



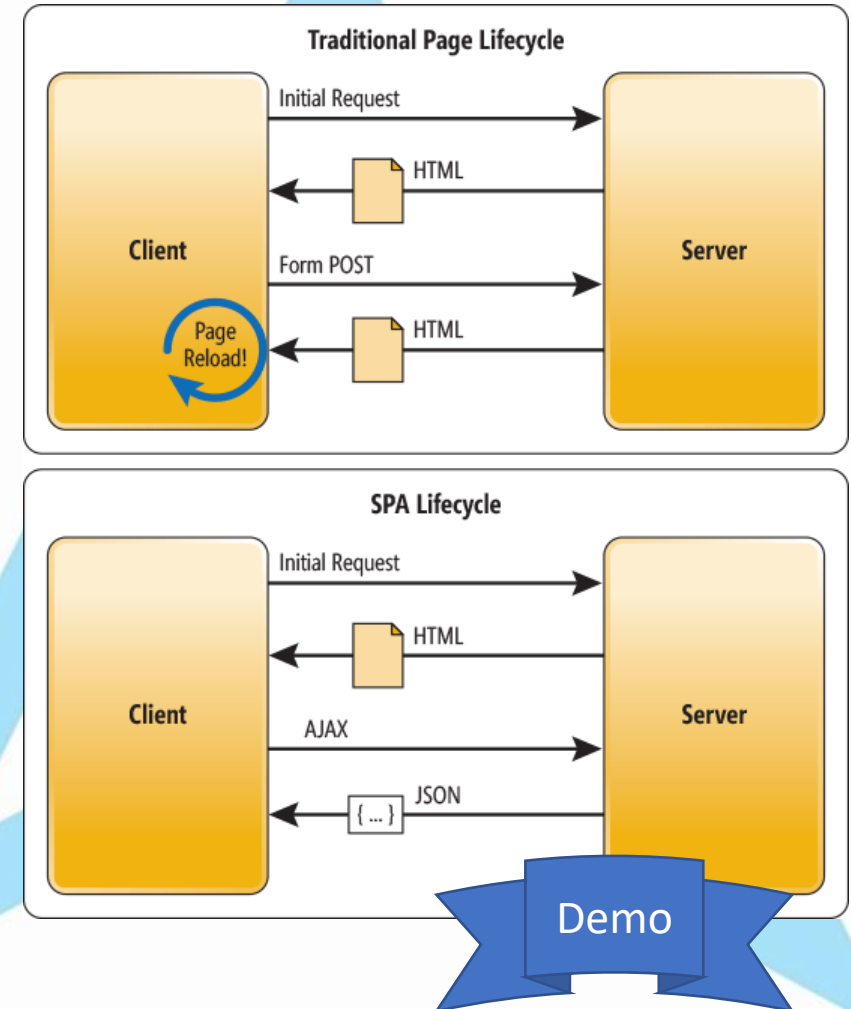
Module 4 Day 17

Vue Router

SPA vs. Traditional Architectures

- Traditional (e.g., Asp.NET MVC)
 - Client requests new pages as user navigates
 - Browser refreshes page
 - Routing on server determines what server resources to load
- Single-Page Application
 - Initial request loads application
 - Sections of the app page are replaced as user navigates
 - DOM is updated; page never fully refreshes
 - Routing on client determines what content to load into page sections

* *Routing – associating a URL with code to run*



Single-Page Applications (SPA)

- Interacts with the user by dynamically rewriting the current page rather than loading entire new pages from a server.
 - Avoids interruption of the user experience between successive pages
 - Behaves more like a desktop application
- All code – HTML, JavaScript, and CSS – is retrieved with a single page load
 - Or the appropriate resources are dynamically loaded as necessary, in response to user actions
- The page does not reload at any point in the process
 - Nor does control transfer to another page
- History API can be used to provide the perception of separate logical pages
 - So that user can still use the Back button and Bookmarks
- Often involves dynamic communication with web server behind the scenes

The Router Project

- router.js
 - Defines routes
- main.js
 - Include the router when the Vue root instance is created
- src/views folder
 - Contains .vue files which are considered "views". These can be "routed to"
- App.vue
 - `<router-link to="">` tag creates works like an anchor (minus the page reload)
 - "to" value can be a route or an object containing a name
 - `<router-view />` is the placeholder to render the View component



Demo

Defining Routes – router.js

```
11 routes: [  
12   {  
13     path: '/',  
14     redirect: {  
15       name: 'users'  
16     }  
17   },  
18   {  
19     path: '/users',  
20     name: 'users',  
21     component: Users  
22   },  
23   {  
24     path: '/users/:user_id',  
25     name: 'user',  
26     component: User  
27   },  
28 ]
```

Let's
Code

Dynamic Routing

- `:` indicates a "dynamic segment"
`/user/:id`
 - The value will be in `this.$route.params.id`
- A route can have more than one dynamic segment
`/user/:user_id/post/:post_id`
- You can get the values from the `$route` object

Route pattern	Matching path	<code>\$route.params</code>
<code>/user/:user_id/post/:post_id</code>	<code>/user/101/post/555</code>	<code>{ user_id: 101, post_id: 555 }</code> <code>\$route.params.user_id</code>

Let's
Code

Navigating a route

In HTML	In JavaScript	Notes
<code><router-link :to="{ name: 'user', params={user_id: 5} }"></code>	<code>this.\$router.push({ name: 'user', params={user_id: 5} });</code>	Normal navigation to next page.
<code><router-link :to="{ name: 'user', params={user_id: 5} }" replace></code>	<code>this.\$router.replace({ name: 'user', params={user_id: 5} });</code>	Does not put the current page in the history stack.
	<code>this.\$router.go(n)</code>	Goes to the n-th entry in the stack from here. +n means forward n pages; -n means backward n pages.

- When using router-link, class will be added if the link is active
 - router-link-active
 - router-link-exact-active

Let's
Code

Vue Component Lifecycle

- This slide for awareness only at this point
- You can "hook" lifecycle events and run code
- E.g., get data from the server on created()

