Getting started with Tabris.js 3.1.0

Tutorial Ebook

EclipseSource

Table of contents

Introduction

Tabris.js is a framework for developing mobile apps in JavaScript.

With Tabris.js you can develop native iOS, Android and Windows apps with a single code base. The code is written entirely in JavaScript. So you don't have to manage code for different platforms individually.

Tabris.js gives you native performance and native look & feel. And you can leverage your existing JavaScript know-how.

In this ebook, you learn how to **get started** with Tabris.js, how to **create your first app**, and how to **build your app**.

1 Get started

Get started with Tabris.js

To **get started** with Tabris.js you only need a mobile device and the Tabris.js Developer App!
You can write your code right away by using the playground on playground.tabris.com.

Set up your mobile device

To set up your **mobile device**, follow these steps:

1. Download the Tabris.js Developer App from the Google Play Store or the Apple App Store. They are available for free.

   Follow the links below or search for Tabris.js in the store on your mobile device.

   GET IT ON Google play

   Download on the App Store

2. Start the app.

2 Tabris.js in action

Execute code on your mobile device

The Tabris.js Developer App can execute JavaScript code directly on your mobile device. The code can be loaded from a remote location (for example your development machine or the online playground).

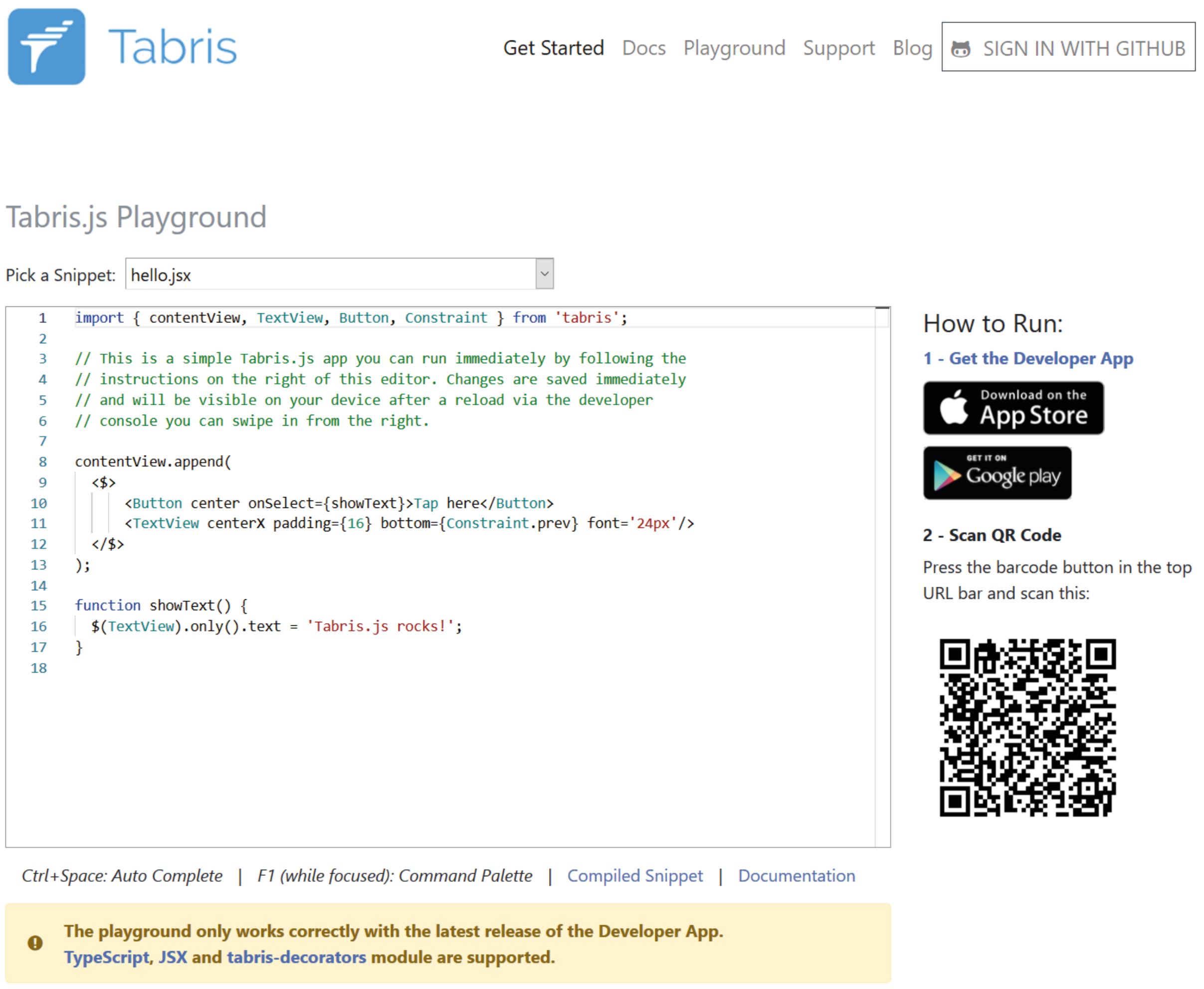
The online playground

A very easy way to write and run your first own code is using the online playground on https://playground.tabris.com/. Here you will find simple Tabris.js scripts (snippets) demonstrating various Tabris.js features. The scripts can be modified as desired. The initially shown snippet is "Hello, World!".

You can run this script immediately in the Tabris.js Developer App on your mobile device.

Please see the screenshot below.

The script in the online playground



Tabris.js Playground

Pick a Snippet: hello.jsx

```
import { contentView, TextView, Button, Constraint } from 'tabris';

// This is a simple Tabris.js app you can run immediately by following the
// instructions on the right of this editor. Changes are saved immediately
// and will be visible on your device after a reload via the developer
// console you can swipe in from the right.

contentView.append(
  <$>
      <Button center onSelect={showText}>Tap here</Button>
      <TextView centerX padding={16} bottom={Constraint.prev} font='24px'/>
  </$>
);

function showText() {
  $(TextView).only().text = 'Tabris.js rocks!';
}
```

How to Run:

1 - Get the Developer App

2 - Scan QR Code

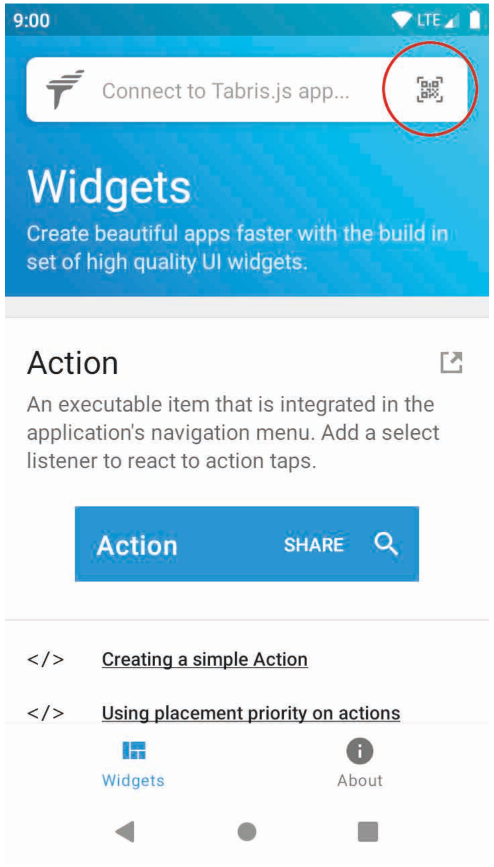
Press the barcode button in the top URL bar and scan this:

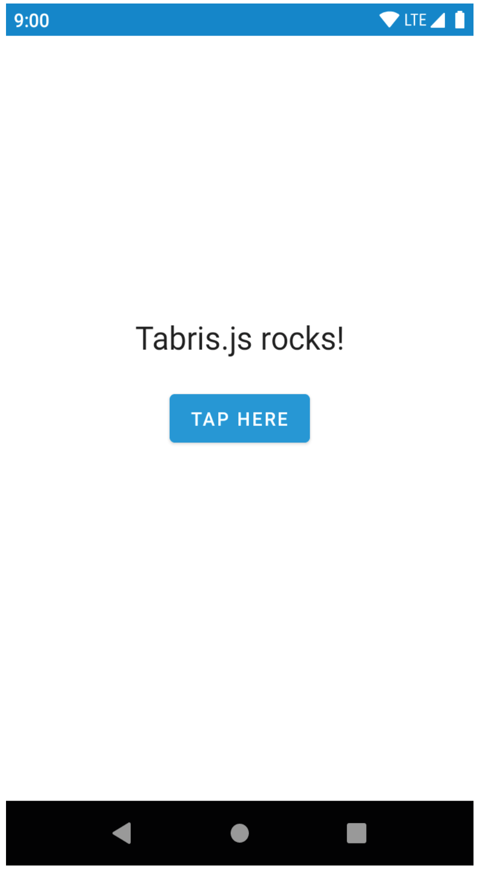
Ctrl+Space: Auto Complete | *F1 (while focused): Command Palette* | Compiled Snippet | Documentation

The playground only works correctly with the latest release of the Developer App.
TypeScript, JSX and tabris-decorators module are supported.

Scan the code from the playground

Scan the code from the playground by tapping on the QR code button in the URL input field of the Tabris.js Developer App:


9:00
Connect to Tabris.js app...
Widgets
Create beautiful apps faster with the build in set of high quality UI widgets.
Action
An executable item that is integrated in the application's navigation menu. Add a select listener to react to action taps.
Action
SHARE
</> Creating a simple Action
</> Using placement priority on actions
Widgets
About



See the snippet on your device

✓ Now the snippet runs on your mobile device.

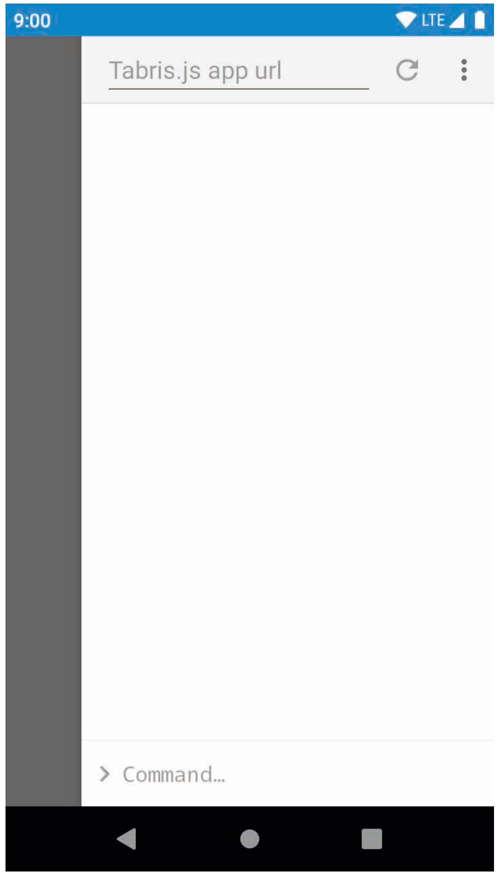
Go back to the home screen

After you have started a snippet, you can go back to the home screen of the Developer App by:

- using the back button on Android
- using the home symbol in the developer console on both platforms.

Open the developer console

To open the developer console, slide from the right edge of the screen to the left.



Tip for iOS tablet devices

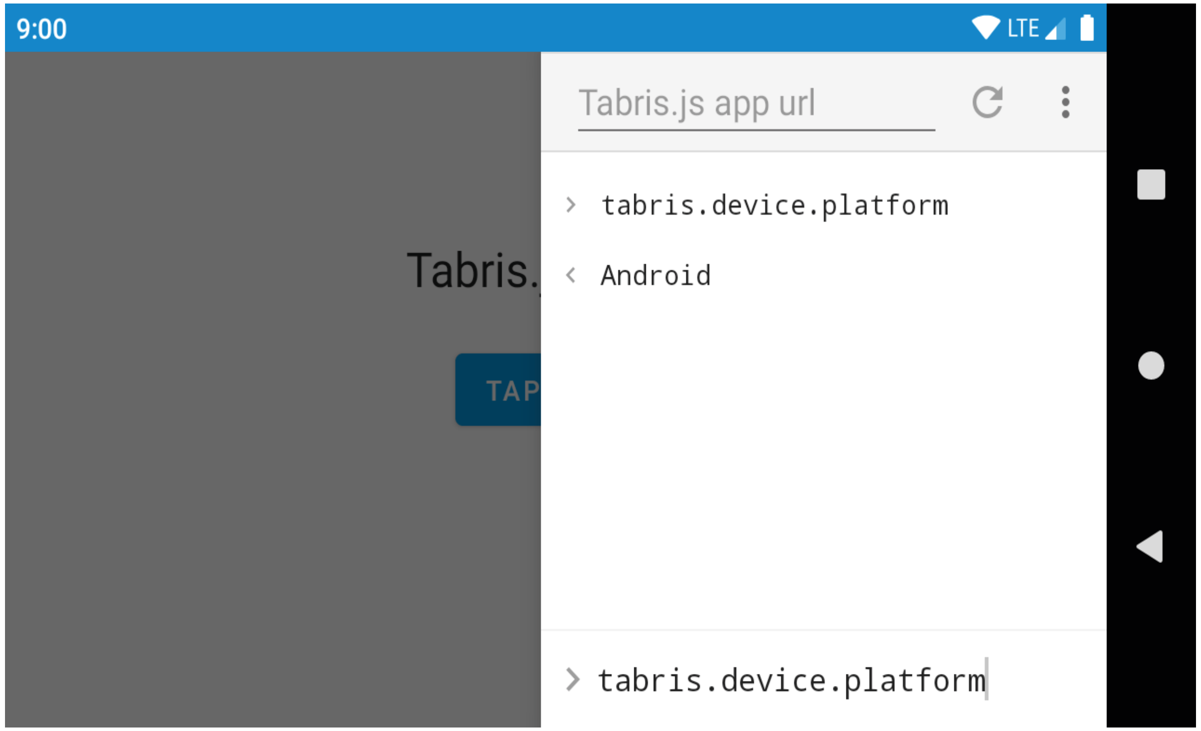
If you have problems opening the developer console: double-tap with four fingers on the screen.

Functions of the developer console

You can use the interactive developer console to:

- go back to the home screen of the Developer App, or reload your code
- view log messages and errors that occur when you run your code. You can filter the log and share it, for example by email
- interact with the running app by executing JavaScript code from the developer console like in browsers. Here is something to try: `tabris.device.platform`

Run JavaScript code from the console



```
> tabris.device.platform
< Android
```

```
> tabris.device.platform
```

Edit the "Hello, World!" example

The ″Hello, World!″ example is fully functional and directly loaded from the playground.
You can edit the code in the playground, and reload to see the changes in action:

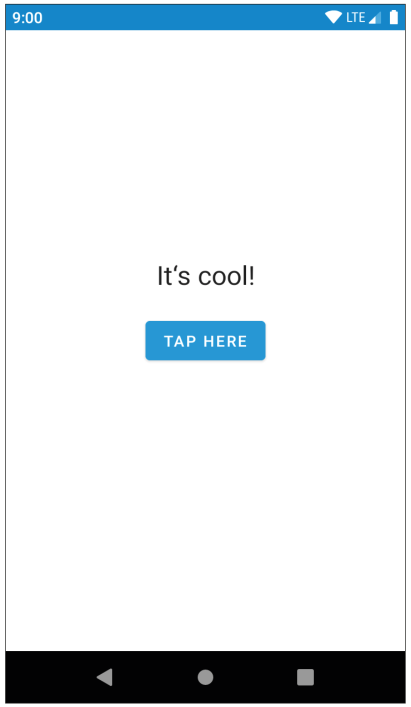
- Use the developer console for reloading.
- You can also reload by scanning the barcode again from the home screen of the Developer App.

Try to change a few things: the title of the page, the button, and the text.

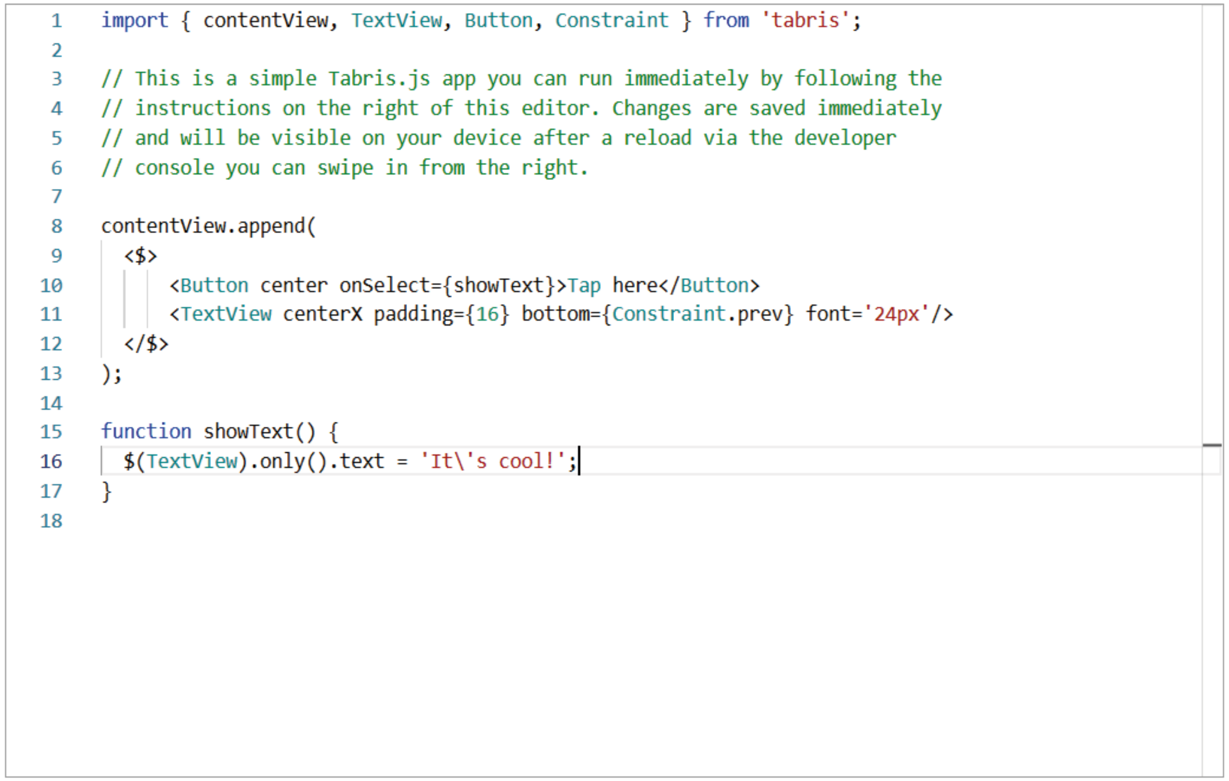
View your changes

✓ Now you can see in your Developer App what you changed in the playground.

Your changes on the mobile device



Your changes in the playground



```
import { contentView, TextView, Button, Constraint } from 'tabris';

// This is a simple Tabris.js app you can run immediately by following the
// instructions on the right of this editor. Changes are saved immediately
// and will be visible on your device after a reload via the developer
// console you can swipe in from the right.

contentView.append(
  <$>
    <Button center onSelect={showText}>Tap here</Button>
    <TextView centerX padding={16} bottom={Constraint.prev} font='24px'/>
  </$>
);

function showText() {
  $(TextView).only().text = 'It\'s cool!';
}

```

Tabris.js snippets

In the Tabris.js GitHub repository, you can find code snippets for nearly every feature in Tabris.js. They can be chosen in the Playground by selecting them from the snippet dropdown menu.

3 Create your first app

Before you start developing

Before you start developing your first app, you need to set up your development machine and your project.
You will also learn how to run your app, and how to structure your app if it has more than one page.

Set up your development machine

To set up your **development machine**, install the following software:

- Node.js
  For more information see nodejs.org
- the Tabris CLI
  In your Terminal type: `npm install -g tabris-cli`
  A Terminal is a command-line interpreter such as Terminal (Mac), Gnome Terminal (Linux) or Command Prompt (Windows).
- a text editor or a JavaScript IDE.

Your mobile device must be connected to the same Wi-Fi network as your development machine.

3.1 Set up your project by using a template

The easiest way to create your project is using Tabris CLI.

Functions of Tabris CLI

Tabris CLI can:

- create a new Tabris.js project to develop your first customized app
- serve your project files to the Developer App
- build an app on your local machine.

Initialize your project

To initialize your project:

Type `cd` to an empty project directory.
Type `tabris init`.

Enter app information

You need to enter some information, like:

- Tabris.js version. This ebook uses 3.x.
- App name as it appears on the device's home screen.
- App ID as it will be used to build and identify the app in the stores.
- Type of project: Compiled (recommended, supports modern JS, JSX and/or TypeScript) or vanilla app (run JavaScript files as-is).
- Whether to include configuration for supported IDEs in the project, like test launch configurations.
- Example code: determines the style of the generated app example code for compiled projects. The examples from this ebook follow the “Minimal (JavaScript/JSX)” style.

The basic files

The Tabris CLI creates a basic example app. The most important files are: a package.json and a src/index.jsx.

The package.json

```
{
 "main": "dist",
 "private": true,
 "scripts": {
   "test": "npm run build && npm run lint",
   "lint": "tslint --project . -t verbose",
   "build": "tsc -p .",
   "watch": "tsc -p . -w --preserveWatchOutput
              --inlineSourceMap",
   "start": "tabris serve -a -w"
 },
 "dependencies": {
   "tabris": "~3.1.0"
 },
 "devDependencies": {
   "tslint": "^5.16.0",
   "typescript": "~3.3.4000"
 }
}
```

More information on package.json

The package.json is a manifest file that describes your application and its dependencies.
For more information on how to use a package.json, see:
https://docs.npmjs.com/getting-started/using-a-package.json

The src/index.jsx

The src/app.js file contains the code of your application.

```
import {Button, TextView, contentView} from 'tabris';

contentView.append(
 <$>
   <Button center onSelect={showText} text='Tap>'/>
   <TextView centerX bottom='prev() 20' font='24px'/>
 </$>
);

function showText() {
 $(TextView).only().text = 'Tabris.js rocks!';
}
```

The config.xml

Tabris CLI also creates a config.xml file for you.
Every Tabris.js project that you want to build needs a config.xml file.
This file describes your app.

A minimal config.xml

A minimal config.xml looks similar to this:

```xml
<?xml version='1.0' encoding='utf-8'?>
<widget id="my.first.app" version="0.1.0">
  <name>Hello World</name>
  <description>Example Tabris.js App</description>
  <author email="dev@example.com">
    Tabris.js Team
  </author>
  <preference name="EnableDeveloperConsole" value="$IS_DEBUG" />
</widget>
```

3.2 Run your app

Run your app

With this basic structure in place, you can now run your app for the first time:

1. In the project directory type `tabris serve`, it will start an HTTP server.
   The server outputs the IP address of your machine on start up.
   Let the server run as long as you develop/test your app.
   To stop the server, hit CTRL-C.
2. In the Developer App, type in the URL tab `http://<development-machine-ip-address>:8080/`.
3. Confirm to run your app.

✓ The Developer App now downloads the script and executes it on your mobile device.

Open the developer console

Swipe from the right edge of the screen to the left, to open the developer console.
You can reload the script or go back to the home screen of the Developer App.

✓ Now you can continue developing.

3.3 Develop-deploy-test cycle

Develop, deploy, test

The develop-deploy-test cycle is very fast with Tabris.js apps:

1. Just edit the JavaScript files that make up your code base in a text editor, and save them.
2. On your mobile device open the developer console, and tap the reload button.

If you need to debug your app, you need an Android device (or an emulator) and a Chrome browser.
A detailed description of debugging Tabris.js can be found online.

Add more pages

Let's continue developing your app by adding a navigation view and some pages.
Apps with multiple pages should be split into several files.

Use src/index.jsx as an entry point, and one file per page.

See the example below.

Sample app structure

As an example, let’s create a News page.
This is how you can create page modules and use them:

src/index.jsx

```
import {Button, NavigationView, Page, contentView}
from 'tabris';
import {NewsPage} from './pages/NewsPage';

// Create a full-size navigation view and add a page
to it
contentView.append(
 <NavigationView stretch>
   <Page title='Main Page'>
     <Button center onSelect={
       () => openNewsPage()}>Open news page</Button>
   </Page>
 </NavigationView>
)

function openNewsPage() {
 $(NavigationView).only().append(
   <NewsPage />
 );
}
```

Sample app structure

src/pages/NewsPage.jsx

```
import {Page, TextView} from 'tabris';

export class NewsPage extends Page {

  constructor(properties) {
    super();
    this.set({title: 'News', ...properties}).append(
      <TextView center>No news yet!</TextView>
    );
  }

};
```

Directory structure

Tabris.js does not force a directory structure upon your JavaScript sources. You can use the project layout as described above or any structure you like.

Just make sure that you reference modules with a relative path (for example "./NewsPage" if NewsPage.jsx is in the same directory as src/app.js).

3.4 Extend your app with libraries and plug-ins

Extend Tabris.js

You can extend Tabris.js with existing JavaScript libraries and native extensions.

The Tabris.js framework supports many W3C APIs out of the box, such as web APIs, Canvas for drawing, and localStorage. Libraries that depend on these APIs will work as long as they don’t use the DOM.

Use Cordova plug-ins

Other features, including native device features like sensors or camera, can be added with Apache Cordova plug-ins.

To add Apache Cordova plug-ins to your app, you need to add them to the config.xml.

Add plug-ins

The online build service supports the Cordova <plugin /> tag. With this tag, you can add plug-ins by using their ID, an HTTP URL or a git URL.

A config.xml with plug-ins

A sample config.xml with a Cordova plug-in could look like this:

```
<?xml version='1.0' encoding='utf-8'?>
<widget
    id="my.first.app"
    version="0.1.0"
    xmlns="http://www.w3.org/ns/widgets"
    xmlns:cdv="http://cordova.apache.org/ns/1.0">
    ...
  <plugin name="cordova-plugin-diagnostic"
   spec="4.0.12" />
</widget>
```

4 Build your app

Bundle, brand and build your app

To publish your app in the Apple App Store or Google Play Store, you need to bundle, brand and build the app.
Tabris.js uses Apache Cordova to build and package apps.

Build without local setup

To build an app without any local setup or hardware, you can use the online build service on tabrisjs.com.
You need to store the source code of your app in a GitHub repository to make it available for the build service.

Local build

For the local build, you can use local tools on your development machine.
This ebook describes how to build your app with the online build service.

The online build service is free for unlimited public GitHub repositories and one private repository. To build from unlimited private repositories, you need a Pro account. The local build is free for everyone.

4.1 Build service

Provide access to the source code

The build service needs access to the source code of your app to package it into a native app.
The easiest way is to push your code to a GitHub repository.

The build service installs the dependencies specified in your package.json from npm on the fly.
As a result, you don't have to put the node_modules folder under version control.

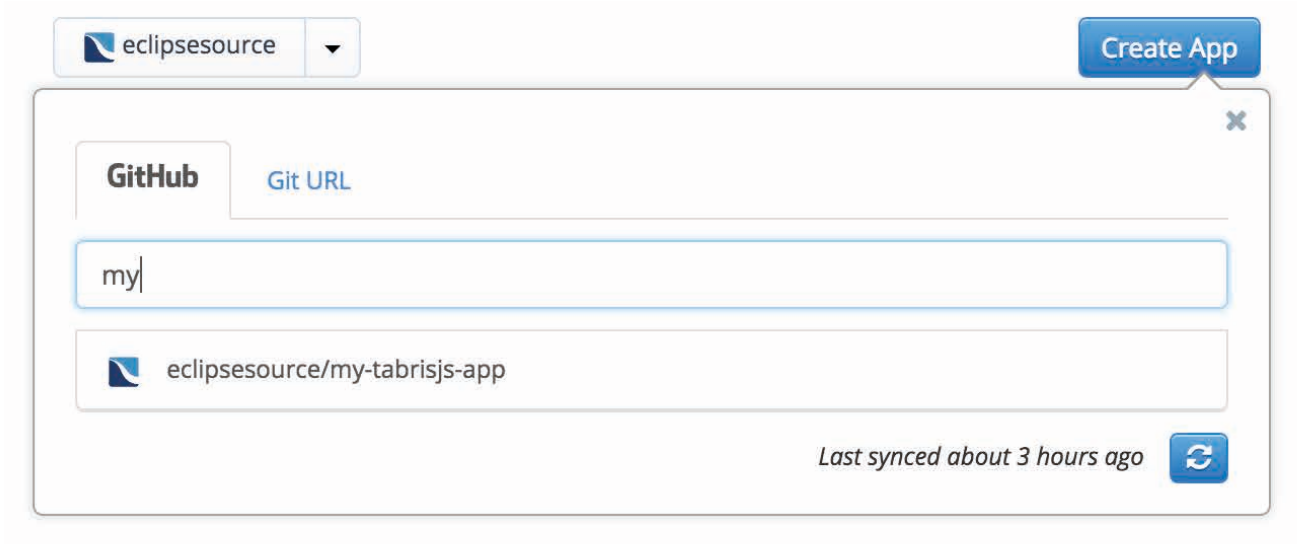
become mobile

Use the build service

To use the build service:

1. Go to tabrisjs.com and sign in with your GitHub account.
2. Go to the **My Apps** section.
3. Click **Create App**.
4. In the list of repositories, select the GitHub repository that contains a Tabris.js app.

   If it is not visible, then you may need to click the **synchronize** button.



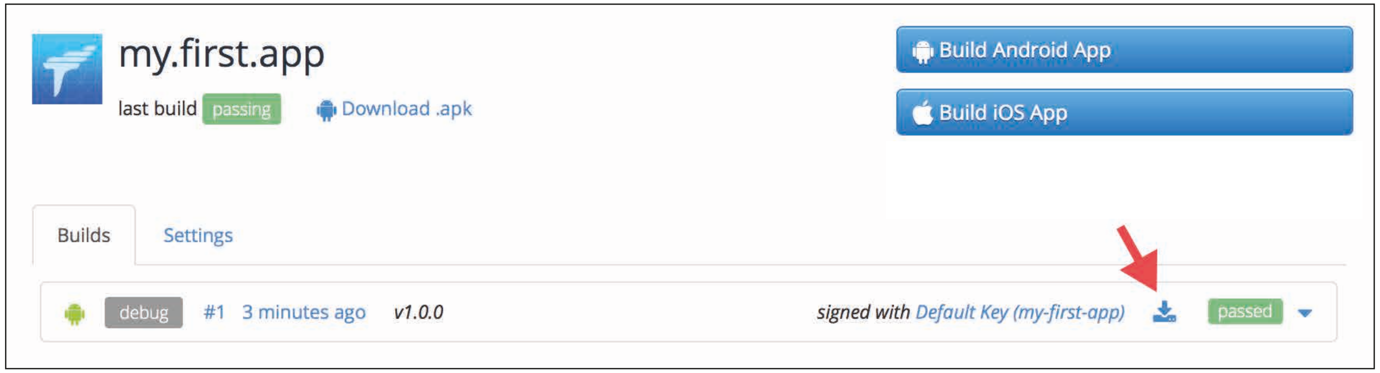
Execute the first build

To create a debug Android app, that means an app containing the developer console, follow these steps:

1. Select the newly created app.
2. Click the **Build Android App** button.

Install the .apk file

A few minutes later you can download an Android .apk file, and install this file on your mobile device.

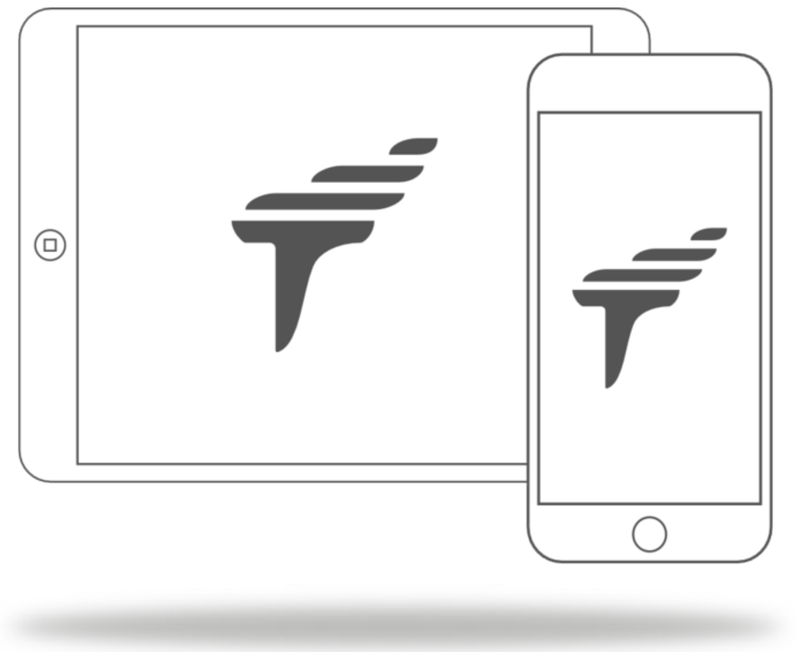


Advanced build configuration

To configure debug builds or release builds on iOS, you need a signing key. The same is true for release builds on Android.

Signing keys are important to secure your app.
Because creating signing keys is rather a complex process, it is out of the scope of this ebook.

You can find detailed guides for signing keys online.



5 Conclusion

Tabris.js blog

By the end of this ebook, you successfully created your first Tabris.js 3 app!

If you like to learn more about the Tabris.js platform, its features and possibilities, then have a look at our blog posts on:

http://eclipsesource.com/blogs/tag/tabris-js/

Feedback

Help us improve Tabris.js! Feedback is always welcome.

Feel free to invite your friends if you find Tabris.js interesting.

Copyright © 2019 by EclipseSource