

Vue js optimisation

Tree shaking

It basic ally deals with the removing of unnecessary code .Try to shi as much less size possible

Tree Shaking vs Code Spilliting

Tree shaking means only bundling that code that is required example if you dont need xyz lib or component thats not shipped and code spilliting means that the code decide to be shipped is break down into smaller and smaller pieces called **chunks** so that browser one big file to load instead small parts are loaded **on demand parallel** if its properly used page can be quickly downloaded and the additional chunks (broken down with the help of chunk) can be loaded later as per the requirement thus dealing with the size effectively .

For vue js components you can use Async Component

Bolated Bundles

If using a build step, prefer dependencies that offer ES module formats and are tree-shaking friendly. For example, prefer **lodash-es** over **lodash**.

Using SHALLOW ref to only track .value and not nested property

Virtualize Large Lists

Reduce Reactivity Overhead

So since the vue tracks the nested property of object and emaulates the behaviour of the reactivity and you mutate a nested object it re-render and tracks which seems great but can be a great performance issue as it creates too many overheads and because every property access trigger **proxy trap** and **dependency tracking** . To come out of this use **shallowRef** as it comes out or removes the nested property tracking and you will need to create new copy

```
const shallowArray = shallowRef(  
  [  
    //big list of deep object  
  ]  
);  
  
shallowArray.value.push() // will not trigger render  
  
shallowArray.value = [  
  ...shallowArray.value,  
  {  
    // it will work  
  }  
]
```

Avoid Unnecessary Component Abstractions

<https://vuejs.org/guide/best-practices/performance.html#avoid-unnecessary-component-abstractions>

Reactivity in Depth

In cases we want to check the detail which reactive value is being changes we can use **onRenderTracked**, **onRenderTriggered** **onRenderTracked**, **onRenderTriggered** and for computed and watch/watchEffect we use **onTrack** and **onTrigger Method**

Vue Rendering process

'<https://vuejs.org/guide/extras/rendering-mechanism.html>'

The vue uses **Virtual DOM** to do it and it a **concept** not qa tech pioneered by the **React** to keep the dom tree copy in memory and then locating where the changes to be done

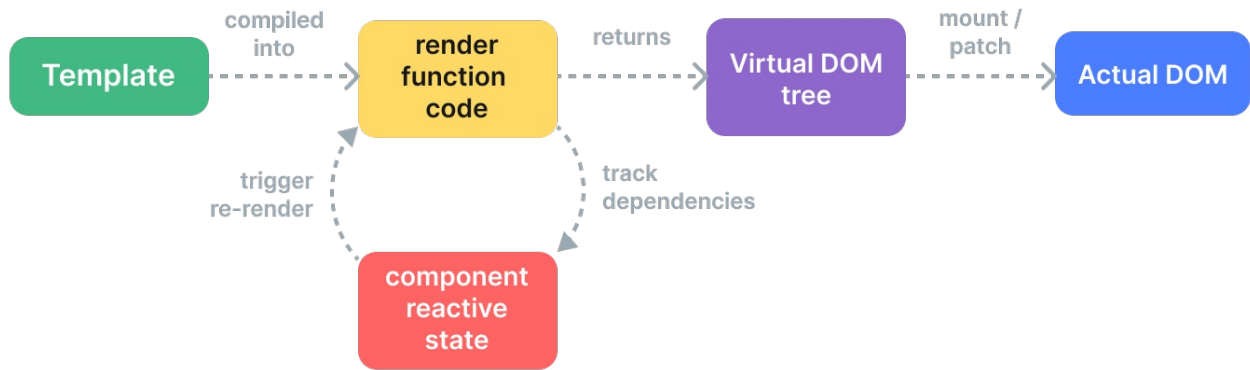
The templates are compiled to the render function whic return dom tree

vnode is a virtual node for a respective **DOM NOde**

At runtime **Renderer** it will tranverse and accordindly build the **DOM NODES** were created

After the nodes are buildled than the **Renderer** might find the difference between old and new dom tree as it will have to find out what part has been chnaged due to **Side Effects** like (Network call , callBack timers , update of state) and render accordindly according the place of change its called **patch or reconciliation**.

At high level this happens :



- Compile: Vue templates are compiled into render function that return the vnode tree .It can be done compile time , build time , run time or ahead of time as well.
- Mount: These render method is invoked and actual dom Node is created.Here the reactivity dependency also tracked
- Patch: Whenever side effects happens or reactivity dependency triggers the render function is re run and new virtual DOM is created and compared with the old one

Template vs Render Function

You can skip the template compilation and directly access render function api and they are more flexible in nature for high dynamic sort of work and more dynamic logic can be written

So the virtual DOM implementation part happens only at the run time as the virtual dom can't be predicated earlier and every time some changes happen a whole new dom tree is created and then the **recoil**(diff) algo is used but the extra memory space is used this is the most criticized or drawback of this **Virtual dom** . But vue js optimized it

Static Hoisting

The nodes that contain static content are hoisted i.e they are nodes are created outside the render function so that each time they run they refer to same node and if they are used somewhere else they are **clone**

Patch Flag

For dynamic attribute the patch flag is used so to check whether a rework is required or not

Tree flattening

The vnode will have lots of children and each render ist will have to go through it but thanks to **createElementBlock** function and patch flag only those nodes who are using dynamic value / attribute and are returned as **flattened array** and the exact are vnodes to be target

Dom Updating nexttick

As soon as something is updated immediately all things are not pushed to the Dom node in synchronously manner Vue run a update cycle and all the changes are buffered and updated in that update cycle .To detect the cycle ` use **nextTick**

ref vs reactive

Shallow Reactivity

Shallow ref

Automatically unwrapping ref

Downside with reactivity

Computed property

It use for caching

- To update something use Writable computation property.
- Avoid Mutation in the computed property .
- Avoid use async or network calls inside the the computed property these are to be done under the **watchers**

Teleport

Many a time by logical a component is a child of the parent component by in the term or point of view it will have a different node like for example of a **modal** you use **<Teleport>** component you provide the **css selector DOM Nodes** where you component needs to be injected visually outside the normal Dom