TECNOLOGÍAS MODERNAS PARA EL DESARROLLO FRONTEND EN APLICACIONES WEB

JOHN ANDREY SERRANO CASELLES

UNIVERSIDAD SANTO TOMÁS

FACULTAD DE INGENIERÍA

PROGRAMA DE INGENIERÍA INFORMATICA

CÚCUTA

2022

Introducción

En el mundo digital tenemos por una parte el desarrollo frontend y el backend donde el frontend es la parte de un sitio web que interactúa con los usuarios, por eso decimos que está del lado del cliente. Backend es la parte que se conecta con la base de datos y el servidor que utiliza dicho sitio web. El frontend son todas las tecnologías de diseño y desarrollo web que corren en el navegador y que se encargan de la interactividad con los usuarios.

Las tecnologías más demandas y usadas para el desarrollo frontend en los últimos años tomando como referencia la encuesta State of JS creada y mantenida por Sacha Greif y Raphaël Benitte, son las siguientes: React.js, Vue.js y Svelte. La mayoría de estas herramientas se les llama framework a excepción de React.js que en realidad es una librería o biblioteca, cada una de ellas tiene sus ventajas y desventajas, tienen sus propias comunidades y grandes empresas que le dan soporte y respaldo.

Estas tecnologías han tenido un gran crecimiento permitiendo a los desarrolladores e ingenieros hacer sitios y aplicaciones web mas atractivas para los usuarios, de igual forma se hace una experiencia mas sostenible y amigable en el tiempo para los desarrolladores permitiendo una mejor experiencia en el desarrollo de sitios web y en el mantenimiento de grandes proyectos siguiendo unos patrones de diseño, de carpetas y un orden a la hora de implementar soluciones para los usuarios.

Gracias a todas estas tecnologías se ha podido mejorar la experiencia del usuario el rendimiento web, el SEO para tener mejor posicionamiento en los resultados de los buscadores, cada una de estas tecnologías tiene su propia curva de aprendizaje para los ingenieros de software, cada una de ellas cuenta con sus ventajas y sus desventajas todo depende del tipo de proyecto que se quiera llegar a realizar y el soporte que se le quiera dar a largo tiempo.

En la presente monografía se verán cada una de estas tecnologías que son como funcionan que tienen de particular una de la otra, cuales son más usadas, las que tienen mayores intereses, sus ventajas y desventajas partiendo de la encuesta State of JS con el fin de conocerlas y saber su nivel de popularidad como de uso en el desarrollo web.

React.js

En la encuesta de State of JS 2020, esta encuesta es muy famosa y se aplica a una gran cantidad de Ingenieros de software o desarrolladores web de todo el mundo, en el año 2020 la encuesta ha llegado a 23,765 personas en 137 países. En esta encuesta se puede observar el panorama y comportamiento que ha tenido React.js en los últimos años.

Siguiendo los datos de la encuesta se puede observar que el porcentaje de satisfacción para el año 2020 es del 88% con respecto a las otras herramientas ubicándose en el segundo lugar. En el interés se ubica en la tercera posición con un porcentaje de 58%, en el uso de la herramienta se ubica en la primera posición con un porcentaje de 80%.

Para el State of JS 2021 se puede encontrar que en satisfacción se encuentra en la tercera posición con un porcentaje de 84% cayendo una posición con respecto al año anterior a nivel de satisfacción, a nivel de interés se ubica en la cuarta posición con un porcentaje de 48% y en el uso de la herramienta se ubica en la primera posición con 80%.

Realmente a nivel de satisfacción no varía mucho con respecto a las otras herramientas la mayor variación se ve con herramientas relativamente nuevas. En cuanto el interés es normal que vaya en decline ya que es una herramienta que lleva muchos años en el mercado. Y en cuanto el uso no hay variación hoy en día sigue siendo la herramienta más usada para el desarrollo del front-end.

¿Qué es React.JS?

React es una biblioteca o librería de código abierto que está escrita en JavaScript. Fue desarrollada por Facebook en el 2013 con la finalidad de facilitar la creación de componentes reutilizables e interactivos para las interfaces de usuario.

Uno de sus puntos más destacados es que no solo se usa en el lado del cliente, sino que también se puede representar en el servidor y trabajar juntos.

Son datos interesantes que React se utiliza en Facebook para la producción de componentes y que Instagram está escrito enteramente en React. Adicionalmente, también se utiliza en otras plataformas como Netflix, PayPal, AirBnb, Uber, Reddit y Twitter.

React es una excelente alternativa para realizar todo tipo de aplicaciones web o para dispositivos móviles, así como para crear single page applications (SPA o aplicaciones de una sola página).

También se destaca por contar con un completo ecosistema de componentes, herramientas y módulos que posibilitan el desarrollo de funcionalidades complejas en poco tiempo.

Tres pilares de React.js:

Es una librería declarativa

React ayuda a crear interfaces de usuario interactivas de forma sencilla. Diseña vistas simples para cada estado en tu aplicación, y React se encargará de actualizar y renderizar de manera eficiente los componentes correctos cuando los datos cambien.

Las vistas declarativas hacen que tu código sea más predecible, por lo tanto, fácil de depurar.

Está Basado en componentes

Crear componentes encapsulados que manejen su propio estado, y conviértelos en interfaces de usuario complejas. Ya que la lógica de los componentes está escrita en JavaScript y no en plantillas, puedes pasar datos de forma sencilla a través de tu aplicación y mantener el estado fuera del DOM.

Aprende una vez, escríbelo donde sea

En React no se deja por fuera el resto de las herramientas tecnológicas, así que se puede desarrollar nuevas características sin necesidad de volver a escribir el código existente.

React se puede también renderizar desde el servidor usando Node.js, así como potencializar aplicaciones móviles usando React Native.

¿Qué es Node.js?

Ideado como un entorno de ejecución de JavaScript orientado a eventos asíncronos, Node.js está diseñado para crear aplicaciones network escalables. Se pueden atenderse muchas conexiones simultáneamente. Por cada conexión, se activa la devolución de llamada o callback, pero si no hay trabajo que hacer, Node.js se dormirá.

Esto contrasta con el modelo de concurrencia más común de hoy en día, en el que se emplean hilos del Sistema Operativo. Las redes basadas en hilos son relativamente ineficientes y muy difíciles de usar. Además, los usuarios de Node.js están libres de preocuparse por el bloqueo del proceso, ya que no existe. Casi ninguna función en Node.js realiza I/O directamente, por lo que el proceso nunca se bloquea. Por ello, es muy propicio desarrollar sistemas escalables en Node.js.

Node.js es similar en diseño y está influenciado por sistemas como Event Machine de Ruby y Twisted de Python. Pero Node.js lleva el modelo de eventos un poco más allá. Incluye un bucle de eventos como runtime de ejecución en lugar de una biblioteca. En otros sistemas siempre existe una llamada de bloqueo para iniciar el bucle de eventos.

HTTP es un elemento destacado en Node.js, diseñado teniendo en cuenta la transmisión de operaciones con streaming y baja latencia. Esto hace que Node.js sea muy adecuado para la base de una librería o un framework web.

Que Node.js esté diseñado para trabajar sin hilos no significa que no pueda aprovechar múltiples núcleos en su entorno. Se pueden generar subprocesos o procesos hijos utilizando nuestra API `child_process.fork`, la cual está diseñada para que la comunicación entre ellos sea fácil mediante su proceso principal.

¿Cómo funciona React JS?

Para comprender cómo funciona React es clave que se situé en un contexto, pues cuando se aprende desarrollo web se obtiene conocimiento de tres conceptos básicos:

HTML: la semántica, estructura e información de la página web; es decir, su esqueleto.

CSS: la apariencia de nuestra página web.

JavaScript: básicamente es el cerebro de nuestra página. Determina qué hacer en función de lo que sucede en ella.

Sin embargo, antes de React estos conceptos funcionaban por separado, en diferentes archivos y carpetas, por lo que escalar y extraer diversas partes del código para reutilizar o migrar funcionalidades era más complicado.

Por esto, los ingenieros de Facebook decidieron incluir todo en un solo paquete, al que llamaron “componente”. En los componentes, la estructura HTML y JS son inseparables, y combinables con CSS.

Esto hizo posible la implementación de una nueva notación que hace más eficiente el desarrollo de aplicaciones escalables: JSX, que significa JavaScript XML.

Hay al menos dos cosas que hacen especial a React.js en comparación de las otras herramientas del frontend que es la notación o sintaxis JSX y el virtual DOM, en esta última tiene similitudes con Vue.js.

¿Qué es JSX?

Consiste en una extensión de la sintaxis de JavaScript que permite que compilemos un objeto JavaScript para mapear un elemento del DOM (Objeto del Documento). A través de JSX se crea un DOM virtual (que es una representación en memoria del DOM) que pesa muy poco y usa menos recursos.

Con esto, cuando React es informado de un cambio de estado, vuelve a ejecutar esas funciones y determina una nueva representación virtual de dicha página. Posterior a esto,

traduce automáticamente ese resultado en los cambios del DOM necesarios, reflejando así la nueva presentación de la página.

A primera vista, esto suena como que fuera más lento que el enfoque JavaScript habitual de actualización de cada elemento según sea necesario. Sin embargo, detrás de escena, React JS tiene un algoritmo muy eficiente para determinar los aspectos desiguales entre la representación virtual de lo que es la página actual y la nueva. A partir de esas diferencias, crea el conjunto mínimo de cambios necesarios en el DOM.

Para esto, utiliza un concepto llamado DOM virtual, el cual selecciona subárboles de los nodos sobre la base de cambios de estado. De esta manera mantiene los componentes actualizados con la menor cantidad de manipulación DOM posible.

¿Cómo funciona el DOM virtual?

Imaginando que se tiene un objeto que es un modelo en torno a una persona. Se tiene todas las propiedades relevantes de una persona que podría tener, y refleja el estado actual de la persona. Esto es básicamente lo que React hace con el DOM.

Ahora si se piensa en que, si tomamos a esa persona y le hacemos algunos cambios, es decir, le añadimos bigote y tatuajes en los brazos, es claro que lo notarás. Y React también lo hace.

Entonces, cuando se aplican estos cambios, ocurren dos cosas a continuación:

- En primer lugar, React ejecuta un algoritmo de diffing, que identifica lo que ha cambiado.
- El segundo paso es la reconciliación, donde se actualiza el DOM con los resultados de diff.

En pocas palabras, React JS está construido para tomar las actualizaciones de estado de la página y que se traduzcan en una representación virtual de la página resultante.

Lo que hace React ante las variaciones —en lugar de tomar la imagen real y reconstruirla desde cero— es tomar los cambios de la cara y los brazos (o cualquiera que sea el cambio

que se haya realizado). Esto significa que, si se tenía el texto en una entrada y se llevó a cabo una actualización con React, el texto no sería cambiado.

Figura 1

Ejemplo de un componente simple

```
function HelloMessage ({ name }) {
  return <div>Hola {name} </div>
}

root.render(<HelloMessage name="John Andrey" />);
```

Nota: Los componentes de React implementan un método llamado render() que recibe datos de entrada y retorna qué mostrar. Puedes acceder a los datos de entrada que se pasan al componente mediante render() a través de this.props.

Ventajas y desventajas

Ventajas

- Es fácil saber cómo se renderiza un componente, solo mira la función de renderizado.
- JSX facilita la lectura del código de sus componentes. También es muy fácil ver el diseño o cómo se conectan/combinan los componentes entre sí.
- Puede renderizar React en el lado del servidor.
- Es fácil de probar y también puedes integrar algunas herramientas como jest.
- Asegura la legibilidad y facilita el mantenimiento.
- Puede usar React con cualquier marco (Backbone.js, Angular.js) ya que es solo una capa de vista.
- Librería JavaScript 100% código abierto con frecuentes actualizaciones y mejoras de desarrolladores de todo el mundo.

Desventajas

- Es solo una capa de vista, aún tiene que conectar su código para solicitudes de Ajax, eventos, etc.
- La biblioteca en sí es bastante grande.
- La curva de aprendizaje puede ser empinada.

- No existe un estándar de desarrollo, de modo que tenemos demasiadas elecciones a tomar.

Vue.js

Volviendo a la encuesta de State of JS 2020 viendo los resultados, en la parte de satisfacción vue.js se encuentra en la tercera posición con 85%, en el interés tiene un 63% ubicándose en la segunda posición y en cuanto al uso de la herramienta se ubica en la tercera posición con un 49%.

En el State of JS 2021 se puede observar que en la satisfacción tiene un 80% pero ubica en la cuarta posición, en el interés tiene un 50% ubicándose en la tercera posición perdiendo una posición con respecto al año anterior y en el uso de la herramienta se sigue manteniendo en la tercera posición con un 51%.

Como conclusiones sobre la encuesta se puede observar que la satisfacción a perdido una posición en los últimos años ya que llego una nueva herramienta de desarrollo para el frontend, en cuanto el interés pasa prácticamente lo mismo que lo mencionado anteriormente y en cuanto el uso tiene el mismo comportamiento de los últimos años manteniendo la posición.

¿Qué es Vue.js?

Creado por Evan You, Vue (pronunciado /vjuː/, como view) es un framework progresivo para construir interfaces de usuario. A diferencia de otros frameworks monolíticos, Vue está diseñado desde cero para ser utilizado incrementalmente. La librería central está enfocada solo en la capa de visualización, y es fácil de utilizar e integrar con otras librerías o proyectos existentes. Por otro lado, Vue también es perfectamente capaz de impulsar sofisticadas Single-Page Applications cuando se utiliza en combinación con herramientas modernas y librerías de apoyo.

Algunas cosas que comparte Vue.js con React.js son:

- utilizan un DOM virtual
- proporcionan componentes de vista reactivos y componibles
- mantienen el enfoque en la librería central, con temas como el enrutamiento y la gestión global del estado manejadas por librerías asociadas

A continuación, se describe una comparación con React ya que comparten cosas:

Rendimiento

Tanto React como Vue ofrecen un rendimiento comparable en los casos de uso más comunes, con Vue normalmente un poco por delante debido a su implementación más ligera del DOM virtual.

Esfuerzos de optimización

En React, cuando el estado de un componente cambia, activa el re-renderizado de todo el subárbol de componentes, comenzando en ese componente como raíz. Para evitar repeticiones innecesarias de componentes hijo, necesita usar PureComponent o implementar shouldComponentUpdate siempre que pueda. También es posible que necesite utilizar estructuras de datos inmutables para que los cambios de estado sean más fáciles de optimizar. Sin embargo, en algunos casos puede que no pueda confiar en dichas optimizaciones porque PureComponent/shouldComponentUpdate asume que el resultado del renderizado de todo el subárbol está determinado por las propiedades del componente actual. Si ese no es el caso, entonces tales optimizaciones pueden llevar a un estado DOM inconsistente.

En Vue, las dependencias de un componente se rastrean automáticamente durante su renderizado, por lo que el sistema sabe con precisión qué componentes deben volver a renderizarse cuando cambia el estado. Se puede considerar que cada componente tiene shouldComponentUpdate automáticamente implementado, sin las precauciones de los componentes anidados.

En general, esto le elimina la necesidad de toda una clase de optimizaciones de rendimiento a los desarrolladores, y les permite centrarse más en la construcción de la aplicación en sí misma a medida que va escalando.

HTML Y CSS

En React, todo es únicamente JavaScript. No sólo las estructuras HTML se expresan a través de JSX, sino que las tendencias recientes también apuntan a poner la gestión de CSS dentro de JavaScript. Este enfoque tiene sus propios beneficios, pero también viene con varias desventajas que no parecen valer la pena para todos los desarrolladores.

Vue adopta las tecnologías web clásicas y construye a partir de ellas. Para mostrarles lo que eso significa, nos sumergiremos en algunos ejemplos.

JSX vs Plantillas

En React, todos los componentes expresan su interfaz de usuario dentro de las funciones de render utilizando JSX, una sintaxis declarativa similar a XML que funciona con JavaScript.

Las funciones de renderizado con JSX tienen algunas ventajas:

Puede aprovechar la potencia de un lenguaje de programación completo (JavaScript) para crear su vista. Esto incluye variables temporales, controles de flujo y referencias directas a los valores JavaScript en el ámbito de aplicación.

El soporte de herramientas (p.e. linting, comprobación de tipos, autocompletado del editor) para JSX es en algunos aspectos más avanzado que el disponible actualmente para las plantillas de Vue.

En Vue, también tenemos funciones de renderizado e incluso soporte JSX, porque a veces se necesita esa potencia. Sin embargo, como experiencia por defecto ofrecemos plantillas como una alternativa más simple. Cualquier HTML válido es también una plantilla Vue válida, y esto lleva a algunas ventajas propias:

Para muchos desarrolladores que han trabajado con HTML, las plantillas se sienten más naturales para leer y escribir. La preferencia en sí misma puede ser algo subjetiva, pero si hace que el desarrollador sea más productivo, entonces el beneficio es objetivo.

Las plantillas basadas en HTML hacen mucho más fácil la migración progresiva de las aplicaciones existentes para aprovechar las características de reactividad de Vue.

También hace que sea mucho más fácil para los diseñadores y desarrolladores menos experimentados analizar y contribuir a la base del código. Incluso puede utilizar preprocesadores como Pug (antes conocido como Jade) para crear sus plantillas de Vue.

Algunos argumentan que necesitarías aprender un DSL (Domain-Specific Language) extra para poder escribir plantillas - creemos que esta diferencia es superficial en el mejor de los casos. Primero, JSX no significa que el usuario no necesite aprender nada - es sintaxis adicional encima del JavaScript regular, así que puede ser fácil de aprender para alguien familiarizado con JavaScript, pero decir que es esencialmente gratis es engañoso. Del mismo modo, una plantilla es sólo una sintaxis adicional encima del HTML regular y, por lo tanto, tiene un coste de aprendizaje muy bajo para aquellos que ya están familiarizados con HTML. Con el DSL también podemos ayudar al usuario a hacer más con menos código (por ejemplo, los modificadores v-on). La misma tarea puede implicar mucho más código cuando se utilizan funciones JSX o de render.

En un nivel superior, podemos dividir los componentes en dos categorías: los de presentación y los lógicos. Se recomienda el uso de plantillas para los componentes de presentación y la función de render / JSX para los lógicos. El porcentaje de estos componentes depende del tipo de aplicación que se esté construyendo, pero en general se encuentra que los de presentación son mucho más comunes.

CSS con alcance a componentes

A menos que extienda los componentes sobre múltiples archivos (por ejemplo, con Módulos CSS) el alcance de CSS en React suele hacerse a través de soluciones CSS-en-JS (por ejemplo, componentes de estilo, glamorous, y emotion). Esto introduce un nuevo paradigma de estilo orientado a componentes que es diferente del proceso normal de creación de CSS. Además, aunque hay soporte para extraer CSS en una sola hoja de estilo en el momento de la compilación, sigue siendo común que se necesite incluir un tiempo de ejecución en el paquete para que el estilo funcione correctamente. Mientras que usted obtiene acceso al dinamismo de JavaScript mientras construye sus estilos, la compensación es a menudo el aumento del tamaño del paquete y el costo del tiempo de ejecución.

Escala

Para aplicaciones grandes, tanto Vue como React ofrecen soluciones de enrutamiento robustas. La comunidad de React también ha sido muy innovadora en términos de

soluciones de gestión de estado (por ejemplo, Flux/Redux). Estos patrones de gestión de estado e incluso el propio Redux pueden integrarse fácilmente en las aplicaciones Vue. De hecho, Vue ha llevado este modelo un paso más allá con Vuex, una solución de gestión de estado inspirada en Elm que se integra profundamente en Vue y que creemos ofrece una experiencia de desarrollo superior.

Otra diferencia importante entre estas ofertas es que las librerías adicionales de Vue para la gestión de estados y enrutamiento (entre otros asuntos) están todas oficialmente soportadas y se mantienen actualizadas con la librería central. React, en cambio, opta por dejar estas preocupaciones en manos de la comunidad, creando un ecosistema más fragmentado. Siendo más popular, sin embargo, el ecosistema de React es considerablemente más rico que el de Vue.

Finalmente, Vue ofrece un generador de proyectos CLI que hace que sea trivialmente fácil comenzar un nuevo proyecto usando el sistema de compilación elegido, incluyendo webpack, Browserify, o incluso ningún sistema de compilación. React también está haciendo progresos en esta área con create-react-app, pero actualmente tiene algunas limitaciones:

No permite ninguna configuración durante la generación del proyecto, mientras que las plantillas de proyecto de Vue permiten una personalización similar a la de Yeoman.

Sólo ofrece una única plantilla que asume que está construyendo una aplicación de una sola página, mientras que Vue ofrece una amplia variedad de plantillas para diversos propósitos y sistemas de construcción.

No puede generar proyectos a partir de plantillas construidas por el usuario, lo que puede ser especialmente útil para entornos empresariales con convenciones preestablecidas.

¿Porque Vue.js?

La modularización de bibliotecas utilizando un marco es común en el desarrollo de frontend. Tanto React como Angular tienen modularización. Pero lo que diferencia a Vue.js de otras alternativas es su "alto desacoplamiento", lo fácil que es extender las funcionalidades y lo bien que funcionan todas las partes una vez que se incluyen más módulos. Por ejemplo, si se quiere organizar y renderizar pequeños componentes visuales, todo lo que necesitamos es la biblioteca "central" de Vue.js; no es necesario incluir bibliotecas adicionales. A medida que la aplicación crece, se tienen bibliotecas para administrar rutas como "vue-router", bibliotecas para administrar el estado global como "vuex" o bibliotecas para crear aplicaciones web receptivas como "bootstrap-vue". Además, si la aplicación necesita ser optimizada o necesita un buen SEO, se puede incluir la biblioteca "vue-server-rendering".

El sistema de componentes de Vue.js es reactivo, lo que significa que Vue.js sabe cómo comunicarse a través de eventos asincrónicos; por ejemplo, un componente hijo puede comunicarse con su componente padre a través de eventos. Con Vue.js no hay fricción con otras bibliotecas o recursos, es decir, podemos utilizar la herramienta con la que nos sentimos más cómodos. Por ejemplo, podemos escribir solo HTML y JavaScript o si queremos podemos agregar CSS, JSX o TypeScript.

Vue.js tiene una línea de comando especial (CLI) creada en Node JS. Esta herramienta permite iniciar un proyecto utilizando un modelo estándar (o plantilla base). Además, el equipo de desarrollo de Vue.js mantiene una extensión de Chrome que permite ver cómo se representa nuestro árbol de componentes, cómo se inician y registran los eventos, cómo se guarda el estado interno de cada componente y cómo se muestra el estado global del componente.

Figura 2

Ejemplo de un renderizado declarativo

```
<div id="app">
  {{ message }}
</div>
```

```
var app = new Vue({
  el: '#app',
  data: {
    message: 'Hola Vue!'
  }
})
```

Nota: En el núcleo de Vue.js se encuentra un sistema que permite procesar datos de forma declarativa al DOM mediante una sintaxis de plantilla sencilla. Esto parece bastante similar a la representación de un template string, pero Vue ha hecho mucho trabajo por debajo. Los datos y el DOM ahora están vinculados, y todo ahora es reactivo.

Ventajas y Desventajas

Ventajas

- Desglosar los componentes en archivos individuales.
- Fácil de aprender y utilizar, teniendo una curva de aprendizaje baja.
- HTML empoderado. Esto significa que Vue.js tiene muchas similitudes con Angular y esto puede ayudar a optimizar el manejo de bloques HTML usando diferentes componentes.
- Documentación detallada. Vue.js tiene una documentación muy completa, que puede acelerar la curva de aprendizaje para desarrolladores y ahorrar mucho tiempo en el desarrollo de una aplicación usando sólo conocimientos básicos de HTML y JavaScript.
- Adaptabilidad. Proporciona un rápido periodo de cambio desde otros frameworks a Vue.js por sus similitudes con Angular y React en términos de diseño y arquitectura.
- Excelente integración. Vue.js puede usarse para construir tantas aplicaciones de página única o complejas interfaces web de aplicaciones. Lo principal son las

pequeñas partes interactivas que pueden ser integradas fácilmente en una infraestructura existente sin efectos negativos en el sistema.
- Gran escalado. Vue.js puede ayudarnos a construir grandes plantillas reutilizables en poco tiempo de acuerdo con su sencilla estructura.
- Pequeño tamaño. Vue.js puede ocupar cerca de 20KB manteniendo su velocidad y flexibilidad que permite alcanzar un mejor rendimiento en comparación con otros frameworks.

Desventajas

- Ser un framework nuevo y con pocos años en el mercado.
- Falta de recursos. Vue.js aún tiene poca cuota de mercado comparado con Angular o React, lo que significa que los recursos disponibles de este framework aún están en su fase inicial.
- Riesgo de excesiva flexibilidad. En ocasiones, Vue.js puede tener problemas para integrarse en grandes proyectos y aún no hay experiencia acerca de posibles soluciones, aunque a buen seguro estarán disponibles pronto.
- Falta de documentación completa en inglés. Esto nos lleva a una complejidad en ciertas fases del desarrollo, sin embargo, de cada vez hay más material traducido al inglés.

Svelte

Retomando la encuesta State of JS 2020 se puede ver los resultados de svelte, a nivel de satisfacción se posiciona en el puesto número uno con un 89%, a nivel de interés sigue manteniendo la primera posición con un 66% y en el uso si cae hasta la posición cuarta con un 15%, al ser una herramienta tan nueva es normal que a nivel del uso no sea tan alta a diferencia de la satisfacción e interés que ocupan la primera posición ya que se espera mucho de esta tecnología para el desarrollo del frontend.

En la encuesta State of JS 2021 se puede observar que a nivel de satisfacción o retención se ubica en la segunda posición con un 90% estando a la par con una nueva herramienta que surgió en el mercado, en la parte de interés sigue manteniendo la primera posición con un 68% y en cuanto el uso mantiene la cuarta posición con un 20%, como se mencionó anteriormente es algo normal que el uso no sea tan grande ya que es relativamente una herramienta nueva.

¿De dónde sale Svelte.js?

Svelte fue creado por Rich Harris, quien es desarrollador y que por mucho tiempo buscaba una forma de mejorar el desempeño de las aplicaciones que desarrollaba, él trabaja cumpliendo ciertas características:

- Entregar productos con un excelente rendimiento y fácil desarrollo.
- Que la librería o framework tuviera una curva de aprendizaje muy corta para que otras personas puedan integrarlo en sus proyectos sin mucha complicación o boilerplate.
- Hacer más liviano el trabajo del navegador del lado cliente y reducir lo más posible la carga de código de librerías o frameworks por código JavaScript puro.

¿Qué es Svelte?

Svelte es un frontend framework. Tiene conceptos parecidos a React, Vue y Angular pero, a la vez, viene con diferencias muy importantes. Concretamente hay una diferencia importantísima y es que Svelte tiene un paso previo de compilación. ¿Por qué? Porque el código que escribimos con Svelte, aunque está basado en lenguajes que ya conoces como HTML, CSS y JavaScript, es un código que no entendería el navegador sin tratar y tiene que ser compilado previamente.

Aunque muchos pueden ver esto como una desventaja, en realidad, es algo con lo que ya convivimos de alguna forma con el resto de frameworks por el uso de ES2015+ y JSX. Aunque, de alguna forma, el resto de las bibliotecas y frameworks sí podrían obviar este paso, con Svelte es completamente necesario.

Svelte es un enfoque radicalmente nuevo para crear interfaces de usuario. Mientras que los marcos tradicionales como React y Vue hacen la mayor parte de su trabajo en el navegador, Svelte cambia ese trabajo a un paso de compilación que ocurre cuando construyes tu aplicación.

En lugar de usar técnicas como la diferenciación de DOM virtual, Svelte escribe un código que actualiza quirúrgicamente el DOM cuando cambia el estado de su aplicación.

Adiós Virtual DOM

No sólo la necesidad de compilar es la diferencia. También la total ausencia de Virtual DOM, o VDOM. El VDOM, que popularizó React en su día, guarda una foto del árbol de elementos del DOM en memoria de forma que, al tener que hacer actualizaciones, sabe los cambios mínimos a realizar. Esto lo hace porque manipular el DOM es lento y, de esta forma, se evita trabajo innecesario.

Entonces, si es algo bueno, ¿por qué Svelte no lo usa? Porque no es gratis. El diffing que realiza el VDOM tiene su coste, y peor será cuanto más compleja sea nuestra app. ¿Eso quiere decir que Svelte es más lento que React y renderiza elementos de forma innecesaria?

Nada de eso. Como Svelte es un compilador, lo que hace, es evitar el uso del Virtual DOM y aprovechándose del paso de compilación, para envolver los cambios de estado y propiedades en métodos que, de forma quirúrgica, podrán actualizar el DOM.

Características de Svelte.js

- No utiliza virtual DOM para realizar cambios en la aplicación, como ya se mencionó anteriormente.
- Tiene su propio manejador de estado (Stores): Para los que sufren con entender Redux o otras librerías, Svelte ya trae su propio manejador y bastante más sencillo de aprender.
- Su sintaxis es sencilla: Esto ayuda al desarrollador escribir componentes con los lenguajes que ya conoce como HTML, CSS, JavaScript e incluso para quienes buscan aprender su primer framework JavaScript esta sería su oportunidad de darle un vistazo.
- Totalmente reactivo: Ayuda en la escalabilidad de nuestros proyectos para que sean más mantenibles y robustos.
- El tamaño de las apps que compilas a producción es bien reducido a diferencia de otras alternativas y con código optimizado para producción.
- La documentación es super mega fácil de seguir paso a paso.
- Puedes utilizarlo para el server side rendering (SSR) con Sapper (propió de Svelte)

Figura 3

Ejemplo de un renderizado en Svelte

Nota: Svelte utiliza el principio de Single File Component. Esto en un mismo archivo, tiene un componente con su marcado (HTML), su funcionalidad (Javascript) y su estilo (CSS). Esto lo ha popularizado mucho Vue y, en Svelte, se puede ver que es incluso un poco más sencillo.

Con esto ya podemos ver las primeras diferencias con otras alternativas como React.

Ventajas y desventajas

Ventajas

La cantidad de código requerida para completar la misma función es relativamente pequeña. Svelte admite la compilación estática, no es necesario presentar el marco en sí, todo el código de tiempo de ejecución requerido está incluido en él, excepto por la introducción de este componente en sí, no sentirá la existencia del marco, y Svelte proporciona algunas plantillas simples y fáciles de usar para mantenimiento y escritura El código se ha vuelto relativamente simple. Comparado con reaccionar, el código para desarrollar la misma función es aproximadamente un 30% menos (basado en RealWorld)

Mayor rendimiento en algunos escenarios menos complejos. El código fuente compilado de Svelte es casi el mismo que el código nativo escrito a mano. Para la mayoría de las páginas puramente de presentación + una pequeña cantidad de escenas de interacción, no hay muchos lugares donde VDOM pueda desempeñar un papel. En este momento, Svelte puede aprovechar al máximo sus ventajas.

Ventajas únicas al hacer pequeños componentes distribuidos independientemente. Debido a que los componentes generados no tienen tiempo de ejecución ni dependencias adicionales, todos los componentes se pueden usar por separado y se pueden importar sin problemas en otros marcos como React, Vue o Angular.

Desventajas

Hay muchos códigos duplicados en algunos casos. Aunque la cantidad de código en una demostración simple es muy pequeña, la cantidad de código generado por tales operaciones imperativas para la misma plantilla de componente será mayor que la de la función de representación de vdom, y habrá muchos códigos duplicados en múltiples componentes. Cuantos más componentes haya en el proyecto, menor será la diferencia en la cantidad de código. Al mismo tiempo, no es realmente "sin tiempo de ejecución" como se anuncia, sino que solo importa bajo demanda de acuerdo con su código. Cuantas más funciones use, más código de "tiempo de ejecución" tendrá que incluir Svelte.

Queda por ver el rendimiento en grandes aplicaciones. En el caso de una gran cantidad de contenido dinámico y componentes anidados, la estrategia de actualización de Svelte determina que también necesita un mecanismo similar a la actualización de componentes de React para evitar actualizaciones excesivas. Por otro lado, su ventaja de rendimiento no es una diferencia cualitativa en comparación con los marcos principales actuales. El rendimiento de la mayoría de los marcos principales puede ser 1.2 ~ 1.5 veces más lento que vanilla js. El infierno basado en DOM virtual está más cerca del original, lo que demuestra En vista de la posibilidad teórica de la dirección del DOM virtual, se puede prever que el cuello de botella de rendimiento de la web en el futuro sea más el DOM en sí mismo que el marco.

Capacidad abstracta débil. La estrategia de compilación de Svelte determina que está aislada del DOM virtual (debido a su naturaleza dinámica, la función de renderizado no puede analizarse de manera confiable estáticamente como una plantilla), y no disfruta de los muchos beneficios del DOM virtual, como los componentes basados en la función de renderizado. La poderosa capacidad de abstracción, basada en vdom para pruebas.

Conclusiones

Se pudo observar que la tecnología react.js es la más usada hoy en día para el desarrollo frontend en aplicaciones web y sitios web, existen grandes empresas usando esta tecnología como lo son el propio Facebook, Airbnb, Discord, Netflix, etc.., vue.js esta hay algo cerca le sigue faltando una gran compañía que este por detrás de ella impulsando el crecimiento y uso pero es muy interesante esta tecnología para el desarrollo del frontend algunas empresas que la están usando son, Adobe, Nintendo, GitLab, etc.., y svelte.js es una herramienta que ha crecido mucho en popularidad y sigue creciendo y es una opción muy interesante para el frontend con otros conceptos como que no usa un virtual DOM para la reactividad del frontend es verdad que su uso es bajo pero esta entre las tecnologías más populares algunas de las empresas que la están usando son, Strapi, Template Monster, Brave, etc...

Para un ingeniero, no hay grandes diferencias en el framework o librería a elegir, porque se tarda poco en acostumbrarse o tener esa dinámica el uso de esa tecnología. Cada tecnología tiene sus ventajas y desventajas, lo que significa que debemos encontrar la elección adecuada durante el desarrollo del producto. Por supuesto existen otras tecnologías a la hora del desarrollo frontend que no se mencionaron como lo es Angular que tiene soporte por Google, en la presente monografía se enfocó en las tecnologías con mayos satisfacción e intereses en el mercado laboral, por otra parte, Angular ha tenido muchos cambios y muchas versiones sin ser estable a lo largo del tiempo.

Existe también nuevas tecnologías que están dando y darán de mucho que hablar como lo son Solid.js y Lit, esta última también creada y soportada por Google y tiene muchas cosas interesantes como son los componentes web nativos aprovechando muchas de las cosas nativas del navegador y del propio lenguaje JavaScript y siendo una librería con un tamaño muy pequeño a comparación con las otras que tienen que hacer demasiados procesos para conseguir un funcionamiento correcto y optimo.

Bibliografía

Greif, S. G.; Benitte, R. B. (2020). *State of JS* [encuesta]. 2020.Stateofjs. Recuperado de: https://2020.stateofjs.com/es-ES/ [2022]

Greif, S. G. (2021). *State of JS* [encuesta]. 2021.Stateofjs. Recuperado de: https://2021.stateofjs.com/es-ES [2022]

React. (s.f) *React* [documentación], reactjs. Recuperado de: https://es.reactjs.org/ [2022]

Next U. (s.f) *¿Qué es React JS? ¿Cómo funciona?* [blog], nextu. Recuperado de: https://www.nextu.com/blog/que-es-react-js-como-funciona-rc22/ [2022]

Trujillo, S. T. (2022). *Ventajas y desventajas de usar ReactJS* [preguntas y respuestas]. Peaku. Recuperado de: https://peaku.co/es/preguntas/13930-ventajas-y-desventajas-de-usar-reactjs [2022]

Vue. (s.f) *Vue* [documentación], vuejs. Recuperado de: https://es.vuejs.org/v2/guide/ [2022]

Lopez, M. L. (2019). *¿Qué es Vue.js y cómo lo usamos?* [blog], encora. Recuperado de: https://www.encora.com/es/blog/qu%C3%A9-es-vue.js-y-c%C3%B3mo-lo-usamos [2022]

Rootstack, (2021). *VueJS: Ventajas y desventajas de este framework* [blog], rootstack. Recuperado de: https://rootstack.com/es/blog/vuejs-ventajas-desventajas [2022]

Javfon1, (2019). *ReactJS vs Angular5 vs Vue.js ¿Cuál elegir?* [blog], developinginspanish. Recuperado de: http://developinginspanish.com/2019/03/08/reactjs-vs-angular5-vs-vue-js-cual-elegir/ [2022]

Duran, M. D. (2019). *Introducción a Svelte, un nuevo framework para tu frontend* [blog], midudev. Recuperado de: https://midu.dev/introducci%C3%B3n-a-svelte/ [2022]

Svelte. (s.f) *Svelte* [documentación], svelte. Recuperado de: https://svelte.dev/ [2022]

Cruz, J. C. (2019). *¿Qué es Svelte.js?* [blog], edteam. Recuperado de: https://ed.team/blog/que-es-sveltejs-1 [2022]

Programadorclic. (s.f) *Svelte-más que rápido* [blog], programmerclic. Recuperado de: https://programmerclick.com/article/3370912932/ [2022]

Nodejs. (s.f) *Nodejs* [documentación], nodejs. Recuperado de https://nodejs.org/es/about/ [2022]