



# FRONTEND PRO

Introducción a herramientas esenciales de Frontend



# Herramientas esenciales

- Javascript
  - Estándares (ECMA Script 5, ES6...)
  - Librerías (jQuery, lodash...)
  - Extensiones del lenguaje
    - Typescript
    - Coffeescript
    - JSX
- Estilos
  - SASS
  - LESS
- Toolkits
  - Bootstrap
- Frameworks
  - React
  - Angular
  - VueJS



# Herramientas esenciales

- **Javascript**
  - **Estándares (ECMA Script 5, ES6...)**
  - **Librerías (jQuery, lodash...)**
  - **Extensiones del lenguaje**
    - **TypeScript**
    - Coffeescript
    - JSX
- **Estilos**
  - **SASS**
  - LESS
- Toolkits
  - Bootstrap
- Frameworks
  - React
  - Angular
  - VueJS



# Herramientas relacionadas con Javascript



# Herramientas esenciales - Javascript

## Estándares (ECMA Script 5, ES6...)

- Arrow function

```
// ES5  
function () {}  
// ES6  
() => {}
```

- “key:value” as “value” in Objects

```
let v = "hey";  
// ES5  
let obj = {v: v}; // obj = {v: "hey"}  
// ES6  
let obj = {v}; // obj = {v: "hey"}
```

- Import & require

```
//ES5  
var _ = require("lodash");  
//ES6  
import from "lodash";
```



# Herramientas esenciales - Javascript

## Estándares (ECMA Script 5, ES6...)

- Template string (backslash `)

```
let a = "hey boy!";
let b = "hey girl!";
// ES5
console.log(a + ", " + b); // "hey boy!, hey girl!"
// ES6
console.log(` ${a} ${b}`);
```

Para más info:

<https://medium.com/@jagogutierrez/ecmascript-es6-diferencias-notables-al-es5-83d3e33ae201>

<https://en.wikipedia.org/wiki/ECMAScript>



# Herramientas esenciales - Javascript

## Librerías - jQuery & Lodash

Son librerías que nos permiten realizar tareas de forma mucho más sencilla

```
// sin jQuery
document.getElementById('post-392').classList.add('background-red');
// con jQuery
$('#post-392').addClass('background-red');

// sin lodash
let arr = [1, 2, 3, 4, 5, 6, 7, 8];
let r = [];
r[0] = arr.map(e => e % 2 === 0);
r[1] = arr.map(e => e % 2 === 1);
// con lodash
_partition(a, n => n % 2);
```

Para más info: <https://api.jquery.com/>, <https://lodash.com/>



# Herramientas esenciales - Javascript

## Extensiones del lenguaje - Typescript

Tiene extensiones de archivo propias (\*.ts, \*.d.ts, \*.map)

Es javascript pero con control de tipos

```
interface User {  
    name: string;  
    id: number;  
}  
  
const user: User = {  
    username: "Hayes",  
    Type '{ username: string; id: number; }' is not assignable to type 'User'.  
        Object literal may only specify known properties, and 'username' does not exist in  
        type 'User'.  
    id: 0,  
};
```

Veamos un poco más:

<https://www.typescriptlang.org/docs/handbook/typescript-in-5-minutes.html>



# Herramientas esenciales - Javascript

## Extensiones del lenguaje - Coffeescript

La extensión es .coffee

Es una transformación de Javascript inspirada en Python. Busca sencillez en la escritura.

Para más info:

<https://coffeescript.org/#overview>



# Herramientas esenciales - Javascript

## Extensiones del lenguaje - JSX

La extensión es .jsx

Es una extensión creada por Facebook que permite escribir HTML dentro de contenido de javascript, sin que sea necesariamente HTML.

Se usa en REACT

Para más info:

<https://es.reactjs.org/docs/introducing-jsx.html>



# Herramientas esenciales - Javascript RESUMEN

Hemos visto que podemos:

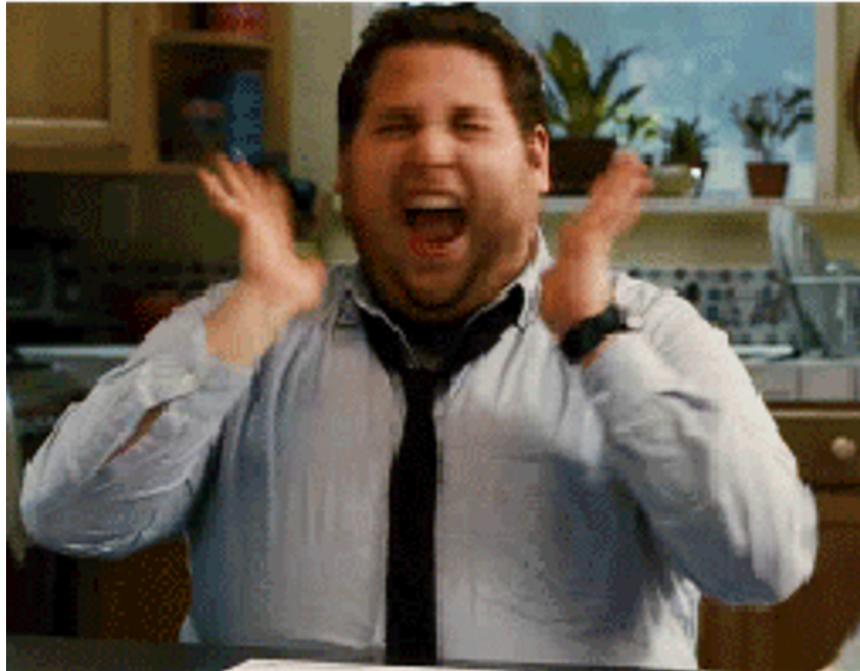
- usar semántica más sencilla
- enriquecer Javascript con librerías externas
- usar extensiones como Typescript o JSX

En resumen, **utilizar herramientas que nos simplifiquen la vida**





# MARAVILLOSO!

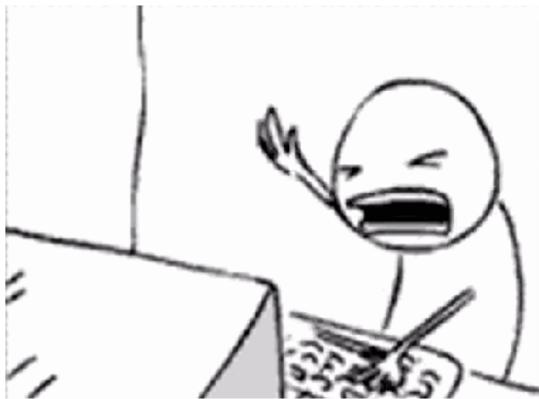


# Herramientas esenciales - Javascript RESUMEN

Sí o sí, un navegador necesita

- HTML o XML (obligatoriamente)
- CSS (vamos a darle algo de color y forma a nuestro código)
- Javascript (vamos a darle algo de vida a nuestro código)

Pero los navegadores solamente entienden javascript, es decir,  
archivos \*.js!!!!





# Ok, pues necesitamos ayuda



# Herramientas esenciales - Javascript RESUMEN

Para poder transformar los archivos que no son \*.js, necesitaremos **compiladores**

Los compiladores suelen ser scripts que podemos lanzar del tipo

```
<comando del compilador> <input> <output>
```

Ejemplos:

```
tsc src/*.ts  
coffee --compile --output lib/ src/
```



# Estilos



# Herramientas esenciales - Estilos

Tal y como acabamos de ver, en algunos casos necesitamos **compiladores** para poder **transformar de un lenguaje a otro**.

El objetivo en este caso será obtener archivos .css ya que el navegador es el tipo de archivo que reconoce. Podemos hacerlo en el backend y servirlo como css o bien que sea el propio navegador (como por ejemplo con [LESS](#))

En el caso de los estilos vamos a ver:

- SASS
- LESS



# Herramientas esenciales - Estilos - SASS

sass podemos encontrarlo en dos formas de archivo:

- \*.scss
- \*.sass

La diferencia está en la sintaxis. Con scss usamos {} para abrir y cerrar secciones y ; para terminar las asignaciones a propiedades

```
// _base.scss
$font-stack: Helvetica, sans-serif;
$primary-color: #333;

body {
  font: 100% $font-stack;
  color: $primary-color;
}
```

```
// _base.sass
$font-stack: Helvetica, sans-serif
$primary-color: #333

body
  font: 100% $font-stack
  color: $primary-color
```



# Herramientas esenciales - Estilos - LESS

less lo encontramos en una forma de archivo:

- \*.less

Los principios son los mismos que SASS, pero cambia la sintaxis. Un archivo LESS y uno SASS, aunque produzcan lo mismo, son incompatibles entre sí (no podemos compilar con sass un \*.less y viceversa)



# Toolkits



# Herramientas esenciales - Toolkits

Para que nos entendamos, los toolkits son paquetes de archivos HTML, CSS y Javascript que nos permiten construir una página o un site con un **menor esfuerzo** y siguiendo una **guía de estilos**

Nacen de la necesidad de cubrir los desarrollos mobile.

Ejemplos más notables:

- Bootstrap (Twitter)
- Bulma
- Foundation (Zurb)
- TailwindCSS



# Herramientas esenciales - Toolkits

## Disclaimer

No hay unidad en el mundo de desarrollo en catalogar Bootstrap (y otros toolkits) como **toolkit**, sino que les llaman **frameworks**

Más adelante veremos qué es un framework y por qué no considero Bootstrap, Foundation, etc frameworks.



# Frameworks



# Herramientas esenciales - Frameworks

De la traducción literal, es un “marco de trabajo”.

El objetivo es el mismo que el de los toolkits: construir una página o un site con un **menor esfuerzo** y siguiendo una **guía de estilos**

La diferencia entre los toolkits y los frameworks, según mi criterio (insisto, soy uno más dentro de millones de desarrolladores), es que **los toolkits son** como las librerías de javascript: **plug&play**. Por lo tanto, **los toolkits no necesitan compilación para que funcionen.**



# Herramientas esenciales - Frameworks

Los frameworks de Javascript buscan optimizar la reutilización de elementos, llamados **components**.

Un componente puede ser:

- <HolaMundo></HolaMundo>
- <FooBar />

Los frameworks más comunes hoy en día son:

- React
- Angular
- VueJS



# Herramientas esenciales - Frameworks: React

Desarrollado por Facebook.

Se usa conjuntamente con las plantillas JSX para renderizar componentes.

Es muy modular.

Lo veréis en siguientes módulos



# Herramientas esenciales - Frameworks: Angular

Creado por Google

Cambian major versions cada 6 meses.

Hay una gran diferencia entre AngularJS (v1.x) y Angular(vx.y x≠1)

Angular usa Typescript y gran parte de su funcionamiento se basa en los patrones observables (vía librería RxJS)



# Herramientas esenciales - Frameworks: VueJS

Creado por Evan You con colaboradores de Netlify y Netguru

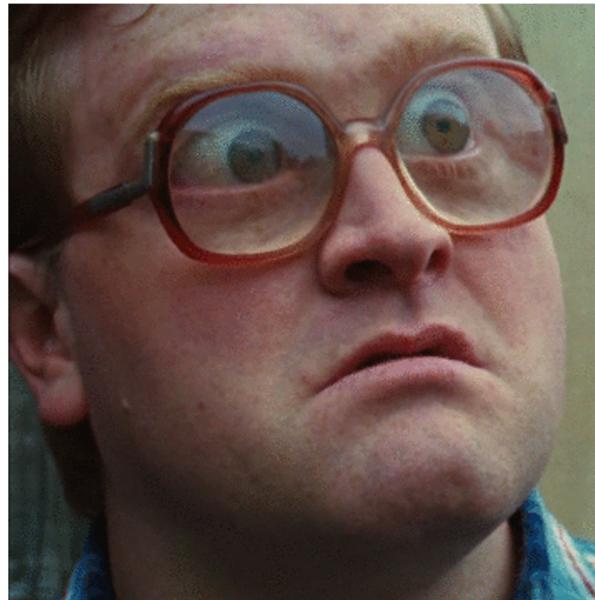
Muy fácil de usar, curva de aprendizaje rápida.

Están a punto de sacar la v3 estable.

Se complementa muy bien con material via plugin vuetyf.



Vale... pero no se pueden usar estos frameworks en una LP sin tener que compilar?  
Entonces son toolkits o frameworks???





# GRACIAS

[www.keepcoding.io](http://www.keepcoding.io)

