The Progressive JavaScript Framework

{dev/talles}

Fácil de iniciar

Si conoces HTML, CSS y JS

Versátil

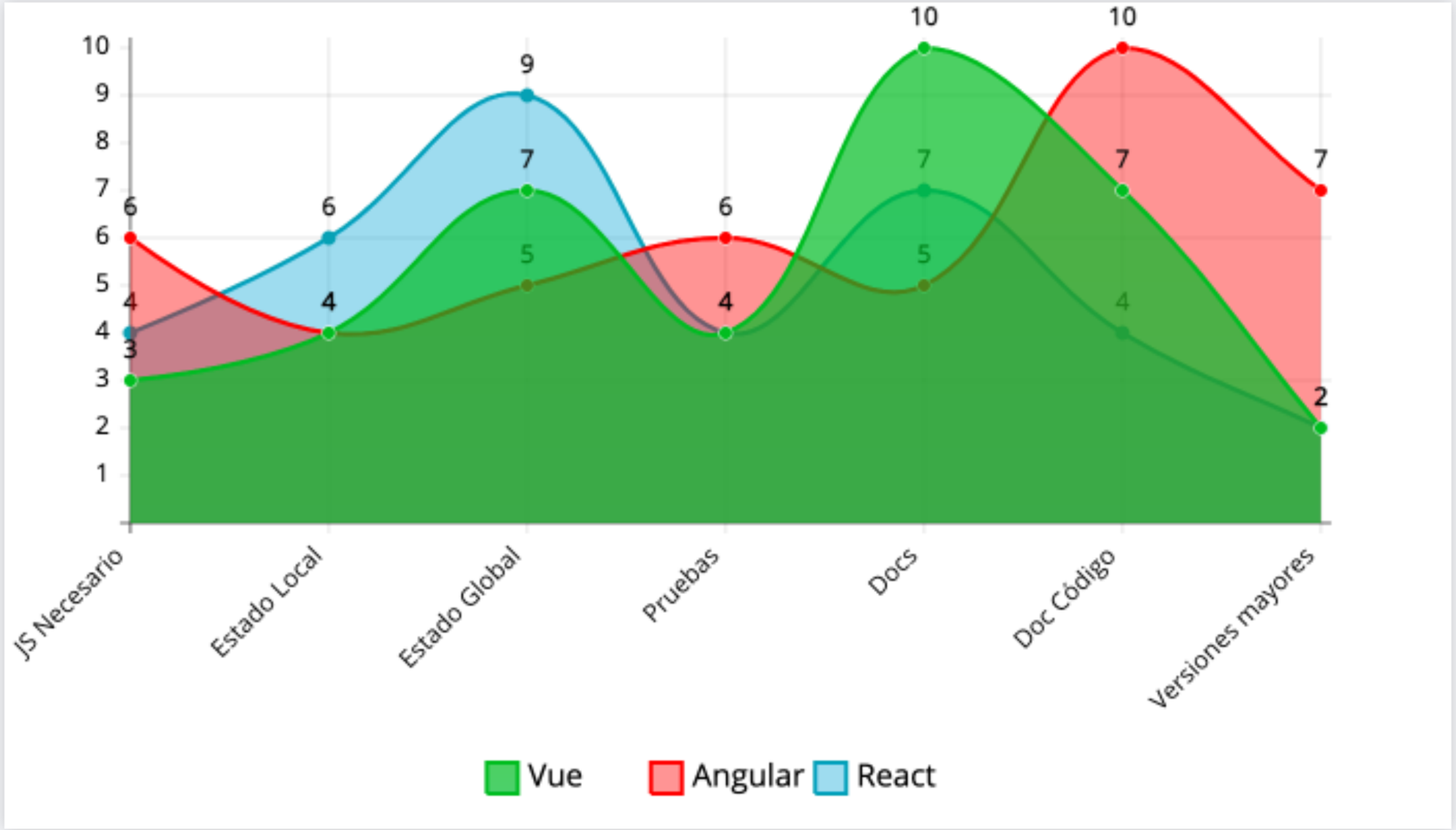
Ecosistema escalable

Rendimiento

DOM virtual súper rápido

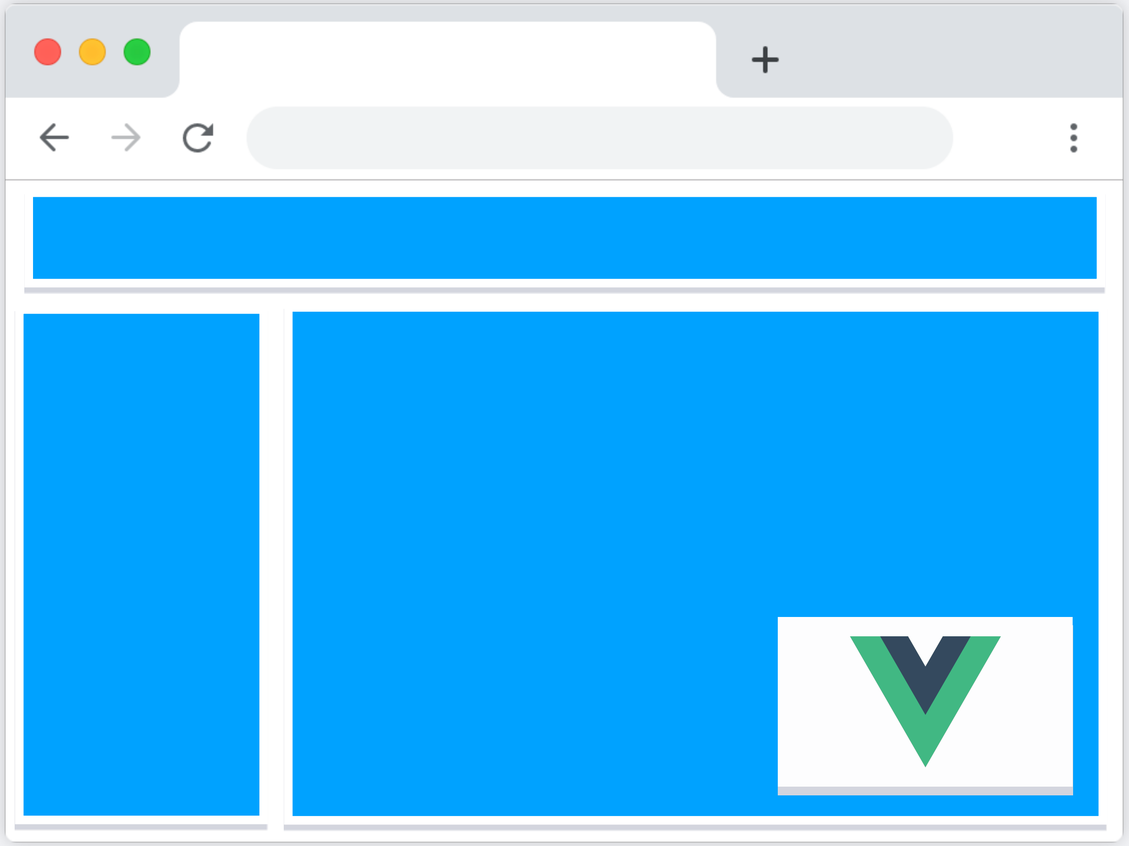
Fácil de iniciar

Si conoces HTML, CSS y JS



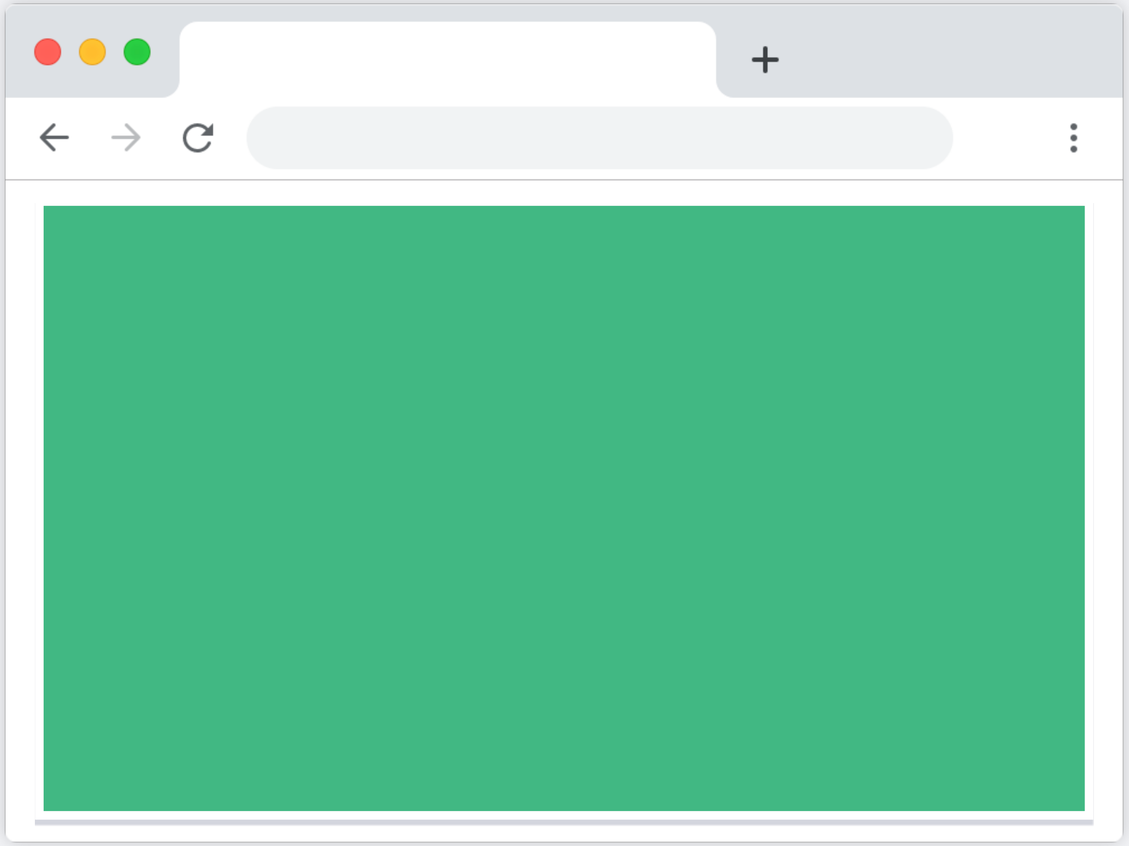
{dev/talles}

The Progressive JavaScript Framework

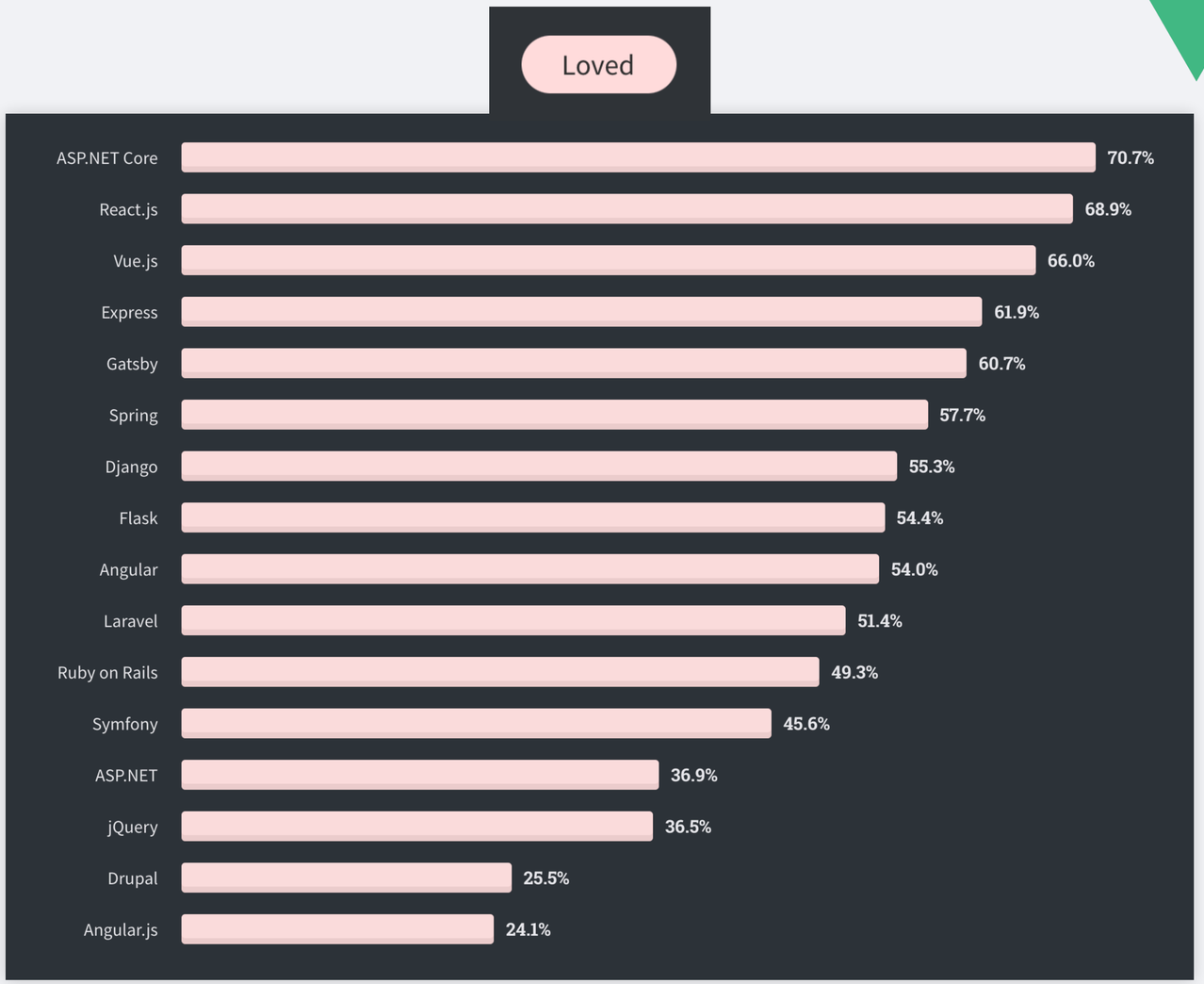


{dev/talles}

The Progressive JavaScript Framework



- Extremadamente rápido
- Fácil de entrar
- Escalable



Fuente: Stack overflow surveys, Medium

En el material adjunto

Introducción

Pruebas unitarias y de integración

¿Qué son?

Lineamientos

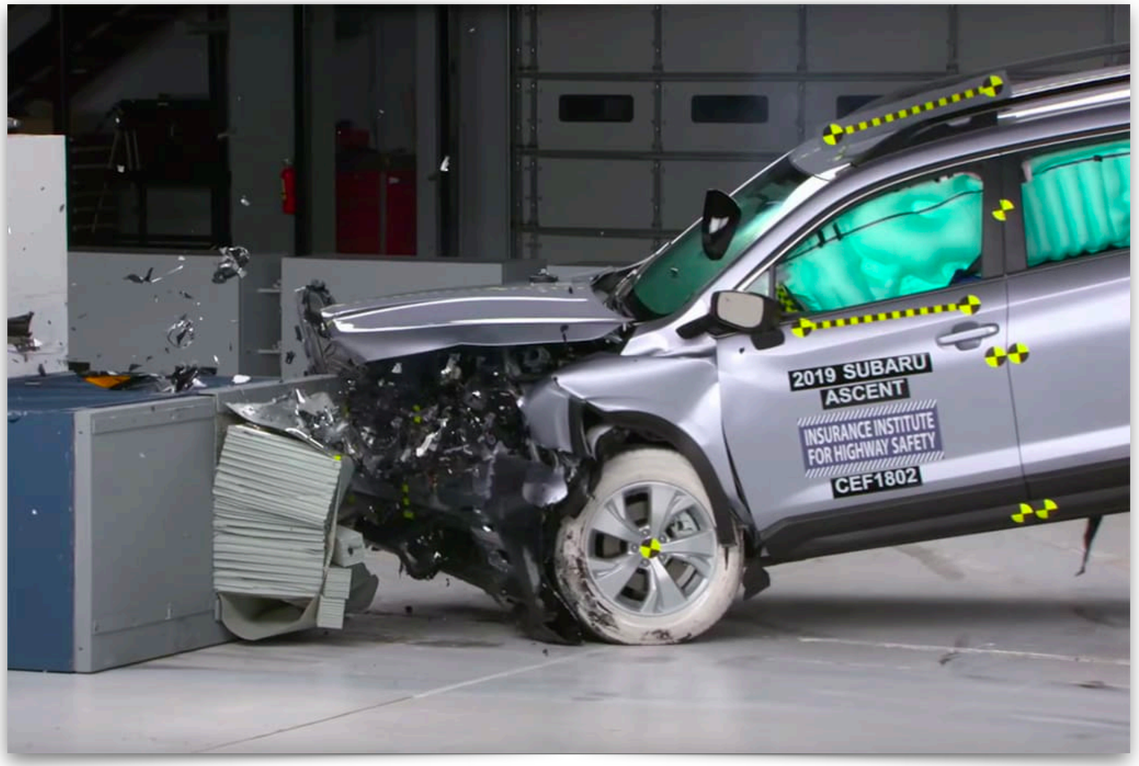
¿Para qué sirven?

Mitos

Características

{dev/talles}

No son una pérdida de tiempo

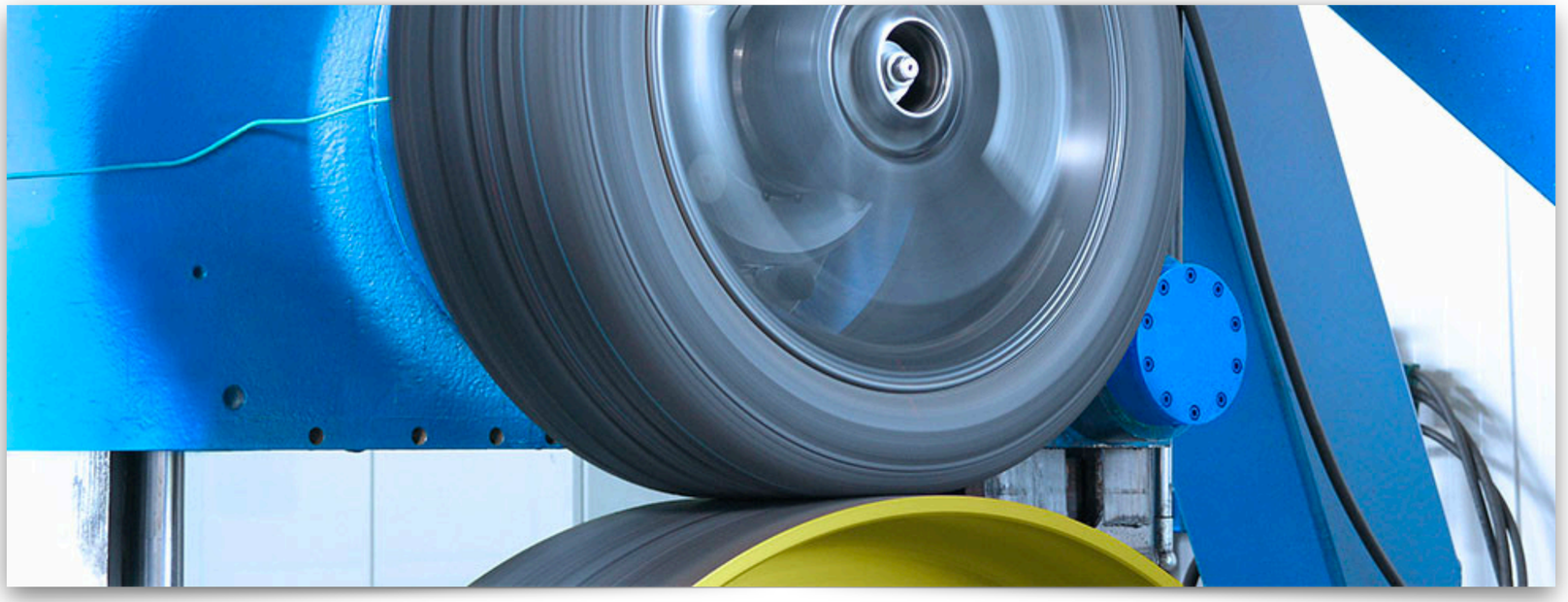


Unitarias y de Integración

Unitarias: Enfocadas en pequeñas funcionalidades.

Integración: Enfocadas en cómo reaccionan varias piezas en conjunto

Unitarias



{dev/talles}

Integración

Características de las pruebas

1. Fáciles de escribir
2. Fáciles de leer
3. Confiables
4. Rápidas
5. Principalmente unitarias

A A A

- Arrange (Arreglar)
- Act (Actuar)
- Assert (Afirmar)

A A A

Preparamos el estado inicial

- Inicializamos variables
- Importaciones necesarias

Arrange - Arreglar

Aplicamos acciones o estímulos

- Llamar métodos
- Simular clicks
- Realizar acciones sobre el paso anterior

Act - Actuar

Observar el comportamiento resultante

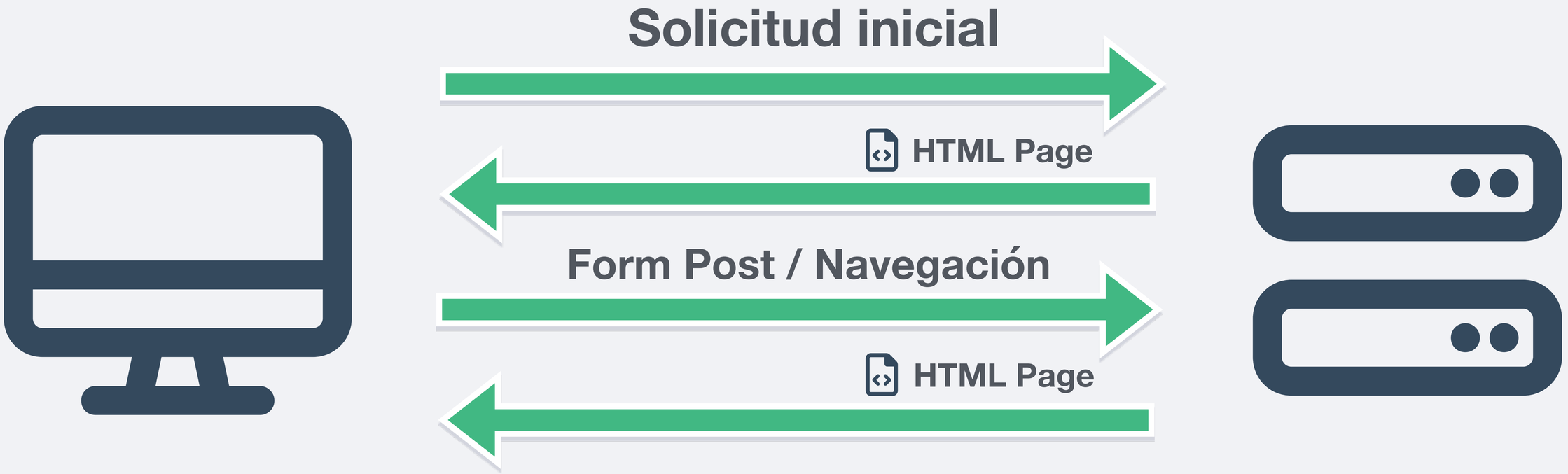
- Son los resultados esperados
- Ej: Que algo cambie, algo incremente o bien que nada suceda

Assert - Afirmar

Mitos

- Hacen que mi aplicación no tenga errores
- Las pruebas no pueden fallar
- Hacen más lenta mi aplicación
- Es una pérdida de tiempo
- Hay que probar todo

Navegación tradicional



{dev/talles}

Full Page Refresh

SPA (Single Page Application)

{dev/talles}

Ventajas

SPA

- Tras la carga inicial una experiencia fluida y rápida
- Menor estrés para el servidor
- El caché puede reducir peticiones
- Mejor experiencia de usuario
- Carga independiente de módulos cuando son necesarios

Desventajas

SPA

- Es complicado implementar SEO (search engine optimization)
- Un cambio pequeño puede requerir un build completo
- Carga inicial pesada* (depende de su implementación)

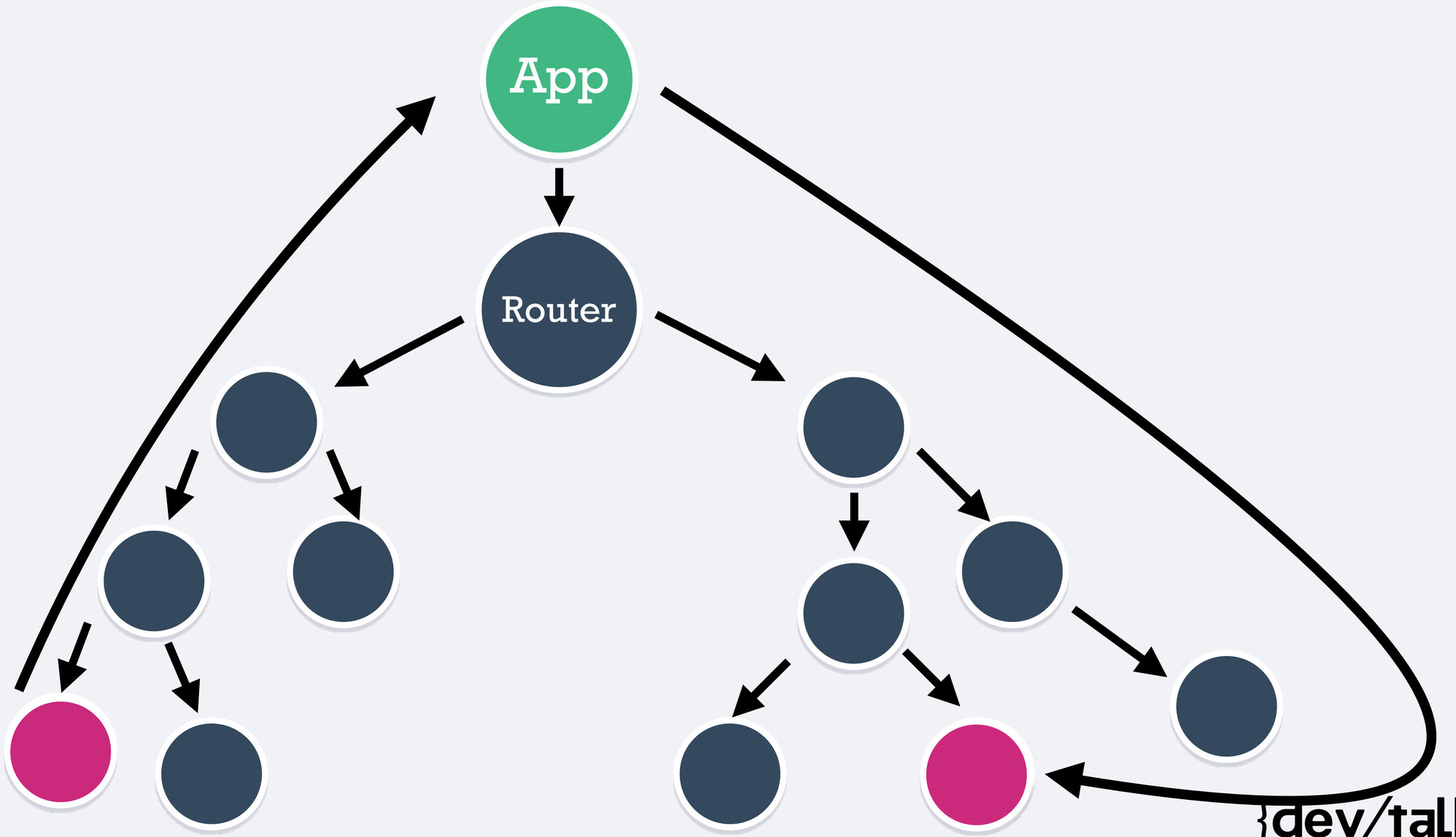
Vuex - Introducción

Vuex ya no es recomendado
Usar pinia ahora ——>

Pero los conceptos y teoría es importante

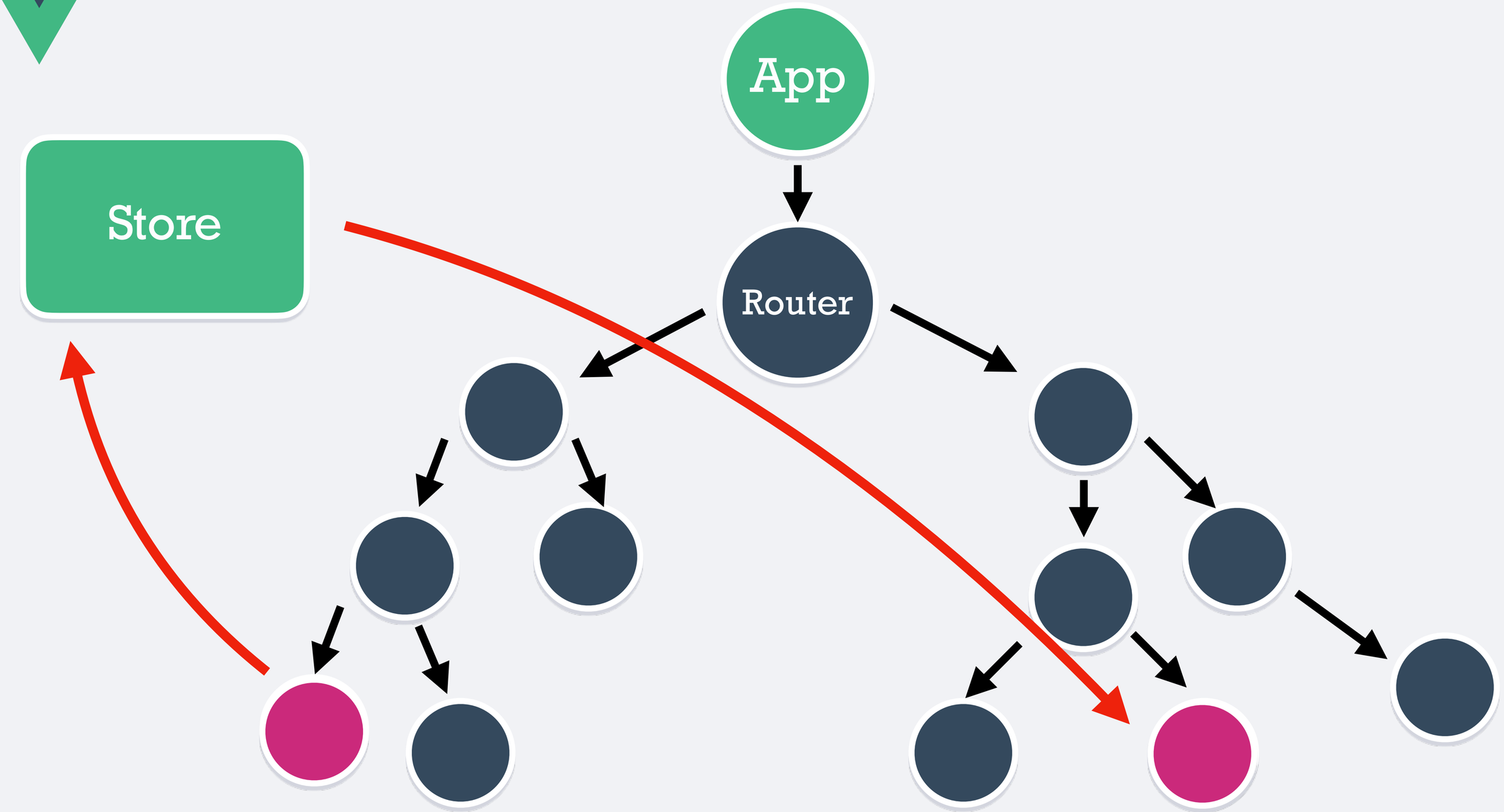
{dev/talles}

El problema que resuelve Vuex



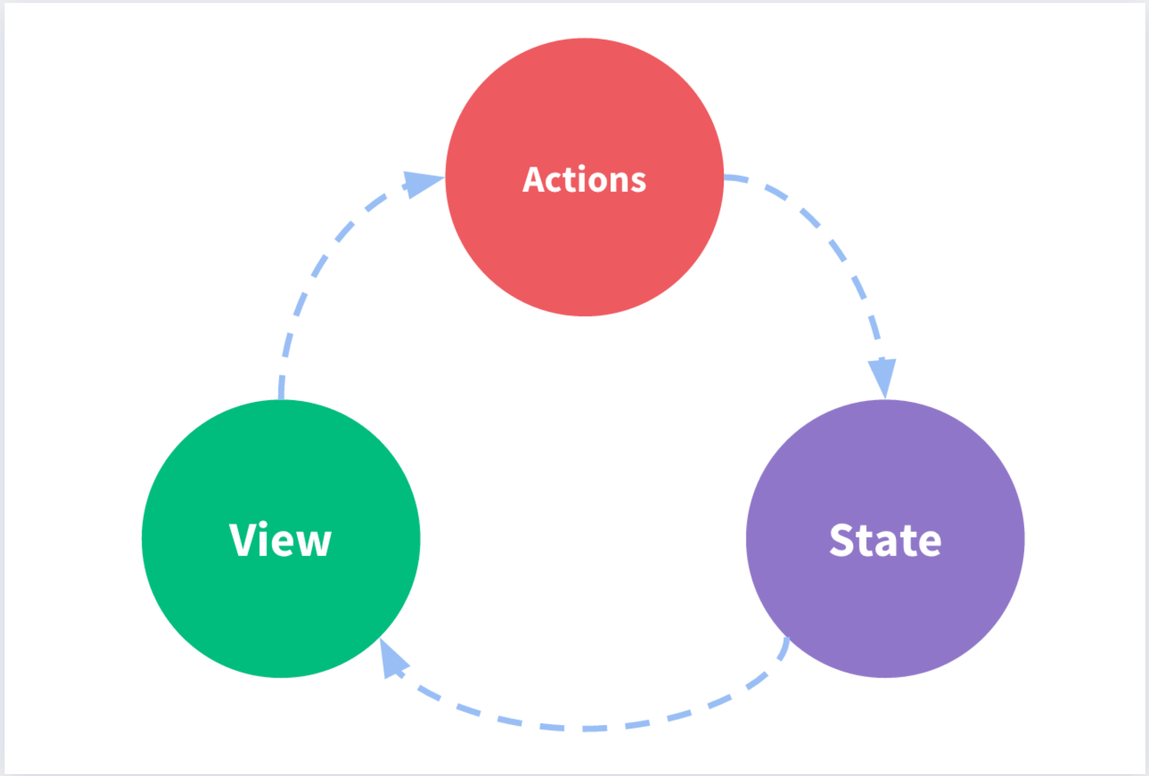
El state del Store, es reactivo

Vuex



Vuex

Fue diseñado especificamente para Vue.js



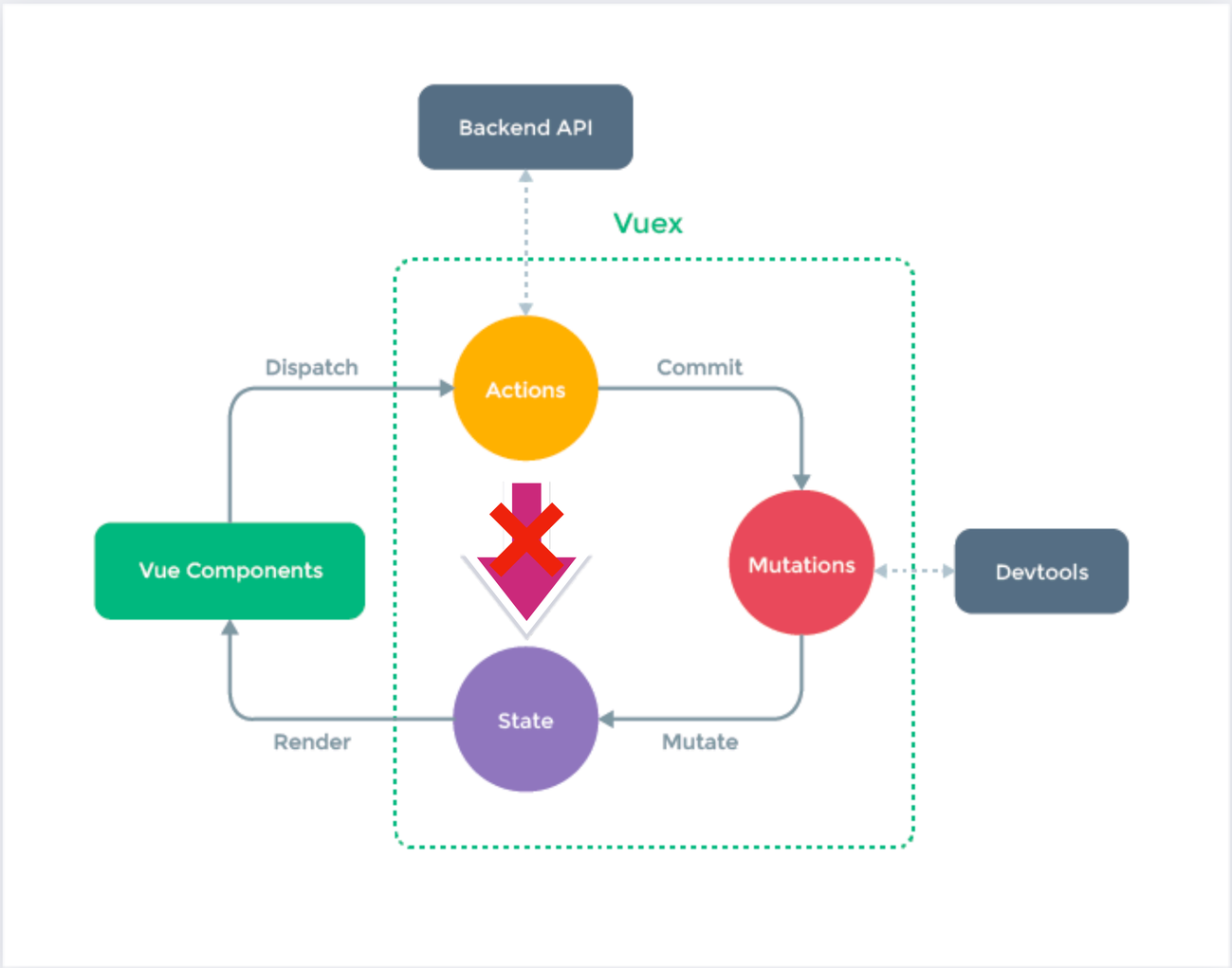
{dev/talles}

Similar a Flux, Redux y Arquitectura Elm

Fuente: next.vuex.vuejs.org

Acciones pueden ser asíncronas
Commits DEBEN de ser sincronos

Vuex



{dev/talles}

Fuente: next.vuex.vuejs.org

Conceptos generales

- State: Similar a la data(), es reactivo
- Getters: Similar a las propiedades computadas
- Mutations: Funciones que pueden modifican el State
- Actions: Funciones que pueden ser asíncronas que pueden llamar mutations
- Modules: Agrupador de state, getters, mutations, actions y otros módulos. (Muuuuuuy útil)

¿Cuándo usar Vuex?

- Vuex nos ayuda a lidiar con la gestión de estado compartida por el costo de mayor estructura y archivos.
- Es una compensación entre la productividad a corto y largo plazo.