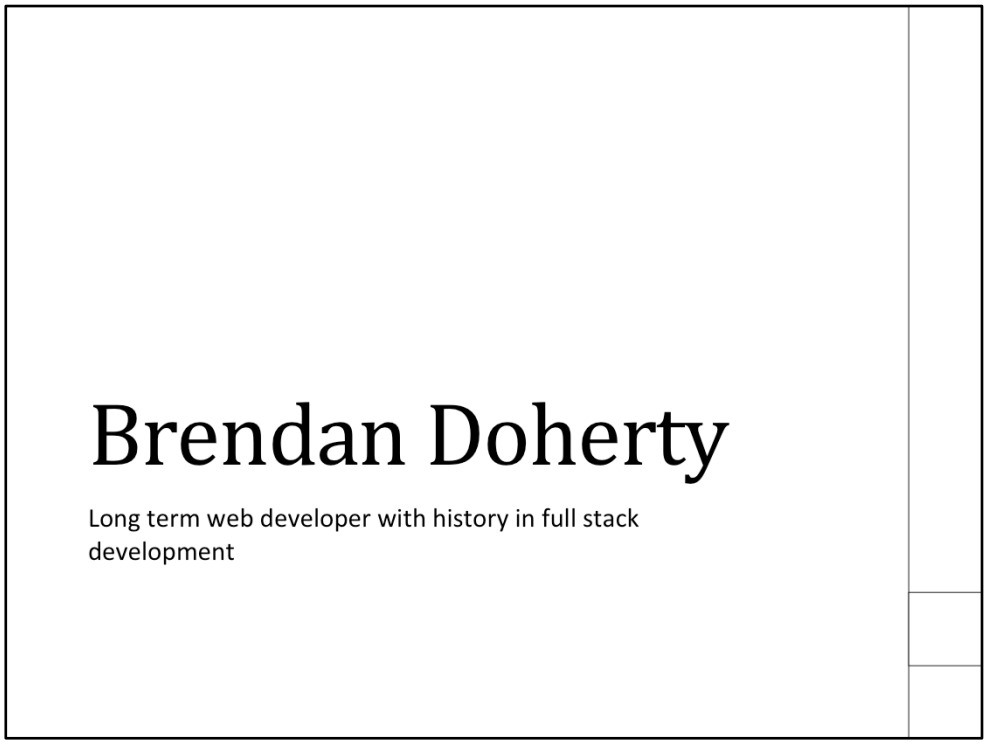


Long term web developer with history in full stack development

And I am wanting to move into a full time JavaScript developer role.

Here to give presentation with a couple of examples of my previous work and why I made it them the way I did

So looking over my previous work, The last piece of work I did where I had a major say in the design and technology used was at the European Commission Economic and Financial Affairs DG, working with the Greek Task force

NEXT

Previous Work

- European Commission Economic and Financial Affairs DG. Greek Task force
- Requirement: Ability to monitor Greek progress in implementing the “bailout” agreement.

At the European Commission we had requirement to build a new application to monitor Greek progress in implementing the “bailout” agreement.

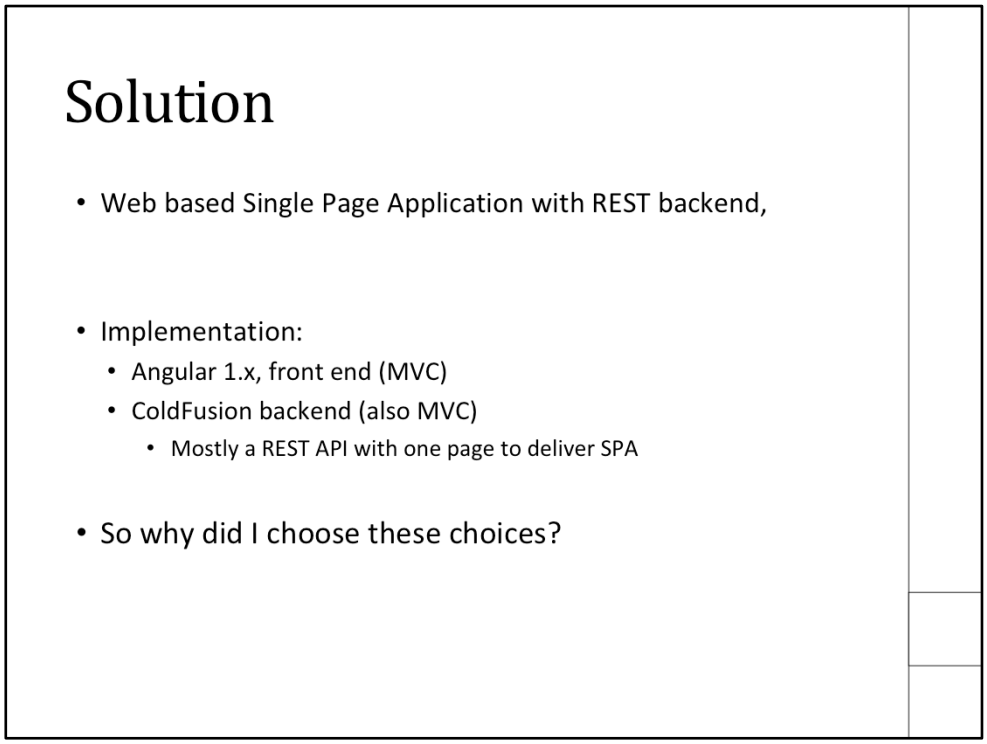
To find a better solution than the numerous excel spreadsheets currently used for the task, which get out of sync quickly and different versions exist at the same time, leading to confusion

Previous applications in the department had been traditional web pages using jQuery. With logic spread between pages, making it difficult to update and change

There was also need to encourage the use of software engineering best practices within the team

So 2 goals, meet the needs of the client, but also to move department forward technically

NEXT



The Solution I came up with was a web based Single Page Application with a REST backend,

This was implemented using:
Angular 1.x, front end (MVC)
ColdFusion backend (also MVC)
Mostly a REST API with one page to deliver the SPA

Why did I choose these choices?:
MVC: tried and tested pattern for graphic user interface. Breaks down problems into component parts

Separating back and front ends, means they can be developed in parallel. Even though in this case I did both

Communication between the front and back ends via REST API, can be documented, well defined, a contract of what the application should do.

Smaller components give higher cohesion and lower coupling, with business logic held in one place

Smaller components also lend themselves to unit testing which gives confidence in

Sudoku Solver

- Why Sudoku?
  - Clearly defined domain model
- Why write it in JavaScript?
  - An example of my work publically available in JavaScript

Why Sudoku:

Has a clearly defined domain to model.

Some aspect of visualization, an area I am interested in

Be accessible to a wide audience, most people have heard of or tried to solve a Sudoku

Sudoku is NP Complete, I just found it interesting, to say if a Sudoku puzzle is valid, you have to solve it, to be classical Sudoku there is only one solution

Non trivial, I didn't want to do a 'to do list' wanted room to grow and to learn and solve design issues

Why JavaScript

To provide an example of my work for others to see, in my language of choice and the language I want to work in

I am looking for a job in JavaScript and its to be expected that people want to see what my code is like

Learn and practice design principles in JavaScript

Design (engine or solver)

- Separate out the Sudoku logic into its own library with an API
- ES6 classes

```
import React, { Component } from 'react';
import {Solver, isValid} from 'sudoku';
```

- Solving techniques should be swappable and new techniques should be able to be added easily.

```
import Hidden from "./strategies/hidden";
import LockedCandidate from "./strategies/lockedCandidate";
import XYChain from "./strategies/XYChain";
import Fish from "./strategies/fish";
```

Separate out the Sudoku logic into its own library with an API,
ES6 classes

A component or object should not know about internal details or other components, which is the Law of Demeter,

A client app doesn't need to know that the solver uses various techniques to find the next deducible number

It just calls next on the Solver object and it will get the next step

Separation of concerns & Single responsibility principle. (so like unix, do one thing, and do it well)

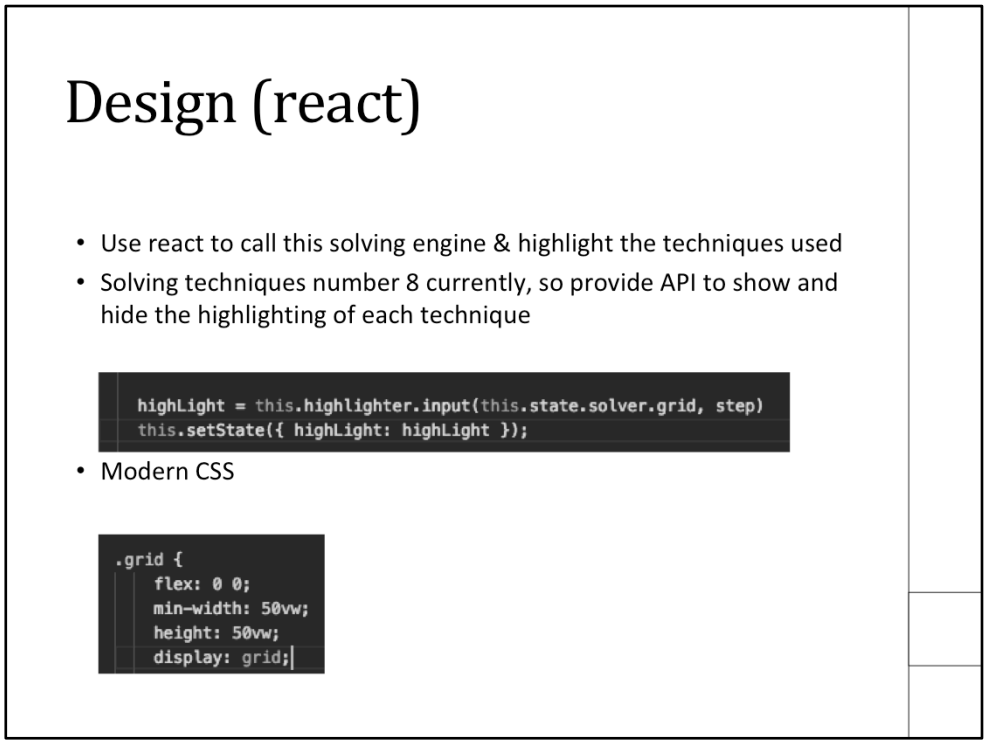
Solver, Grid, isValid deal with different concerns of the overall Sudoku puzzle

Solving techniques should be swappable and new techniques should be able to be added easily.

To put it a different way, they should be Open for extension, closed for modification. This is the Open/Closed Principle

and the Strategy Pattern seemed like a good fit for this, to me

NEXT



Use react to call this solving engine & highlight the techniques used
Solving techniques number 8 currently, so provide API to show and hide the highlighting of each technique

I wanted to pass in the grid and the step that contains details of how the next number was deduced,

and then return what in the ui needs to change to highlight the found number

Similar to strategy pattern in the solving engine in that I want to add new highlighters for each new solving technique added

Again separation of concerns, breaking down components into simpler, more cohesive, less coupled parts

Each highlight is a well defined ui state, so I experimented with a finite state machine to see if it would be of benefit.

Maybe in future versions I will use Redux, but for now I am happy to understand the issues of using react to manage state without another library

Its work in progress so if you look at the code you will see large commented out parts in the react part, as I refactor and rename and become happier with the solution.

The Sudoku engine I am happy with as this is what I worked on first, has unit tests and 100% code coverage

Tools Used

Choosing tools to make JavaScript applications is significant part in the process of building modern JavaScript applications

vscode

Modern IDE for JavaScript, makes developer environment better

Node

Glue that binds modern JavaScript development together

react

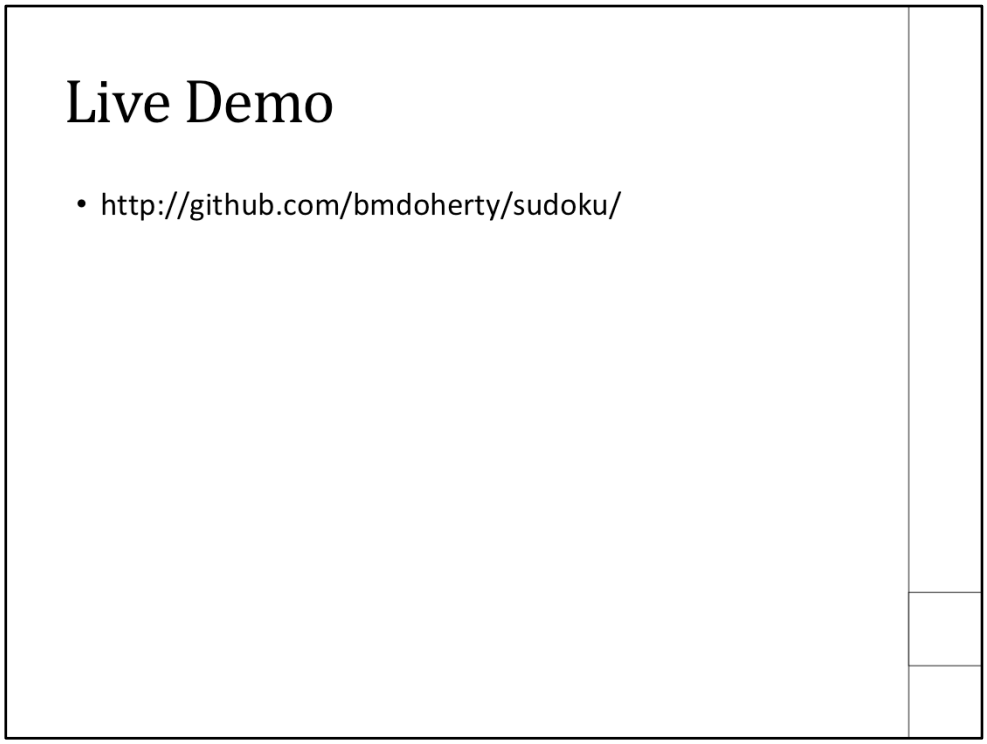
Popular SPA framework

babel-preset-env : Fine grained targeting of browsers ability to write in es7 and deploy to older browsers

```
["last 2 versions", "safari >= 7"]
> 5% (by market share)
```

webpack

Create distribution to be used in browsers /dist. Even though I write the the solver engine in es7

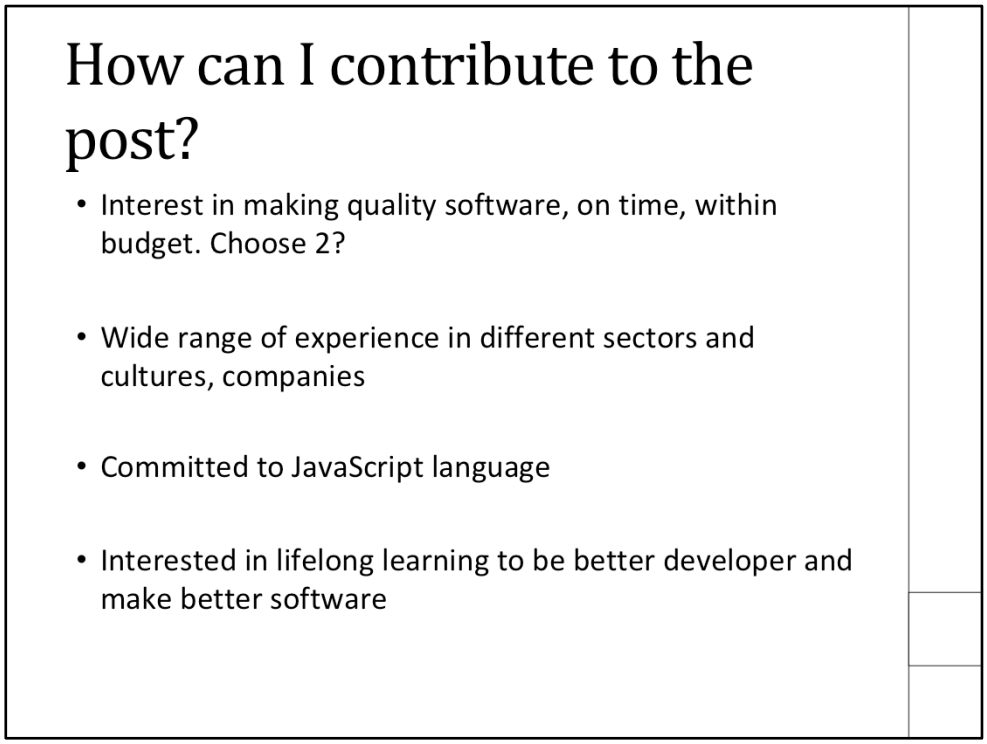


Live Demo
https://github.com/bmdoherty/sudoku

Build artifacts, currently test output, but could just as easily be documentation of the API
https://bmdoherty.github.io/sudoku/

Add todays guardian Sudoku before presentation.

Responsive, looks ok on tablet.
Progressive, works offline, using service workers.



How can I contribute to the post?

Interest in making quality software, on time, within budget. Choose 2?

Where to compromise and where not to as it might be detrimental in the long run.

How to better estimate.

How do developers communicate what they are doing better

interest in the craftsmanship of software development, how do we get better at what we do.

Wide range of experience in different sectors and cultures.

Experience of what works and what doesn't

How teams work together

Committed to JavaScript language

Going to keep learning and getting better in JavaScript until I get a job with it full time.

Interested in lifelong learning to be better developer and make better software

END