১. **Vue.js** কী? **(framework, spz, reactive & component)**
২. **Vue-**এর প্রধান বৈশিষ্ট্য কী কী?
৩. **Vue instance** কী এবং এর ভূমিকা কী?
৪. **Computed Properties** এবং **Methods-**এর মধ্যে পার্থক্য কী?
৫. **Vue-**তে `v-for` কীভাবে কাজ করে?
৬. **Vue Lifecycle Hooks** কী?
৭. `v-model` কী?
৮. **Vue Router** কী?
৯. **Vuex** কী এবং এর ব্যবহারের ক্ষেত্র কী?
১০. **Vue-**তে **Event Handling** কীভাবে করা হয়?
১১. **Vue.js-**এ অ্যাসিনক্রোনাস ডাটা হ্যান্ডলিং কীভাবে করবেন?
১২. **Vue-**তে `watchers` কী?
১৩. **Composition API** এবং **Options API-**এর মধ্যে পার্থক্য কী?
GET, POST, PUT, PATCH, DELETE সম্পর্কে বিস্তারিত আলোচনা করো
১. **Vue.js-**এর **Composition API** কী? কেন এটি **Options API** থেকে আলাদা?
২. **Vue.js-**এ অ্যাসিনক্রোনাস অপারেশন পরিচালনার সেরা পদ্ধতি কী?
৩. **Vue.js-**এ **Custom Directives** কীভাবে তৈরি করবেন?
৪. **Vue.js-**এ **Render Function** কী এবং কেন এটি গুরুত্বপূর্ণ?
৫. **Vue-**তে কীভাবে **State Management** করা হয়? **Vuex** এবং **Pinia-**এর মধ্যে পার্থক্য কী?
৬. **Vue.js-**এ **Lazy Loading** কীভাবে কাজ করে?
৭. **Vue.js-**এ **Slots** এবং **Scoped Slots-**এর মধ্যে পার্থক্য কী?
৮. **Vue.js-**এ **HOC (Higher-Order Components)** কী এবং এটি কীভাবে কাজ করে?
৯. **Vue 3-**এ **Teleport** কী?
১০. **Vue.js-**এ **SSR (Server-Side Rendering)** কী এবং এটি কেন ব্যবহৃত হয়?
১১. **Vue.js-**এ কীভাবে **WebSocket** ইন্টিগ্রেট করবেন?
১২. **Vue.js-**এ **Code Splitting** কী?

১. **JavaScript** কী এবং এটি কীভাবে কাজ করে?
২. **JavaScript-**এর ভেরিয়েবল ডিক্লারেশনের জন্য `var`, `let` এবং `const` এর মধ্যে পার্থক্য কী?
৩. **Arrow Function** এবং **Regular Function-**এর মধ্যে পার্থক্য কী?
৪. **JavaScript-**এর `==` এবং `===` এর মধ্যে পার্থক্য কী?
৫. **JavaScript-**এ **Closures** কী?
৬. **JavaScript-**এ **Event Loop** কী?
৭. **JavaScript-**এ `async` এবং `await` কী?
৮. **JavaScript-**এ **Hoisting** কী?
৯. **JavaScript-**এ `call()`, `apply()`, এবং `bind()` এর মধ্যে পার্থক্য কী?
১০. **Promises** কী এবং এটি কীভাবে কাজ করে?
১১. **JavaScript-**এ `this` কী?
১২. **JavaScript-**এ **Debouncing** এবং **Throttling-**এর মধ্যে পার্থক্য কী?

—

১. **JavaScript-**এ **Prototype** কী এবং এটি কীভাবে কাজ করে?
২. **JavaScript-**এ **Event Delegation** কী?
৩. **JavaScript-**এ **WeakMap** এবং **WeakSet** কী?
৪. **Currying** কী এবং এটি কেন ব্যবহার করা হয়?
৫. **JavaScript-**এ **Memoization** কী?
৬. **JavaScript-**এ **Generators** এবং **Iterators** কী?
৭. **JavaScript-**এ `Object.create()` কীভাবে কাজ করে?
৮. **JavaScript-**এ **Debouncing** এবং **Throttling-**এর পার্থক্য কী?
৯. **JavaScript-**এ `call`, `apply` এবং `bind` এর মধ্যে পার্থক্য কী?
১০. **JavaScript-**এ `Promise.all` এবং `Promise.race` কী?

Link: https://chatgpt.com/share/674f056a-ced4-8005-9f61-f0fa421da158

১. **Vue.js** কী**?** (framework, spa, reactive & component)
২**. Vue-**এর প্রধান বৈশিষ্ট্য কী কী**?** (5 - declarative rendering, components, directives, reactivity system, routing, state management**)**
৩**. Vue instance** কী এবং এর ভূমিকা কী**?** (ডম (DOM) এর সাথে সংযুক্ত করে এবং ডাটা মডেল এবং ভিউয়ের মধ্যে সম্পর্ক স্থাপন করে।)
৪**. Computed Properties** এবং **Methods-**এর মধ্যে পার্থক্য কী**?**
(
Computed Properties:

- এটি নির্ভরশীল ডাটার ভিত্তিতে একটি ক্যাশড (cached) ভ্যালু রিটার্ন করে।
- এটি তখনই পুনর্গণনা করে, যখন নির্ভরশীল ডাটা পরিবর্তিত হয়।

Methods:

- এটি প্রতিবার কল করার সময় নতুন ভ্যালু রিটার্ন করে।
- এটি কোনো ক্যাশিং করে না।

)
৫**. Vue-**তে `v-for` কীভাবে কাজ করে**?** (`v-for` একটি ডিরেক্টিভ, যা একটি লুপ তৈরি করতে ব্যবহার করা হয়। এটি লিস্টের প্রতিটি আইটেমের জন্য একটি DOM এলিমেন্ট জেনারেট করে।)
৬**. Vue Lifecycle Hooks** কী**?**
Answer: (beforeCreate(),Created(), beforeMount(), Mounted()), (beforeUpdate(), Updated(), beforeUnmount(), unmounted())
৭**.** `v-model` কী**?** (যা দুই দিকের ডাটা বাইন্ডিং (Two-way Data Binding) নিশ্চিত করে।)
৮**. Vue Router** কী**?**
(Vue Router একটি লাইব্রেরি, যা Vue.js-এ রাউটিং ব্যবস্থাপনা করতে ব্যবহৃত হয়। এটি বিভিন্ন URL-এর ভিত্তিতে ভিউ পরিবর্তন করতে সাহায্য করে।)
৯**. Vuex** কী এবং এর ব্যবহারের ক্ষেত্র কী**?** (Vuex হলো একটি স্টেট ম্যানেজমেন্ট লাইব্রেরি, যা ডাটা এবং স্টেট সেন্ট্রালাইজড করে।, state, getter, mutation, action)
১০**. Vue-**তে **Event Handling** কীভাবে করা হয়**?** (Vue-তে ইভেন্ট হ্যান্ডলিং করার জন্য `v-on` ডিরেক্টিভ বা সংক্ষেপে `@` ব্যবহার করা হয়।)
১১**. Vue.js-**এ অ্যাসিনক্রোনাস ডাটা হ্যান্ডলিং কীভাবে করবেন**?** (API কল করতে Axios বা Fetch ব্যবহার করা হয়।)
১২**. Vue-**তে `watchers` কী**?** (ডাটার পরিবর্তন পর্যবেক্ষণ করে এবং সেই অনুযায়ী কোড এক্সিকিউট করে।)
১৩**. Composition API** এবং **Options API-**এর মধ্যে পার্থক্য কী**?**
(
Composition API:

- ফাংশন ভিত্তিক।
- Vue 3-এ পরিচিত হয়েছে।
- কোড আরও রিইউজেবল এবং মডুলার।

Options API:

- অবজেক্ট ভিত্তিক।

- Vue 2 এবং আগের ভার্সনে ব্যবহৃত হয়।

)
GET, POST, PUT, PATCH, DELETE সম্পর্কে বিস্তারিত আলোচনা করো
(

সারাংশ টেবিল:

মেথড	উদ্দেশ্য	Idempotent	ব্যবহার
GET	ডাটা রিড করা	হ্যাঁ	সার্ভার থেকে ডাটা ফেচ করা।
POST	নতুন ডাটা তৈরি	না	নতুন রিসোর্স যোগ করা।
PUT	সম্পূর্ণ রিসোর্স আপডেট করা	হ্যাঁ	বিদ্যমান ডাটা রিপ্লেস করা।
PATCH	আংশিক রিসোর্স আপডেট করা	হ্যাঁ	ডাটার নির্দিষ্ট অংশ আপডেট করা।
DELETE	রিসোর্স ডিলিট করা	হ্যাঁ	সার্ভার থেকে ডাটা ডিলিট করা।

উদাহরণ**:**

GET এবং **POST:**

- GET: ইউজারের লিস্ট দেখতে।
- POST: নতুন ইউজার অ্যাড করতে।

PUT এবং **PATCH:**

- PUT: পুরো প্রোফাইল আপডেট করতে।
- PATCH: শুধুমাত্র নাম বা ইমেল আপডেট করতে।

DELETE:

- ইউজার ডিলিট করার জন্য।

)
১**. Vue.js-**এর **Composition API** কী**?** কেন এটি **Options API** থেকে আলাদা**?**
(

Composition API বনাম Options API:

Composition API	Options API

কোড ফাংশন ভিত্তিক।	কোড অবজেক্ট ভিত্তিক।
ভালো স্কোপ ম্যানেজমেন্ট।	স্কোপ ম্যানেজমেন্ট জটিল।
বড় এবং জটিল অ্যাপ্লিকেশনে কার্যকর।	ছোট অ্যাপ্লিকেশনে সহজে ব্যবহৃত।

)
২**. Vue.js-**এ অ্যাসিনক্রোনাস অপারেশন পরিচালনার সেরা পদ্ধতি কী**?**
(
Axios বা Fetch API দিয়ে ডাটা ফেচ করা হয়।
অ্যাসিনক্রোনাস অপারেশনগুলো try-catch ব্লকের মাধ্যমে হ্যান্ডল করা হয়।

)
৩**. Vue.js-**এ **Custom Directives** কীভাবে তৈরি করবেন**?** (Custom Directives এমন একটি পদ্ধতি, যা Vue-তে DOM ম্যানিপুলেশনের জন্য ব্যবহার করা হয়।)
৪**. Vue.js-**এ **Render Function** কী এবং কেন এটি গুরুত্বপূর্ণ**?**
(

Render Function একটি JavaScript ফাংশন, যা Vue কম্পোনেন্টকে ডাইনামিকভাবে **DOM** তৈরি করতে সক্ষম করে। এটি Template-এর বিকল্প।

কেন গুরুত্বপূর্ণ:

- ডাইনামিক কম্পোনেন্ট তৈরি।
- জটিল লজিক হ্যান্ডল করা।
- কোড আরও কাস্টমাইজড করা।

)
৫**. Vue-**তে কীভাবে **State Management** করা হয়**? Vuex** এবং **Pinia-**এর মধ্যে পার্থক্য কী**?**
(

Vuex	Pinia
মিউটেশন ব্যবহার করে স্টেট আপডেট।	সরাসরি স্টেট পরিবর্তন করা যায়।
বড় অ্যাপ্লিকেশনে ব্যবহৃত।	সহজ এবং দ্রুত কাজের জন্য।

)
৬**. Vue.js-**এ **Lazy Loading** কীভাবে কাজ করে**?**
(Lazy Loading একটি কৌশল, যা প্রয়োজন অনুযায়ী কম্পোনেন্ট বা রিসোর্স লোড করে। এটি অ্যাপ্লিকেশনের পারফরম্যান্স বাড়াতে সাহায্য করে।)
৭**. Vue.js-**এ **Slots** এবং **Scoped Slots-**এর মধ্যে পার্থক্য কী**?**

(
Slots:

- প্যারেন্ট কম্পোনেন্ট থেকে চাইল্ড কম্পোনেন্টে কন্টেন্ট পাস করার জন্য।
- সাধারণত স্ট্যাটিক ডাটা পাস করতে ব্যবহৃত হয়।

Scoped Slots:

- চাইল্ড কম্পোনেন্ট থেকে ডাটা প্যারেন্টে ফেরত পাঠাতে ব্যবহৃত হয়।
- ডাইনামিক ডাটা শেয়ার করতে সক্ষম।

)
৮. **Vue.js-**এ **HOC (Higher-Order Components)** কী এবং এটি কীভাবে কাজ করে**?**
(

HOC হলো একটি ফাংশন, যা একটি কম্পোনেন্টকে ইনপুট হিসেবে গ্রহণ করে এবং একটি নতুন কম্পোনেন্ট আউটপুট হিসেবে প্রদান করে।

ব্যবহার:

- কোড রিইউজ।
- লজিক ভাগাভাগি করা।

)
৯. **Vue 3-**এ **Teleport** কী**?** (Teleport Vue 3-এর একটি ফিচার, যা একটি কম্পোনেন্টকে **DOM** ট্রি থেকে অন্য স্থানে সরানোর অনুমতি দেয়।)
১০. **Vue.js-**এ **SSR (Server-Side Rendering)** কী এবং এটি কেন ব্যবহৃত হয়**?**
(

SSR একটি কৌশল, যা সার্ভারে **HTML** জেনারেট করে এবং ব্রাউজারে প্রেরণ করে।

কেন ব্যবহৃত হয়:

1. SEO-বান্ধব।
2. দ্রুত লোডিং সময়।

SSR ফ্রেমওয়ার্ক:

- Nuxt.js

)
১১. **Vue.js-**এ কীভাবে **WebSocket** ইন্টিগ্রেট করবেন**?**
(
Vue.js-এ WebSocket ইন্টিগ্রেট করে রিয়েল-টাইম ডাটা আপডেটের জন্য ব্যবহার করা হয়।)
১২. **Vue.js-**এ **Code Splitting** কী**?** (Code Splitting হলো অ্যাপ্লিকেশনের বিভিন্ন অংশকে আলাদা আলাদা চাংকে ভাগ করার প্রক্রিয়া, যা অ্যাপের লোডিং টাইম কমায়।)

১. **JavaScript** কী এবং এটি কীভাবে কাজ করে**?**
(
JavaScript হলো একটি ডাইনামিক , loosly type, এবং ইভেন্ট-ড্রিভেন প্রোগ্রামিং ভাষা, যা ওয়েব পেজে ইন্টার‍্যাক্টিভ ফিচার যুক্ত করতে ব্যবহৃত হয়। এটি ব্রাউজারে রান করার জন্য JavaScript Engine ব্যবহার করে। উদাহরণস্বরূপ, Chrome-এ V8 Engine এবং Firefox-এ SpiderMonkey।
)
২**. JavaScript-**এর ভেরিয়েবল ডিক্লারেশনের জন্য **`var`**, **`let`** এবং **`const`** এর মধ্যে পার্থক্য কী**?**
(

ফিচার	var	let	const
Scope	Function Scope	Block Scope	Block Scope
Redeclaration	সম্ভব	সম্ভব নয়	সম্ভব নয়
Reassignment	সম্ভব	সম্ভব	অসম্ভব (Primitive Data)
Hoisting	হ্যাঁ	হ্যাঁ (Temporal Dead Zone)	হ্যাঁ (Temporal Dead Zone)

)
৩**. Arrow Function** এবং **Regular Function-**এর মধ্যে পার্থক্য কী**?**
(
Arrow Function:

- `this` কনটেক্সট প্যারেন্ট অবজেক্টের সাথে বাঁধা।
- সংক্ষিপ্ত সিনট্যাক্স।

Regular Function:

- `this` কনটেক্সট নিজের অবজেক্টের সাথে বাঁধা।
- আরও বিস্তারিত কোড লিখতে হয়।

)
৪**. JavaScript-**এর **`==`** এবং **`===`** এর মধ্যে পার্থক্য কী**?**
(

- `==` (Loose Equality): ডাটা টাইপ ইগনোর করে শুধু ভ্যালু চেক করে।
- `===` (Strict Equality): ভ্যালু এবং টাইপ দুটোই চেক করে।

)
৫. **JavaScript-**এ **Closures** কী? (Closure হলো একটি ফাংশন, যা তার আশেপাশের স্কোপ থেকে ভ্যারিয়েবলের অ্যাক্সেস রাখে, এমনকি সেই স্কোপ শেষ হলেও।)
৬. **JavaScript-**এ **Event Loop** কী?
(

Event Loop হলো একটি মেকানিজম, যা কলব্যাক কিউতে থাকা কাজগুলো একটার পর একটা প্রক্রিয়া করে। এটি JavaScript-এর single-threaded asynchronous nature ম্যানেজ করে।

ধাপ:

1. Call Stack
2. Web APIs
3. Callback Queue
4. Event Loop

)
৭. **JavaScript-**এ `async` এবং `await` কী?
(

- `async`: একটি ফাংশনকে asynchronous বানায়।
- `await`: একটি প্রমিস রেজল্ভ না হওয়া পর্যন্ত অপেক্ষা করে।

)
৮. **JavaScript-**এ **Hoisting** কী? (Hoisting হলো একটি প্রক্রিয়া, যেখানে ভ্যারিয়েবল এবং ফাংশন ডিক্লারেশনগুলো স্কোপের শীর্ষে নিয়ে যাওয়া হয়।)
৯. **JavaScript-**এ `call()`, `apply()`, এবং `bind()` এর মধ্যে পার্থক্য কী?
(
`call()`: ফাংশনকে ইনভোক করে এবং আর্গুমেন্ট আলাদাভাবে পাস করে।
`apply()`: ফাংশনকে ইনভোক করে এবং আর্গুমেন্ট একটি অ্যারে হিসেবে পাস করে।
`bind()`: নতুন ফাংশন তৈরি করে এবং আর্গুমেন্ট পরে পাস করা যায়।
)
১০. **Promises** কী এবং এটি কীভাবে কাজ করে? (Promise হলো একটি JavaScript অবজেক্ট, যা অ্যাসিনক্রোনাস অপারেশনের সফলতা বা ব্যর্থতার **(resolve, reject)** রেজাল্ট রিপ্রেজেন্ট করে।)
১১. **JavaScript-**এ `this` কী? (`this` হলো একটি কিওয়ার্ড, যা বর্তমান অবজেক্টকে রেফার করে। এটি কনটেক্সটের উপর নির্ভর করে পরিবর্তিত হয়।)
১২. **JavaScript-**এ **Debouncing** এবং **Throttling-**এর মধ্যে পার্থক্য কী?
(
Debouncing: একটি ফাংশন নির্দিষ্ট সময়ের পরে ট্রিগার হয়।
Throttling: একটি ফাংশন একটি নির্দিষ্ট সময়ের ব্যবধানে বারবার ট্রিগার হয়।
)
—
১. **JavaScript-**এ **Prototype** কী এবং এটি কীভাবে কাজ করে?
(

Prototype হলো JavaScript-এর একটি মেকানিজম, যা অবজেক্টের মধ্যে প্রপার্টি এবং মেথড শেয়ার করার সুযোগ দেয়। প্রতিটি অবজেক্টের একটি prototype থাকে এবং এটি তার প্যারেন্ট অবজেক্টের প্রপার্টি এবং মেথড অ্যাক্সেস করতে পারে।
)
২. **JavaScript-**এ **Event Delegation** কী?
৩. **JavaScript-**এ **WeakMap** এবং **WeakSet** কী?
৪. **Currying** কী এবং এটি কেন ব্যবহার করা হয়?
৫. **JavaScript-**এ **Memoization** কী?
৬. **JavaScript-**এ **Generators** এবং **Iterators** কী?
৭. **JavaScript-**এ `Object.create()` কীভাবে কাজ করে?
৮. **JavaScript-**এ **Debouncing** এবং **Throttling-**এর পার্থক্য কী?
৯. **JavaScript-**এ `call`, `apply` এবং `bind` এর মধ্যে পার্থক্য কী?
১০. **JavaScript-**এ `Promise.all` এবং `Promise.race` কী?

Vue js key features:
1. Declarative Rendering:- fixed data, auto update করবে না।
2. Reactivity:- automatically update করবে।

App.vue == sfc(single page component)
I. options API; ii. Composition API

- [ ] Ref method use করে korbo করবো।

{{ username }} -> text interpolation

- [ ] Ref এর variable গুলো যখন <template> এ use করি, তখন ওইখানে variable.value এর মানটাই ওইখানে নেয়। তাই <template> এ value লিখতে হয় না। কিন্তু <script> এ value দিয়ে access করতে হয়।
- [ ] Two-way-binding = v-model

Event Modifiers:

I. .stop
II. .prevent
III. .once

Watcher: যখন কোনো change করবো, সাথে সাথে দেখতে পারবো & capture করতে পারবো।
:v-show = derective

- ☐ Conditional element rendering এর জন্য <template> tag use হয়।
- ☐ Computed property: Method এর মতো কাজ করে, একটি value return করতে পারি, যা state এর মতো কাজ করবে।
- ☐ Js এ shallow copy মানে হলোঃ Memeory location আগেরটাই থাকে।
- ☐ JS এ deep copy মানে হলোঃ নতুন Memeory location নেয়।
- ☐ Question: কোন সিনারিয়োতে computed method use করবো??
  Answer:
  `computed` method ব্যবহার করা হয় তখন, যখন আপনাকে **Vue.js**-এর ভিতরে কোনো ডেটা ভিত্তিক ডেরাইভড **(derived)** মান বা ফলাফল গণনা (**calculation**) করতে হবে। অর্থাৎ, যখন কোনো মান **Vuex state**, **props**, বা অন্যান্য ডেটার উপর ভিত্তি করে নির্ভরশীল এবং পরিবর্তিত ডেটার সাথে স্বয়ংক্রিয়ভাবে আপডেট হতে হবে।

  ### উপসংহার:

- `computed` ব্যবহার করবেন যখন ডেরাইভড ডেটার জন্য ক্যাশিং দরকার হয়।
- `methods` ব্যবহার করবেন যখন শুধুমাত্র অ্যাকশন বা ফাংশনালিটি প্রয়োজন হয়।

  ### সহজ নিয়ম:

  "ডেটা তৈরি করতে হলে `computed`, অ্যাকশন সম্পাদন করতে হলে `methods`।"

Props use: parent componet থেকে child এ data pass করার জন্য props use করতে পারি। import {defineProps}

Emit use: emit এর মাধ্যমে child থেকে parent এ event এর মাধ্যমে method call করতে পারি।

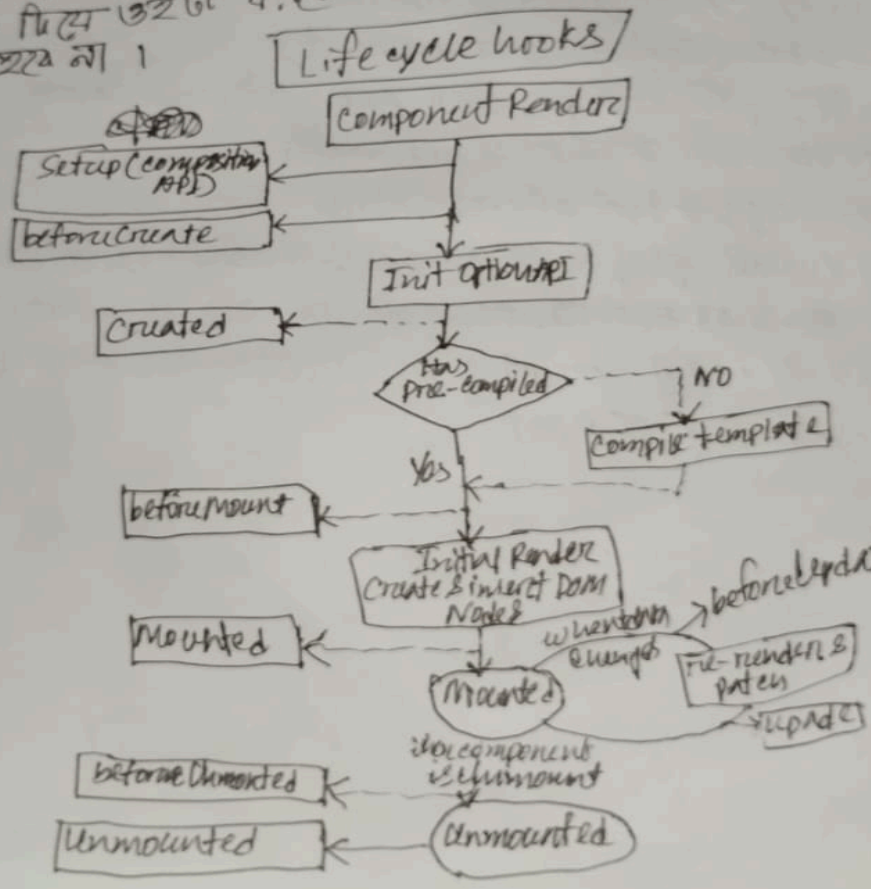
class: 8

① Slot :- একাধিক Named slot use করে করতে পারি। (slot)

② Fall through attribute :- কোনো কিছু component এ design এ component এ call করার সময় design থেকে করতে পারি।

③ component v-model :- Child component এর input field এ data ফিরিয়ে দিতে Parent component
① Import করতে হবে {define model};
② Parent & child component এ Two-way binding করতে পারি।

④ Provide & inject :- Props বারবার pass না করে, Provide & inject দিয়ে ওটা ধরতে হবে। এতে বার বারার props
parent component এ। child component এ
ফিরে হবে না।



REDMI 10C 04/12/2024 23:07

Teviral

class:-4 → Vuex

① State:- বিভিন্ন array or object কে var আকারে রাখবো।
② getter:- state এর বিভিন্ন value নিয়ে আসি, get করি
③ mutation:- পরিবর্তন :- mutation দিয়ে state এর value গুলো change করি / mutate করি। commit করতে হয় by default → synchronous; so, async operation করতে পারবো না। only state modify করতে।
④ Action:- বিভিন্ন Action perform করা যায়. api call. mutation এর বিভিন্ন method call হয়ে থাকে। Asyonc operation করতে হলে action ও করতে হবে। dispatch use করতে হবে।

- Vuex এ Two-way-bindings করা যায়। @input এ value নিয়ে সেটা, mutation এ call করে state এর value change করে করতে পারি।
- আমরা, v-model use না করে @get(), set করলাম, এভাবেও use করতে পারি।

* গুছিয়ে এর জন্য → namespace use করতে পারি।

যে pilgrim-list ...

ছু না

Class:-5 ⇒ Route

① :to={ name: 'create-post' }

② router.push({ name: 'Post' })

③ ② ⇒ direct use তে যায়।

④ Navigation guard :- Route এ validation add হয়।

*Router.beforeEach((to, from, next) => { }):

*go

*history

*useRoute :- Current Route এর Information পেতে পারি।

*useRouter :- পুরো Router এর Information পাওয়া যায়।

*component: () ⇒ import (../view/view.vue)

↓

lazy loading (আলাদা .js file load হয় & সেটা দরকারের সময় run হয়)

beforeMount

Mounted

beforeUnmount

Vue

(ii) Public ও রাখবো, যেগুলো Normally আমরা change করতে না, যেগুলো। Suppose: logo.Image

(iii) Src→assets ⇒ ও রাখবো, যেগুলো Regular change হবে। Suppose ⇒ css, js

(iv) mounted() { } ⇒ Component টা load হওয়ার আগে সাথে কিছু করার দরকার হয়। like ⇒ document.ready(), then যে কাজগুলো করি আরকি।

(v) RouteView ⇒ কাজ :- আমরা Page এর ভিতর আসি যদি বসিয়ে না রেখে, তাহলে আমরা যেয়ে দেখাবে. সেটা দেখানোর জন্য RouteView টা দরকার।

(vi) Vue Templating :- ৪টা section থাকে

(i) <template></template> ⇒ section
(ii) <script></script> ⇒ section
(iii) <style></style> ⇒ section ⇒ (i) Global style
(a) scope করলে শুধু ওই component (ii) Component wise style
(ওই style টা থাকে।
(b) Globally use করতে চাইলে; আলাদা css file create করে, main.js file টা import করে দিতে হবে।

Lifecycle Hooks:

(i) Component load হওয়ার সময়ই ৪টা function call হয়।
(a) beforeCreate() → Initially component যখন load হয়, তখনই এই ২টা hook call হয়।
(b) created()
(c) beforeMount() → যখন script file টা load হয়, তখনই এই ২টা call হয়।
(d) mounted()

এইগুলো call হওয়ার পরই আমরা component কে দেখতে পাই।

Teviral

(a) suppose আমাদের table show করাইতে হবে। এইজন্য আমাদের mounted এর ভিতর axios call করে Data আনতে হবে। তারপরে করলে আর আসবে না। Bez তারপরে call হয় :-
(i) beforeUpdate (ii) updated; (iii) beforeUnmount; (iv) unmounted
(component টার আর কোনো অস্তিত্ব থাকে না)

(v) যে পেজে আছি, ও পেজে যেকোনো কিছু পেজে পেজে আলাদা component এ (beforeUnmounted) unmounted() call হয় & new page এ mounted() call করি

(vi) Route তৈরি করার ২টা way আছে। (a) path: '/', name: 'home', component: HomePage

সব ফাইল JS এ load হবে

(b) { path: '/about', name: 'about', component: () => import('../views/AboutView') }

এখানে আমার যদি JS এর ফাইল থাকে, সেইটা আগে load হয়। Lazy loading

(vii) যখন, যখন আমি component এ কিছু update করবো, তখন beforeUpdate() & updated() এই ২টা call হয়।

(iv) data() {
return {
string: '',
object: {},
array: [],
boolean: false,
number: 0
}
}
→ static value নিতে হলে এইভাবে নিবো।

method: {
}
→ function নিয়ে এখানে write করি এর মধ্যে নিবো।

(vi) Slot :- ধরেন, আমার একটা Button আছে, ঐ বাটনের Just নাম change হবে, Design change হবে। তখন আমরা Slot use করবো।

(vii) Props পাঠানো। <SinglePost :posts="posts" />
অপরিবা: props: {
post: {
type: Object
required: true
}
}

(viii) Child থেকে Parent এ কিছু পাঠাইতে হলে emit use করতে হবে। $emit এর মধ্যে function এর নামটা বলে দিলেই হবে
this.$emit('delete-post', post);
value

ধরেন নিবো:-

Teviral

(a) Parent এ @delete="handleDelete" ; Parent এ emit এর event টাকে ধরতে হবে।

(ix) class Binding :- :class="{'fav': post.fav}"

(x) Computed: { } ⇒ Property এর মধ্যে কিছু লিখলে সয়াসরি রান করবে।

Router

(i) R.R Router= <router-view></router-view> ⇒ Route গুলো use করার জন্য।

(ii) <router-link to="/">Home</router-link>

Prop & Pass

:guideData="guideData"

Receive: ⇒ const props = defineProps({ guideData: Object. })

toRefs: const { guideData } = toRefs(props)

Function Pass

@previous-click="HandlePreviousClick"

receive: ⇒ const props = defineProps({ previousClick: Function() });

emit: ⇒ const emit = defineEmits(['previous-click'])

use: ⇒ const emitPreviousClick = () => { emit('previous-click'); }