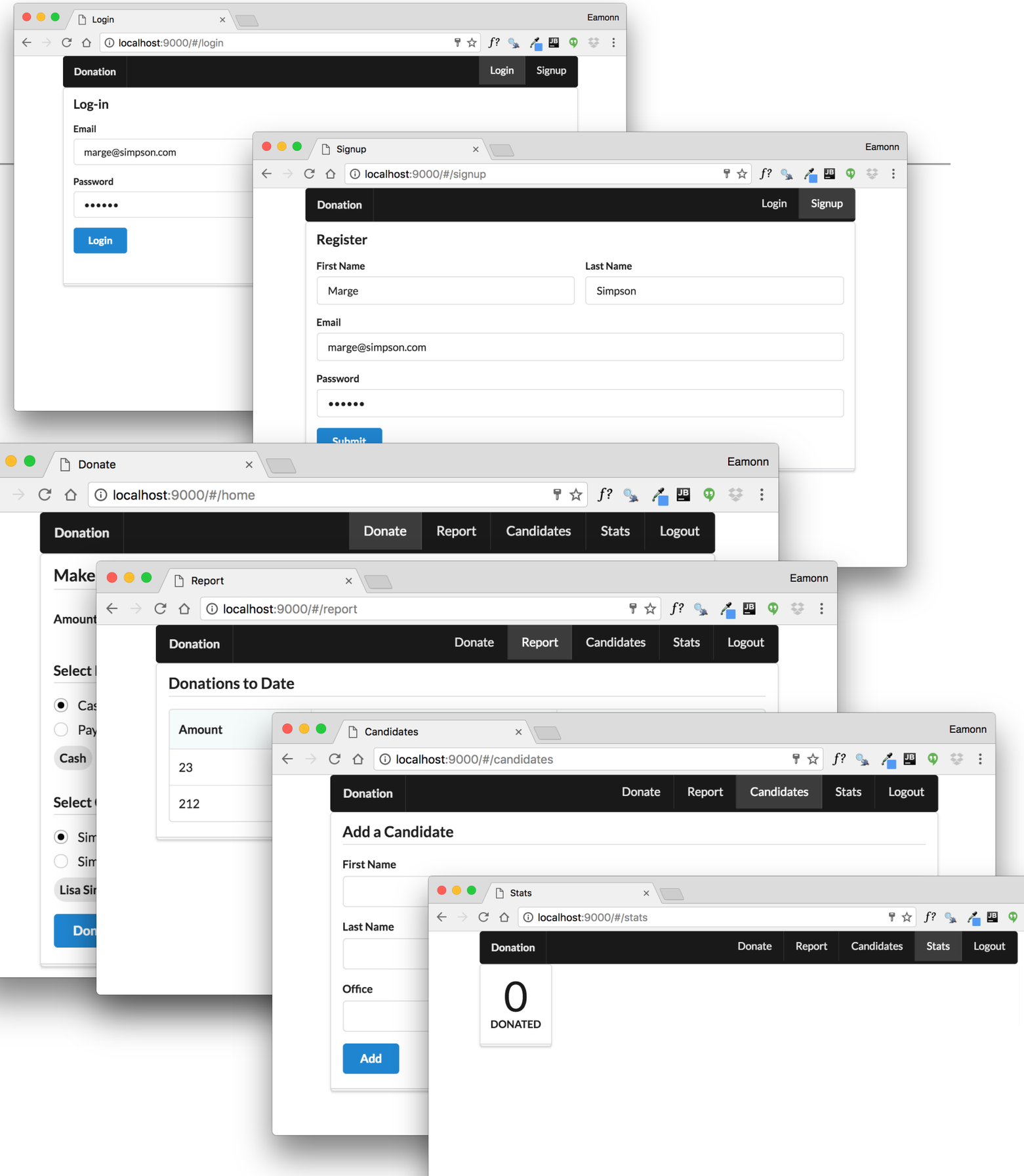


- Login/ signup ‘routes’

Aurelia Supports Multiple Routers

- Router for user prior to authentication

- Router for authenticated user



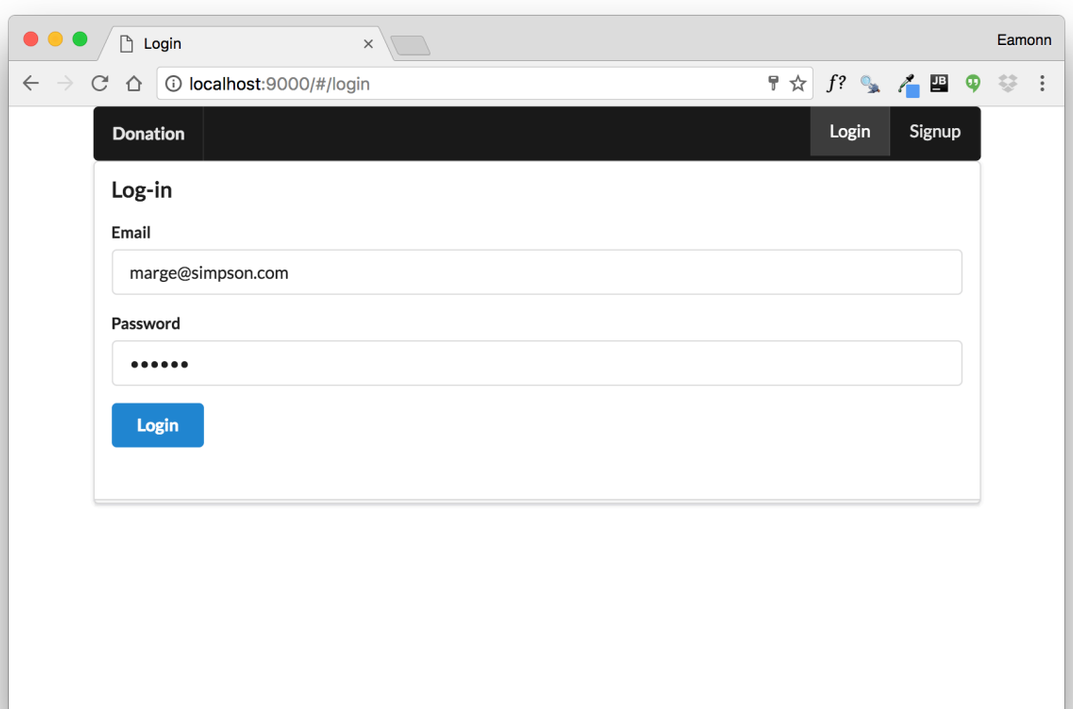
Configuring a Router

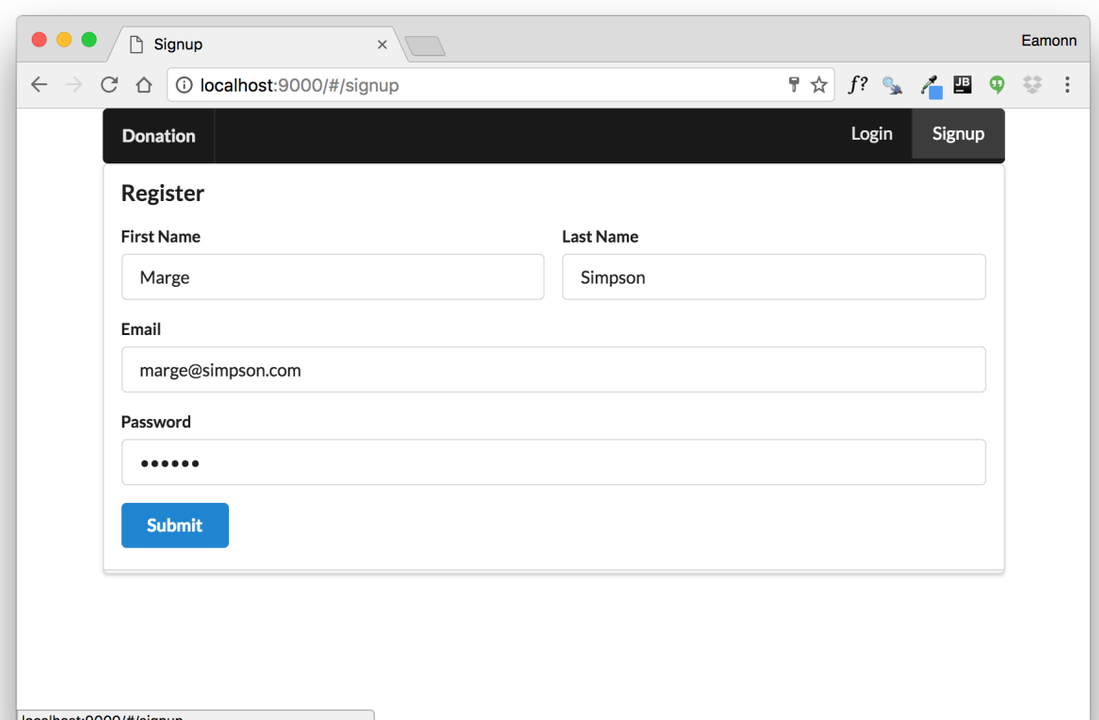
app.html

```
<template>
  <require from="nav-bar.html"></require>
  <div class="ui container">
    <nav-bar router.bind="router"></nav-bar>
    <router-view></router-view>
  </div>
</template>
```

app.js

```
export class App {

  configureRouter(config, router) {
    config.map([
      { route: ['', 'login'], name: 'login',  moduleId: 'viewmodels/login/login',   nav: true, title: 'Login' },
      { route: 'signup',      name: 'signup', moduleId: 'viewmodels/signup/signup', nav: true, title: 'Signup'}
    ]);
    this.router = router;
  }

}
```





nav-bar.html

```
<template bindable="router">
  <nav class="ui inverted menu">
    <header class="header item"><a href="/"> Donation </a></header>
    <div class="right menu">
      <div repeat.for="row of router.navigation">
        <a class="${row.isActive ? 'active' : ''} item"  href.bind="row.href"> ${row.title} </a>
      </div>
    </div>
  </nav>
</template>
```

app.html

```
<template>
  <require from="nav-bar.html"></require>
  <div class="ui container">
    <nav-bar router.bind="router"></nav-bar>
    <router-view></router-view>
  </div>
</template>
```

import the nav-bar

render and bind the nav-bar

placeholder for loaded view-models

app.js

```
export class App {

  configureRouter(config, router) {
    config.map([
      { route: ['', 'login'], name: 'login',  moduleId: 'viewmodels/login/login',   nav: true, title: 'Login' },
      { route: 'signup',      name: 'signup', moduleId: 'viewmodels/signup/signup', nav: true, title: 'Signup'}
    ]);
    this.router = router;
  }

}
```

iterate through the router array

nav-bar.html

```
<template bindable="router">
  <nav class="ui inverted menu">
    <header class="header item"><a href="/"> Donation </a></header>
    <div class="right menu">
      <div repeat.for="row of router.navigation">
        <a class="${row.isActive ? 'active' : ''} item"  href.bind="row.href"> ${row.title} </a>
      </div>
    </div>
  </nav>
</template>
```

render a single menu item in the router array

the #path (fragment)

these are the items - including title, navigable, active etc…

app.js

```
export class App {

  configureRouter(config, router) {
    config.map([
      { route: ['', 'login'], name: 'login',  moduleId: 'viewmodels/login/login',   nav: true, title: 'Login' },
      { route: 'signup',      name: 'signup', moduleId: 'viewmodels/signup/signup', nav: true, title: 'Signup'}
    ]);
    this.router = router;
  }

}
```

Route Attributes

- **route** (required) Either a string or an array of multiple routes (which are strings). If the value supplied is '' then it will be used as the default route. Additional route values can be supplied.

- **name** (required) A unique identifying name for our route. This allows you to reference the route, by generating links or navigating directly to it.

- **moduleId** This is the view-model that you want to be loaded when the route is hit. In the case of our welcome route, it is saying that you want to load src/welcome.js whenever the welcome route is loaded.

- **nav** A boolean value indicating whether or not this route is to be included in the navigation routes array. If this value is true, when iterating over the nav routes array, you'll be able to see it and construct it into a menu. If it is false, then it won't be seen in the navigation routes array.

- **title** The page title value when this route is hit. It will append itself to the provided title value supplied inside of config.title (if supplied).

app.js

```
export class App {

  configureRouter(config, router) {
    config.map([
      { route: ['', 'login'], name: 'login',  moduleId: 'viewmodels/login/login',   nav: true, title: 'Login' },
      { route: 'signup',      name: 'signup', moduleId: 'viewmodels/signup/signup', nav: true, title: 'Signup'}
    ]);
    this.router = router;
  }

}
```

app.js

```
export class App {

  configureRouter(config, router) {
    config.map([
      { route: ['', 'login'], name: 'login',  moduleId: 'viewmodels/login/login',   nav: true, title: 'Login' },
      { route: 'signup',      name: 'signup', moduleId: 'viewmodels/signup/signup', nav: true, title: 'Signup'}
    ]);
    this.router = router;
  }

}
```

login

```
<template>

  <form submit.delegate="login($event)" class="ui stacked segment form">
    <h3 class="ui header">Log-in</h3>
    <div class="field">
      <label>Email</label> <input placeholder="Email" value.bind="email"/>
    </div>
    <div class="field">
      <label>Password</label> <input type="password" value.bind="password"/>
    </div>
    <button class="ui blue submit button">Login</button>
    <h3>${prompt}</h3>
  </form>

</templ
```

```
import {inject} from 'aurelia-framework';
import DonationService from '../../services/donation-service';

@inject(DonationService)
export class Login {

  email = 'marge@simpson.com';
  password = 'secret';

  constructor(ds) {
    this.donationService = ds;
    this.prompt = '';
  }

  login(e) {
    console.log(`Trying to log in ${this.email}`);
    this.donationService.login(this.email, this.password);
  }
}
```

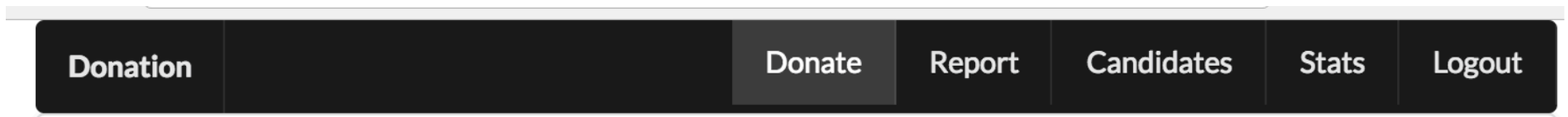
signup

```
<template>
  <form submit.delegate="register($event)" class="ui stacked segment form">
    <h3 class="ui header">Register</h3>
    <div class="two fields">
      <div class="field">
        <label>First Name</label>
        <input placeholder="First Name" type="text" value.bind="firstName">
      </div>
      <div class="field">
        <label>Last Name</label>
```

```
import {inject} from 'aurelia-framework';
import DonationService from '../../services/donation-service';

@inject(DonationService)
export class Signup {

  firstName = 'Marge';
  lastName = 'Simpson';
  email = 'marge@simpson.com';
  password = 'secret';

  constructor(ds) {
    this.donationService = ds;
  }

  register(e) {
    this.showSignup = false;
    this.donationService.register(this.firstName,
                                  this.lastName, this.email, this.password);
    this.donationService.login(this.email, this.password);
  }
}
```

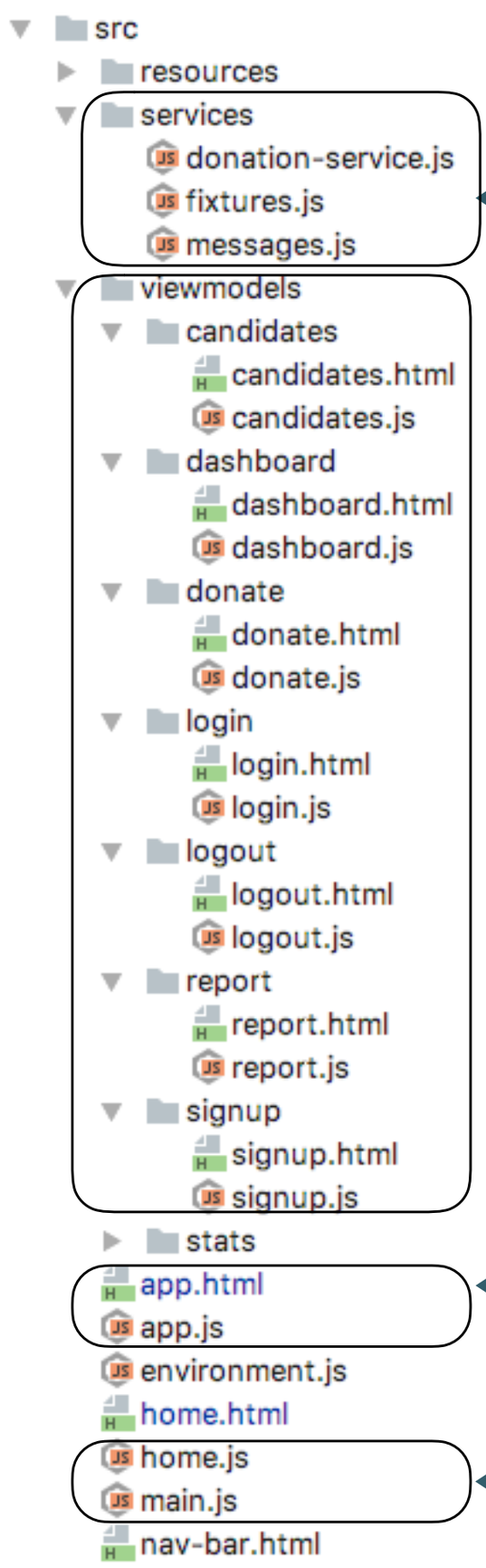
Alternative Router

```
export class Home {

configureRouter(config, router) {
  config.map([
    { route: ['', 'home'], name: 'donate',     moduleId: 'viewmodels/donate/donate', nav: true, title: 'Donate' },
    { route: 'report',     name: 'report',     moduleId: 'viewmodels/report/report', nav: true, title: 'Report' },
    { route: 'candidates', name: 'candidates', moduleId: 'viewmodels/candidates/candidates', nav: true, title: 'Candidates' },
    { route: 'stats',      name: ‘stats',      moduleId: ‘viewmodels/stats/stats',   nav: true, title: 'Stats' },
    { route: 'logout',     name: ‘logout',     moduleId: 'viewmodels/logout/logout', nav: true, title: 'Logout' }
  ]);
  this.router = router;
}
```



- Router for a logged in user.
- Loads different set of view/view-models
- Need seem mechanism to switch routers in response to a successful login

Revised Project Structure



Core Service

ViewModel sets (in subfolders)

Starter Router - **app**

Logged in Router - **home**

Start Router in **app** view/view-model

subscribe to LoginStatus event

app.js

```
@inject(Aurelia, EventAggregator)
export class App {

  constructor(au, ea) {
    ea.subscribe(LoginStatus, msg => {
      if (msg.status.success === true) {
        au.setRoot('home').then(() => {
          this.router.navigateToRoute('donate');
        });
      } else {
        au.setRoot('app').then(() => {
          this.router.navigateToRoute('login');
        });
      }
    });
  }

  configureRouter(config, router) {
    config.map([
      { route: ['', 'login'], name: 'login', moduleId: 'viewmodels/login/login', nav: true, title: 'Login' },
      { route: 'signup', name: 'signup', moduleId: 'viewmodels/signup/signup', nav: true, title: 'Signup' }
    ]);
    this.router = router;
  }
}
```

Login true -> switch to home router

.. and load donate route

.. and load login route

Login false -> switch to app router

Logged in Router in **home** view/view-model

```
export class Home {

configureRouter(config, router) {
  config.map([
    { route: ['', 'home'], name: 'donate',     moduleId: 'viewmodels/donate/donate', nav: true, title: 'Donate' },
    { route: 'report',     name: 'report',     moduleId: 'viewmodels/report/report', nav: true, title: 'Report' },
    { route: 'candidates', name: 'candidates', moduleId: 'viewmodels/candidates/candidates', nav: true, title: 'Candidates' },
    { route: 'stats',      name: ‘stats',      moduleId: ‘viewmodels/stats/stats',   nav: true, title: 'Stats' },
    { route: 'logout',     name: ‘logout',     moduleId: 'viewmodels/logout/logout', nav: true, title: 'Logout' }
  ]);
  this.router = router;
}
```