# React Native

Part 1 – Introduction and hello world app

**Teacher**

Mahyar Tourchi Moghaddam
mtmo@mmmi.sdu.dk

**Instructors**

Morten & Frederik

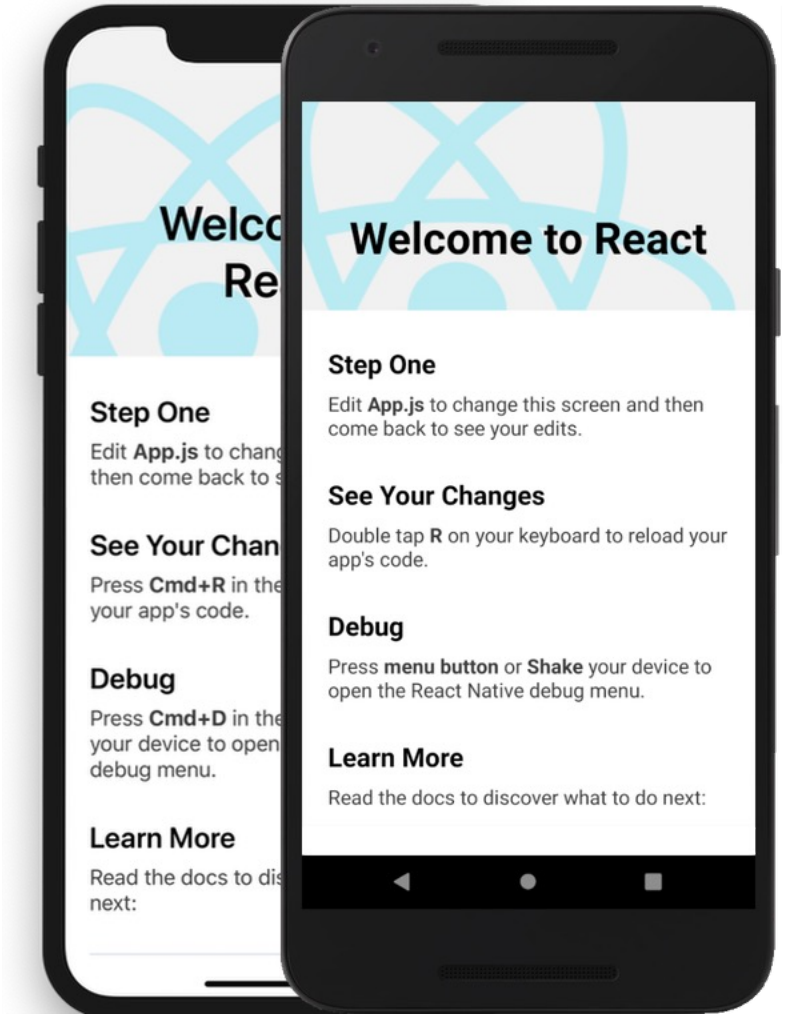
monoe18@student.sdu.dk
frhel18@student.sdu.dk

# Introduction: What is React Native?

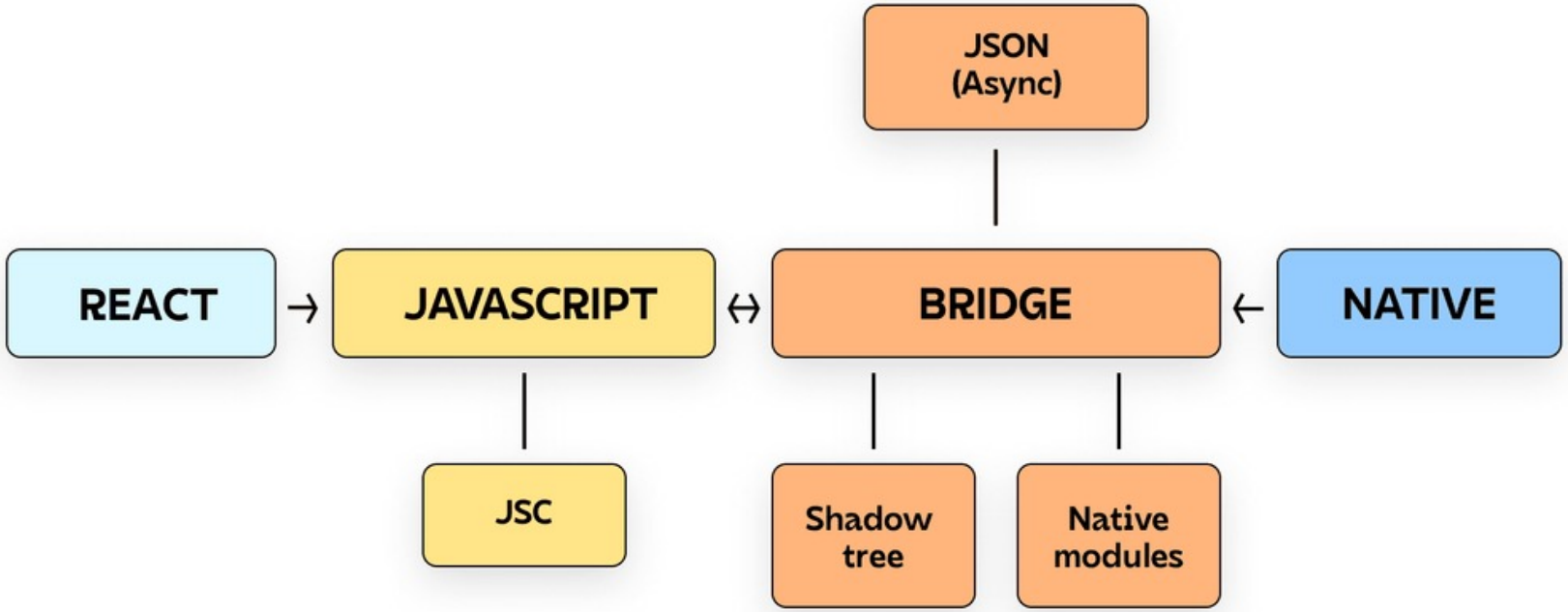
**How Meta explains React Native**

Do **more** with **less**

Write **once**, write **everywhere**



# The Bridge


JSON
(Async)
REACT
JAVASCRIPT
BRIDGE
NATIVE
JSC
Shadow
tree
Native
modules

# Build using ”Components”

**React Native** is a mobile app development framework that enables the development of multi-platform Android and iOS apps using native UI elements.

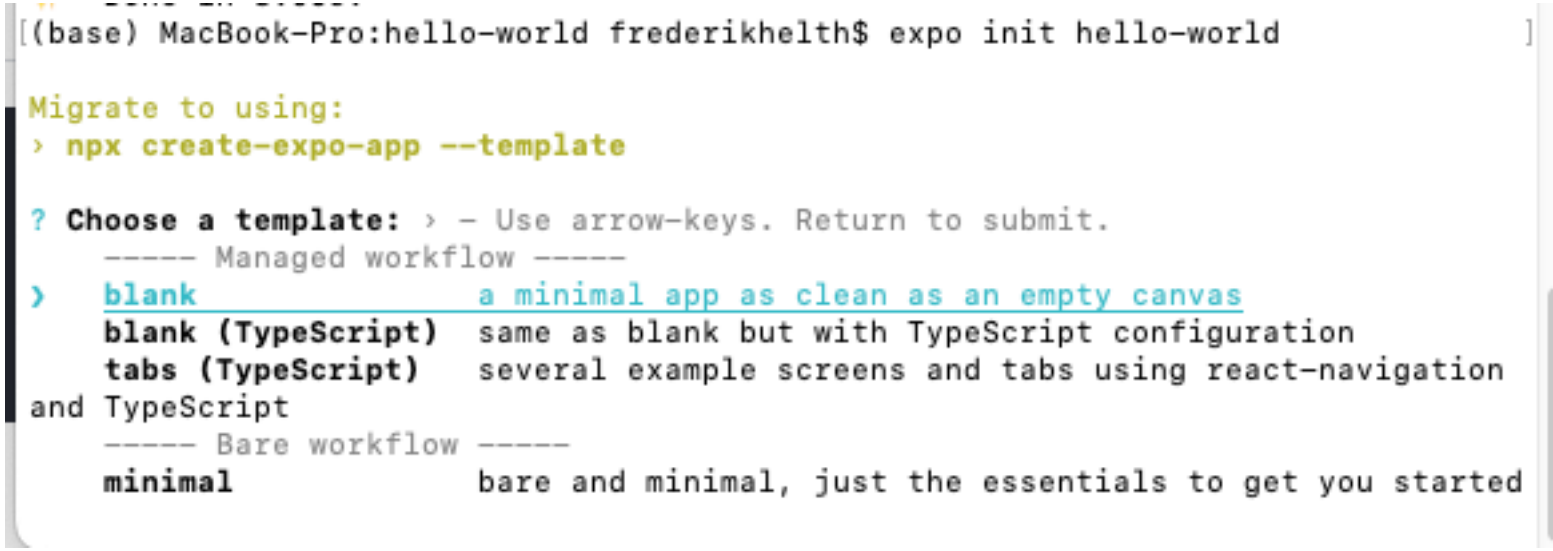
# Base components

Default components out of the box. Use them to create **custom components**

| REACT NATIVE UI COMPONENT | ANDROID VIEW | IOS VIEW | WEB ANALOG | DESCRIPTION |
|---|---|---|---|---|
| `<View>` | `<ViewGroup>` | `<UIView>` | A non-scrollling `<div>` | A container that supports layout with flexbox, style, some touch handling, and accessibility controls |
| `<Text>` | `<TextView>` | `<UITextView>` | `<p>` | Displays, styles, and nests strings of text and even handles touch events |
| `<Image>` | `<ImageView>` | `<UIImageView>` | `<img>` | Displays different types of images |
| `<ScrollView>` | `<ScrollView>` | `<UIScrollView>` | `<div>` | A generic scrolling container that can contain multiple components and views |
| `<TextInput>` | `<EditText>` | `<UITextField>` | `<input type="text">` | Allows the user to enter text |

# 10 - 15 min Exercise: Create hello world app

1. Go to: https://reactnative.dev/docs/environment-setup
2. Install expo-cli
3. Create a mobile development folder
4. Run expo init AwesomeProject
5. Choose blank
6. Run start



# Our Hello world app

**What you should see**

```
import { StatusBar } from 'expo-status-bar';
import { StyleSheet, Text, View } from 'react-native';

export default function App() {
  return (
    <View style={styles.container}>
      <Text>Open up App.js to start working on your app!</Text>
      <StatusBar style="auto" />
    </View>
  );
}

const styles = StyleSheet.create({
  container: {
    flex: 1,
    backgroundColor: '#fff',
    alignItems: 'center',
    justifyContent: 'center',
  },
});
```

# React Native components

- Export and import
- return
- Styling
- Props
- States
- Lifecycle
  - UseEffect

# Export and import

How does React Native find your component?
Which components is allowed to be used by others?

1. Imports is placed in the top of our file, to import components needed for our view.
2. The export annotation is used before a function to define if a component should be exported from a file
3. Add the default annotation after export, to tell the import which component will be the default one when imported

```
import MyComponent, { NoneDefaultCompnent } from "./MyComponent";
```

```
export default function MyComponent() {
  return (
    <View style={styles.container}>
      <Text />
    </View>
  );
}
```

# return

Contains JSX that allows us to use XML like structure to build the visuals of our app!

```
export default function MyComponent() {
  return (
    <View style={styles.container}>
      <Text />
    </View>
  );
}
```

# Styling components

To change the visuels, we use StyleSheet.create({}) object.

The object follows the structure of normal CSS

Create and object, like ”styles” from the image to right, and apply the CSS rules to any base compoentn

```
import { StatusBar } from 'expo-status-bar';
import { StyleSheet, Text, View } from 'react-native';

export default function App() {
  return (
    <View style={styles.container}>
      <Text>Open up App.js to start working on your app!</Text>
      <StatusBar style="auto" />
    </View>
  );
}

const styles = StyleSheet.create({
  container: {
    flex: 1,
    backgroundColor: '#fff',
    alignItems: 'center',
    justifyContent: 'center',
  },
});
```

# Props

Props allows us to pass data into a child component

When you develop a component, you can allow parent components to change ”some” defined variables output. Maybe you want to develop a button, that takes a onClick function, and allows the parent component to decide what to happens when a click occurs

**You can define props in two ways:**

```
export default function MyComponent({text, onClick}) {

  return (
    <View style={styles.container}>
      <Text onClick={onClick}>{text}</Text>
    </View>
  );
}
```

```
export default function MyComponent(props) {

  return (
    <View style={styles.container}>
      <Text onClick={props.onClick}>{props.text}</Text>
    </View>
  );
}
```

# States

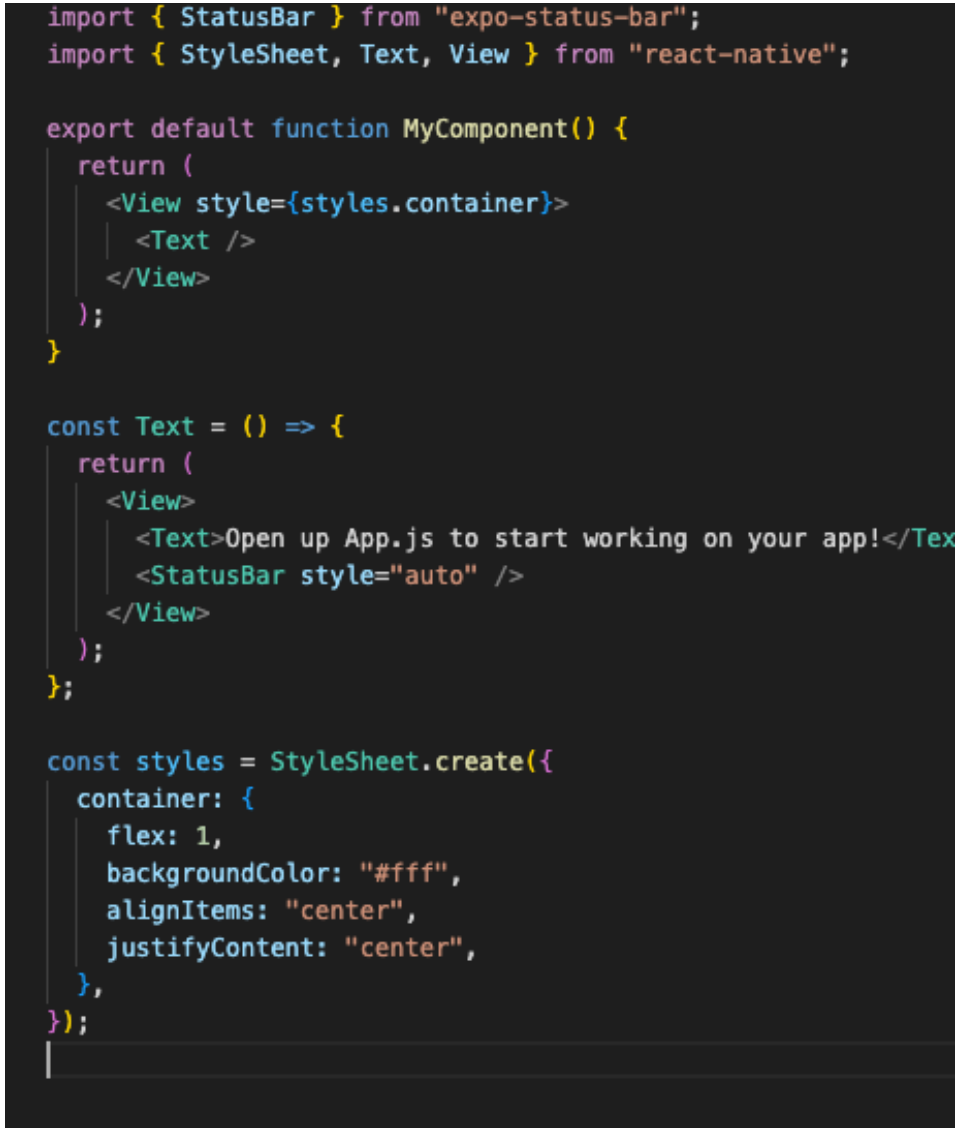
States allows us to update the component based on a new “state”

Imagen that you want to change the color of some text, when a user clicks on a button.
This is where you want to use a state!

```
export default function MyComponent() {

  const [someState, setSomeSet] = useState("defaultValue");

  function onClick(){
    setSomeSet("newValue");
  }

  return (
    <View style={styles.container}>
      <Text onClick={onClick}>{someState}</Text>
    </View>
  );
}
```

# 15 - 20 min Exercise: Create a component

1. Create a new file called ”MyComponent.js”
2. Create a export default function
3. Create a function that is implemented in the default component
4. Import the component to the App.js file
5. Extras
    1. Create properties for the component
    2. Create a state that can be changed.
    3. Play around with the StyleSheet object



```
import { StatusBar } from "expo-status-bar";
import { StyleSheet, Text, View } from "react-native";

export default function MyComponent() {
  return (
    <View style={styles.container}>
      <Text />
    </View>
  );
}

const Text = () => {
  return (
    <View>
      <Text>Open up App.js to start working on your app!</Tex
      <StatusBar style="auto" />
    </View>
  );
};

const styles = StyleSheet.create({
  container: {
    flex: 1,
    backgroundColor: "#fff",
    alignItems: "center",
    justifyContent: "center",
  },
});
```

# Component lifecycle: UseEffect

Sometimes we want to do stuff, when the component is mounted, destroyed, some property changes and so fourth

```
import { useEffect, useState } from "react";
```

```
export default function MyComponent() {
  useEffect(() => {
    console.log("I im use!");

    return function () {
      console.log("Okay, bye then..");
    };
  });

  return (
    <View style={styles.container}>
      <Text>Hello</Text>
    </View>
  );
}
```

# Component lifecycle: UseEffect

Sometimes you want to “do something” when a property or state changes.

```
export default function MyComponent({text}) {
  useEffect(() => {
    console.log("text changed");
  }, [text]);

  return (
    <View style={styles.container}>
      <Text>{text}</Text>
    </View>
  );
}
```

# 15 - 20 min Exercise: Play around with lifecycle

1. Use your new created component
2. Import useEffect from React Native
3. Do something when the component changes
4. Do something when component gets unmounted
5. Create a property
6. Do something when the property changes

```
export default function MyComponent() {
  useEffect(() => {
    console.log("I im use!");

    return function () {
      console.log("Okay, bye then..");
    };
  });

  return (
    <View style={styles.container}>
      <Text>Hello</Text>
    </View>
  );
}
```

Let’s do a kahoot quiz!