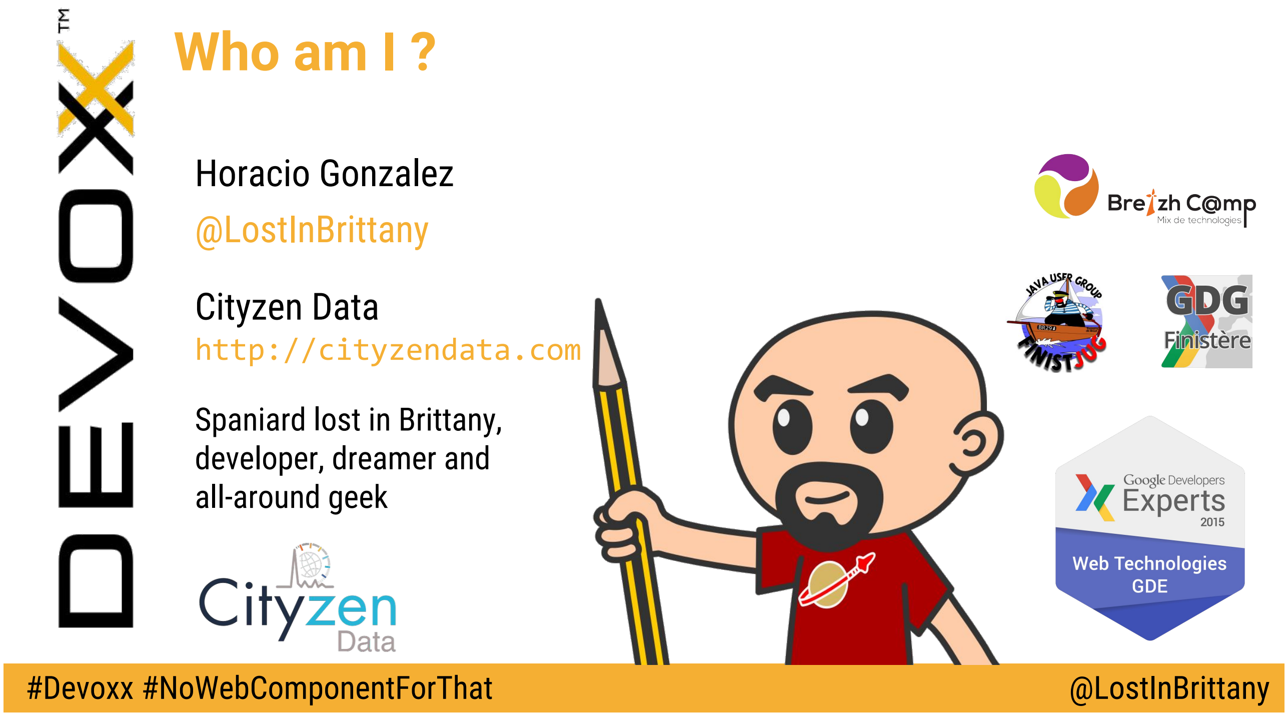
But there is no web component for that!

Horacio Gonzalez
@LostInBrittany



Who am I ?

Horacio Gonzalez

@LostInBrittany

Cityzen Data

http://cityzendata.com

Spaniard lost in Brittany, developer, dreamer and all-around geek

#Devoxx #NoWebComponentForThat @LostInBrittany

So, who am I? I'm Horacio, I'm a Spanish software developer living in France. I have been here for almost twenty years, so my English accent combines the worse traits of both the Spanish and French people. Yeah, I know, you are going to endure it for thirty minutes, yeah….

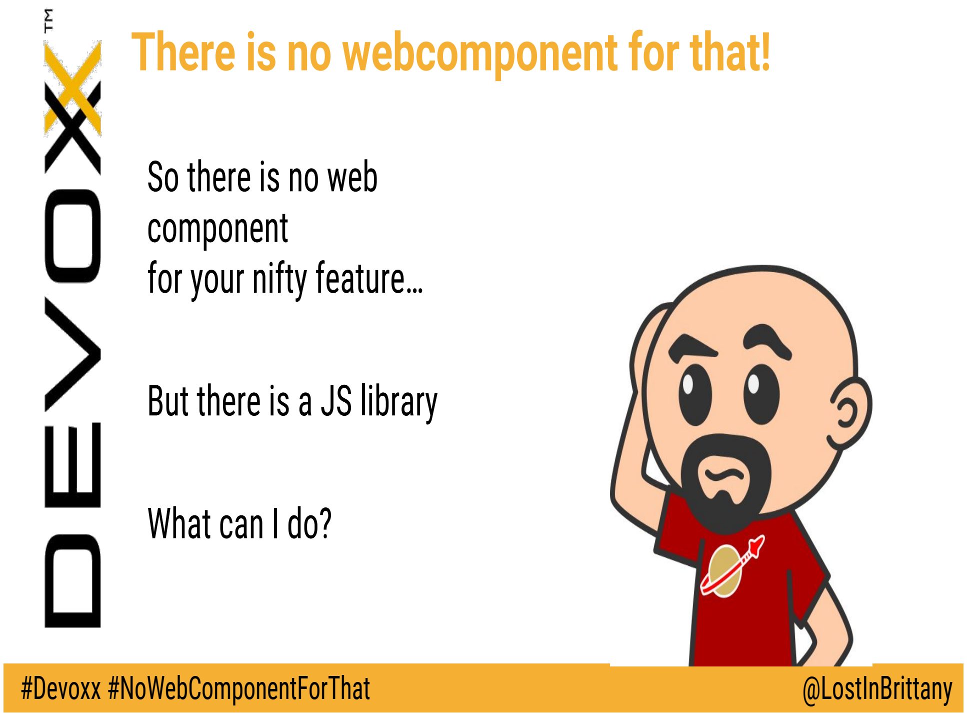
I'm rather active in the local developer ecosystem, the Java User Group, the Google Developers Group, I help to organize several local conferences. I also do open source code, teaching and speaking about web components and Polymer in quite a few conferences and meetups, being so active in the subject that last year Google appointed me as google Developer Expert in Web Technologies.

But my real work, the one that make me eat, is being the frontend guy for Cityzen Data, a startup doing an open source time series database, Warp10.

Introduction

Context is everything

So let's begin with a short introduction, because a bit of context is always useful. What am I going to talk about, and why. Because as you will see, the why is at the center of my love for the webcomponents standard...



There is no webcomponent for that!

So there is no web
component
for your nifty feature…

But there is a JS library

What can I do?

#Devoxx #NoWebComponentForThat @LostInBrittany

In this presentation I will talk you about that uncomfortable moment where you need a complex feature for your application but there is no webcomponent to do it. So you look around and you see a JavaScript library that do it well. And you wonder yourself what to do...

It isn't a theoretical problem

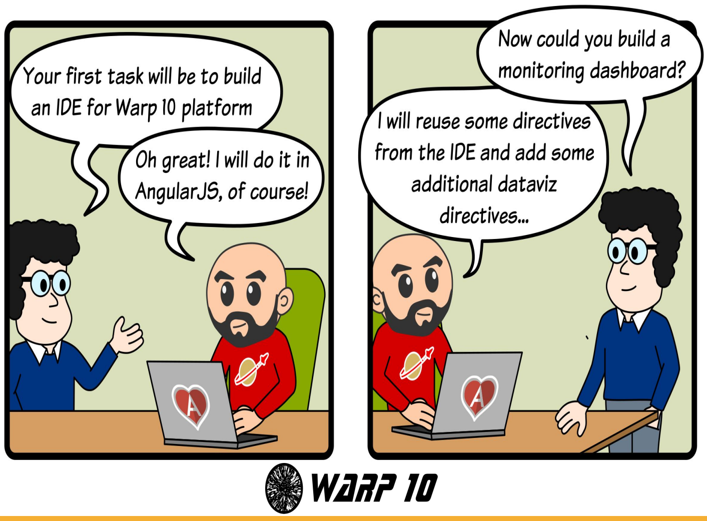
Maybe you won't never have this problem

And that's OK!

But I had the problem,
so I had to find an answer

I want to stress that it isn't a theoretical question, at least for me, it has been a real one since the first moment I worked on web components almost four years ago. You might never find this problem, and it's ok. Many developers will never code a webcomponent themselves, they will only integrate existing components into their Angular, React or whatever application. But I found the problem, and I really had to find an answer

I was kinda an AngularJS fanboy


Your first task will be to build an IDE for Warp 10 platform
Oh great! I will do it in AngularJS, of course!
Now could you build a monitoring dashboard?
I will reuse some directives from the IDE and add some additional dataviz directives...
WARP 10

#Devoxx #NoWebComponentForThat @LostInBrittany

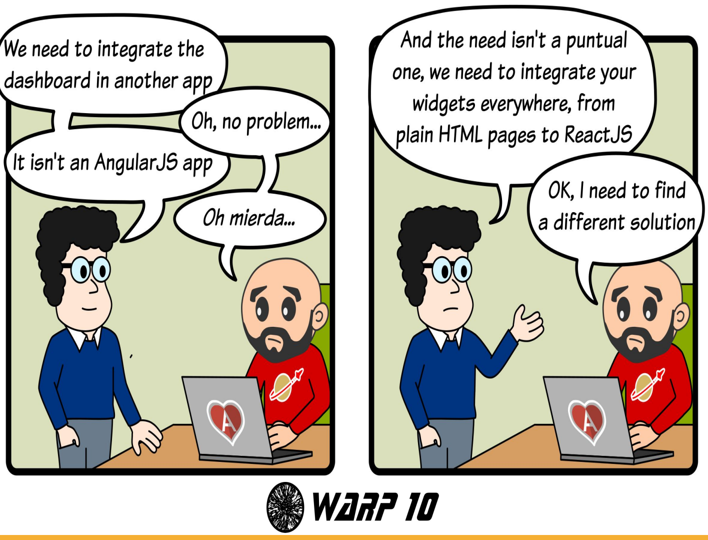
Let's go back to 20-13. I was an AngularJS fanboy. After having done GWT for years, Angular was like a sunshine in a cloudy day. I had done many apps in Angular, I loved how easy it was to deal with big apps.

I was (and I still am) the frontend guy in our startup, Cityzen Data. We work on a time-series database, Warp10, and I did the frontend apps. One day my boss asked me to do a web IDE for Warp10, letting people to write their analytical queries, interact with the store, get the results and manipulate them graphically. Of course, I chose Angular to do it, I did lots of directives and services, it was a nice app.

Some time later, my boss asked me for a monitoring dashboard, and once again I choose Angular, and I was able to reuse many of the directives, like those that interacted with Warp10 or that plotted the time-series.

Everything was great in the best of the worlds

Until I hit a wall

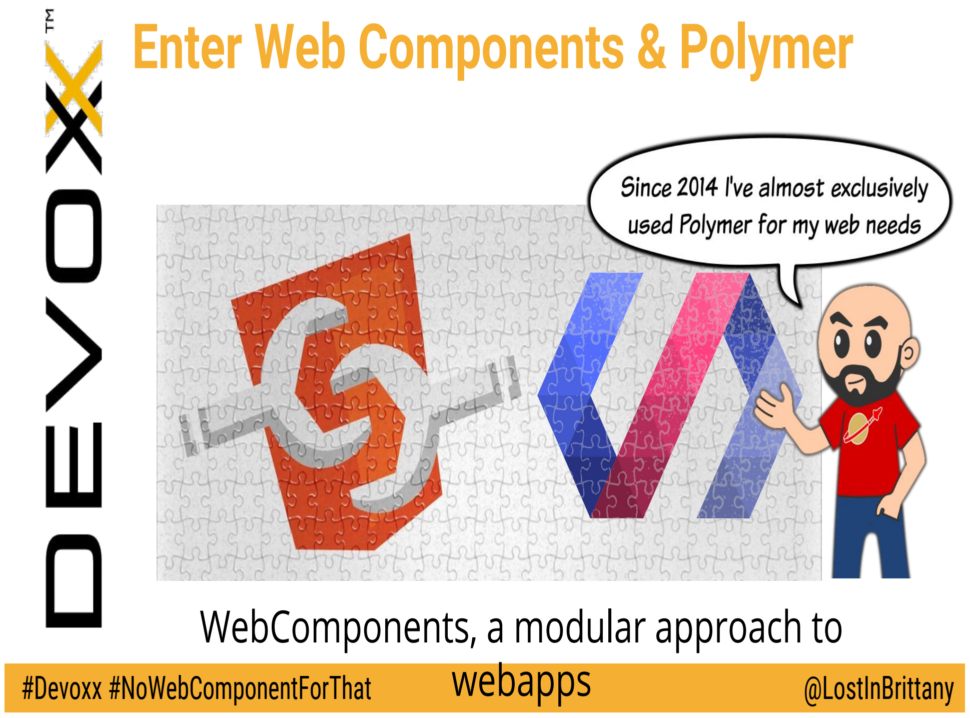

We need to integrate the dashboard in another app
Oh, no problem...
It isn't an AngularJS app
Oh mierda...
And the need isn't a puntual one, we need to integrate your widgets everywhere, from plain HTML pages to ReactJS
OK, I need to find a different solution
WARP 10

#Devoxx #NoWebComponentForThat @LostInBrittany

Until one day my boss told me that we need to integrate our dashboard in a customer's app… Oh, don't worry, I said, I will extract the directives and… Nope, guy, it won't be so easy - he told me - they aren't using Angular JS, it's something based on Ember JS. And then I only said BEEEEEEP.

O.K., so I did some Ember version of my dashboard, I didn't like it a lot but well, it worked. And then my boss came back and he told me bad news again. I needed to integrate the dashboard, but this time the need was some widgets to put in plain old HTML pages. And maybe in a React app some time later.

This time it was too much, I didn't want to spend my life recoding the same thing in every JavaScript framework, I needed to find another way...


Since 2014 I've almost exclusively used Polymer for my web needs

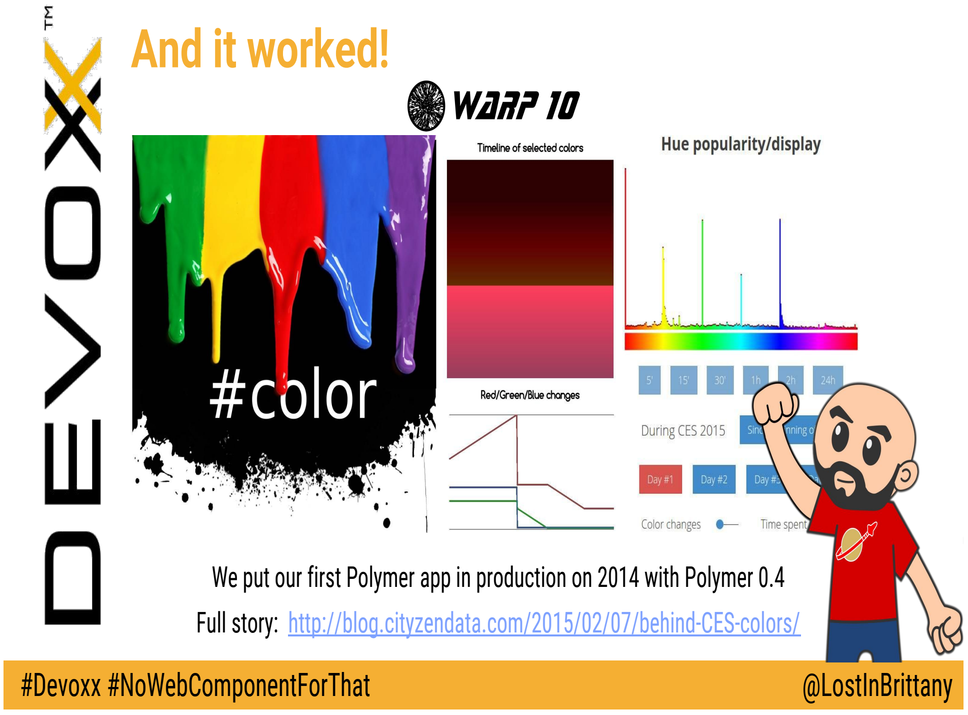
Enter Web Components & Polymer

WebComponents, a modular approach to webapps

#Devoxx #NoWebComponentForThat @LostInBrittany

And then I remembered a talk I had watched at Google IO 2013, about Web Components and Polymer. It seemed a good solution for my problem, I could re-code all my widgets one last time, as web components, and the I would be able to use them in any framework, from Angular to Ember to React to the next shiny JavaScript thing.

Well, in fact the web components standard was not finished yet, no browser implemented it besides Chrome, the polyfill was compulsory, Polymer was very young… but when you are in need you cannot afford to be picky, so I took Polymer 0.4 and I recoded my widgets with it.



And it worked!

We put our first Polymer app in production on 2014 with Polymer 0.4

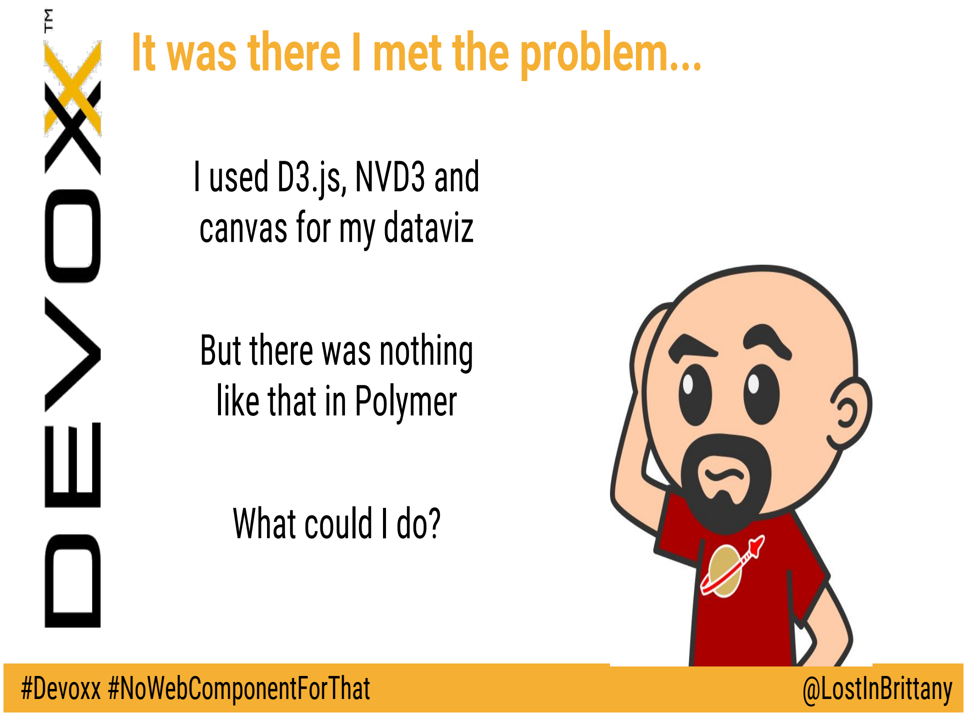
Full story: http://blog.cityzendata.com/2015/02/07/behind-CES-colors/

#Devoxx #NoWebComponentForThat @LostInBrittany

And it worked! O.K., it wasn't easy, and even some friends from Google told me it was a bad idea, and I was crazy. But in January 2014 I had a first Polymer app in production, in our stand at the CES at Vegas. And it ran on computers, tablets and phones; Windows, Linux, Mac, Android and iOS.

You have the full story in that address, you can go and read all the details.

It was a nice moment for me: it validated our approach, it set the course for our frontend development, it showed that it was possible to use web components in real life, even outside Google. But it wasn't easy...



It was there I met the problem...

I used D3.js, NVD3 and canvas for my dataviz

But there was nothing like that in Polymer

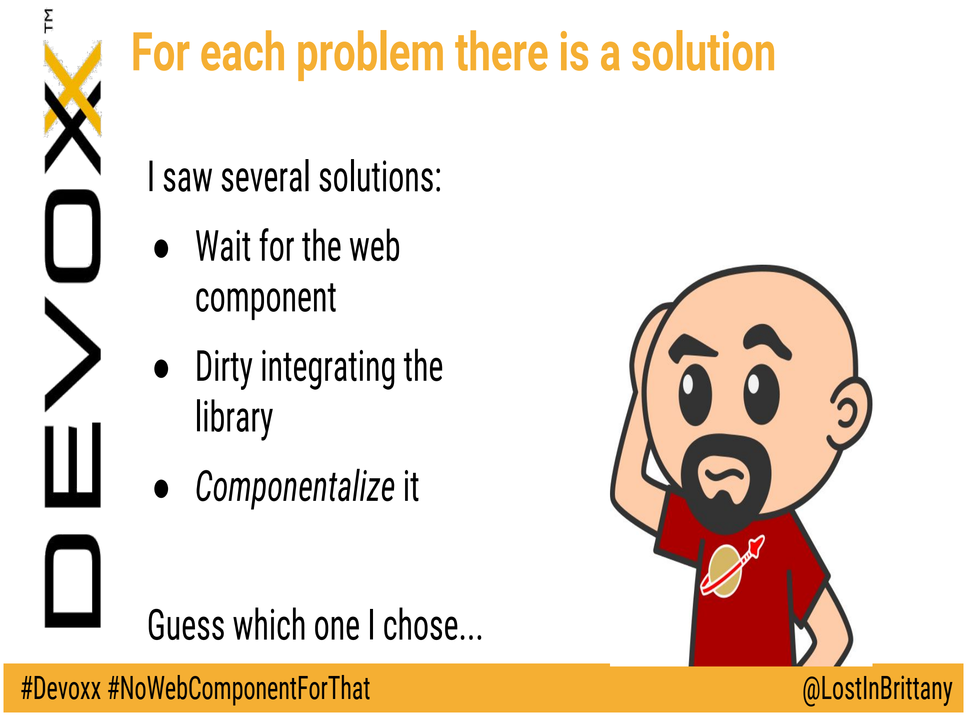
What could I do?

#Devoxx #NoWebComponentForThat @LostInBrittany

It was doing this first app that I met the problem. In my widgets I did a lot of time series dataviz, using mainly D3.js and NVD3 libraries, and some custom made libraries over canvas.

I looked around the early Polymer ecosystem, there was nothing to do dataviz.

So what could I do?



For each problem there is a solution

I saw several solutions:

- Wait for the web component
- Dirty integrating the library
- *Componentalize* it

Guess which one I chose...

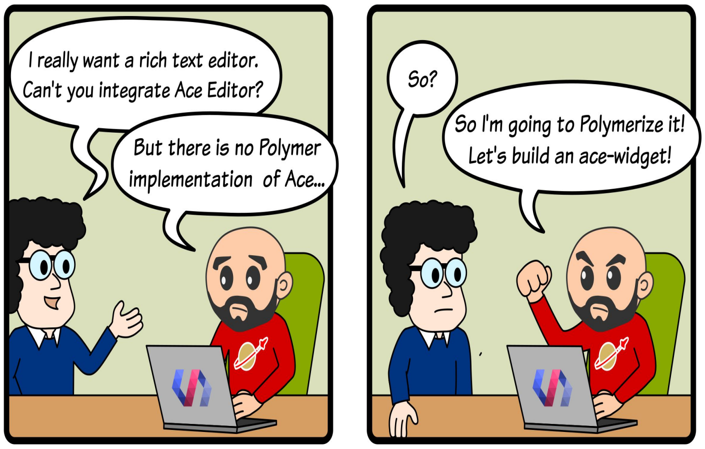
#Devoxx #NoWebComponentForThat @LostInBrittany

In my head I only saw three possible solutions:

- I could wait for somebody to implement a web component based on the library I needed, or on a similar one. It was a nice idea in theory, but I hadn't neither the time not the patience for that, I needed the feature, and I needed it quickly
- I could integrate the library in my code, in a quick and dirty way. I bit like the people who used jQuery in their Angular applications. Been there, done that, felt the pain. I wasn't going to do it!
- Or I could simply componentalize the library, wrap it in a clean web component structure, respecting the attribute-in event-out model and cleanly solving the problem once forever

Do I need to tell you which choice I took?

It was only the first time...

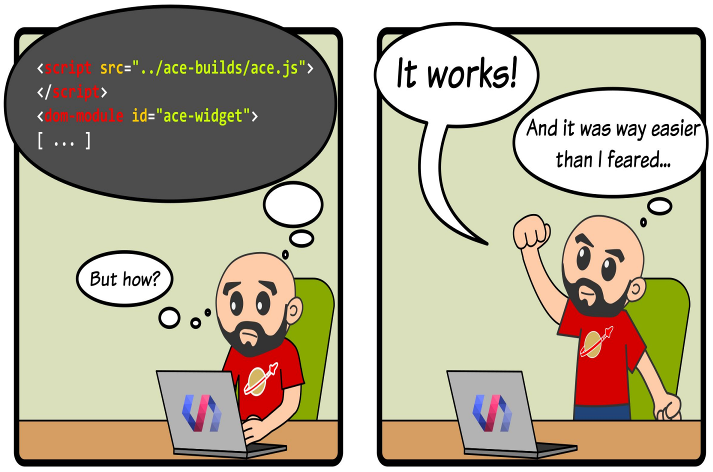

I really want a rich text editor. Can't you integrate Ace Editor?
But there is no Polymer implementation of Ace...
So?
So I'm going to Polymerize it! Let's build an ace-widget!

#Devoxx #NoWebComponentForThat @LostInBrittany

And the problem came back often, as many of the libraries I wanted to use hadn't a web component version yet. Like one or two months later when I needed a rich text editing widget based on Ace editor, or more recently when I looked for a QR Code generator…

And the only answer that I found was to com-po-nen-ta-lize them.

How do I *componentalize* them?



```
<script src="../ace-builds/ace.js">
</script>
<dom-module id="ace-widget">
[ … ]
```

WARP 10

#Devoxx #NoWebComponentForThat @LostInBrittany

The first time I had to componentalize a library it was rather complicated, because I didn't know exactly what I needed to do.

I took inspiration from some Polymer components, specially from iron-ajax, that wraps the native XHR (ajax) function in a component form, because it has the right mix of simplicity, richness of features, good component patterns (attribute-in, events-out) and clean code that make it a really nice example.

And then, quicker that I thought, it was done. And it worked! It wasn't as hard as I had thought, and I was a pleasure to use it in my apps, integrating the new component everywhere without problem.

Componentalizing a library

Let's begin with a simple example

#Devoxx #NoWebComponentForThat @LostInBrittany

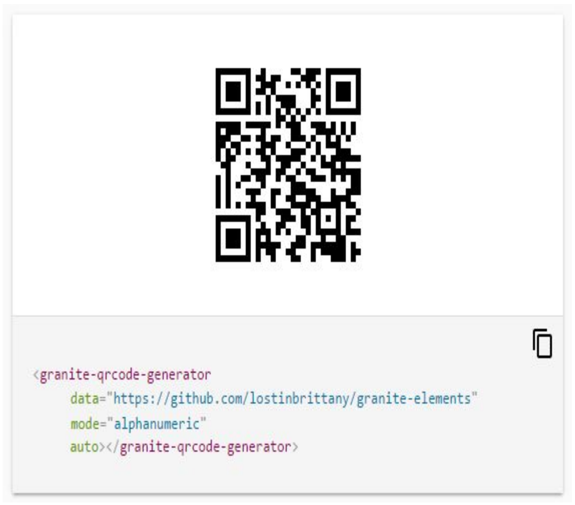
So let's see how do I tackle the problem of creating a web components wrapping a JavaScript library.

A quick disclaimer: I'm showing how to componentalize a library using Polymer. The process would be almost the same if you wanted to use vanilla web components, SkateJS, BramJS or any other web component library. The only thing that would change is the idiomatics and tooling of the process.

And I use the "I" because it's only one way to do it, my way. There are surely lots of other approaches, but this one works for me. Oh, and it's rather similar to what the Polymer teams guys do in some of their components, so I guess it isn't a bad approach.

Enough introduction, let's begin with an example.

granite-qr-generator



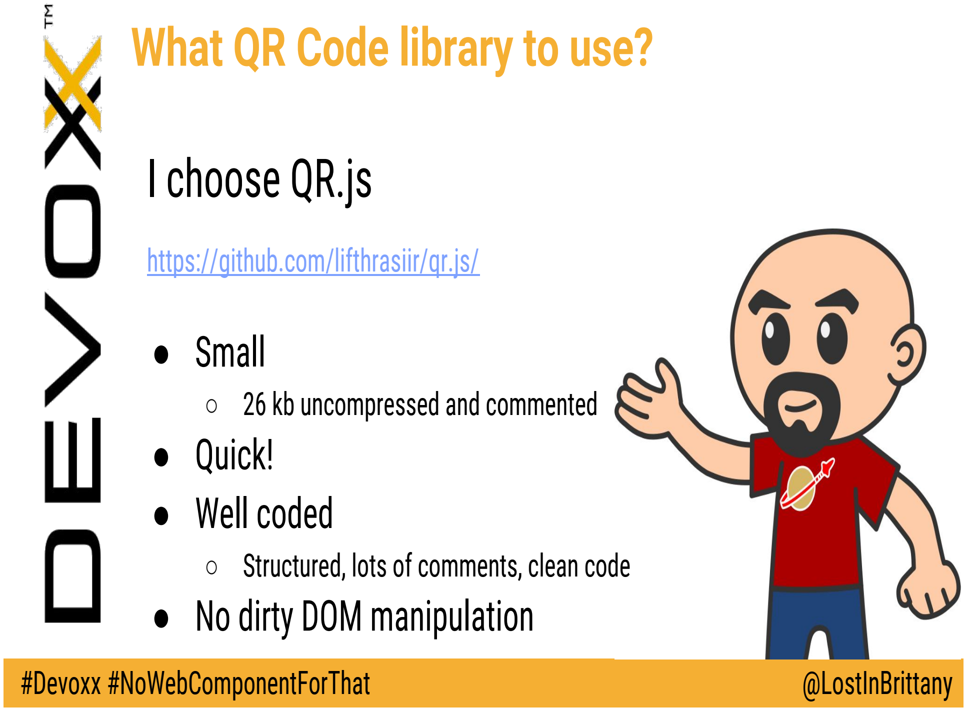
```
<granite-qrcode-generator
      data="https://github.com/lostinbrittany/granite-elements"
      mode="alphanumeric"
      auto></granite-qrcode-generator>
```

#Devoxx #NoWebComponentForThat @LostInBrittany

Let's begin with a small example, something I needed last summer, an element to generate a QR Code. I needed to be able to generate several types of QR Codes (alphanumeric, numeric, URL…), be interactive (if the data attribute changed, I needed the QR Code to actualize) and really simple to integrate, because it was a small need and I didn't wanted to lost time on it.

As I didn't find what I needed in the webcomponents' catalogs, I decided to do it myself. It was then that granite-qr-generator was born.

Why granite ? Because most Polymer element families seems to be named on materials : iron, paper, gold… So I wanted my components to be easily recognizable but following the same convention, and as I live in the granitic coasts of Brittany, I choose granite as material, and named my element collection granite-elements.



What QR Code library to use?

I choose QR.js

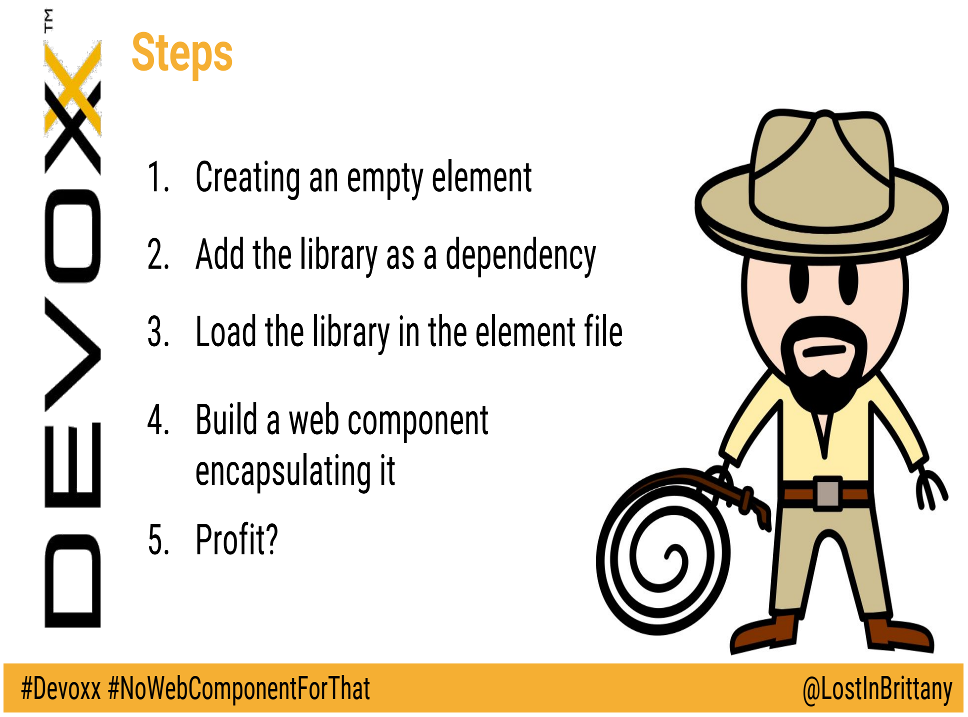
https://github.com/lifthrasiir/qr.js/

- Small
  - 26 kb uncompressed and commented
- Quick!
- Well coded
  - Structured, lots of comments, clean code
- No dirty DOM manipulation

#Devoxx #NoWebComponentForThat @LostInBrittany

I spent some time looking at the various QR code JavaScript libraries, trying to choose which one to use. The one I choose was qr.js, a rather old (2011) QR code generator in pure Javascript. It was small, quick, clean (well coded, structuted, with many comments…) and it seemed not to do any dirty DOM tricks.

I will tell you later about the dirty DOM tricks, one of the thing you will have to think about when componentalizing libraries.



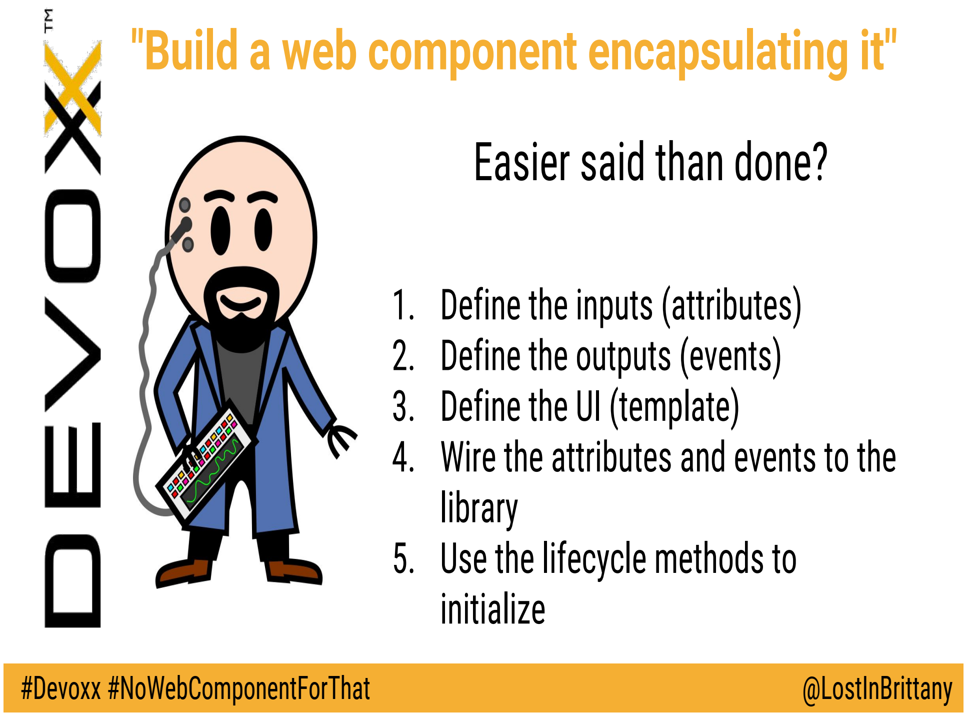
Steps

1. Creating an empty element
2. Add the library as a dependency
3. Load the library in the element file
4. Build a web component encapsulating it
5. Profit?

#Devoxx #NoWebComponentForThat @LostInBrittany

So, if I wanted to quickly describe the process of componentalizing a library, I would use that five steps:

1. Creating an empty Polymer element, either by hand or using the Polymer CLI
2. Add the library as a dependency to the bower.json
3. In your element, load the library, either in the global scope or in a local one if the library is modularized
4. Build a web component encapsulating the library, proposing the attribute-in events-out pattern that allows to interact with the library
5. Profit ? Or better said, enjoy your newly created element!



"Build a web component encapsulating it"

Easier said than done?

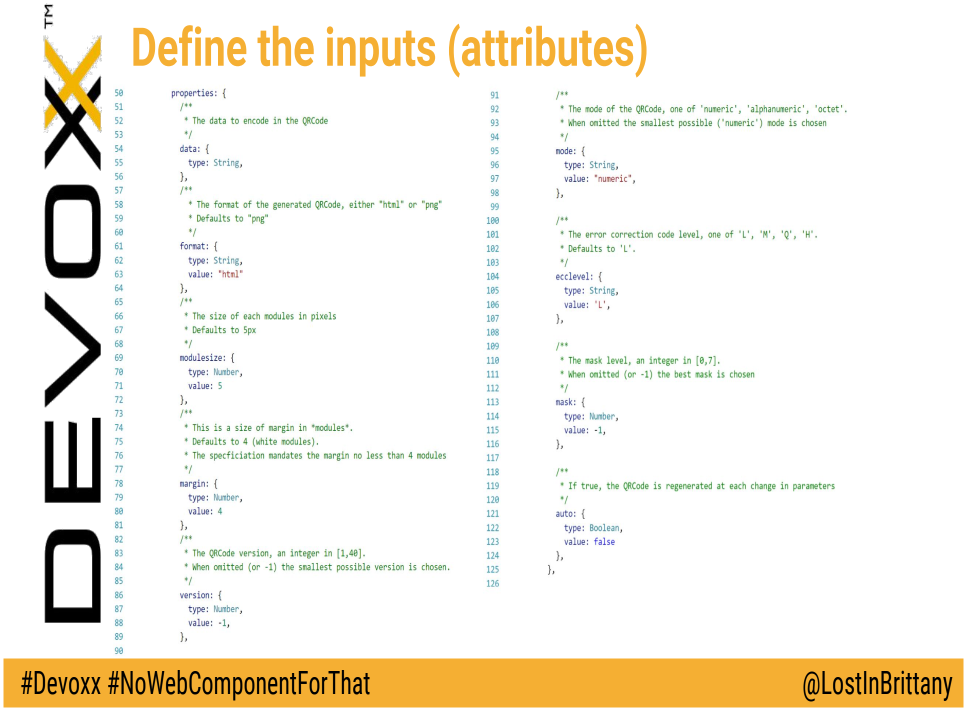
1. Define the inputs (attributes)
2. Define the outputs (events)
3. Define the UI (template)
4. Wire the attributes and events to the library
5. Use the lifecycle methods to initialize

#Devoxx #NoWebComponentForThat @LostInBrittany

"Build a web component encapsulating the library", that is really easy to say. But is it so easy to do? Well, to be fully honest, I would say that it isn't ALWAYS so easy... but in most normal cases it's really NOT difficult.My method:

1. Define the attributes, or in Polymer syntax the properties. Here you put the parameters needed to pilot your component from the outside, and if needed some properties to finely tune the library
2. Define the events that will be emitted by your component, if needed
3. Define the element's template
4. Wire the attributes and events to the library
5. Use the livecycle methods (ready and attached in Polymer) to initialize the library

Let's see how do we do it for our QR Code generator



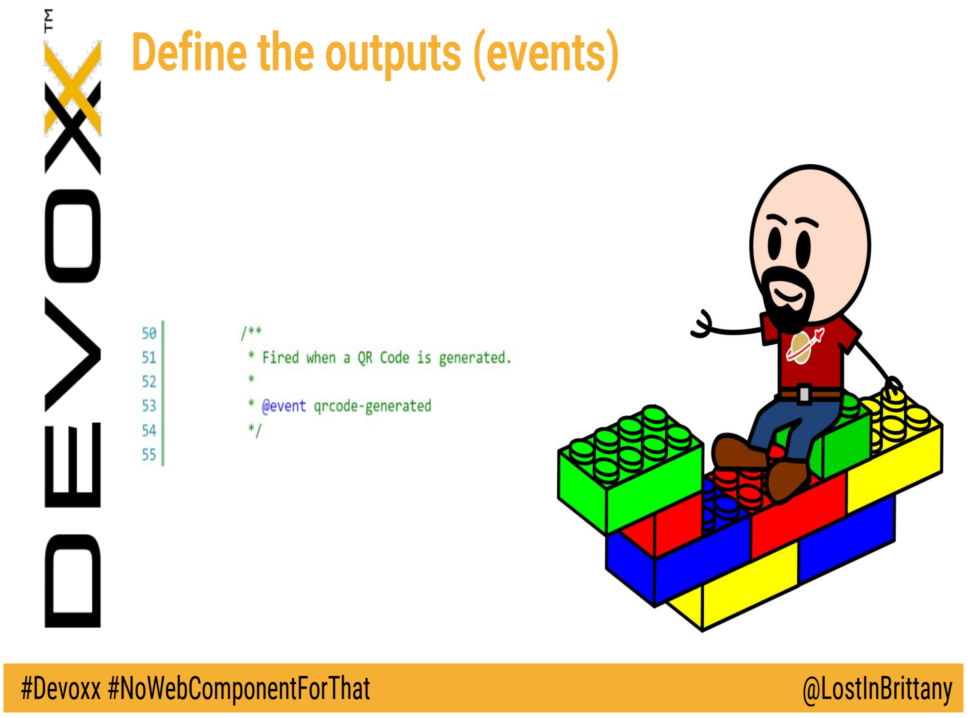
Define the inputs (attributes)

```
      properties: {
        /**
         * The data to encode in the QRCode
         */
        data: {
          type: String,
        },
        /**
          * The format of the generated QRCode, either "html" or "png"
          * Defaults to "png"
          */
        format: {
          type: String,
          value: "html"
        },
        /**
         * The size of each modules in pixels
         * Defaults to 5px
         */
        modulesize: {
          type: Number,
          value: 5
        },
        /**
         * This is a size of margin in *modules*.
         * Defaults to 4 (white modules).
         * The specficiation mandates the margin no less than 4 modules
         */
        margin: {
          type: Number,
          value: 4
        },
        /**
         * The QRCode version, an integer in [1,40].
         * When omitted (or -1) the smallest possible version is chosen.
         */
        version: {
          type: Number,
          value: -1,
        },

        /**
         * The mode of the QRCode, one of 'numeric', 'alphanumeric', 'octet'.
         * When omitted the smallest possible ('numeric') mode is chosen
         */
        mode: {
          type: String,
          value: "numeric",
        },

        /**
         * The error correction code level, one of 'L', 'M', 'Q', 'H'.
         * Defaults to 'L'.
         */
        ecclevel: {
          type: String,
          value: 'L',
        },

        /**
         * The mask level, an integer in [0,7].
         * When omitted (or -1) the best mask is chosen
         */
        mask: {
          type: Number,
          value: -1,
        },

        /**
         * If true, the QRCode is regenerated at each change in parameters
         */
        auto: {
          type: Boolean,
          value: false
        },
      },
```

#Devoxx #NoWebComponentForThat @LostInBrittany

So I looked at the QR.js library, searching for the input parameters it needed to generate the QR Codes. There were some 8 parameters, some of them needed every time it generates a QR code (like the data parameter, that is the data to encode), other are only used if we want to diverge from the behavior by default. I defined a property for each parameter, and I set the right by-default values according to the values proposed in the library. Thats allows you to use the component defining only the obligatory attributes if we want the by default result, or setting other attributes to finely tune the result.

Inspired by iron-ajax, I added an auto property to define if we want to generate a new code as soon as any parameter change (by default behavior with web components) or if the prefer to manually tell the element when to generate (by calling the generateQRCode method of the element,

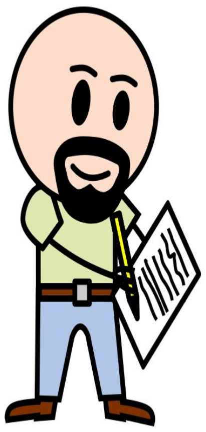


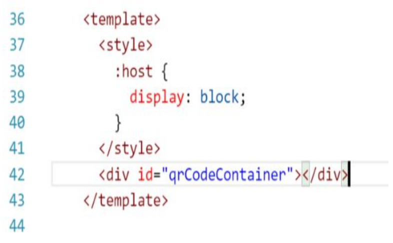
Define the outputs (events)

```
/**
 * Fired when a QR Code is generated.
 *
 * @event qrcode-generated
 */
```

Events are used by the element to signal the outside world that something interesting has happened. For our QR Code generator there is really no need to raise any event, besides maybe one when a new code is generated.

Define the UI (template)



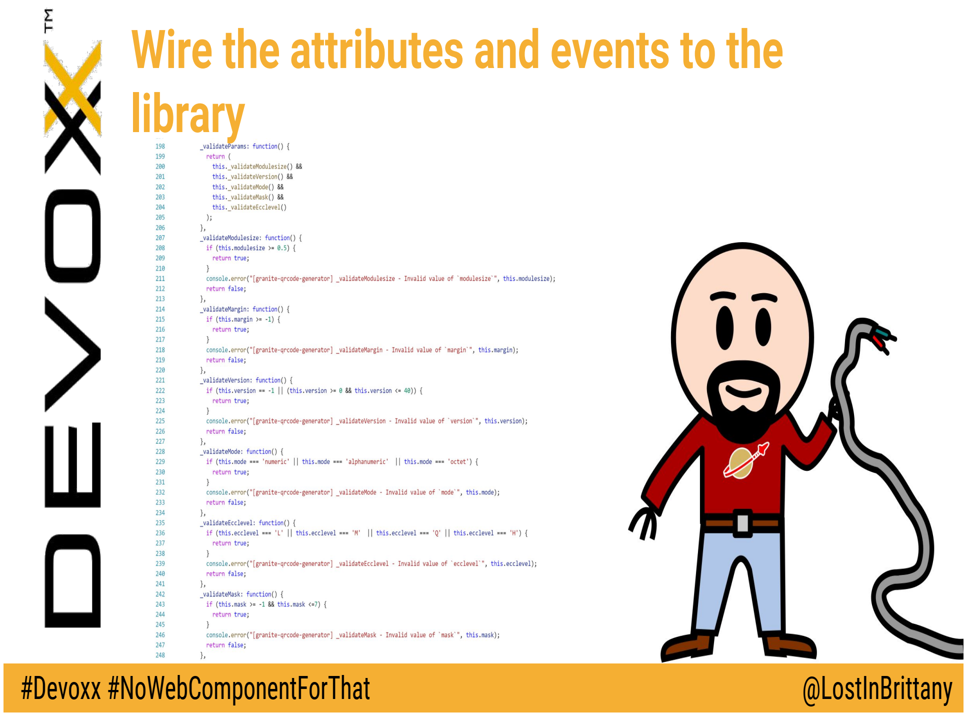


```
<template>
  <style>
    :host {
      display: block;
    }
  </style>
  <div id="qrCodeContainer"></div>
</template>
```

#Devoxx #NoWebComponentForThat @LostInBrittany

The template is really simple for our element, the only thing we need is an UI element, a DIV for example, that we will pass to the QR.js library and that it will use as root element for the QR Code.

Do you remember when I told you about playing fair with the DOM ? QR.js does it, it generates an HTML element or image element that you can directly append to the node you want to use as root. It doesn't ask neither for a selector, nor for an id or a classname. Libraries that ask for an id, for example, are difficult to use with web components, as they usually do a document.getElementById using the id, and it doesn't work inside the ShadowDom boundary.

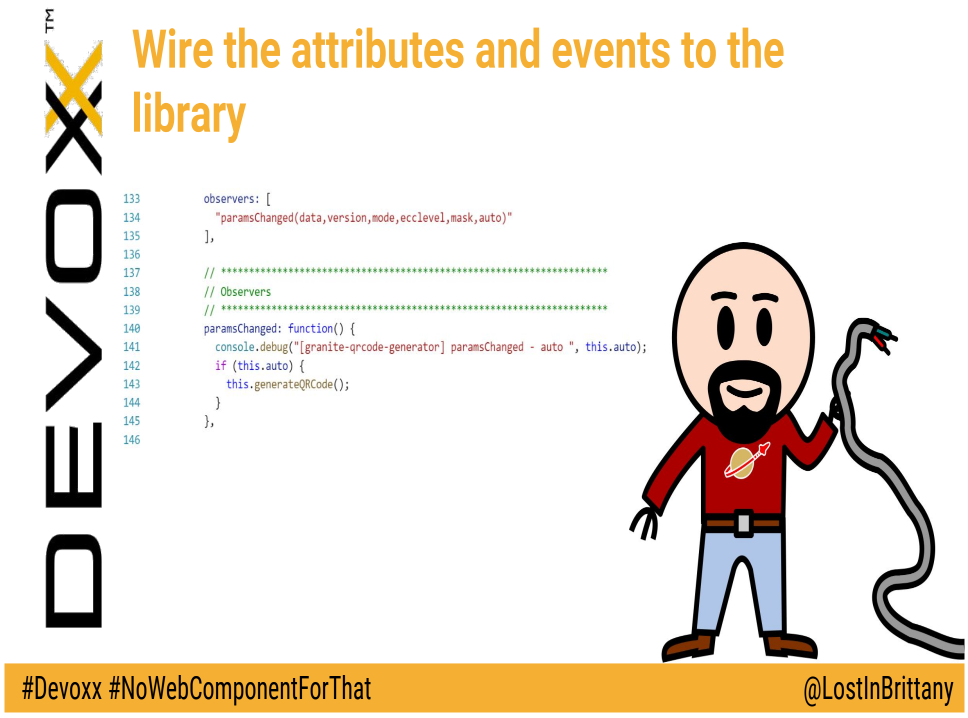


Wire the attributes and events to the library

```
    _validateParams: function() {
      return (
        this._validateModulesize() &&
        this._validateVersion() &&
        this._validateMode() &&
        this._validateMask() &&
        this._validateEcclevel()
      );
    },
    _validateModulesize: function() {
      if (this.modulesize >= 0.5) {
        return true;
      }
      console.error("[granite-qrcode-generator] _validateModulesize - Invalid value of `modulesize`", this.modulesize);
      return false;
    },
    _validateMargin: function() {
      if (this.margin >= -1) {
        return true;
      }
      console.error("[granite-qrcode-generator] _validateMargin - Invalid value of `margin`", this.margin);
      return false;
    },
    _validateVersion: function() {
      if (this.version == -1 || (this.version >= 0 && this.version <= 40)) {
        return true;
      }
      console.error("[granite-qrcode-generator] _validateVersion - Invalid value of `version`", this.version);
      return false;
    },
    _validateMode: function() {
      if (this.mode === 'numeric' || this.mode === 'alphanumeric'  || this.mode === 'octet') {
        return true;
      }
      console.error("[granite-qrcode-generator] _validateMode - Invalid value of `mode`", this.mode);
      return false;
    },
    _validateEcclevel: function() {
      if (this.ecclevel === 'L' || this.ecclevel === 'M'  || this.ecclevel === 'Q' || this.ecclevel === 'H') {
        return true;
      }
      console.error("[granite-qrcode-generator] _validateEcclevel - Invalid value of `ecclevel`", this.ecclevel);
      return false;
    },
    _validateMask: function() {
      if (this.mask >= -1 && this.mask <=7) {
        return true;
      }
      console.error("[granite-qrcode-generator] _validateMask - Invalid value of `mask`", this.mask);
      return false;
    },
```

#Devoxx #NoWebComponentForThat @LostInBrittany

In this step we begin by adding some validators, in order to verify the coherency of the input parameters, to sanitize the inputs. This step could be optional, as the library will usually do a similar thing, but there is a nice side effect in doing it here, it's to put the sanitized values back to the attributes, that makes things easier to debug from the outside of the element.

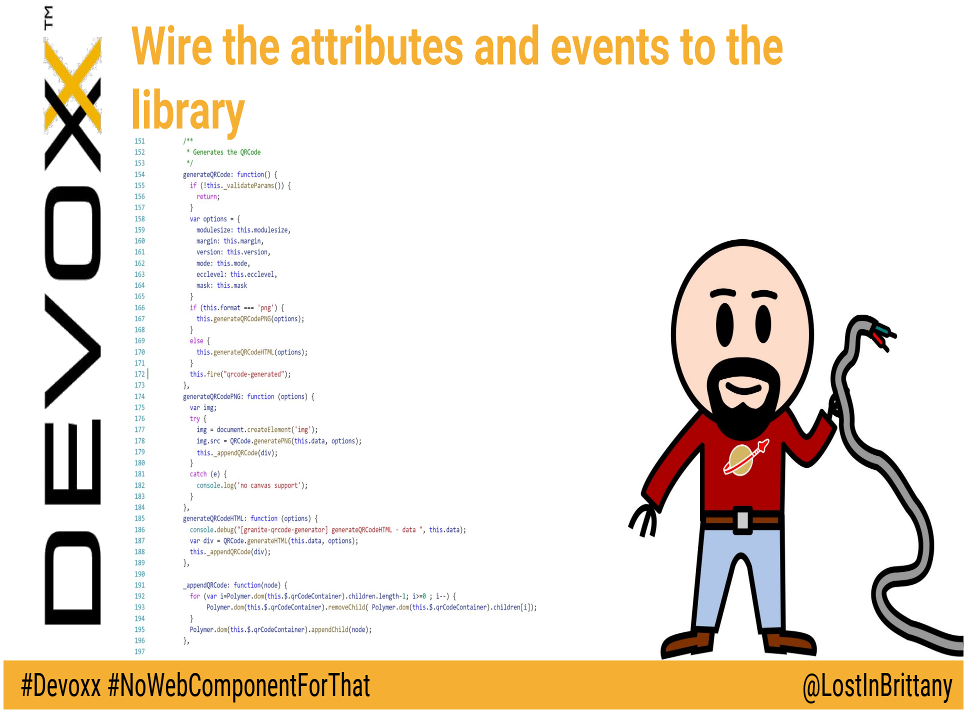


Wire the attributes and events to the library

```
observers: [
  "paramsChanged(data,version,mode,ecclevel,mask,auto)"
],

// ********************************************************************
// Observers
// ********************************************************************
paramsChanged: function() {
  console.debug("[granite-qrcode-generator] paramsChanged - auto ", this.auto);
  if (this.auto) {
    this.generateQRCode();
  }
},
```

#Devoxx #NoWebComponentForThat @LostInBrittany

An observer listen to the changes in the input parameters (the attributes) and, if the auto parameter is true, launches the generation of a QR code.
If auto is false, the only way to make the library generate a new QR code is to call directly the generatedQRCode method



Wire the attributes and events to the library

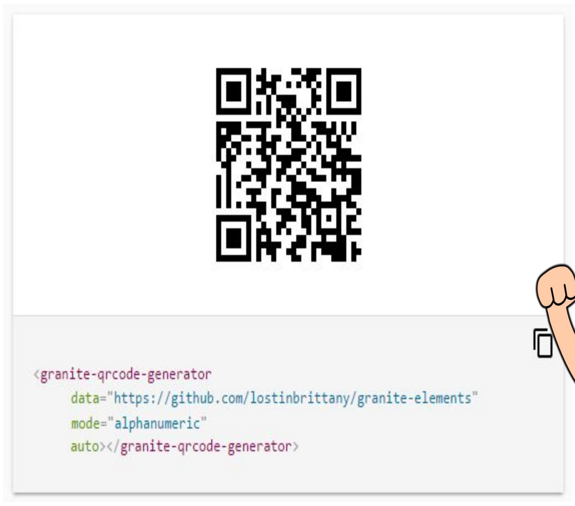
```
/**
 * Generates the QRCode
 */
generateQRCode: function() {
  if (!this._validateParams()) {
    return;
  }
  var options = {
    modulesize: this.modulesize,
    margin: this.margin,
    version: this.version,
    mode: this.mode,
    ecclevel: this.ecclevel,
    mask: this.mask
  }
  if (this.format === 'png') {
    this.generateQRCodePNG(options);
  }
  else {
    this.generateQRCodeHTML(options);
  }
  this.fire("qrcode-generated");
},
generateQRCodePNG: function (options) {
  var img;
  try {
    img = document.createElement('img');
    img.src = QRCode.generatePNG(this.data, options);
    this._appendQRCode(div);
  }
  catch (e) {
    console.log('no canvas support');
  }
},
generateQRCodeHTML: function (options) {
  console.debug("[granite-qrcode-generator] generateQRCodeHTML - data ", this.data);
  var div = QRCode.generateHTML(this.data, options);
  this._appendQRCode(div);
},

_appendQRCode: function(node) {
  for (var i=Polymer.dom(this.$.qrCodeContainer).children.length-1; i>=0 ; i--) {
      Polymer.dom(this.$.qrCodeContainer).removeChild( Polymer.dom(this.$.qrCodeContainer).children[i]);
  }
  Polymer.dom(this.$.qrCodeContainer).appendChild(node);
},
```

#Devoxx #NoWebComponentForThat @LostInBrittany

In the generatedQRCode method be gein by mapping the attributes to the input parameters of qr.js, an options object. Then, according to the chosen format (PNG or HTML), we call the right function of QR.js to generate the QR Code, either as an image or an HTML element. At last, we use Polymer DOM API to remove the ancient codes from the div and appending the generated code.

As no initialisation is needed, we don't need to add any code in the lifecycle, so our element is done.

granite-qr-generator

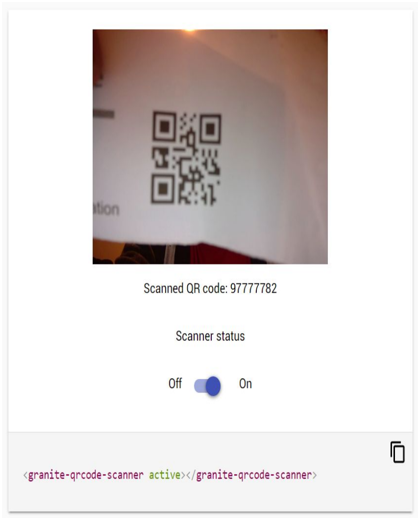


```
<granite-qrcode-generator
     data="https://github.com/lostinbrittany/granite-elements"
     mode="alphanumeric"
     auto></granite-qrcode-generator>
```

#Devoxx #NoWebComponentForThat @LostInBrittany

So the element is complete, and if you scan the QR Code currently in your screen you will go directly to the granite-elements home page, when you can find granite-qr-code and many other of its brothers and sisters

granite-qrcode-scanner



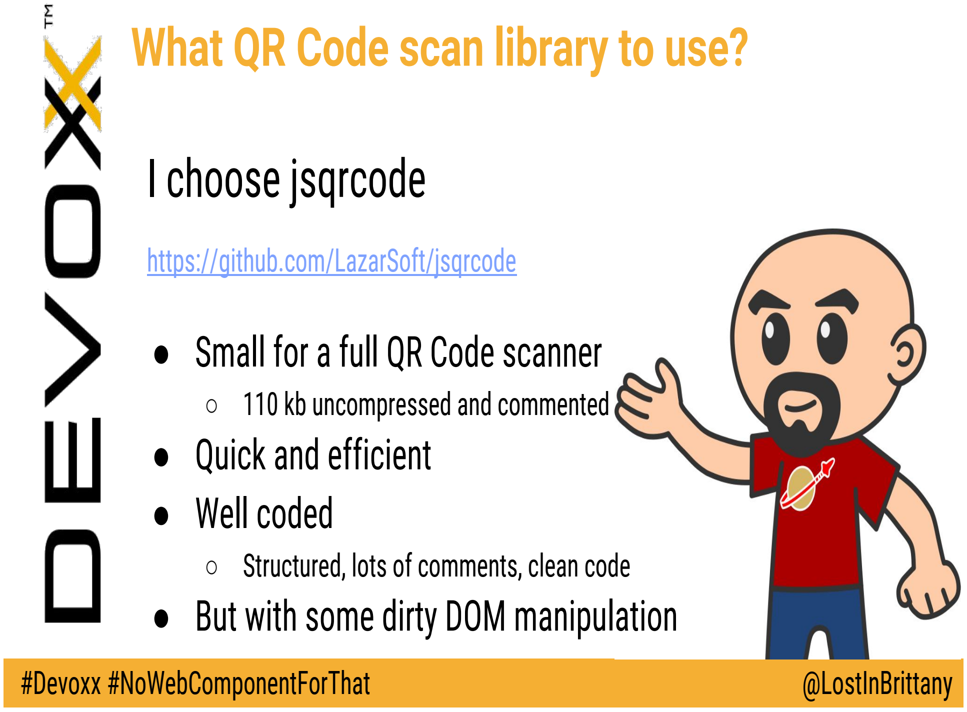
Scanned QR code: 97777782

Scanner status

```
<granite-qrcode-scanner active></granite-qrcode-scanner>
```

#Devoxx #NoWebComponentForThat @LostInBrittany

But what about libraries that don't play fair with the DOM?
Let's see the case of another of my granite elements, granite-qrcode-scanner.
The need came when I was asked to make a Progressive Web Apps for a conference, to replace their different native apps and their mobile site.
One of the functions of the app was to be able to scan the badges of other attendees in order to get their public contact information.
So I needed a QR Code scanner. Well, several months ago I had done the qr code generator, the scanner couldn't be more difficult, could it?



What QR Code scan library to use?

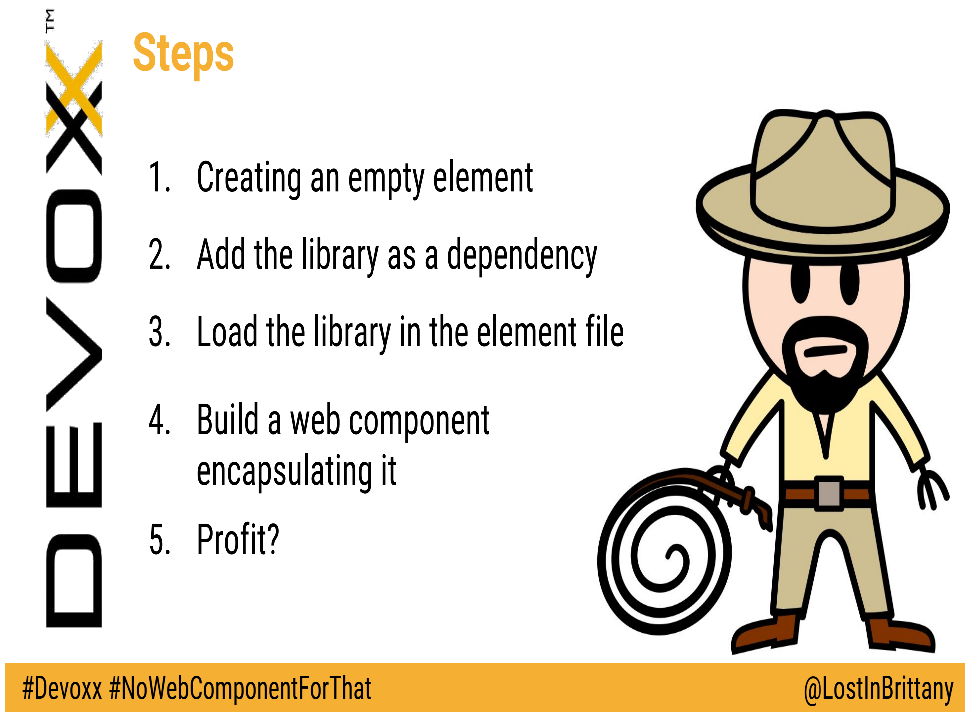
I choose jsqrcode

https://github.com/LazarSoft/jsqrcode

- Small for a full QR Code scanner
  - 110 kb uncompressed and commented
- Quick and efficient
- Well coded
  - Structured, lots of comments, clean code
- But with some dirty DOM manipulation

#Devoxx #NoWebComponentForThat @LostInBrittany

So, once again, I spent some time looking at the various QR code scan JavaScript libraries, trying to choose which one to use. The one I choose was jsqrcode, a rather old (2010) QR code scanner in pure Javascript. A port of the well known ZXing qrcode scanner. It was small for such a scanner, it was quick and efficient, (able to decode quickly and with good decoding rate), clean (well coded, structured, with many comments. But I hand't seen is that it did some dirty DOM manipulation...

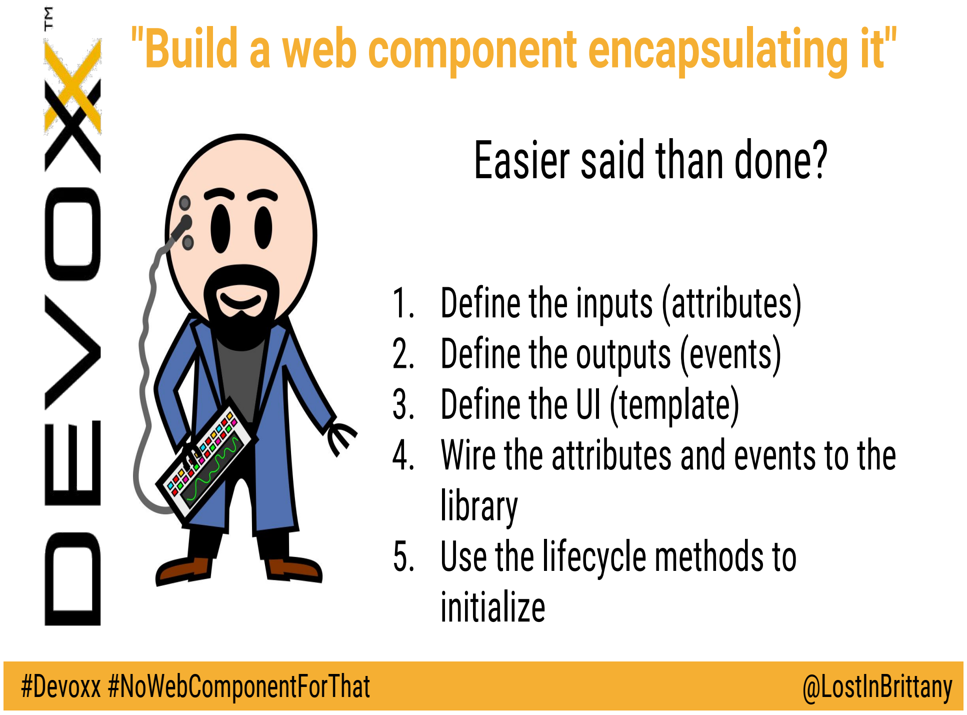


Steps

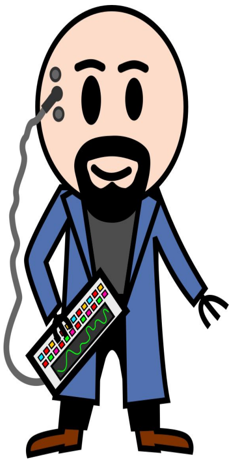
1. Creating an empty element
2. Add the library as a dependency
3. Load the library in the element file
4. Build a web component encapsulating it
5. Profit?

#Devoxx #NoWebComponentForThat @LostInBrittany

1. So I did the initial steps, creating the element, adding the library as a dependency, loading it in the element file…
2. And then I attacked the building of the component...
3.



"Build a web component encapsulating it"

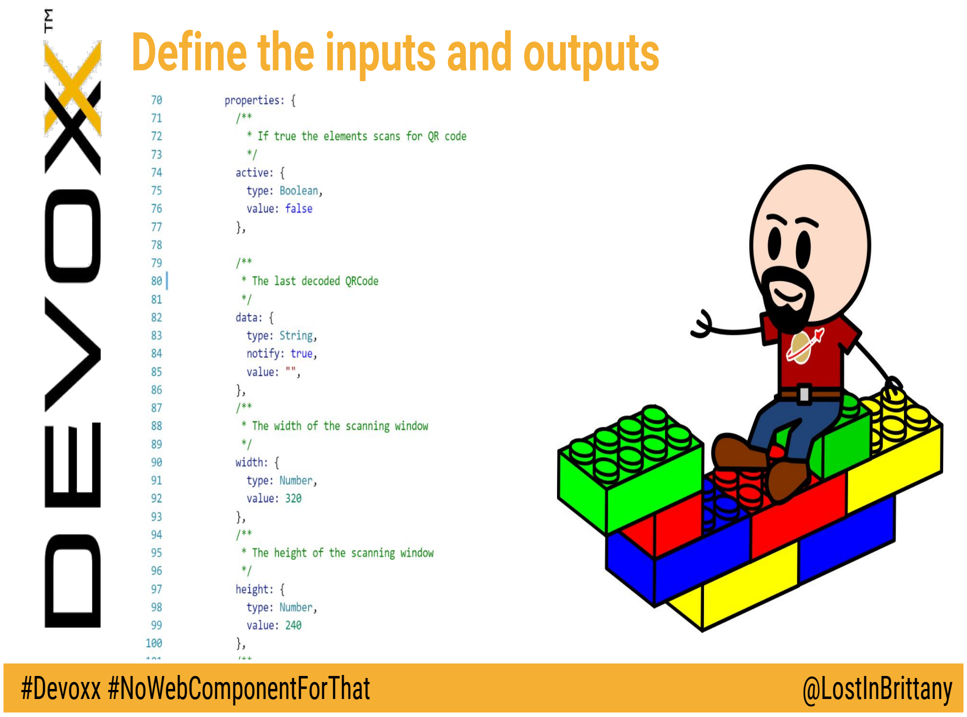


Easier said than done?

1. Define the inputs (attributes)
2. Define the outputs (events)
3. Define the UI (template)
4. Wire the attributes and events to the library
5. Use the lifecycle methods to initialize

#Devoxx #NoWebComponentForThat @LostInBrittany

Following the same five steps that I described earlier...



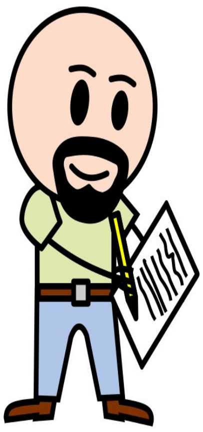
Define the inputs and outputs

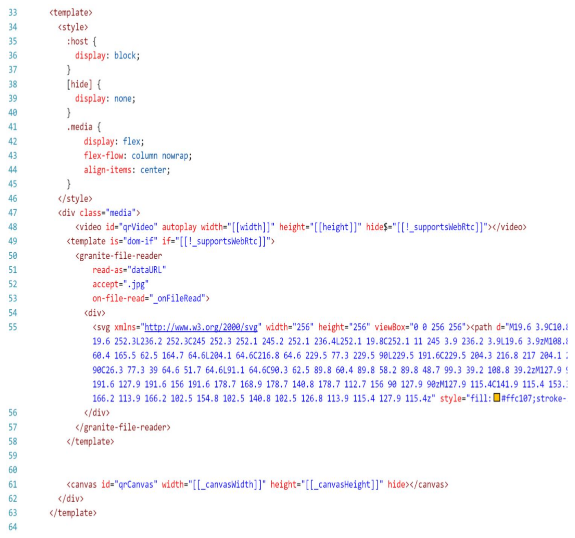
```
properties: {
  /**
    * If true the elements scans for QR code
    */
  active: {
    type: Boolean,
    value: false
  },

  /**
   * The last decoded QRCode
   */
  data: {
    type: String,
    notify: true,
    value: "",
  },
  /**
   * The width of the scanning window
   */
  width: {
    type: Number,
    value: 320
  },
  /**
   * The height of the scanning window
   */
  height: {
    type: Number,
    value: 240
  },
```

#Devoxx #NoWebComponentForThat @LostInBrittany

There are fewer input, described as Polymer properties: the state of the scanner (when it's working it eats a lot of battery, so we want to control when to switch it on, the last decoded data, and the dimensions of the scanning window
As in the qr code generator case, the only output is an event to signal we have decoded a qrcode

Define the UI (template)



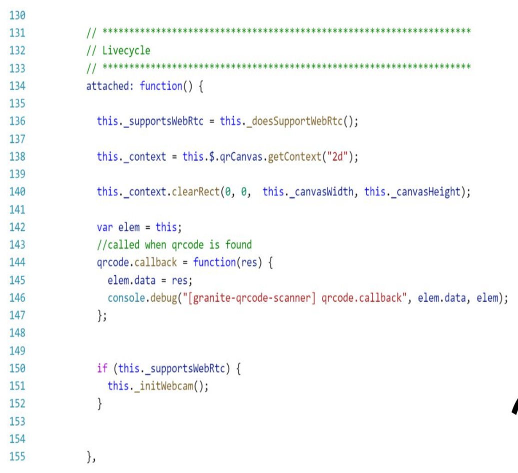


```
33        <template>
34          <style>
35            :host {
36              display: block;
37            }
38            [hide] {
39              display: none;
40            }
41            .media {
42                display: flex;
43                flex-flow: column nowrap;
44                align-items: center;
45            }
46          </style>
47          <div class="media">
48              <video id="qrVideo" autoplay width="[[width]]" height="[[height]]" hide$="[[!_supportsWebRtc]]"></video>
49            <template is="dom-if" if="[[!_supportsWebRtc]]">
50              <granite-file-reader
51                  read-as="dataURL"
52                  accept=".jpg"
53                  on-file-read="_onFileRead">
54                <div>
55                  <svg xmlns="http://www.w3.org/2000/svg" width="256" height="256" viewBox="0 0 256 256"><path d="M19.6 3.9C10.
                    19.6 252.3L236.2 252.3C245 252.3 252.1 245.2 252.1 236.4L252.1 19.8C252.1 11 245 3.9 236.2 3.9L19.6 3.9zM108.
                    60.4 165.5 62.5 164.7 64.6L204.1 64.6C216.8 64.6 229.5 77.3 229.5 90L229.5 191.6C229.5 204.3 216.8 217 204.1
                    90C26.3 77.3 39 64.6 51.7 64.6L91.1 64.6C90.3 62.5 89.8 60.4 89.8 58.2 89.8 48.7 99.3 39.2 108.8 39.2zM127.9
                    191.6 127.9 191.6 156 191.6 178.7 168.9 178.7 140.8 178.7 112.7 156 90 127.9 90zM127.9 115.4C141.9 115.4 153.
                    166.2 113.9 166.2 102.5 154.8 102.5 140.8 102.5 126.8 113.9 115.4 127.9 115.4z" style="fill:🟨#ffc107;stroke-
56                </div>
57              </granite-file-reader>
58            </template>
59
60
61            <canvas id="qrCanvas" width="[[_canvasWidth]]" height="[[_canvasHeight]]" hide></canvas>
62          </div>
63        </template>
64
```

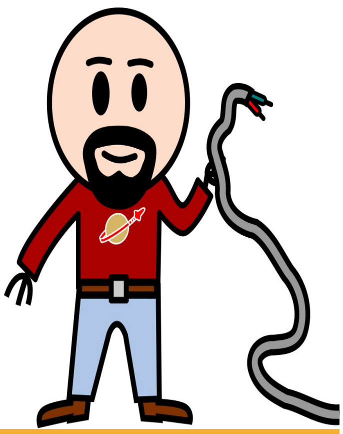
#Devoxx #NoWebComponentForThat @LostInBrittany

The template looks more complex… but that's thanks to Safari, who doesn't support WebRTC and forces us to do some alternative method, that's the lines 49 to 58. Besides that we have a video tag, and a canvas. Why ? Wait a second, I will tell you

Initializing in the lifecycle methods

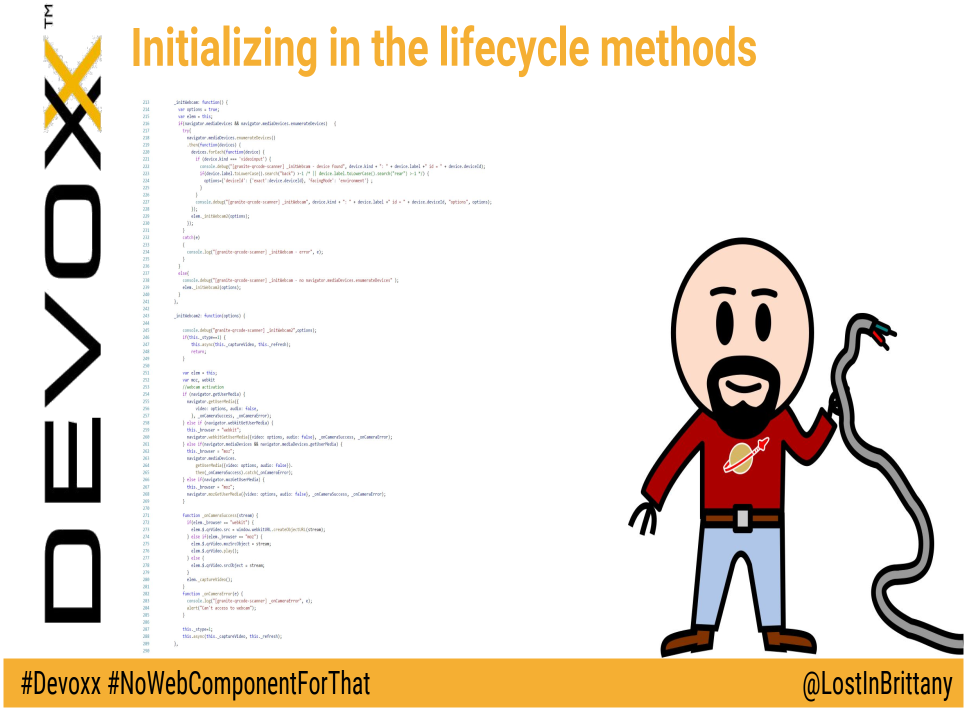


```
// ***********************************************************************
// Livecycle
// ***********************************************************************
attached: function() {

  this._supportsWebRtc = this._doesSupportWebRtc();

  this._context = this.$.qrCanvas.getContext("2d");

  this._context.clearRect(0, 0,  this._canvasWidth, this._canvasHeight);

  var elem = this;
  //called when qrcode is found
  qrcode.callback = function(res) {
    elem.data = res;
    console.debug("[granite-qrcode-scanner] qrcode.callback", elem.data, elem);
  };


  if (this._supportsWebRtc) {
    this._initWebcam();
  }


},
```



#Devoxx #NoWebComponentForThat @LostInBrittany

In the lifecycling methods, here in attached, we do our setup: we test if the browser supports WebRTC, we initialize the canvas that will receive the image to decode, and then we do all the dirty code to initialize the webcam...



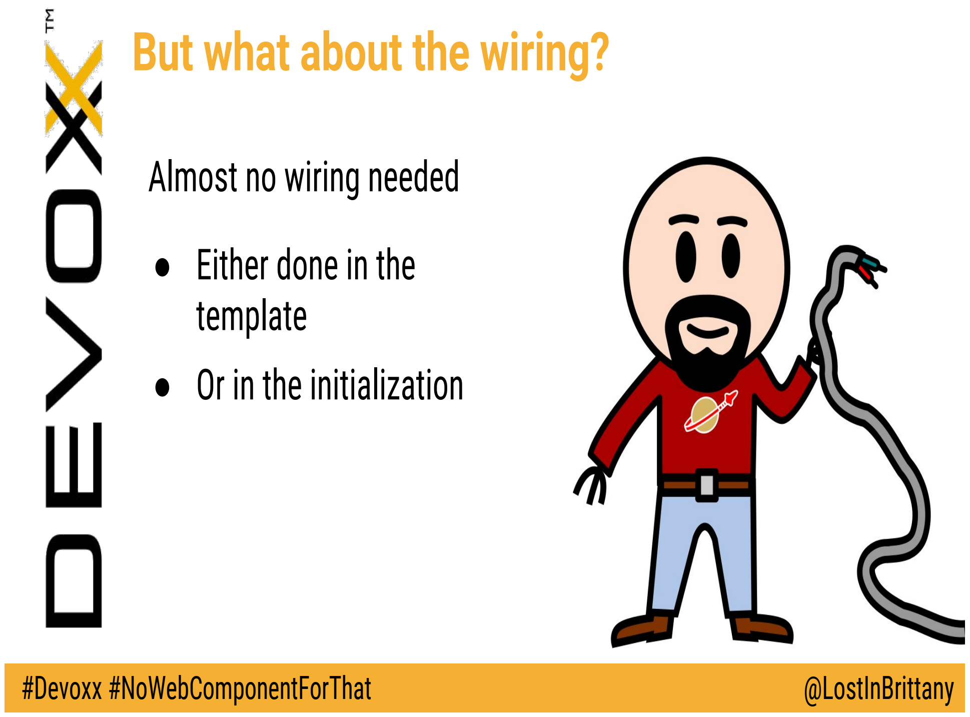
Initializing in the lifecycle methods

```
    _initWebcam: function() {
      var options = true;
      var elem = this;
      if(navigator.mediaDevices && navigator.mediaDevices.enumerateDevices)   {
        try{
          navigator.mediaDevices.enumerateDevices()
          .then(function(devices) {
            devices.forEach(function(device) {
              if (device.kind === 'videoinput') {
                console.debug("[granite-qrcode-scanner] _initWebcam - device found", device.kind + ": " + device.label +" id = " + device.deviceId);
                if(device.label.toLowerCase().search("back") >-1 /* || device.label.toLowerCase().search("rear") >-1 */) {
                  options={'deviceId': {'exact':device.deviceId}, 'facingMode': 'environment'} ;
                }
              }
              console.debug("[granite-qrcode-scanner] _initWebcam", device.kind + ": " + device.label +" id = " + device.deviceId, "options", options);
            });
            elem._initWebcam2(options);
          });
        }
        catch(e)
        {
          console.log("[granite-qrcode-scanner] _initWebcam - error", e);
        }
      }
      else{
        console.debug("[granite-qrcode-scanner] _initWebcam - no navigator.mediaDevices.enumerateDevices" );
        elem._initWebcam2(options);
      }
    },

    _initWebcam2: function(options) {

        console.debug("granite-qrcode-scanner] _initWebcam2",options);
        if(this._stype==1) {
            this.async(this._captureVideo, this._refresh);
            return;
        }

        var elem = this;
        var moz, webkit
        //webcam activation
        if (navigator.getUserMedia) {
          navigator.getUserMedia({
              video: options, audio: false,
            }, _onCameraSuccess, _onCameraError);
        } else if (navigator.webkitGetUserMedia) {
          this._browser = "webkit";
          navigator.webkitGetUserMedia({video: options, audio: false}, _onCameraSuccess, _onCameraError);
        } else if(navigator.mediaDevices && navigator.mediaDevices.getUserMedia) {
          this._browser = "moz";
          navigator.mediaDevices.
              getUserMedia({video: options, audio: false}).
              then(_onCameraSuccess).catch(_onCameraError);
        } else if(navigator.mozGetUserMedia) {
          this._browser = "moz";
          navigator.mozGetUserMedia({video: options, audio: false}, _onCameraSuccess, _onCameraError);
        }

        function _onCameraSuccess(stream) {
          if(elem._browser == "webkit") {
            elem.$.qrVideo.src = window.webkitURL.createObjectURL(stream);
          } else if(elem._browser == "moz") {
            elem.$.qrVideo.mozSrcObject = stream;
            elem.$.qrVideo.play();
          } else {
            elem.$.qrVideo.srcObject = stream;
          }
          elem._captureVideo();
        }
        function _onCameraError(e) {
          console.log("[granite-qrcode-scanner] _onCameraError", e);
          alert("Can't access to webcam");
        }

        this._stype=1;
        this.async(this._captureVideo, this._refresh);
    },
```

#Devoxx #NoWebComponentForThat @LostInBrittany

Did I say the dirty code ? I wanted to be sure the element works in as many browsers as possible, so the code was not pretty…

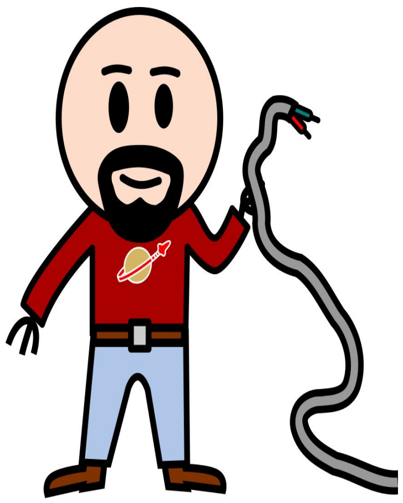
Please, don't try to read it here… and if you do it later and see a simpler way to do it, please make a pull request!



But what about the wiring?

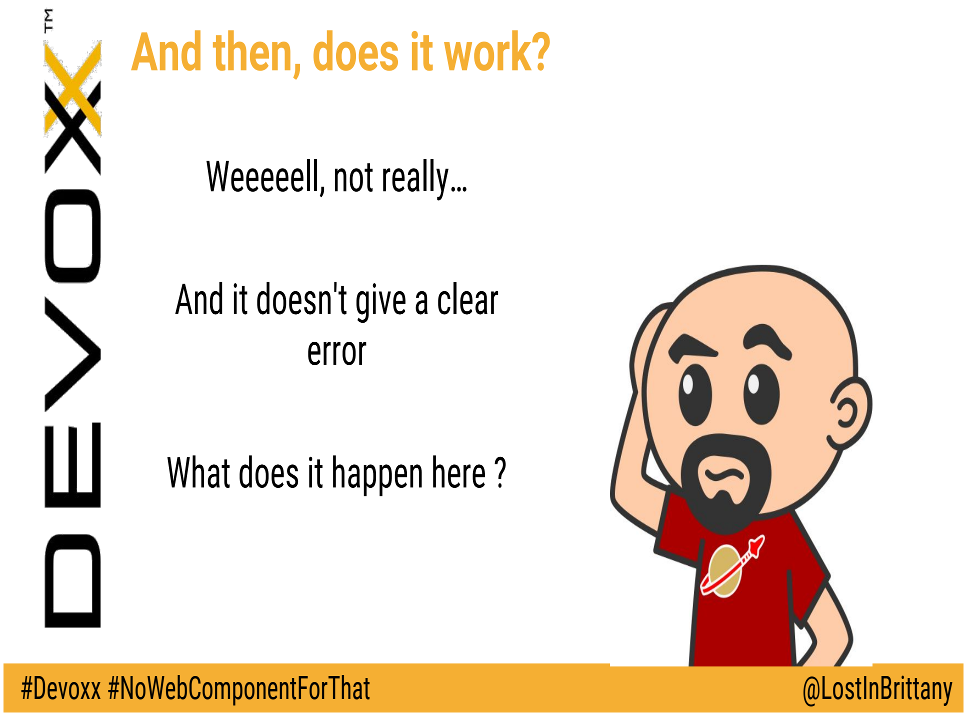
Almost no wiring needed

- Either done in the template
- Or in the initialization



#Devoxx #NoWebComponentForThat @LostInBrittany

Hey, I did forgot the wiring, didn't I ? Not really, but in this case the wiring is rather minimalistic, and it's done either in the template or in the initialization, in the lifecycle methods.



And then, does it work?

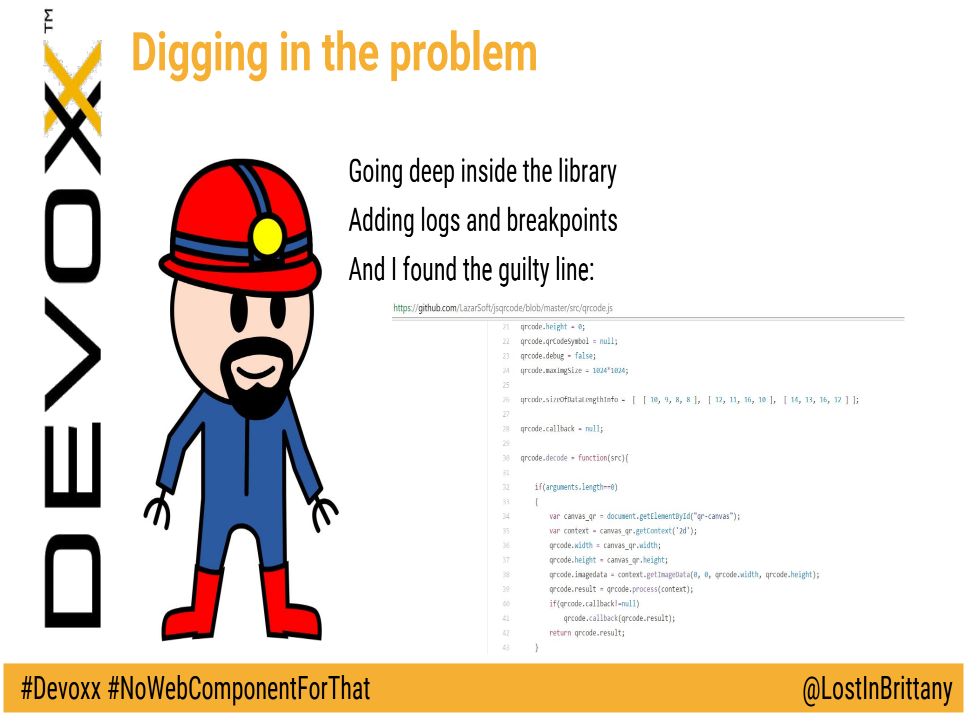
Weeeeell, not really…

And it doesn't give a clear error

What does it happen here ?

#Devoxx #NoWebComponentForThat @LostInBrittany

So then I test it at last, full of hope… and it doesn't work!
But why ?!?! There was no clear error in the console, it failed somewhere inside jsqrcode…
What to do?



Digging in the problem

Going deep inside the library

Adding logs and breakpoints

And I found the guilty line:

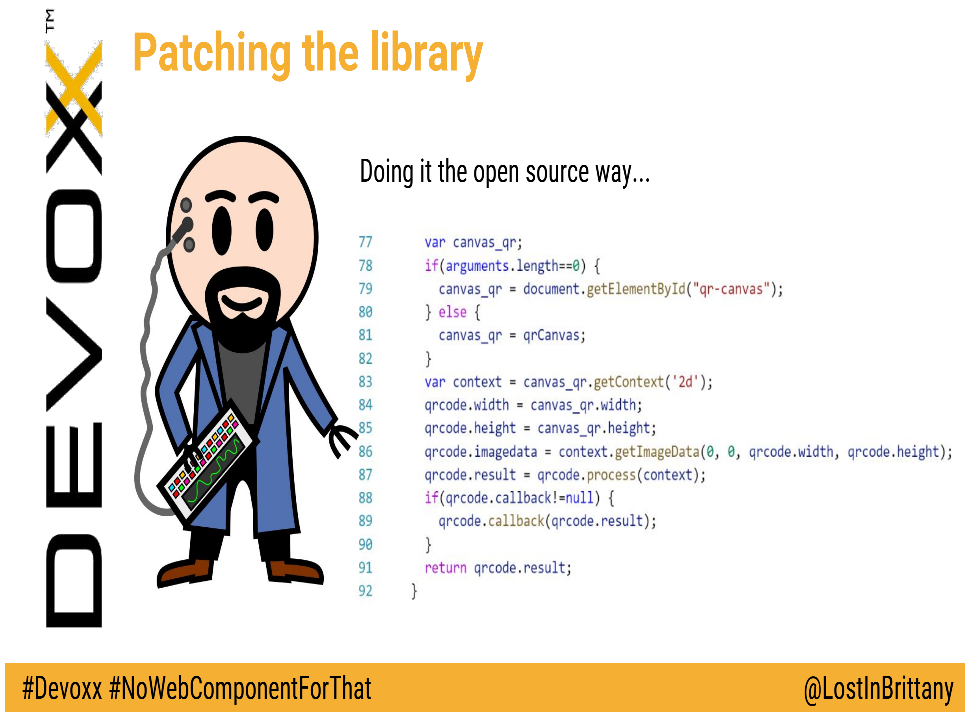
https://github.com/LazarSoft/jsqrcode/blob/master/src/qrcode.js

```
21  qrcode.height = 0;
22  qrcode.qrCodeSymbol = null;
23  qrcode.debug = false;
24  qrcode.maxImgSize = 1024*1024;
25
26  qrcode.sizeOfDataLengthInfo =  [  [ 10, 9, 8, 8 ],  [ 12, 11, 16, 10 ],  [ 14, 13, 16, 12 ] ];
27
28  qrcode.callback = null;
29
30  qrcode.decode = function(src){
31
32      if(arguments.length==0)
33      {
34          var canvas_qr = document.getElementById("qr-canvas");
35          var context = canvas_qr.getContext('2d');
36          qrcode.width = canvas_qr.width;
37          qrcode.height = canvas_qr.height;
38          qrcode.imagedata = context.getImageData(0, 0, qrcode.width, qrcode.height);
39          qrcode.result = qrcode.process(context);
40          if(qrcode.callback!=null)
41              qrcode.callback(qrcode.result);
42          return qrcode.result;
43      }
```

#Devoxx #NoWebComponentForThat @LostInBrittany

So I dag in the library, trying to understand how does it work. I put logs, set breakpoints… And I found it, one guilty line.
Do you spot it? Yes, it's line 34, a getElementById with a set id, that couldn't work on a component world, inside Shadow DOM…

What to do?



Patching the library

Doing it the open source way...

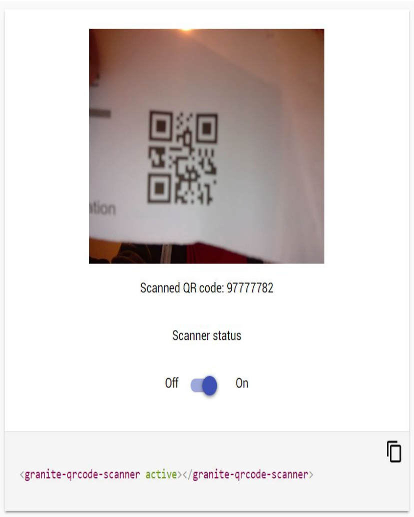
```
var canvas_qr;
if(arguments.length==0) {
  canvas_qr = document.getElementById("qr-canvas");
} else {
  canvas_qr = qrCanvas;
}
var context = canvas_qr.getContext('2d');
qrcode.width = canvas_qr.width;
qrcode.height = canvas_qr.height;
qrcode.imagedata = context.getImageData(0, 0, qrcode.width, qrcode.height);
qrcode.result = qrcode.process(context);
if(qrcode.callback!=null) {
  qrcode.callback(qrcode.result);
}
return qrcode.result;
}
```

#Devoxx #NoWebComponentForThat @LostInBrittany

I did the only sane thing for an open source developer, I forked the project, patched the library in order to keep exactly the same behavior (I didn't want to break anyones's code) but with a new options that allowed to explicitly pass a HTML canvas object.

So now my dependency wasn't on Lazersoft jsqrcode but on my fork. I'm putting a pull request on their repository as soon as I will find time to correctly explain what I do and why, of course.

granite-qrcode-scanner



Scanned QR code: 97777782

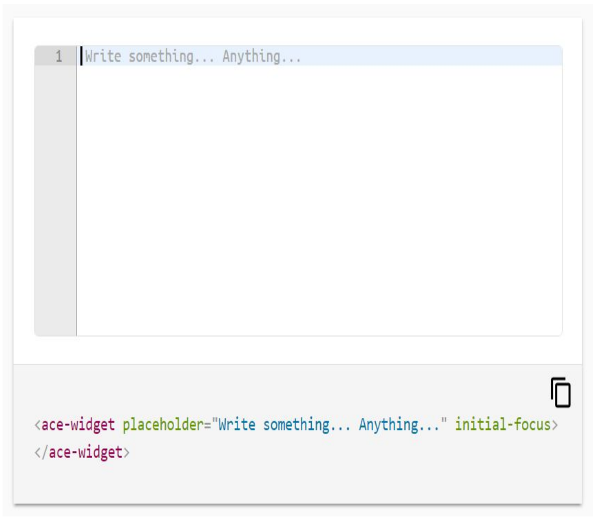
```
<granite-qrcode-scanner active></granite-qrcode-scanner>
```

#Devoxx #NoWebComponentForThat @LostInBrittany

And it worked, the element detected and scanned the QR Code, quickly and efficiently.

If I had more time, I'd tell you the hack I had to do to make it work on iOS, but as they say, that's another story that will be tell another time...

Other examples: ace-widget



```
Write something... Anything...
```

```
<ace-widget placeholder="Write something... Anything..." initial-focus>
</ace-widget>
```

#Devoxx #NoWebComponentForThat @LostInBrittany

Ace editor is a well known JavaScript library to make rich editors. When I had to do an IDE for Warp 10, I wanted to use all the power of ace, but keeping the component approach, so I did a Polymer version.

It worked like a charm… until I tested it in full ShadowDOM mode. While it worked in Shaddy DOM, in ShadowDOM all the styles were messed up. It was there that I discovered the effets of dirty DOM manipulation, Ace creates some styles on the fly using the ids in your document, and put them in the header of the HTML page. Of course, it doesn't pass well the Shadow DOM boundary…

I had to create a Polymer behavior and some custom JS code in order to intercept that styles and creating them in the template of the component instead. And then, it worked!

Thanks!

I hope you liked this talk!

Don't hesitate to send me your questions
by email, twitter, hangout, carrier pigeon...