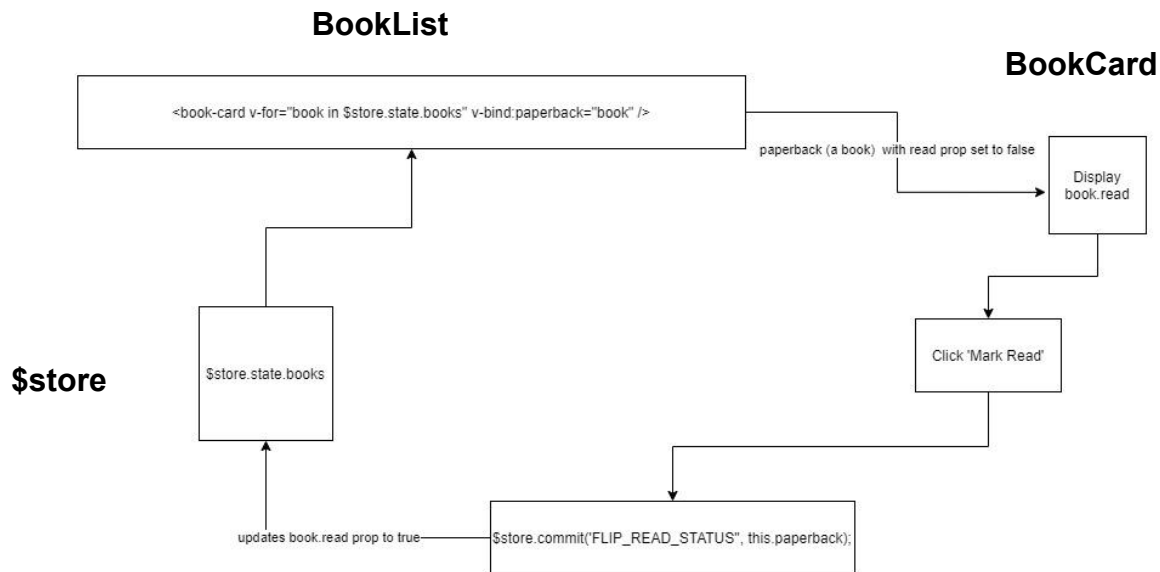


# DIAGRAM OF VUE FLOW

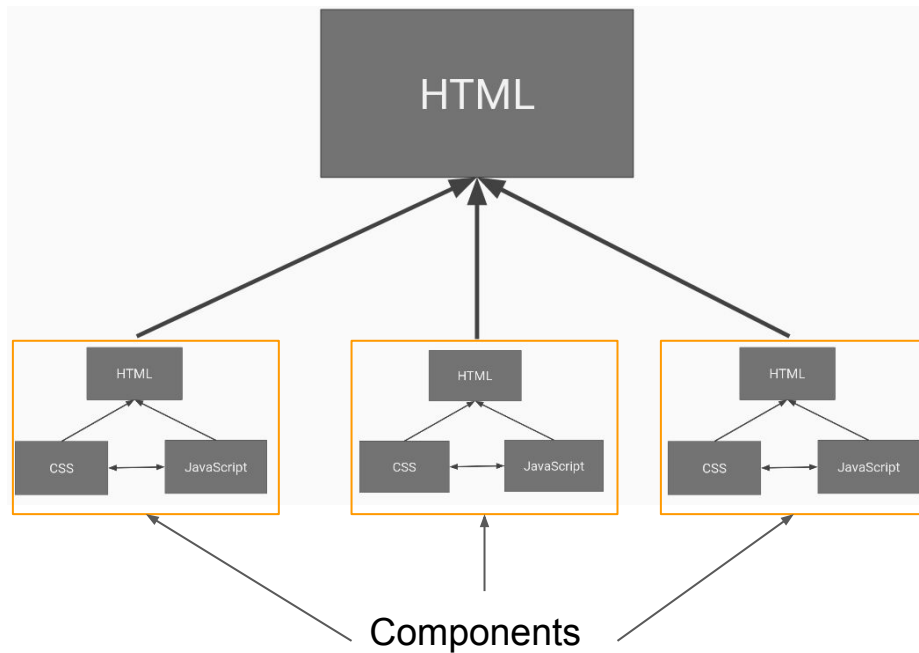
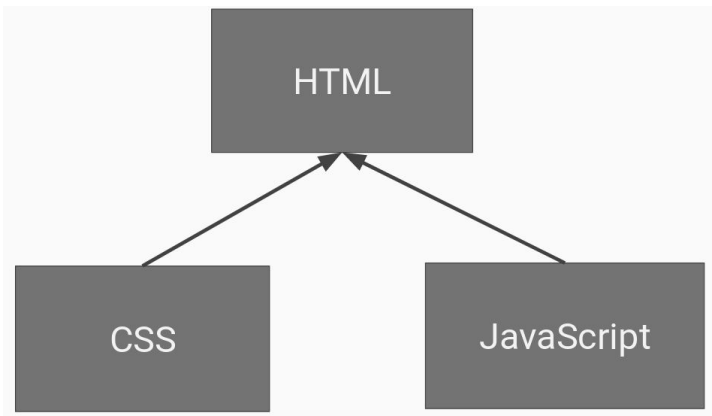
When the data in the `$store` changes it is **automatically** reflected in entire flow



VUE ROUTER

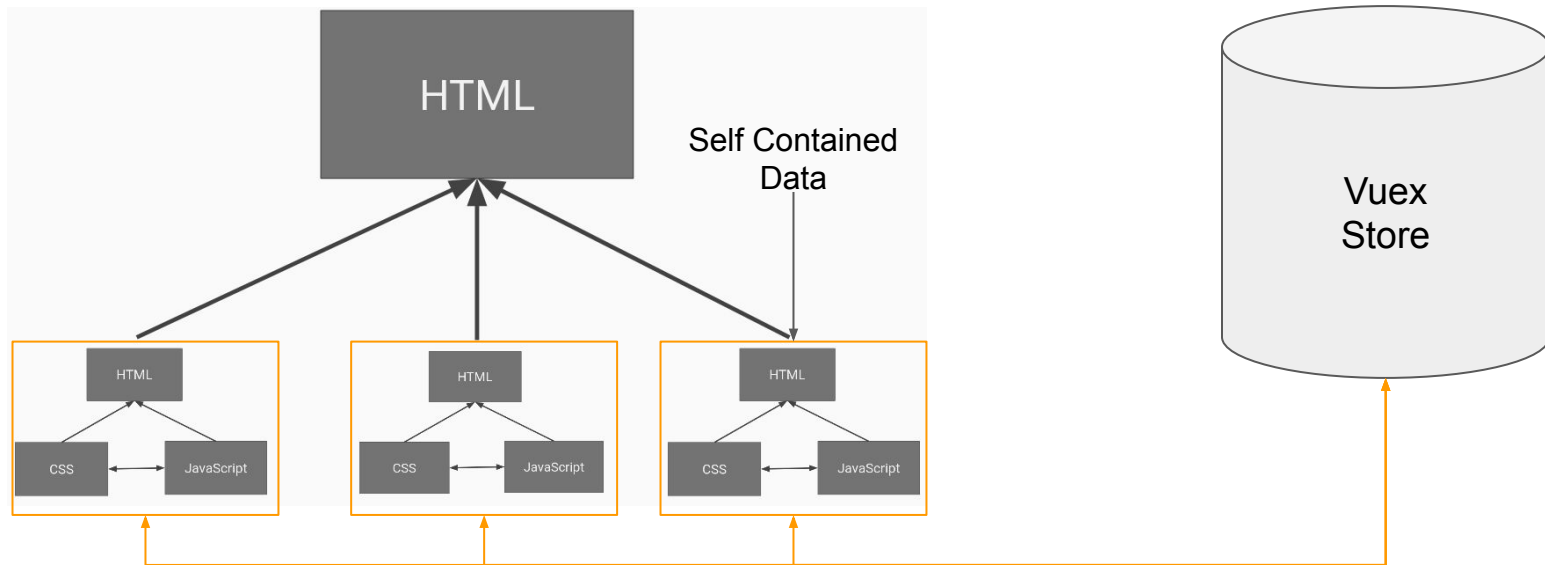
# VUE: EVOLUTION OF CONCEPTS - HTML TO VUE

Index.html & App.vue



# VUE: EVOLUTION OF CONCEPTS - VUE TO VUEX

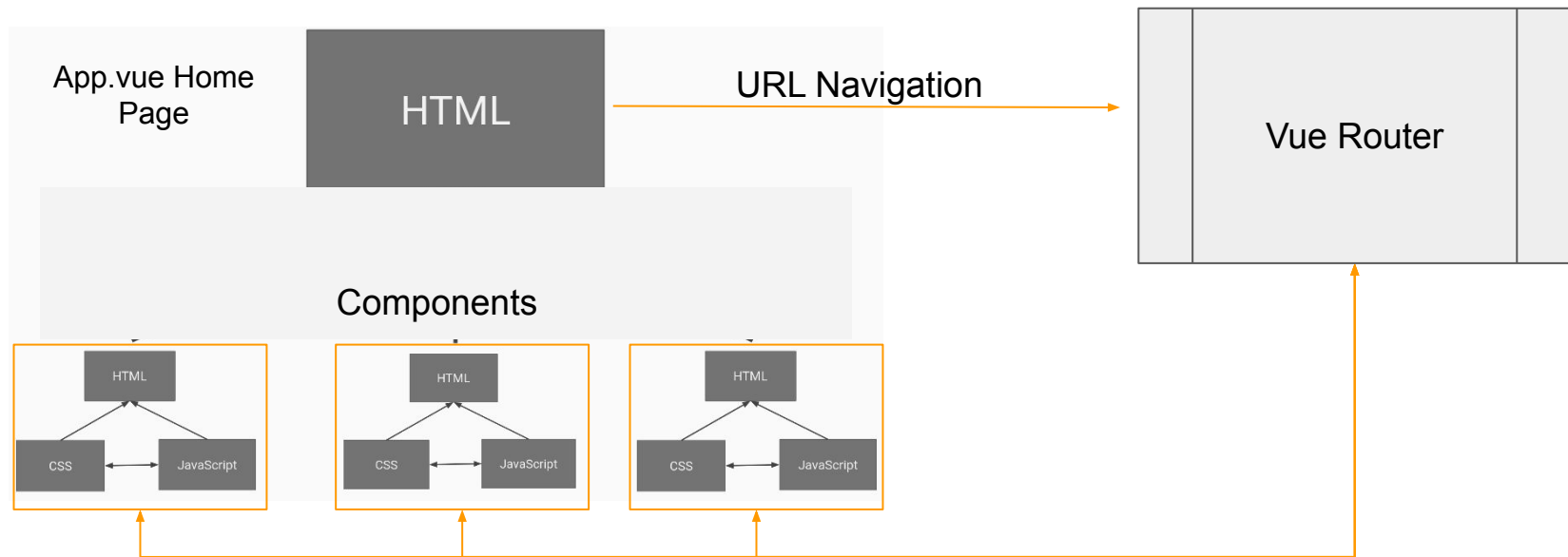
Index.html & App.vue



# VUE: EVOLUTION OF CONCEPTS - COMPONENTS AS SITE PAGES

- Single Page Applications allow you to write complete applications that do not require reloading of the full page.
- Even though we only have one physical page, we can create components that serve as virtual site pages.
- Components that serve as site pages are still on same physical page and can be loaded and unloaded on the page as needed.
- Physical pages are specified by URLs, in Single Page Applications, we use **routes** to apply this paradigm to components acting as virtual site pages.

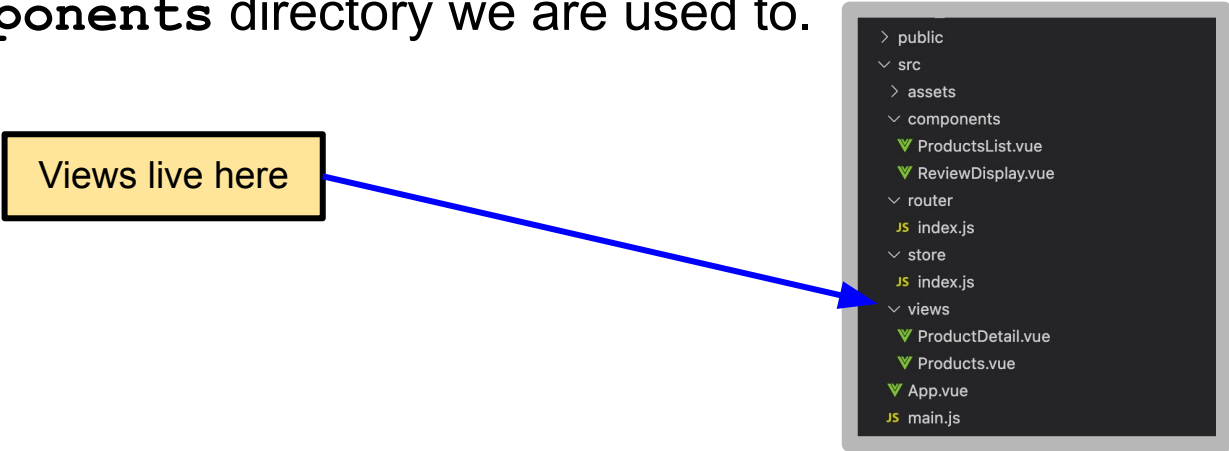
# VUE: EVOLUTION OF CONCEPTS - ROUTING TO SITE PAGES



# COMPONENTS VS. VIEWS

- **Views** are just components that serve a special purpose: acting as virtual pages.
- Difference between a View and Component is conceptual, aside from the fact that Views live in a **views** directory rather than in the **components** directory we are used to.

Views live here




```
> public
  > src
    > assets
    > components
      ▼ ProductsList.vue
      ▼ ReviewDisplay.vue
    > router
      JS index.js
    > store
      JS index.js
    > views
      ▼ ProductDetail.vue
      ▼ Products.vue
      ▼ App.vue
  JS main.js
```

# USING ROUTERVIEW

- **RouterView** is a functional component that renders components (views) for a given path.
- **RouterView** does not require view components to be in the views directory, but they should be.

The **App** component includes the **RouterView** component in its `<template>` section.

**RouterView** handles all other rendering of components which are views.




```
<template>
  <div>
    <router-view />
  </div>
</template>
```



# DEFINING ROUTES

- Routes used by **RouterView** are defined in `src/router/index.js`

The `routes` array holds objects representing routes.



**RouterView** route objects require the route `path` and the `component` being routed to.

Optionally, a route can be given a `name`. *This is best practice.*

```
import Vue from 'vue'
import VueRouter from 'vue-router'
import Products from '@/views/Products'

Vue.use(VueRouter)

const routes = [
  {
    path: '/',
    name: 'products',
    component: Products
  }
]

const router = new VueRouter({
  mode: 'history',
  base: process.env.BASE_URL,
  routes
})
```

# DYNAMIC ROUTING

- We can specify dynamic values in a router path. much like we did with `@PathVariable` in Java.
- A route in the router/index.js file can use a dynamic value by using a placeholder preceded by a :
  - `path="/product/:productId`
- Any variables in the router path will be accessible to the component that was routed to via the `$route.params` object.
  - `const product = this.$route.params.productId;`

# USING ROUTES

- The router-link tag can be used to define a link to a route in another component's `<template>` section.

```
<template>
  <div id="app">
    <nav>
      <router-link v-bind:to="{ name: 'home' }"> Home </router-link>
    </nav>
    <router-view />
  </div>
</template>
```

# ANATOMY OF A ROUTER-LINK

```
<router-link v-bind:to="{ name: 'home' }" tag="div">Home</router-link>
```

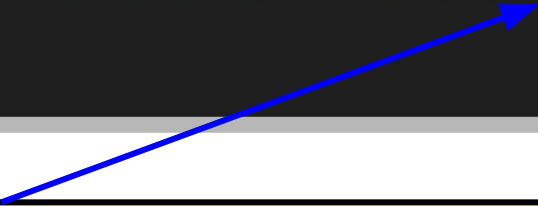
The `to` attribute can take a literal path (`to="/"`) or bind to an object with properties. This one has the **name** of the route to use as the property. Using the **name** of a route is the best practice method of specifying the route.

The `tag` attribute is optional. It allows you to specify that the router link should be rendered as a different type of tag. In this instance it will render as a `div`.

# USING ROUTE PARAMETERS WITH ROUTER-LINK

- The properties object passed in to the **router-link** tag can have a **params** property containing any dynamic router params.

```
<router-link :to="{ name: 'product-details', params: {productId: product.id} }">
  {{ product.name }}
</router-link>
```



The **productId** property in the **params** object will be used to populate the **productId** in the dynamic route (when **path="/product/:productId"**, for instance).

LET'S WORK THROUGH  
ANOTHER EXAMPLE