

# Programmation Réactive

---

Principes fondamentaux et application au Web

# Plan

---

- ▶ Quoi et pourquoi la réactivité
- ▶ Les principes de la programmation réactive
- ▶ Réactivité et Vue

# Qu'est ce que la programmation réactive ?

---

Une approche visant à mieux gérer les flux

Deux types de flux

- ▶ Des événements discrets : frappe clavier
- ▶ Des évènements continus ou *comportements* : position souris

Idée : dépasser les callbacks ou le patron Observer.

# Où avez vous vu ça ?

A screenshot of the Microsoft Excel ribbon interface. The 'Home' tab is selected. In the 'Font' section, the font is set to 'Calibri (Body)' and the size is '12'. The 'Font Color' dropdown shows yellow and red. Below the font section, there are buttons for bold ('B'), italic ('I'), underline ('U'), and other styling options. The formula bar shows the formula '=A1\*2'. In the worksheet area, row 1 and column A are selected. Cell A1 contains the formula '=A1\*2'. The formula is also displayed in cell B1. The rest of the grid is empty.

<http://www.hanselsolutions.com/blog/surf-talk/shiny-surf.html#/9>

# Où avez vous vu ça ?

---

The screenshot shows a Microsoft Excel interface. The ribbon menu is visible at the top, with 'Home' selected. The formula bar below shows 'B2' is selected. A table is present with data in cells B1, B2, and B3. Cell B2 contains the value '10'. The table has columns labeled A through F and rows labeled 1 through 3. The 'Font' section of the ribbon shows 'Calibri (Body)' and '12' selected. The 'Font Color' dropdown is open, showing red as the current color. The 'Font Size' dropdown shows '12'.

	A	B	C	D	E	F
1		5	10			
2						
3						

<http://www.hanselsolutions.com/blog/surf-talk/shiny-surf.html#/9>

# Où avez vous vu ça ?

---

The screenshot shows the Microsoft Word ribbon with the "Home" tab selected. Below the ribbon, the "Font" group is expanded, showing "Calibri (Body)" as the font, "12" as the size, and various bold, italic, and underline options. The table below has 3 rows and 6 columns labeled A through F. Row 1 contains values 1, 6, and 12 in columns A, B, and C respectively. Row 2 is empty. Row 3 is also empty. The cell containing the value 12 is highlighted with a blue border.

	A	B	C	D	E	F
1	6	12				
2						
3						

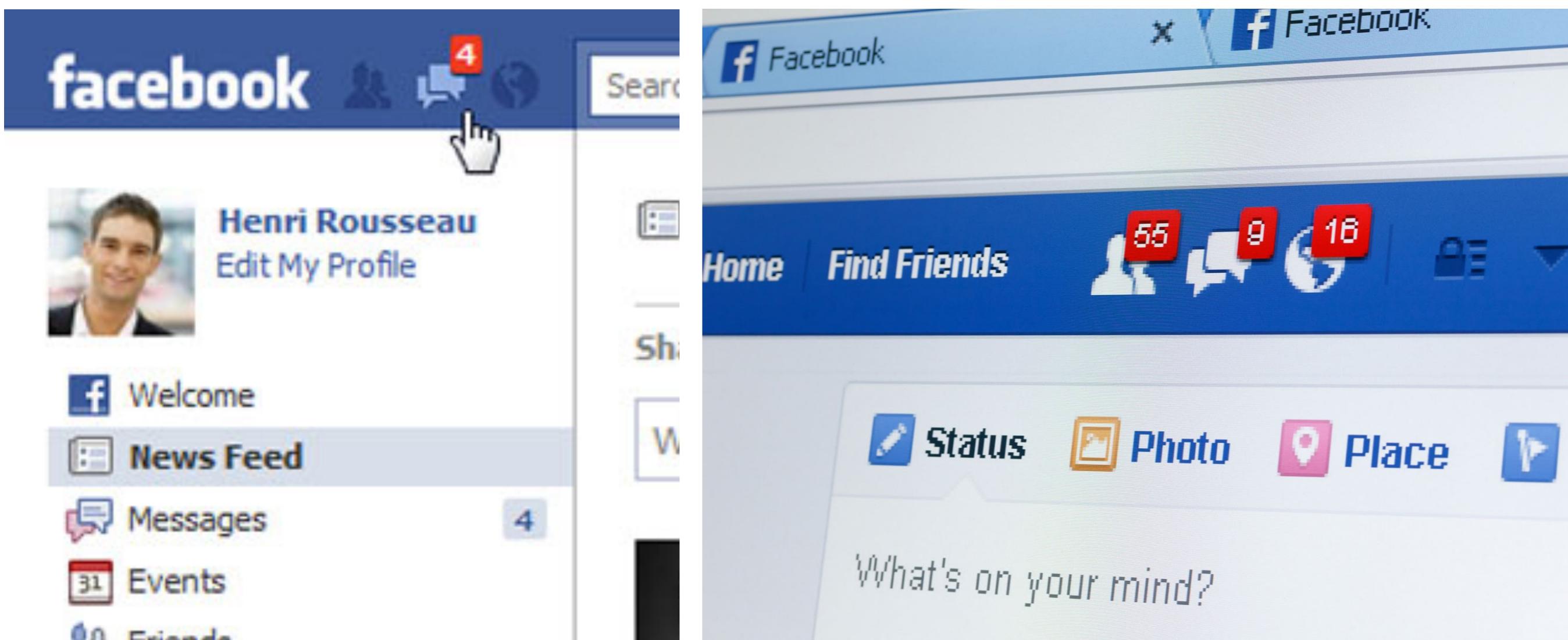
<http://www.hanselsolutions.com/blog/surf-talk/shiny-surf.html#/9>

# Pourquoi la prog. réactive sur le Web ?

The screenshot shows a ticket management interface with the following details:

- Left Sidebar:** Includes icons for IMs (12), Labels, Flags, Inbox (12), My Tickets (5), Tickets I Follow (34), My Teams (9), Unassigned Tickets (132), Tickets by User (5), All Tickets (132), Secondary Tickets (Mine On Hold 26, All On Hold 324), and a checkmark icon.
- Header:** Features a search bar ("Search Tickets or everything"), advanced filtering options, and links for Chat (12) and Calls.
- Table View:** Displays a list of tickets with columns for ID, Subject, Status, Assignee, Department, and Flag. The table includes rows for:
  - #34528: Count records per hour within a time span (Status: 1, BitDefender, CEO, Last reply: 25m ago)
  - #34528: Problems using windows 7 built-in virtual wifi hotspot (Status: 2, SLA: 25m 4h33m)
  - #34528: JIRA REST API to get work log - "You do not have the permission to see the specified issue" (Status: 3, SLA Failed)
  - #34528: Count records per hour within a time span (Status: 4, CP, Last reply: 25m ago)
  - #34528: How could manage the communication between a WCF thread and another thread? (BUG, UPDATE, Status: 5, CP, Last reply: 25m ago)
  - #34528: Customize auto-configured Spring Boot Bean (Status: 4, CP, Last reply: 25m ago)
  - #34528: How could manage the communication between a WCF thread and another thread? (Status: 5, CP, Last reply: 25m ago)
- Right Sidebar:** Shows ticket properties for the selected ticket (#34528), including BitBucket, Status: Awaiting Agent, Language: English, Product: DeskPRO Cloud, Brand: DeskPro Cloud, Participants, and Billing information.

# Les origines de React



# Pourquoi la programmation réactive ?

---

- ▶ Gestion d'évènements et de l'asynchrone
- ▶ Faible latence (contraintes sur les temps de réponse)
- ▶ Flux de données importants (et rapides).
- ▶ Tolérance aux fautes

# Exemples

---

À vous

# Les bibliothèques Javascript

Guide API Examples Blog Community ▾



## Vue.js

Reactive Components for Modern Web Interfaces

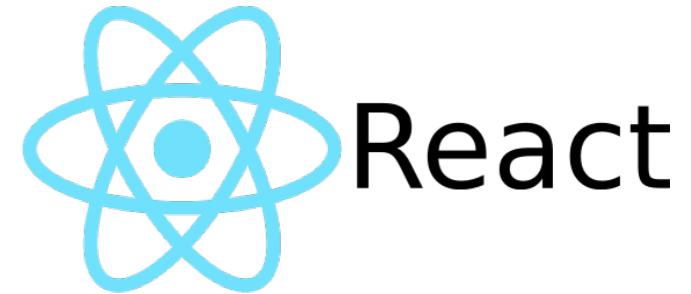
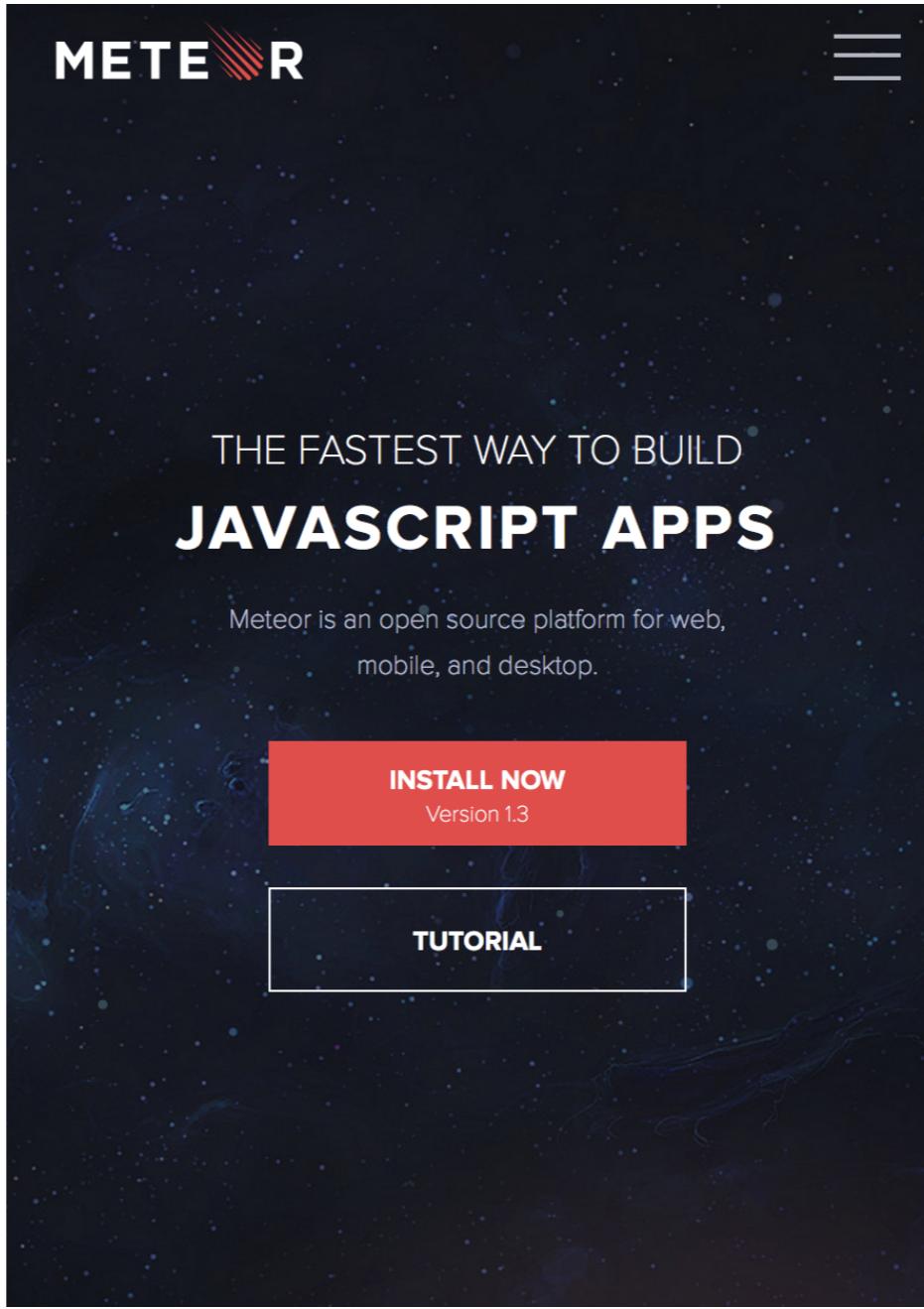
Install v1.0.24

[Follow @vuejs](#)

[Star 18,631](#)

[Support Vue.js](#)

中文 | 日本語 | Italiano



# Plan

---

- ▶ Quoi et pourquoi la réactivité
- ▶ **Les principes de la programmation réactive**
- ▶ Réactivité et Vue

# Les principes de base

---

- ▶ Responsive,
- ▶ Résilient,
- ▶ Élastique,
- ▶ Orienté message

# Responsive

---

- ▶ Réponse en temps voulu, si possible
- ▶ Temps de réponses rapides et fiables (limites hautes)

# Résilient

---

- ▶ Résiste à l'échec
- ▶ Principes :  
RéPLICATION, conteneurs, isolement, délégation
- ▶ On fait en sorte qu'un échec n'impacte qu'un seul composant

# Élastique

---

Le système reste réactif en cas de variation de la charge de travail.

- ▶ Pas de point central
- ▶ Pas de goulot
- ▶ Distribution des entrées entre composants

# Orienté message

---

- ▶ Passage de messages asynchrones
  - > Couplage faible, isolation
- ▶ Pas de blocage, les composants consomment les ressources quand ils peuvent

# Un concept important : l'immuabilité

---

## Objet immuable (Immutable object)

- ▶ Objet dont l'état ne peut pas être modifié après sa création
- ▶ Opposé d'objet variable

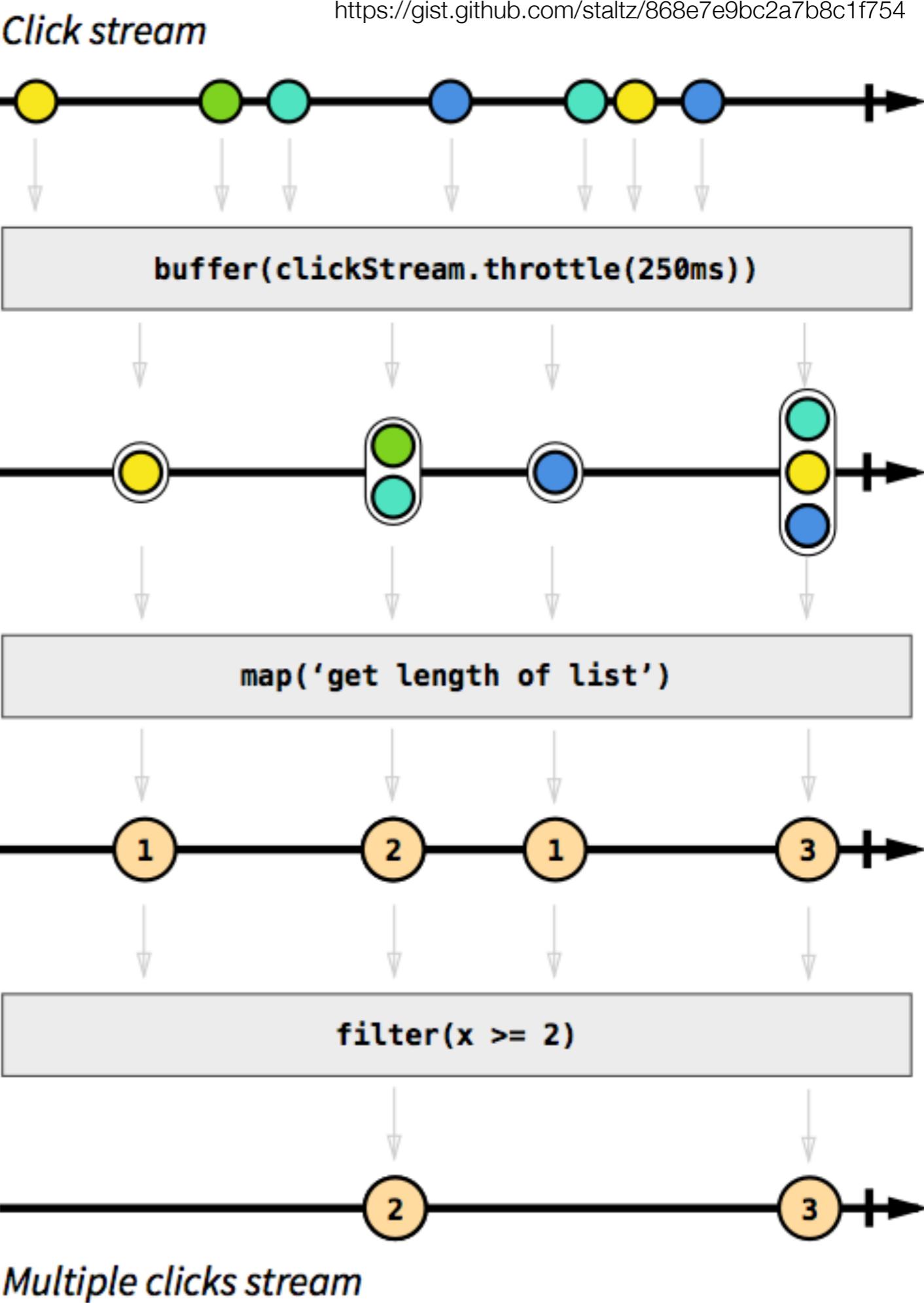
Facilite la prog. purement fonctionnelle (pratique pour plein de choses, évite les effets de bords, facilite le undo)

Une seule source de “vérité”

Facilite le caching

Mais ce n'est pas forcément assez : <https://codewords.recurse.com/issues/six/immutability-is-not-enough>

# Un exemple de transformation

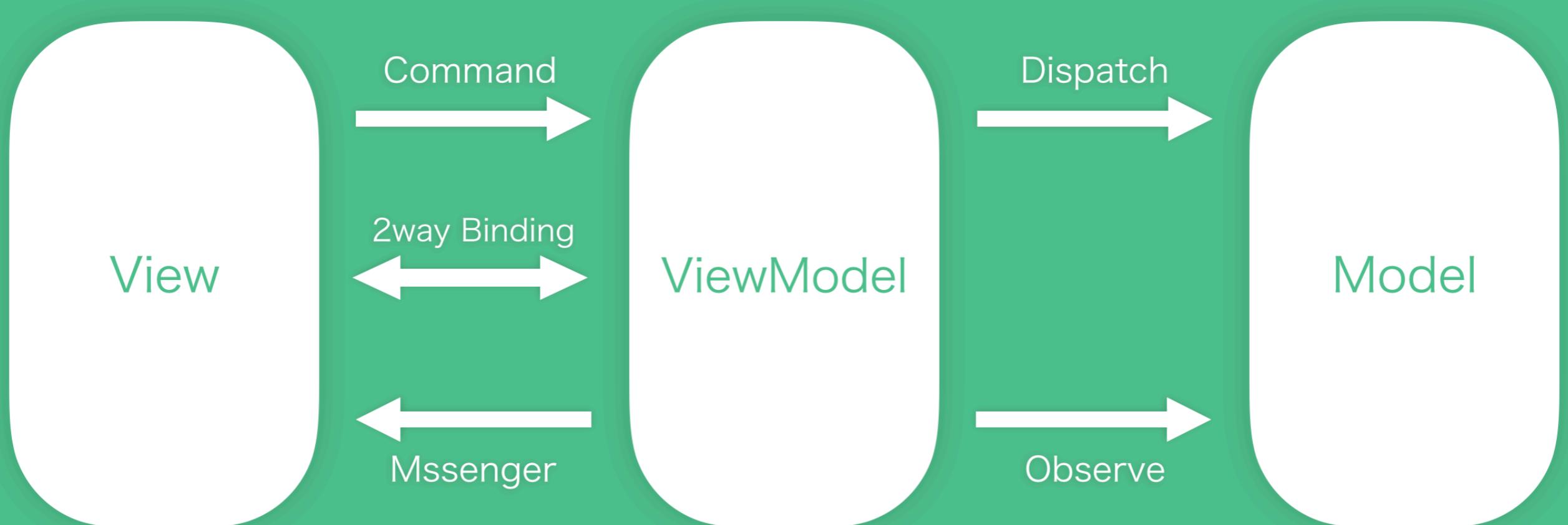


# Plan

---

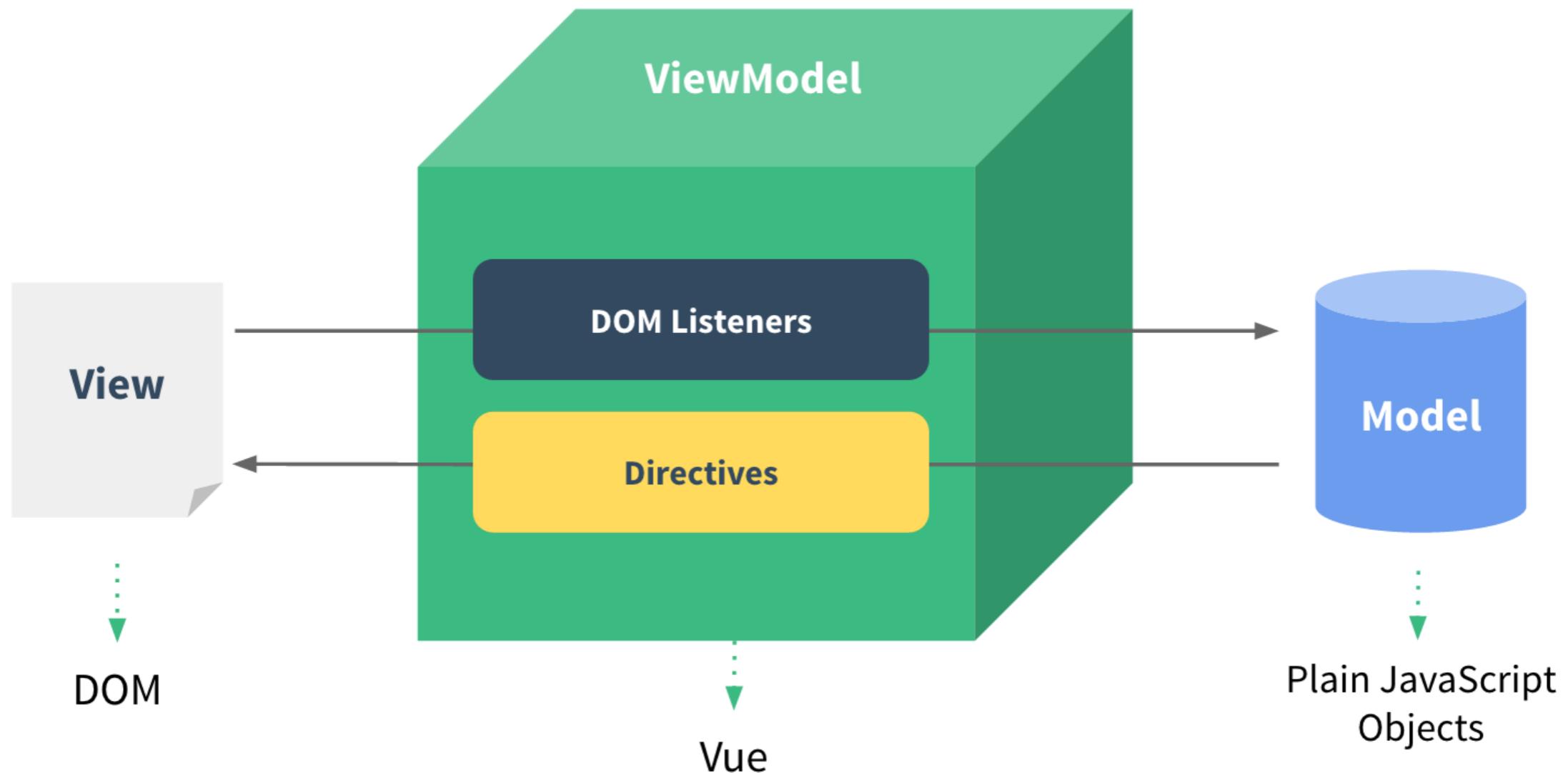
- ▶ Quoi et pourquoi la réactivité
- ▶ Les principes de la programmation réactive
- ▶ **Réactivité et Vue**

# Vous avez aimé MVC voilà MVVM

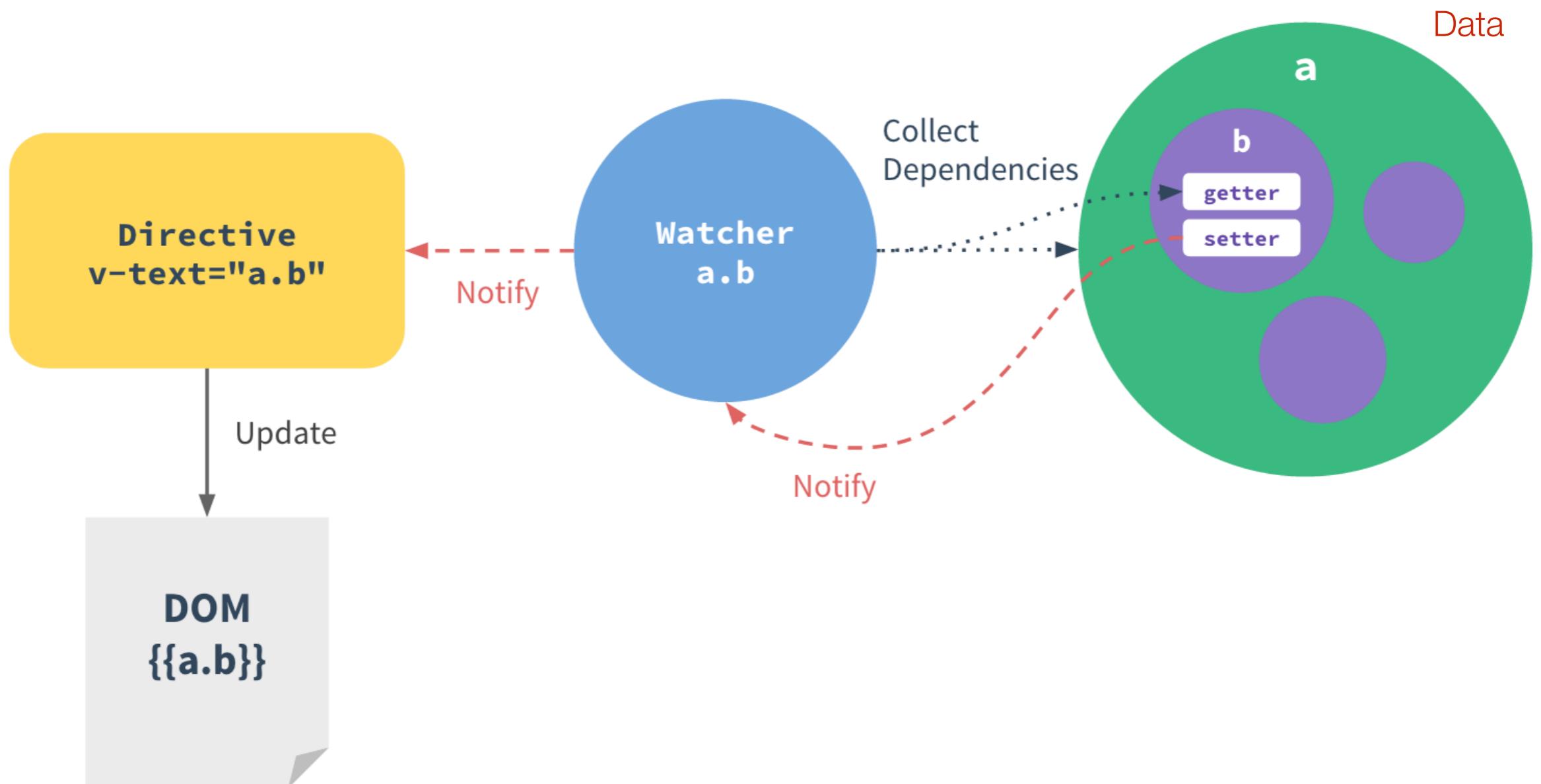


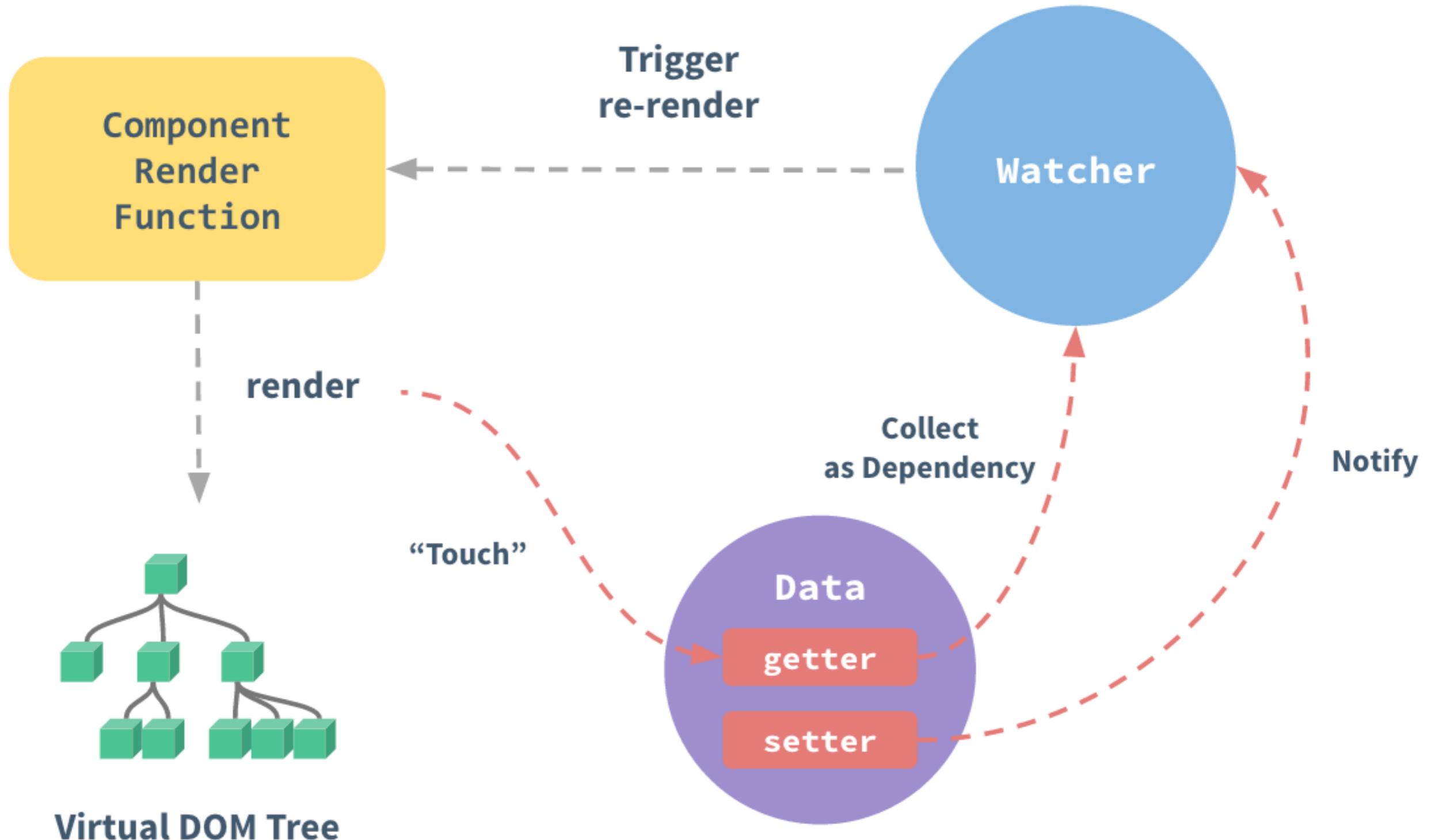
# MVVM avec Vue

---



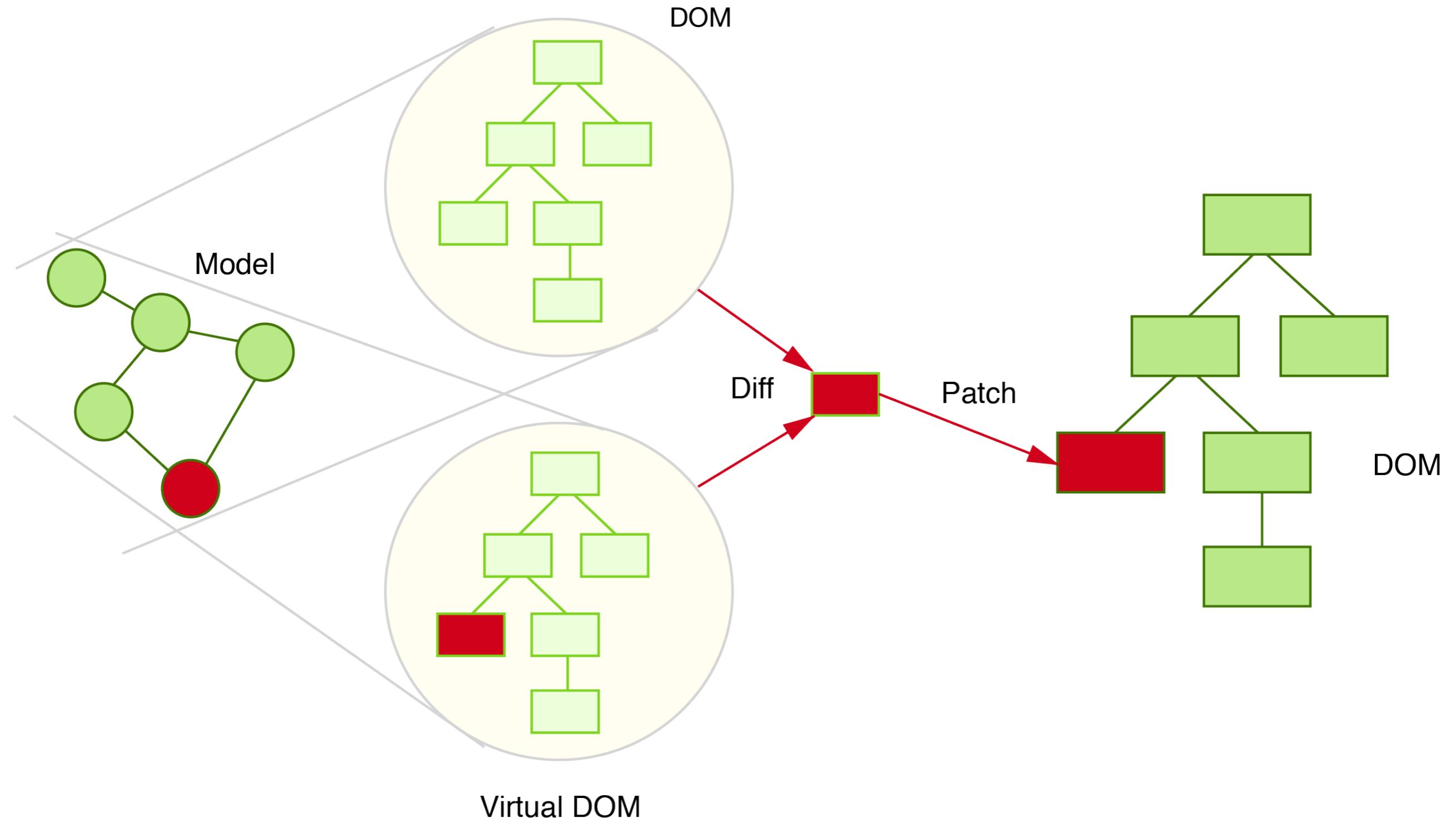
# En pratique avec Vue





# Un DOM Virtuel

<https://teropa.info/blog/2015/03/02/change-and-its-detection-in-javascript-frameworks.html>



# Flux de données avec Vue

---

Deux approches au data-binding (lien entre données et vue):

- ▶ Two way data-binding  
Fait dans le TP précédent avec v-model
- ▶ One way data binding  
Le flux de données est uni-directionnel, les enfants ne modifient pas les données qui sont contrôlées par leur parents.  
Mais les enfants peuvent demander au parent de changer.

# Deux façons de gérer les données

---

- ▶ Données qui changent (mutable) :  
on utilise un état (data)
- ▶ Données qui ne changent pas (immutable) :  
on utilise des propriétés (props)
- ▶ On essaie de minimiser les données qui changent  
quitte à refaire des calculs

# Vuex : gestion avancée des états

# Vuex : gestion d'états

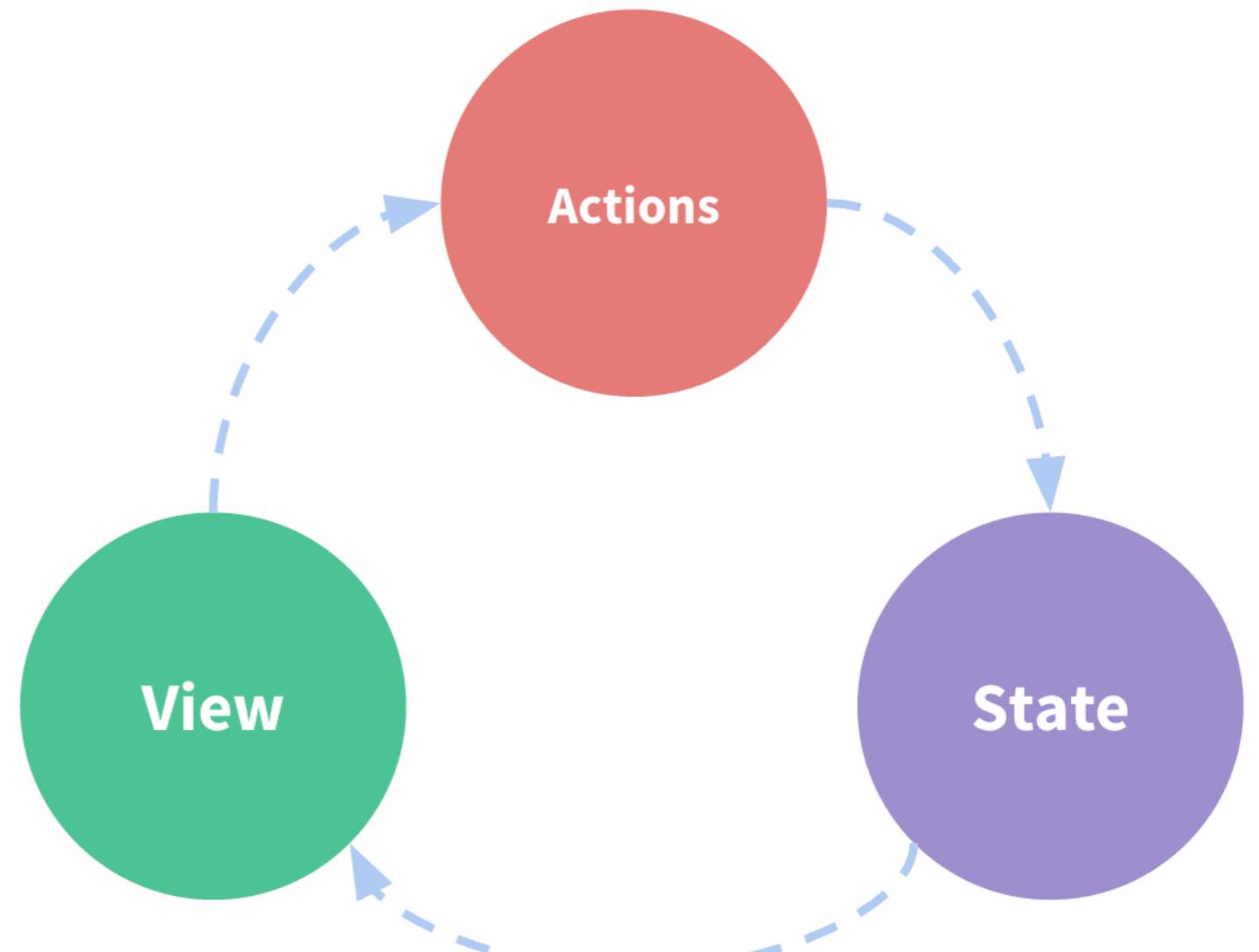
<https://vuex.vuejs.org/en/intro.html>

*One-way data flow*

**State** (état), source de “vérité” pour l’application

**View** (vue), représentation de l’état (mapping déclaratif)

**Actions**, changements de l’état en réaction à des entrées de l’utilisateur au niveau de la vue (ou d’autres inputs)



# Mais...

---

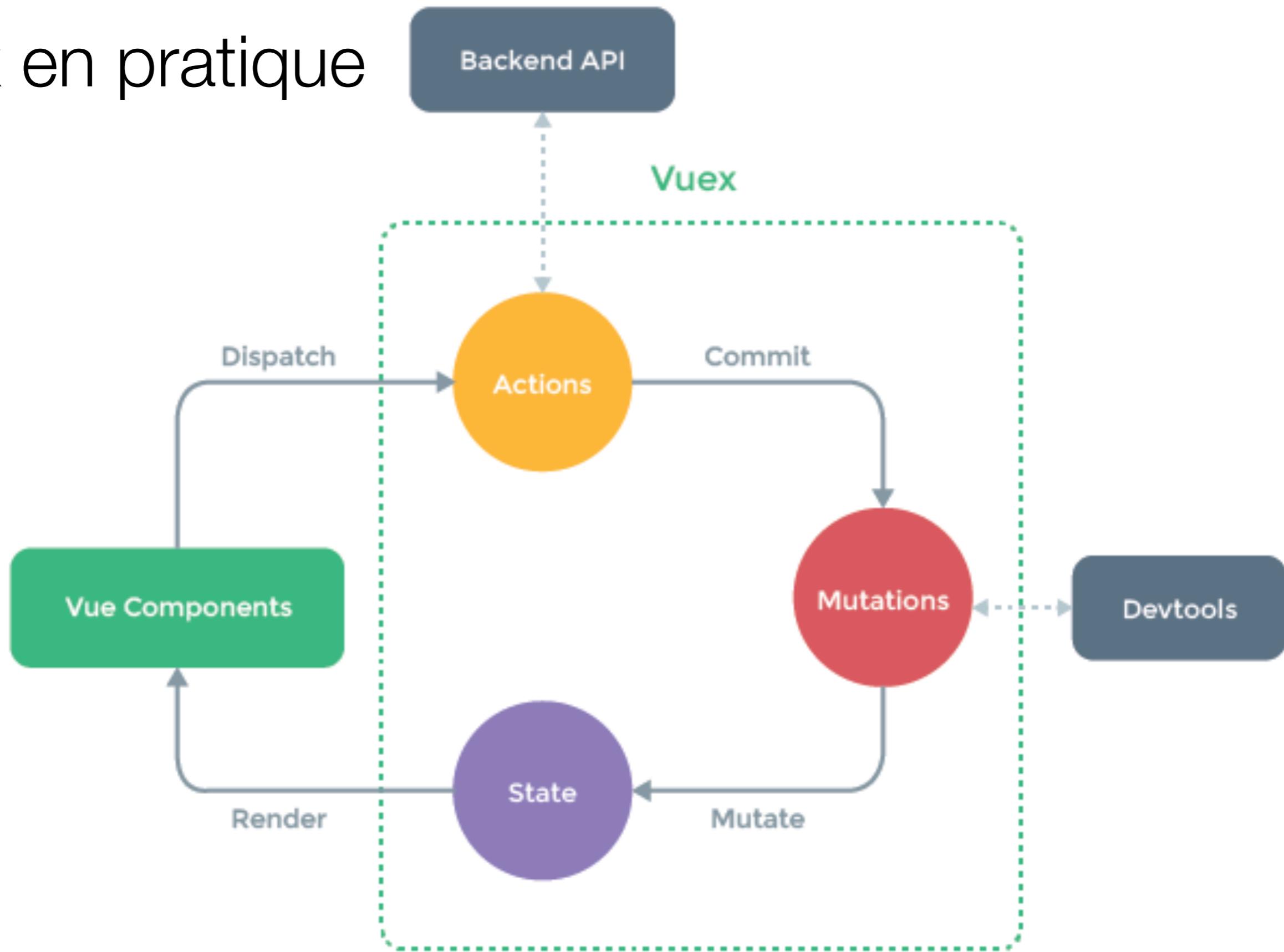
Problèmes quand :

- ▶ Plusieurs vues dépendent du même bout d'état
- ▶ Plusieurs actions de différentes vues vont changer (mutate) un même bout d'état

Solution (vuex):

- ▶ l'état devient un gros singleton partagé par tous les composants
- ▶ même principe du côté de React avec l'architecture Flux implémentée dans Redux, ou du côté de l'architecture de Elm.

# Vuex en pratique



# Qui utilise Vue ?

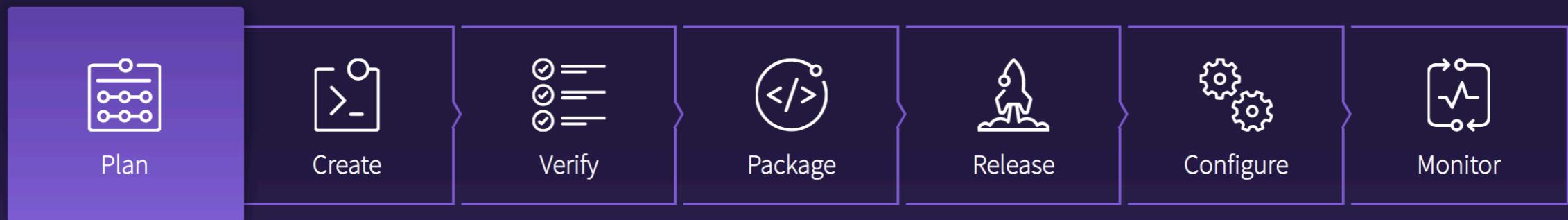
<https://about.gitlab.com/2016/10/20/why-we-chose-vue/>



## From planning to monitoring

Create value faster with a single application for the whole software development and operations lifecycle.

[Try GitLab for Free](#)

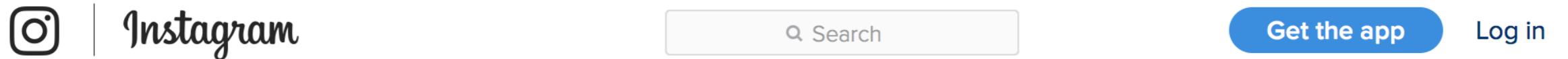


### Plan: Get your best ideas into development.

Whether you use Waterfall, Agile, or Conversational Development, GitLab streamlines your collaborative workflows. Visualize, prioritize, coordinate, and track your progress your way with GitLab's flexible

A screenshot of the GitLab interface, specifically the Issue Boards feature. The top navigation bar shows 'Projects', 'Groups', 'Activity', 'Milestones', and 'Snippets'. Below the navigation is a breadcrumb trail: 'GitLab.org &gt; GitLab Community Edition &gt; Issue Boards'. The main area displays three columns of issues: 'Development' (Backlog, UX, frontend), 'UX' (bug, merge requests), and 'frontend' (bug, frontend, user profile). Each issue card includes a title, a link, and labels like 'bug', 'merge requests', 'UX', 'frontend', 'user profile', and 'external services'.

# Un exemple d'application React + Backbone



jr ✅ [Follow](#)

JR Artist until i find a real job :) Snapchat : JRARTISTE [presse.louvre.fr/jr-au-louvre/](http://presse.louvre.fr/jr-au-louvre/)

3,160 posts    827k followers    774 following

