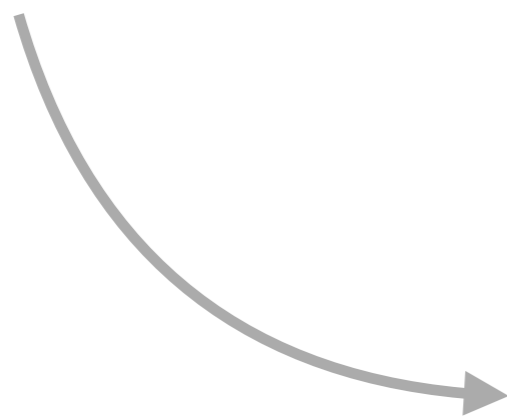


# Front End Full Stack Evolution

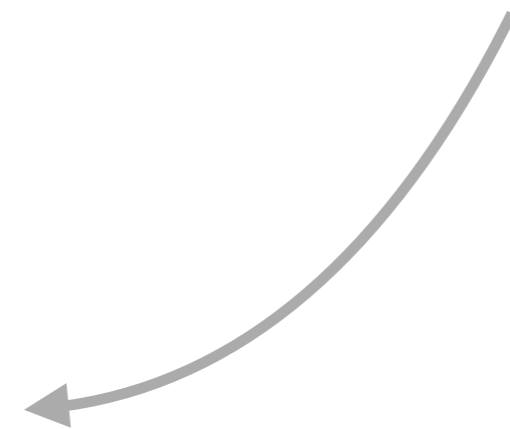
---

Native Applications  
Snappy, interactive UI

Web Pages  
Ease of distribution



Web Applications



1990



GUI

1990

UI Interactive Dynamic

View Logic

Business Logic

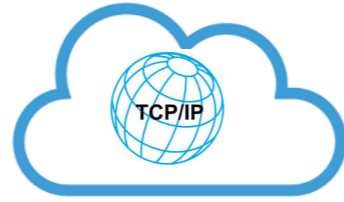
Database

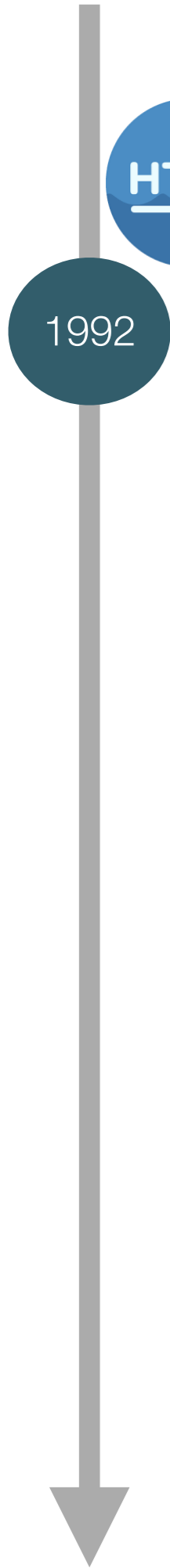
Native GUI  
Desktop  
Applications

1990



1992





HTML

{ CSS }

UI Static



View Logic

Business Logic

Database



First Simple  
Web Apps

1985



1992



1995





UI Interactive



View Logic

Business Logic

Database



html  
css  
js

1995

Web Apps with  
JS Enabled  
Interactive  
Features





1985



1992

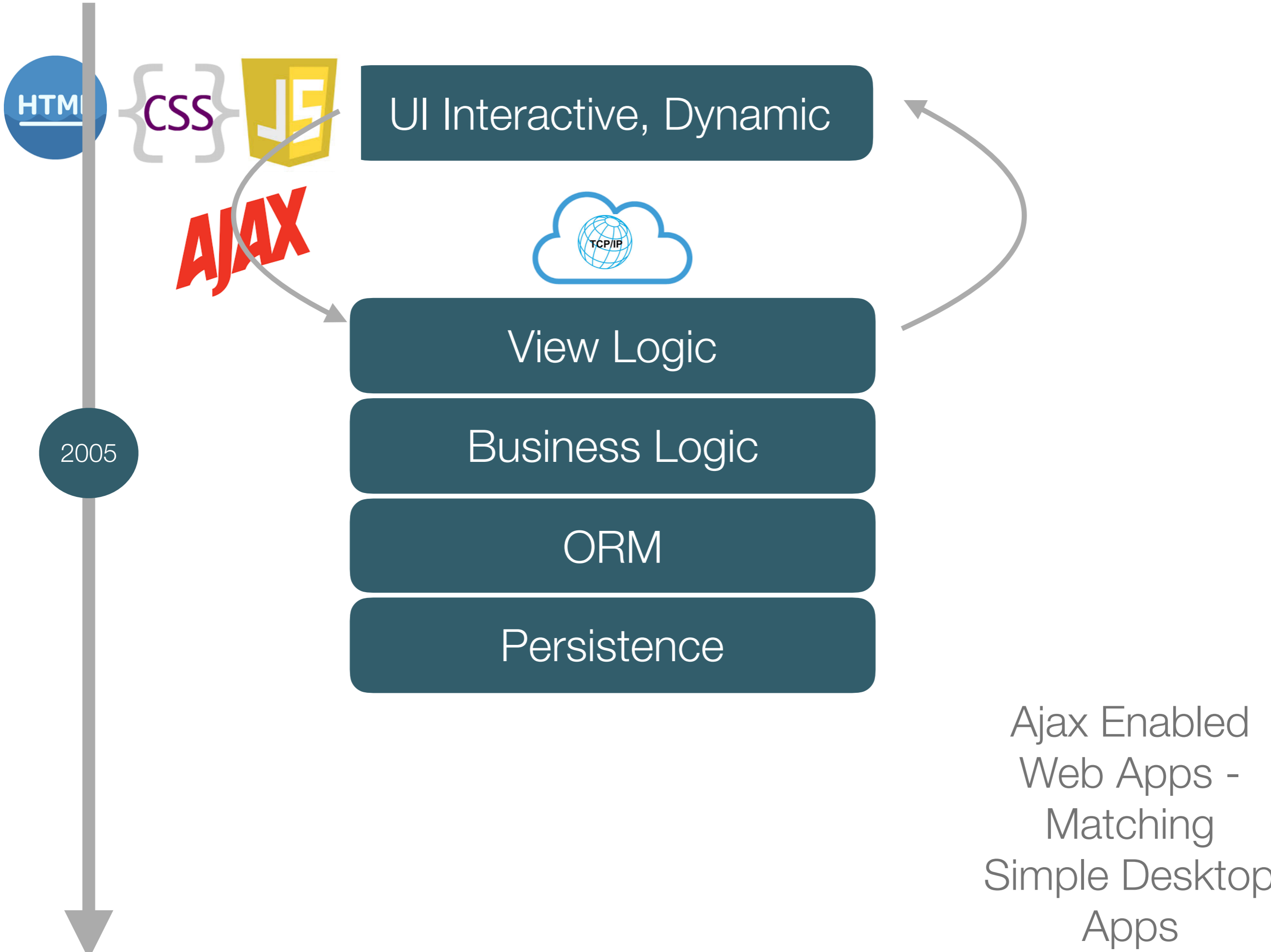


1995



2005





1985



1992



1995



2005



2010





UI Interactive, Dynamic

View Logic

**AJAX**



REST API

Business Logic

ORM

Persistence



{JSON}

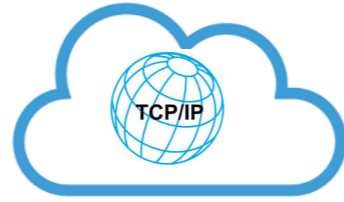
2010

Web Apps fully match sophisticated Desktop Apps

1985



1992



1995



2005



2010



Angular 1



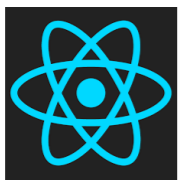
ember



Backbone



2013



React



Angular 2



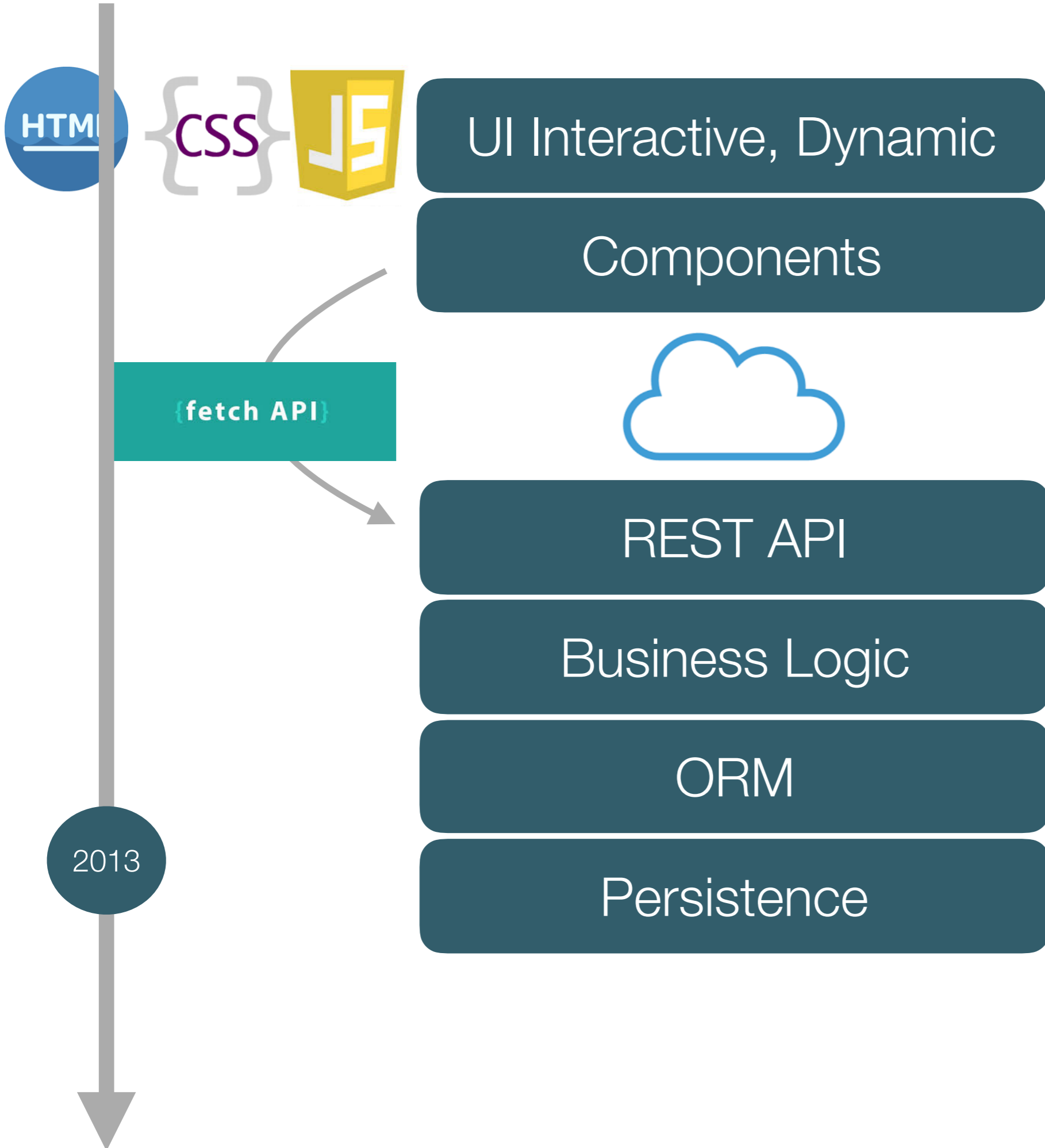
Vue



Aurelia



{fetch API}



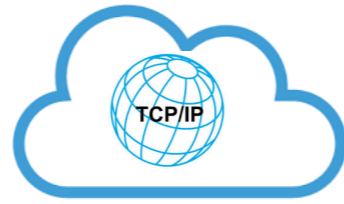
{JSON}

Web Apps based on modular architecture - become extensible and reusable

1985



1992



1995



2005



2010



Angular 1



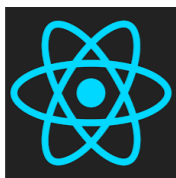
ember



Backbone



2013



React



Angular 2



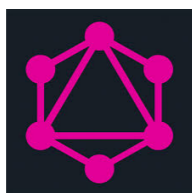
Vue



Aurelia

`fetch API`

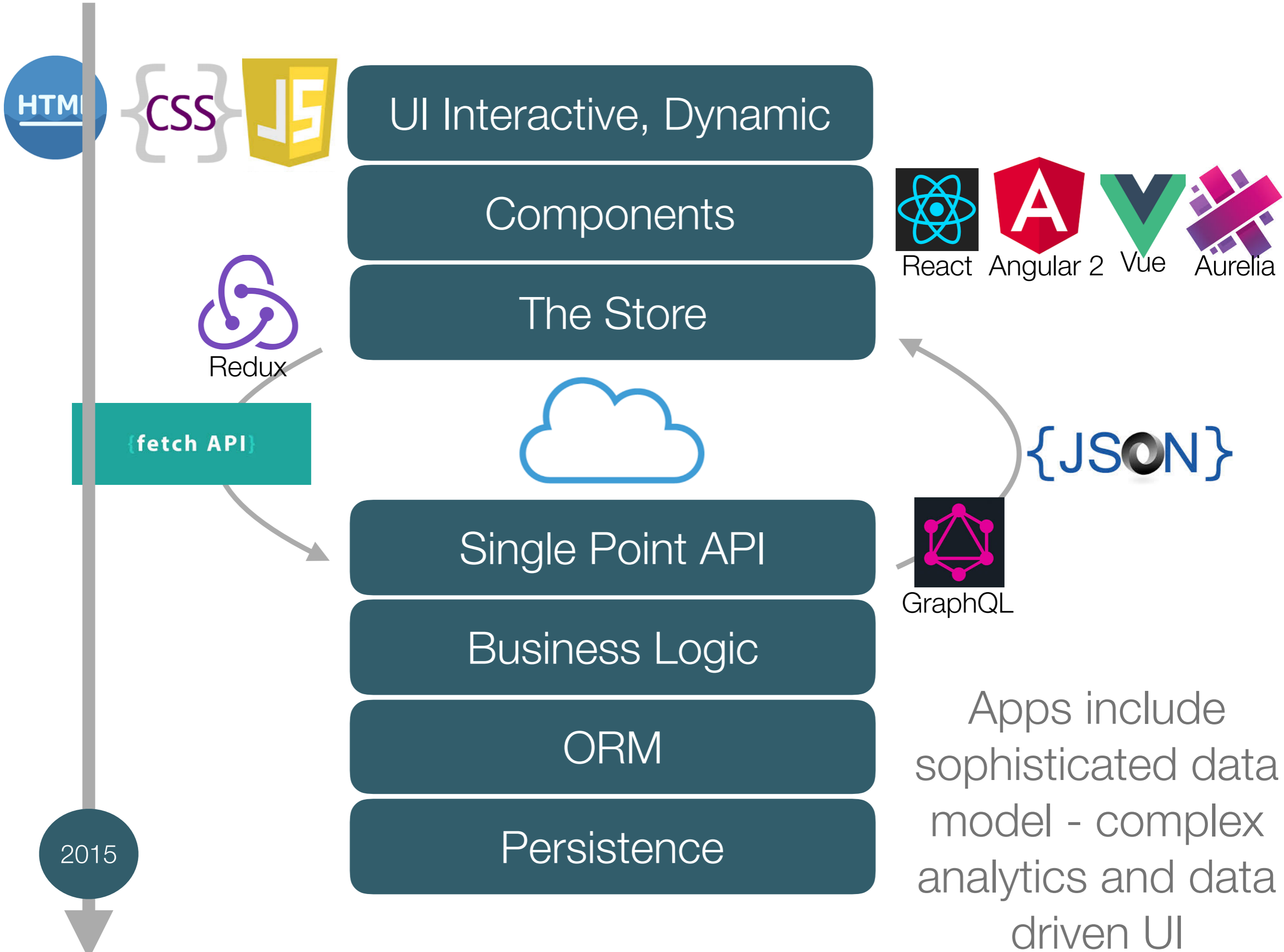
2015



GraphQL



Redux



HTML



UI Interactive, Dynamic

Components

The Store



{fetch API}



Single Point API

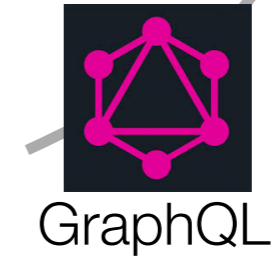
Business Logic

ORM

Persistence



{JSON}



Apps include sophisticated data model - complex analytics and data driven UI

2015



# Donation Versions

---

<https://bitbucket.org/edeleastar/donation-web-final>

<https://bitbucket.org/edeleastar/donation-client>

# donation-web

- Labs 5-11
- Tags:
  - lab.XX.start
  - lab.XX.end
  - lab.XX.exercises

lab.11.exercises totals in report, variable donation amounts + validation in donate view

lab.11.end render candidates in report view

create candidate reference in donation object

introduce and seed candidate model

more extensive set of initial objects

lab.11.start seed single user + donation

lab.10.end production mode db uri

lab.10.start start script for app deployment

lab.9.exercises include validation in login and settings views

lab.9.end turn off abort early to enable multiple errors

engage validation in register form

form error partial

install joi plugin

populate donor object in donation query

lab.9.start incorporate donor object reference into donation

lab.8.exercises fix settings to use database stored user model

lab.8.end display donor email in donations

introduce donation model

introduce user model

lab.8.start connect to mongo via mongoose

lab.7.exercises implementation of settings view

lab.7.end dry - active menu item support

redirect protected routes to login page

cookie retrieval

cookie setting and clearing

default strategy + open accounts routes

lab.7.start cookie plugin initialisation

lab.6.exercises track users and associate with donations

lab.6.end very simple donation persistence

accounts and donations controllers + routes

report and home views + partials

vision layout + handlebar partials

first handlebar expressions

lab.6.start introduce vision hapi plugin + handlebars engine

lab.5.exercises additional routes + static views for login and signup

lab.5.end introduce and serve static assets

revised project structure + first simple view (using inert)

java code standards. airbnb

first route + controller

first simple hapi server

lab.5.start initial empty project

# donation-web

---

- Labs 12-15
- Tags:
  - lab.XX.start
  - lab.XX.end
  - lab.XX.exercises

**origin/master** **origin/HEAD** **master** populate donation with donor and candidate details before returning

include cors headers to support aurelia client app

remove jwt from createUser endpoint + return user details with token when authentication success

remove jwt from getCandidates endpoint

change auth test to use allDonations instead of allCandidates endpoint

**lab.15.exercises** record the donor when a donation is made

**lab.15.end** jwt enabled unit tests

jwt test support classes

install security strategy

secure app api routes (except authenticate)

first test for authentication

authenticate route

**lab.15.start** jwt libraries

**lab.14.exercises** delete donations test

delete donations for a specific candidate

**lab.14.end** donations api tests

preparing to test the donations api

make donation and delete all donation handlers (+auth fix)

retrieve all donations for a candidate

**lab.14.start** retrieve all donations api

**lab.13.exercises** bug fix in userapi - wrong case for create user route

usersapi test + support functions in donation-service

**lab.13.end** comprehensive rewrite of candidatesapitest.js

delete candidate test

support delete methods for candidates and users

fix bug in findOne queries

equals superset using lodash

text fixture + simple test

**lab.13.start** http and donation-service encapsulation classes

**lab.12.exercises** lab.12.exercises

**lab.12.end** create candidate test

create and delete candidate endpoints

second unit test - get single candidate

first mocha test

mocha & chai modules

get candidate endpoint

first simple endpoint - get all candidates

**lab.12.start** boom error reporting module

# donation-client

- Labs - aurelia 1-5

- Tags:

- aurelia.X.start
- aurelia.X.end
- aurelia.X.exercises

<a href="#">aurelia.5.end</a>	<a href="#">origin/master</a>	map unknown routes
		expose isAuthenticated method to clients of DonationService
		recover token from local storage if available
		revise donations-service login/logout to use the authenticate/clearauthenticcte methods
<a href="#">aurelia.5.start</a>		include jwt authentication via authenticate feature
<a href="#">aurelia.4.end</a>		add candidate & register implementations
		donate implementation
		first attempt at accessing donation-service: retrieve candidates + users. Modify login to search returned users list
		remove cached data from fixtures + introduce base url for remote sevice
		simple wrapper for aurelia-http-client
<a href="#">aurelia.4.start</a>		install aurelia-http-client libraries
<a href="#">aurelia.3.exercise.2</a>		dashboard view/viewmodel that integrates other view models
<a href="#">aurelia.3.exercise.1</a>		adjust stats view model to attach to dom
<a href="#">aurelia.3.end</a>	<a href="#">master</a>	<b>13 behind</b> <a href="#">logout view/viewmodel</a>
		introduce routers - one for a logged out status (app) and one for a logged in status
		revised service-donation to use EventAggregator for login/signup + login/signup viewmodels updates
		refactor app viewmodel to use revised viewmodels
		new viewmodels for signup & login
		introduce new LoginStatus message
<a href="#">aurelia.3.start</a>		revise viewmodel folder structure
<a href="#">aurelia.2.exercise.2</a>		signup feature
<a href="#">aurelia.2.exercise.1</a>		login in app uses donation-service login
<a href="#">aurelia.2.end</a>		users fixture + login function in donation-service
		incorporate menu to support logout behaviour
		selectively reveal rows
		display login view in app
		bind to login view in app.js
		new login viewmodelless view
<a href="#">aurelia.2.start</a>		total updated event + new stats view model to display total donated
<a href="#">aurelia.1.exercises</a>		candidates view model, support in donation-service + simplified layouts
<a href="#">aurelia.1.end</a>		using composition, render donate and report on the same view
		display report instead of donate
		report viewmodel + template
		simplify donation-service to use fixtures
		externalise test data into a fixture class
		refactor donate to use the service
		donation-service class to represent api in client
		candidate list support
		include donation method in view/model
		display view model
		introduce view/mode donate
		assets + semantic ui references in index
<a href="#">aurelia.1.start</a>		initial version as generated by au-cli 0.21.0

UI Static



View Logic



handlebars



Business Logic

mongoose

ORM



Persistence

UI Dynamic



Components



{JSON}

View Logic

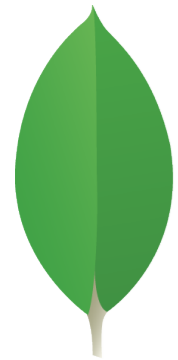
REST API



handlebars



mongoose



MongoDB

Business Logic

ORM

Database

Final donation-web + donation-client

UI Interactive, Dynamic



Components



Aurelia



The Store



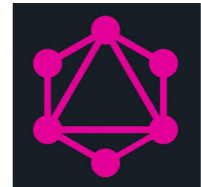
Redux



{JSON}

{fetch API}

Single Point API



GraphQL

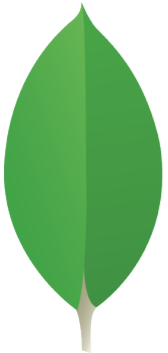
Business Logic

ORM

Persistence



mongoose



MongoDB

*Future* donation-web + donation-client

# donation-web package.json

```
{
  "name": "donation-web",
  "version": "1.0.0",
  "description": "an application to host donations for candidates",
  "main": "index.js",
  "scripts": {
    "start": "node index",
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "author": "",
  "license": "ISC",
  "dependencies": {
    "boom": "^3.2.2",
    "handlebars": "^4.0.5",
    "hapi": "^14.1.0",
    "hapi-auth-cookie": "^6.1.1",
    "hapi-auth-jwt2": "^7.0.1",
    "hapi-cors-headers": "^1.0.0",
    "inert": "^4.0.1",
    "joi": "^9.0.4",
    "jsonwebtoken": "^7.1.9",
    "lodash": "^4.15.0",
    "mongoose": "^4.5.8",
    "mongoose-seeder": "^1.2.1",
    "vision": "^4.1.0"
  },
  "devDependencies": {
    "chai": "^3.5.0",
    "mocha": "^3.0.2",
    "sync-request": "^3.0.1"
  }
}
```



# packages

"hapi"

"handlebars"

"inert"

"vision"

"joi"

"hapi-auth-cookie"

View Logic

REST API

Business Logic

ORM

Database

"boom"

"hapi-auth-jwt2"

"hapi-cors-headers"

"jsonwebtoken"

"mongoose"

"mongoose-seeder"

"lodash"

"chai"

"mocha"

"sync-request"