

INTRODUCCIÓN A LOS FRAMEWORKS DE JAVASCRIPT

Desarrollo Web en Entorno Cliente

¿QUÉ ES UN FRAMEWORK?

Un conjunto de herramientas, reglas y componentes prediseñados que nos ayudan a construir interfaces de manera más rápida y organizada

FRAMEWORKS

- Hay una gran variedad de frameworks disponibles, cada uno con sus ventajas específicas y áreas de enfoque.
- Frameworks como **React**, **Angular** y **Vue.js** nos permiten desarrollar interfaces de usuario dinámicas y reactivas, facilitando la creación de experiencias interactivas y modernas.
- Usarlos no sólo acelera el proceso de desarrollo, sino que también promueve buenas prácticas, como el uso de componentes reutilizables.

LO IMPORTANTE...

No se trata solo de usar tecnología avanzada; se trata de mejorar la experiencia del usuario final con interfaces **robustas, accesibles y atractivas.**

FRAMEWORKS DE JAVASCRIPT

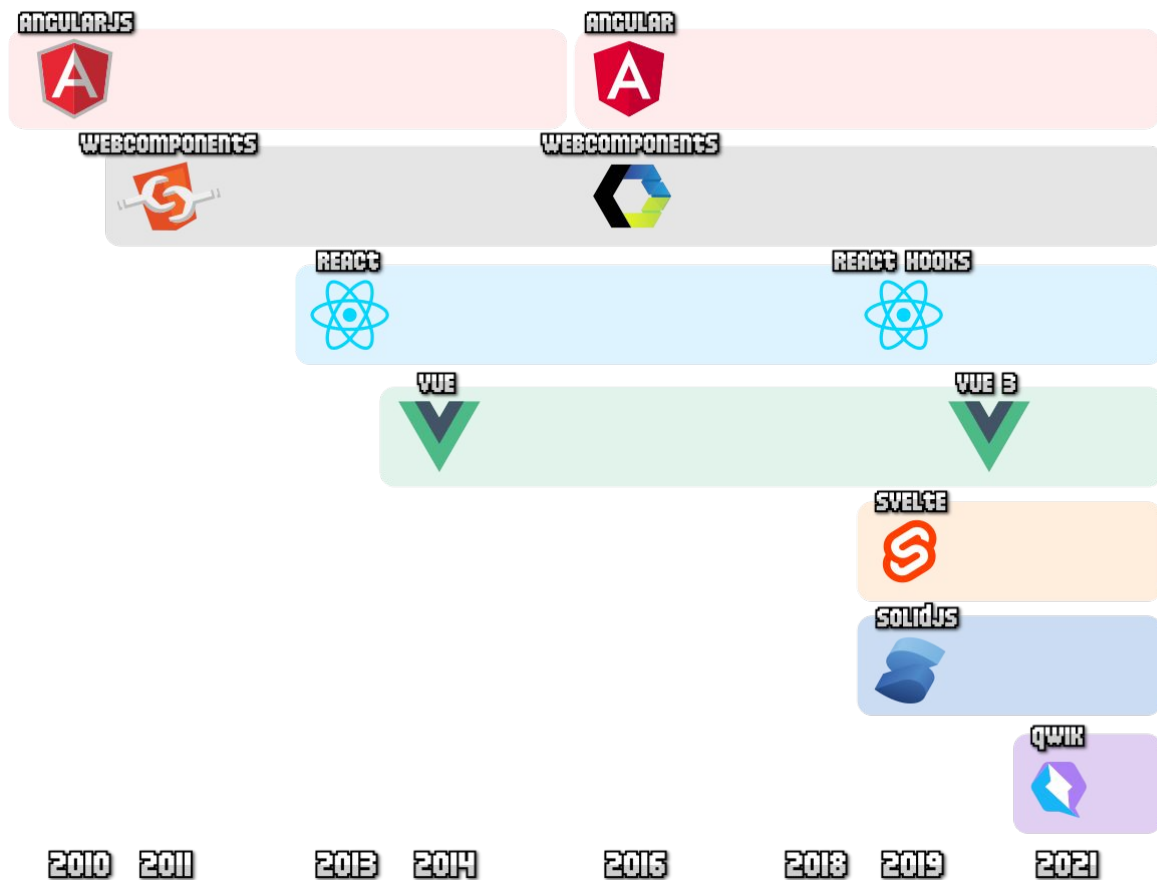
HISTORIA DE LOS FRAMEWORKS JAVASCRIPT

GUERRA ENTRE FRAMEWORKS

A lo largo de la última década, han surgido varias librerías y frameworks pero vamos a centrarnos en los que están destacando en el panorama actual, marcando una tendencia clara.

Todos ellos incorporan la posibilidad de crear **componentes**, puesto que se descubre que es un modelo que resuelve muchos problemas de escalabilidad en desarrollo y que resulta muy cómodo para trabajar.

FRAMEWORK RACE



LA TRAYECTORIA DE ANGULAR



AngularJS nace como un framework que revoluciona el panorama del desarrollo web, mostrando una forma de trabajar bastante cómoda y agradable.

En 2014 se anuncia que Angular comienza un nuevo camino como un framework que sufre un giro de 180 grados, reescribiéndose casi por completo y perdiendo la compatibilidad con sus versiones previas.

Actualmente, Angular es un framework monolítico que suele venir con casi todo preincorporado. Es por ello que su curva de aprendizaje es más compleja, pero también es por ello que suele ser la opción más escogida en las empresas.

LA TRAYECTORIA DE WEBCOMPONENTS

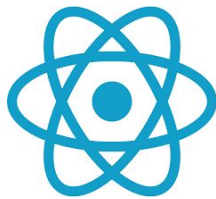


Los WebComponents no se tratan de un framework, ni de una librería, si no de extender el lenguaje HTML para incorporar componentes como un estándar (de forma nativa en el navegador).

No es hasta 2016 que se lanza la nueva versión de WebComponents, la cuál por fin es adoptada por la mayoría de navegadores (incluido Safari, que retrasó el despegue de los WebComponents por no soportarlo durante años) y comienza a ser posible utilizarlos de forma nativa.

Actualmente, los WebComponents suelen usarse en lugares donde los desarrolladores tienen preferencias por seguir un enfoque estándar o agnóstico, y apuestan por “La Plataforma Web”.

LA TRAYECTORIA DE REACT



En 2013 una de las grandes revoluciones en el mundo del desarrollo de frontend lo provoca React. Una librería de facebook para crear interfaces web de usuario que cambia de forma considerable la manera de hacer aplicaciones web, convirtiéndose en la actualidad, en casi un estándar.

React aborda un enfoque diferente donde se comienza a tomar Javascript como el centro neural de las aplicaciones web considerando tecnologías como HTML, CSS o similares, un complemento que se añade a Javascript.

Actualmente se podría decir que React persigue el objetivo de convertirse en una capa de abstracción para desarrollar tanto aplicaciones web, como aplicaciones móviles (React Native) u otras, más que un framework específico y único para aplicaciones web.

LA TRAYECTORIA DE VUE



Vue es un framework mucho más próximo a la línea de pensamiento de Frontend. Se basa en ciertos estándares y no abandona el enfoque `html-centric`, de modo que incluso sus componentes `.vue` son sólo una especie de HTML reinventado, que permite abordar marcado, estilos y funcionalidad.

En 2020 nace Vue 3 y se moderniza la aplicación, añadiendo un enfoque más vanilla (y similar a los hooks de React al utilizar la API de composición y permitiendo usar la API de opciones que ya venía usando en Vue 2).

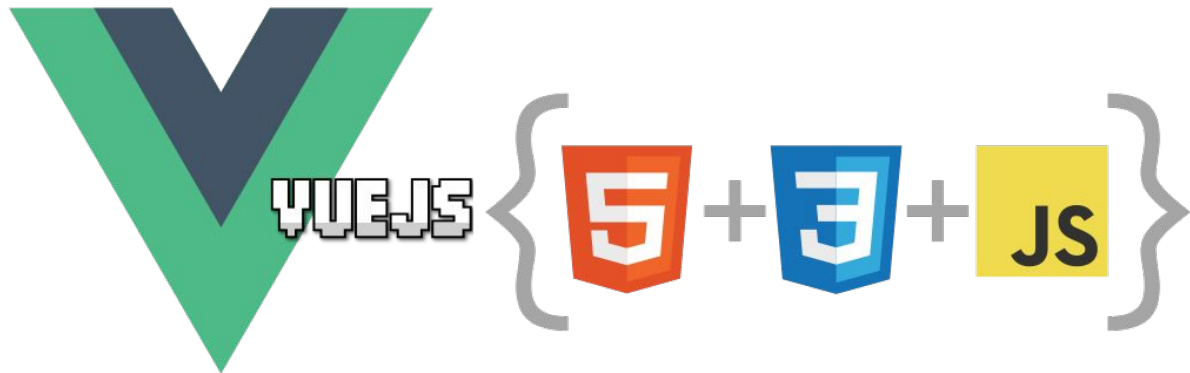


VueJS

¿QUÉ ES VUE?

¿QUÉ ES VUE?

Vue.js (también llamado Vue o VueJS) es un framework progresivo de Javascript para crear interfaces de usuario que nos permite crear aplicaciones de forma rápida, agradable, sencilla y muy práctica.



¿QUÉ ES VUE?

La palabra Vue (pronunciado “viú” viene del francés y significa “view” (vista) que es como se le denomina a la parte visual del modelo MVC, parte en la que se centran estos frameworks.

Vue es un framework frontend, es decir, que si vienes de framework del estilo de Laravel (PHP), Ruby on rails (Ruby), Spring (Java) u otros framework de backend, encontrarás múltiples diferencias. En los últimos años se ha popularizado la creación de aplicaciones de tipo SPA (Single Page Application), que es la categoría donde se encuentra Vue, React, Angular, etc...

¿POR QUÉ ELEGIR VUE?

Las características de Vue son las siguientes:

- La curva de aprendizaje es, con diferencia, la más sencilla de los tres frameworks más populares.
- Se trata de un framework muy amigable y respetuoso con las tecnologías de frontend y los estándares. Utiliza HTML, CSS y Javascript y es compatible con WebComponents (de hecho, sus componentes se basan en ellos).
- Se trata de un framework progresivo. Esto significa que es ideal para migrar y adaptar proyectos existentes hechos en otras tecnologías y pasarlos poco a poco a Vue.
- Vue le da mayor protagonismo al enfoque tradicional «centrado en **HTML**», así como a los sistemas de plantillas. Si te gustan, **Vue** probablemente te resulte muy atractivo. Por otro lado, el enfoque de **React** se suele centrar más en programación pura en Javascript, utilizando HTML y CSS sólo como complementos que se añaden a Javascript.

FRAMEWORK SIN OPINIÓN

Vue (al igual que React) se considera una herramienta no opinionated (sin opinión), o lo que es lo mismo, ni Vue ni React te van a guiar por una forma única de hacer las cosas, sino que a lo largo de tus proyectos lo habitual suele ser ir incorporando las características que necesites. También existen múltiples formas de hacer las mismas cosas, cosa que a ciertos desarrolladores les encanta, pero otros lo odian.

Por su parte, Angular se considera un framework opinionated (con opinión), ya que es más estricto en cuanto a las decisiones de arquitectura y te obligará en cierta forma a utilizar determinadas tecnologías o características (o al menos te hará más complicado hacerlo de otra forma).

MI PRIMERA APP CON VUE

COMENZAMOS

Siempre es bueno consultar la documentación oficial, ahí tenemos todo lo necesario para poder partir con nuestras aplicaciones. Les dejo el enlace [aquí](#) para que puedan revisarlo.

VUE CLI

Vue CLI es una interfaz de línea de comandos para el desarrollo de Vue que sirve como base del ecosistema de Vue.

En nuestro caso, nos permitirá crear una aplicación en Vue.

Primero, debemos asegurarnos de tener instalado Vue CLI, ejecutamos este comando en nuestra terminal:

```
sara_@Sara MINGW64 ~/Desktop/Vue
$ npm install -g @vue/cli
.
```

Si ya tenemos instalado Vue CLI, pero es una versión antigua, podemos actualizarlo con el siguiente comando:

```
sara_@Sara MINGW64 ~/Desktop/Vue  
$ npm update -g @vue/cli  
•
```

Comprobamos que todo está correctamente instalado:

```
sara_@Sara MINGW64 ~/Desktop/Vue  
$ vue --version  
@vue/cli 5.0.8
```

A continuación, para crear nuestro primer proyecto, ejecutamos:

```
sara_@Sara MINGW64 ~/Desktop/Vue
$ vue create mi-primer-vueapp
```

Si instalamos o actualizamos con éxito Vue CLI, deberíamos tener las siguientes opciones en consola:

```
Vue CLI v5.0.8
? Please pick a preset: (Use arrow keys)
> Default ([Vue 3] babel, eslint)
  Default ([Vue 2] babel, eslint)
  Manually select features
```


La primera opción hace referencia a si queremos hacer una creación default con Vue 3, la segunda opción, como pueden ver, hace referencia a una creación default con Vue 2 con Babel y Lint.

Para el ejemplo, usaremos la tercera opción, **Manually select features**. Bajamos con las flechas del teclado y presionamos “Enter”, se desplegará el siguiente listado:

```
Vue CLI v5.0.8
? Please pick a preset: Manually select features
? Check the features needed for your project: (Press <space> to select,
>(*) Babel
  ( ) TypeScript
  ( ) Progressive Web App (PWA) Support
  ( ) Router
  ( ) Vuex
  ( ) CSS Pre-processors
  (*) Linter / Formatter
  ( ) Unit Testing
  ( ) E2E Testing
```

Aquí tenemos todas las “**features**” que podemos seleccionar a la hora de crear un proyecto en Vue 3, aquí podríamos activar la opción para trabajar con TypeScript, o activar el soporte para aplicaciones PWA, si queremos añadir el sistema de rutas, trabajar con Vuex, añadir pre-procesadores de css, agregar Linter o añadir algún tipo de Testing.

La selección que haremos será la siguiente:

```
Vue CLI v5.0.8
? Please pick a preset: Manually select features
? Check the features needed for your project: (Press <space> to select, <a> to toggle all)
(*) Babel
( ) TypeScript
( ) Progressive Web App (PWA) Support
( ) Router
( ) Vuex
( ) CSS Pre-processors
> ( ) Linter / Formatter
( ) Unit Testing
( ) E2E Testing
```

Al presionar enter nos pedirá que seleccionemos la versión de vue que queremos usar, seleccionamos Vue 3:

```
Vue CLI v5.0.8
? Please pick a preset: Manually select features
? Check the features needed for your project: Babel
? Choose a version of Vue.js that you want to start the project with (Use arrow keys)
> 3.x
  2.x
```

Luego nos pedirá dónde queremos colocar la configuración para Babel, ESLint, etc. Le indicaremos que sea la opción de “**In dedicated config files**” y presionamos “Enter”:

```
Vue CLI v5.0.8
? Please pick a preset: Manually select features
? Check the features needed for your project: Babel
? Choose a version of Vue.js that you want to start the project with 3.x
? Where do you prefer placing config for Babel, ESLint, etc.? (Use arrow keys)
> In dedicated config files
  In package.json
```

Ahora nos preguntará si queremos guardar esta configuración para futuros proyectos, le diremos que No, digitando “N” y dando “**Enter**”, como es una configuración básica, tal vez no sea lo que deseemos guardar para futuros proyectos.

Esto comenzará a generar nuestro proyecto en Vue 3. Una vez que haya finalizado el proceso, ingresamos a nuestro proyecto con:

```
sara_@Sara MINGW64 ~/Desktop/Vue
$ cd mi-primer-vueapp/
```

y ejecutamos:

```
sara_@Sara MINGW64 ~/Desktop/Vue/mi-primer-vueapp (master)
$ npm run serve
```

DONE Compiled successfully in 2352ms

App running at:

- Local: <http://localhost:8080/>
- Network: <http://172.16.140.98:8080/>

Note that the development build is not optimized.
To create a production build, run `npm run build`.

Con el comando de arriba corriendo podemos ingresar a: <http://localhost:8080/>

Y si todo va bien...



Welcome to Your Vue.js App

For a guide and recipes on how to configure / customize this project,
check out the [vue-cli documentation](#).

Installed CLI Plugins

[babel](#)

Essential Links

[Core Docs](#) [Forum](#) [Community Chat](#) [Twitter](#) [News](#)

Ecosystem

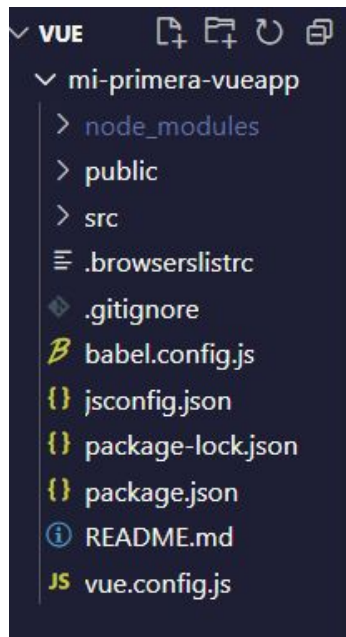
[vue-router](#) [vuex](#) [vue-devtools](#) [vue-loader](#) [awesome-vue](#)

¡Perfecto, ya tenemos nuestra primera aplicación en Vue 3 corriendo!



ANALIZANDO LA ESTRUCTURA DE NUESTRO PROYECTO

Si abrimos nuestro proyecto en Visual Studio Code, veremos una estructura inicial:



Y encontraremos lo siguiente:

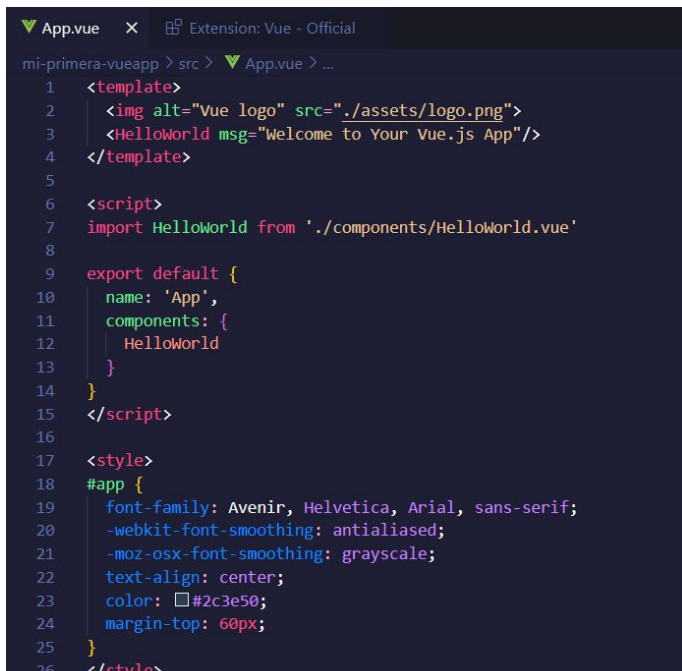
- **node_modules:** Esta carpeta contiene todas las dependencias que Vue necesita para funcionar, incluyendo las que nosotros instalemos.
- **public:** Esta carpeta contiene un archivo index.html y un archivo favicon.ico
- **src:** Esta carpeta “src” es donde vamos a construir nuestra aplicación, dentro de ella encontramos:

- > Carpeta “**assets**”, donde podemos almacenar imágenes, documentos o archivos multimedia que necesitemos utilizar dentro de nuestra aplicación.
- > Carpeta “**components**”, aquí iremos agregando y construyendo la estructura de componentes en nuestra aplicación.
- > **App.vue**, este suele ser el componente raíz de su aplicación, el cual no es más que un simple componente.
- > **main.js**, este es el primer archivo que se ejecutará en nuestra aplicación, también es responsable de configurar complementos y componentes de terceros que quizás desees utilizar en tu proyecto.

- **.browserslistrc:** especifica un rango de navegadores al que apunta el proyecto. Este valor será utilizado por [@babel/preset-env](#) y [autoprefixer](#) para determinar automáticamente las características de JavaScript que deben transpilarse y los prefijos de proveedor de CSS necesarios. Más detalles [aquí](#).
- **.gitignore:** En este archivo especificamos los archivos o carpetas que deseamos que sean ignorados o que no sean subidos a nuestros repositorios de Github, Bitbucket o GitLab.
- **babel.config.js:** Es un archivo para configurar Babel, Vue CLI usa `babel.config.js` que es un nuevo formato de configuración en Babel 7. Más detalles revisar [aquí](#)
- **package.json:** Este archivo es una especie de manifiesto, contiene varios metadatos relevantes para el proyecto. También es donde “npm” y “yarn” almacenan los nombres y versiones de los paquetes que instalados. Más detalles [aquí](#).
- **README.md:** Este archivo usualmente es utilizado como una forma de documentación de software, donde se muestra cómo arrancar un proyecto o información relevante de su aplicación.
- **vue.config.js:** permite personalizar y extender la configuración predeterminada del proyecto que genera Vue CLI, sin la necesidad de modificar directamente la configuración interna de Webpack u otras herramientas subyacentes.

PRIMEROS CAMBIOS...

Vamos a modificar el componente **App.vue** para que vean lo sencillo que es, dentro de nuestro proyecto vamos a `src > App.vue`, lo abrimos y podemos ver la siguiente estructura en nuestro componente:

A screenshot of a code editor window showing the App.vue file. The editor has a dark theme. The file path in the breadcrumb is 'mi-primera-vueapp > src > App.vue > ...'. The code is as follows:

```
1 <template>
2   
3   <HelloWorld msg="Welcome to Your Vue.js App"/>
4 </template>
5
6 <script>
7   import HelloWorld from './components/HelloWorld.vue'
8
9   export default {
10     name: 'App',
11     components: {
12       HelloWorld
13     }
14   }
15 </script>
16
17 <style>
18 #app {
19   font-family: Avenir, Helvetica, Arial, sans-serif;
20   -webkit-font-smoothing: antialiased;
21   -moz-osx-font-smoothing: grayscale;
22   text-align: center;
23   color: #2c3e50;
24   margin-top: 60px;
25 }
26 </style>
```

Como ven, un componente en Vuejs se podría dividir o estructurar en tres partes:

<template>: Aquí es dónde agregamos nuestro html

<script>: Aquí agregamos nuestro javascript, importaciones y/o lógica de nuestro componente.

<style>: Y aquí agregamos los estilos css para nuestro componente.

Vamos a modificar <template>, quitaremos el componente “HelloWorld.vue” y agregaremos el típico “Hola Mundo!”:

```
▼ App.vue M x  Extension: Vue - Official
mi-primera-vueapp > src > ▼ App.vue > ...
1  <template>
2    
3    <!--<HelloWorld msg="Welcome to Your Vue.js App"/>-->
4    <h1>¡Hola mundo!</h1>
5  </template>
6
7  <script>
8    //import HelloWorld from './components/HelloWorld.vue'
9
10   export default {
11     name: 'App',
12     /*components: {
13       HelloWorld
14     }*/
15   }
16 </script>
17
18 <style>
19 #app {
20   font-family: Avenir, Helvetica, Arial, sans-serif;
21   -webkit-font-smoothing: antialiased;
22   -moz-osx-font-smoothing: grayscale;
23   text-align: center;
24   color: #2c3e50;
25   margin-top: 60px;
26 }
```



¡Hola mundo!

¡Genial, ya creaste tu primera aplicación en Vue 3 y modificaste tu primer componente!

