

CS47: Cross-Platform Mobile Development

Lecture 1B: Setting the Stage for Making Legit React Native Apps ➔

James Landay
Abdallah AbuHashem
Claire Rosenfeld
Ryan Chen

A dynamic fireworks display against a dark sky, featuring multiple overlapping bursts of various colors including red, orange, yellow, green, blue, and purple. The fireworks range from small sparks to large, multi-layered bursts.

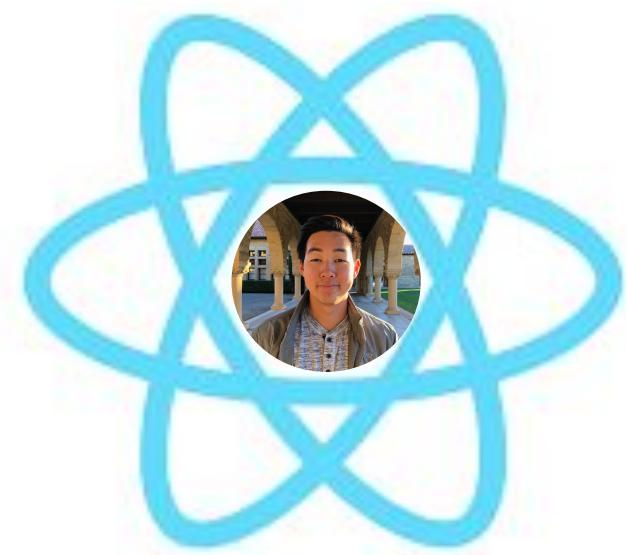
Welcome back
to CS 47!

Course logistics + words of encouragement

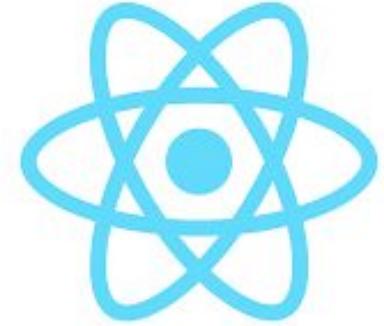
- Enrollment decisions will be made this weekend. The class is capped at 50 students. Students taking CS 147 have priority.
 - Auditing is an **excellent** option!
- Check the syllabus on [the website](#) for what topics we will cover.
- You will get a ton of practice with JavaScript throughout the quarter.
- OH will always be available for extra help.
- We are confident that **everyone** will be able to make great mobile apps by the end of the course! We are here to support everyone along the way.
- We take feedback seriously – please continue providing feedback on exit tickets and sending us emails!

Last lecture

- Ran through logistics
- Introduced React Native
- Dived into JavaScript
- Assigned Assignment 1
 - Environment setup
 - Due next Tuesday January 19th at 11:59 PM ([macOS](#), [Windows](#))



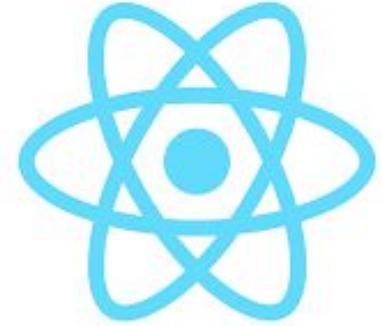
Reviewing what we learned last lecture



- Why React Native?
- Discuss in breakout rooms. Be ready to share your thoughts.

Reviewing what we learned last lecture

- Why React Native?



Today's goals

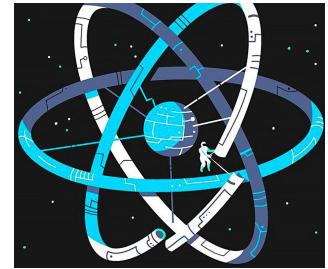
- Discuss the relationships between JavaScript, JSX, React, and React Native
- Learn more about Javascript
- Introduce Expo tools
- Build a React Native app



What are we working with?

- **JavaScript** is a programming language
- **JSX** is an extension of JavaScript that we use to describe what UIs should look like
- **UIs** are what people interact with on an app (text, images, buttons, etc.)
- **React** is a library for building UIs
- **React Native** is framework on top of React

```
let x = 1;  
if (x === 1) {  
  let x = 2;  
}  
  
const displayX = (  
  const x = 42;  
  <View>  
    <Text>{x}</Text>  
  </View>  
)
```



Adding to our JavaScript knowledge

- Asynchrony
- Promises
- Fetch()

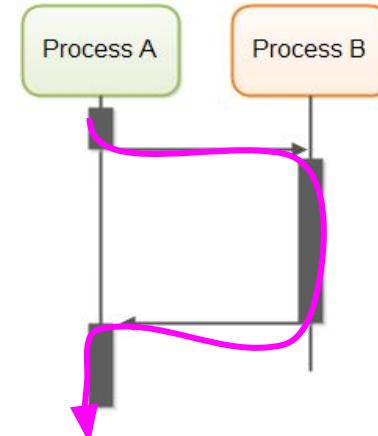
Don't worry about memorizing everything. Let these concepts marinate. They will come into play later in the course. In general, they are important programming concepts to be familiar with.



Understanding JavaScript

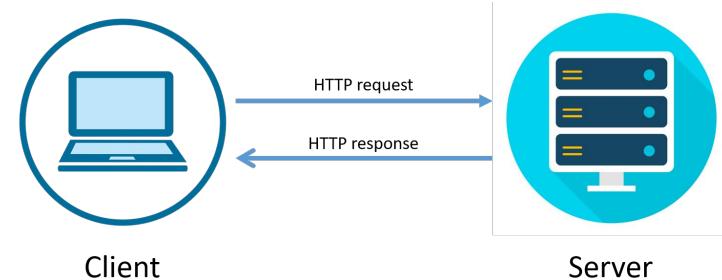
- JavaScript is **single threaded**
 - Only 1 operation can happen at a time (on 1 main thread)
 - Everything else is blocked until that 1 main thread completes its current operation

Sometimes this isn't ideal. Why?



Example: HTTP Requests

- HTTP = Hypertext Transfer Protocol
- Structure (“protocol”) of requests and responses sent over the internet



We're building an app to help people organize what movies they want to watch. We want a list of movies. We found the perfect resource to fetch movie data. We can fetch that data using HTTP requests.

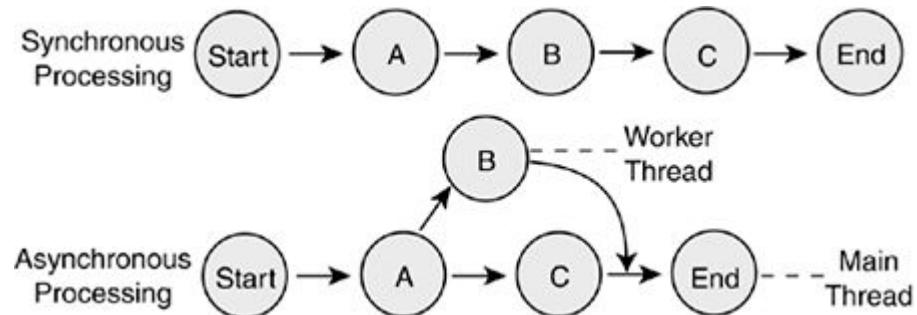
... this “movie fetching” operation might take a really long time.

Problem: Sometimes, operations (like HTTP requests) are time-intensive. We don't want our main thread focusing on completing these operation.

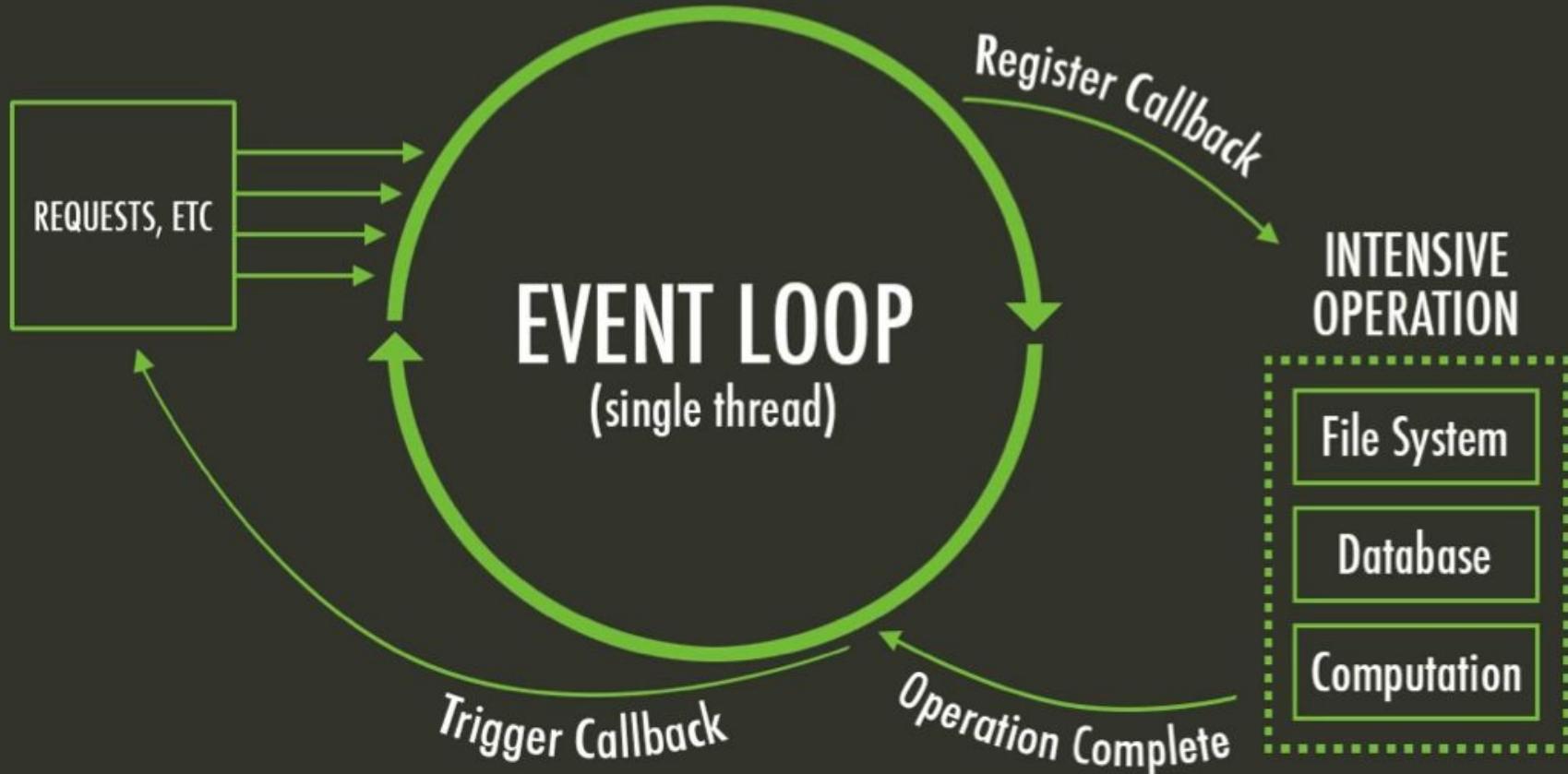
Solution: Allow operations (like HTTP requests) to happen independently from the main thread.

Introducing: Asynchrony

- “Asynchrony, in computer programming, refers to the occurrence of **events independent of the main program flow** and ways to deal with such events” - [https://en.wikipedia.org/wiki/Asynchrony_\(computer_programming\)](https://en.wikipedia.org/wiki/Asynchrony_(computer_programming))
- Allows our code to do several things at the same time
- ... without disturbing our main thread!



Useful link: <https://developer.mozilla.org/en-US/docs/Learn/JavaScript/Asynchronous/Introducing>



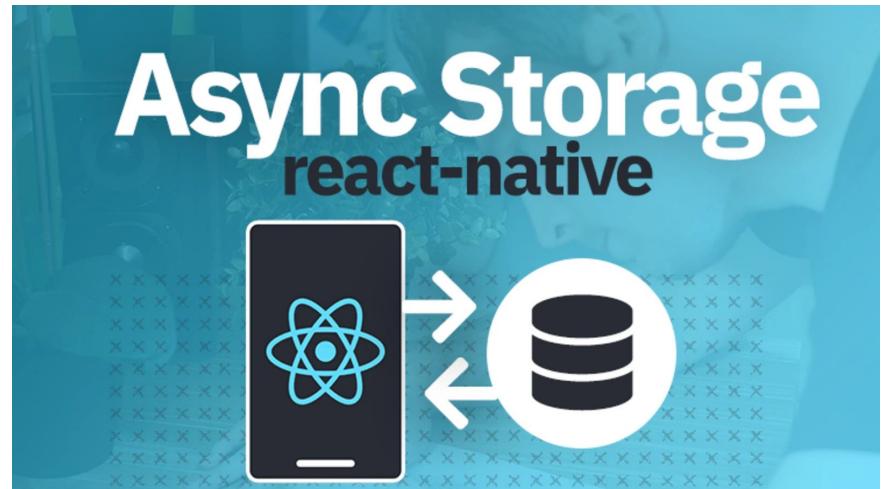


"If you really want to grow
as an entrepreneur,
**you've got to learn
to delegate."**

- Richard Branson

When do we use asynchrony?

- If we're running an operation that takes time
- In this course:
 - HTTP requests
 - Accessing storage



We'll learn about storage in Week 7!

How do we deal with asynchrony?

- A **Promise** represents the work that is happening asynchronously
- A Promise can be in one of 3 states
 - **Pending** : initial state, neither fulfilled nor rejected
 - **Fulfilled** : the operation was completed successfully
 - **Rejected** : the operation failed
- A Promise **resolves** into its resulting value
 - We must wait for the Promise to resolve (sneak peak: await)

Back to Movie Fetching

This is how we can fetch the movie data, utilizing asynchrony:

```
const getMovies = async () => {
  try {
    let response = await fetch(
      'https://reactnative.dev/movies.json'
    );
    let json = await response.json();
    return json.movies;
  } catch (error) {
    console.error(error);
  }
};
```

The fetch function

- An easy, logical way to **fetch resources asynchronously**
- Takes one mandatory argument: the path to the resource you want to fetch
- **Returns a Promise**, which resolves to the Response object associated with the resource

Example of using `fetch()`:

```
let response = await fetch('https://reactnative.dev/movies.json');
```



Promise



Path



Useful link: <https://reactnative.dev/docs/network#using-fetch>

Let's give it a shot!

You can run this code yourself on <https://es6console.com/> !

```
const getMovies = async () => {
  try {
    let response = await fetch(
      'https://reactnative.dev/movies.json'
    );
    let json = await response.json();
    return json.movies;
  } catch (error) {
    console.error(error);
  }
};

getMovies();
```

Source: <https://reactnative.dev/docs/network#using-fetch>

```
const getMoviesFromApiAsync = async () => {
  try {
    let response = await fetch(
      'https://reactnative.dev/movies.json'
    );
    let json = await response.json();
    return json.movies;
  } catch (error) {
    console.error(error);
  }
};
```

```
const getMoviesFromApiAsync = async () => {
  try {
    let response = await fetch(
      'https://reactnative.dev/movies.json'
    );
    let json = await response.json();
    return json.movies;
  } catch (error) {
    console.error(error);
  }
};
```

Declaring an
async function

```
const getMoviesFromApiAsync = async () => {
  try {
    let response = await fetch(
      'https://reactnative.dev/movies.json'
    );
    let json = await response.json();
    return json.movies;
  } catch (error) {
    console.error(error);
  }
};
```

Try this code... and
if it fails, catch
the error

```
const getMoviesFromApiAsync = async () => {
  try {
    let response = await fetch(
      'https://reactnative.dev/movies.json'
    );
    let json = await response.json();
    return json.movies;
  } catch (error) {
    console.error(error);
  }
};
```

Asynchronously fetch the movie data. `fetch()` returns a Promise that is resolved into a Response object.

```
const getMoviesFromApiAsync = async () => {
  try {
    let response = await fetch(
      'https://reactnative.dev/api/movies.json'
    );
    let json = await response.json();
    return json.movies;
  } catch (error) {
    console.error(error);
  }
};
```

Asynchronously retrieve the JSON of the data.

... and what's a JSON?

- **JavaScript Object Notation** is a syntax for exchanging data
- A JSON is represented in text

```
{  
    "courseNumber": 47,  
    "courseName": "Cross-Platform Mobile Development",  
    "studentCount": 76,  
    "schedule": "Tu/Th 10:30-11:50PST",  
    "quarter": "Winter",  
    "year": 2021,  
    "teachingTeam": ["James", "Abdallah", "Ryan", "Claire"]  
    "isCoolClass": true,  
    ...  
}
```

```
const getMoviesFromApiAsync = () => {
  try {
    let response = await fetch('https://reactnative.dev/api/movies');
    let json = await response.json();
    return json.movies;
  } catch (error) {
    console.error(error);
  }
};
```

Asynchronously get the JSON of the Response.
.json() returns a Promise that is resolved into a JSON object.

```
const getMoviesFromApiAsync = async () => {
  try {
    let response = await fetch(
      'https://reactnative.dev/movies.json'
    );
    let json = await response.json();
    return json.movies;
  } catch (error) {
    console.error(error);
  }
};
```

```
json = {
  ...
  "movies" : [...]
  ...
};
```

We return the "movies" value
from the JSON object

Conceptual Takeaways

- Issue: JavaScript is single threaded
- Solution: Asynchrony 
- Asynchronous operation = independent of the main program flow
- Promise = an object that represents the status/result of an asynchronous operation
- Fetch function = fetches data asynchronously and returns a Promise

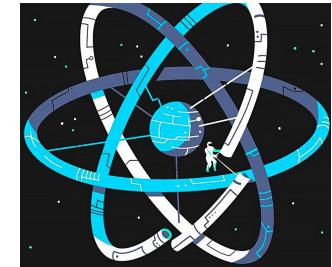
What are we working with?

- **JavaScript** is a programming language
- **JSX** is an extension of JavaScript that we use to describe what UIs should look like
- **UIs** are what people interact with on an app (text, images, buttons, etc.)
- **React** is a library for building UIs
- **React Native** is framework on top of React

```
let x = 1;
```

```
if (x === 1) {  
  let x = 2;  
}
```

```
const displayX = (  
  <View>  
    <Text>{x}</Text>  
  </View>  
)
```



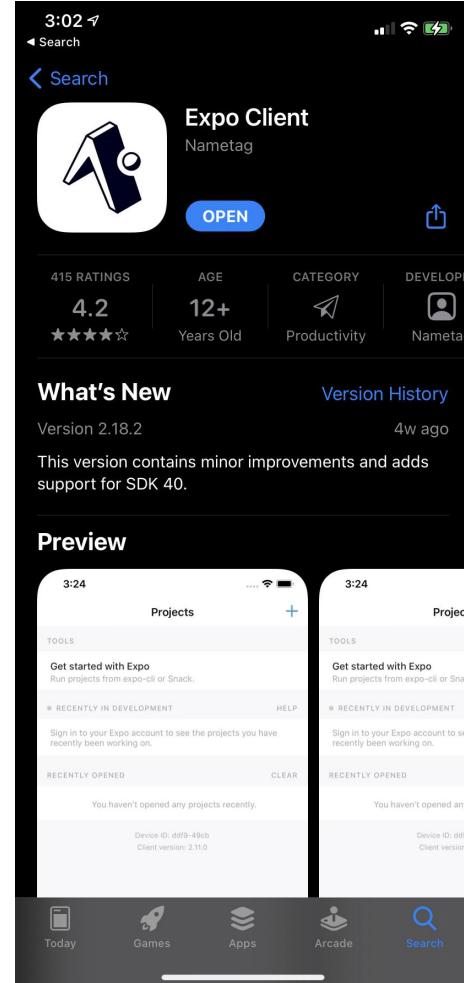
Today's goals

- Discuss the relationships between JavaScript, JSX, React, and React Native
- Learn more about Javascript (asynchrony, Promises, fetch)
- Introduce Expo tools
- Build a React Native app

Break

Instructions:

1. Make an Expo account @
<https://expo.io/signup>
2. Download the Expo Client app →
3. Sign in to Expo Client app



Today's goals

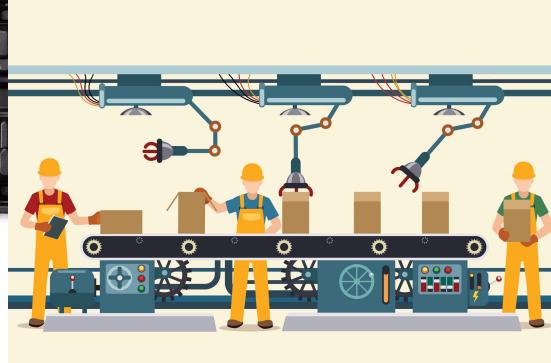
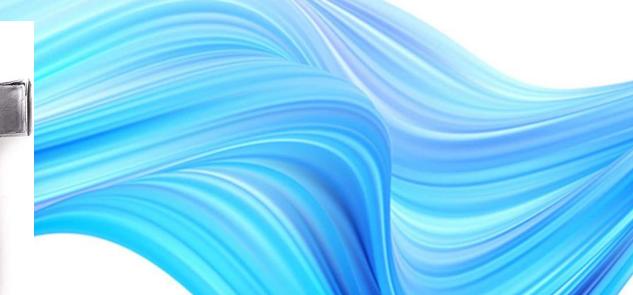
- Discuss the relationships between JavaScript, JSX, React, and React Native
- Learn more about Javascript (asynchrony, Promises, fetch)
- Introduce Expo tools
- Build a React Native app

How can we build a
React Native app?



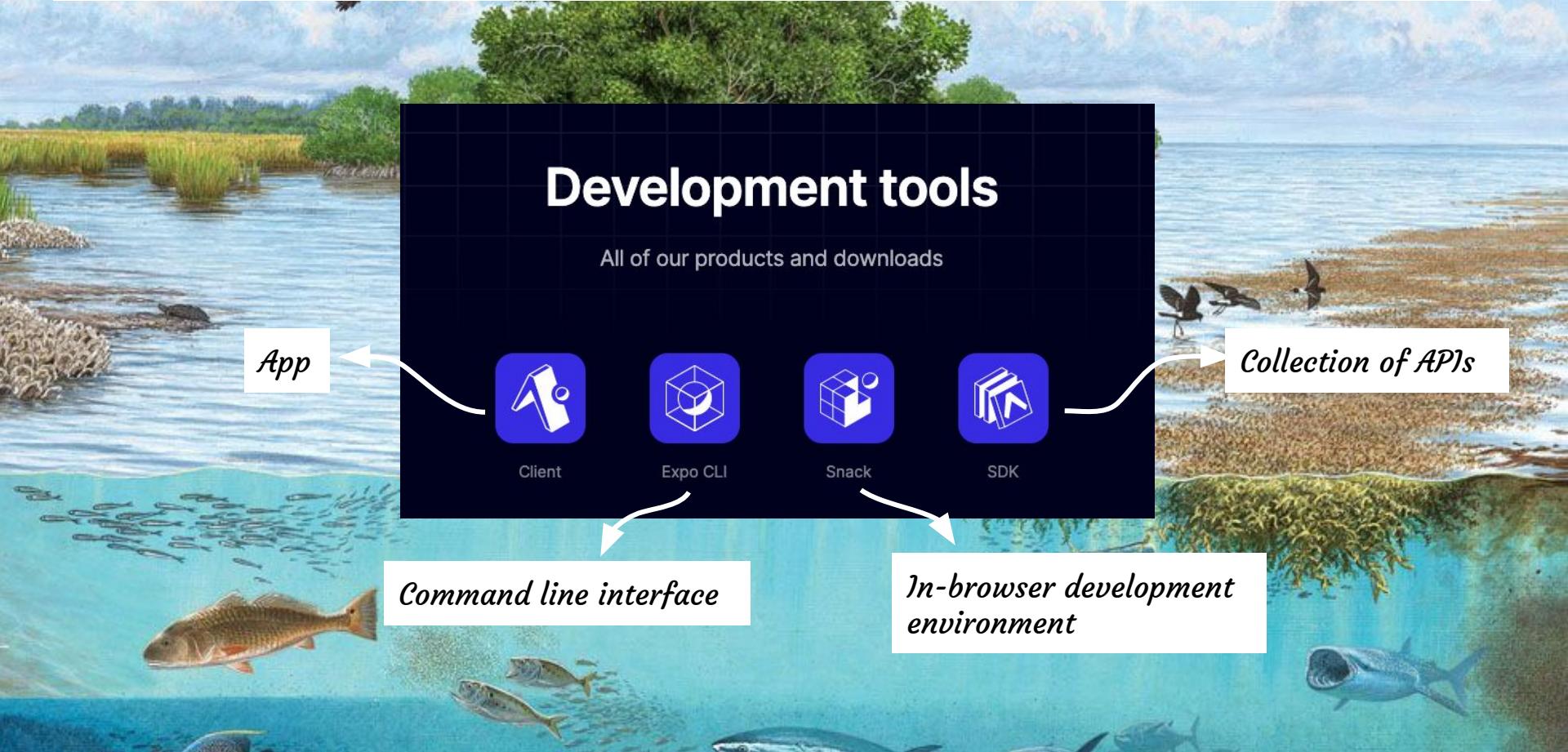
Introducing... Expo!

A toolkit, a workflow, a manufacturing process, an environment, an ecosystem...



Useful link: <https://docs.expo.io/>

Expo as an ecosystem



How does Expo define Expo?

“Expo is a framework and a platform for universal React applications. It is a set of tools and services built around React Native and native platforms that help you develop, build, deploy, and quickly iterate on iOS, Android, and web apps from the same JavaScript/TypeScript codebase.”

<https://docs.expo.io/>

JavaScript is a programming language

JSX is an extension of JavaScript that we use to describe what UIs should look like

UIs are what people interact with on an app (text, images, buttons, etc.)

React is a library for building UIs

React Native is framework on top of React

Expo is a set of tools for React Native



Why do we use Expo?

- ★ Sets up our projects
- ★ Deals with native libraries and 3rd party packages
- ★ Allows for iOS development on Windows
- ★ Expo Client app lets us to test apps on our phones
- ★ Offers in-browser development environment

Yes, you should start your React Native project with Expo

#reactnative #expo #ios #android

App & Flow Nov 29, 2019 · 4 min read

The screenshot shows a Stack Overflow article page. At the top, there's a header with the title and some social sharing icons. Below the header, there's a navigation bar with links like 'About', 'Products', 'For Teams', and a search bar. The main content area has a section titled 'Expo' with two subsections: 'Advantages' and 'Disadvantages', each containing a bulleted list of pros and cons.

Expo

Advantages

- Setting up a project is easy and can be done in minutes
- You (and other people) can open the project while you're working on it
- Sharing the app is easy (via QR-code or link), you don't have to send the whole .ipa or .apk file
- No build necessary to run the app
- Integrates some basic libraries in a standard project (Push Notifications, Asset Manager, etc.)
- You can eject it to ExpoKit and integrate native code continuing using some of the Expo features, but not all of them
- Expo can build .apk and .ipa files (distribution to stores possible with Expo)

Disadvantages

- You can't add native modules (probably a gamechanger for some)
- You can't use libraries that use native code in Objective-C/Java
- The standard Hello World app is about 25MB big (because of the integrated libraries)
- If you want to use: FaceDetector, ARKit, or Payments you need to eject it to ExpoKit
- Ejecting it to ExpoKit has a trade-off of features of Expo, e.g. you cannot share via QR code

Why you should start to use Expo



Alex Melnyk Follow

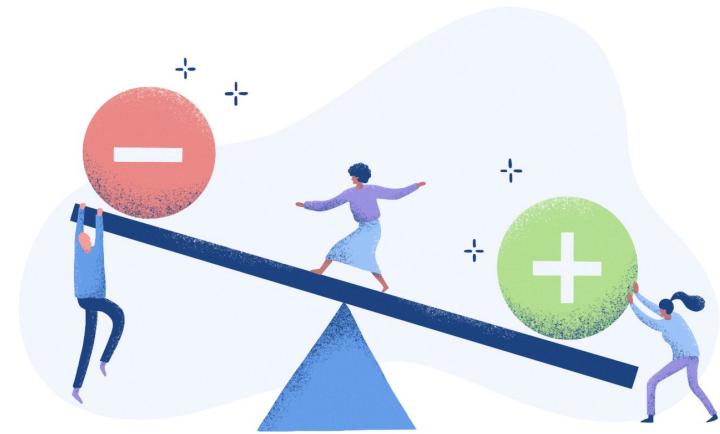
Nov 12, 2020 · 6 min read



Useful link: <https://docs.expo.io/>

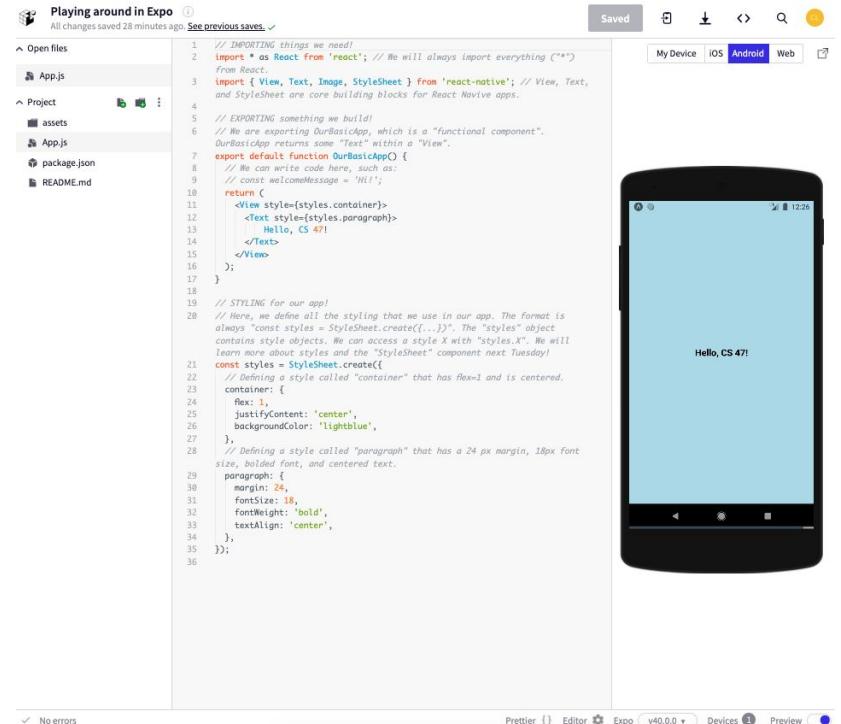
Is Expo perfect?

- Nope
 - Can't add custom native modules
 - Some native APIs are not available
 - Projects are big
- See [Expo Docs - Limitations](#)
- You can "[eject](#)" and turn Expo project into a pure RN app



Expo's in-browser development environment

- Easily build and test projects
- Great for sharing
- We will use this tool often for in-class activities



The screenshot shows the Expo development interface. On the left, the project structure includes files like App.js, assets, package.json, and README.md. The main area displays the code for App.js:

```
// IMPORTING something we need!
import React from 'react'; // We will always import everything ("*")
From React.
import { View, Text, Image, StyleSheet } from 'react-native'; // View, Text,
and StyleSheet are core building blocks for React Native apps.

// EXPORTING something we build!
// We are exporting OurBasicApp, which is a "functional component".
OurBasicApp returns some "Text" within a "View".
export default function OurBasicApp() {
  // We can write code here, such as:
  // const welcomeMessage = 'Hello!';
  return (
    <View style={styles.container}>
      <Text style={styles.paragraph}>
        Hello, CS 47!
      </Text>
    </View>
  );
}

// STYLING for our app!
// Here, we define all the styling that we use in our app. The Format is
always "const styles = StyleSheet.create({...})". The "styles" object
contains style objects. We can access a style X with "styles.X". We will
learn more about styles and the "StyleSheet" component next Tuesday!
const styles = StyleSheet.create({
  // Defining a style called "container" that has flex=1 and is centered.
  container: {
    flex: 1,
    justifyContent: 'center',
    backgroundColor: 'lightblue',
  },
  // Defining a style called "paragraph" that has a 24 px margin, 18px font
  size, bold font, and centered text.
  paragraph: {
    margin: 24,
    fontSize: 18,
    fontWeight: 'bold',
    textAlign: 'center',
  },
});

```

On the right, a mobile phone icon shows the app running with the text "Hello, CS 47!". The interface also includes tabs for My Device, iOS, Android, Web, and a preview section at the bottom.

Useful link: <https://blog.expo.io/sketch-a-playground-for-react-native-16b2401f44a2>

Our First Expo Project

Playing around in Expo ①
All changes saved 28 minutes ago. See previous saves. ✓

Open files

App.js

Project

assets

App.js

package.json

README.md

```
// IMPORTING things we need!
import * as React from 'react'; // We will always import everything ("*")
from React.

import { View, Text, Image, StyleSheet } from 'react-native'; // View, Text,
and StyleSheet are core building blocks for React Native apps.

// EXPORTING something we build!
// We are exporting OurBasicApp, which is a "functional component".
// OurBasicApp returns some "Text" within a "View".
export default function OurBasicApp() {
  // We can write code here, such as:
  // const welcomeMessage = 'Hello!';
  return (
    <View style={styles.container}>
      <Text style={styles.paragraph}>
        Hello, CS 47!
      </Text>
    </View>
  );
}

// STYLING for our app!
// Here, we define all the styling that we use in our app. The format is
// always "const styles = StyleSheet.create({ ... })". The "styles" object
// contains style objects. We can access a style X with "styles.X". We will
// learn more about styles and the "StyleSheet" component next Tuesday!
const styles = StyleSheet.create({
  // Defining a style called "container" that has flex=1 and is centered.
  container: {
    flex: 1,
    justifyContent: 'center',
    backgroundColor: 'lightblue',
  },
  // Defining a style called "paragraph" that has a 24 px margin, 18px font
  // size, bolded font, and centered text.
  paragraph: {
    margin: 24,
    fontSize: 18,
    fontWeight: 'bold',
    textAlign: 'center',
  },
});

```

Saved

My Device iOS Android Web

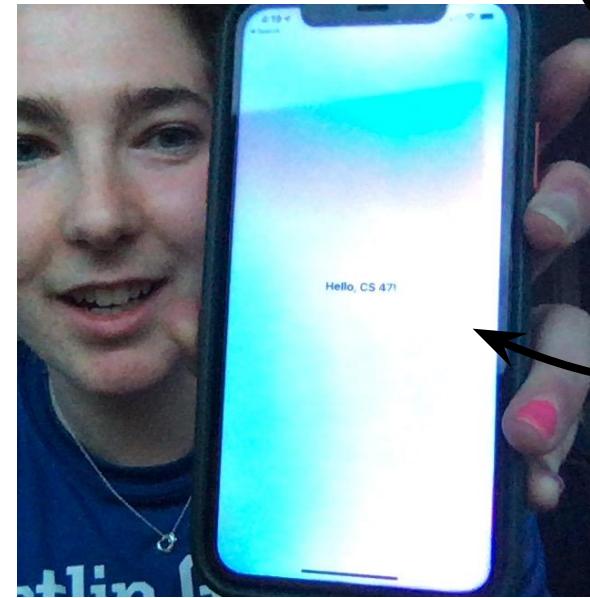
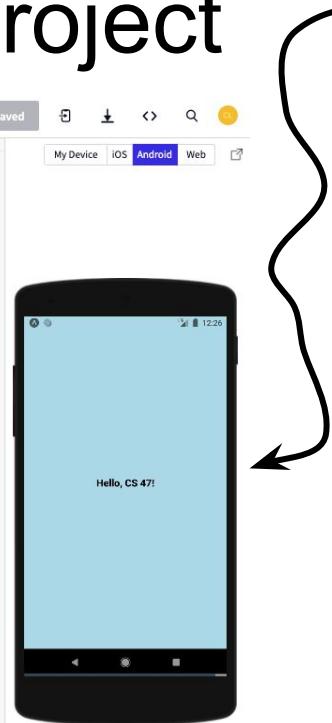
12:26

Hello, CS 47!

Prettier {} Editor Expo v40.0.0 Devices 1 Preview

No errors

A simple “Hello, CS 47!” app



Render your app on your phone!

Our First Expo Project

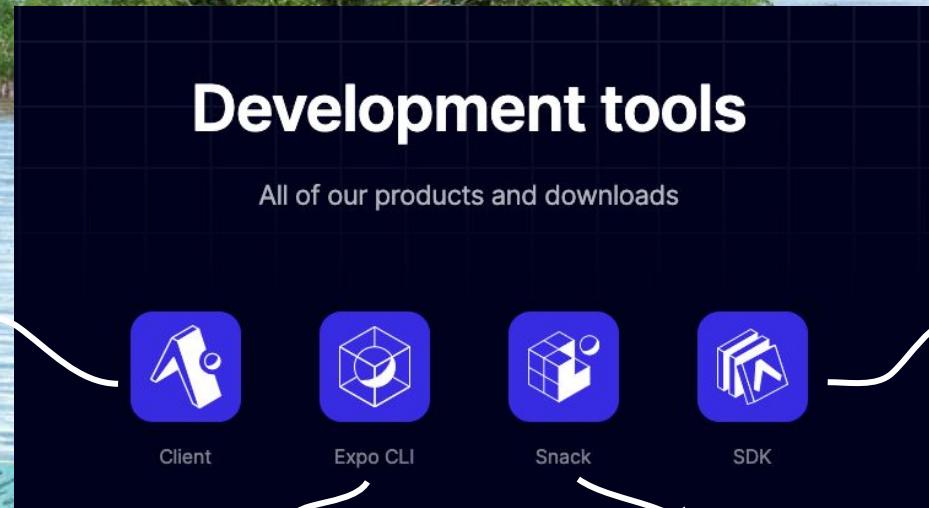
- Sign into your Expo account in your browser (<https://expo.io/>)
 - Have your phone handy (with Expo Client app)
1. Go to <https://snack.expo.io/@claire/playing-around-in-expo>
 2. Save the project to your own Expo account
 3. On your phone, scan the project's QR code

... so that's Expo's in-browser development environment



- It's cool, right?!
- We'll use the Expo ecosystem the rest of the quarter.
- Get familiar with it!
- You might find Expo's [Glossary of terms](#) helpful
- (Cool [CS 47 shoutout](#) 😎)

Expo is a ecosystem



App

Collection of APIs

Command line interface

In-browser development environment

Assignment 1: Setup

- Set up your local development environment ([macOS](#), [Windows](#))
- Submission: email us a screenshot of the running simulator
with the text changed
- Due next Tuesday January 19th at 11:59pm PT

Simulator File Edit Device I/O Features Debug Window Help

testingExpoFirstApp — node + node ~/nvm/versions/node/v10.14.2/bin/expo start — 105x66

To run the app, choose one of:
> Scan the QR code above with the Expo app (Android) or the Camera app (iOS).
> Press a for Android emulator, or i for iOS simulator, or w to run on web.
> Press e to send a link to your phone with email.

Press ? to show a list of all available commands.
Logs for your project will appear below. Press Ctrl+C to exit.
Opening exp://127.0.0.1:19080 on iPhone 12
Finished building JavaScript bundle in 84786ms.
Running application on iPhone 12.

Stopping packager...
> Closing Expo server
> Stopping Metro bundler
Packager stopped.
clairerosenfeld@Claire's-MacBook-Air testingExpoFirstApp % expo start
Starting project at /Users/clairerosenfeld/Expo App Development/testingExpoFirstApp
Expo DevTools is running at http://localhost:19080
Opening DevTools in the browser... (press shift-d to disable)
Starting Metro Bundler

exp://192.168.12.22:19080



EXPLORER

OPEN EDITORS

TESTINGEXPOFIRSTAPP

- .expo
- packager-info.json
- README.md
- settings.json
- .expo-shared
- .vscode
- assets
- node_modules
- .gitignore

JS App.js M

{} app.json
{} babel.config.js
{} package.json M
yarn.lock U

JS App.js x yarn.lock

JS App.js > styles > container

```
4 export default function App() {
5   return (
6     <View style={styles.container}>
7       <Text>Tada!</Text>
8       <StatusBar style="auto" />
9     </View>
10   );
11 }
12 }
13
14 const styles = StyleSheet.create(
15   container: [
16     flex: 1,
17     backgroundColor: "#fff",
18     alignItems: "center",
19     justifyContent: "center",
20   ],
21 });
22 }
```

OUTLINE

- App
- styles
- container
- alignItems
- backgroundColor
- flex

TIMELINE

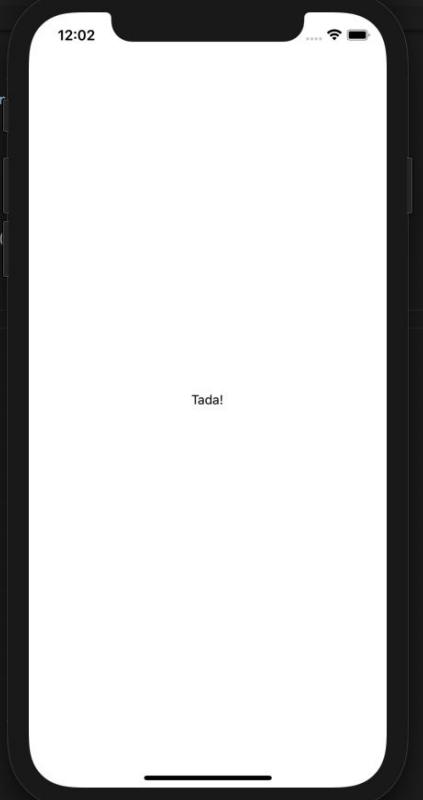
HT SO EXCITED! 🎉

12:02

Tada!

Ln 18, Col 26 Spaces: 2 UTF-8 LF JavaScript ✓ ES

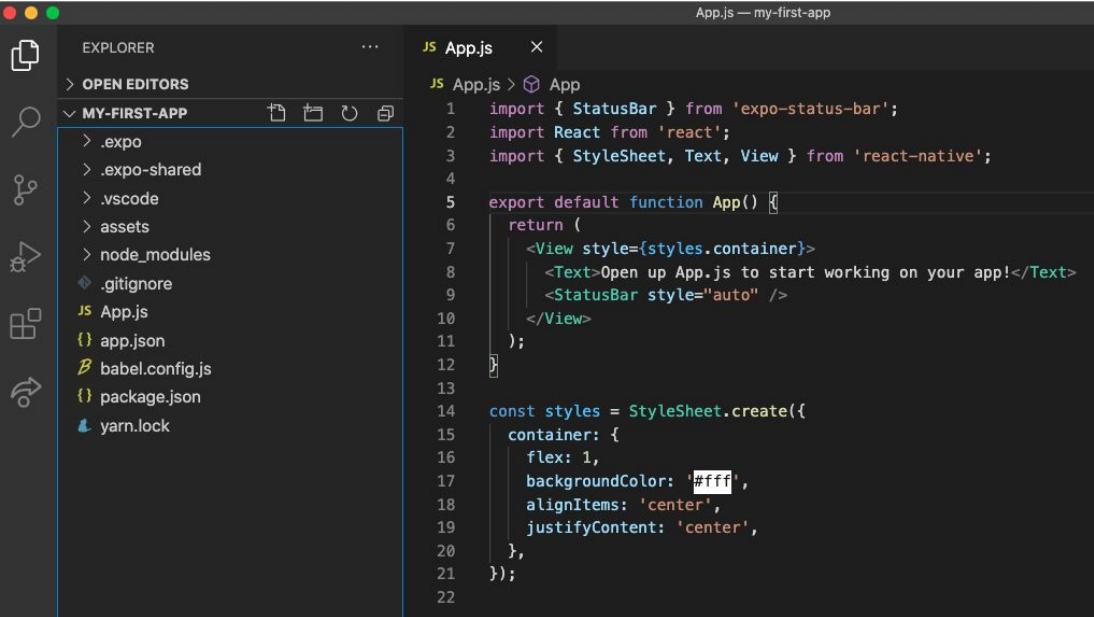
React Native Packager Live Share



Submission: email us a screenshot of your Simulator with the text changed.



What makes up an Expo React Native project?



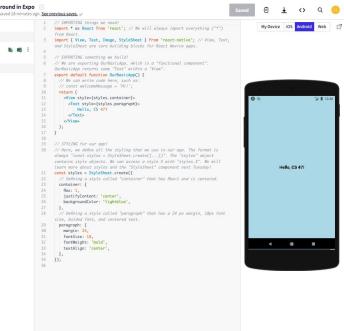
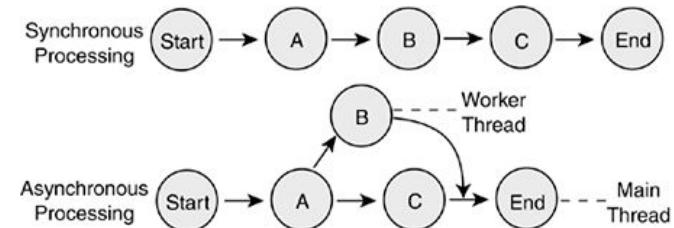
The screenshot shows a dark-themed instance of Visual Studio Code. On the left is the Explorer sidebar, which lists the project structure under 'MY-FIRST-APP': .expo, .expo-shared, .vscode, assets, node_modules, .gitignore, App.js (selected), app.json, babel.config.js, package.json, and yarn.lock. The main editor area displays the contents of App.js:

```
JS App.js — my-first-app
JS App.js > App
1 import { StatusBar } from 'expo-status-bar';
2 import React from 'react';
3 import { StyleSheet, Text, View } from 'react-native';
4
5 export default function App() {
6   return (
7     <View style={styles.container}>
8       <Text>Open up App.js to start working on your app!</Text>
9       <StatusBar style="auto" />
10    </View>
11  );
12}
13
14 const styles = StyleSheet.create({
15   container: {
16     flex: 1,
17     backgroundColor: '#fff',
18     alignItems: 'center',
19     justifyContent: 'center',
20   },
21 });
22
```

See [last slide in deck](#) for detailed descriptions.

Reflecting on what we learned today

- JavaScript
 - What is asynchrony?
 - What are Promises?
- Expo
 - Why are we using Expo?
 - How do we use Expo?
- React Native
 - What does a React Native project look like?



Takeaways

- A takeaway can be a concept, idea, analogy, epiphany, question, goal, etc..
- Discuss in breakout rooms
- When you return, be ready to share one takeaway!

Takeaway

- ★ Be curious about the technological universe we exist within. Utilize the Internet, ask questions, and get excited about exploring new concepts and tools.

Next lecture...

- What makes up a UI?
 - What are **Components**? 
 - How do we **style** our Components? 

Course logistics + words of encouragement

- Enrollment decisions will be made this weekend. The class is capped at 50 students. Students taking CS 147 have priority.
 - Auditing is an **excellent** option!
- Check the syllabus on [the website](#) for what topics we will cover.
- You will get a ton of practice with JavaScript throughout the quarter.
- OH will always be available for extra help.
- We are confident that **everyone** will be able to make great mobile apps by the end of the course! We are here to support everyone along the way.
- We take feedback seriously – please continue providing feedback on exit tickets and sending us emails!

Exit ticket

Share a screenshot of your favorite mobile app UI!

Are you enraptured with your Instagram Profile page? Do you love the New Message popup in your Mail app? Do you respect the simplicity of your Camera app? Take a screenshot of a favorite app UI and add it to our class Google Slides deck. Feel free to block out anything you don't want your peers to see;)

LINK TO GOOGLE SLIDES DECK

LINK TO TODAY'S EXIT TICKET FORM

Office Hours

OH start on week 2. Until then, email us directly or Slack us!

Abdallah AbuHashem:

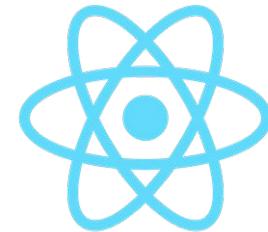
OH: TBD & by appointment
Email: aabuhash@stanford.edu

Claire Rosenfeld

OH: TBD & by appointment
Email: clairero@stanford.edu

Ryan Chen

OH: TBD & by appointment
Email: rjc45@stanford.edu



Today's terms

- JavaScript = scripting language commonly used in web development
- JSX = extension of JS, produces React elements
- UI = user interface
- React = JS library for building UIs
- React Native = uses React framework + native platforms to develop mobile apps
- Asynchrony = independent of the main program flow
- Promises = an object that represents the status of an asynchronous operation
- Fetch = function that fetches data asynchronously and returns a Promise
- Expo = tools/ecosystem/workflow for building apps , built around React Native
- Expo Client app = iOS + Android app that runs Expo apps
- Expo CLI = command line tool for working with Expo
- Simulator = simulates mobile devices on your computer
- Package manager = manages libraries (aka dependencies) for our project

Everything in an Expo project

- .expo folder = contains configuration information for your machine
- .expo-shared folder = contains info about asset optimization
- assets folder = contains assets for project (images, icons, fonts, videos)
- node_modules folder = contains dependencies of project
- .gitignore file = specifies files that Git should ignore
- App.js file = code to configure our app
- app.json file = metadata
- babel.config.js file = configuration of how JS code is compiled
- package.json file = manages dependencies
- yarn.lock file = more info about dependencies (stored by Yarn)
- Yarn = package manager

- ★ Out of this list, the only files/folders that are interesting to us right now are assets and App.js
- ★ Do not worry about knowing/memorizing/understanding this :D