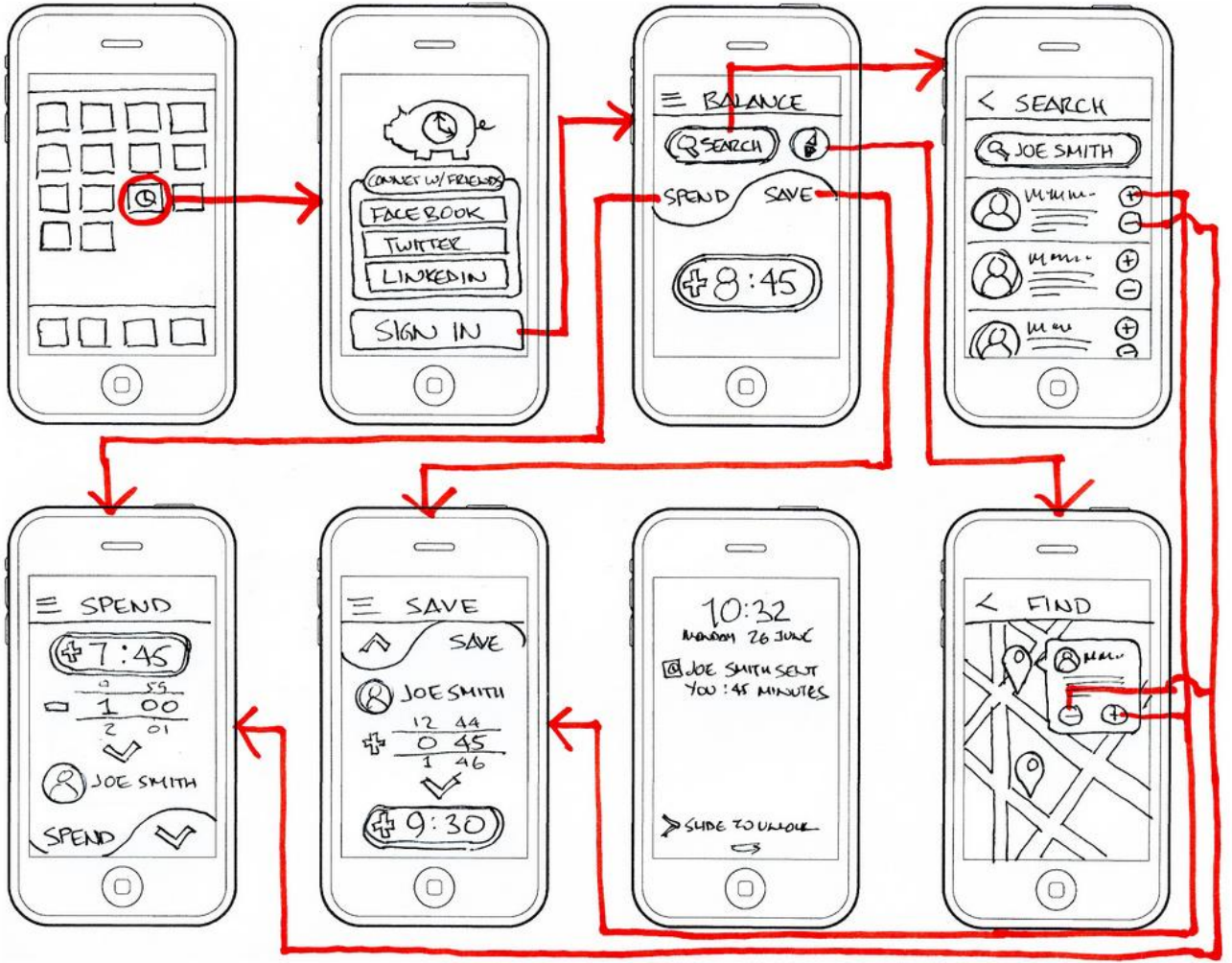
App Navigation

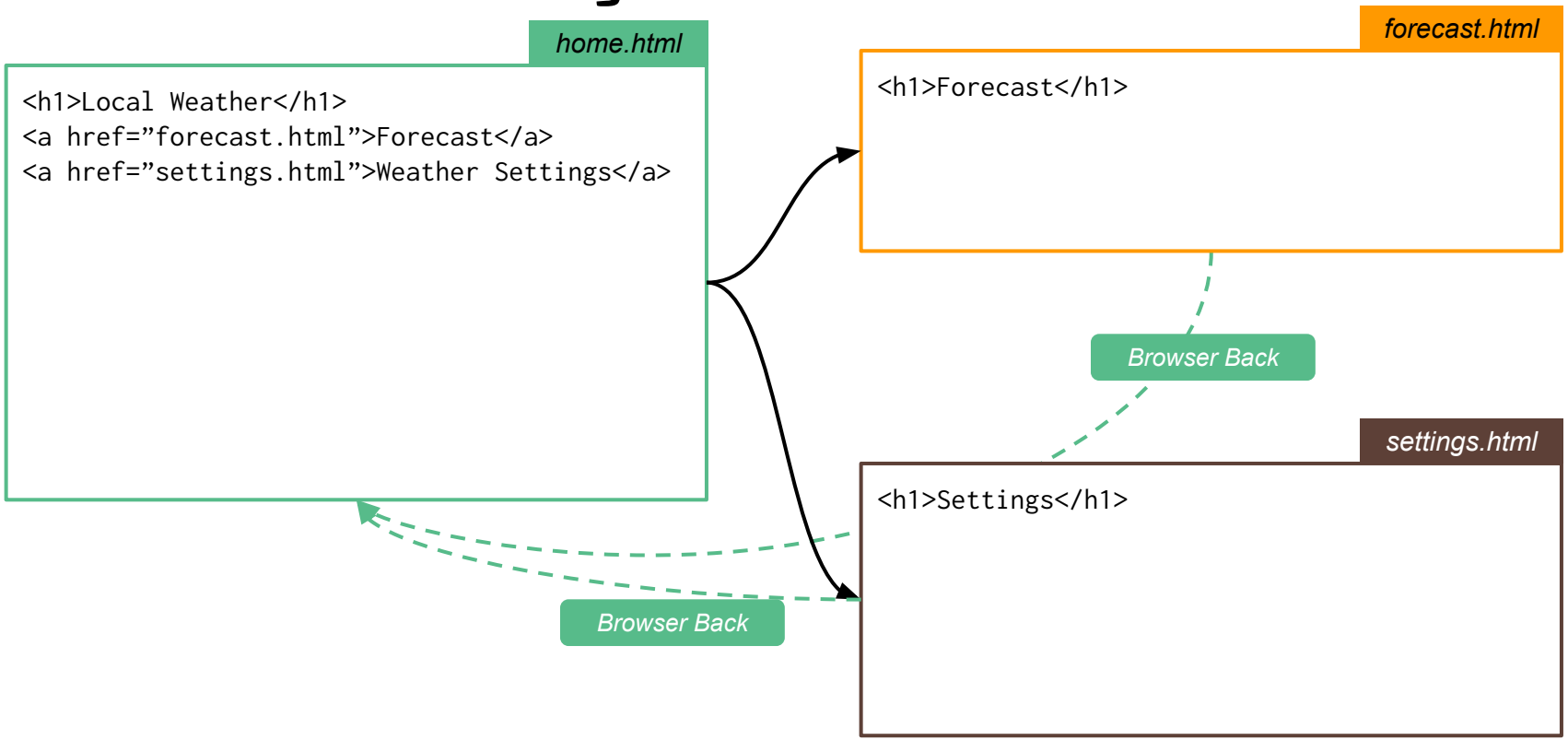
Navigate Among Multiple Views

Routing

WebApp Navigation



Traditional Web Page Solution



Legacy Web Pages

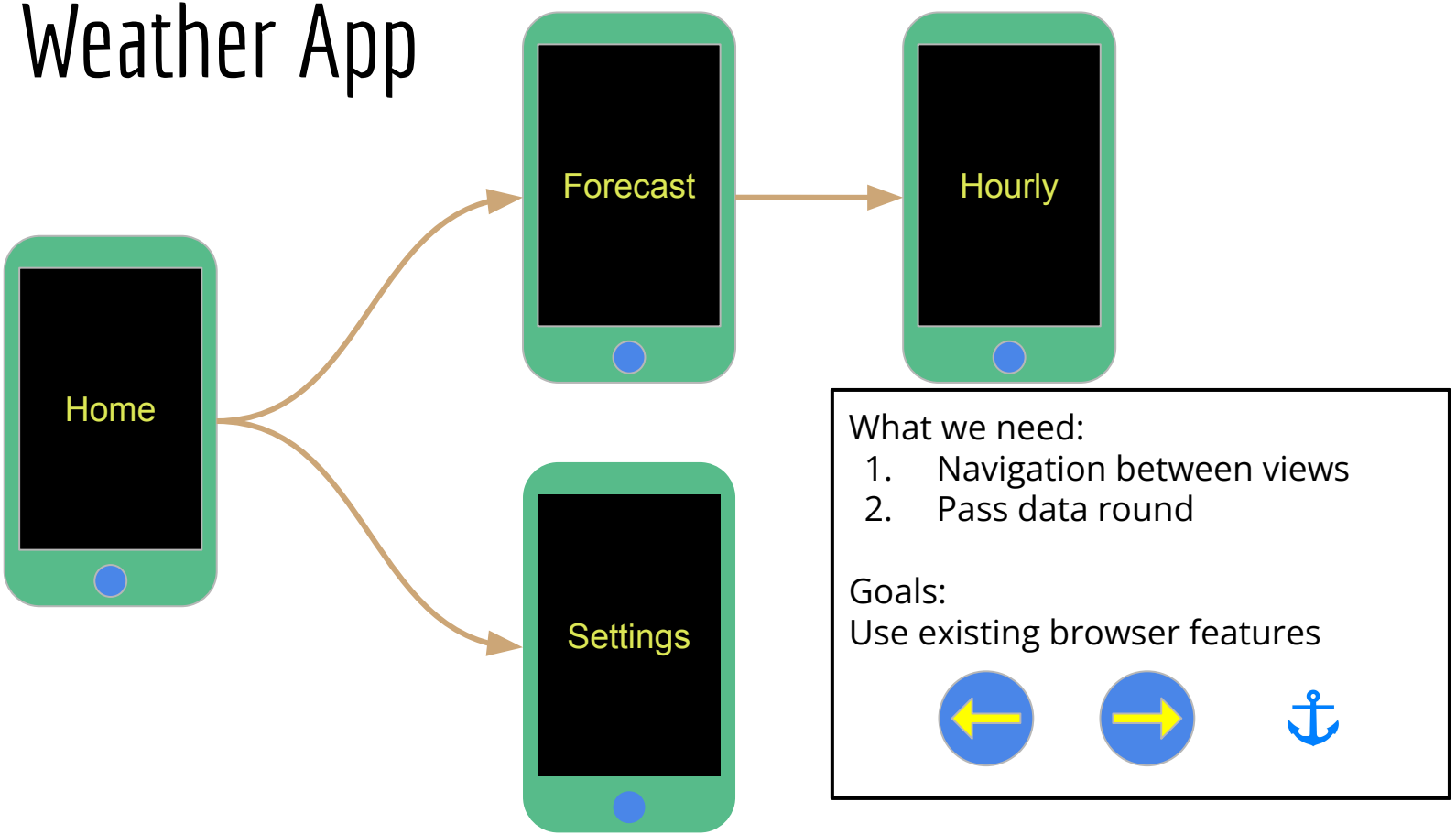
Modern Web Apps

	Legacy Web Sites	Modern Web Apps
Structure	One .html file per page	Single Page Application (SPAs)
Contents Organization	Multiple pages	Multiple web components (“views” / “screens”)
Transitions	Replace the entire page with new .html from the server	Replace only part of the page with new component

Commonality: The browser features for maintaining *navigation history*

- Back / Forward buttons
- History Stack (Show this on the browser!!!)
- Each page/component is associated with a **unique URL path**

Weather App



What we need:

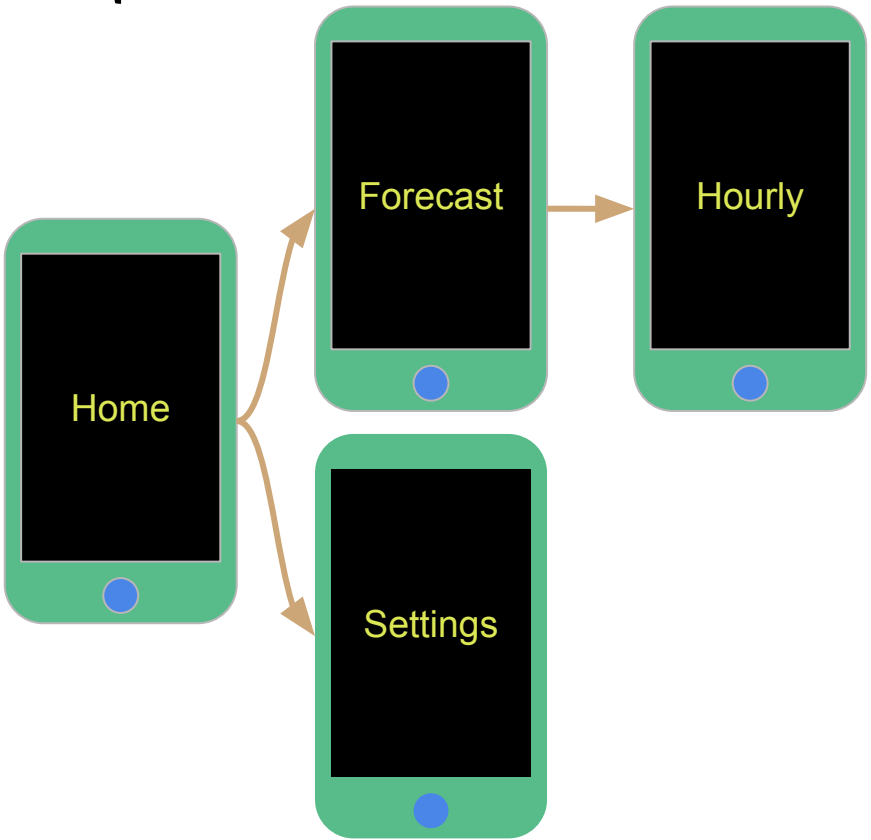
1. Navigation between views
2. Pass data round

Goals:

Use existing browser features

Vue Router 4.x

Step 1: Installation & Setup




```
npm install --save vue-router@4
# yarn add vue-router@4   # version 4.x
```

Component	URL (internal) Path
Home.vue (*landing page*)	`/`
Forecast.vue	`/forecast`
Hourly.vue	`/hourly`
Settings.vue	`/settings`

Step 2: Define Routes (in an array)

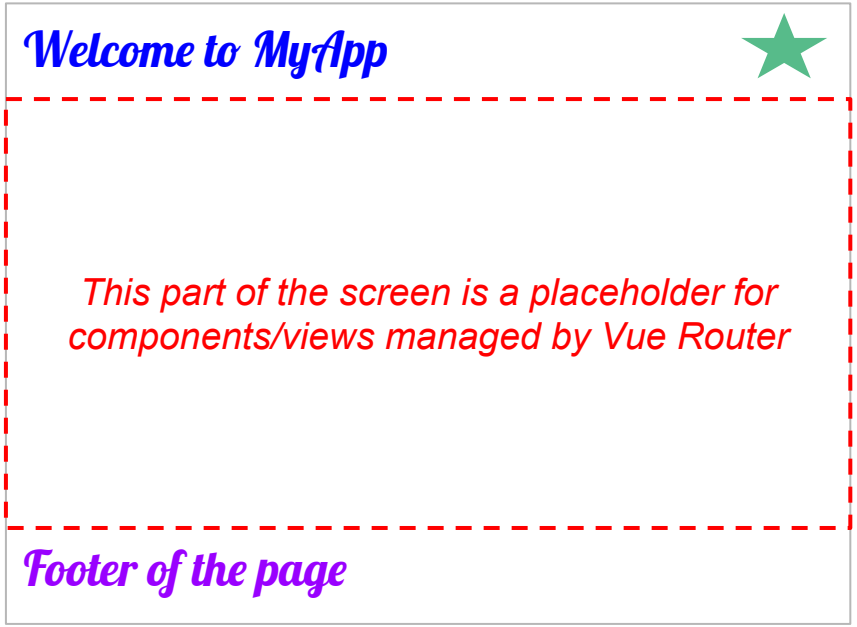
main.ts

```
import {createApp} from “vue”
import {createRouter, createWebHasHishtory} from “vue-router”
import App from “./App.vue”
import Home from “./components/Home.vue”;
import Forecast from “./components/Forecast.vue”
import Settings from “./components/Settings.vue”;

// STEP 2A: Define the routes/navigation paths
const myComponentRoutes = [
  { path: “/”, component: Home },
  { path: “/forecast”, component: Forecast },
  { path: “/settings”, component: Settings },
];

const myRouter = createRouter({routes: myComponentRoutes, /* more option */ });

// STEP 2B: Use the router with your VueJS app
createApp(App)
  .use(myRouter)
  .mount(“#app”)
```

Step 3: Include “View Container” in App.vue

App.vue

```
<template>
  <div>
    <span>Welcome to MyApp</span>
    <img src=”./yourLogo.png” />
    <router-view></router-view>
    <div>Footer of the page</div>
  </div>
</template>
<script lang=”ts”>
  // No special code needed
</script>
```



Step 4: Use Links in Components

Home.vue

```
<template>
  <h1>Home</h1>
  <a        href="sample.html">For your comparison</a>
  <RouterLink to="/forecast">Forecast</RouterLink>
  <RouterLink to="/settings">Settings</RouterLink>
</template>
```

Settings.vue

```
<template>
  <h1>Settings</h1>

</template>
```

Forecast.vue

```
<template>
  <h1>Forecast</h1>
  <RouterLink to="/hourly">Hourly</RouterLink>

</template>
```

Live Demo
(Code Sample in GitHub)

How To Transition *Programmatically*

Use Case: Hourly Forecast only for Prime Members

Forecast.vue (before)

```
<template>
  <h1>Forecast</h1>
  <RouterLink to=”/hourly”>Hourly</RouterLink>

</template>
```

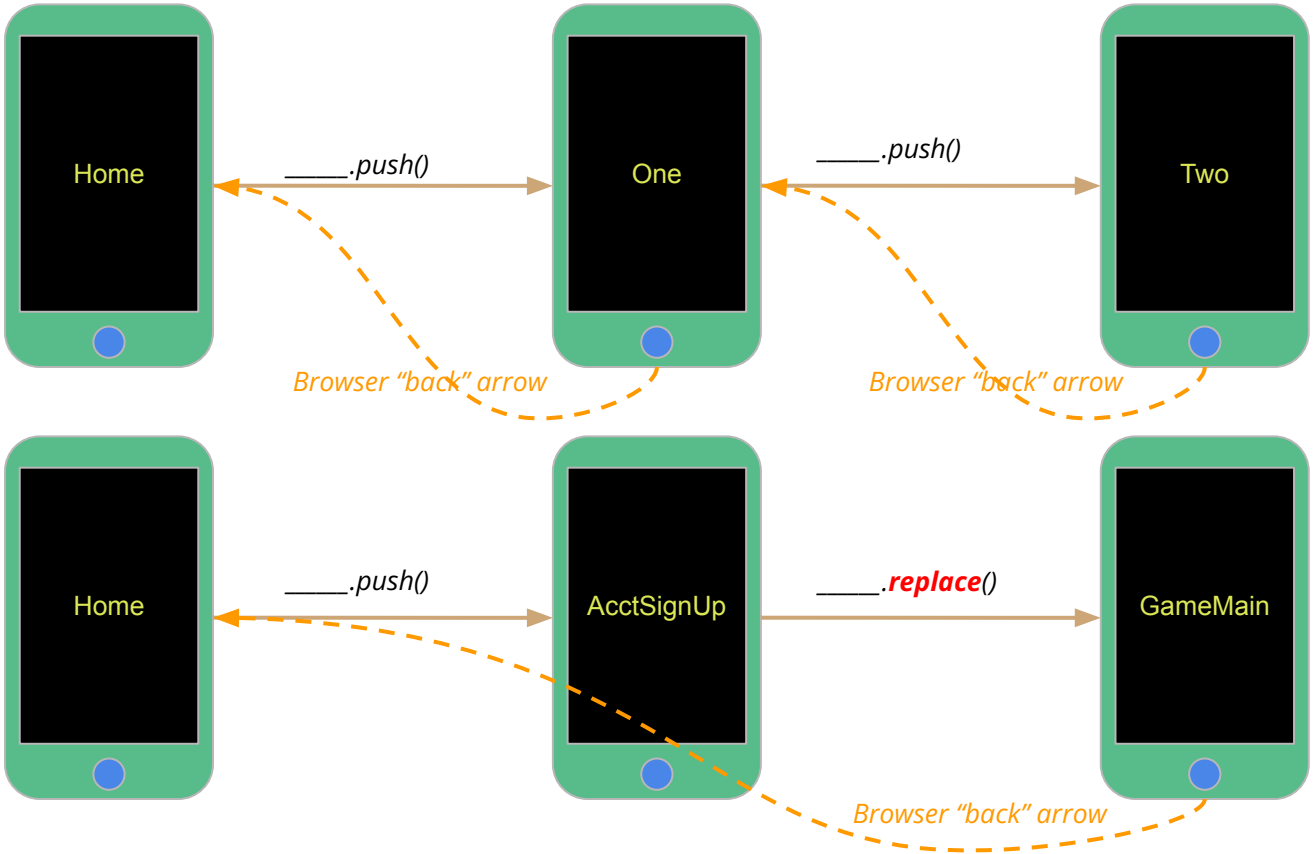
Forecast.vue (after)

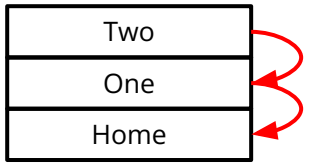
```
<template>
  <h1>Forecast</h1>
  <button @click=”canIHourly”>Prime Hourly</button>
</template>

<script setup lang=”ts”>
import { useRouter } from “vue-router”
const appNav = useRouter()

function canIHourly() {
   if (prime_membership_is_confirmed) {
     appNav.push({ path: “/hourly” })
   }
}
</script>
```

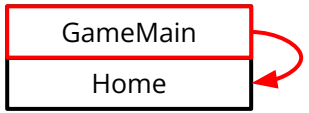
Vue Router Navigation Functions

Function	By Name	Explanation
push()	____.push({path: “/fooPath”});	
replace()	_____.replace({path: “/fooPath”});	
back()	_____.back()	
forward()	_____.forward()	
go()	_____.go(-2)	Same as calling $router.back() twice
	_____.go(1)	Same as calling $router.forward()

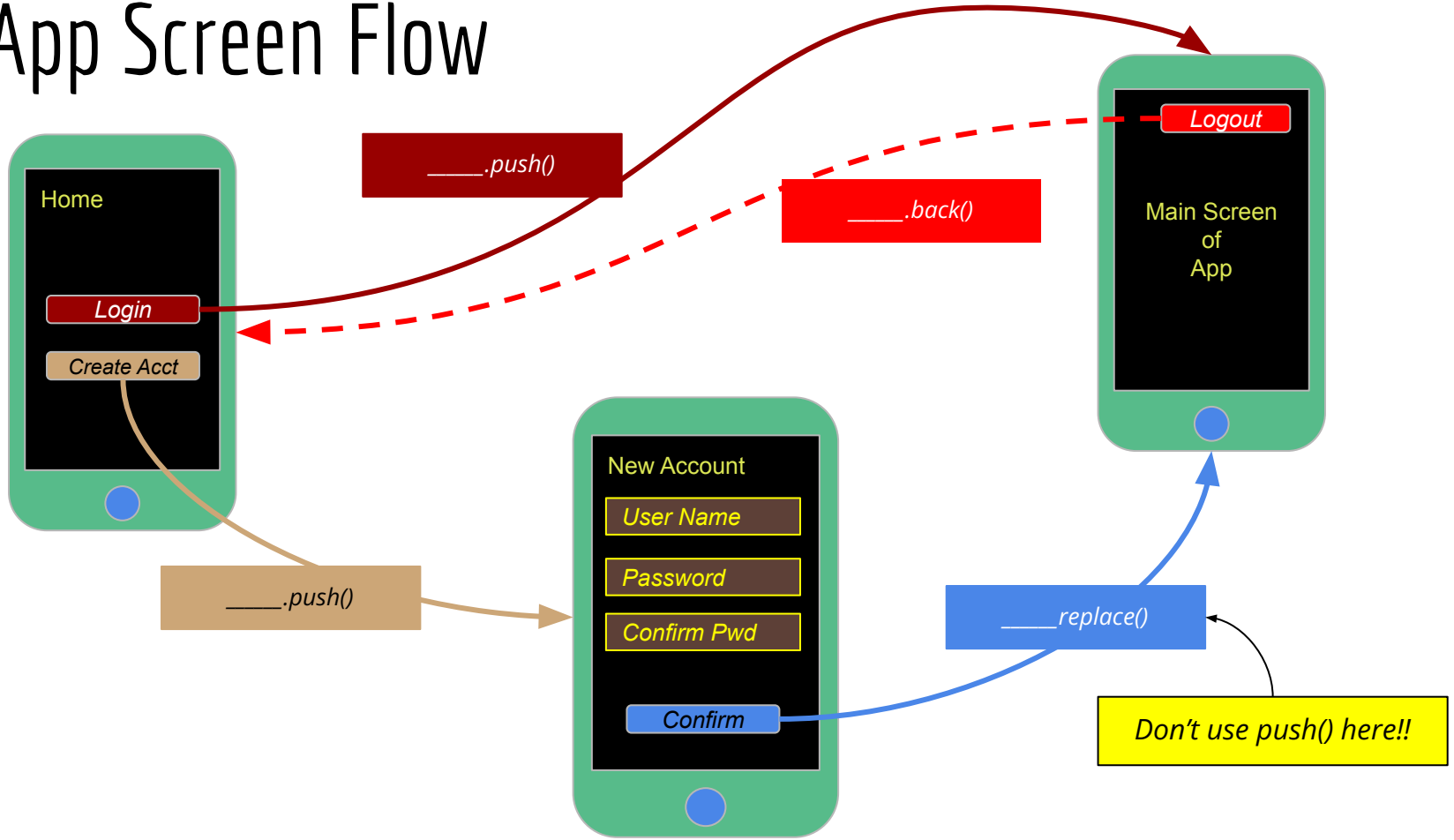
Browser History Stack: push() vs. replace()

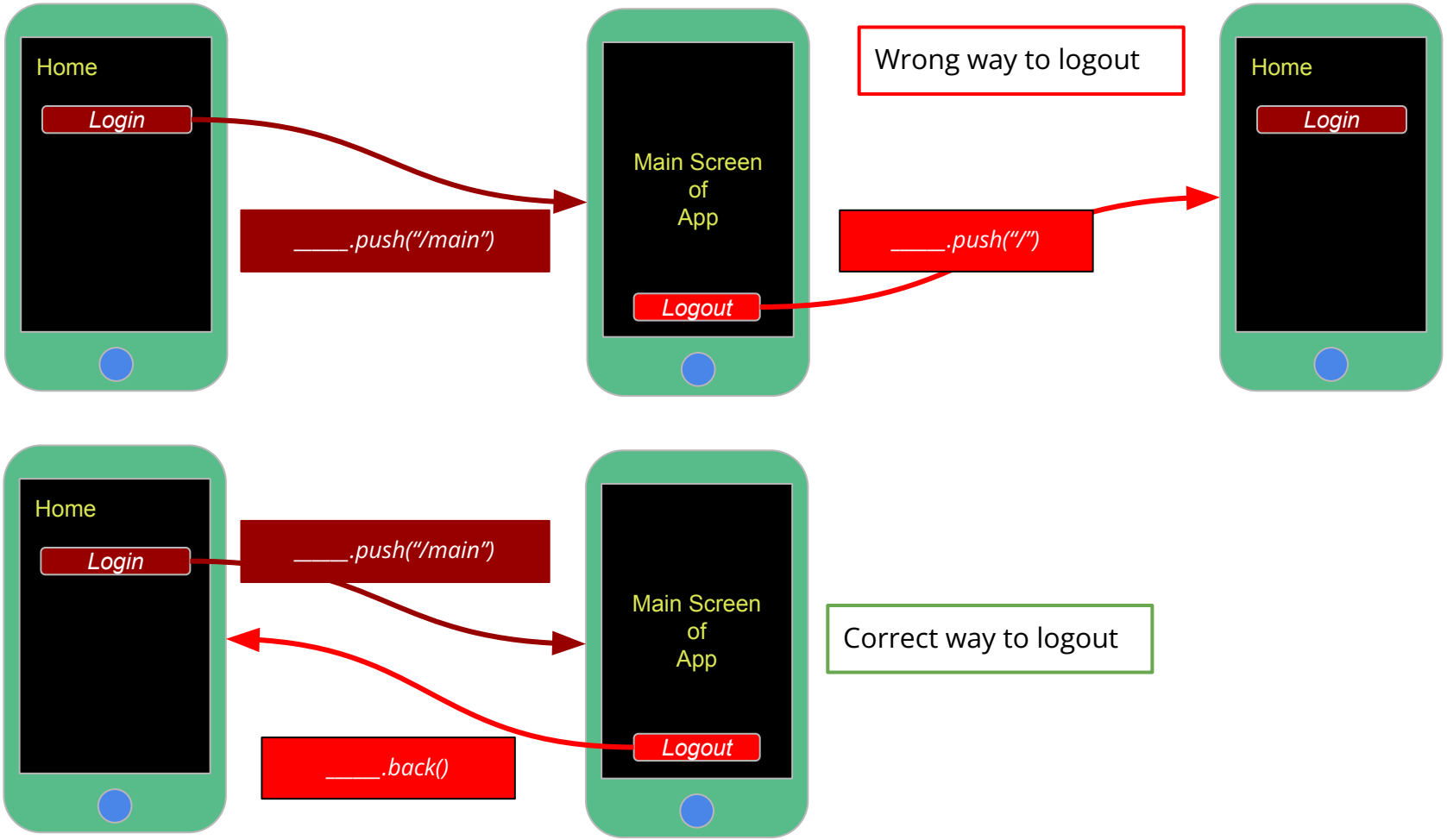






App Screen Flow

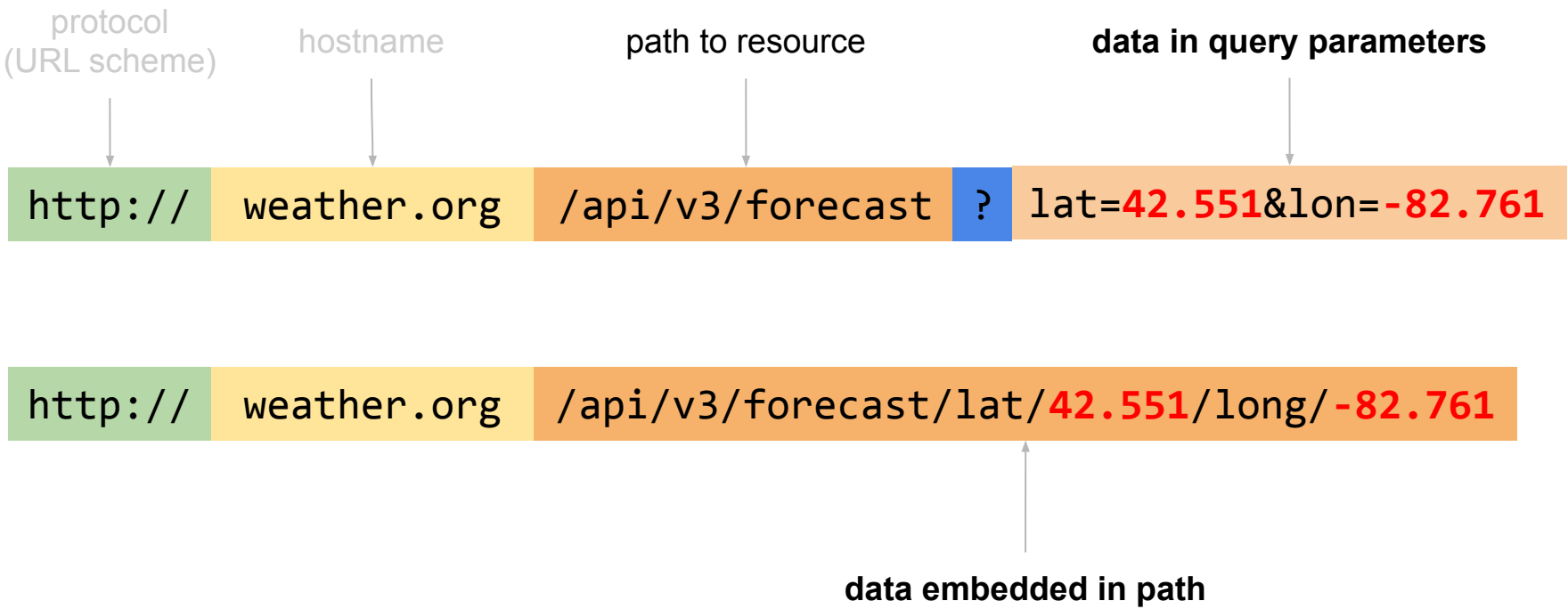



Home
Login
_____.push("/main")
Main Screen of App
Logout
Wrong way to logout
_____.push("/")
Home
Login
Home
Login
_____.push("/main")
Main Screen of App
Logout
_____.back()
Correct way to logout

Problem #2: Passing Data To Components

Limited to "string-like" data

HTTP paths and query parameters



Passing “String-like” Data To Components

Show **7-day** forecast for ZIP code **49401**

Component	Path	Name	Data Embedded in Path
ForecastByZip.vue	`/region`	**ByZip**	`/region/49401/7d` `/region/49401/next/7d`

main.ts

```
import FBZ from “./components/ForecastByZip.vue”

const myComponentRoutes = [
  {                component: Home,              path: “/”, },
  {                component: Forecast,          path: “/forecast”, },
  {                component: Settings,          path: “/settings”, },
  { name: “ByZip”, component: FBZ, props: true, path: “/region/:zipCode/:numDays” },
//{ name: “ByZip”, component: FBZ, props: true, path: “/region/:zipCode/next/:numDays” }
];
```

Sending & Receiving Props

main.ts

```
import FBZ from “./components/ForecastByZip.vue”

const myComponentRoutes = [
  /* more routes here */
  { name: “ByZip”, component: FBZ, props: true, path: “/region/:zipCode/:numDays” },
];
```

_____.vue (sender)

```
<script setup lang=”ts”>
import {useRouter} from “vue-router”
const appNav = useRouter()

function checkAtZip() {
  appNav.push({ name: “ByZip”,
                params: {
                  zipCode: “48823”, numDays: 10
                }
               })
}
</script>
```

ForecastByZip.vue (recipient)

```
<script setup lang=”ts”>
type ForecastDetailType = {
  zipCode: string;
  numDays: number
}
const props = defineProps<ForecastDetailType>()
</script>
```

Live Demo

Advanced:
CSS View Animations/Transitions

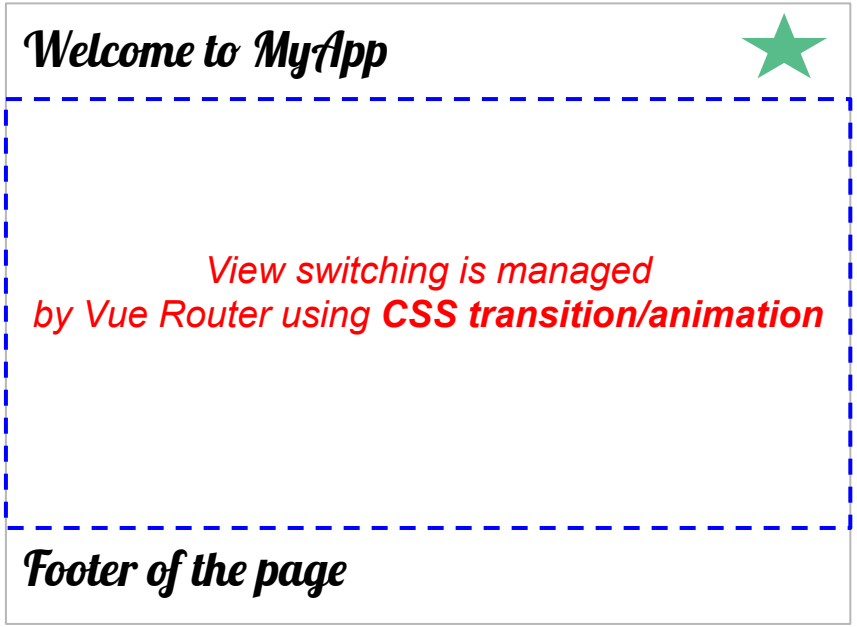
App.vue

No <transition>

```
<template>
  <div>
    <span>Welcome to MyApp</span>
    <img src=”./yourLogo.png” />
    <router-view></router-view>
    <div>Footer of the page</div>
  </div>
</template>
```

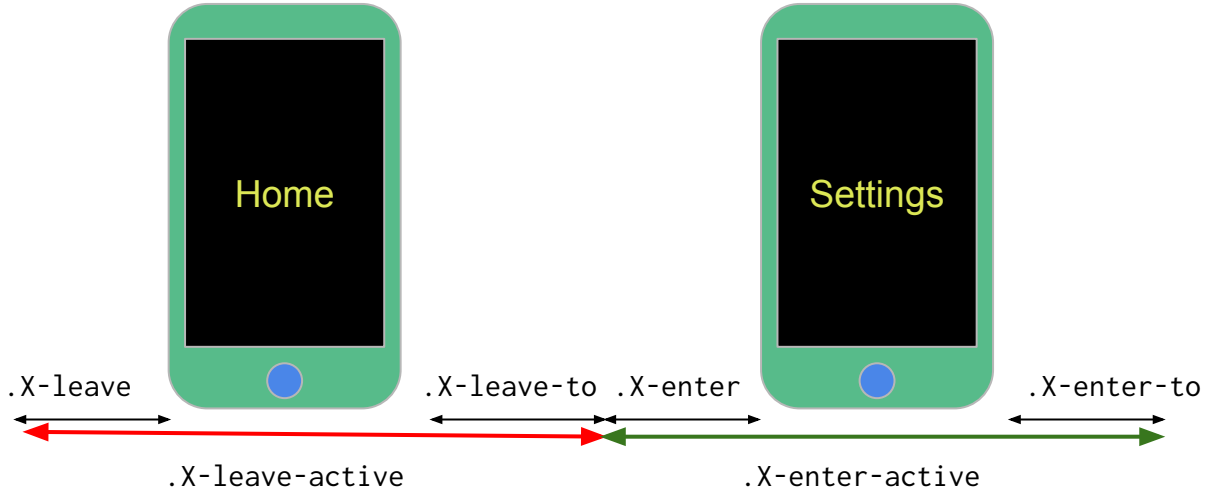
With <transition>

```
<template>
  <div>
    <span>Welcome to MyApp</span>
    <img src=”./yourLogo.png” />
    <transition>
      <router-view></router-view>
    </transition>
    <div>Footer of the page</div>
  </div>
</template>
```



Transition Classes: Page Transition Animation

- Page transition: Home.vue ⇒ Settings.vue
  - Leaving Page: Home (removed from router-view)
  - Entering Page: Settings (inserted into router-view)
  - Controlled by six CSS classes



‹transition› & Transition CSS Classes

```
<transition name="XYZ">
  <router-view></router-view>
</transition>
```

CSS for outgoing view

```
.XYZ-leave-active {
  /* General animation/transition control */
}
.XYZ-leave {
  /* CSS properties BEFORE the view appears */
}
.XYZ-leave-to {
  /* CSS properties AFTER the view appears */
}
```

CSS for incoming view

```
.XYZ-enter-active {
  /* General animation/transition control */
}
.XYZ-enter {
  /* CSS properties BEFORE the view appears */
}
.XYZ-enter-to {
  /* CSS properties AFTER the view appears */
}
```

Transition Example

```
<transition name="xampl">
  <router-view></router-view>
</transition>
```

CSS for outgoing view

```
.xampl-leave-active {
  transition-property: all;
  transition-duration: 100ms;
  transition-timing-function: ease;
}
.xampl-leave {
  transform: translateY(0%);
  background: red;
}
.xampl-leave-to {
  transform: translateY(100%);
  background: green;
}
```

Outgoing view slides DOWN (from 0% to 100%) in 100 ms

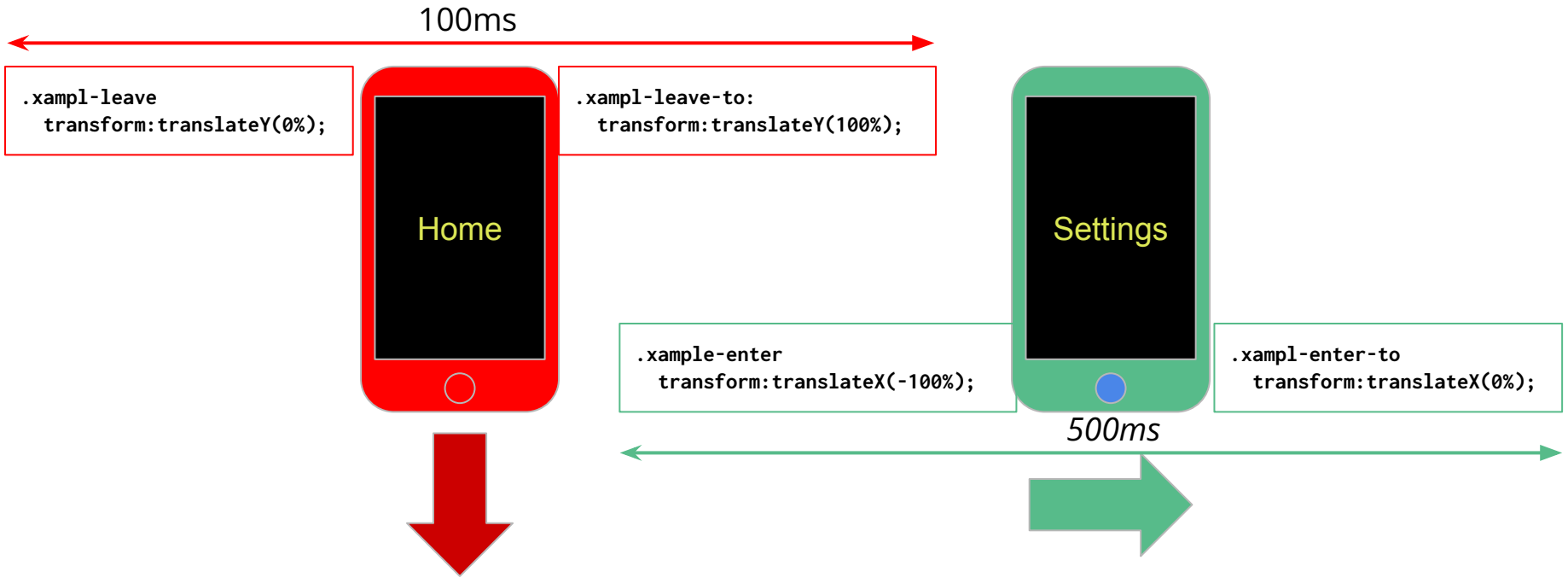
[Graphs of Timing Functions]

CSS for incoming view

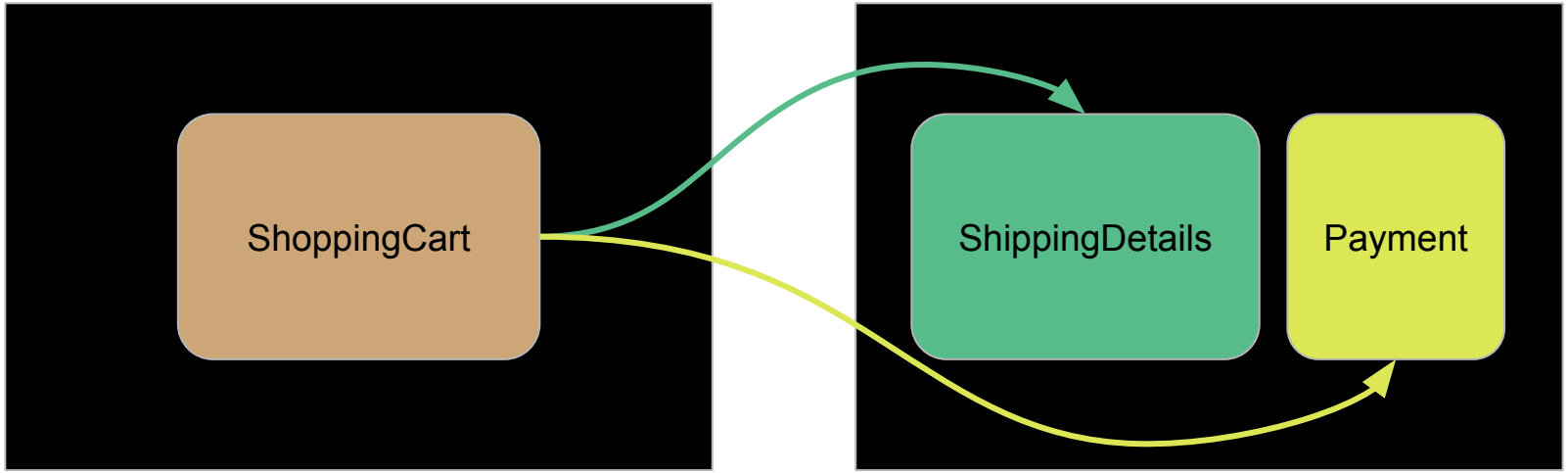
```
.xampl-enter-active {
  transition-property: all;
  transition-duration: 500ms;
  transition-timing-function: ease-out;
}
.xampl-enter {
  transform: translateX(-100%);
}
.xampl-enter-to {
  transform: translateX(0%);
}
```

Incoming view slides RIGHT (from -100% to 0%) in 500ms

Detailed Transition Sequence



Navigate into Multi-View Destination?



Navigate into Multi-View Destinations

```
<!-- UI content -->
<div>
  <router-view></router-view>
  <router-view name="side" /></router-view>
</div>
```

```
/* routing table */
{
  name: 'shipAndPay',
  path: '/____________',
  components: {
    default: ShippingDetails,
    side: Payment
  }
 }
```

Vue router navigation guards (hook functions)

Vue Router Navigation Guards

main.ts

```
const appRouter = new VueRouter({
  // vue router options go here
 });

appRouter.beforeEach(to: Route, from: Route: next: any) {
  // your code here
}

appRouter.afterEach(to: Route, from: Route: next: any) {
  // your code here
}
```

Global navigation guards: applied to the entire app

MyComponent.vue

```
<script lang=”ts”>
export default class MyComponent extends Vue {
  beforeRouteEnter(to: Route, from: Route: next: any) {
    // your code here
  }

  beforeRouteLeave(to: Route, from: Route: next: any) {
    // your code here
  }
}
</script>
```

In Component navigation guards: applied only to a specific component

Navigation Guard: Typical Use Case

```
guardFunctionName (to: Route, from: Route, next: any): void {
  if(______) {
    next();       // OK, proceed to the next view
  }
  else {
    showDialog("Can't take that transition");
    next(false);  // Don't go to the next view
  }
}
```

```
type Route = {
  name: string
  path: string,
  params: object.
  /* and more fields */
}
```

Vue Router Per Component Guards

Navigation Guard	Description	Example of Use Cases
`beforeRouteEnter()`	Called before Vue Router navigates **into** this component	● Keep statistics of how users enter a specific component ● Prevent users from entering a specific component based on some conditions
`beforeRouteLeave()`	Called before Vue Router navigate **away** from this component	● Warn the user to save data when changes have been made ● Undo any actions performed in beforeRouteEnter()

Navigation Guard Functions Registration

main.ts

```
import Component from “vue-class-component”
Component.registerHooks([
  “beforeRouteEnter”,
  “beforeRouteLeave”
]);

import Vue from “vue”
import VueRouter from “vue-router”
Vue.use(VueRouter);

/* more setup code here */
```

- *Registration must be performed BEFORE importing “vue”*
- *Without registration the guard functions will not be called*

Sample Navigation