ReactJS

Structuring development

ES6

The code in this presentation makes heavy use of *ES6* http://es6-features.org/. If you are not familiar with the syntax please look it up.

- Arrow Functions http://es6-features.org/#ExpressionBodies
- Constants http://es6-features.org/#Constants
- Object.assign http://es6-features.org/#ObjectPropertyAssignment
- Default values for parameters http://es6-features.org/#DefaultParameterValues
- Exporting and importing http://es6-features.org/#ValueExportImport

Or read a full introduction to ES6 features https://github.com/lukehoban/es6features.

An introduction

ReactJS https://facebook.github.io/react/ takes a simple enough approach:

> For a given state describe how to render your application.

Thus we need concepts and tools to compliment ReactJS when we want to build an application.

Components

A *Component* is a description of how to render a part of our application, like a button.

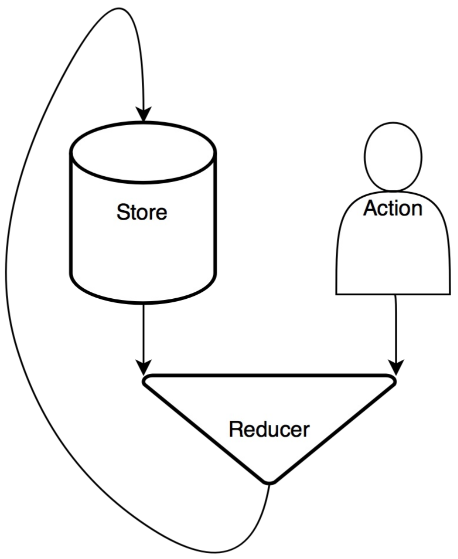
```
//A simple button component.
import React from 'react'

const button = ({disabled, text, click}) => (
    <button onClick={disabled ? () => {} : click} >
        {text}
    </button>
)

export default button
```

Redux

> Redux is a predictable state container for JavaScript apps.

A popular approach to handle this `state` that ReactJS renders is *Redux* http://redux.js.org/.

It takes a unidirectional approach to dataflow. Meaning data only flows in a single direction. This makes our application more predictable.



Visualizing dataflow in Redux

Store

The *Store* is the current representation of the state of your application.

```
//The store is simply one big object in JavaScript.
{
    printing: false,
    orders: [...]
}
```

Actions

You can think of this as an event. While the *Action* is the actual thing being propagated there are also *Actioncreators* which are functions used to create an action.

```
//Use ES6 Syntax to define a function.
export const startPrinting = () => {
    return {
        type: 'PRINTING_START'
    }
}
```

Reducers

Reducers are function that take a current store and return a new one based on an Action.

```
//Return a state for the action or a standard one.
const printing = (state = false, action) => {
    if(action.type === 'PRINTING_START') {
        return true
    } else if(action.type === 'PRINTING_STOP') {
                return false
        } else {
        return state
    }
}
```

Folderstructure

Overview

```
.
├── docs                        All documentation lives here
│   ├── actions                 Redux Action documentation
│   ├── config                  Config to generate docs
│   └── templates               Templates to generate docs
├── node_modules                NPM dependencies
├── package.json
├── src
│   ├── cssPre                  Your CSS preprocessing language of choice
│   ├── img                     Image resources
│   └── js                      JavaScript files
├── test
│   └── reducers                Testing your reducers
└── webpack.config.js           Webpack configuration
```

JS Folderstructure

```
.
├── actions
│   └── index.js                Your Actioncreators
├── components                  Visible components
│   ├── button.js
│   └── orderList.js
├── containers                  Redux containers
│   └── visibleOrderList.js
├── index.js                    The main entry point
└── reducers                    Reducers for each part of the store
    ├── index.js
    ├── ordersReducer.js
    └── printingReducer.js
```

Testing

Debugging

To tackle a problem

Links

Helpful things and further reading.

This is build using:

- Reveal for JS based slides
- Reveal-md for prototyping
- nodetree https://www.npmjs.com/package/nodetree for nice filetrees