
Do you like to have a beach View

My Intro

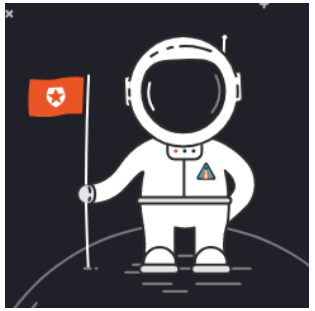
- Baskar Rao

Twitter - baskarmib

https://www.linkedin.com/in/baskarrao-dandlamudi

baskarrao.dandlamudi@outlook.com

https://github.com/baskar3078/OrlandoCodeCamp2019



Agenda

- Overview of Vue Framework
- Vuex Concepts
- Native Script Vue and Vue Synergy
- Demo using Vue and Native Script Vue
- Resources for Further Learning

Overview of Vue Framework

- Vue - > Progressive Web Application Framework
  - Reactive Approach
  - Changes in the model are updated to View.
- Vue Application are made up of Vue Instances.
  - Each Components are translated to respective Vue Instances
- Vue Components.
  - Vue Supports Single File Components
  - Supports using different files for Components

```
var app = new Vue({ el: '#app', data: { message:
'Hello Vue!' } })
```

Overview of Vue Framework

- Vue - > Light Weight Framework
  - Can be referred by using script tag
- Vue Files can be implemented using html or .vue Files.
- Vue CLI can be used to create application using interactive console.
- Vue UI can also be used to create application.
- Vue Dev Tools and Google Chrome Console can be used to debug Vue Applications

Overview of Vue Framework

- Content of Vue File
  - Template Section
  - Script Section
  - Style Section

```
<script>
export default {
  name: 'HomePage',
  props: {
    msg: String,
  },
};
</script>
```

```
<template>
  <div class="home">
   <div>
     <img class="logo" src="../assets/CCC_logo.1.png" aria-hidden="true"/>
   </div>
   <div class="get-started">
   Hello World Vue Js Columbus Meetup
  </div>
  </div>
</template>
```

```
<!-- Add "scoped" attribute to limit CSS to this component only -->
<style scoped>
.home
{
  text-align: center;
}
.logo
{
  height: 300px;
}
</style>
```

Overview of Vue Framework

- Vue Application Lifecycle
    - Before Create
    - Created - Primarily to perform your API fetch
    - Before Mount –Changes to DOM before render
    - Mounted
    - Before Update
    - Updated
    - Before Destroy
    - Destroyed

- Great Resource to understand - https://alligator.io/vuejs/component-lifecycle/

Basics of Vue Framework

- Properties in Vue
  - Access property using text interpolations {{propertyName}}
  - Properties can be in data or computed.

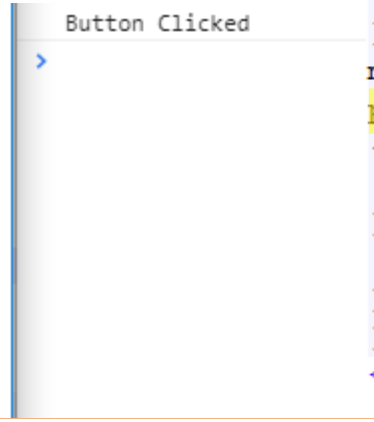
- Binding in Vue
  - v-bind:attribute = “propertyName” or :attribute = “propertyName”
  - Any thing prefixed using “v” is referred to as directive.

```
<body>
<div id="app">
<header>
<h1 v-text="headerText"></h1>
<h2>{{secondText}}</h2>
<input type="text" v-bind:value="textvalue"></input>
<input type="text" :value="textvalue"></input>
</header>
</div>

<script>
var application = new Vue({
el:"#app",
data:{

   headerText: "Hello World Vue",
   secondText:"Second Header",
   textvalue :"Sample"
}
});
</script>

</body>
```

Overview of Vue Framework

- Bind Events to Controls
  - v-on can be used to bind functions to events
- Functions which need to be executed in response to events should be defined in “methods” object.

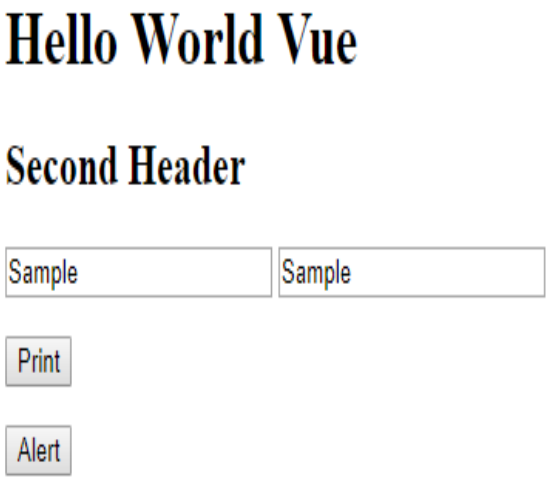
```
<p>
<input type="button"
v-on:click="printtoconsole"
 value="Print"></input>
</p>
</div>
<script>
var application = new Vue({
el:"#app",
data:{

   headerText: "Hello World Vue",
   secondText:"Second Header",
   textvalue :"Sample"
},
methods:{
printtoconsole : function()
{
   console.log("Button Clicked");
}

}
});
</script>
```



Overview of Vue Framework

- Control Visibility using v-show
  - v-show – The element is rendered to DOM
  - Display is controlled by inline styling.
  - Recommended to group elements.

- Control Visibility using v-if and v-else
  - The element is removed from DOM
  - v-if and v-else should be used together




```
> this.webstore
<  undefined
> this.application
<  ▸wn {_uid: 0, _isVue: true, $options: {…}, _renderProxy: wn, _sel
   f: wn, …}
> this.application._data.showalertbutton
<  false
> this.application._data.showalertbutton = false
<  false
> this.application._data.showalertbutton = true
<  true
>
```

Overview of Vue Framework

Hello World Vue

Second Header

Sample | Sample

Alert

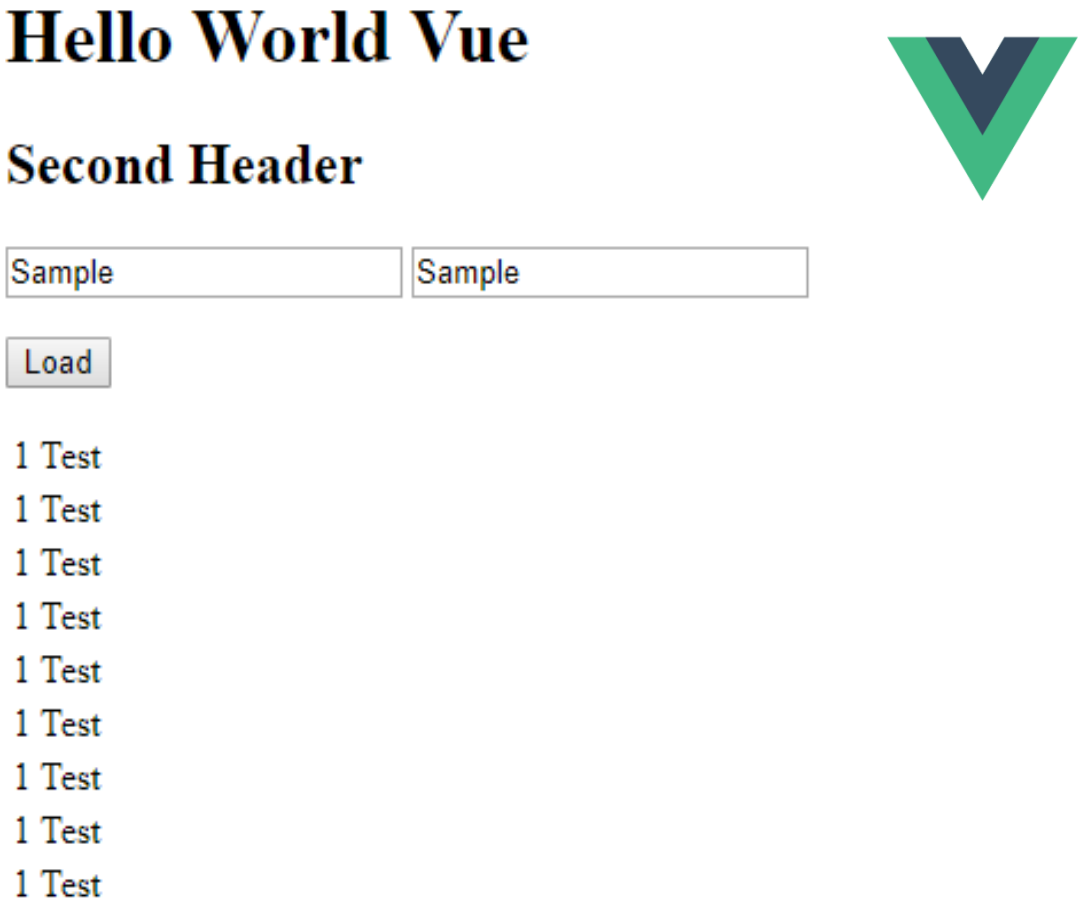
```
<html>
▶ <head>…</head>
▼ <body> == $0
  ▼ <div id="app">
    ▶ <header>…</header>
    ▼ <p>
        <input type="button" value="Print" style="display: none;">
      </p>
    ▼ <div>
        <input type="button" value="Alert">
      </div>
    </div>
  ▶ <script>…</script>
  </body>
</html>
```

```
> this.webstore
<  undefined
> this.application
<  ▶ wn {_uid: 0, _isVue: true, $options: {…}, _renderProxy: wn, _sel
     f: wn, …}
> this.application._data.showalertbutton
<  false
> this.application._data.showalertbutton = false
<  false
> this.application._data.showalertbutton = true
<  true
> this.application.$data.showbutton = false
<  false
>
```

Overview of Vue Framework

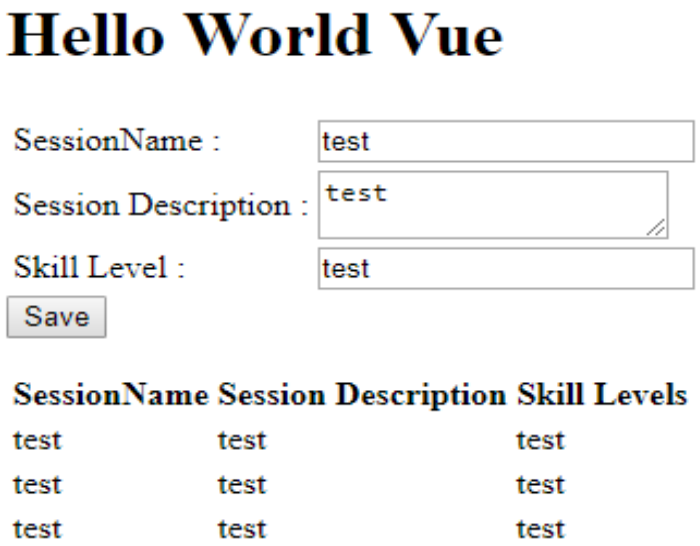
- v-for can also be used in range.
- v-for Loop to display iterative elements

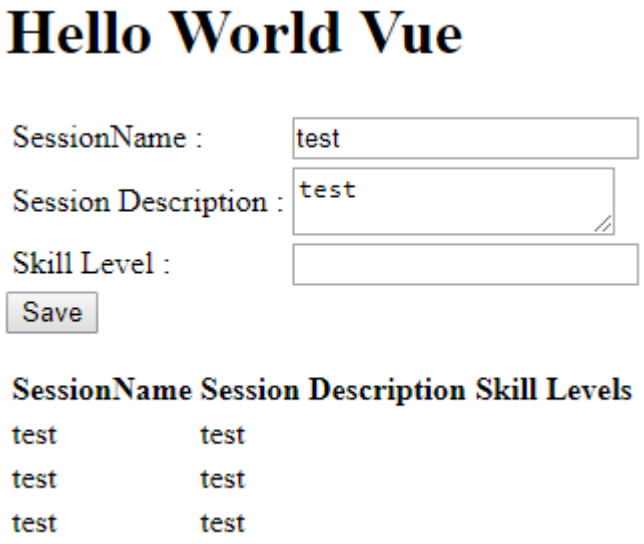
```
<table>
<tr v-for="item in listItems">
 <td>{{item.id}}</td>
 <td>{{item.productName}}</td>
</tr>
</table>
</div>
</div>
<script>
var application = new Vue({
 el:"#app",
data:{

    headerText: "Hello World Vue",
    secondText:"Second Header",
    textvalue :"Sample",
    showbutton : true,
    showalertbutton : false,
    listItems:[],
},
methods:{
 loadItems : function()
{
    this.listItems.push({"id":"1", "productName" : "Test"});
}
```



Overview of Vue Framework

- v-model –Two way databinding
  - v-once can be used to enable one way binding
  - Uses v-bind behind the hoods
  - Used for form based input fields





Overview of Vue Framework

- v-model –Modifiers
  - Number Modifier
  - Trim Modifier
  - Lazy Modifier – This behaves similar to onblur.
  - Can combine multiple modifiers

```
<table>
<tr>
<td><label>SessionName :</label> </td>
<td> <input type="text" v-model.trim="sessiondetails.sessionName"></input></td>
</tr>
<tr>
<td><label>Session Description :</label> </td>
<td> <textarea v-model.lazy="sessiondetails.sessionDescription"></textarea></td>
</tr>
<tr>
<td><label>Skill Level :</label> </td>
<td> <input type="text" v-model.lazy.trim="sessiondetails.skillLevel"></input></td>
</tr>
</table>
```

Overview of Vue Framework

- Vue Components
  - Collection of Elements accessed by single line
  - Supports both Local and Global Components
  - Components to be defined before Vue Instance Creation.
  - Props are used to pass data from Root to Components.

```
<p>
<div>
<session-component v-bind:session="sessiondetails">
</session-component>
</div>
<input type="button"
v-show="showbutton"
@click = "saveSession(sessiondetails)"
 value="Save"></input>
</p>
```

```
<script>
const sessionComponent =
{
template:'<table><tr><td><label>SessionName :</label>
,
props:['session']
};

var application = new Vue({
el:"#app",
components : {'session-component':sessionComponent},
data(){
return{

    headerText: "Hello World Vue",
    showbutton : true,
    sessiondetails:
    {
        id :"",
        sessionName : "",
        sessionDescription : "",
        skillLevel :""
    },
    submittedSessions :[]
}
},
```

Overview of Vue Framework

- Vue Components
  - Content of components can be defined inline
  - Content of components can be defined in script

  tag of type text/x-template
  - These all holds good for small applications.
  - Single File Components with extension .vue can be used for large applications.

```
<session-component v-bind:session="sessiondetails" inline-template>
<table>
<tr>
<td><label>SessionName :</label> </td>
<td> <input type="text" v-model.trim="session.sessionName"></input></td>
</tr>
<tr>
<td><label>Session Description :</label> </td>
<td> <textarea v-model.lazy="session.sessionDescription"></textarea></td>
</tr>
<tr>
<td><label>Skill Level :</label> </td>
<td><input type="text" v-model.lazy.trim="session.skillLevel"></input></td>
</tr>
</table>
</session-component>
```

```
<script>
const sessionComponent =
{
props:['session']
};

var application = new Vue({
el:"#app",
components : {'session-component':sessionComponent},
data(){
return{

   headerText: "Hello World Vue",
   showbutton : true,
   sessiondetails:{
   sessionName:"",
   sessionDescription:"",
```

Overview of Vue Framework

- Custom Events
  - Custom Events can be used to pass data from child to parent if used with v-on directive.

```
<submitted-sessions :submitted="submittedSessions" :selected="selectedSessions"
v-on:updateselection="updateparentselection" inline-template>
<table>
<th>SessionId</th>
<th>Session Name</th>
<th>Session Description</th>
<th>Skill Level</th>
<th>Selected</th>
<tr v-for="sessions in submitted" :key="sessions.id" >
<td>{{sessions.id}}</td>
<td>{{sessions.sessionName}}</td>
<td>{{sessions.sessionDescription}}</td>
<td>{{sessions.skillLevel}}</td>
<td><input type="checkbox" :value="sessions.id" v-model="selected" v-on:change="updateselection(selected)" ></input></td>
</tr>
</table>
</submitted-sessions>
```

Overview of Vue Framework

- Custom Events
  - Custom Events can be used to pass data from child to parent if used with v-on directive.

```
const submittedsessionComponent =
{
  props:
  {
    submitted:Array ,
    selected:Array
  },
  methods:{
      updateselection(selectedsession)
      {
        console.log("child");
        console.log(selectedsession);
        if(selectedsession.length >0)
        {
            this.$emit('updateselection',selectedsession);
        }
      }
  }
};
```

```
updateparentselection(first)
{
  console.log("parent");
  var session, selectedids="";
  if (first && this.submittedSessions && this.submittedSessions.length > 0)
  {
    for(session in this.submittedSessions)
    {
      this.submittedSessions[session].selected = false;
      for(selectedids in first)
      {
          if(first[selectedids] === this.submittedSessions[session].id)
          {
          this.submittedSessions[session].selected = true;
          }
      }
    }
  }
}
}
```

Vue Router and Vuex

Vue Router Overview

- Vue Router can be used to set up routing for Vue Applications
- Make sure to make mode - history

```
import Vue from 'vue';
import App from './App.vue';

import router from './router';

Vue.config.productionTip = false;

new Vue({
  render: h => h(App),
  router,
}).$mount('#app');
```

```
import Vue from 'vue';
import Router from 'vue-router';

import HomePage from '../Home/HomePage.vue';
import SessionsPage from '../Sessions/SessionPage.vue';


Vue.use(Router);

export default new Router({
  routes: [
    {
      name: 'Home',
      path: '',
      component: HomePage,
    },
    {
      name: 'Sessions',
      path: '/Sessions',
      component: SessionsPage,
    },
  ],
});
```

Vue Router Overview

- Router-Link can be used for navigation
- Router-view can be used to show component

```
<template>
  <div id="app">
    <header>
<nav>
  <ul>
      <li class="nav-item">
      <router-link class="nav-link" :to="{name :'Home'}" exact>
      <img class="logo" src="./assets/logo.png"/>
      Home </router-link></li>
      <li class="nav-item">
       <router-link class="nav-link" :to="{name :'Sessions'}" exact>Sessions</router-link>
      </li>
  </ul>
</nav>
    </header>
    <main>
    <div id="demogrid">
    <router-view></router-view>
   </div>
    </main>
  </div>
</template>
```

Vuex Overview

- Vuex is used for maintain state of application
- Stores state in a central location
- All changes to state are synchronous
- For small applications props and data can do the trick
- State is updated using mutations.

```
<script>
  const store = new Vuex.Store({
      state: {
        msg: 'Hello World',
        count: 0
      },
      mutations: {
     increment(state,payload) {
      state.count += payload;
  }
}
  });
  new Vue({
    el: '#app',
    data() {
      return {
        header: 'Vuex App'
      }
```

Vuex Overview

- For large applications it is recommended to have Vuex to enable cross component communication and sync state.
- Store.commit is used to update the state.
- Actions can be used update state asynchronously.
- Modules can be used to split the state into multiple objects.

- Further Learning – Refer to Vuex Documentation.

```
var application = new Vue({
  el: '#app',
  data() {
    return {
      header: 'Vuex App'
    }
  },
  computed: {
    welcome() {
      return store.state.msg
    },
    counter() {
      return store.state.count;
    }
  },
  methods: {
    increment() {
  store.commit('increment', 10)
}
  }
});
```

Native Script - Overview

- Native Script is free open source framework for building native IOS and Android Apps.
- Build Cross Platform mobile apps using single code base.
- Develop Mobile Apps using JavaScript , Angular , Typescript and Vue

Pre-Requisites of Native Script

- Prior Knowledge of HTML, CSS and any one of scripting language JavaScript , Angular, Typescript or Vue

- Node.js Server, Command Line Terminal and Preferred Text IDE Editor

- Android Studio, Android SDK and Android Emulator, Java SDK

Why Native Script

- Web Developers with knowledge of HTML, CSS and JavaScript can use the same to develop rich native mobile applications.
- Easily develop apps using existing plugins from npm [Node], Gradle [Android] and IOS Plugins.
- Easy to learn and develop apps using pre-defined templates.
- Angular supports Angular Schematics. One Code three displays – Web, IOS, Android
- NativeScript Vue only supports manual routing. Vue Router is not supported.

Native Script – App Structure

Native Script App Structure

- Any Native Script app contains the below folders
- Root – Project Name
- App Folder
- Node_Modules
- Package.Json
- Platforms

```
└── Project Name
    ├── app
    │   └── ...
    ├── node_modules
    │   └── tns-core-modules
    ├── package.json
    └── platforms
        ├── android
        └── ios
```

App Structure - Contd

- App_Resources

  This folder contains platform specific resources.

- Shared

  This folder contains files that needs to be shared across views

  In app.

- Views

  This folder contains the code to build apps views. Each view is made

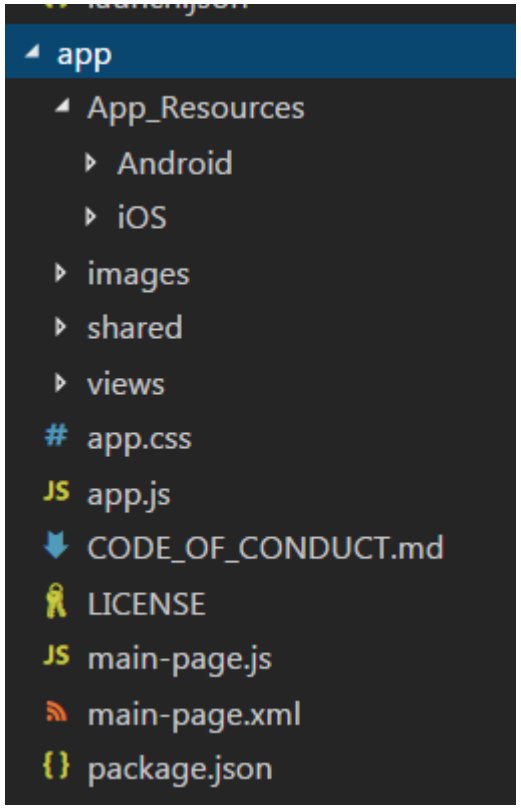
Up of XML File, a Javascript file and optional css file.

- App.css

  File contains global styles used in the app.

- App.Js

  This file sets up applications starting module and initializes the app.



Native Script CLI Basic Commands

- tns create appname --template template name

  The above command is used to create a native script app using Native Script CLI.

- tns platform add android

  This creates the android specific platform folder.

- tns platform add ios

  This creates the IOS specific platform folder.

Native Script CLI Basic Commands

- tns build android

  This command is used to build nativescript app using android platform

  --bundle is used only for Angular and Vue since it used webpack

- tns build ios

  This command is used to build nativescript app using ios platform

- tns run android --emulator

  This command is used to build and deploy your app using android emulator

- IOS related commands cannot be run in Windows OS. They can be used only in Mac.

Native Script CLI Basic Commands

- tns prepare android

  This command is used to update android platform folder with changes from app folder.

- tns prepare ios

  This command is used to update IOS platform folder with changes from app folder

- tns doctor

  This command is used to verify if all required components are setup in development machine.

Native Script CLI Basic Commands

- tns test init

  This command is used to initialize testing folder.

- tns test platform

  This command is used to run tests against the selected platform.

- tns debug platform

  This command is used to debug Nativescript apps using chrome dev tools

Native Script Controls

Native Script Basic Controls

HTML Control	Native Script Control
	LayOut – Stack Layout, Grid LayOut, Wrap LayOut, FlexBox Layout, Dock Layout
<input type="button" value="Sign In" onclick="signIn()">	<Button text="Sign In" @tap="signIn"/>
<label>Click me </label>	<Label text="name" />
<img src="mdn-logo-sm.png" alt="MDN">	<Image src="res://logo" ></Image>
<input id="email" type="email">	<TextField id="email" :text="email" keyboardType="email" autocorrect="false" autocapitalizationType="none" />

Native Script Basic Controls

Image

```
<!-- Load image from app/App_Resources/<platform> folders-->

<Image src="res://logo_white_bg" stretch="none" class="img-rounded p-l-15 p-r-15 p-t-15"></Image>

<!-- Load image from app/images folder -->

<Image src="~/images/logo.png" stretch="none" class="img-rounded p-l-15 p-r-15 p-t-15"></Image>

<!-- Load image from url -->

<Image src="https://docs.nativescript.org/img/NativeScript_logo.png" stretch="none" class="img-rounded p-l-15 p-r-15 p-t-15"></Image>
```

Native Script Basic Controls

Stack Layout

```
<StackLayout class="layoutBackgroundImageFromFolder">

<Button text="About" @tap="loadAbout" />

<Button text="Schedule" @tap="loadSchedule"/>

</StackLayout>
```

Stack Layout can be used to stack the controls vertical or horizontal similar to .

You guys are Awesome!!!

https://speakerdeck.com/reverentgeek

YOU ARE AWESOME

Dev Tools

- Visual Studio Code

  Install NativeScript PlugIn

  Create a native script app using tns create app command.

  Navigate to the app folder.

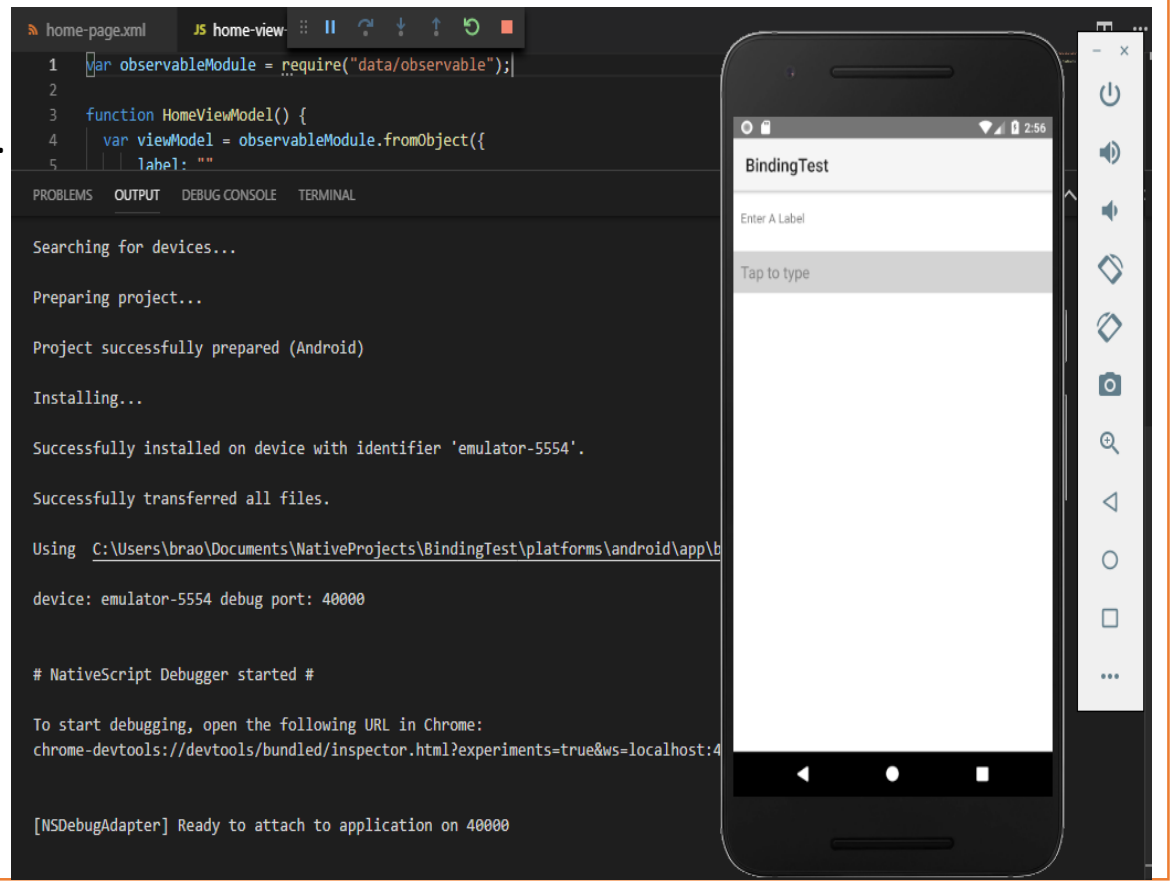
  Enter code . Command .

- Chrome Developer Tools

  Chrome Developer Tools can also be used to

  debug NativeScript Applications.

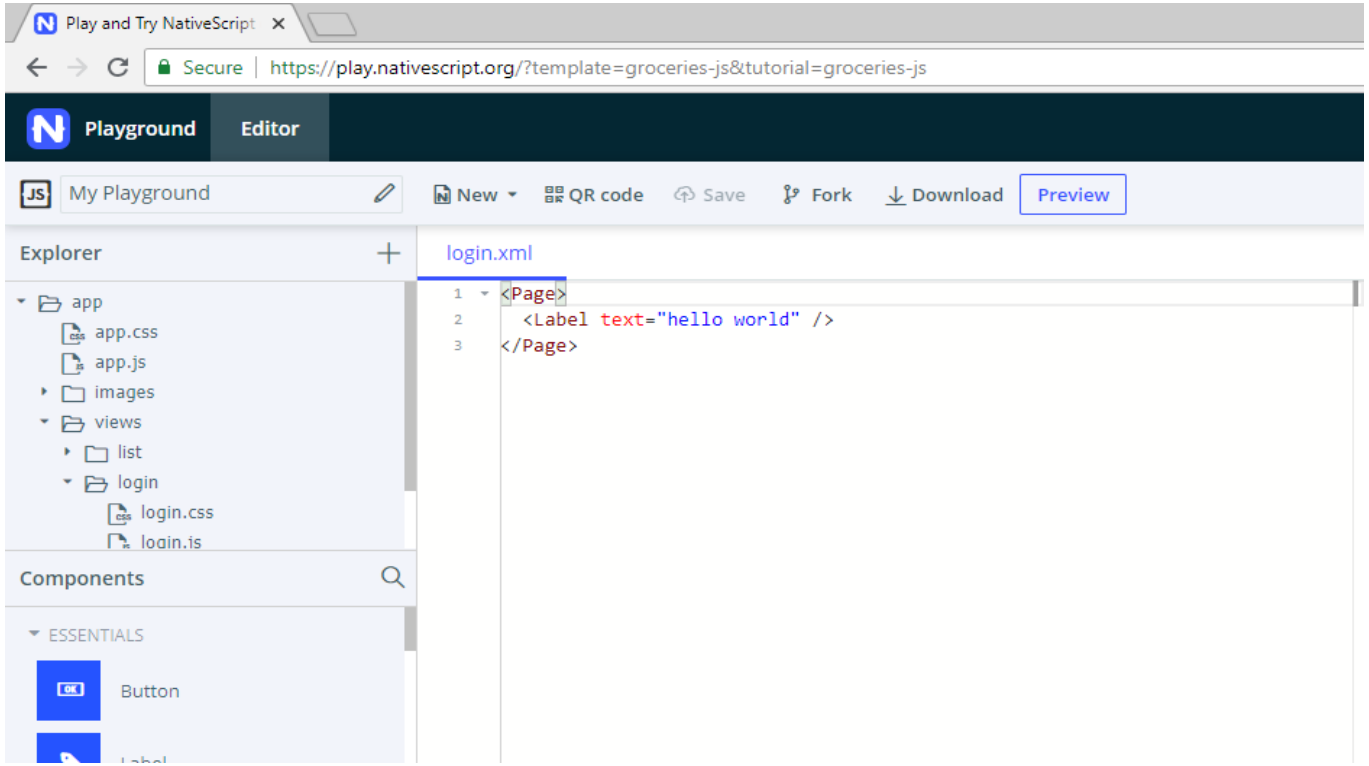
```
C:\Users\brao\Documents\NativeProjects>cd BindingTest

C:\Users\brao\Documents\NativeProjects\BindingTest>code .
```



Dev Tools

- Native Script Play Ground

Native Script Play Ground is a browser based platform to develop and preview mobile applications



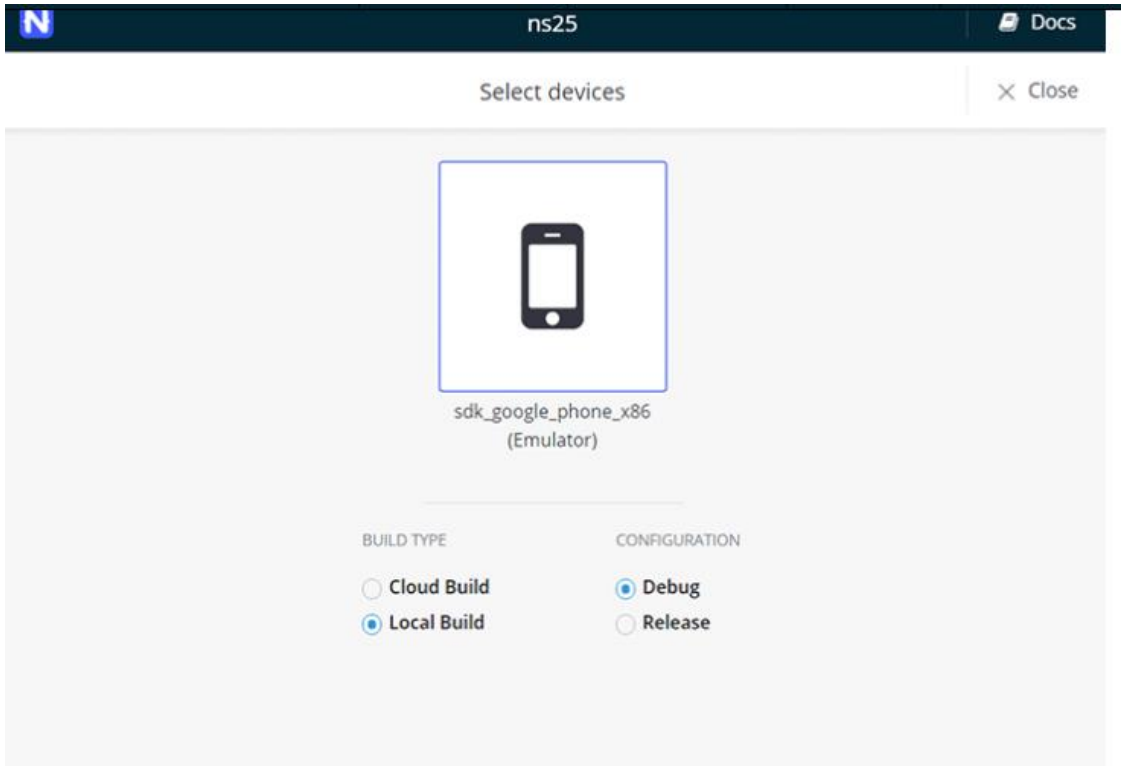
Dev Tools

- Native Script Side Kick

  Can be used to develop applications with pre-defined starter templates

  Perform cloud or local builds and deploy to test devices

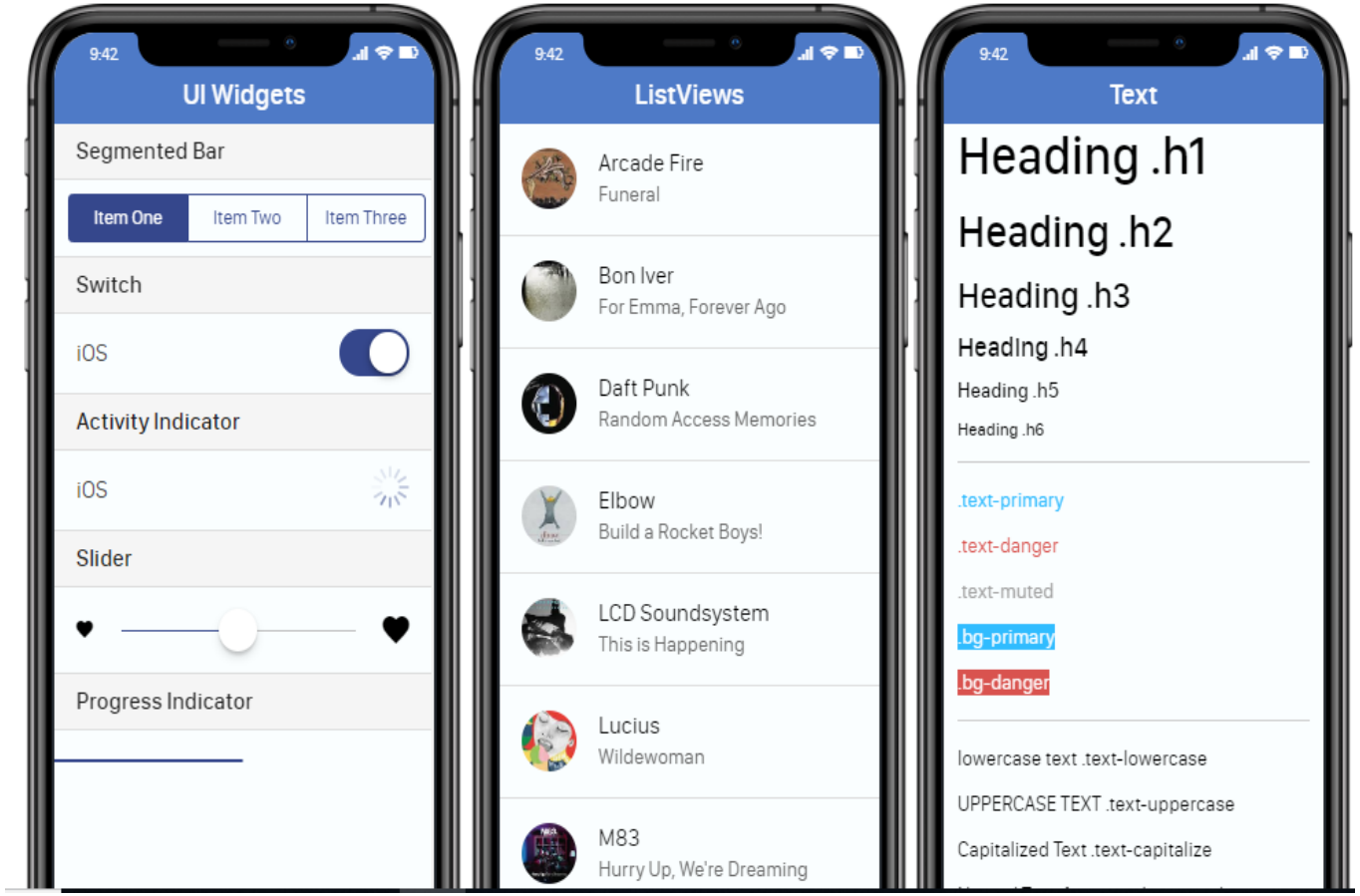
  Helps to develop IOS Applications on Windows O/S



Dev Tools

- NativeScript Theme Builder

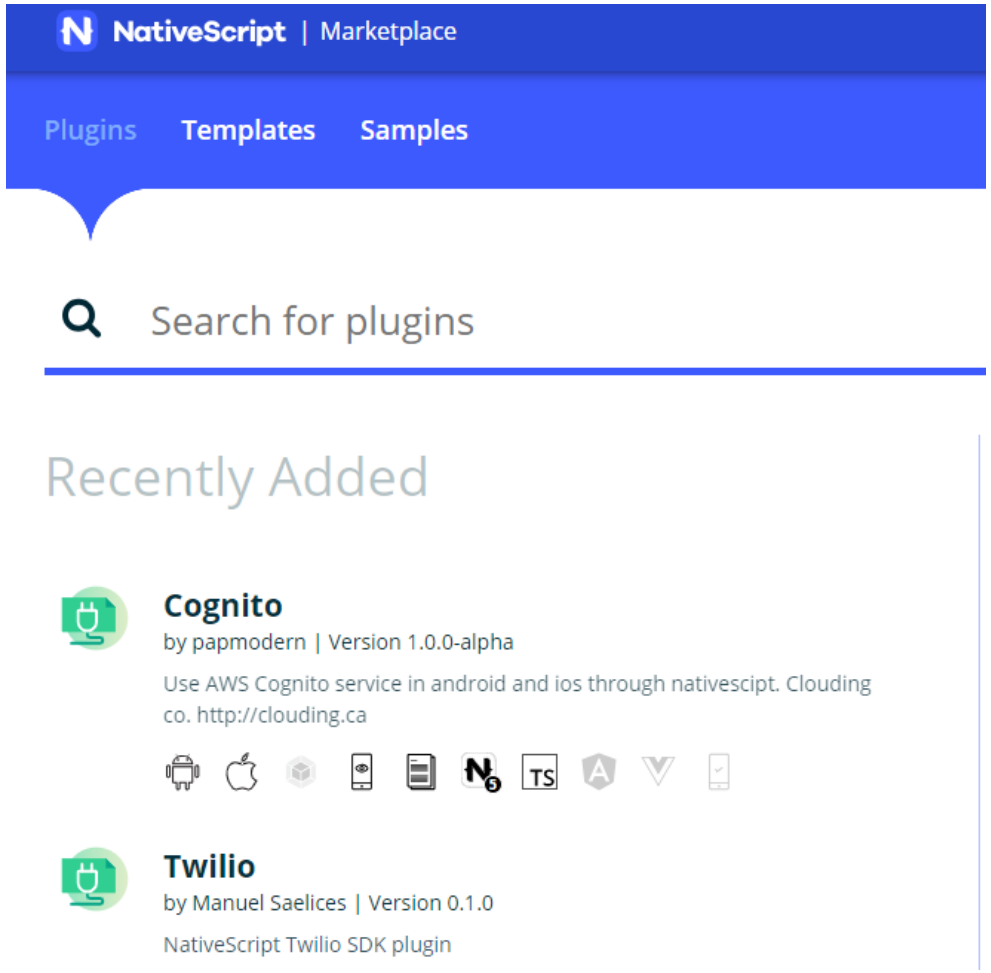
  https://www.nativescriptthemebuilder.com/



Dev Tools

- NativeScript Marketplace

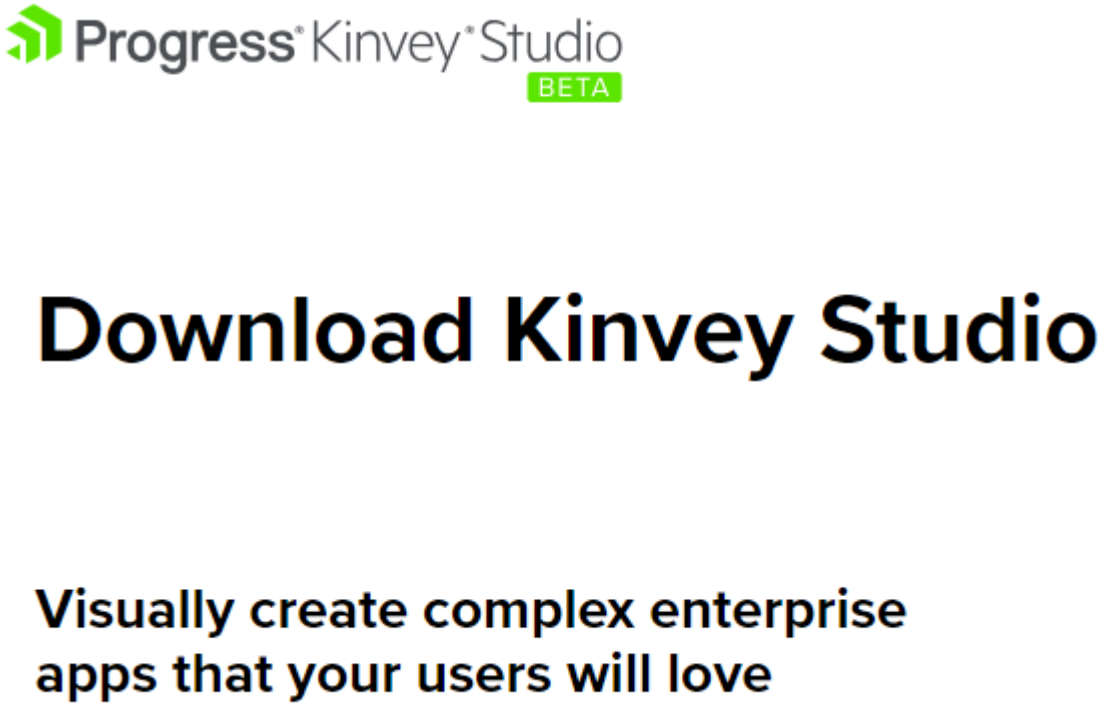
  https://market.nativescript.org/



Dev Tools

- Kinvey Data Studio

  https://studio.kinvey.com/



Walkthrough

- Demo of an existing app using Play Ground
- Demo live sync feature
- Demo Hot Module Reload using Visual Studio Code

Some Tips

- Before finalizing the Framework check if there are samples built in community already.
- Before starting development with NativeScript check if there are official verified plugins which you need to use.
- Any time stuck during development, deleting the platform folder and perform clean build.
- If Android Emulator does not start as expected after installation of NativeScript , try installing Android Studio and update Android Emulator through Android Studio.
- NativeScript Vue does not support controls which are based on DOM. We cannot use Vuetify in NativeScript Vue.
- While using NativeScript Vue occasionally Vue Dev Tools in some instances might not display debug information due to connection resets.
- Check with NativeScript Community of Developers in Slack Channel for any help during development.

Further Resources

- Vue Js Documentation
- Vuetify Documentation
- Native Script - Documentation
- Native Script eBook is free book by @brosteins available for download
- Native Script Blog
- Native Script Code Snippets
- Native Script Playground - Browser based development tool
- Native Script Sidekick – Useful for developing IOS apps on Windows Machine
- Native Script Vue Documentation
- NativeScripting Online Course
- NativeScript Vue CLI Code Sharing Plugin

Native Script Contributions

- https://bit.ly/2RDRnd6

Eligibility

Every *meaningful* first contribution counts. We reserve the right to evaluate what a "meaningful" contribution means, but to put it simply - adding whitespace or deleting a sentence doesn't count. Apart from that, you would be good to go. If you decide to write content - it should be your own, don't repost or copy other people's blog posts. Think about how your knowledge can be useful to other people.

Are You Excited Yet?

If you already have an idea - awesome! We can't wait to read about what you are up to. As soon as you are done with your work, fill in the NativeScript First-Time Contribution Program Form to claim your reward. If you are excited, but don't know where to start from - fear not. In the following weeks we are going to publish a series of blog posts with guidelines how you can contribute in one of the ways described above. Meanwhile, if you need support - you can jump in the special **#contributors-squad** channel in our Community Slack. We are there to help you.

JavaScript Themed Conference

https://www.javascriptandfriends.com/

Check out Auth0 and Auth0 Ambassador Sessions

Stephanie Chamblee

Title All About JWT's
Track Name Security and Best Practices
Timeslot Name 10:05-10:55AM

Bobby Johnson

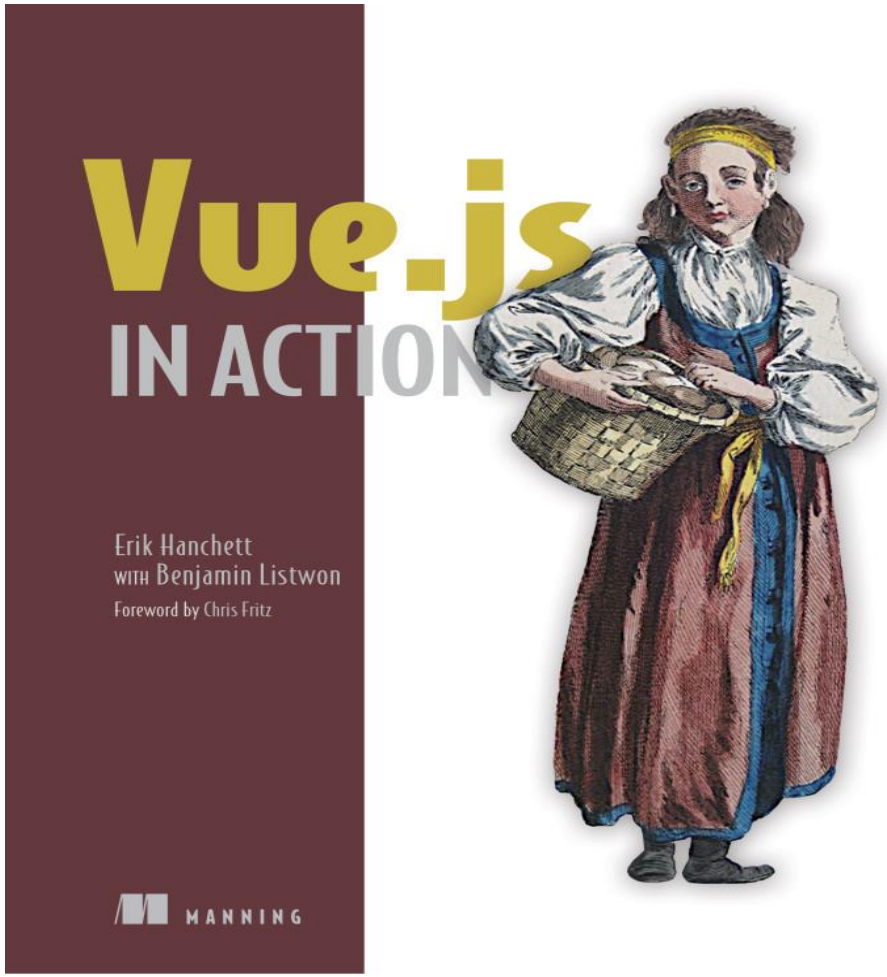
Title Identity 101: How Username/Password Got So Complicated
Track Name Security and Best Practices
Timeslot Name 1:00-1:50PM

Timothy Ferrell

Title Creating a gRPC service using Node
Track Name Cool Tech
Timeslot Name 2:00-2:50PM

Vue.Js in Action

- I recommend Vue.Js In Action book !!



Thank you Orlando Code Camp

Questions

https://www.linkedin.com/in/baskarrao-dandlamudi

baskarrao.dandlamudi@outlook.com

https://baskarrao.wordpress.com/

https://github.com/baskar3078/OrlandoCodeCamp2019