React Native/Expo
Electron

PV247

Denis Slávik slavik@inqool.cz
Mikuláš Ponechal ponechal@inqool.cz

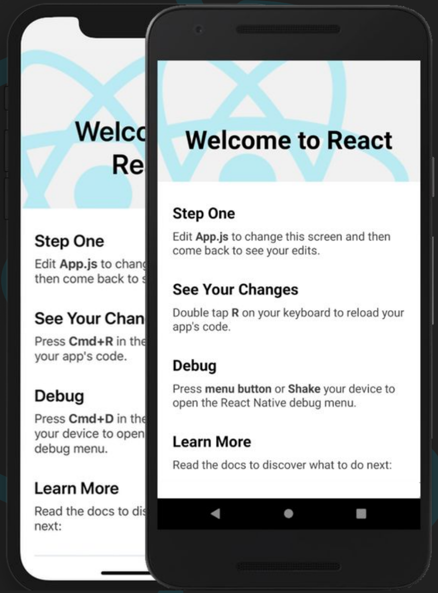
inQool a.s.

Today’s topics

- React Native
- Expo
- What is Electron
- Quick start guide with Electron
- Examples

React Native

- Create native apps for Android and iOS using React
- You don't build a mobile web app, an HTML5 app, or a hybrid app
- You just put those building blocks together using JavaScript and React



Welcome to React

Step One

Edit **App.js** to change this screen and then come back to see your edits.

See Your Changes

Double tap **R** on your keyboard to reload your app's code.

Debug

Press **menu button** or **Shake** your device to open the React Native debug menu.

Learn More

Read the docs to discover what to do next:

Welco
Re

Step One

Edit **App.js** to chang
then come back to s

See Your Chan

Press **Cmd+R** in the
your app's code.

Debug

Press **Cmd+D** in the
your device to open
debug menu.

Learn More

Read the docs to dis
next:

React Native - Code

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

const App = () => {
    return (
        <View style = {styles.container} >
          <Text>Open up App.js to start working on your app! </Text>
          <Text>Changes you make will automatically reload. </Text>
          <Text>Shake your phone to open the developer menu. </Text>
        </View>
    );
};

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

export default App;
```

React Native - Core Components and APIs

- Basic Components
  - View, Text, Image, TextInput, ScrollView, StyleSheet
- User Interface
  - Button, Switch
- List Views
  - FlatList, SectionList
- Android Components and APIs
  - BackHandler, DrawerLayoutAndroid, PermissionsAndroid, ToastAndroid
- iOS Components and APIs
  - ActionSheetIOS, SafeAreaView
- Others
  - ActivityIndicator, Alert, Animated, Dimensions, KeyboardAvoidingView, Linking, Modal, PixelRatio, RefreshControl, StatusBar

React Native - Advantages

- You can use the existing JavaScript knowledge to build native mobile apps
- Multi Platform − Android, Android TV, iOS, macOS, tvOS, Web, Windows and UWP
- Code sharing − You can share most of your code on different platforms
- Community − The community around React and React Native is large, and you will be able to find any answer you need

React Native - Limitations

- Native Components − If you want to create native functionality which is not created yet, you will need to write some platform specific code

Expo

- The “create-react-app” of React Native
- Expo is built around React Native leverages native platforms to provide a set of tools and services for developers
- React Native environment, but without the need of native configuration
- 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

Expo - Advantages

- No need to setup development environment (Xcode/Android Studio)
- No linking of additional native libraries for each feature needed in each environment needed
- Can manage assets, handle cross-platform push notifications, and publish updates to app store
- Can be updated Over The Air (OTA), without involving app stores at all
- Web editor https://snack.expo.io/

Expo - Limitations

- Not all iOS and Android APIs are available
- Native libraries to integrate with proprietary services are usually not included in the SDK
- Free builds can sometimes be queued

Who is using React Native?

- Facebook
- Instagram
- Oculus
- Discord
- Skype
- Tesla
- Pinterest
- Uber Eats
- Etc. https://reactnative.dev/showcase

What is Electron.js

- framework for creating **desktop** application using HTML, JS, CSS
- created apps can be packaged for macOS, Windows, Linux
- Electron consist of:
  - **Chromium,**
  - **Node.js,**
  - **Custom APIs**

Main and Renderer processes

Main process:

- **creates** web pages by creating **`BrowserWindow`** instances,
- each instance runs the web page in its **Renderer process**.

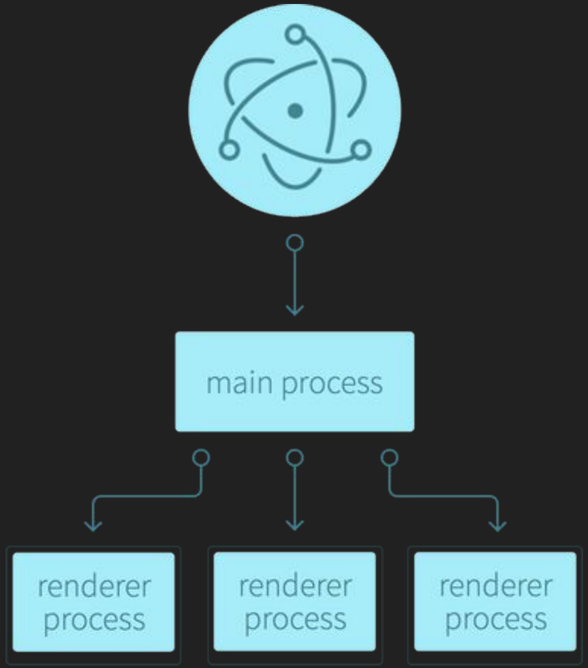
Renderer process:

- manages **only corresponding** web page,
- crash in one renderer process **does not** affect other renderer processes.

Communication between processes is possible via IPC modules:

1. **_ipcMain**
2. **_ipcRenderer**

Main and Renderer processes



Source: https://dzone.com/articles/building-a-desktop-application-with-electron

Quickstart guide

Prerequisites: Node.js

Minimal Electron app structure:

```
my-electron-app/
├── package.json
├── main.js
└── index.html
```

Quickstart guide

Steps for creating your basic app would be:

1. Create project folder and install **Electron**

   ```
   npm init -y
   npm i --save-dev electron
   ```

2. Create **main.js** (main script file)
3. Create your web page (**index.html)**
4. Modify **package.json** file
5. Run your app

Quickstart guide - main.js

```
const { app, BrowserWindow } = require('electron');
function createWindow() {
 const win = new BrowserWindow({
   width: 800,
   height: 600,
   webPreferences: {
     nodeIntegration: true,
   },
 });
 win.loadFile('index.html');
}

app.whenReady().then(createWindow);
app.on('window-all-closed', () => {
 if (process.platform !== 'darwin') {
   app.quit();
 }
});

app.on('activate', () => {
 if (BrowserWindow.getAllWindows().length === 0) {
   createWindow();
 }
});
```

Quickstart guide - index.html

```html
<!DOCTYPE html>
<html>
 <head>
   <meta charset="UTF-8">
   <title>Hello World!</title>
   <meta http-equiv="Content-Security-Policy" content="script-src 'self' 'unsafe-inline';" />
 </head>
 <body style="background: white;">
   <h1>Hello World!</h1>
   <p>
       We are using node <script>document.write(process.versions.node)</script>,
       Chrome <script>document.write(process.versions.chrome)</script>,
       and Electron <script>document.write(process.versions.electron)</script>.
   </p>
 </body>
</html>
```

Quickstart guide - package.json

```
{
   name: 'my-electron-app',
   version: '0.1.0',
   main: 'main.js',
   scripts: {
     start: 'electron .',
   },
};
```

Then run **`npm start`**.

Examples

Applications built with Electron:

- VS Code,
- FB messenger,
- WhatsApp,
- Twitch,
- Slack,
- Skype,
- … many more.

Another way of using react

INK - https://github.com/vadimdemedes/ink

- component UI building using REACT but for CLI applications,
- it is a React renderer

Useful links

- Electron - Whole documentation: https://www.electronjs.org/docs,
- Electron - Quick start guide: https://www.electronjs.org/docs/tutorial/quick-start
- React Native - https://reactnative.dev/
- Expo - https://docs.expo.io/
- React Native Windows - https://microsoft.github.io/react-native-windows/