coligo.io

Building a Mobile App with Cordova and Vue.js

About the Author

17-22 minutes

Cordova is a framework that lets you create mobile apps using web technologies like HTML, Javascript and CSS. This allows you to target multiple platforms like Android and iOS using one code base. Although you still need platform-specific technologies like the Android SDK and XCode to build an app, you can create apps without having to write any Android or iOS code.

Since you can write code in HTML and Javascript, it's very easy to use front-end Javascript libraries like Vue.js with Cordova.

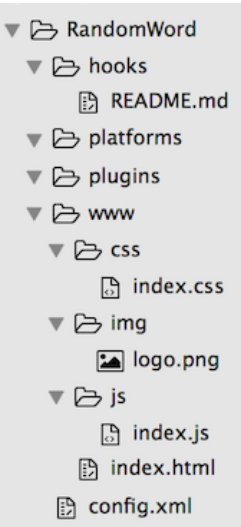
This tutorial will show you how to create a simple mobile app which generates random words by using Cordova and Vue.js.

- Download Node.js
- Install Cordova: `npm install -g cordova`
- Vue.js Basics

  Create a Cordova project called RandomWord:

  ```
  cordova create RandomWord
  cd RandomWord
  ```

  This will create the directory structure of a Cordova project:



- **config.xml** - contains info about the app, plugins it uses and platforms it targets
- **platforms** - contains the Cordova libraries for the targeted platforms like Android and iOS that the app will run on
- **plugins** - contains the Cordova libraries for plugins used by the app which allow the app to access things related to the device like Camera and Battery Status
- **www** - contains source code for the app like HTML, Javascript and CSS files
- **hooks** - contains scripts used to customize the build system for the app

  Add the Android platform:

  ```
  cordova platform add android
  ```

  This will add the Android platform library to the platforms directory (platforms/android).

  It will also add the whitelist plugin which is used to specify which URLs the app can link to or open in the browser. The random word generator app will not need this functionality but you can read more about the whitelist plugin here.

  The `--save` flag adds the platform engine to config.xml which is used by the cordova prepare command when initializing a Cordova project from a **config.xml** file.

  ```
  ...
      <engine name="android" spec="~5.2.1" />
  </widget>
  ```

  Check if you have the requirements for building/running Android apps through Cordova:

  ```
  cordova requirements
  ```

  If missing requirements, see the Cordova Docs for Android and the Help section at the bottom of the tutorial. This is definitely the hardest part of the tutorial. Just be patient and refer to the links mentioned. Once you get all the requirements working, the rest of the tutorial is a breeze.

  Build the app for Android:

  ```
  cordova build android
  ```

  Plug-in your phone to your computer and run the app for Android:

```
cordova run android
```

If there is no Android phone connected to the computer, Cordova will run the app on an emulator.

The sample app is very simple and all it does is change the background colour of a label.

To use iOS instead of Android, do the same steps as above but replace `android` with `ios`. If missing requirements, see Cordova Docs for iOS and the Help section at the bottom of the tutorial. If running Cordova on a Windows computer, you can NOT build/run the app on iOS since the iOS Cordova platform needs Apple OS X.

Alternatively, you can use your browser instead of a mobile device by using the `browser` platform. Do the same steps as above but replace `android` with `browser`.

Modify the app info in **config.xml** to be about our random word generator:

```
<?xml version='1.0' encoding='utf-8'?>
<widget id="io.coligo.randomword" version="1.0.0"
xmlns="http://www.w3.org/ns/widgets"
xmlns:cdv="http://cordova.apache.org/ns/1.0">
    <name>RandomWord</name>
    <description>
        A mobile app for generating a random
word.
    </description>
```

```
    <author email="michaelviveros@gmail.com"
href="http://www.michaelviveros.com/">
        Michael Viveros
    </author>
    ...
```

Like in any HTML file, add the Vue.js CDN to the bottom of **www/index.html**:

```
...
        <script type="text/javascript"
src="cordova.js"></script>
        <script src="http://cdn.jsdelivr.net
/vue/1.0.16/vue.js"></script>
        <script type="text/javascript"
src="js/index.js"></script>
    </body>
</html>
```

To allow the app to access the Vue.js library, we also need to add the following to the end of the Content Security Policy (CSP) `meta` tag in www/index.html:

```
; script-src 'self' http://cdn.jsdelivr.net
/vue/1.0.16/vue.js 'unsafe-eval'
```

The Content Security Policy of a webpage allows you to create a whitelist of sources of trusted content and instructs the browser to only execute or render resources from those sources. This is different from the whitelist plugin mentioned above since the whitelist plugin is used mainly to define which links the app is allowed to open whereas the CSP is used to define which scripts the app can execute and which urls the app can make http requests to.

The `script-src` part of the CSP `meta` tag defines which scripts can be executed by the app.

- 'self' - allows scripts from the same origin like www/js/index.js
- http://cdn.jsdelivr.net/vue/1.0.16/vue.js - allows the Vue.js library
- 'unsafe-eval' - allows unsafe dynamic code evaluation since parts of the Vue.js library code use strings to generate functions

The CSP meta tag should look like this:

```
<meta http-equiv="Content-Security-Policy"
```

```
content="default-src 'self' data: gap:
https://ssl.gstatic.com 'unsafe-eval'; style-src
'self' 'unsafe-inline'; media-src *; script-src
'self' http://cdn.jsdelivr.net/vue/1.0.16/vue.js
'unsafe-eval'">
```

For more info about CSP, see html5rocks and the Cordova Docs.

After replacing the code in the body of **www/index.html** with some Vue.js code to show a random word and removing some comments, **wwww/index.html** will look like:

```
<!DOCTYPE html>
<html>
    <head>
        <meta http-equiv="Content-Security-
Policy" content="default-src 'self' data: gap:
https://ssl.gstatic.com 'unsafe-eval'; style-src
'self' 'unsafe-inline'; media-src *; script-src
'self' http://cdn.jsdelivr.net/vue/1.0.16/vue.js
'unsafe-eval'">
        <meta name="format-detection"
content="telephone=no">
        <meta name="msapplication-tap-highlight"
content="no">
        <meta name="viewport" content="user-
scalable=no, initial-scale=1, maximum-scale=1,
minimum-scale=1, width=device-width">
        <link rel="stylesheet" type="text/css"
href="css/index.css">
        <title>Random Word</title>
    </head>
    <body>
        <div id="vue-instance" class="app">
            <h1>Random Word</h1>
            <button id="btn-get-random-word"
@click="getRandomWord">Get Random Word</button>
            <p>{{ randomWord }}</p>
        </div>
        <script type="text/javascript"
src="cordova.js"></script>
        <script src="http://cdn.jsdelivr.net
/vue/1.0.16/vue.js"></script>
```

```
        <script type="text/javascript"
src="js/index.js"></script>
    </body>
</html>
```

Now we will add some Javascript to generate the random word that gets shown.

www/js/index.js currently has some code to change the background colour of a label when the app receives the `deviceready` event. We won't need to do anything extra when the app receives the `deviceready` event for our simple random word generator but it's good to know that you can use the `bindEvents` method to do different things at different stages of the app's lifecycle. See Cordova Events for more info.

We will add a new method in **www/js/index.js** called `setupVue` which will create a new Vue instance and mount it to the random word `div`. The new Vue instance will have a `getRandomWord` method that will pick a random word from a list of words when the Get Random Word button is clicked. We also need to call `setupVue` from the `initialize` method.

```
var app = {
    initialize: function() {
        this.bindEvents();
        this.setupVue();
    },
    ...
    setupVue: function() {
        var vm = new Vue({
            el: "#vue-instance",
            data: {
                randomWord: '',
                words: [
                    'formidable',
                    'gracious',
                    'daft',
                    'mundane',
                    'onomatopoeia'
                ]
            },
            methods: {
                getRandomWord: function() {
```

```
                    var randomIndex =
Math.floor(Math.random() * this.words.length);
                    this.randomWord =
this.words[randomIndex];
                }
            }
        });
    }
};

app.initialize();
```

After removing the old code that changed the background colour of a label in `receivedEvent` and removing some comments, **www/js/index.js** will now look like:

```
var app = {
    initialize: function() {
        this.bindEvents();
        this.setupVue();
    },
    bindEvents: function() {
        document.addEventListener('deviceready',
this.onDeviceReady, false);
    },
    onDeviceReady: function() {
        app.receivedEvent('deviceready');
    },
    receivedEvent: function(id) {
        console.log('Received Event: ' + id);
    },
    setupVue: function() {
        var vm = new Vue({
            el: "#vue-instance",
            data: {
                randomWord: '',
                words: [
                    'formidable',
                    'gracious',
                    'daft',
                    'mundane',
                    'onomatopoeia'
```

```
                ]
            },
            methods: {
                getRandomWord: function() {
                    var randomIndex =
Math.floor(Math.random() * this.words.length);
                    this.randomWord =
this.words[randomIndex];
                }
            }
        });
    }
};

app.initialize();
```

Build it, plug-in your phone and run it:

```
cordova build android
cordova run android
```

The app should look like this:

Instead of picking a random word from a hard-coded list of words, the app can do a GET request to an API that generates random words like the Wordnik Random Word API

To allow the app to make an http request to the random word API,

add the following to the end of the CSP meta tag:

```
; connect-src http:
```

The `connect-src` part of the CSP meta tag defines which origins the app can make http requests to.

The app will use the vue-resource library to make an HTTP request so we have to add vue-resource to the `script-src` part of the CSP meta tag and add the vue-resource CDN.

index.html will look like:

```
<!DOCTYPE html>
...
        <meta http-equiv="Content-Security-
Policy" content="default-src 'self' data: gap:
https://ssl.gstatic.com 'unsafe-eval'; style-src
'self' 'unsafe-inline'; media-src *; script-src
'self' http://cdn.jsdelivr.net/vue/1.0.16/vue.js
https://cdn.jsdelivr.net/vue.resource/0.7.0/vue-
resource.min.js 'unsafe-eval'; connect-src
http://api.wordnik.com:80/v4/words.json
/randomWord">
...
        <script src="http://cdn.jsdelivr.net
/vue/1.0.16/vue.js"></script>
        <script src="https://cdn.jsdelivr.net
/vue.resource/0.7.0/vue-resource.min.js">
</script>
        <script type="text/javascript"
src="js/index.js"></script>
    </body>
</html>
```

To make the HTTP request to the random word API, we can use the http service of the vue-resource library in the `getRandomWord` method of the Vue instance in **www/js/index.js**:

```
...
    setupVue: function() {
        var vm = new Vue({
            el: "#vue-instance",
            data: {
                randomWord: ''
            },
```

```
            methods: {
                getRandomWord: function() {
                    this.randomWord = '...';
                    this.$http.get(

'http://api.wordnik.com:80/v4/words.json
/randomWord?api_key=a2a73e7b926c924fad7001ca3111acd55af2ffabf50
                    ).then(function (response) {
                        this.randomWord =
response.data.word;
                    }, function (error) {
                        alert(error.data);
                    });
                }
            }
        });
    }
};

app.initialize();
```

Build it, plug-in your phone and run it:

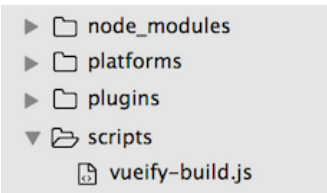
```
cordova build android
cordova run android
```

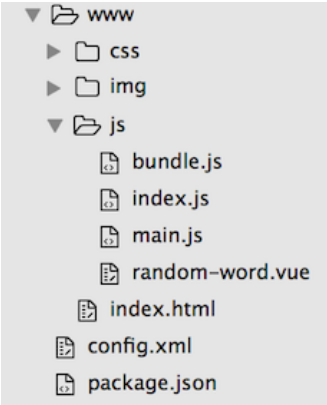
The app should look the same as before but now it will get the random words from the random word API.

Vueify is a library for Vue.js that lets you break down the UI into individual components with their own HTML, JavaScript and CSS. This will make your app more modular and it allows you to define components in a hierarchical manner.

Using Vue components will add an additional step to your build system to bundle all the components together. Cordova makes this really easy by using hooks which let you specify additional scripts to be run at different parts of your build system.

This is what your directory will look like after adding the Vue Component:





Create a component with all the code for the random word generator called **www/js/random-word.vue**:

```
<template>
  <div class="app">
    <h1>Random Word</h1>
    <button id="btn-get-random-word"
@click="getRandomWord">Get Random Word</button>
    <p>{{randomWord}}</p>
  </div>
</template>

<script>
export default {
  data () {
    return {
      randomWord: ''
    }
  },
  methods: {
    getRandomWord: function() {
        this.randomWord = '...';
        this.$http.get(
            'http://api.wordnik.com:80
/v4/words.json
/randomWord?api_key=a2a73e7b926c924fad7001ca3111acd55af2ffabf50
        ).then(function (response) {
            this.randomWord = response.data.word;
        }, function (error) {
            alert(error.data);
        });
    }
```

```
  }
}
</script>
```

The HTML from **www/index.html** goes in the `template` tag and the Javascript from **www/js/index.js** goes in the `script` tag of **random-word.vue**.

Create a new Vue instance that contains the random word component in a new file called **www/js/main.js**:

```
var Vue = require('vue');
var VueResource = require('vue-resource');
var RandomWord = require('./random-word.vue');

Vue.use(VueResource);

var vm = new Vue({
  el: 'body',
  components: {
    'random-word': RandomWord
  }
});
```

To bundle our component, we will use browserify and vueify to create a file called bundle.js. Make a new directory called scripts and a new file in it called **vueify-build.js** which will contain the code for bundling the random word component.

In the past, a script like vueify-build.js would have gone in the hooks directory that gets created from `cordova create` but using the hooks directory has been deprecated. So you can delete the hooks directory and use the scripts directory instead.

scripts/vueify-build.js will look like:

```
var fs = require('fs');
var browserify = require('browserify');
var vueify = require('vueify');

browserify('www/js/main.js')
  .transform(vueify)
  .bundle()
  .pipe(fs.createWriteStream('www/js/bundle.js'))
```

Before, we were using CDNs in **www/index.html** to reference the

Vue.js libraries but now **www/js/main.js** is using javascript to do that. So we have to add a **package.json** file to define all the dependencies for the Vue.js libraries:

```
{
  "name": "random-word",
  "version": "1.0.0",
  "description": "A mobile app for generating a
random word",
  "main": "index.js",
  "dependencies": {
    "browserify": "~13.0.1",
    "vue": "~1.0.24",
    "vue-resource": "~0.7.4",
    "vueify": "~8.5.4",
    "babel-core": "6.9.1",
    "babel-preset-es2015": "6.9.0",
    "babel-runtime": "6.9.2",
    "babel-plugin-transform-runtime": "6.9.0",
    "vue-hot-reload-api": "2.0.1"
  },
  "author": "Michael Viveros",
  "license": "Apache version 2.0"
}
```

All the babel modules, browserify and vue-hot-reload-api are used by vueify, see vueify Docs.

Get all the node modules for the dependencies defined in **package.json**:

```
npm install
```

Add a hook to the bottom of **config.xml** to tell Cordova to bundle the random word component before building the rest of the app:

```
...
    <hook type="before_compile"
src="scripts/vueify-build.js" />
</widget>
```

Recall that scripts/vueify-build.js will generate the bundled component and put it into www/js/bundle.js

Add the random word component to the body of **www/index.html** by adding a `random-word` tag and a `script` tag pointing to the

bundled component.

```
...
        <link rel="stylesheet" type="text/css"
href="css/index.css">
        <title>Random Word</title>
    </head>
    <body>
        <random-word></random-word>
        <script src="js/bundle.js"></script>

        <script type="text/javascript"
src="js/index.js"></script>
        <script type="text/javascript"
src="cordova.js"></script>
    </body>
</html>
```

Note that the link tag in **www/index.html** defines the CSS for the app and the `div` in **www/js/random-word.vue** uses the "app" class defined in the CSS.

Since our random word component has all the code for generating the random word, we can remove the `setupVue` method from **www/js/index.js** which will now look like:

```
var app = {
    initialize: function() {
        this.bindEvents();
    },
    bindEvents: function() {
        document.addEventListener('deviceready',
this.onDeviceReady, false);
    },
    onDeviceReady: function() {
        app.receivedEvent('deviceready');
    },
    receivedEvent: function(id) {
        console.log('Received Event: ' + id);
    }
};

app.initialize();
```

Build it, plug-in your phone and run it:

```
cordova build android
cordova run android
```

The app should look and function the same as before but now it is using a Vue component.

All Done.

Cordova makes developing mobile apps with web technologies super simple. Connecting Cordova and Vue.js is also very easy and lets you take advantage of all the cool things about Vue.js (2-way data binding, components, ...) in mobile apps. Now you can make an app in HTML, JavaScript and CSS that targets multiple platforms with 1 code base.

This tutorial went over:

- Creating a Cordova project
- Connecting Cordova and Vue.js
- Making HTTP Requests in a Cordova app by updating it's Content Security Policy
- Using Vue Components in a Cordova app by adding Hooks

Android

After installing the Android SDK, you can run the following to open the Android SDK Manager:

```
/Users/your_username/Library/Android/sdk/tools
/android sdk
```

I installed the following packages:

Tools

- Android SDK Tools
- Android SDK Platform-tools
- Android SDK Build-tools

Android 6.0 (API 23)

- SDK Platform
- Intel x86 Atom_64 System Image

Extras

- Intel x86 Emulator Accelerator (HAXM Installer)

iOS

I got an error when trying to install ios-deploy through npm while running OS X El Capitan 10.11, resolved it by running:

```
sudo npm install -g ios-deploy –unsafe-perm=true
```

See StackOverflow

My name is Michael Viveros and I am in my 5th year of studying Software Engineering. I am a passionate programmer, incredibly inconsistent golfer and sarcastically subtle joker. I am developing a golf stat-tracking website and mobile app using Cordova and Vue.js. You can read more @ michaelviveros.com.