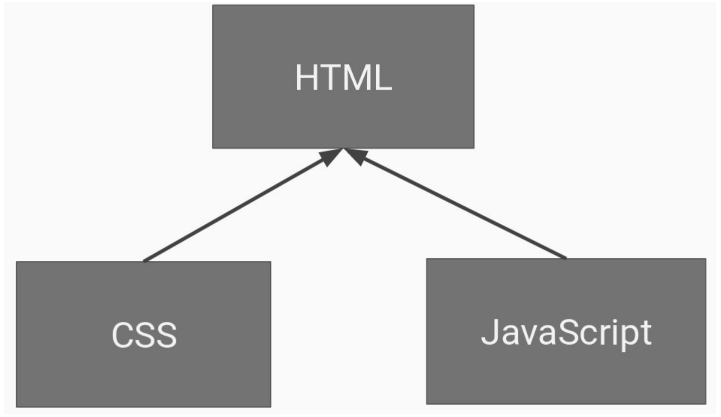
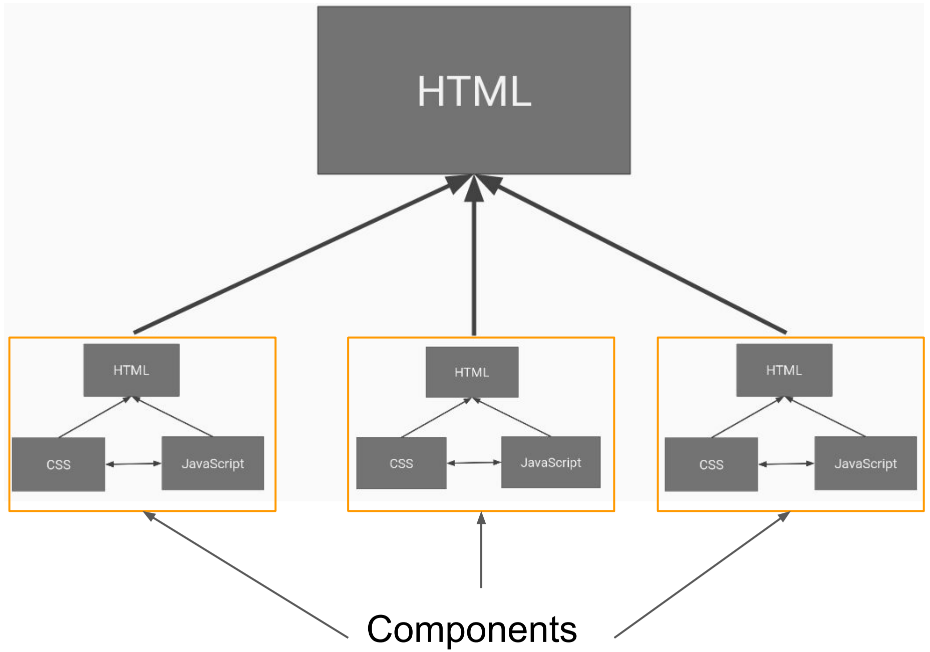
Module 3 Day 16

Vue Router

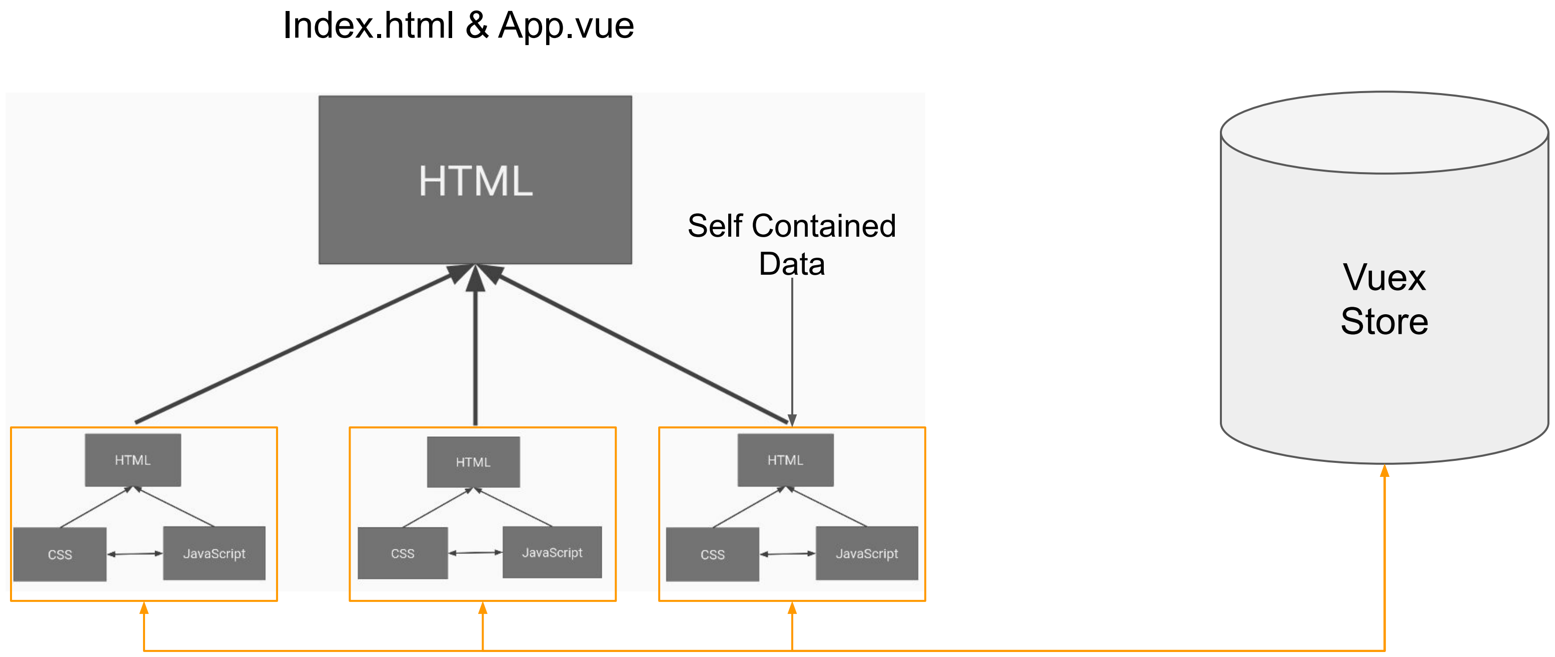
Vue: Evolution of Concepts - HTML to Vue



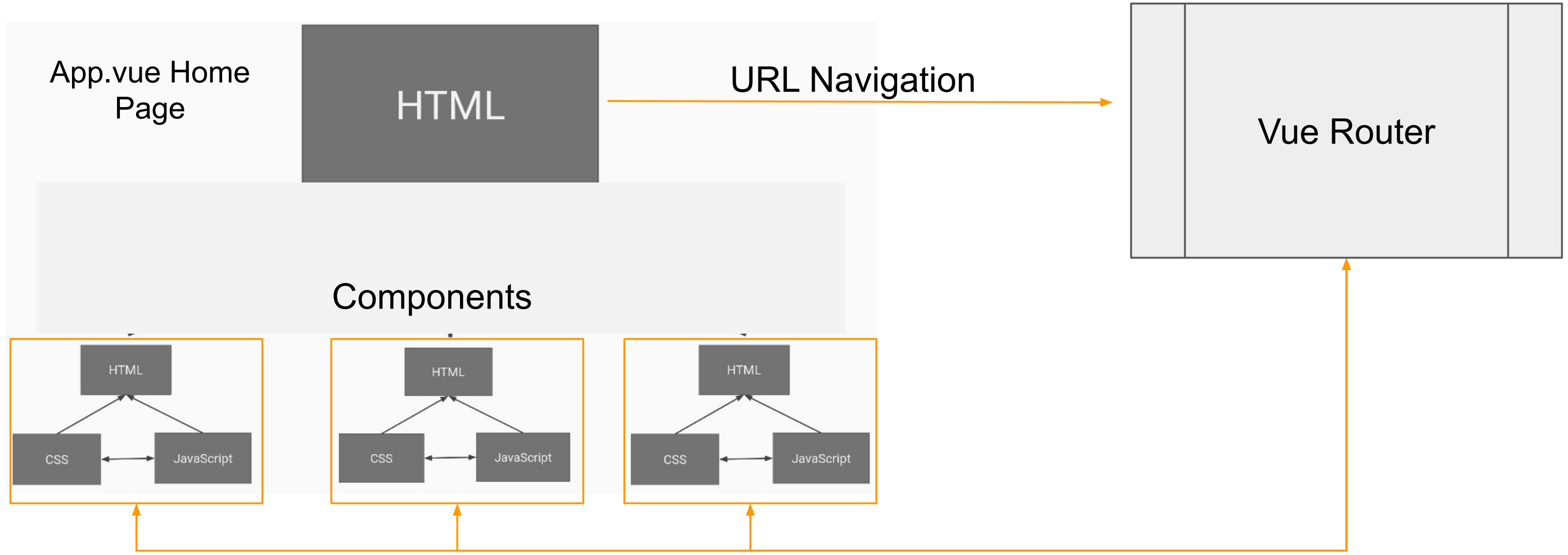
Index.html & App.vue



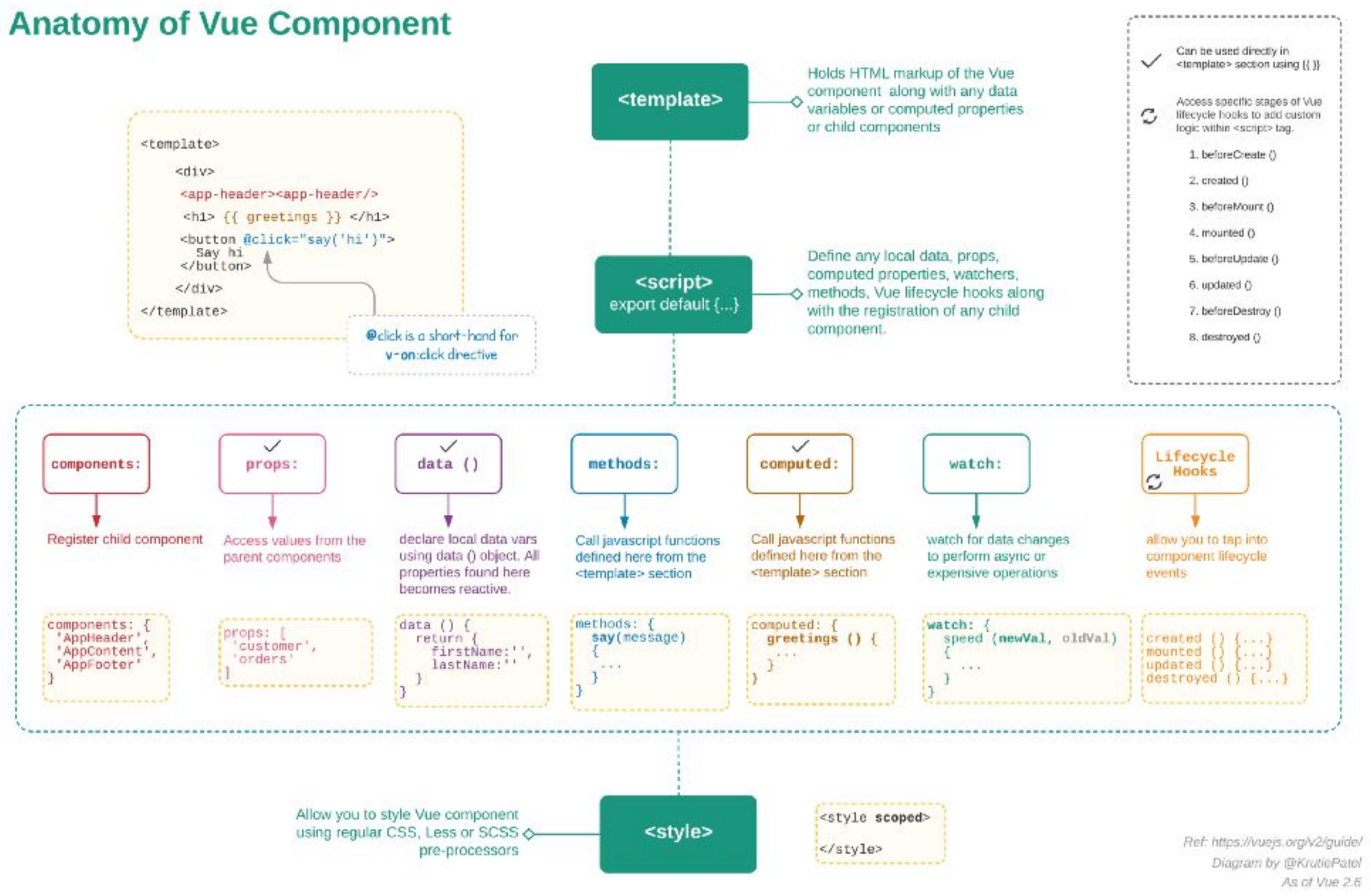
Vue: Evolution of Concepts - Vue to Vuex



Vue: Evolution of Concepts - Site Pages to Routes



Review



Anatomy of Vue Component

```
<template>
    <div>
     <app-header><app-header/>
     <h1> {{ greetings }} </h1>
     <button @click="say('hi')">
       Say hi
     </button>
    </div>
</template>
```

@click is a short-hand for v-on:click directive

<template>

Holds HTML markup of the Vue component along with any data variables or computed properties or child components

<script> export default {...}

Define any local data, props, computed properties, watchers, methods, Vue lifecycle hooks along with the registration of any child component.

✓ Can be used directly in <template> section using {{ }}

↻ Access specific stages of Vue lifecycle hooks to add custom logic within <script> tag.

1. beforeCreate ()
2. created ()
3. beforeMount ()
4. mounted ()
5. beforeUpdate ()
6. updated ()
7. beforeDestroy ()
8. destroyed ()

components:

Register child component

```
components: {
 'AppHeader',
 'AppContent',
 'AppFooter'
}
```

✓ **props:**

Access values from the parent components

```
props: [
 'customer',
 'orders'
]
```

✓ **data ()**

declare local data vars using data () object. All properties found here becomes reactive.

```
data () {
  return {
    firstName:'',
    lastName:''
  }
}
```

methods:

Call javascript functions defined here from the <template> section

```
methods: {
  say(message)
  {
    ...
  }
}
```

✓ **computed:**

Call javascript functions defined here from the <template> section

```
computed: {
  greetings () {
    ...
  }
}
```

watch:

watch for data changes to perform async or expensive operations

```
watch: {
  speed (newVal, oldVal)
  {
    ...
  }
}
```

↻ **Lifecycle Hooks**

allow you to tap into component lifecycle events

```
created () {...}
mounted () {...}
updated () {...}
destroyed () {...}
```

<style>

Allow you to style Vue component using regular CSS, Less or SCSS pre-processors

```
<style scoped>

</style>
```

Ref: https://vuejs.org/v2/guide/
Diagram by @KrutiePatel
As of Vue 2.6

What is Routing?

- Routing allows users to be redirected to a certain component via a URL.

- A view component is really like any other component and only differs in how it is called within the SPA framework (app.vue).

Vue Router: Creating a View and Route

1. Create View in src/views

```
<template>
  <div>
    <h1>Hi!</h1>
    <p>I am learning VueJS Routing</p>
  </div>
</template>
```

This is JUST LIKE any other component!

2. Update app.vue to include <router-views />

```
<template>
  <div id="app">
    <nav>
    </nav>
    <router-view />
  </div>
</template>
```

3. Import View in src/router/index.js

```
import Home from '../views/Home.vue'
```

4. Add route attributes to Routes Constant

```
const routes = [
  {
    path: '/',
    name: 'home',
    component: Home
  }
]
```

The Vue Router index.js file overview

To define a route, an index.js file is needed. This file is functionally a peer of App.vue and lives in the ./src/router directory.

```
import Vue from 'vue'
import Router from 'vue-router'
import Home from './views/Home.vue'
import About from './views/About.vue'
Vue.use(Router)
const routes = [
  {
        path: '/',
        name: 'home',
        component: Home },
  {
        path: '/about',
        name: 'about',
        component: About}
]
const router = new VueRouter({
  mode: 'history',
  base: process.env.BASE_URL, routes
})
export default router
```

- There is some boiler plate code beyond the scope of this class, you can carry those over for now, they are highlighted in blue.
- Our focus will be on the sections in red, which we define.
- NOTE: mode and base values can be modified but our outside the scope of this lecture

The index.js file: Importing Views

We need to first define the components a user can potentially be routed to, this can be achieved through imports:

```
import Home from './views/Home.vue'
import About from './views/About.vue'
import Books from './views/Books.vue'
```

Note that in this example the components are in a folder called views, this should make no difference, Home, About, and Books are VUE components.

The index.js file: Assigning Routes to Views

Next, we define the routes, these physically map the view components we imported.

```
import Home from './views/Home.vue'
import About from './views/About.vue'
import Books from './views/Books.vue'
```

```
Routes = [
  {
    path: '/',
    name: 'home',
    component: Home
  },
  {
    path: '/about',
    name: 'about',
    component: About
  },
  {
    path: '/books',
    name: 'books',
    component: Books
  }
]
```

Each route is a JSON object, comprised of three key value pairs:

- **path**: what the user types into the URL
- **name**: This is how we will refer to the route within App.vue
- **component**: The component that was imported in, that the user will be redirected to

Adding Router Navigation Links to App.vue

We are almost done! The last step is want to define the routes within App.vue.

```
<template>
  <div id="app">
    <header>
      <ul class="nav">
        <router-link :to="{ name: 'home' }" tag="li" exact>Home</router-link>
        <router-link :to="{ name: 'about' }" tag="li">About the Author</router-link>
        <router-link :to="{ name: 'books' }" tag="li">Related Books</router-link>
      </ul>
    </header>
    <router-view class="content"/>
  </div>
</template>
```

Again, this is some boiler plate code highlighted in blue that we can just carry over, what matters is what's in red.

Adding the router links to App.vue

This is the basic structure of a router-link

```
<router-link :to="{ name: 'home' }" tag="li" exact>Home</router-link>
```

This should match up to the name of the route specified in router.js

You can define the type of html element you want rendered as the link here.

Dynamic Routing

- Sometimes data is encapsulated within a URL path:
    - Consider the following URL: (account/**135**)
    - The value 135 could very well be and ID that is associated with a “row” of data.

- The purpose of dynamic routing is to pass these URL parameters through the application.

Dynamic Routing : index.js

We must first implement a route , where the path parameter contains a placeholder for the path variable.

```
{
  path: '/users/:id',
  name: 'user',
  component: User
}
```

Dynamic Routing : Defining the Router Links

We can now define router links

```
<tr v-for="user in users" :key="user.id">
      <td><router-link :to="{name: 'user', params: {id: user.id}}">{{user.id}}</router-link></td>
      <td>{{user.name}}</td>
</tr>
```

Template section

```
data() {
  return {
    users: [
      {
        "id": 1,
        "name": "Leanne Graham"
      },
      {
        "id":2,
        "name": "Ervin Howell"
      }
    ]
  }
```

Script section

Here we have a v-for that will iterate through every object in the users array, each time it does so it generates a new router-link with its respective id value.

Two links are generated:

- /users/1
- /users/2

Let’s Implement Routes!