CS47: Cross-Platform Mobile Development

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

James Landay
Abdallah AbuHashem
Claire Rosenfeld
Ryan Chen

https://cs47.stanford.edu

Winter 2021

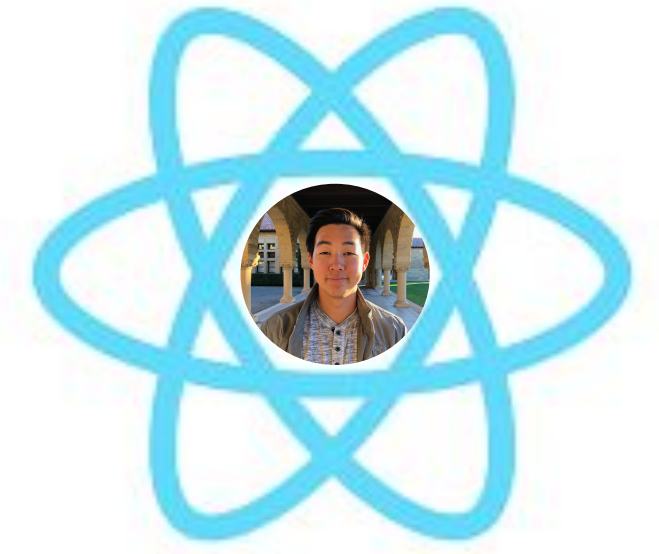

Welcome back
to CS 47!

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!

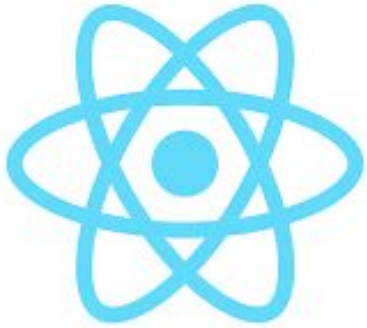
Last lecture

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



Slides from last lecture

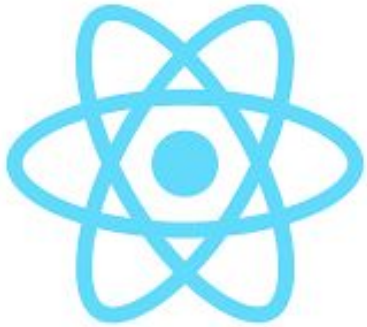
Reviewing what we learned last lecture



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

Useful link: https://reactnative.dev/

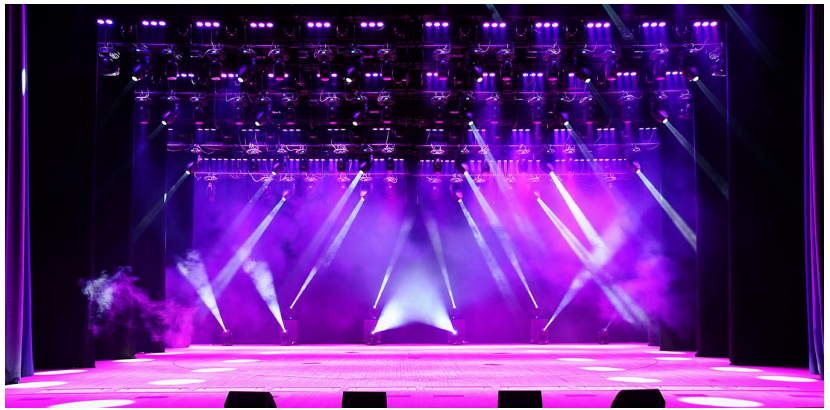
Reviewing what we learned last lecture



- 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

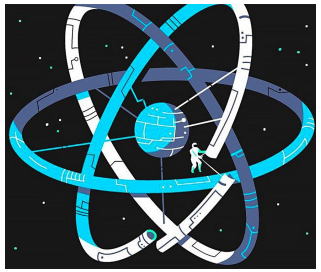


What are we working with?

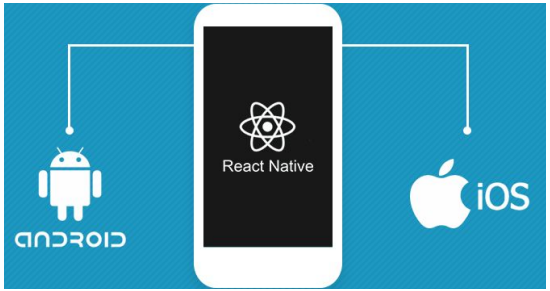
```
let x = 1;

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

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



- **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



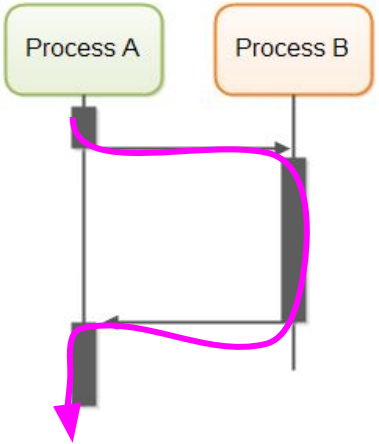
Adding to our JavaScript knowledge

- 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

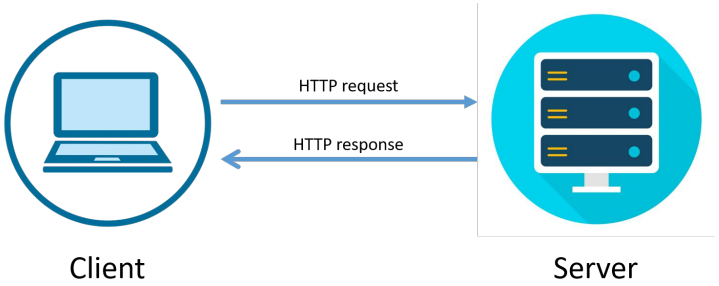


Sometimes this isn't ideal. Why?

Useful link: https://developer.mozilla.org/en-US/docs/Glossary/Thread

Example: HTTP Requests

- HTTP = **H**yper**t**ext **T**ransfer **P**rotocol
- 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.

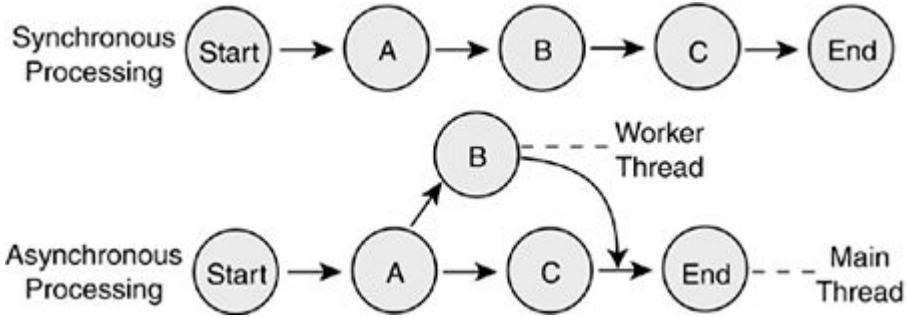
Useful link: https://developer.mozilla.org/en-US/docs/Web/HTTP/Overview

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

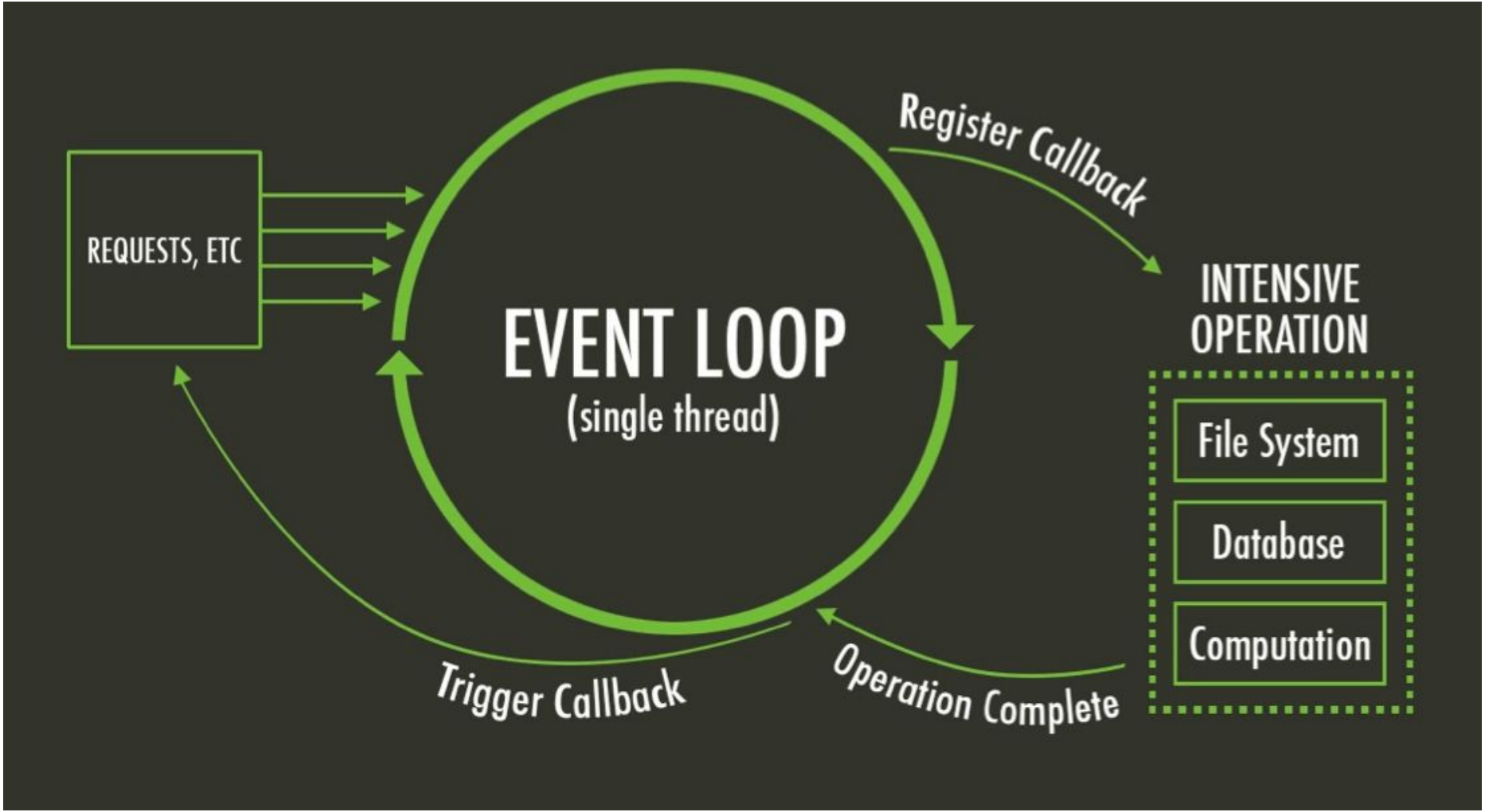
Solution: Allow operations (like HTTP requests) to happen independently from the main thread.

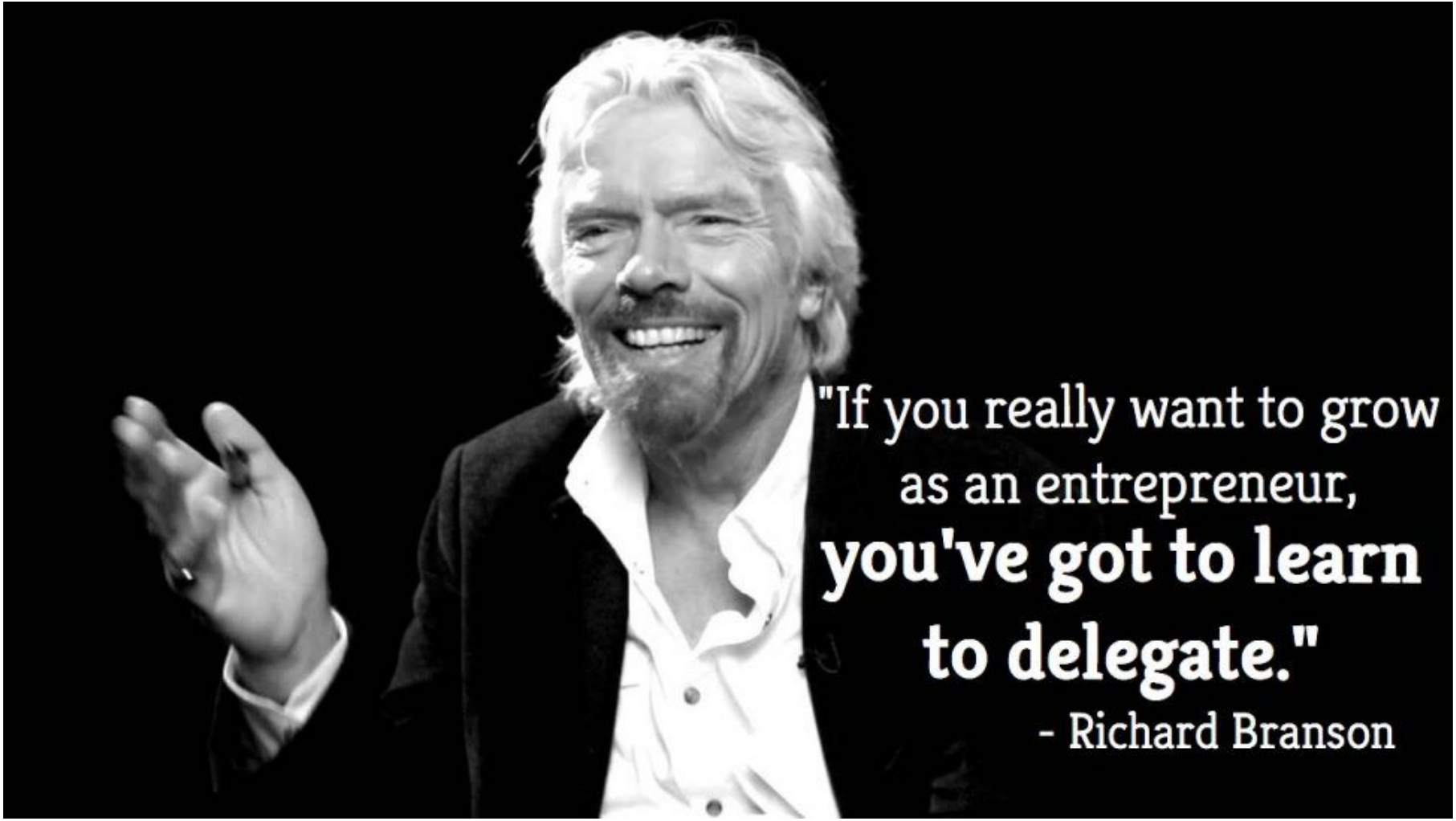
Introducing: Asynchrony

- “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)
- Allows our code to do several things at the same time
- … without disturbing our main thread!



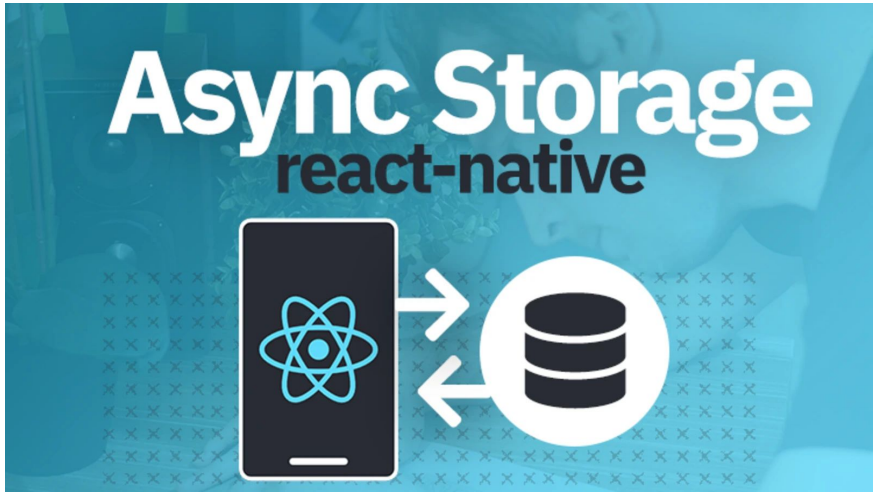
Useful link: https://developer.mozilla.org/en-US/docs/Learn/JavaScript/Asynchronous/Introducing


REQUESTS, ETC
EVENT LOOP
(single thread)
Register Callback
INTENSIVE
OPERATION
File System
Database
Computation
Trigger Callback
Operation Complete


"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`)

Useful link: https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Promise

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);
  }
};
```

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

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);
  }
};
```

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);
  }
};
```

Declaring an async function

Useful link: https://javascript.info/async-await

```
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

Useful link: https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Statements/try...catch

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

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

Useful link: https://developer.mozilla.org/en-US/docs/Web/API/Response

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

Asynchronously retrieve the JSON of the data.

Useful link: https://developer.mozilla.org/en-US/docs/Web/API/Body/json

… and what's a JSON?

- **J**ava**S**cript **O**bject **N**otation 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,
   ...
}
```

Useful link: https://developer.mozilla.org/en-US/docs/Learn/JavaScript/Objects/JSON

```
const getMoviesFromApiAsyn
  try {
    let response = await f
      'https://reactnative
    );
    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.

Useful link: https://developer.mozilla.org/en-US/docs/Web/API/Response

```
const getMoviesFromApiAsync = async () => {
  try {
    let response = await fetch(
      'https://reactnative.dev/m
    );
    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

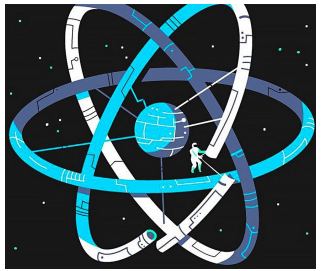
What are we working with?

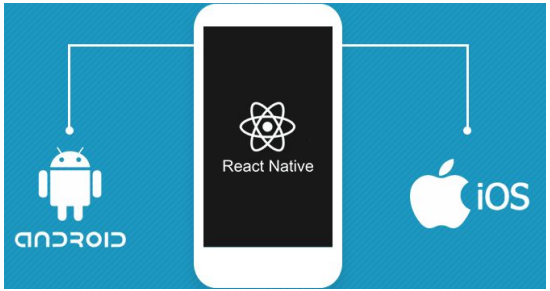
```
let x = 1;

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

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

- **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





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

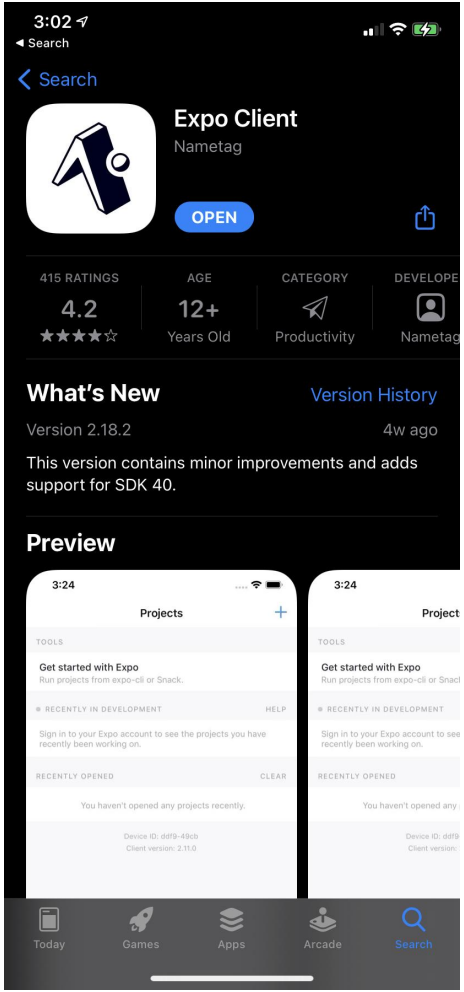
Break 

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

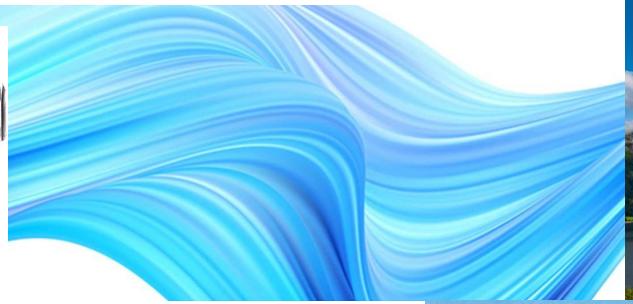
- Discuss the relationships between JavaScript, JSX, React, and React Native
- Learn more about Javascript (asynchrony, Promises, fetch)
- Introduce Expo tools
- Build a React Native app

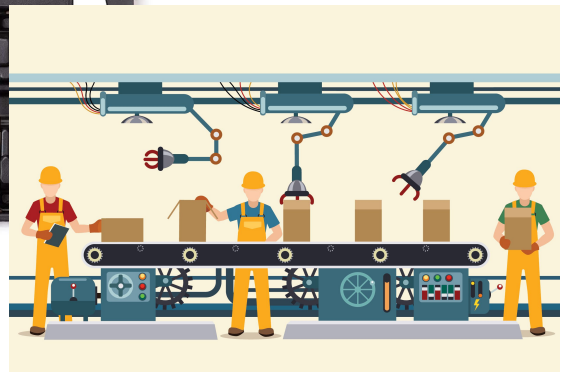
How can we build a React Native app?


Expo

Introducing... Expo!

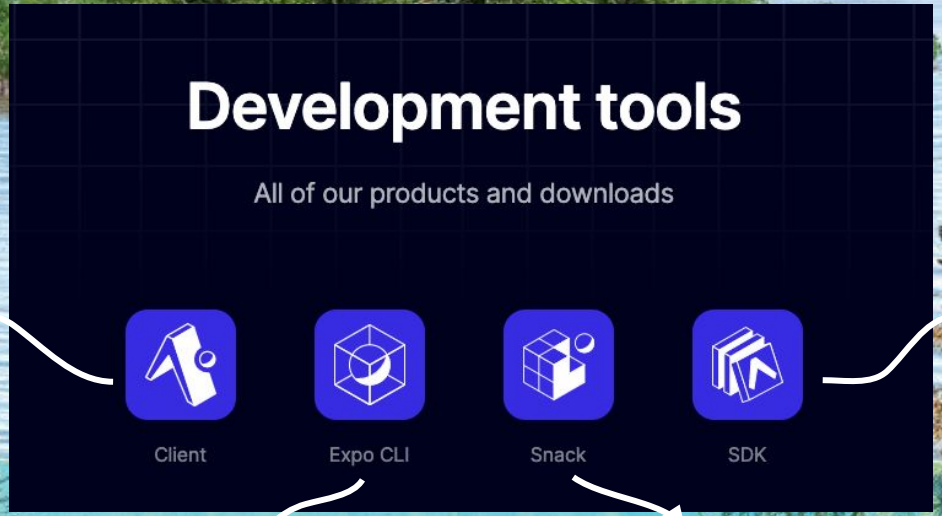
A toolkit, a workflow, a manufacturing process, an environment, an ecosystem…





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

Expo as an ecosystem



App

Collection of APIs

Command line interface

In-browser development environment

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

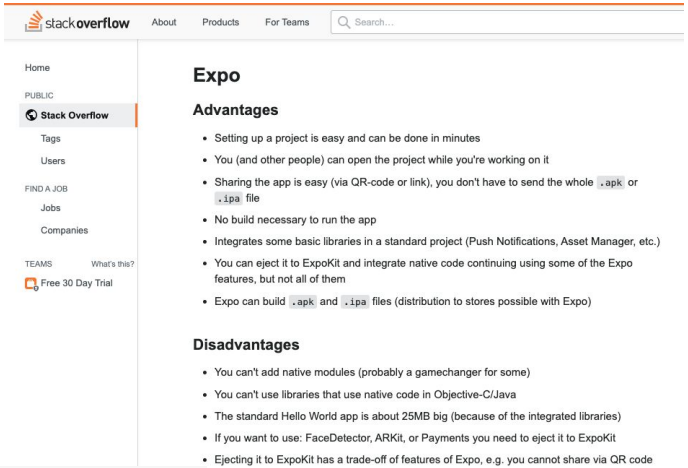
Expo is a set of tools for React Native

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


Expo

Why do we use Expo?

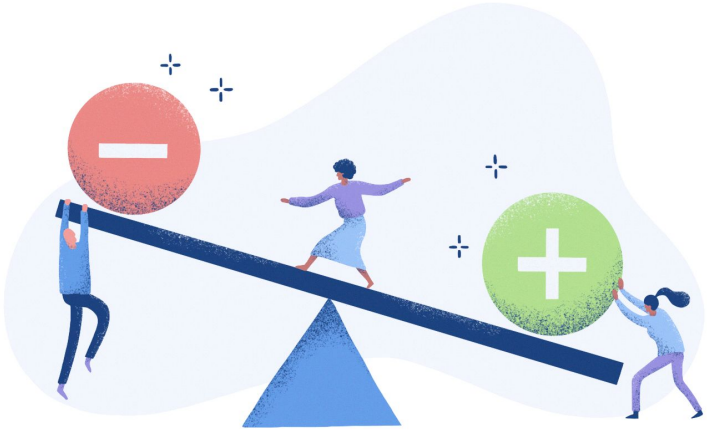
- ★ Sets up our projects
- ★ Deals with native libraries and 3rd party packages
- ★ Allows for iOS development on Windows
- ★ Expo Client app lets us to test apps on our phones
- ★ Offers in-browser development environment



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

Is Expo perfect?

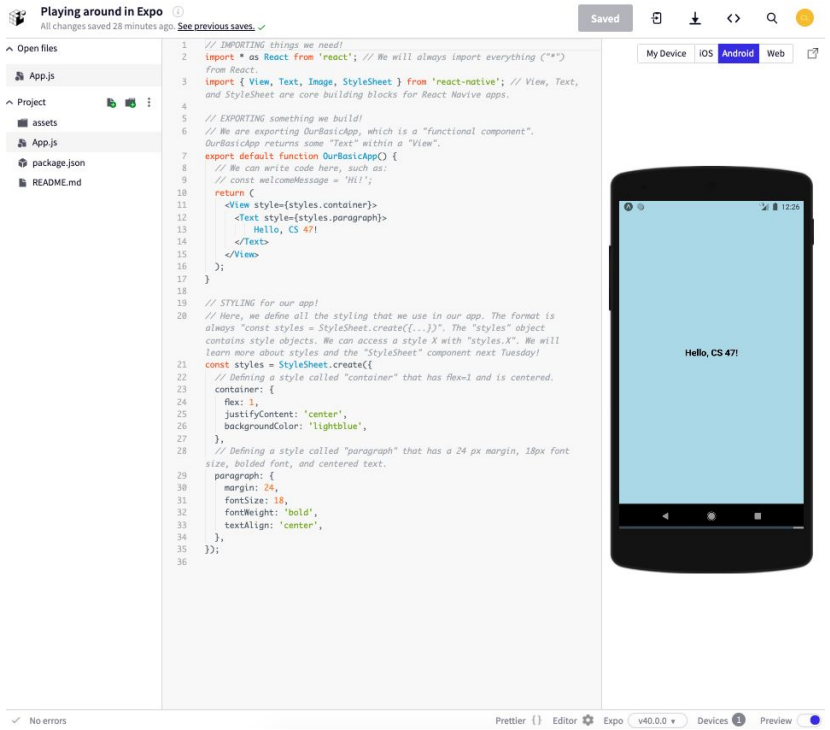
- Nope
  - Can’t add custom native modules
  - Some native APIs are not available
  - Projects are big
- See Expo Docs - Limitations
- You can “eject” and turn Expo project into a pure RN app



Useful link: https://docs.expo.io/introduction/why-not-expo/

Expo’s in-browser development environment

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

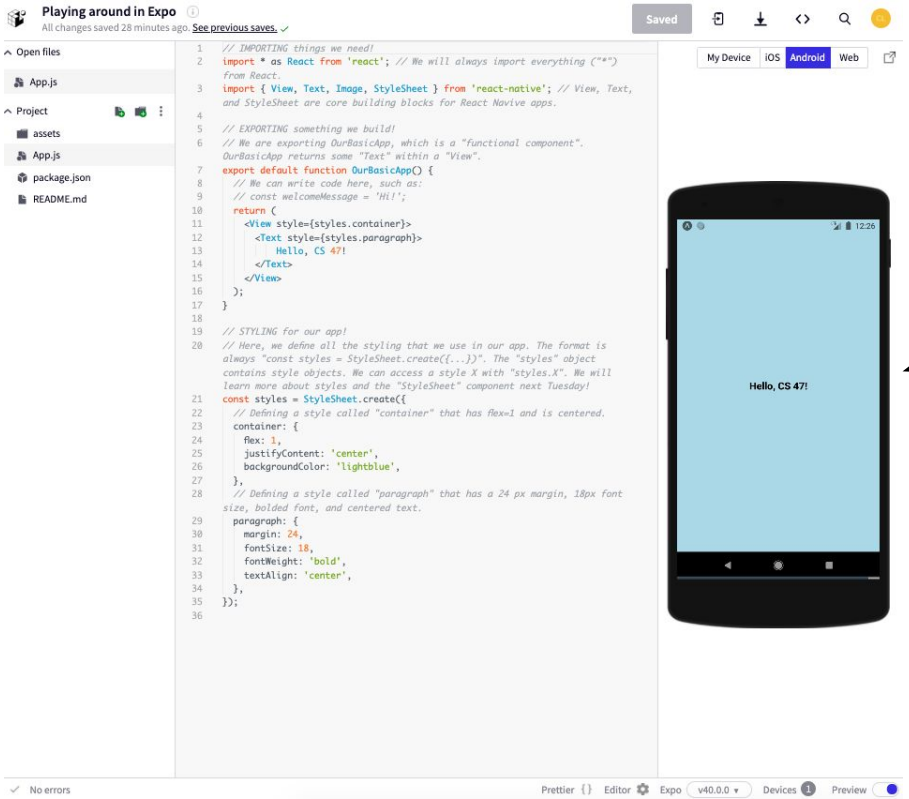


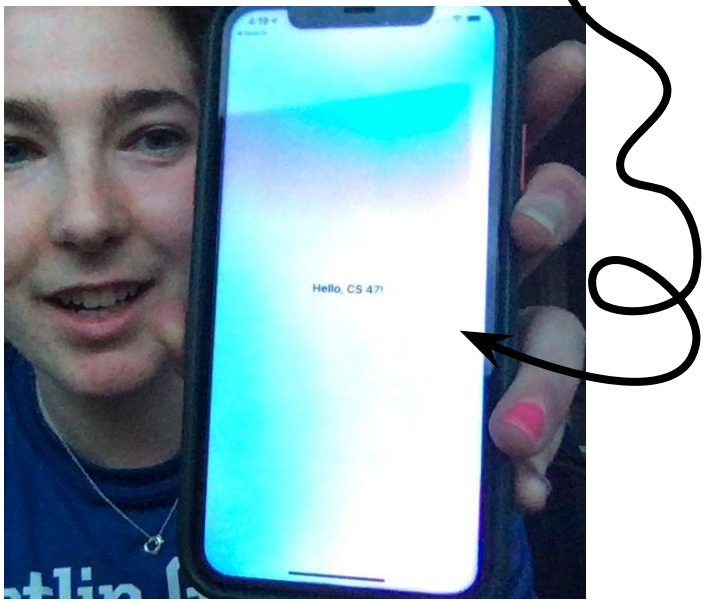
Useful link: https://blog.expo.io/sketch-a-playground-for-react-native-16b2401f44a2

Our First Expo Project

A simple “Hello, CS 47!” app

Render your app on <u>your</u> 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

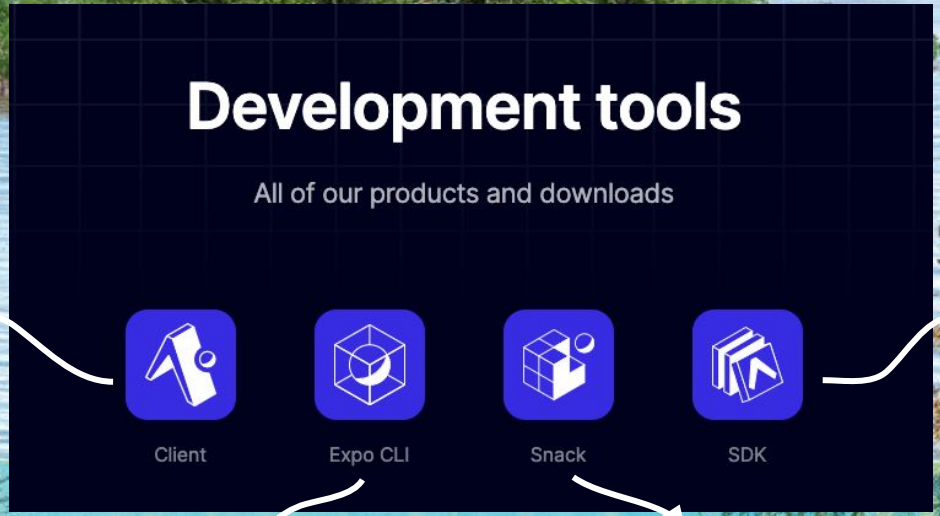
… so that's Expo's in-browser development environment

- 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 🤓)

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

Expo is a ecosystem



App

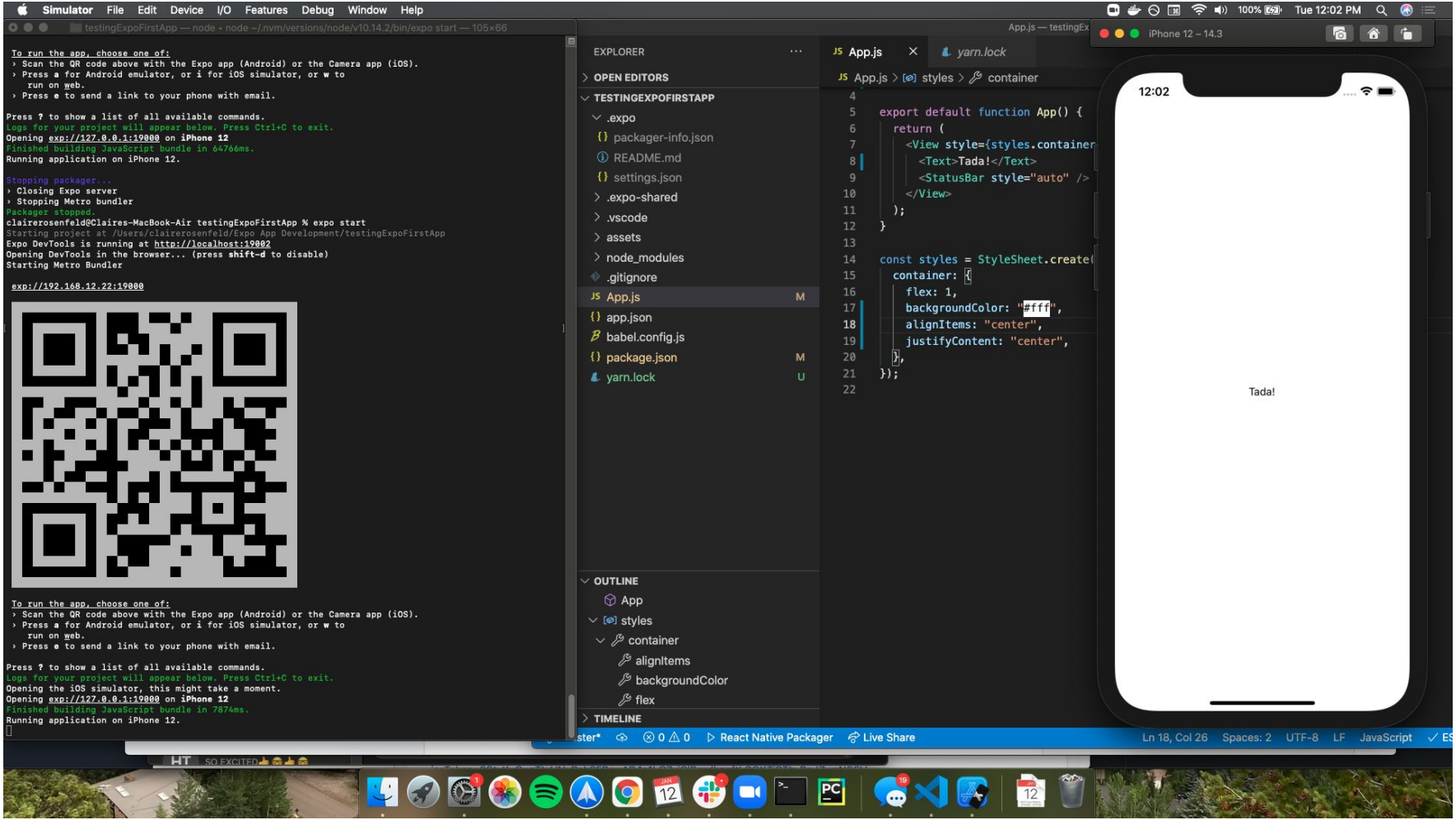
Collection of APIs

Command line interface

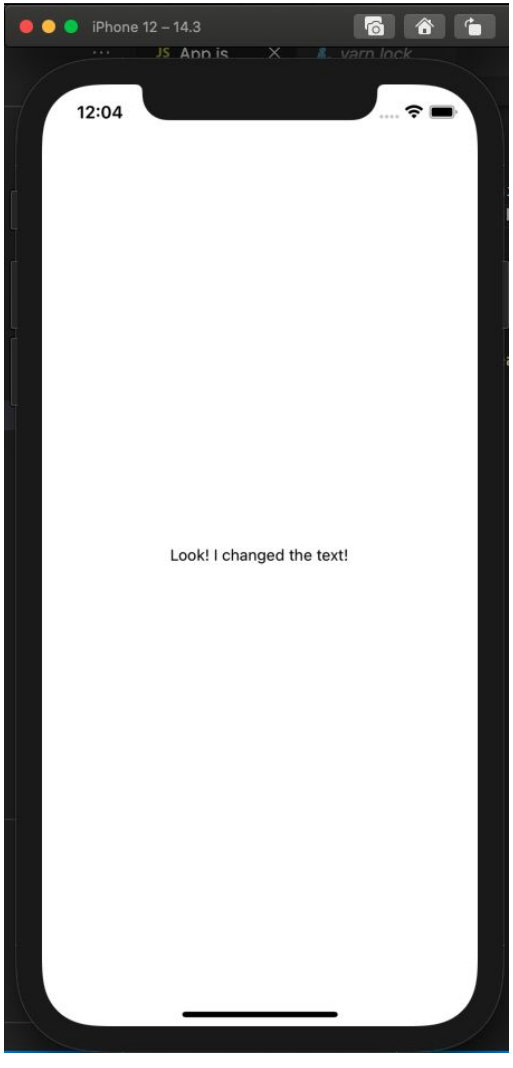
In-browser development environment

Assignment 1: Setup

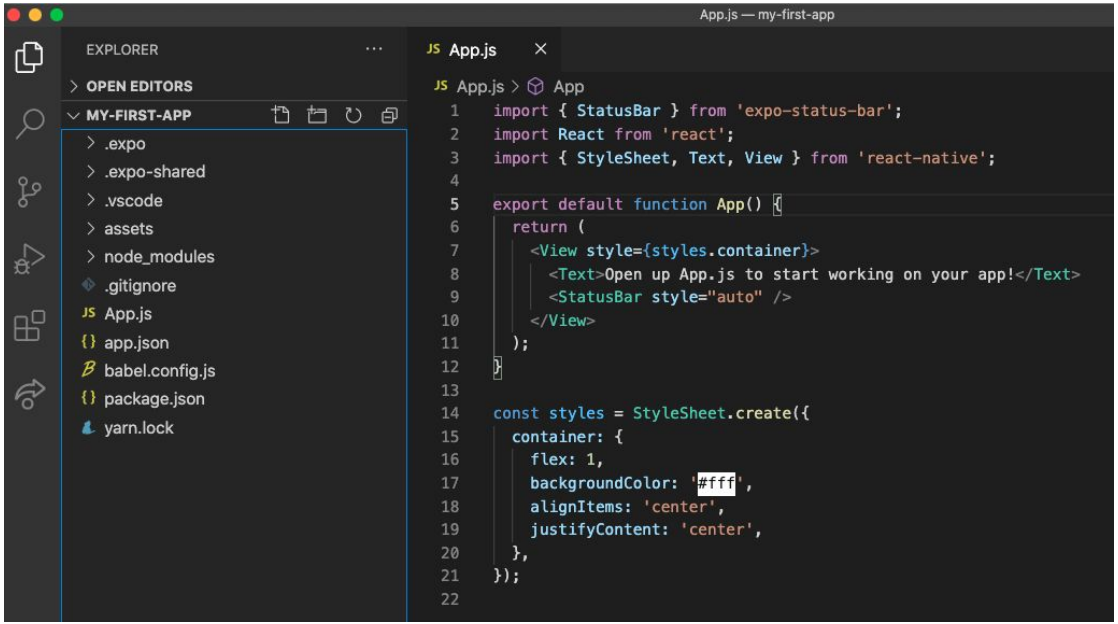
- Set up your local development environment (macOS, Windows)
- Submission: email us a screenshot of the running simulator with the text changed
- Due next Tuesday January 19th at 11:59pm PT


Simulator File Edit Device I/O Features Debug Window Help
Tue 12:02 PM
testingExpoFirstApp — node • node ~/.nvm/versions/node/v10.14.2/bin/expo start — 105×66
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://127.0.0.1:19000 on iPhone 12
Finished building JavaScript bundle in 6476ms.
Running application on iPhone 12.
Stopping packager...
› Closing Expo server
› Stopping Metro bundler
Packager stopped.
clairerosenfeld@Claires-MacBook-Air testingExpoFirstApp % expo start
Starting project at /Users/clairerosenfeld/Expo App Development/testingExpoFirstApp
Expo DevTools is running at http://localhost:19002
Opening DevTools in the browser... (press shift-d to disable)
Starting Metro Bundler
exp://192.168.12.22:19000
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 the iOS simulator, this might take a moment.
Opening exp://127.0.0.1:19000 on iPhone 12
Finished building JavaScript bundle in 7874ms.
Running application on iPhone 12.
EXPLORER
OPEN EDITORS
TESTINGEXPOFIRSTAPP
.expo
packager-info.json
README.md
settings.json
.expo-shared
.vscode
assets
node_modules
.gitignore
App.js M
app.json
babel.config.js
package.json M
yarn.lock U
OUTLINE
App
styles
container
alignItems
backgroundColor
flex
TIMELINE
App.js
yarn.lock
App.js > styles > container
export default function App() {
return (
<View style={styles.container
<Text>Tada!</Text>
<StatusBar style="auto" />
</View>
);
}
const styles = StyleSheet.create(
container: {
flex: 1,
backgroundColor: "#fff",
alignItems: "center",
justifyContent: "center",
},
});
iPhone 12 – 14.3
12:02
Tada!
React Native Packager
Live Share
Ln 18, Col 26
Spaces: 2
UTF-8
LF
JavaScript

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



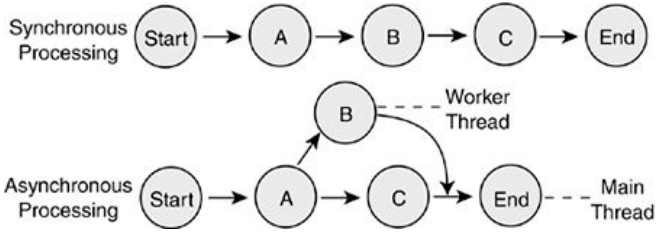
What makes up an Expo React Native project?

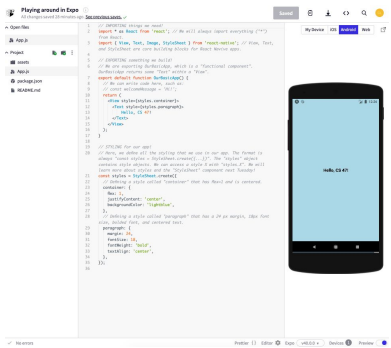


See last slide in deck for detailed descriptions.

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

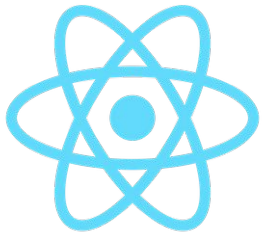
Office Hours

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