Webpack, Cordova & Electron: cross platform development by example

...

Troyes.js meetup - Axel Mousset

Outline

- Javascript, SPAs, PWAs
- Going native
- Tooling
- Dev stack example : webpack, cordova, electron

Javascript, a brief introduction

- Started as a “toy” language, now universally used
- Used to develop both server-side and client-side
- In 2016, StackOverflow survey stated “JavaScript is the most commonly used programming language on earth. Even Back-End developers are more likely to use it than any other language.”

Pro: **fast** growing community and ecosystem

Con: TOO FAST growing community and ecosystem

Single Page Applications (SPAs)

- GUI development is a tough subject
- The Web bring a sandbox (the browser) along with platform-agnostic langages (Html, css, javascript)
- Graphical apps, though not “native”, can be developed regardless of the OS
- The browser become a vessel, downloading and executing a remote application hosted on the web
- Not a “website” anymore

Progressive Web Applications (PWAs)

Progressive Web Apps are user experiences that have the reach of the web, and are:

- Reliable - Load instantly and never show the downasaur, even in uncertain network conditions.
- Fast - Respond quickly to user interactions with silky smooth animations and no janky scrolling.
- Engaging - Feel like a natural app on the device, with an immersive user experience.

(Source: developers.google.com)

Going native

Suppose we have a web-based app, how to make it a “real” app ?

- Interact with the host system (socket, GPS, usb, files...) where the browser doesn’t have an API for it
- Feeling native (invisible browser, custom icon...)

The desktop app case

- Browser are becoming more and more tied to the host OS
  (e.g: Mozilla web NFC API Draft, WebAssembly...)
- Although bridges exists, they're still not sufficient

Solution : wrap the web app and it's webview into a generic app, making the bridge with the OS

Electron

> “Build cross platform desktop apps with JavaScript, HTML, and CSS”

Electron brings a Node.js thread (Main process) along a WebKit view (Render process)

- From a development point of view, super smooth to work with
- Performance wise, a bit disappointing

The mobile case

- Embedding a webview and making the bridge with the host is a valid solution (Cordova)
- Mobile devices lack of performances led to more “native” solutions: React native, Native script, Weex ...
- However, native solutions are still immature

Tooling

Task runner, transpiler, compilers... javascript tooling has been created through sweat, tears and blood

Now, there's this nice guy doing everything: Webpack

- Used to bundle both Node.js, web and electron apps
- Watch, live reload, hot reload, minification....

A simple cross-platform development stack

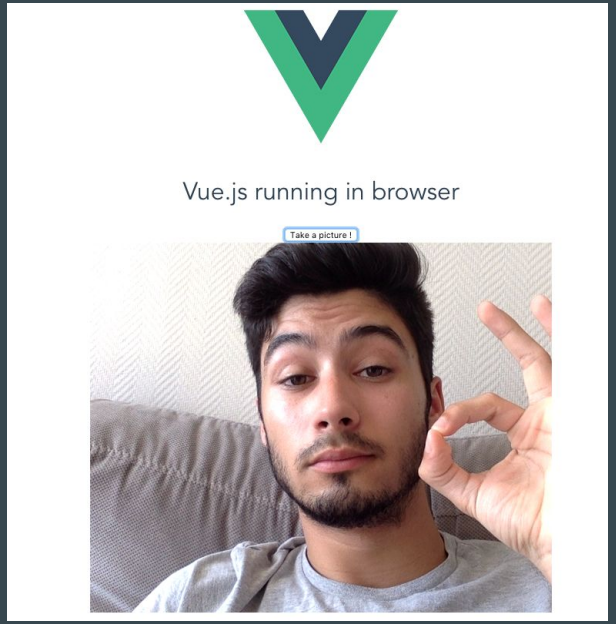
- The stack presented is an example and not a perfect magical recipe
- The goal is to have a single codebase and tooling for every platforms

Chosen stack:

- Vue.js app
- Electron for desktop
- Cordova for mobile
- Webpack for building everything (with live and hot reload)

The app

- Display the platform on which it’s running
- “Take a picture” button accessing the device camera
- Display the picture taken



Problem: host camera API

- Browser: modern ones all implement NavigatorUserMedia
- Electron: as it's based on webkit, the same API can be used
- Cordova: accessing the mobile camera cannot be done by WebKit.
  Solution -> make native calls using a cordova plugin (cordova-plugin-camera)

The exact same codebase cannot be used as the camera API differ depending on the target :'(:'(

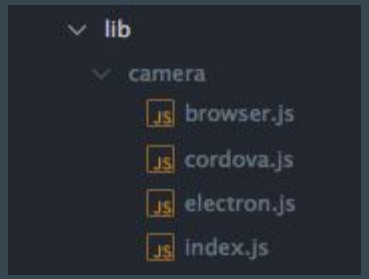
Solution: Webpack the cool guy

```
1 #if defined ANDROID
2   #include "android.h"
3 #elif defined BROWSER
4   #include "browser.h"
5 #endif
```

Similar to preprocessor directives in C, webpack can be used to make “find and replace” at compile time

Preprocessed require

```
1  export default require('./' + process.env.TARGET || 'browser')
2
```



Will be included in the final bundle, only the required files !

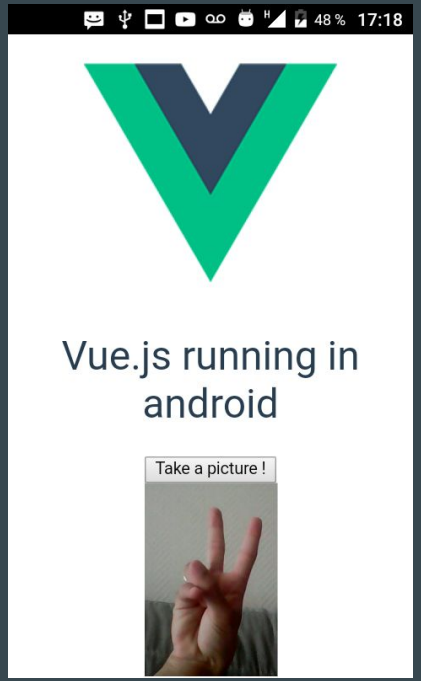
Browser | electron implementation

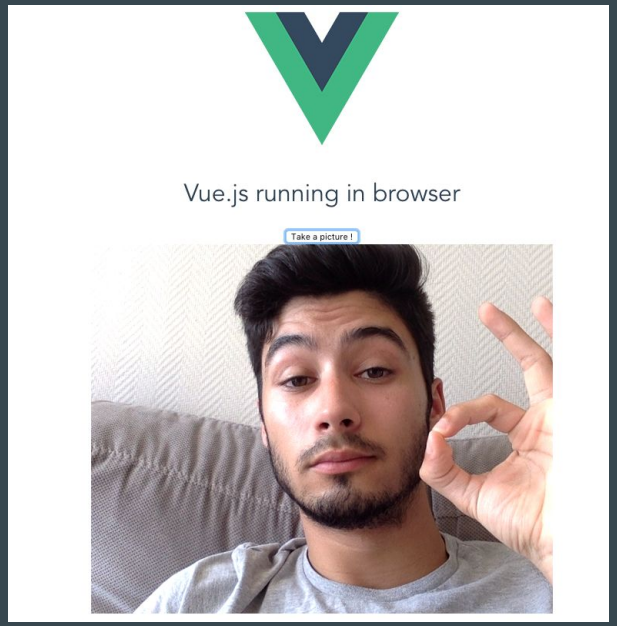
```
let img;
let video;
let canvas;

export function init(domElement, onFinished) {
  img = domElement;
  let parent = domElement.parentElement
  video  = document.createElement('video')
  canvas = document.createElement('canvas')

  video.style.display = 'none'
  canvas.style.display = 'none'
  canvas.width = 600
  canvas.height = 600
  parent.appendChild(video)
  parent.appendChild(canvas)

  navigator.getMedia = (navigator.getUserMedia ||
                        navigator.webkitGetUserMedia ||
                        navigator.mozGetUserMedia ||
                        navigator.msGetUserMedia)

  navigator.getUserMedia({ video: true }, (stream) => {
    video.src = window.URL.createObjectURL(stream);
    video.onloadedmetadata = function (e) {
      onFinished()
    }
  }, onFinished);
}

export function takePicture() {
  canvas.getContext("2d").drawImage(video, 0, 0);
  img.src = canvas.toDataURL("image/png");
}

```

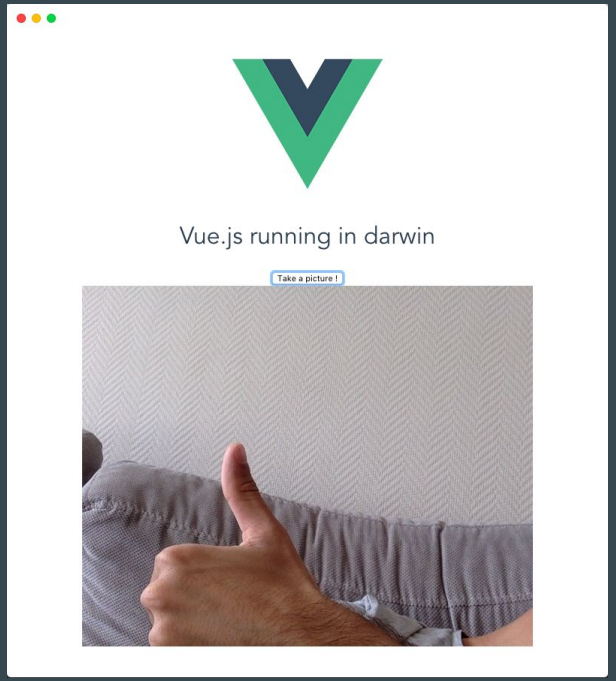
Cordova implementation

```
let img;
export function init(domElement, onFinished) {
  img = domElement;
  onFinished()
}

export function takePicture() {
    let options = {
      // Some common settings are 20, 50, and 100
      quality: 50,
      destinationType: Camera.DestinationType.FILE_URI,
      // In this app, dynamically set the picture source, Camera or photo gallery
      sourceType: Camera.PictureSourceType.CAMERA,
      encodingType: Camera.EncodingType.JPEG,
      mediaType: Camera.MediaType.PICTURE,
      allowEdit: true,
      correctOrientation: true  //Corrects Android orientation quirks
    }

    navigator.camera.getPicture(
      (imageUri) => img.src = imageUri,
      ()         => onFinished(new Error('Unable to optain picture: ' + error)),
      options
    )
}
```

Result


48 % 17:18
Vue.js running in android
Take a picture !


Vue.js running in browser
Take a picture !


Vue.js running in darwin
Take a picture !

Conclusion

- Smooth development process, using webpack-dev-server and hot reload to develop the main application
- Abstract peripherals and system calls in one library, preprocessing platform-specific requires using webpack DefinePlugin
- Performance-wise, cordova is not a killer.
  Let’s see how real native solutions evolves in the future

Sources

How it feels to learn javascript in 2016 - http://bit.ly/2d9UZAw

The javascript fatigue - http://bit.ly/1Ou6frX

Example project stucture - https://github.com/troyesjs/webpack-cordova-electron

Progressive web apps - https://developers.google.com/web/progressive-web-apps/