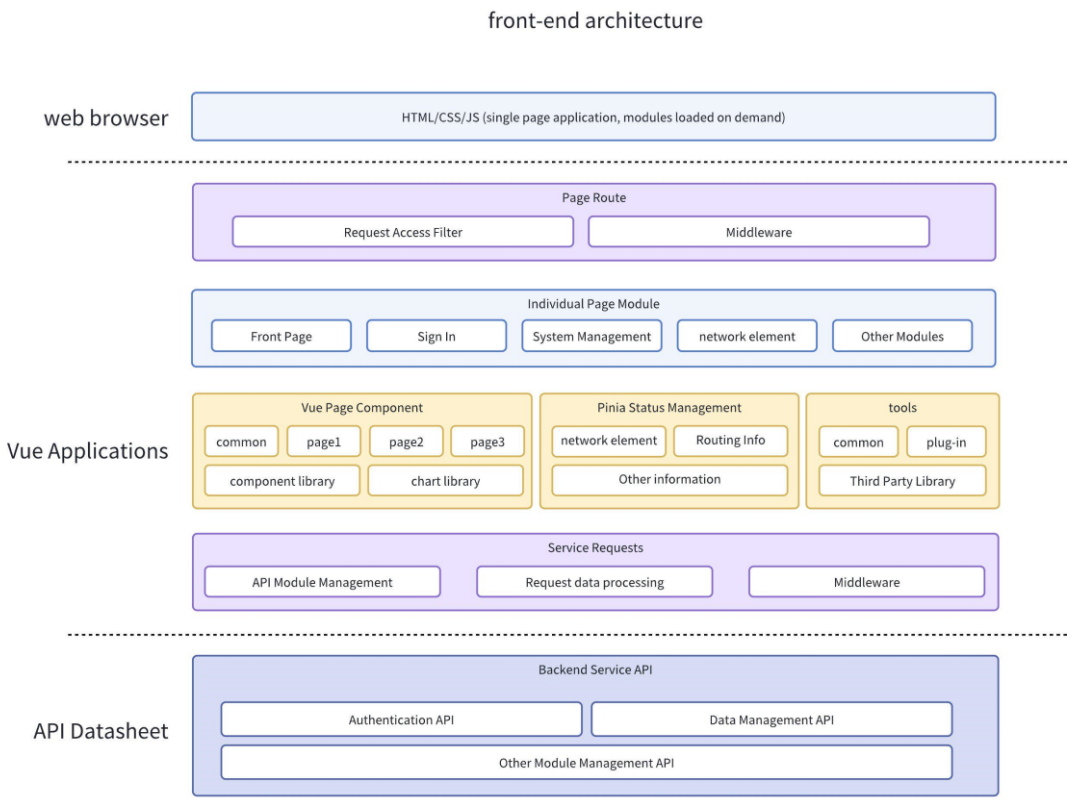
Front-end architecture design



Front-end technology selection

Vue3 and React have many similarities in terms of functional features, such as virtual DOM and component-based features. However, Vue3 has some advantages over React in terms of performance, such as faster rendering speed, smaller code size, better typescript support, etc. In terms of benchmark scores, Vue performs much better than React and Angular in js-framework-benchmark. In this benchmark test, it is also on par with some of the fastest non-virtual DOM frameworks in production environments.

Compared to the familiarity within the development team, lower entry barriers, and excellent documentation, it can significantly reduce the onboarding and training costs for novice developers. The trend of the surrounding community ecology continues to grow, so choosing Vue3 is more advantageous. The final decision on overall architecture uses a technology stack.

1. A front-end/design solution for the middle and back-end based on Vue 3.x.
2. Adopt Pinia as a state management pattern specifically designed for Vue.js application development.
3. Using Vue Router routing manager, we implement page routing control, local refresh, and on-demand loading, build a Single Page Application, and achieve front-end and back-end separation.
4. Native support for promise-based HTTP libraries in Fetch browser.
5. TypeScript provides strong typing support for coding standards, which is more suitable for developing large and medium-sized projects.
6. Using Ant Design of Vue, a web component library from Ant Financial's enterprise-level backend product visual style.
7. Responsive layout, designed for different screen sizes, suitable for computers, tablets, and mobile phones.
8. Implement an industry-wide internationalization solution that supports Simplified Chinese and English.
9. Using Vite as the standard tool for Vue.js development, it provides fast service startup and hot reloading for efficient coding work, packaging and publishing to obtain the optimal output application.

Browser support: Supports common browsers on the market, does not support IE, and does not support versions below Chrome 97.

Vue Introduction

Main features:

- Easy to learn and use

Built on standard HTML , CSS and JavaScript , with easy-to-use APIs and top-notch documentation.

- Excellent performance

After being optimized by the compiler, a fully responsive rendering system requires almost no manual optimization.

- Flexible and changeable

A rich and progressively integrated ecosystem that allows for easy switching between libraries and frameworks based on the size of the application.

Vue is a mature, battle-tested framework. It is one of the most widely used JavaScript frameworks in production environments today, with over 1.50 million users worldwide and over 10 million monthly downloads on npm .

Vue is a completely free and open source project released under the MIT License .

The latest version of Vue (3.x) only supports browsers that natively support ES2015 . This does not include IE11.

Responsive system

Vue's most iconic feature is its low-intrusive reactive system. Component states are made up of reactive JavaScript objects. Views are automatically updated when they are changed.

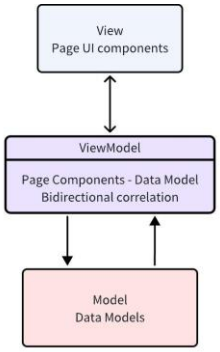
Vue 2 uses Object.defineProperty's getter and setter to create reactive objects, while Vue 3 uses Proxy to create reactive objects.

The reactive system uses the MVVM architecture pattern, which consists of three parts: Model, View, and ViewModel.

The Model layer represents the data model, usually defining the business logic for data modification and operation around the backend interface data rules.

View represents the UI component of the page, which is composed of HTML + CSS to form an interactive component and transform the data model into UI display.

ViewModel is an object that synchronizes View and Model. Using the API provided by Vue can better synchronize the transformation.



Rendering mechanism

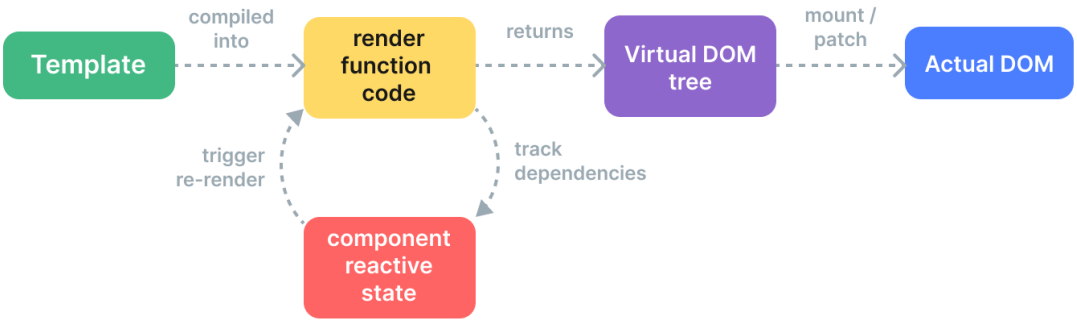
From a macro perspective, the following things happen when Vue components are

mounted:

1. Compile: The Vue template is compiled as a render function: that is, a function that returns the virtual DOM tree. This step can be done in advance through the build step or immediately through the use of the runtime compiler.

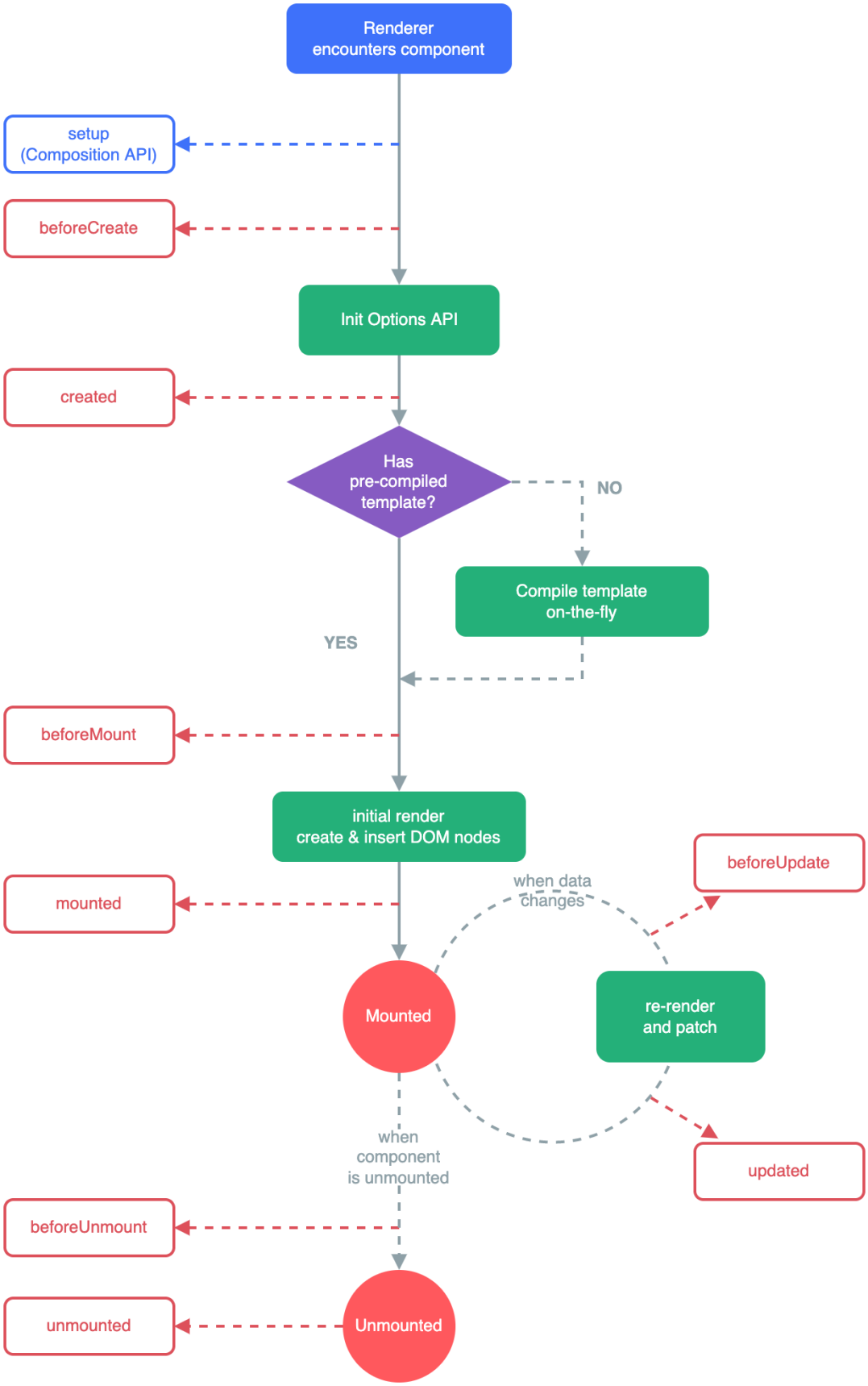
2. Mount: The runtime renderer calls the render function, traverses the returned virtual DOM tree, and creates the actual DOM node based on it. This step is performed as a reactive side effect , so it tracks all reactive dependencies used in it.

3. Update: When a dependency changes and the side effect reruns, an updated virtual DOM tree is created. The runtime renderer traverses the new tree, compares it to the old tree, and applies the necessary updates to the real DOM.



Component life cycle

Each Vue component instance goes through a series of initialization steps when created, such as setting up data listeners, compiling templates, mounting instances to the DOM , and updating the DOM when data changes. During this process, it also runs functions called lifecycle hooks, giving developers the opportunity to run their code at specific stages.


Renderer
encounters component
setup
(Composition API)
beforeCreate
Init Options API
created
Has
pre-compiled
template?
NO
Compile template
on-the-fly
YES
beforeMount
initial render
create & insert DOM nodes
mounted
when data
changes
beforeUpdate
Mounted
re-render
and patch
when
component
is unmounted
updated
beforeUnmount
unmounted
Unmounted

Interact with backend data

Through the API request provided by the backend, use the HTTP Client Fetch based on Promise to access the API, get the response result and display the data.

The following is an example code of components built in engineering:

```
TypeScript
<script lang="ts" setup>
import { ref, onMounted } from 'vue';

// Declaring Variables - Page Usage Data
let resultData = ref({})

// Declaration period - after page rendering
onMounted(() => {
  // For API access
  fetch('https://api.example.com/x/web-
interface/zone?jsonp=jsonp', {
    headers: {
      'accept-language': 'zh-CN,zh;q=0.9',
    },
    referrerPolicy: 'no-referrer-when-downgrade',
    body: null,
    method: 'GET',
    mode: 'cors',
  })
    .then(res => {
      return res.json();
    })
    .then(data => {
      // Assign the response result to the page variable
      resultData.value = data
      // Browser Console output response result
      console.log(data);
    });
});
</script>

<template>
  <div>
    <!-- After the page is opened, wait for ms to get the data
```

```
display -->
    {{ resultData }}
  </div>
</template>

<style lang="less" scoped></style>
```

In actual engineering projects, intercept request data formats are used for request responses.

Vue project

The background front-end project built with vue3 + vite4 + typescript , the following is the common directory structure, in the actual development process, will make some small changes according to the project.

```
Plain Text
probject_vue3
├── bin                                directory-executing
scripts
├── public                             directory-Public file
definitions
│   ├── favicon.ico                    file-favicon icon
│   ├── font_8d5l8fzk5b87iudi.js       file-menu font icon
│   ├── loading.js                     file-first time loading
Loading
│   └── ...
├── src                                directory-source code
│   ├── api                            directory-all requests
│   ├── assets                         directory-images,
styles, etc. than pre-compiled resources
│   ├── components                     directory-global utility
components
│   ├── constants                      directory-global
constants
│   ├── directive                      directory-global
directives
│   ├── hooks                          directory-Hook Tools
│   ├── layouts                        directory-Framework
```

```
Layouts
│   ├── plugins                          directory-generic
variable methods
│   ├── router                           directory-Routing
│   ├── store                            directory-Global
Variable Cache
│   ├── typings                          directory-global type
declarations
│   ├── utils                            directory-static utility
methods
│   ├── views                            directory-page display
components
│   ├── App.vue                          file-entry page
component
│   └── main.js                          file-entry Load
components Initialization, etc.
├── .editorconfig
├── .env.development                     file-development
environment configuration
├── .env.production                      file-production
environment configuration
├── .gitignore
├── index.html                           file-root HTML template
├── LICENSE
├── package.json                         file-Project Package
Information
├── README.md                            file-project description
├── tsconfig.json                        file-ts configuration
├── tsconfig.node.json
└── vite.config.ts                       file-vite configuration
```