

Web natívan

JS a mobilon

```
const speakers = [ 'Tamás Csaba', 'Kálóczi Dávid' ];
```

Before we start

install tools 

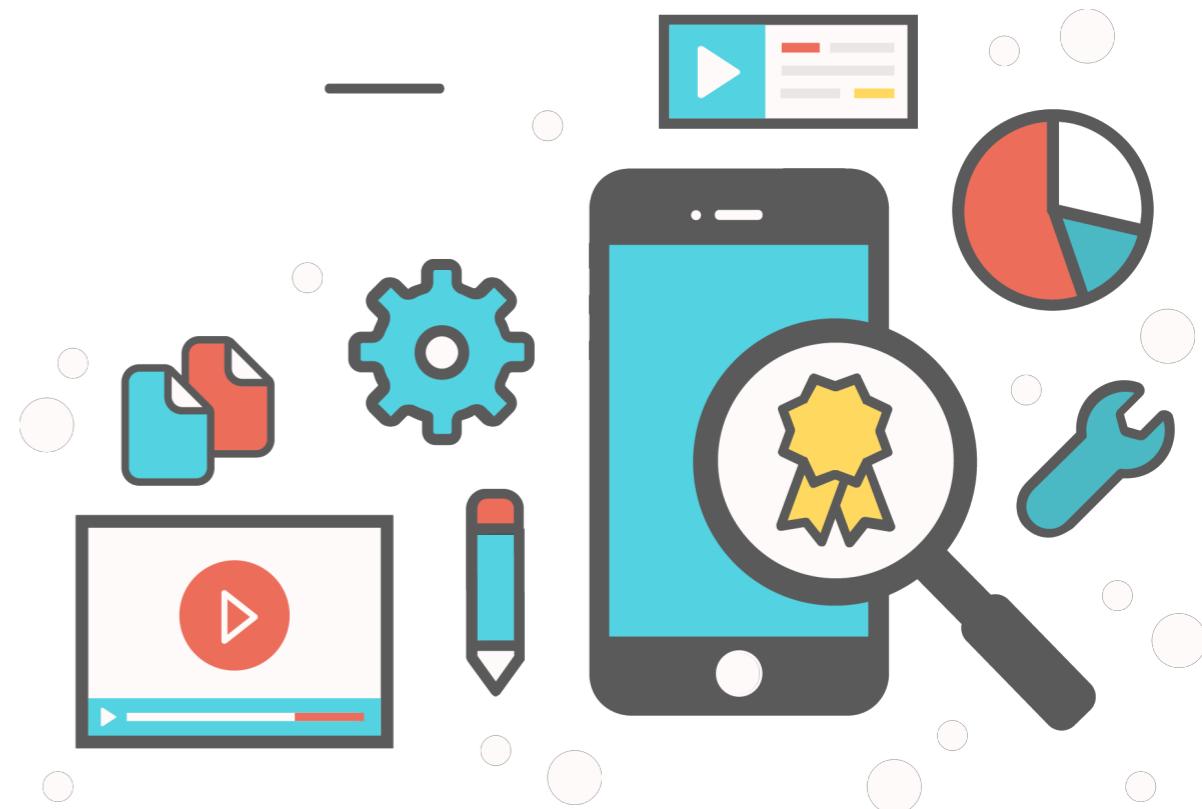
- node and npm
- npm i -g expo-cli
- Android/iOS: nativescript (playground, preview), expo

<https://tlk.io/openacademy>

Today

Today

- iOS and Android dev.
- Sync UI
- Implement BL twice





“You only need an iframe with a responsive website.”

-voice in your head

Browsers!

- browsers evolved significantly
- You only need an iframe with a responsive website.
- have so many advanced APIs
- let's call them PWAs

PWA



Browsers?

- Its okay, but I need
 - Camera
 - Push Notifications
 - MicroTransactions
 - NFC support
 - a new Ferrari!

Pimp up browsers

- **WebView, iframe**
 - “`navigator.camera`”
 - let's call it ApacheCordova



Goodn't

Goodn't

- “Write once, run everywhere”
- It still a web app
- does not really adapts the host

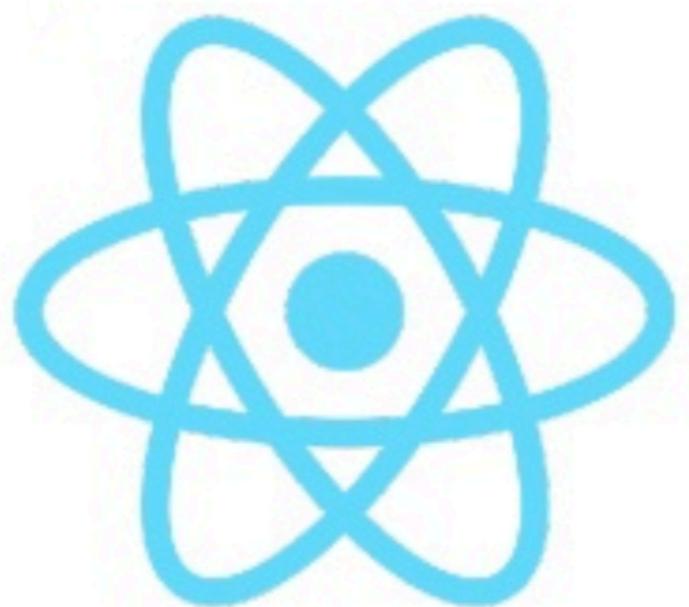
“Learn once, use it everywhere.”

different approach

Better?

- Learn a framework
- Pimp the framework
- JS of course!
- Let's call it...





React Native
(by Facebook)

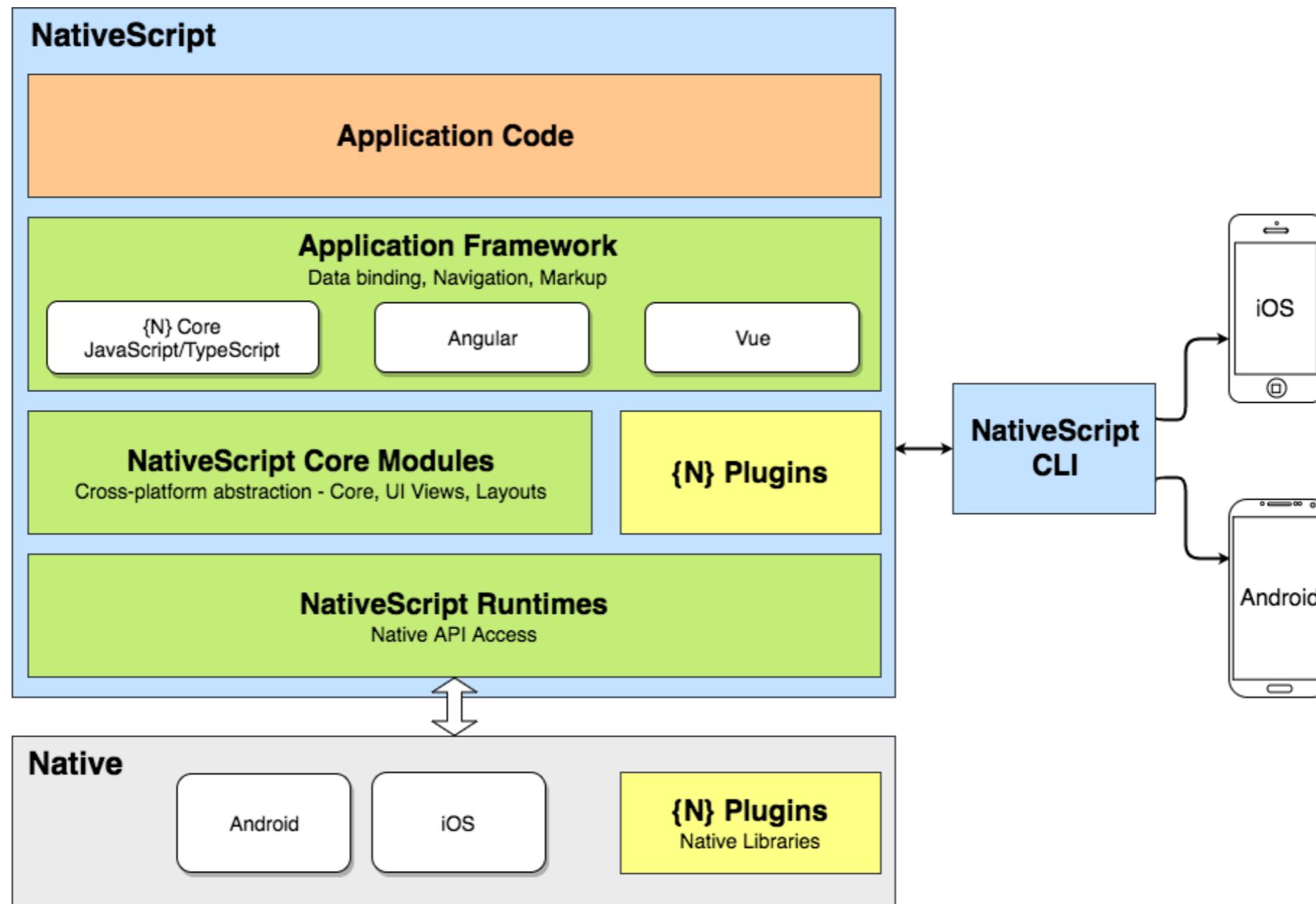


NativeScript
(by Telerik)

NativeScript



Architecture



Android and iOS Examples

```
var view1 = UIView.alloc().init();  
// Or with the short-cut  
var view2 = UIView.new();
```

```
UIView *view1 = [[UIView alloc] init];  
// Or with the short-cut  
UIView *view2 = [UIView new];
```

```
var myObject = java.lang.Object.extend({  
    hashCode: function(){  
        return 10;  
    }  
});
```

```
public class MyObject {  
    @Override  
    public int hashCode(){  
        return 10;  
    }  
}
```

Modules

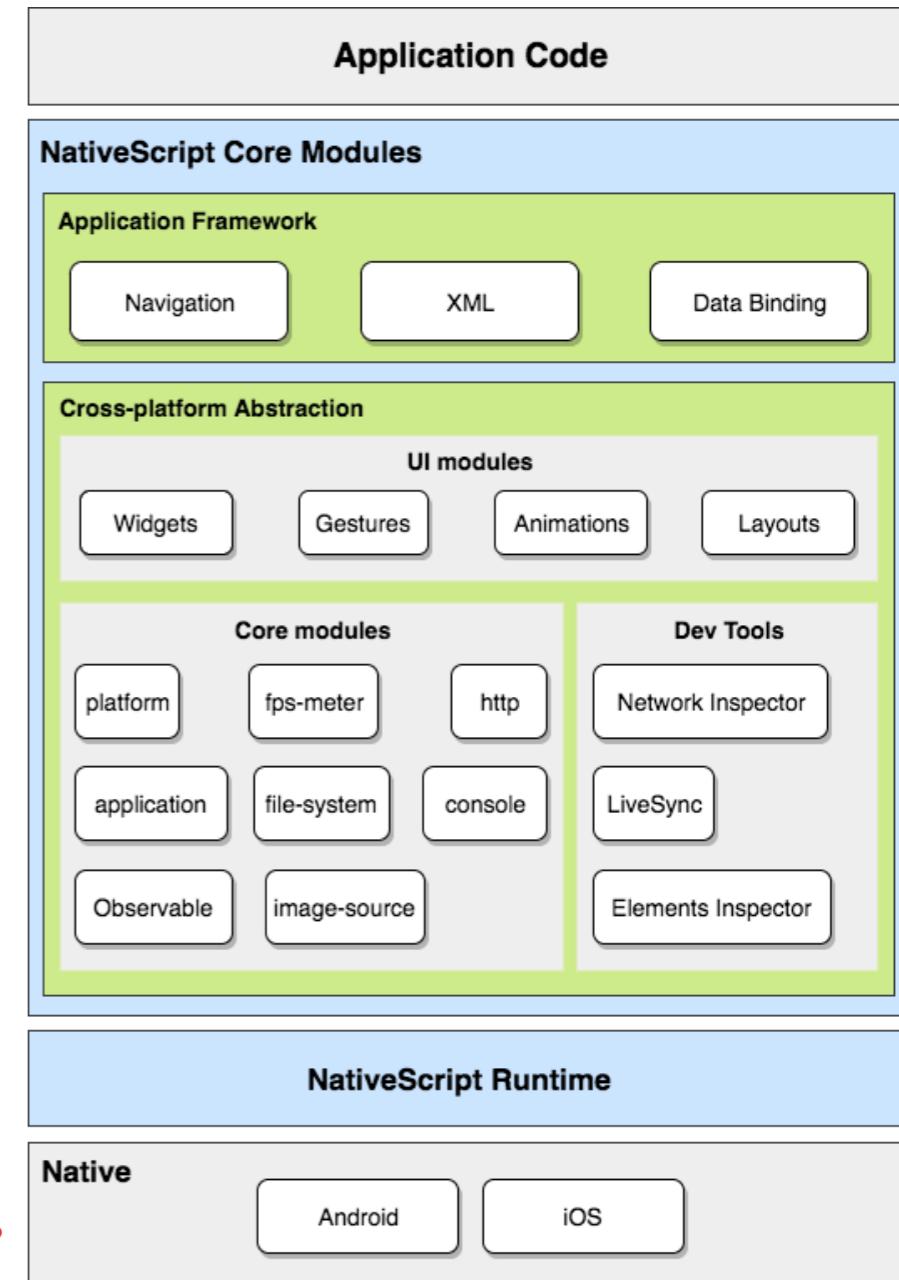
```
var fileSystemModule = require( "file-system" );
new fileSystemModule.File( path );
```



```
new java.io.File( path );
```



```
NSFileManager.defaultManager();
fileManager.createFileAtPathContentsAttributes(path);
```



Plugins

```
tns plugin add nativescript-accelerometer
```

Accelerometer plugin for NativeScript

The screenshot shows a GitHub repository page for the 'nativescript-accelerometer' plugin. At the top, there's a summary bar with 30 commits, 1 branch, and 0 releases. Below it, a 'Branch: master' dropdown and a 'New pull request' button are visible. The main area lists commit details for each of the 30 commits, showing the author (vakrilov), the file changes, and a brief description of the commit.

Commit Details	Description
vakrilov FIX: IOS updates do not work when done in the update callback	FIX: IOS updates do not work when done in the update callback
demo	Added sensor delay option and version bump.
.gitignore	Added sensor delay option and version bump.
.npmignore	update to {N} 2 + demo
LICENSE	Initial commit
README.md	Added sensor delay option and version bump.
index.android.ts	Added sensor delay option and version bump.
index.d.ts	Added sensor delay option and version bump.
index.ios.ts	FIX: IOS updates do not work when done in the update callback
package.json	FIX: IOS updates do not work when done in the update callback
tsconfig.json	Updated the plugin to {N}3.0

- npm packages
- native functionality

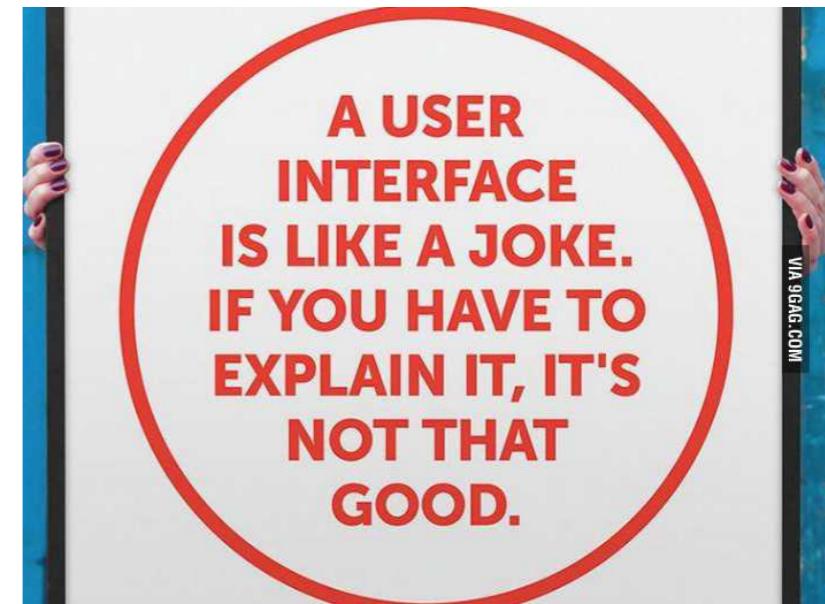
```
var acc = require("nativescript-accelerometer");

acc
  .startAccelerometerUpdates(function(data) {
    console.log(data.x, data.y, data.z);
}, { sensorDelay: "ui" });
```

Application code

What about the UI

- Use JS/TS and XML
- Simple data-binding
- Use shared or specific UI
- Style with CSS/SCSS/LESS



```
<StackLayout>
    <Button text="Test Button For Binding" tap="{{ onTap }}" />
    <TextField text="{{ textSource | dateConverter('YYYY') }}" />
</StackLayout>
```

Layouts

LEARN {N} LAYOUTS

Get ready to Learn NativeScript Layouts! Inspired by [Flexbox Froggy](#), this tool is a fun and easy way for you to learn about mobile app layouts with NativeScript. Each level presents a different mobile app view that needs native UI elements laid out a certain way. NativeScript provides a variety of layout tags to help with this:

- `<StackLayout>` stacks components on top of [or next to] each other;
- `<WrapLayout>` uses available space to wrap components on new rows/columns;
- `<AbsoluteLayout>` places components with absolute top/left coordinates;
- `<GridLayout>` acts like an HTML table, with rows and columns;
- `<DockLayout>` positions components in docked containers [top/bottom/left/right];
- `<FlexboxLayout>` lets you leverage CSS flexbox.

Let's start with a `<StackLayout>`. By default, a `<StackLayout>` will stack components vertically. Change the code below to make it so the UI components align each other, in a *horizontal* orientation 😊.

```
1 <StackLayout orientation="vertical">
2   <Image src="res://nativescript" stretch="none"></Image>
3   <Image src="res://angular" stretch="none"></Image>
4   <Image src="res://vue" stretch="none"></Image>
5 </StackLayout>
6
7
8
9
10
```

[Help!](#) [Next](#)

Try NativeScript in Your Browser: [Playground](#)

Find NativeScript on [GitHub](#) • [Twitter](#) • [Slack](#) • [Stack Overflow](#)



Playground

The screenshot shows the NativeScript Playground interface. At the top, there's a browser-like header with a back/forward button, refresh, home, and search fields (https://play.nativescript.org). Below the header is a dark navigation bar with the 'Playground' logo, 'Editor' tab (which is active), 'Get started', 'Feedback', and 'Log In' buttons.

The main area has several tabs: 'My Playground' (selected), 'New', 'QR code', 'Save', 'Fork', and 'Download'. On the far right is a 'Preview' button.

On the left, there's an 'Explorer' sidebar showing a project structure under 'home': home-routing.module.ts, home.component.css, home.component.html, home.component.ts*, and home.module.ts. There's also a main.ts file. A '+' icon is at the bottom of the sidebar.

The central part of the screen displays the 'home.component.ts' file content:

```
1 import { Component, OnInit } from "@angular/core";
2
3 @Component({
4   selector: "Home",
5   moduleId: module.id,
6   templateUrl: "./home.component.html",
7   styleUrls: ['./home.component.css']
8 })
9 export class HomeComponent {
10
11   constructor() {
12
13
14 }
```

Below the code editor is a 'Components' sidebar with a search icon. It lists 'ESSENTIALS' components: Button, Label, Image, and Slider, each with a preview icon.

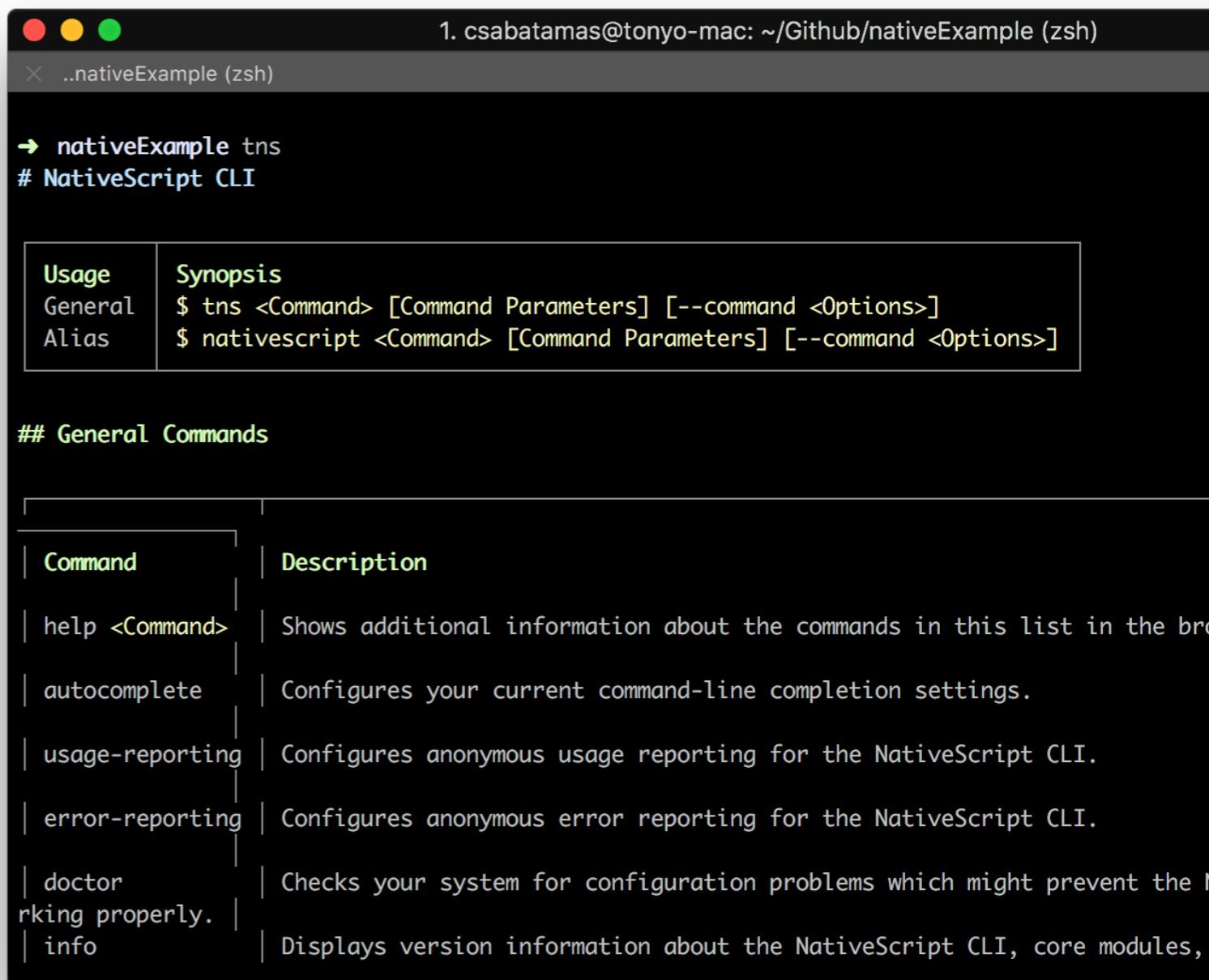
At the bottom right is a 'Devices' panel with tabs for 'Errors', 'Device Logs', and 'Devices'. It shows a single device entry:

Device Name	Model	OS version	Preview app version	Runtime version
▶ iOS Csaba's iPhone	iPhone 8	iOS 12.0.1	1.17.0	5.0.0

CLI

npm install nativescript

- **tns create my-app-name**
- **tns preview –bundle**
- **tns doctor**
- **tns build ios –release**



1. csabatamas@tonyo-mac: ~/Github/nativeExample (zsh)

```
..nativeExample (zsh)
→ nativeExample tns
# NativeScript CLI
```

Usage	Synopsis
General	\$ tns <Command> [Command Parameters] [--command <Options>]
Alias	\$ nativescript <Command> [Command Parameters] [--command <Options>]

```
## General Commands
```

Command	Description
help <Command>	Shows additional information about the commands in this list in the browser.
autocomplete	Configures your current command-line completion settings.
usage-reporting	Configures anonymous usage reporting for the NativeScript CLI.
error-reporting	Configures anonymous error reporting for the NativeScript CLI.
doctor	Checks your system for configuration problems which might prevent the application from working properly.
info	Displays version information about the NativeScript CLI, core modules, and plugins.

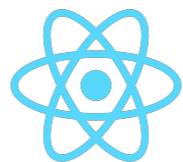
Sidekick

The screenshot shows the NativeScript Sidekick application interface. At the top, there's a dark header bar with the title "NativeScript Sidekick". Below it is a toolbar with several icons: "Back to My Apps", "Show Files", "Terminal", "Open in Editor" (which is highlighted in blue), "New Page", "Build", "Send Feedback", "Getting Started", "Csaba", and "Publish". On the left side, there's a sidebar with icons for "Devices" (1 device connected), "Properties", "Plugins", "Assets", and "Services". The main workspace has two main sections: "CONNECTED DEVICES" and "BUILD SETTINGS". Under "CONNECTED DEVICES", there's a list with "iPhoneX (Emulator)" selected. Under "BUILD SETTINGS", there are options for "Build Type" (Local is selected over Cloud), "Build Configuration" (Debug is selected over Release), and checkboxes for "Webpack" (checked), "Hot module replacement" (unchecked, marked as BETA), and "Start Debugger". At the bottom right of the workspace is a large blue button labeled "Run on Device". Below the workspace is a "Device Console" window showing log output. The log output includes:

```
[18-12-06 14:22:58.785] (CLI) Using NativeScript CLI located in /Users/csabatamas/n/lib/node_modules/nativescript
[18-12-06 14:23:00.910] (CLI) Searching for devices...
[18-12-06 14:23:01.278] Loaded CLI extension nativescript-cloud, version 1.14.4.
[18-12-06 14:23:01.281] Loaded CLI extension nativescript-starter-kits, version 0.3.5.
[18-12-06 14:23:08.014] Devices service initialized.
[18-12-06 14:23:09.712] The user Csaba Tamas logged in.
[18-12-06 14:23:19.929] App with path: /Users/csabatamas/Projects/nativescriptExample was opened.
[18-12-06 14:23:30.950] Virtual device iPhoneX was started.
```

At the very bottom right of the interface, there's a small "OUTPUT" tab.

React Native



React

- UI Library for web by Facebook
- component based
- JSX

```
import React, { Component } from 'react';

export default class MyComponent extends Component {
  render () {
    return <div> Hello </div>;
  }
}
```

React Native

- “learn once, use it everywhere”
- writing Mobile app is the same as Web apps
- its a thing

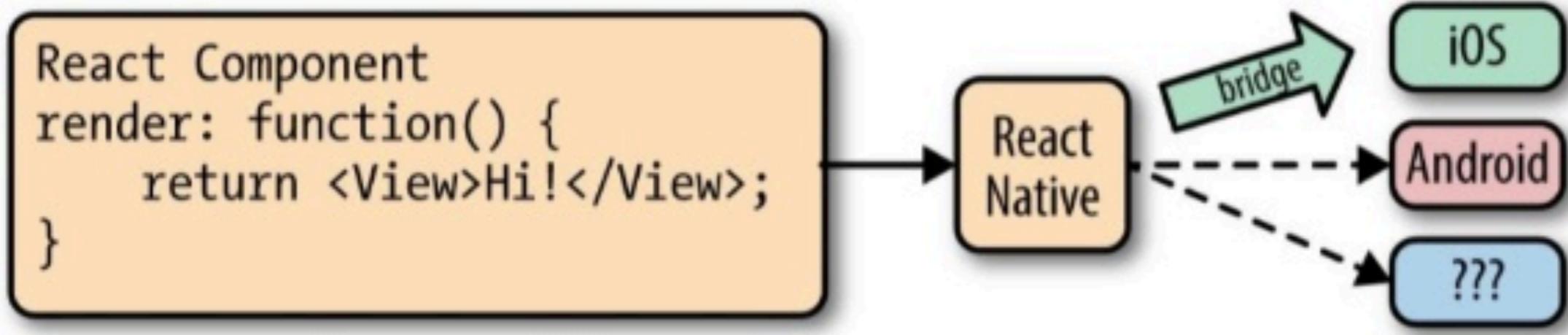
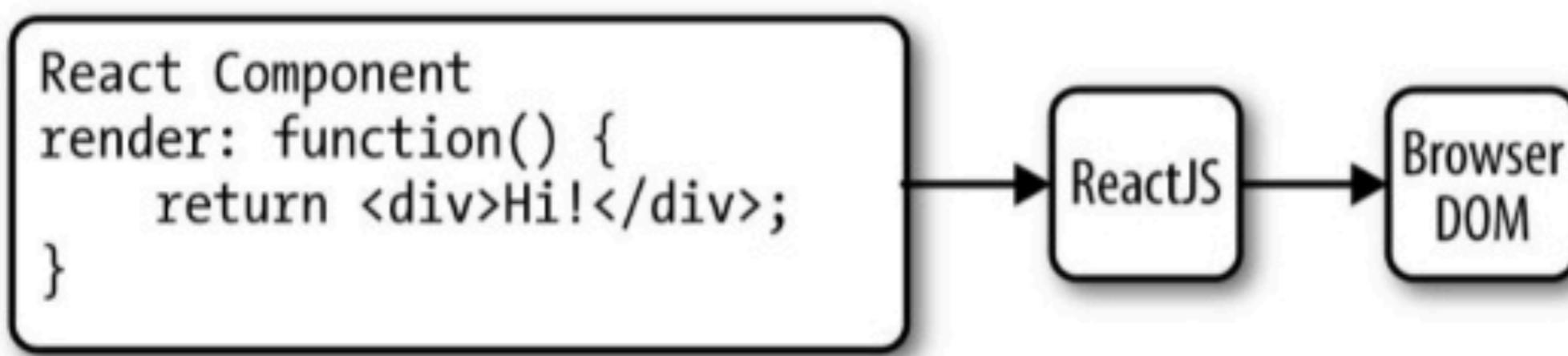


React Native

- The same React, just for mobile OS
- that's it?

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

export default class MyComponent extends Component {
  render () {
    return <Text> Hello </Text>;
  }
}
```

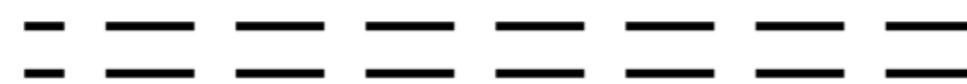


Bridge

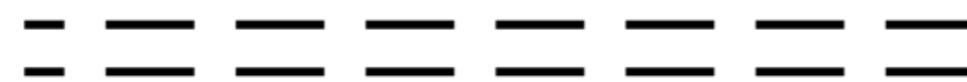


Native Modules

Andorid - Java
IOS - Obj C /Swift



RN Bridge
(Java/C++)



JS

Virtual Machine
(JavaScriptCore)

not for everything

- Events are on the JS layer, view is on the other side of the bridge
- many native calls can overwhelm the bridge
 - it the bottleneck 
- you can hurt yourself



setting up RN dev.

- `npm i react-native` 
- `react-native init myApp` 
- install xcode  / Android Studio 
 - build for your platform 
 - manage codesigns and so.. 
 - deploy 

expo

- provides a shell and platform for jsbundles
- makes React Native development easier
 - restrictions
 - only need a single install, and a companion app.

Let's build
okay, cool
Let's help Santa

