COMPONENTS

More than a shinny frenzy!

@expensivestevie

Hello! Terve!

I'm Kenigbolo Meya Stephen

Front End Engineering Lead @bcaster

Arch Conveyer/Community Manager @TheCodeAfrique

You can find me on twitter at @expensivestevie

BCASTER

@expensivestevie

WHAT WE'LL BE LOOKING AT

- What exactly are web components?
- Early days of web components
- HTML components
- XBL
- Similarities, Differences, Failures
- Modern web components

@expensivestevie

1.
What exactly are web components

The beauty lies in the eyes of the beholder

@expensivestevie

According to WebComponents.Org

Web components are a set of web platform APIs that allow you to create new custom, reusable, encapsulated HTML tags to use in web pages and web apps.

2.
Early days of web components

Yeah, this idea is not so new 😂

@expensivestevie

> The initial attempts to extend the default DOM API came from the big browser companies (not facebook 😉). Their desire (and failure) to provide developers features to enhance web possibilities has led to what we today call web components

3.
HTML Components

Despite all the stick they get, MIcrosoft started us on the path to web components 👏👏👏

@expensivestevie

HTML COMPONENTS

- 1998 Microsoft proposed HTML Components (**IE 5.5 support**).
- The Goal - Change behaviour of default tags.
- Could be written in JScript or VBScript.
- MicroSoft ActiveX Scripting Interfaces was required

```
<PUBLIC:COMPONENT>
<PUBLIC:ATTACH EVENT="onmouseover" ONEVENT="Hilite()" />
<PUBLIC:ATTACH EVENT="onmouseout"  ONEVENT="Restore()" />
<SCRIPT LANGUAGE="JScript">
   var normalColor, normalSpacing;

   function Hilite()
   {
     // save original values
     normalColor  = runtimeStyle.color;
     normalSpacing= runtimeStyle.letterSpacing;

     runtimeStyle.color  = "red";
     runtimeStyle.letterSpacing = 2;
   }

   function Restore()
   {
     // restore original values
     runtimeStyle.color  = normalColor;
     runtimeStyle.letterSpacing = normalSpacing;
   }
</SCRIPT>
</PUBLIC:COMPONENT>
```

@expensivestevie

```
1 <HEAD>
2 <STYLE>
3    LI {behavior:url(hilite.htc)}
4 </STYLE>
5 </HEAD>
6
7 <P>Mouse over the two list items below to see this effect.
8 <UL>
9   <LI>HTML Authoring</LI>
10  <LI>Dynamic HTML</LI>
11 </UL>
```

@expensivestevie

4.
XBL (XML Binding Language)

Don’t we all just love Mozilla 💖💖💖💖

@expensivestevie

XML Binding Language (XBL)

- In **2001, XBL** was created with **XBL 2** following in **2007**.
- The **Goal** - Extend default DOM tags with custom behaviour.
- Was an addition to Mozilla's **XUL**
- Full support on all Mozilla products

MOZILLA XBL

Concept

Extending the DOM with custom tags. This would have improved readability as well as sustainable reusability. In general the concept was good

Results

Despite the concept being good, the implementations weren't as good as the idea.

```
scrollbar {
    -moz-binding: url('chrome://findfile/content/findfile.xml#binding1');
}
```

```
<?xml version="1.0"?>
<bindings xmlns="http://www.mozilla.org/xbl">
  <binding id="binding1">
    <!-- content, property, method and event descriptions go here -->
  </binding>
  <binding id="binding2">
    <!-- content, property, method and event descriptions go here -->
  </binding>
</bindings>
```

@expensivestevie

5.
Similarities, Differences, Failures!

The devil always lies in the details! Maybe not!

@expensivestevie

Similarities

- Extending the Default DOM tags behaviour
- Improve reusability
- Better code composition
- Improving developer experience

Differences

- Microsoft used HTML (htc) while Mozilla used XML(xbl)
- Microsoft used a paradigm of one component per file,

  Mozilla however went with multiple bindings.
- Microsoft required ActiveX(IE) whilst Mozilla required it to

  have GRE (All mozilla products by default)

Failures!

There are no **Failures** - Just **experiences** and your **reactions** to them

@expensivestevie

- **2001 - XBL** dropped in favour of **XBL 2**. **W3C** released candidate recommendation of XBL 2
- **2011 - Microsoft** discontinues **HTC**, subsequently deprecates it in **IE 10** release
- **2012 - Mozilla** XBL 2 abandoned. No browser implemented the specs anywhere

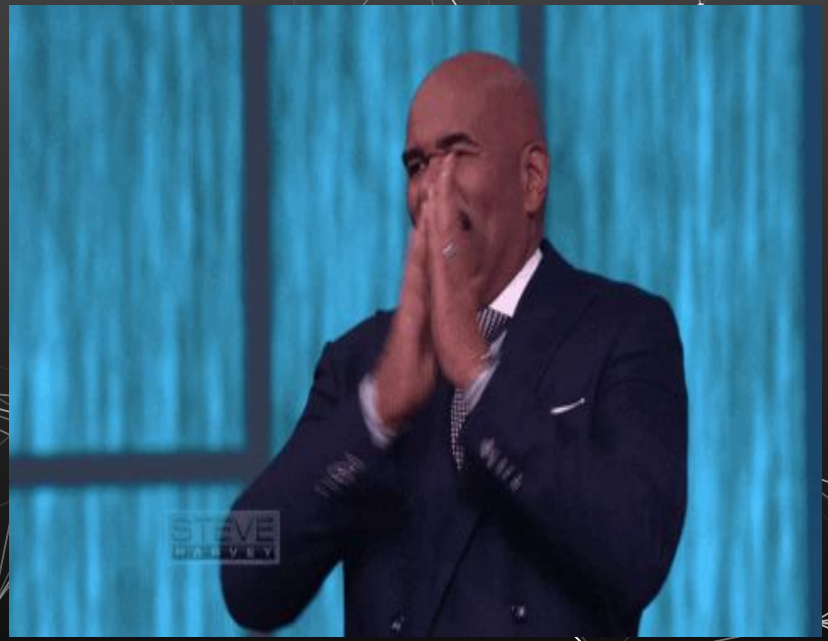
6. Modern Web Components

Huge thanks to Google's work on Angularjs

@expensivestevie

AngularJS

In order to allow developers create their own tags as well as help with a better semantic for web apps, Google brought us AngularJS



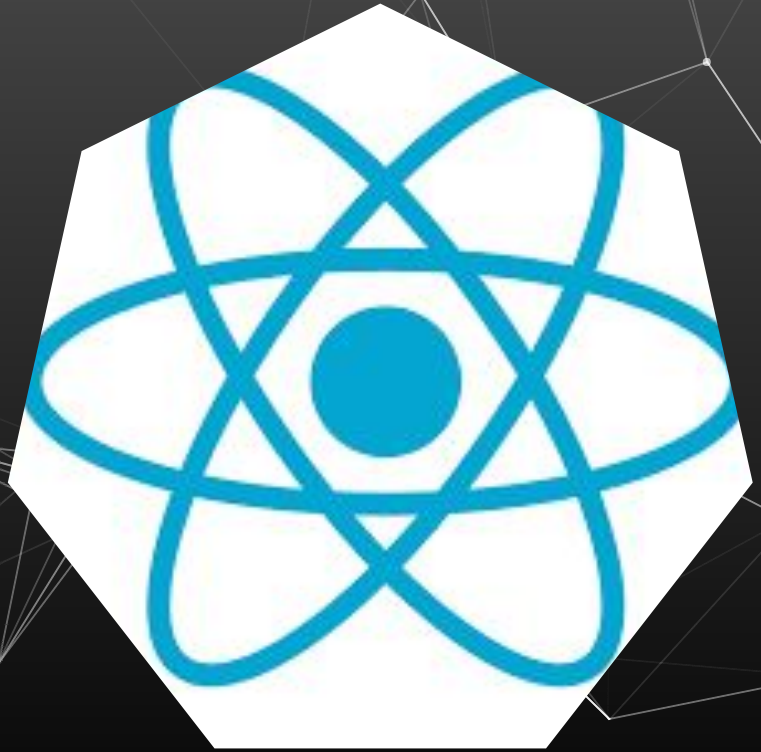
AngularJS Timeline

- Born in 2009.
- Released on Github in 2010.
- Stable version 1.0 rolled out in 2013

As much as we wouldn't like to say it, the truth remains that the implementation of "components" in most of our favorite frontend frameworks originated from this

React - Facebook

In order to allow reusability of widgets, Facebook released its own library called **React.** React was born in **2011** and was released on Github in **2013**. While not the most popular in term of Github stars, it's the most popular in terms of keyword search and actual usage/adoption



> React allows developers create reusable web components as well as handle data change without reloading the page. It currently is used more than any framework out there and has a huge community.

Vue - OSS Community

Vue is a progressive framework for building user interfaces. Currently the most starred on github (in comparison with react and angular). Vue JS comes across as a pick of all the good parts in both **Angular** and **React.** It is community maintained and spearheaded by its creator **Evan You**

> As the core team, we obviously like Vue a lot. There are some problems we think it solves better than anything else out there. If we didn't believe that, we wouldn't be working on it.

https://vuejs.org/v2/guide/comparison.html

Web Component API Specs - Main concepts

- Custom Elements
- Shadow DOM
- ES Modules
- HTML Template


Custom Elements

HTML ▼

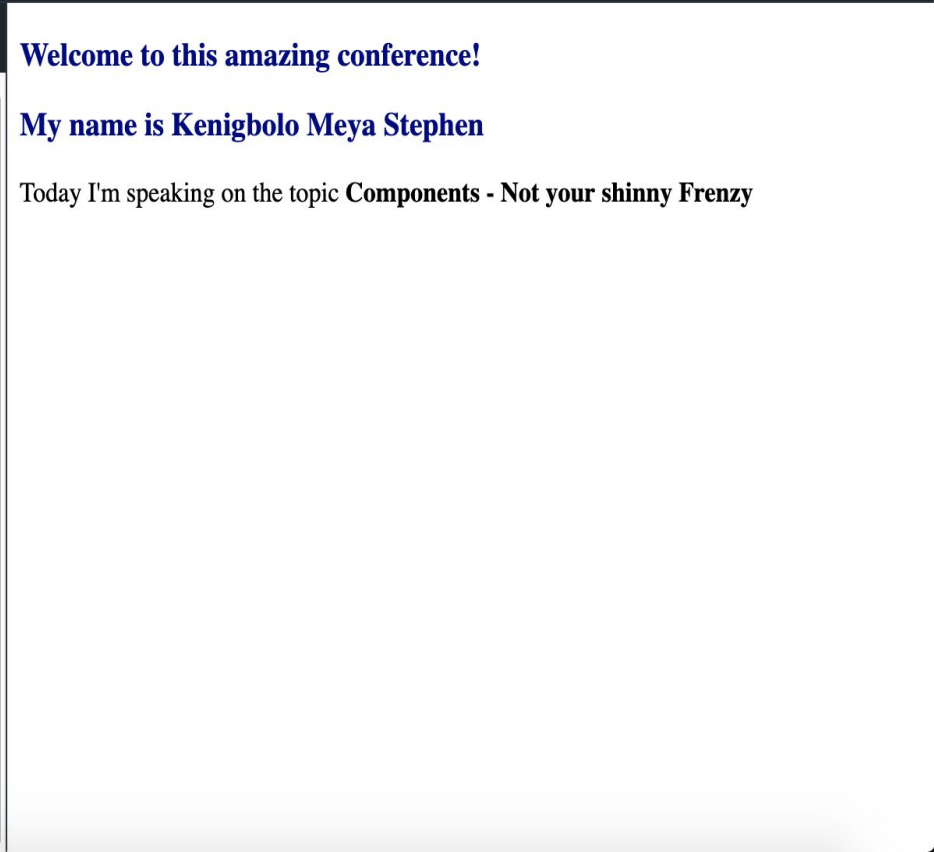
```html
<h3>Welcome to this amazing conference!</h3>

<shinny-frenzy full-name="Kenigbolo Meya Stephen" title="Components - Not your shinny Frenzy" conference="FSDC Finland" job-position="Developer">
</shinny-frenzy>
```

CSS ▼

```css
h3 {
  color: red;
}
```

JavaScript + No-Library (pure JS) ▼

```javascript

class ShinnyFrenzyElement extends HTMLElement {
  // We have to declare a custom constructor
  constructor() {
    // First, call super() to be sure everything was made alright
    super()

    // Then we can take care of our attributes
    const title = this.getAttribute('title')
    const fullName = this.getAttribute('full-name')

    // Finaly, we can render the real DOM behind our custom tag
    this.innerHTML = `
      <style>
        h3 {
          color: navy;
        }
      </style>
      <h3>My name is ${fullName}</h3>
      <span>Today I'm speaking on the topic <b>${title}</b></span>
    `
  }
}

// We have to register our custom tag to window.customElements which lists all the custom tags we have.
window.customElements.define('shinny-frenzy', ShinnyFrenzyElement)

```



Custom Elements (V1) - LS

Method of defining new HTML tags.

Usage: Global

% of all users ?

71.73% + 15.13% = 86.86%

Current aligned | Usage relative | Date relative | Apply filters | Show all | ?

IE	Edge *	Firefox	Chrome	Safari	iOS Safari *	Opera Mini *	Chrome for Android	UC Browser for Android	Samsung Internet
			49						
			72		11.4 [1]				4
	17		73	12 [1]	12.1 [1]				8.2
11	18	66	74	12.1 [1]	12.2 [1]	all	74	11.8	9.2
	75	67	75	TP [1]					
		68	76						
			77						

@expensivestevie


2016
NEW
Shadow DOM

HTML ▼

```html
<h3>Welcome to this amazing conference!</h3>

<shinny-frenzy full-name="Kenigbolo Meya Stephen" title="Components - Not your shinny Frenzy"
conference="FSDC Finland" job-position="Developer">
</shinny-frenzy>
```

CSS ▼

```css
h3 {
  color: red;
}
```

JavaScript + No-Library (pure JS) ▼

```javascript
class ShinnyFrenzyElement extends HTMLElement {
  // We have to declare a custom constructor
  constructor() {
    // First, call super() to be sure everything was made alright
    super()

    // Then we can take care of our attributes
    const title = this.getAttribute('title')
    const fullName = this.getAttribute('full-name')
    // Let's declare a Shadow DOM within our tag and our rendering will be done IN the Shadow DOM
    const shadow = this.attachShadow({mode: 'open'});

    // Finaly, we can render the real DOM behind our custom tag
    shadow.innerHTML = `
      <style>
        h3 {
          color: navy;
        }
      </style>
      <h3>My name is ${fullName}</h3>
      <span>Today I'm speaking on the topic <b>${title}</b></span>
    `
  }
}

// We have to register our custom tag to window.customElements which lists all the custom tags we
have.
window.customElements.define('shinny-frenzy', ShinnyFrenzyElement)
```

Shadow DOM (V1) - WD

Method of establishing and maintaining functional boundaries between DOM trees and how these trees interact with each other within a document, thus enabling better functional encapsulation within the DOM & CSS.

Usage
Global

% of all users ?
74.48% + 12.68% = 87.16%

Current aligned | Usage relative | Date relative | Apply filters | Show all ?

IE	Edge *	Firefox	Chrome	Safari	iOS Safari *	Opera Mini *	Chrome for Android	UC Browser for Android	Samsung Internet
			49		11.4 [1]				4
	17		72	12 [1]	12.1 [1]				8.2
11	18	66	73	12.1 [1]	12.2 [1]	all	74	11.8	9.2
	75	67	74	TP [1]					
		68	75						
			76						
			77						

ES Modules

```
1 <link rel="import" href="shinny-frenzy-component.html"/>
```

HTML Imports - WD

Usage % of all users ?

Global 13.77%

Method of including and reusing HTML documents in other HTML documents.

Current aligned | Usage relative | Date relative | Apply filters | Show all | ?

IE	Edge *	Firefox	Chrome	Safari	iOS Safari *	Opera Mini *	Chrome for Android	UC Browser for Android	Samsung Internet
			49						
			72		11.4				4
	17		73	12	12.1				8.2
11	18	66	74	12.1	12.2	all	74	11.8	9.2
	75	67	75	TP					
		68	76						
			77						


HTML Template

```
<template id="not-your-shinny-frenzy">
  <h2></h2>
  <h3></h3>
</template>
```

t + No-Library (pure JS) ▼

```
class ShinnyFrenzyElement extends HTMLElement {
  // We have to declare a custom constructor
  constructor() {
    // First, call super() to be sure everything was made alright
    super()

    // Then we can take care of our attributes
    const title = this.getAttribute('title')
    const fullName = this.getAttribute('full-name')
    // Let's declare a Shadow DOM within our tag and our rendering
will be done IN the Shadow DOM
    const shadow = this.attachShadow({mode: 'open'});
    const template = document.querySelector('#not-your-shinny-
frenzy')
    const content = document.importNode(template.content, true);
    content.querySelector('h2').textContent = title;
    content.querySelector('h3').textContent = fullName;
    shadow.appendChild(content)
  }
}

// We have to register our custom tag to window.customElements which
lists all the custom tags we have.
window.customElements.define('shinny-frenzy', ShinnyFrenzyElement)
```

CSS ▼

```
1  h2 {
2    color: red;
3  }
4
5  h3 {
6    color: blue;
7  }
```

```
<head>…</head>
<body>
  <template id="not-your-shin
  frenzy"> == $0
    #document-fragment
    </template>
  <script>…</script>
  </body>
</html>
```

... div iframe html body template#not-your-shinny

Styles Computed Event Listeners DOM Breakpoin

Filter :hov .c

```
element.style {
}
```

```
template {                                  user agent sty
  display: none;
}
```

Inherited from html

```
html {                                      user agent sty
  color: -internal-root-color;
}
```

margin -

Console Remote devices What's New ×

Highlights from the Chrome 74 update

Highlight all nodes affected by CSS property

Hover over a CSS property like padding or margin in the Styles pane to highlight all nodes affected by that declaration.

Lighthouse v4 in the Audits panel

Featuring a new "tap targets" audit for checking that mobile links and buttons are properly sized,

HTML templates - LS

Method of declaring a portion of reusable markup that is parsed but not rendered until cloned.

Usage: Global — % of all users: 93.03% + 0.41% = 93.44%

Current aligned | Usage relative | Date relative | Apply filters | Show all | ?

IE	Edge *	Firefox	Chrome	Safari	iOS Safari *	Opera Mini *	Chrome for Android	UC Browser for Android	Samsung Internet
			49						
			72		11.4				4
	17		73	12	12.1				8.2
11	18	66	74	12.1	12.2	all	74	11.8	9.2
	75	67	75	TP					
		68	76						
			77						

Notes | Known issues (0) | Resources (8) | Feedback

That’s all for today folks

Thank You! KeyToss!

Any questions?

You can find me at

- Twitter **@expensivestevie**
- Github **@kenigbolo**

@expensivestevie