Nieuwe technologie: React Native

# Wat is React

“Learn once, write anywhere: Build mobile apps with React”

React Native lets you build mobile apps using only JavaScript. It uses the same design as React, letting you compose a rich mobile UI from declarative components.

React Native lets you build your app faster. Instead of recompiling, you can reload your app instantly. With Hot Reloading, you can even run new code while retaining your application state. Give it a try - it's a magical experience.

Wiki:

In computing, React (sometimes styled React.js or ReactJS) is a JavaScript library for building user interfaces.

It is maintained by Facebook, Instagram and a community of individual developers and corporations.

React allows developers to create large web-applications that use data and can change over time without reloading the page. It aims primarily to provide speed, simplicity, and scalability. React processes only user interfaces in applications. This corresponds to View in the Model-View-Controller (MVC) pattern, and can be used in combination with other JavaScript libraries or frameworks in MVC, such as AngularJS.

## Basic usage:

<**div** id="myReactApp"></**div**>

<**script** type="text/babel">

**class** Greeter **extends** React.Component {

render() {

**return** <h1>{**this**.props.greeting}</h1>

}

}

ReactDOM.render(<Greeter greeting="Hello World!" />, document.getElementById('myReactApp'));

</**script**>

## Result:

<**div** id="myReactApp">

<**h1**>Hello World!</**h1**>

</**div**>

# Props

Most components can be customized when they are created, with different parameters. These creation parameters are called props.

Bij image -> prop source

Declareren met let

# View

The other new thing going on here is the View component. A View is useful as a container for other components, to help control style and layout.

# State

To learn how to make your app change over time, you need to learn about State.

There are two types of data that control a component: props and state. props are set by the parent and they are fixed throughout the lifetime of a component. For data that is going to change, we have to use state.

# Style

With React Native, you don't use a special language or syntax for defining styles. You just style your application using JavaScript. All of the core components accept a prop named style. The style names and values usually match how CSS works on the web, except names are written using camel casing, e.g backgroundColor rather than background-color.

# Height & width

A component's height and width determine its size on the screen.

## Fixed Dimensions

The simplest way to set the dimensions of a component is by adding a fixed width and height to style. All dimensions in React Native are unitless, and represent density-independent pixels.

## Flex Dimensions

Use flex in a component's style to have the component expand and shrink dynamically based on available space. Normally you will use flex: 1, which tells a component to fill all available space, shared evenly amongst each other component with the same parent. The larger the flex given, the higher the ratio of space a component will take compared to its siblings.

A component can only expand to fill available space if its parent has dimensions greater than 0. If a parent does not have either a fixed width and height or flex, the parent will have dimensions of 0 and the flex children will not be visible.

# Layout with Flexbox

A component can specify the layout of its children using the flexbox algorithm. Flexbox is designed to provide a consistent layout on different screen sizes.

You will normally use a combination of flexDirection, alignItems, and justifyContent to achieve the right layout.

Flexbox works the same way in React Native as it does in CSS on the web, with a few exceptions. The defaults are different, with flexDirection defaulting to column instead of row, and the flex parameter only supporting a single number.

## Flex Direction

Adding flexDirection to a component's style determines the primary axis of its layout. Should the children be organized horizontally (row) or vertically (column)? The default is column.

## Justify Content

Adding justifyContent to a component's style determines the distribution of children along the primary axis. Should children be distributed at the start, the center, the end, or spaced evenly? Available options are flex-start, center, flex-end, space-around, and space-between.

## Align Items

Adding alignItems to a component's style determines the alignment of children along the secondary axis (if the primary axis is row, then the secondary is column, and vice versa). Should children be aligned at the start, the center, the end, or stretched to fill? Available options are flex-start, center, flex-end, and stretch.

For stretch to have an effect, children must not have a fixed dimension along the secondary axis. In the following example, setting alignItems: stretch does nothing until the width: 50 is removed from the children.

# Handling Text Input

TextInput is a basic component that allows the user to enter text. It has an onChangeText prop that takes a function to be called every time the text changed, and an onSubmitEditing prop that takes a function to be called when the text is submitted.

For example, let's say that as the user types, you're translating their words into a different language. In this new language, every single word is written the same way: 🍕. So the sentence "Hello there Bob" would be translated as "🍕🍕🍕".

# Handling Touches

Users interact with mobile apps mainly through touch. They can use a combination of gestures, such as tapping on a button, scrolling a list, or zooming on a map.

If the basic button doesn't look right for your app, you can build your own button using any of the "Touchable" components provided by React Native. The "Touchable" components provide the capability to capture tapping gestures, and can display feedback when a gesture is recognized. These components do not provide any default styling, however, so you will need to do a bit of work to get them looking nicely in your app.

•Generally, you can use TouchableHighlight anywhere you would use a button or link on web. The view's background will be darkened when the user presses down on the button.

•You may consider using TouchableNativeFeedback on Android to display ink surface reaction ripples that respond to the user's touch.

•TouchableOpacity can be used to provide feedback by reducing the opacity of the button, allowing the background to be seen through while the user is pressing down.

•If you need to handle a tap gesture but you don't want any feedback to be displayed, use TouchableWithoutFeedback.

# ScrollView

The ScrollView is a generic scrolling container that can host multiple components and views. The scrollable items need not be homogenous, and you can scroll both vertically and horizontally (by setting the horizontal property).

# List Views

React Native provides a suite of components for presenting lists of data. Generally, you'll want to use either FlatList or SectionList.

The FlatList component displays a scrolling list of changing, but similarly structured, data. FlatList works well for long lists of data, where the number of items might change over time. Unlike the more generic ScrollView, the FlatList only renders elements that are currently showing on the screen, not all the elements at once.

The FlatList component requires two props: data and renderItem. data is the source of information for the list. renderItem takes one item from the source and returns a formatted component to render.

If you want to render a set of data broken into logical sections, maybe with section headers, similar to UITableViews on iOS, then a SectionList is the way to go.

# Networking

Many mobile apps need to load resources from a remote URL. You may want to make a POST request to a REST API, or you may simply need to fetch a chunk of static content from another server.

## Fetch

React Native provides the Fetch API for your networking needs. Fetch will seem familiar if you have used XMLHttpRequest or other networking APIs before. You may refer to MDN's guide on Using Fetch for additional information.

In order to fetch content from an arbitrary URL, just pass the URL to fetch:

fetch('https://mywebsite.com/mydata.json')

Networking is an inherently asynchronous operation. Fetch methods will return a Promise that makes it straightforward to write code that works in an asynchronous manner.