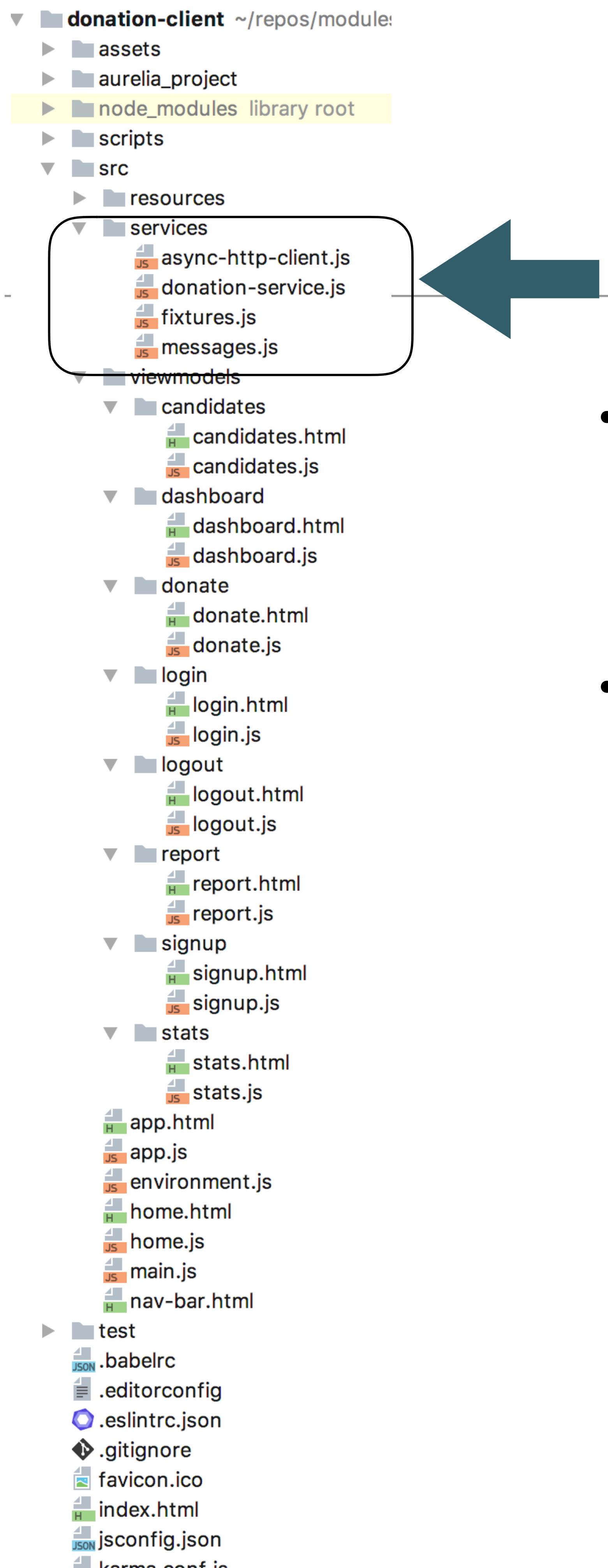
donation-service

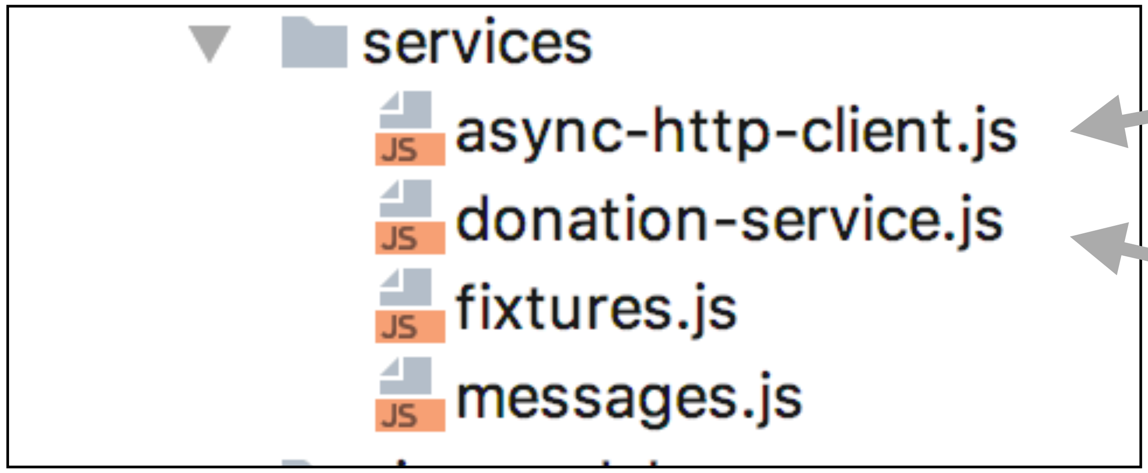
Core Service API Access



```
▼ donation-client ~/repos/module
  ▶ assets
  ▶ aurelia_project
  ▶ node_modules library root
  ▶ scripts
  ▼ src
    ▶ resources
    ▼ services
        async-http-client.js
        donation-service.js
        fixtures.js
        messages.js
    ▼ viewmodels
      ▼ candidates
          candidates.html
          candidates.js
      ▼ dashboard
          dashboard.html
          dashboard.js
      ▼ donate
          donate.html
          donate.js
      ▼ login
          login.html
          login.js
      ▼ logout
          logout.html
          logout.js
      ▼ report
          report.html
          report.js
      ▼ signup
          signup.html
          signup.js
      ▼ stats
          stats.html
          stats.js
      app.html
      app.js
      environment.js
      home.html
      home.js
      main.js
      nav-bar.html
  ▶ test
    .babelrc
    .editorconfig
    .eslintrc.json
    .gitignore
    favicon.ico
    index.html
    jsconfig.json
```

- All access to donation-web centralised in donation-service
- Able to connect to api just be refactoring this class

services


services
async-http-client.js
donation-service.js
fixtures.js
messages.js

Simple wrapper around aurelia-http-client

Refactored to use the async-client

async-http-client

```
export default class Fixtures {
  baseUrl = 'http://localhost:4000';
  methods = ['Cash', 'PayPal'];
}
```

All requests prefixed by fixtures.baseUrl

Pass promises back to callers

```
import {inject} from 'aurelia-framework';
import {HttpClient} from 'aurelia-http-client';
import Fixtures from './fixtures';

@inject(HttpClient, Fixtures)
export default class AsyncHttpClient {

  constructor(httpClient, fixtures) {
    this.http = httpClient;
    this.http.configure(http => {
      http.withBaseUrl(fixtures.baseUrl);
    });
  }

  get(url) {
    return this.http.get(url);
  }

  post(url, obj) {
    return this.http.post(url, obj);
  }

  delete(url) {
    return this.http.delete(url);
  }
}
```

DonationService - 1

- View-Models attached to these arrays
- Retain 3 injected dependencies

```
import {inject} from 'aurelia-framework';
import Fixtures from './fixtures';
import {TotalUpdate, LoginStatus} from './messages';
import {EventAggregator} from 'aurelia-event-aggregator';
import AsyncHttpClient from './async-http-client';

@inject(Fixtures, EventAggregator, AsyncHttpClient)
export default class DonationService {

  donations = [];
  methods = [];
  candidates = [];
  users = [];
  total = 0;

  constructor(data, ea, ac) {
    this.methods = data.methods;
    this.ea = ea;
    this.ac = ac;
    this.getCandidates();
    this.getUsers();
  }
...
```

DonationService - 2

- Fulfilled promises update model objects
- The magic of data binding ensures the UX is updated

```
...

getCandidates() {
  this.ac.get('/api/candidates').then(res => {
    this.candidates = res.content;
  });
}

getUsers() {
  this.ac.get('/api/users').then(res => {
    this.users = res.content;
  });
}

addCandidate(firstName, lastName, office) {
  const candidate = {
    firstName: firstName,
    lastName: lastName,
    office: office
  };
  this.ac.post('/api/candidates', candidate).then(res => {
    this.candidates.push(res.content);
  });
}

...
```

DonationService - 3

```
export class TotalUpdate {
  constructor(total) {
    this.total = total;
  }
}
```

```
...

donate(amount, method, candidate) {
  const donation = {
    amount: amount,
    method: method
  };
  this.ac.post('/api/candidates/' + candidate._id + '/donations', donation).then(res => {
    const returnedDonation = res.content;
    this.donations.push(returnedDonation);
    console.log(amount + ' donated to ' + candidate.firstName + ' ' +
                candidate.lastName + ': ' + method);

    this.total = this.total + parseInt(amount, 10);
    console.log('Total so far ' + this.total);

    this.ea.publish(new TotalUpdate(this.total));
  });
}


...
```

Publish new Total to all interested parties (stats)

```
register(firstName, lastName, email, password) {
  const newUser = {
    firstName: firstName,
    lastName: lastName,
    email: email,
    password: password
  };
  this.ac.post('/api/users', newUser).then(res => {
    this.getUsers();
  });
}

login(email, password) {
  const status = {
    success: false,
    message: 'Login Attempt Failed'
  };
  for (let user of this.users) {
    if (user.email === email && user.password === password) {
      status.success = true;
      status.message = 'logged in';
    }
  }
  this.ea.publish(new LoginStatus(status));
}

logout() {
  const status = {
    success: false,
    message: ''
  };
  this.ea.publish(new LoginStatus(status));
}
```

```
export class LoginStatus {
  constructor(status) {
    this.status = status;
  }
}
```

- These events will trigger router change in app