Development Primer

- JavaScript Basics
- React Basics
- Mobile Basics

JS Basics

- What is JS?
- Callbacks
- Async/Await
- Higher Order Funcs
- JSON
- Var/Let/Const

What is JS?

- Event driven, functional, imperative
- High-level
- Dynamic-typing
- Object-oriented
- Extremely quick
- De Facto standard for Web dev
- Frameworks
  - Node.js
  - Angular
  - React / React Native
  - Electron,
  - ...

Callbacks

- Send a function as an argument and call from function

```
// Declare
const add10 = (val, callback) => {
  console.log(val); // 100
  callback(val + 10);
}

// Usage
add10(100, value => {
    console.log(value); // 110
});
```

Async / Await

- Used for asynchronous functionality
- Waiting for HTTP request, component to load, etc.

```
function resolveAfter2Seconds() {
  return new Promise(resolve => {
    setTimeout(() => {
      resolve('resolved');
    }, 2000);
  });
}

async function asyncCall() {
  console.log('calling');
  const result = await resolveAfter2Seconds();
  console.log(result);
  // expected output: "resolved"
}

asyncCall();
```

Higher Order Funcs

- Functional programming
- Instead of for loops, recreating arrays, finding elements, etc.
- .forEach(), .map(), .filter(), .reduce(), and .find()

```
// Add role to every element in an array
let arr = [{name: ‘Hayden’}, {name: ‘Gregg’}];

// Traditional:

for (let i = 0; i < arr.length; i++) {
  arr[i] = { ...arr[i], role: “SWE” };
}

// .map
arr = arr.map(o => { ...o, role: “SWE” }
```

JSON

- JavaScript Object Notation
- Object based data structure
- Standard for HTTP requests
- Can be converted to string and back with JSON.parse and JSON.stringify

```
let obj = {
  Name: “Hayden”,
  Age: 20,
  School: “Stevens”,
  Interests: [ “Running”, “Reading” ]
  Courses: {
    SSW423: “Senior Design”,
    IDE400: “Senior Innovation”
  }
}
```

Var / Let / Const

- Used to declare variables
- Var
  - Bad
- Let
  - Mutable variable declaration
  - Block defined
- Const
  - Immutable - constant

```
let greeting = "say Hi";
let times = 4;

if (times > 3) {
  let hello = "say Hello instead";
  console.log(hello); // "say Hello instead"
}
console.log(hello) // hello is not defined
```

React Basics

- What is React?
- Core Components
- Components
- Styling
- Hooks
- Modules

What is React?

- Declarative Style
- Component-Based
- External Plugins
- JavaScript Based
- Lifecycle Methods
- NPM
- Web but also Mobile

Core Components

- Similar to HTML
- Components have names, attributes, and children
- Just like div, span, p, etc.
- Basic tags include:
  - Button -> TouchableOpacity
  - FlatList
  - Image
  - ImageBackground
  - Text
  - View
  - TextInput,
  - ...

```
import React from 'react';
import { Text, View } from
'react-native';

const HelloWorldApp = () => {
  return (
    <View
      style={{
        flex: 1,
        justifyContent: "center",
        alignItems: "center"
      }}>
      <Text>Hello, world!</Text>
    </View>
  );
}

export default HelloWorldApp;
```

Components

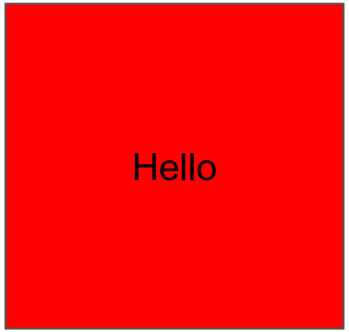
- Can create own components
- Functional vs Class
  - Func is easier to manage / more standard now
- Pass through props as args
- Reusable -> build once use throughout

```
import React from 'react';
import { Text } from 'react-native';

const HelloWorldText = props => {
  return (
    <Text>{props.text}</Text>
  );
}

const HelloWorldApp = () => (
  return (
    <View>
      <HelloWorldText text={“Hi”} />
    </View>
  );
)

export default HelloWorldApp;
```

Styling

- Style components like CSS but using objects
- Camel-cased styles
- Want to centralize and reuse styles
- Can be programmatic since objects
  - Use variables and built in conditionals
- Global colorScheme object

```
Style = {
  backgroundColor: “red”,
  height: 100,
  width: 100,
  alignItems: “center”,
  justifyContent: “center”
}
...
<View style={Style}>
  <Text>Hello</Text>
</View>
...
```



Hooks

- Ways of managing state
- Store “global” variables that can update multiple components
- useState stores value
- useEffect listens to values
  - Similar to observer design pattern
- Lifecycle management
  - Similar to class based state and lifecycle methods
- Can create own hooks with hooks
- Facebook Intro

```
const [free, setFree] =
useState(false);

const isFree = updatedItem => {
  setFree(updatedItem.price === 0);
}

let item = {
  price: 10,
  Name: “Lego …”
}

useEffect(isFree, [item]);

// do a bunch with the item
return <View>
  {free ? ‘Free’ : item.price}
</View>;
```

Modules

- NPM - Node Package Manager
- Open-sourced functionality and components
- Ex:
  - Redux
  - Bootstrap 4
  - MaterialUI
  - Mongoose
- Reuse other people's code
- Command line tool
  - Ex: npm install redux
- Stored in node_modules and managed with package.json

Mobile Basics

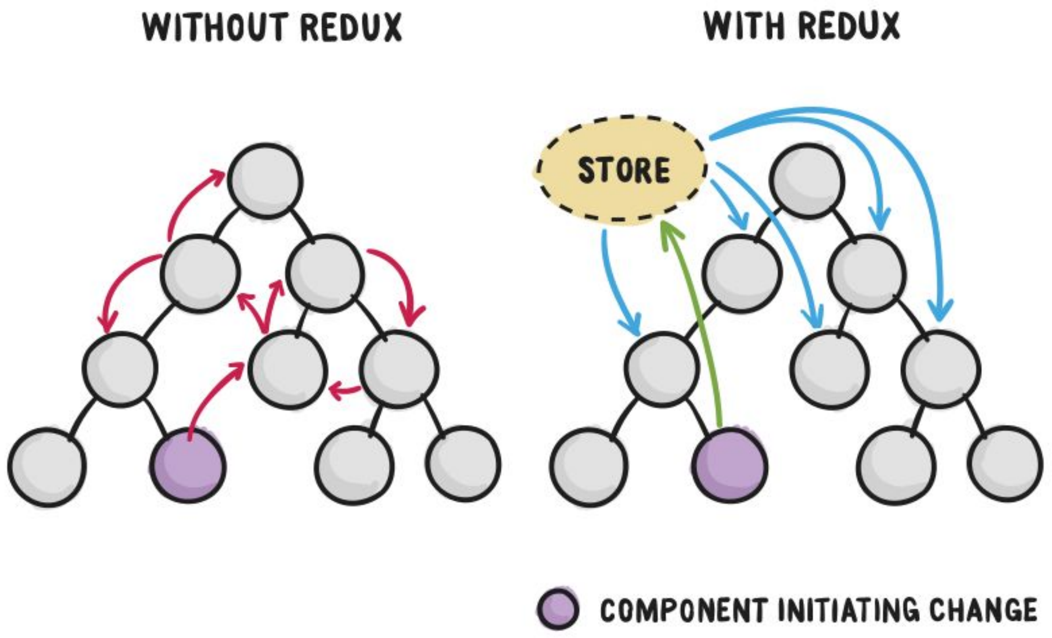
- React Native / Expo
- MobX / Redux
- React Navigation
- Firebase
- Random

React Native/Expo

- React but compiles to Native iOS/Android
- Expo provides framework to develop and run RN code
- Provides structure where Pods/Libraries aren’t necessary
    - Individual development per operating system typically required
- Makes deployment easier
- Bunch of built in modules to interact directly with OS

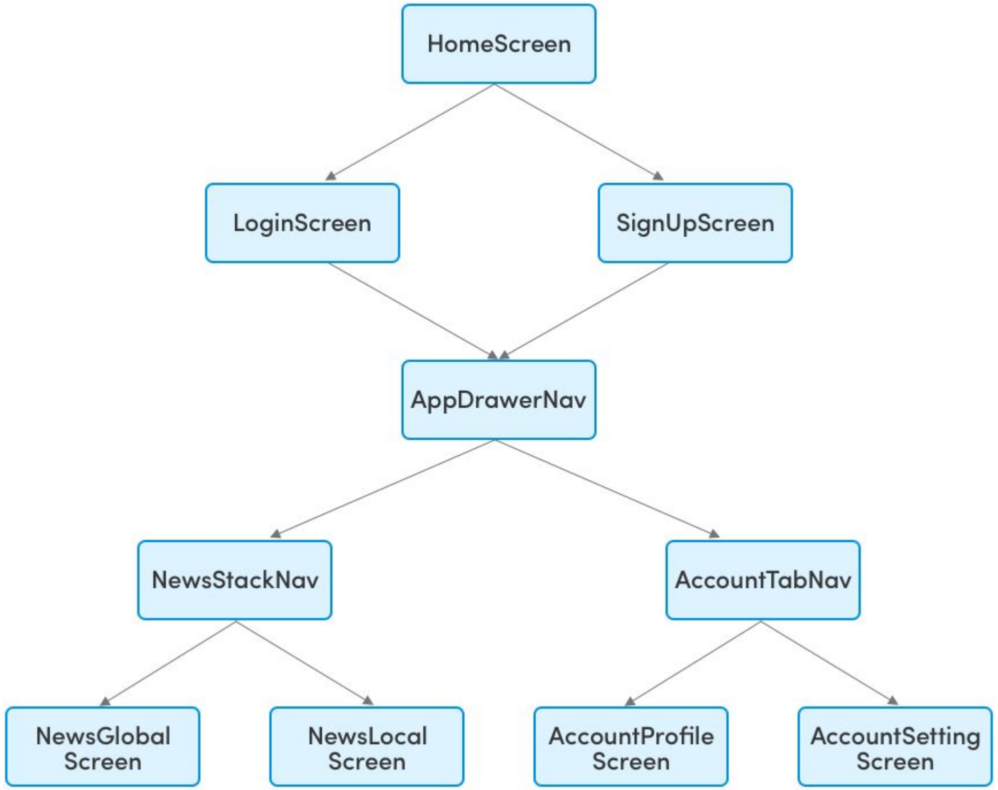
MobX / Redux

- Manage global state across application.
- Passes “props” to rest of components
- Ex: make user data accessible throughout entire app
- Ex2: color scheme



React Navigation

- Manage component hierarchy
- Built in functionality to shift views
- Drawer, Stack, Tab
- Can pass props
- navigation.navigate(routeName)



Firebase

- BaaS (Backend as a Service)
- Database
  - Cloud Firestore
  - Realtime
  - File Storage
- Authentication
- User Management
- Cloud Functions
- Access to GCP infra

Random

- asyncStorage
- Mixpanel
- Twilio
- Notifications
- Webhooks
- SVGs
- ESLint
- Code splitting
- KanBan