Vue.js 2 for Real World

and Firebase

by Acoshift

Project

- Twitty (Twitter)
  - Sign in/Sign out
  - Profile / User Profile
  - Timeline

Course Outline

S1: Intro

1. What is Vue.js ?
2. Basic ES6
3. Reactive Programming
4. Example Simple Web
5. Web Application
6. Vue Project

S2: Components

1. What is Component ?
2. Directives
3. Life Cycle
4. Working with DOM
5. Data Binding
6. Event

S3: Router

1. What is Router ?
2. Nested Route
3. Custom Path
4. Navigation Guards
5. Route Meta Fields
6. Position
7. Navigation

S4: Services

1. Services
    1. Object/Class
    2. Vue Instance
2. Filter
3. Mixin

S5: State Management

1. Reactive Object
2. Rxjs + VueRx
3. Vuex

S6: Production

1. Environment
2. Makefile
3. Deploy to Firebase
4. Deploy to Docker

Section 1

— Intro —

What is Vue.js ?

JavaScript Framework

Pure JavaScript

```
<p id="data"></p>

const data =
document.getElementById("data")

data.innerHTML = 'some text'
```

With Framework

```
<p>{{ data }}</p>

this.data = 'some text'
```

Basic ES6

ES6

- let, const
- arrow function
- class
- module
- destructuring
- default, rest, spread

Reactive Programming

Reactive ?

Imperative

```
a := b + c
```

Imperative

```
b, c := 1, 2
a := b + c
b := 10
a == ?
```

Reactive ?

Reactive Programming

Data Flow

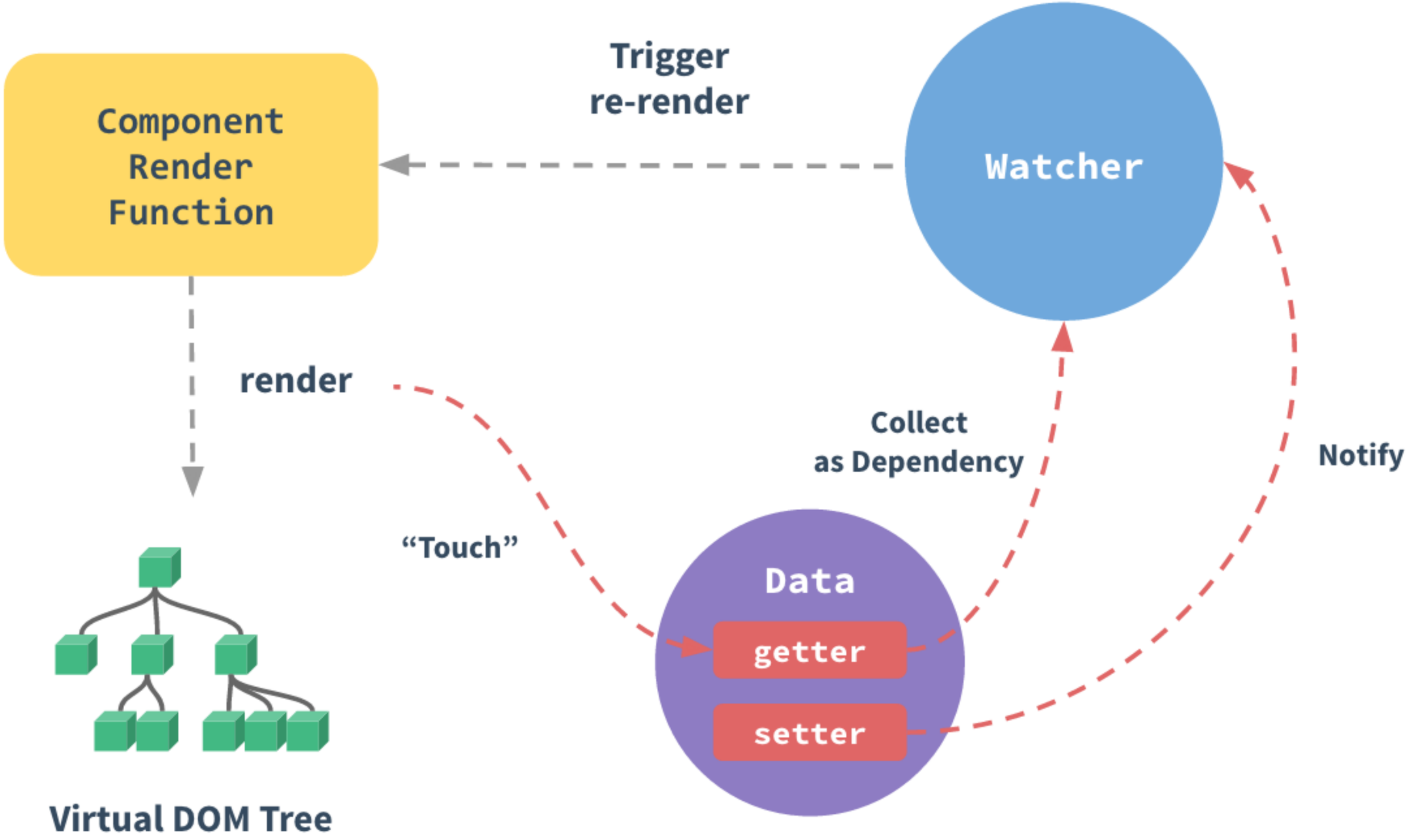
+

Propagation of Change

Propagation of Change in JavaScript

Change Detection

```
function reactiveSetter (property, value) {
  this.$data[property] = value
  console.log(`${property} change to ${value}`)
}

function reactiveGetter (property) {
  console.log(`get ${property}`)
  return this.$data[property]
}
```


Component
Render
Function
Trigger
re-render
Watcher
render
Collect
as Dependency
Notify
"Touch"
Data
getter
setter
Virtual DOM Tree

Computed

Example Simple Web

Web Site
VS
Web Application

Vue Project

Homework

- Create Vue.js project using vue-cli

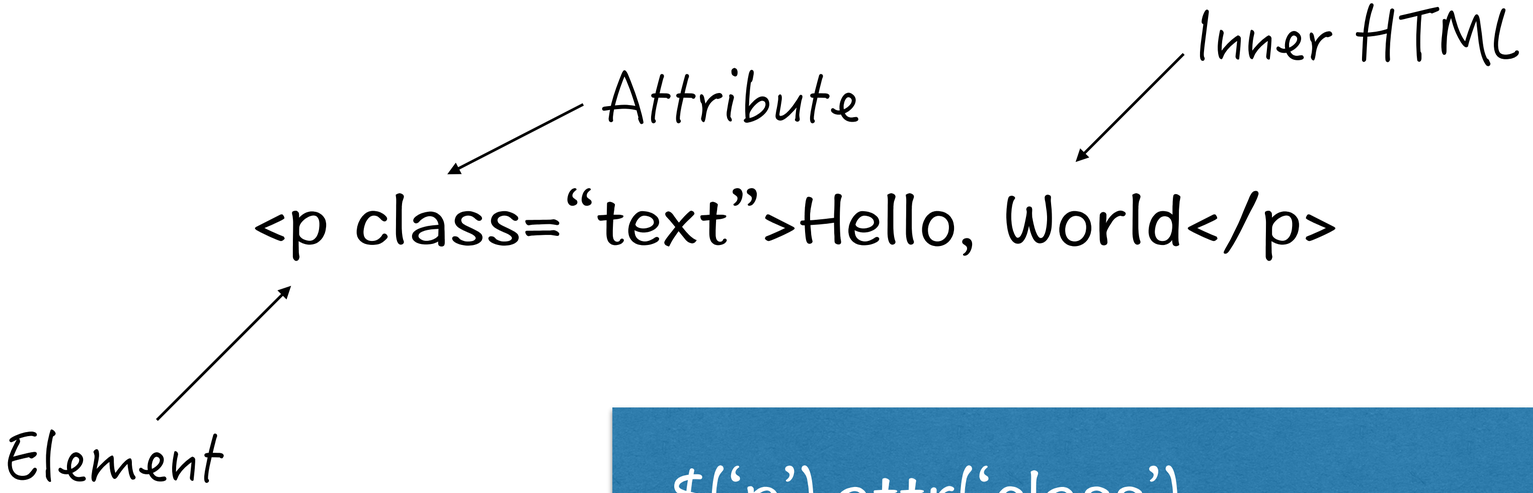
Section 2

— Components —

What is Component ?

- HTML Element
- Encapsulate
- Reusable

HTML




```
$('p').attr('class')
text
```

Component


```
<!— Hello Component —>
<div>
  Hi, {{ name }}<br>
  <slot></slot>
</div>
```

Registration

Global

```
Vue.component('hello', {
  // options
})
```

Local

```
const Hello = {
  // options
}

new Vue({
  // ...
  components: {
    Hello
  }
})
```

Hello.vue

```
<template>
  <div>
    Hi, {{ name }}<br>
    <slot></slot>
  </div>
</template>
```

```
<style scoped>
  div {
    color: green;
  }
</style>
```

```
<script>
export default {
  props: ['name']
}
</script>
```

Component name

```
new Vue({
  components: {
    'NavBar': NavBar,
    'navBar': NavBar,
    'nav-bar': NavBar
  }
})
```

```
<NavBar></NavBar>
<navBar></navBar>
<nav-bar><nav-bar>
```

Directives

- v-model
- v-text, v-html
- v-bind
- v-on
- v-if, v-else, v-else-if, v-show
- v-for

Component Structure

- template
- data
- props
- components

Component Structure

```
import Comp1 from './Comp1'

export default {
  data () {
    return {
      data1: '',
      data2: 0
    }
  },
  props: ['prop1', 'prop2'],
  components: {
    Comp1
  }
}
```

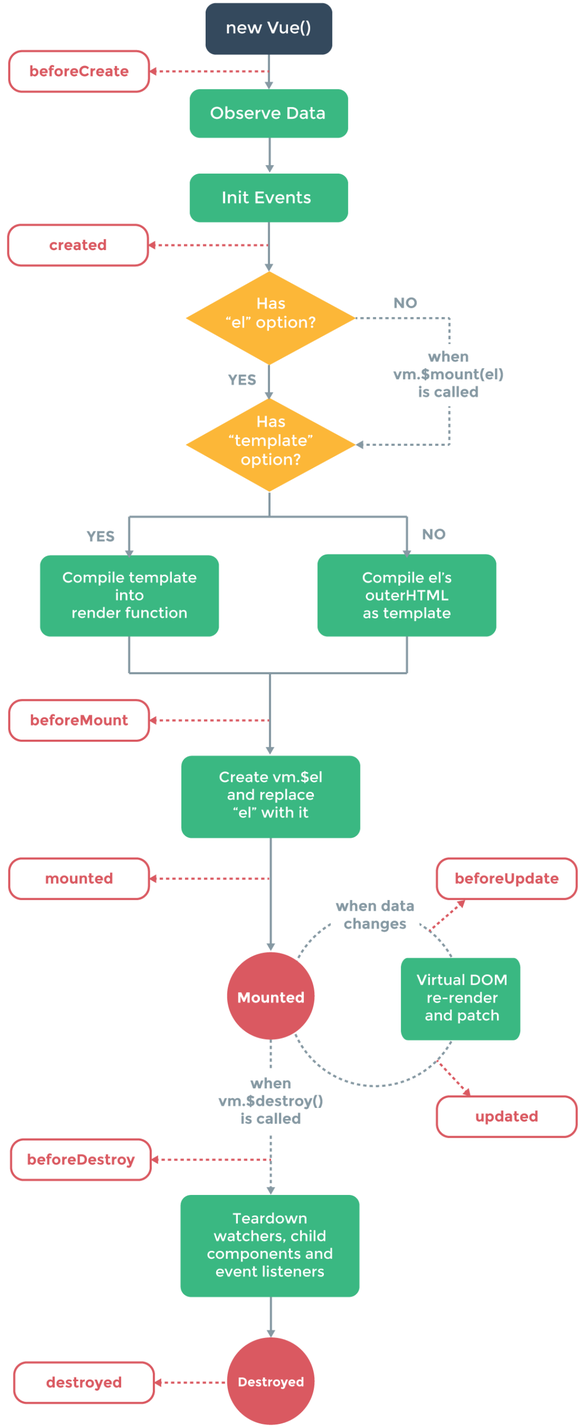
Vue.nextTick

- Call after next DOM update cycle
- Wait for DOM Update

```
this.rating = 5
// DOM not updated yet
this.$nextTick(() => {
  // DOM updated
  $(this.$refs.rating).rating()
})
```

Component Lifecycle

- beforeCreate [SSR]
- created [SSR]
- beforeMount
- mounted
- beforeUpdate
- updated
- beforeDestroy
- destroyed



Lifecycle: beforeCreate

- Called before data observation, event/watcher setup
- Use case
  - Setup
  - Tracking

Lifecycle: created

- Called after created
- Element not mounted yet (can not use $el, $refs)
- Use case
  - Get async data

Lifecycle: beforeMount

- Called before mount for first time
- Element not mounted yet (can not use $el, $refs)

Lifecycle: mounted

- Called after mount for first time
- Can use $el, $refs
- Element in DOM

Lifecycle: mounted

```
mounted () {
  console.log('before nextTick parent isMounted: ' + this.$el.__vue__.$parent._isMounted)
  this.$nextTick(() => {
    console.log('after nextTick parent isMounted: ' + this.$el.__vue__.$parent._isMounted)
  })
}
```

before nextTick parent isMounted: false

after nextTick parent isMounted: true

Lifecycle: beforeUpdate

- Called when data changed, before VDOM re-render/patched
- Change data here won't trigger another re-render

Lifecycle: updated

- Called after VDOM re-render/patched
- DO NOT change state (infinite loop)
- Use case
  - Call jQuery

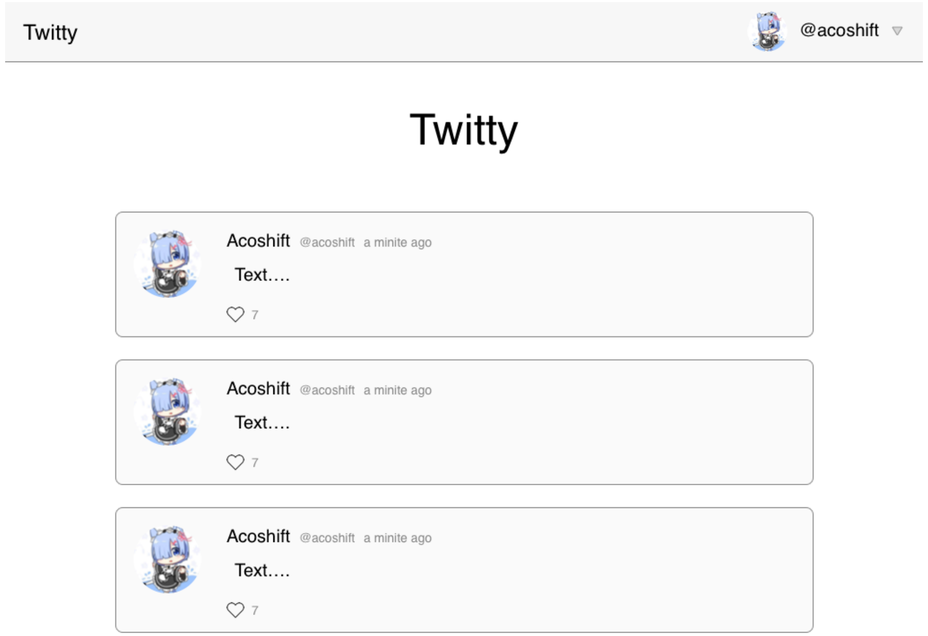
Lifecycle: beforeDestroy

- Called before instance is destroyed
- Use case
  - Cleanup listener/callback/subscriber

Lifecycle: destroyed

- Called after instance has been destroyed
- All child have also been destroyed

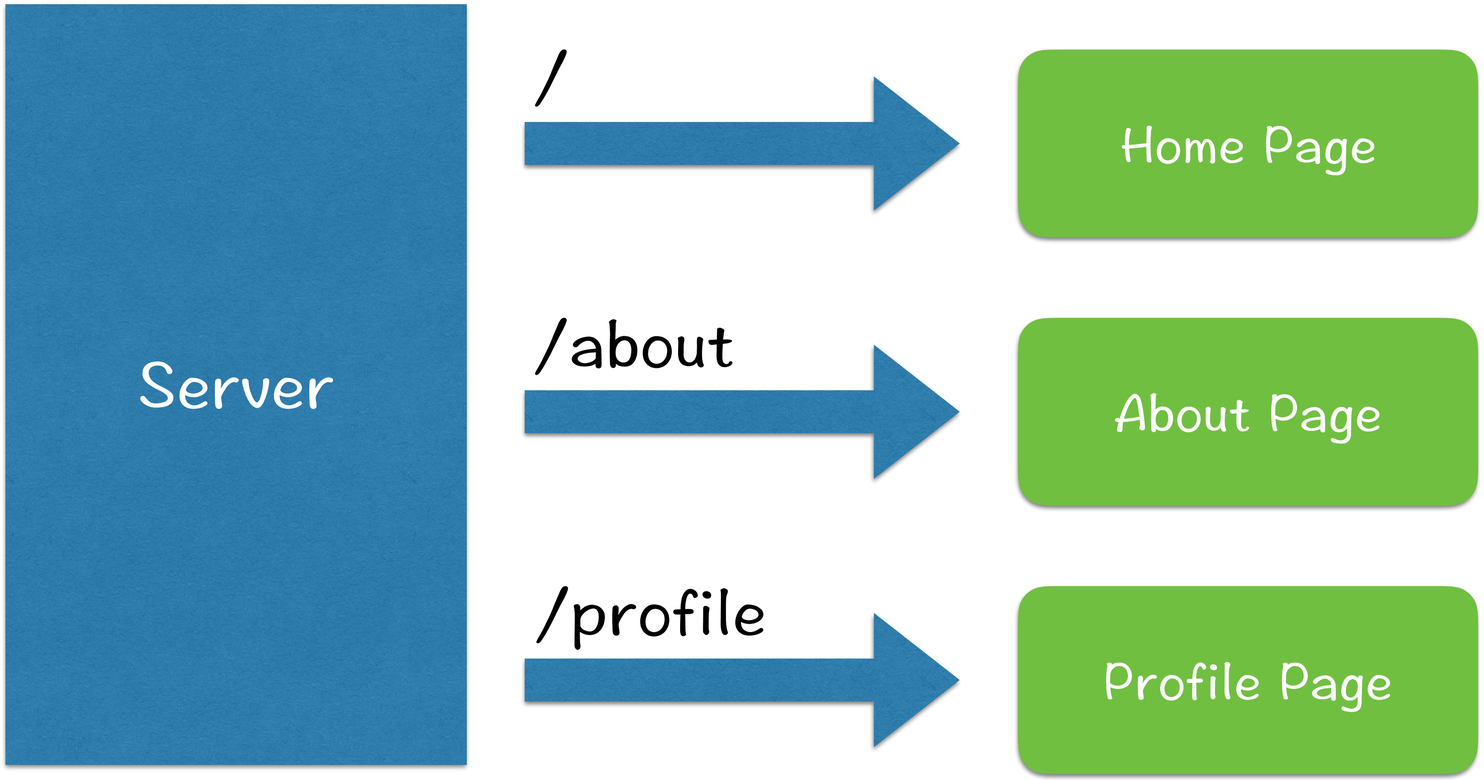
Homework


Twitty
@acoshift
Twitty
Acoshift @acoshift a minite ago
Text....
7
Acoshift @acoshift a minite ago
Text....
7
Acoshift @acoshift a minite ago
Text....
7

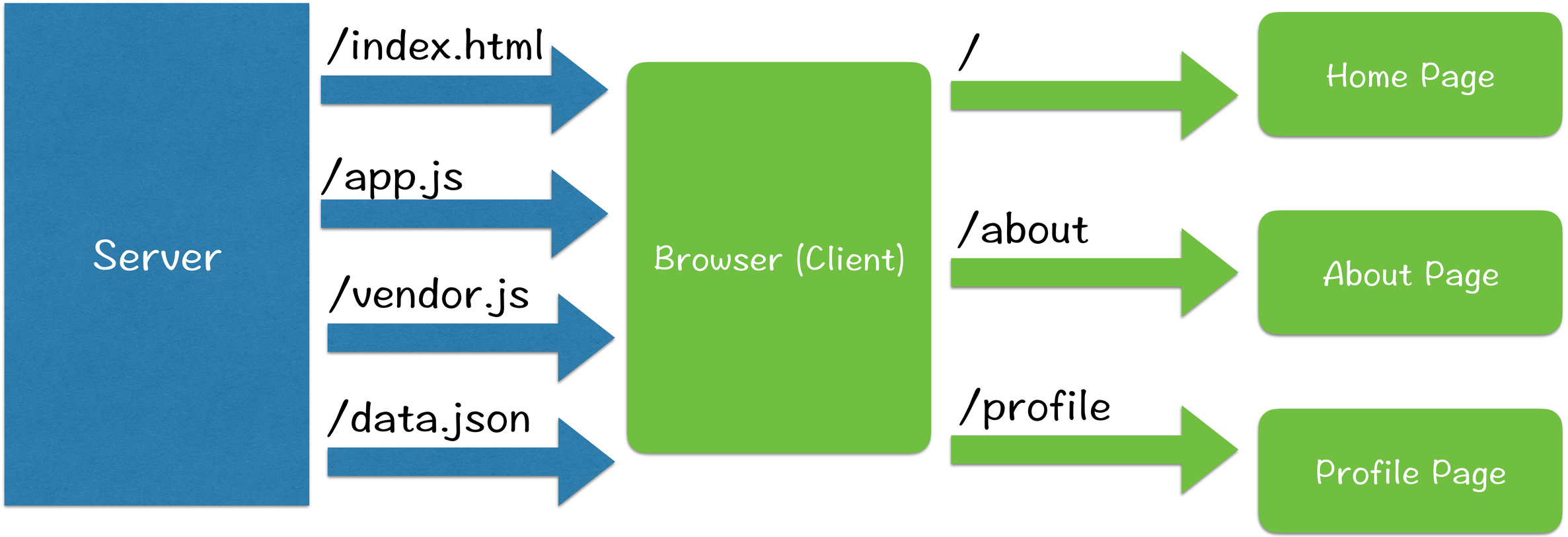
Section 3

— Router —

What is Router ?



What is Router ?



Create Router

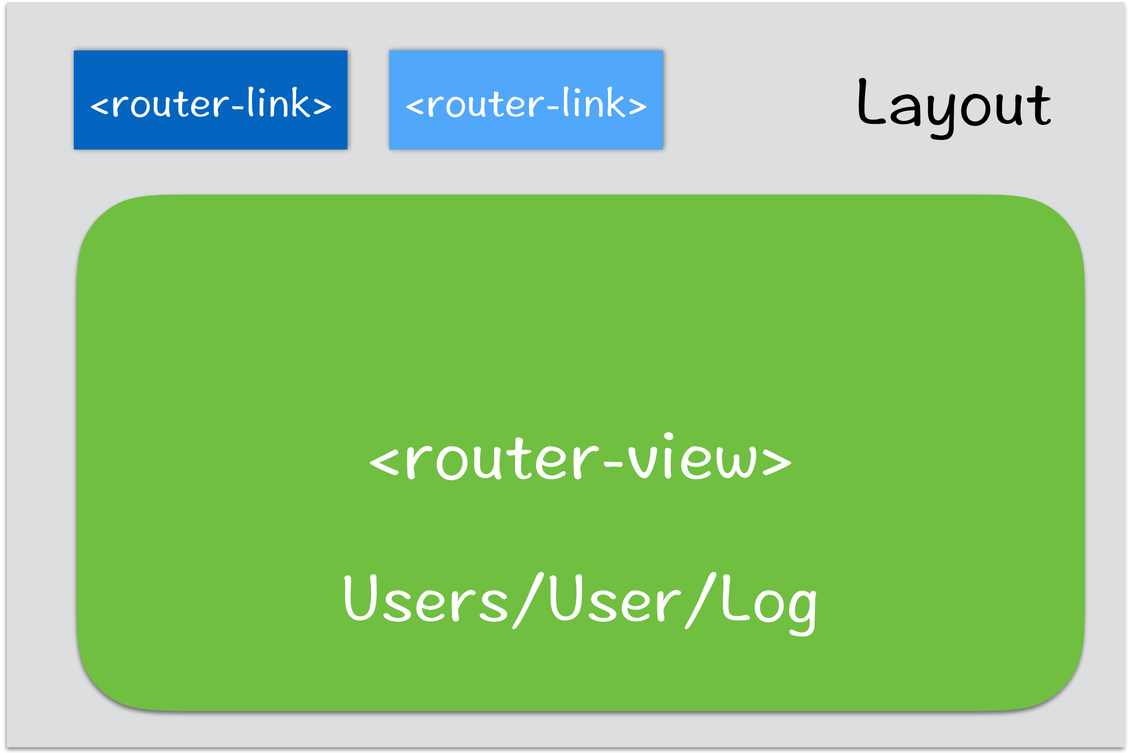
```
import VueRouter from 'vue-router'

Vue.use(VueRouter)

const router = new VueRouter({
  mode: 'history',
  routes: [
    { path: '/', component: Home },
    { path: '/about', component: About },
    { path: '*', redirect: '/' }
  ]
})

new Vue({
  router,
  ...
})
```

Mode

- history => /about
- hash => /#/about
- abstract => /

Nested Route

```
{
  path: '/dashboard',
  component: Layout,
  children: [
    { path: 'users', component: Users },
    { path: 'user/:id', component: User },
    { path: 'log', component: Log }
  ]
}
```



Custom Path

```
{
  path: '/dashboard',
  component: Layout,
  children: [
    { path: 'users', component: Users },
    { path: 'user/:id', component: User },
    { path: '/log', component: Log }
  ]
}
```

- /dashboard/users
- /dashboard/user/:id
- /log

Navigation Guards

- Global
  - beforeEach
  - afterEach
- Pre-Route
  - beforeEnter
- In-Component
  - beforeRouteEnter
  - afterRouteLeave

(Global) beforeEach

```
const router = new VueRouter({ ... })



router.beforeEach((to, from, next) => {
  // ...
})
```

(Global) afterEach

```
const router = new VueRouter({ ... })



router.afterEach((to, from) => {
  // ...
})
```

(Pre-Route) beforeEnter

```
const router = new VueRouter({
  routes: [
    {
      path: '/foo',
      component: Foo,
      beforeEnter: (to, from, next) => {
        // ...
      }
    }
  ]
})
```

(In-Comp) beforeRouteEnter

```
<script>
  export default {
    beforeRouteEnter (to, from, next) {
      // ...
      // Can not use `this`
    }
  }
</script>
```

(In-Comp) beforeRouteLeave

```
<script>
  export default {
    beforeRouteLeave (to, from, next) {
      // ...
    }
  }
</script>
```

Route Meta Fields

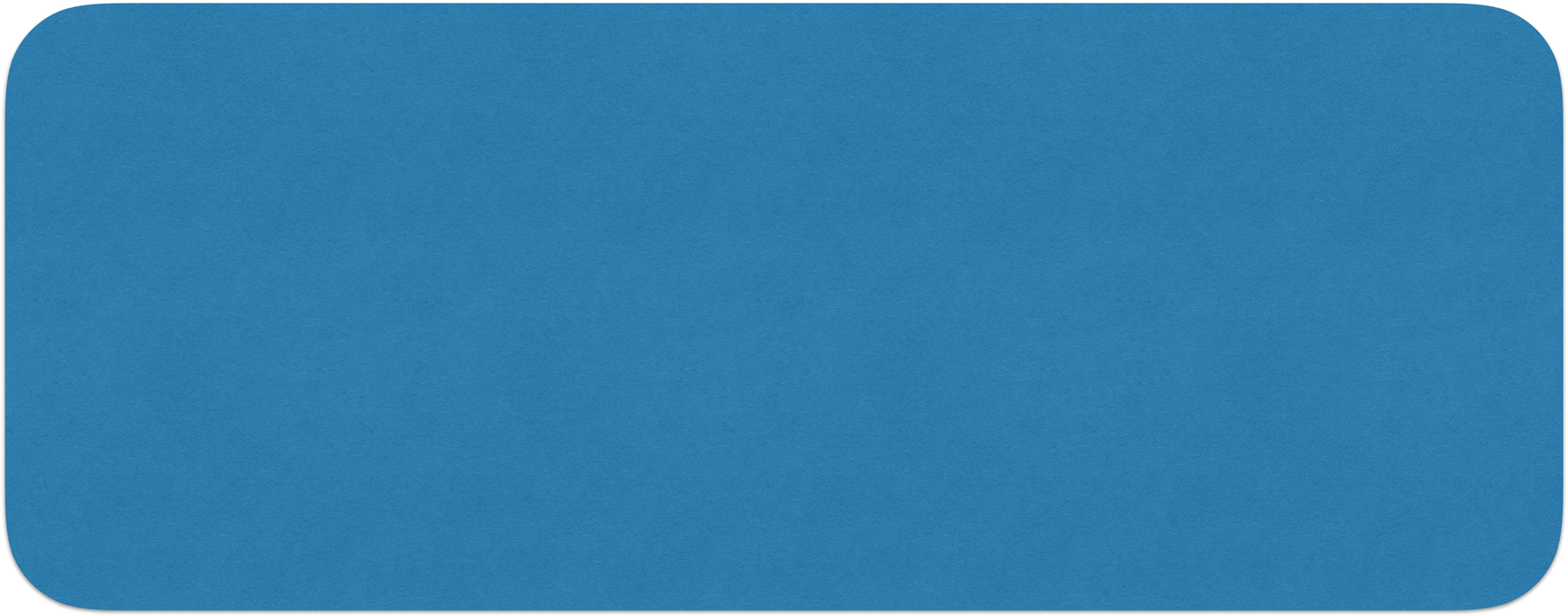
```
const router = new VueRouter({
  routes: [
    {
      path: '/user', component: UserLayout,
      children: [
        { path: 'me', component: Me },
        { path: ':id', component: User }
      ]
    }
  ]
})
```

Route Meta Fields

```
const router = new VueRouter({
  routes: [
    {
      path: '/user', component: UserLayout,
      children: [
        { path: 'me', component: Me, meta: { requiresAuth: true } },
        { path: ':id', component: User }
      ]
    }
  ]
})
```

Route Meta Fields

```
router.beforeEach((to, from, next) => {
  next()
})
```



Route Meta Fields

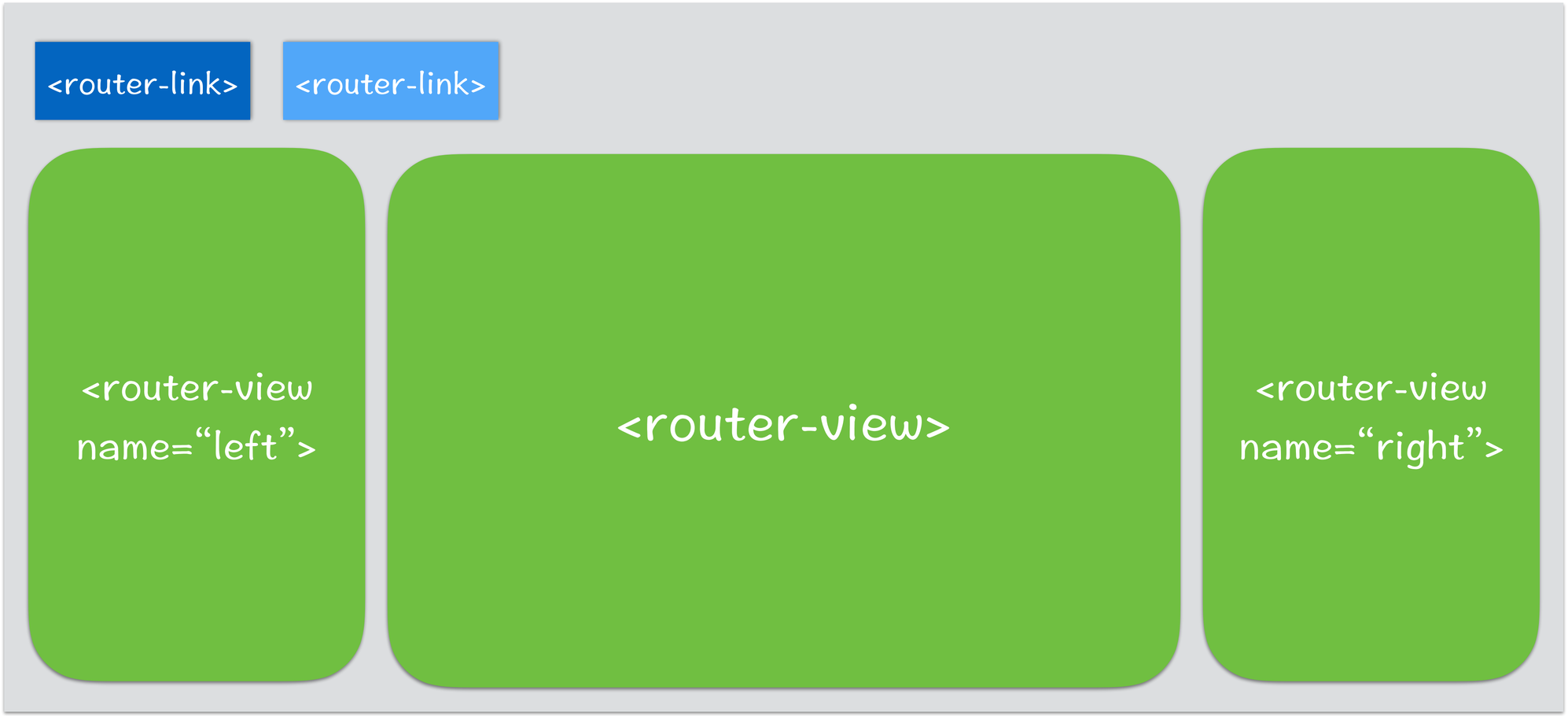
```
router.beforeEach((to, from, next) => {
  if (to.matched.some((record) => record.meta.requiresAuth)) {
    if (!Auth.IsSignedIn()) {
      next({
        path: '/signin',
        query: { redirect: to.fullPath }
      })
      return
    }
  next()
})
```

Position

```
const router = new VueRouter({
  scrollBehavior (to, from, savedPosition) {
    if (to.hash) {
      return { selector: to.hash }
    }
    if (savedPosition) {
      return savedPosition
    }
    return { x: 0, y: 0 }
  },
  ...
})
```

/#contact

Named Views


<router-link>
<router-link>
<router-view name="left">
<router-view>
<router-view name="right">

Named Views

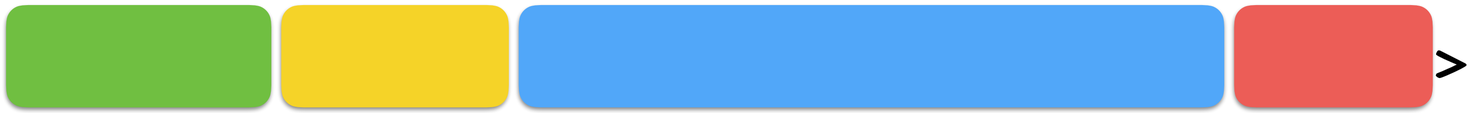
```
const router = new VueRouter({
  routes: [
    {
      path: '/',
      components: {
        default: Foo,
        left: Bar,
        right: Baz
      }
    }
  ]
})
```

Programmatic Navigation

- router.push
- router.replace
- route.go

- router.push(`/user/${userId}`)
- router.push({ path: '/signin', query: { redirect: '/profile' } })
- router.push({ name: 'user', params: { id: 8 } })
- router.replace('/not-support')
- router.go(-1)

Navigation

<router-link  >

Navigation

Navigation

<router-link tag="a" to="/" >

- to="/"
- :to="`/user/${id}?tab=review`"
- :to="{ path: '/home'}"
- :to="{ name: 'user', params: { userId: 123 }}"
- :to="{ path: 'register', query: { mode: 'email' }}"

Navigation

<router-link tag="a" to="/" active-class="active" >

Navigation

<router-link tag="a" to="/" active-class="active" exact>

Profile

=> /user/:id

exact

Review

=> /user/:id/review

Navigation

<router-link tag="a" to="/" active-class="active" ????>

- replace => router.replace
- append
  - /a
    - to="b" => /b
    - to="b" append => /a/b

Reuse Component

```
export default {
  data () {
    return {
      loading: false,
      id: 0,
      data: null
    }
  },
  created () {
    this.reload()
  },
  watch: {
    $route: 'reload'
  },
  methods: {
    reload () {
      this.id = this.$route.params.id
      this.loading = true
      User.get(this.id)
        .finally(() => { this.loading = false })
        .subscribe(
          (res) => {
            if (!res) {
              this.$router.replace('/users')
              return
            }
            this.data = res
          }
        )
    }
  }
}
```

Homework

- Connect to Firebase
- Sign in / Sign out

Section 4

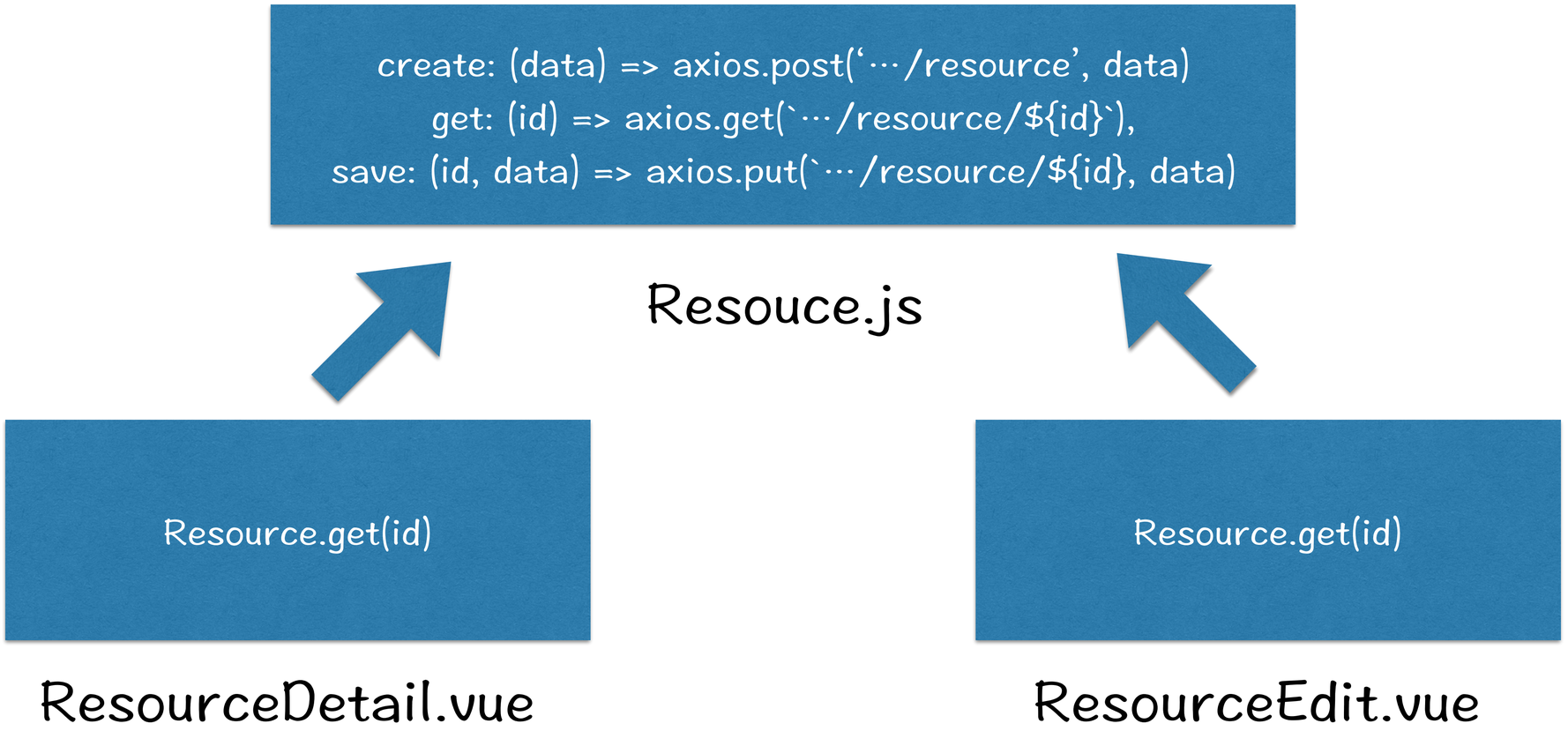
— Services —

What is Service ?

ResourceDetail.vue

ResourceEdit.vue

What is Service ?



What is Service ?

```
get: (url) => axios.get(BASE_URL + url),
post: (url, data) => axios.post(BASE_URL + url, data),
put: (url, data) => axios.put(BASE_URL + url, data)
```

API.js

```
create: (data) => API.post('/resource', data)
get: (id) => API.get(`/resource/${id}`),
save: (id, data) => API.put(`/resource/${id}, data)
```

Resouce.js

```
signIn: (email, password) =>
API.post('/auth', { email, password })
```

Auth.js

Object/Class

```
// API.js
import axios from 'axios'
const BASE_URL = 'http://localhost:9000'

let token = ''
const makeUrl = (url) => BASE_URL + url
const getData = (res) => res.data
const makeConfig = () => ({
  headers: {
    Authorization: `Bearer ${token}`
  }
})

const get = (url) => axios.get(makeUrl(url), makeConfig()).map(getData)
const post = (url, data) => axios.post(makeUrl(url), data, makeConfig()).map(getData)
const put = (url, data) => axios.put(makeUrl(url), data, makeConfig()).map(getData)
const del = (url) => axios.delete(makeUrl(url), makeConfig()).map(getData)
const setToken = (value) => { token = value }

export default {
  get,
  post,
  put,
  delete: del,
  setToken
}
```

Vue Instance

```
// API.js
import Vue from 'vue'
import axios from 'axios'

const BASE_URL = 'http://localhost:9000'

let token = ''
const makeUrl = (url) => BASE_URL + url
const getData = (res) => res.data
const makeConfig = () => ({
  headers: {
    Authorization: `Bearer ${token}`
  }
})

export default new Vue({
  methods: {
    get: (url) => axios.get(makeUrl(url), makeConfig()).map(getData),
    post: (url, data) => axios.post(makeUrl(url), data, makeConfig()).map(getData),
    put: (url, data) => axios.put(makeUrl(url), data, makeConfig()).map(getData),
    delete: (url) => axios.delete(makeUrl(url), makeConfig()).map(getData),
    setToken: (value) => { token = value }
  }
})
```

Filter

```
<template>
  <div>
    <input v-model="name">
    Hello, {{ upperName }}
  </div>
</template>

<script>
export default {
  data: () => ({
    name: ''
  }),
  computed: {
    upperName () { this.name.toUpperCase() }
  }
}
</script>
```

Filter

```
Vue.filter('toUpper', (value) => {
  if (typeof value === 'string') return value.toUpperCase()
  return value
})
```

Filter

```
import toUpper from 'lodash/toUpper'

Vue.filter('toUpper', toUpper)
```

Filter

```
<template>
  <div>
    <input v-model="name">
    Hello, {{ name | toUpper }}
  </div>
</template>

<script>
export default {
  data: () => ({
    name: ''
  })
}
</script>
```

Reactive Filter

```
import moment from 'moment'

Vue.filter('fromNow', (value) => moment(value).fromNow())
```

Reactive Filter

```
import moment from 'moment'

const ticker = new Vue({
  data () {
    return {
      tick: 0
    }
  },
  created () {
    setInterval(() => {
      this.tick = Date.now()
    }, 1000)
  }
})

Vue.filter('fromNow', (value) => {
  ticker.tick
  return moment(value).fromNow()
})
```

Mixin

Local

Global

Local Mixin

```
// APIMixin.js
const APIMixin = {
  computed: {
    API: () => API
  }
}
```

```
import APIMixin from './APIMixin'
export default {
  mixins: [APIMixin],
  data: () => ({ data: null })
  created () {
    this.API.get(`/user/${this.$route.params.id}`)
        .then((res) => { this.data = res })
  }
}
```

Global Mixin

```
Vue.mixin({
  computed: {
    API: () => API
  }
})
```

```
export default {
  data: () => ({ data: null })
  created () {
    this.API.get(`/user/${this.$route.params.id}`)
        .then((res) => { this.data = res })
  }
}
```

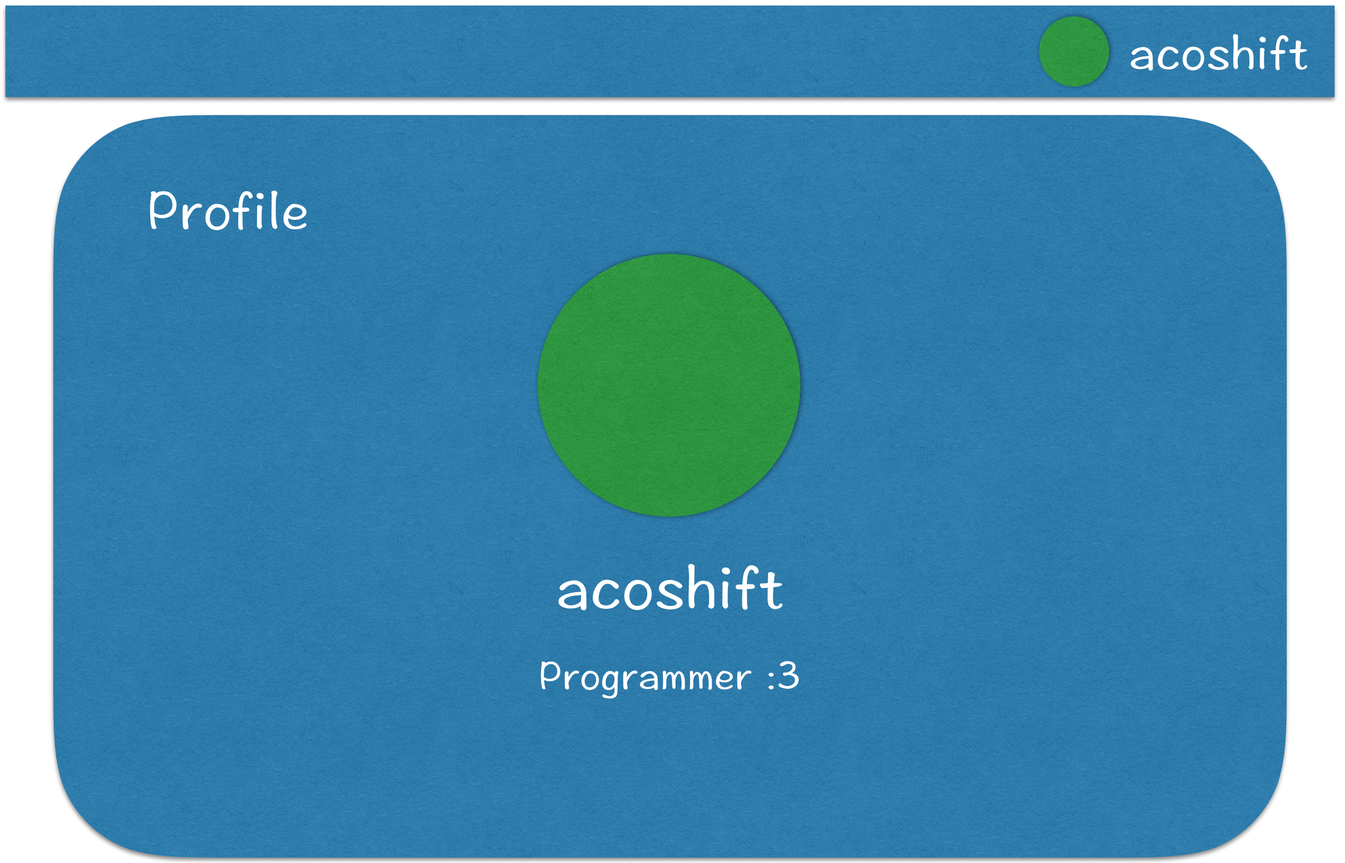
Homework

- Refactor to service
- Profile (View/Edit)
- View Other User Profile

Section 5

— State Management —

State Management ?



Local State

```
// NavBar.vue

created () {
  Me.subscribe((data) => {
    this.profile = data
  })
}
```

```
// Profile.vue

created () {
  Me.subscribe((data) => {
    this.profile = data
  })
}
```

Hardcore Props

```
// App.vue

<nav-bar :profile="profile"></nav-bar>
<router-view :profile="profile"></router-view>
...
data: () => ({ profile: null }),
created () {
  Me.subscribe((data) => {
    this.profile = data
  })
}
```

Global State

```
// store.js

const store = {
  me: null
}

Me.subscribe((data) => {
  store.me = data
})

export default store
```

Global State

```
import store from '../store'
...

data: () => ({
  store
})
// or use global mixin
```

Reactive Object

```
<div v-if="store.me">
  <img :src="store.me.photo">
  <span>{{ store.me.name }}</span>
</div>
```

Reactive Object

```
// store.js
export default new Vue({
  data: () => ({
    me: null
  }),
  created () {
    Me.subscribe((data) => {
      store.me = data
    })
  }
})
```

Vue Instance

Rxjs

```
// auth.js

const $currentUser = new BehaviorSubject(undefined)
const init = () => {
  firebase.auth().onAuthStateChanged((user) => {
    $currentUser.next(user)
  })
}
export default {
  init,
  get currentUser () { return $currentUser.filter((x) => x !== undefined) }
}
```

Rxjs

```
// router.js
...
Auth.currentUser
  .first()
  .subscribe((user) => {
    if (user) {
      next()
      return
    }
    next({ path: '/signin', query: { redirect: to.fullPath } })
  })
```

Rxjs

```
// user.js
const get = (id) => Observable.create((o) => {
  const ref = firebase.database().ref(`user/${id}`)
  const fn = ref.on('value', (snapshot) => {
    o.next(snapshot.val())
  }, (err) => { o.error(err) })
  return () => ref.off('value', fn)
})
const getOnce = (id) => get(id).first()
```

Rxjs

```
// me.js
const get =
  Auth.currentUser
    .flatMap((user) => User.get(user.uid))
```

Rxjs w/o VueRx

```
export default {
  data: () => ({
    me: null,
    me$: null
  }),
  created () {
    this.me$ = Me.get().subscribe((data) => { this.me = data })
  },
  beforeDestroy () {
    this.me$.unsubscribe()
  }
}
```

VueRx

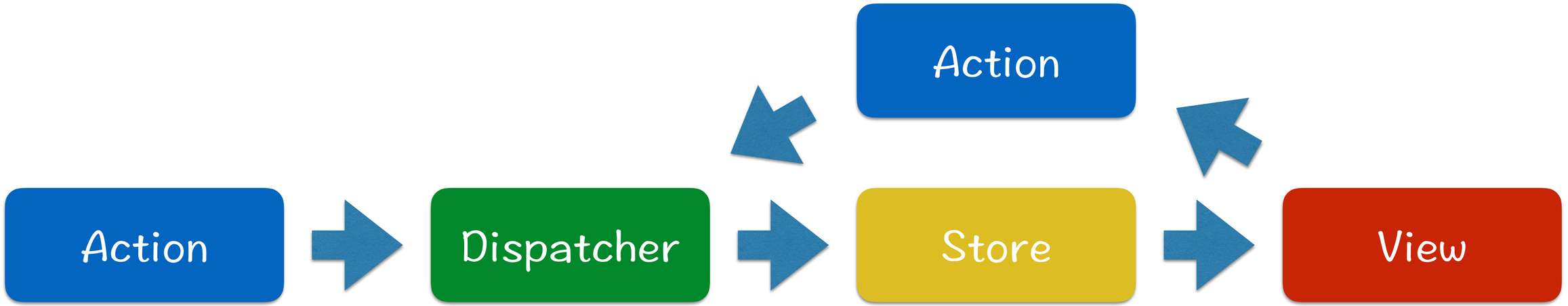
```
import Vue from 'vue'
import VueRx from 'vue-rx'
import { Observable, Subscription } from 'rxjs'

Vue.use(VueRx, { Observable, Subscription })
```

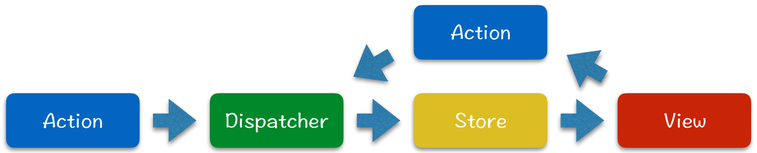
Rxjs + VueRx

```
export default {
  subscriptions: () => ({
    me: Me.get()
  })
}
```

Observable

Flux



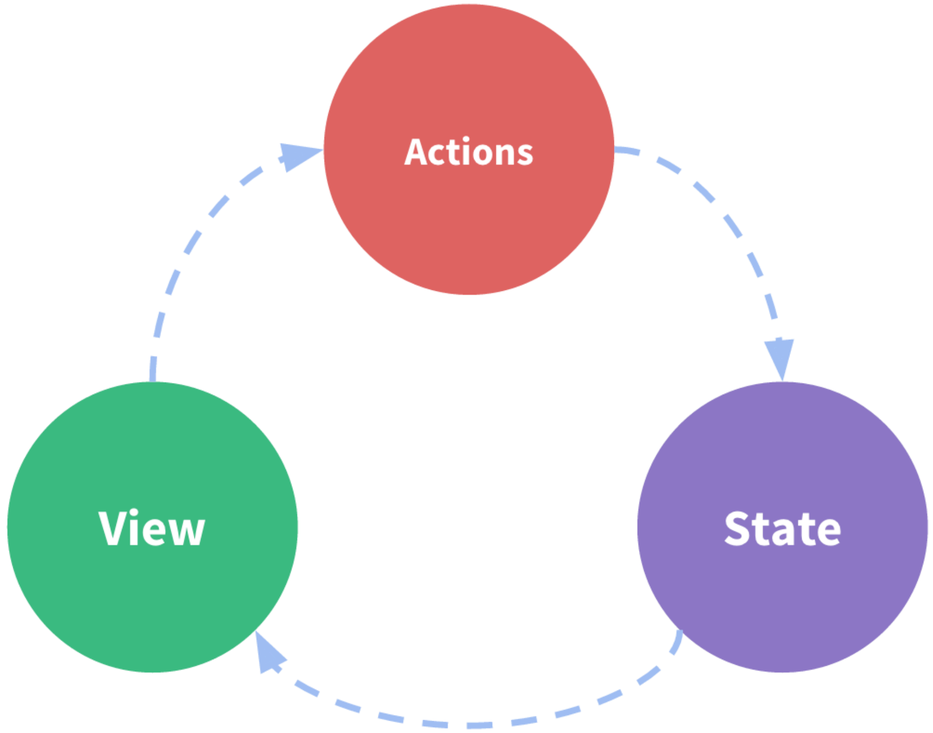


Vue + Flux

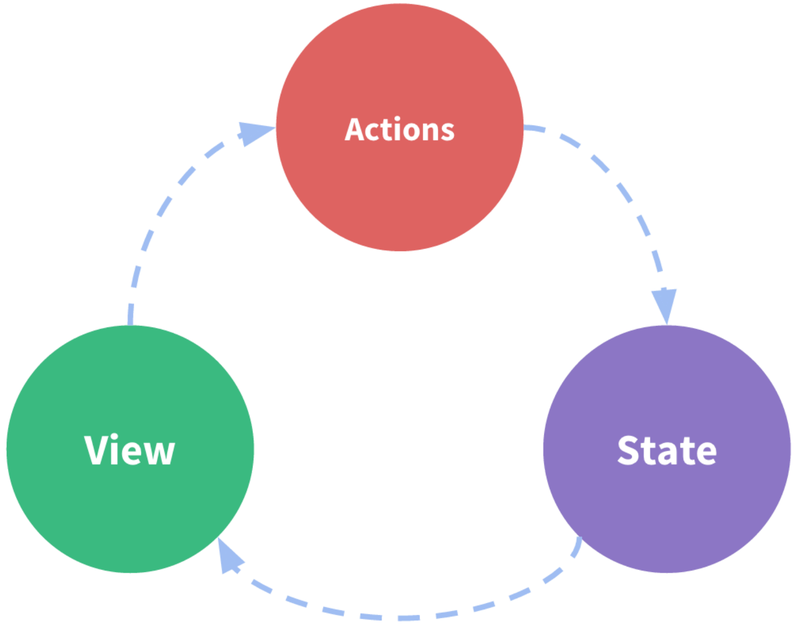
```
Vue.mixin({
  data: () => ({
    store: {…}
  }),
  methods: {
    dispatch (type, payload) {
      …
    }
  }
})
```

```
export default {
  computed: {
    value () { return this.store.value }
  }
  methods: {
    add () {
      this.dispatch('add', 1)
    }
  }
}
```

Vue

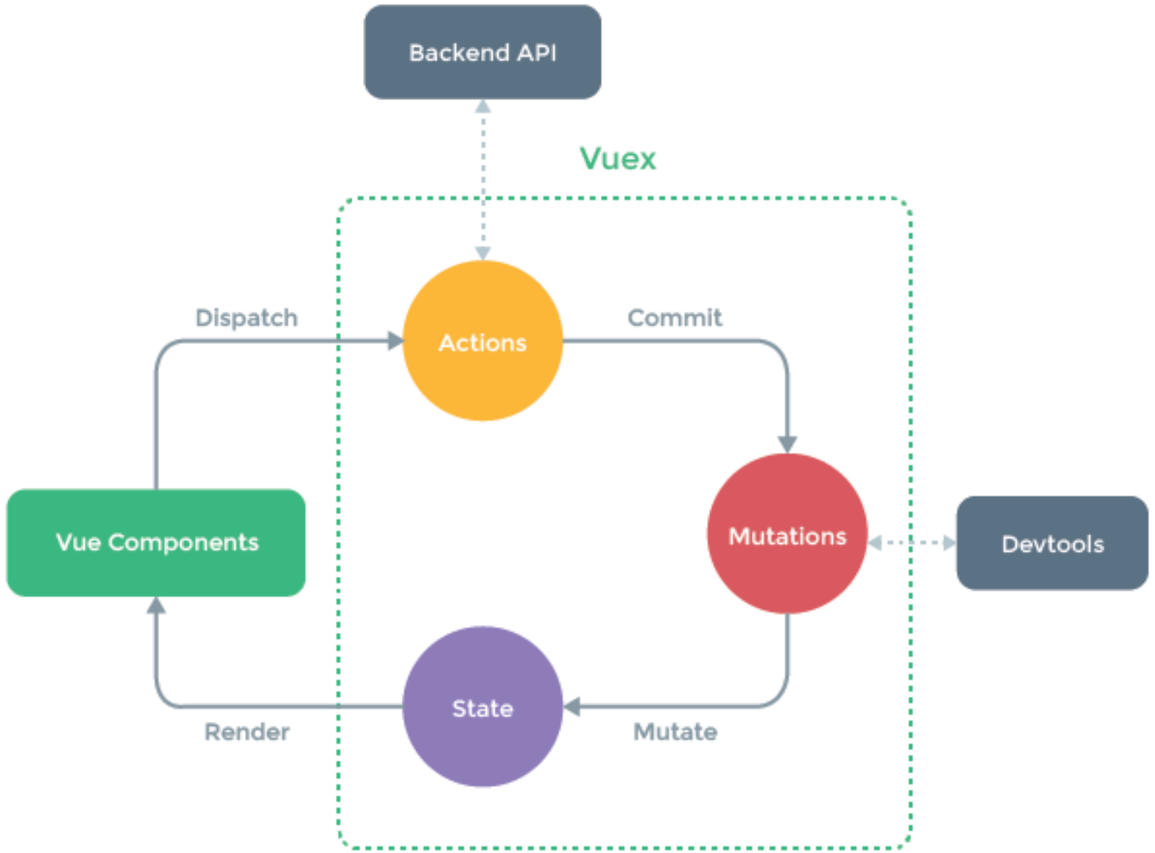


Vue




```
<div>{{ count }}</div>

export default {
  data: () => ({ count: 0 }),
  methods: {
    incr () {
      ++this.count
    }
  }
}
```

Vuex

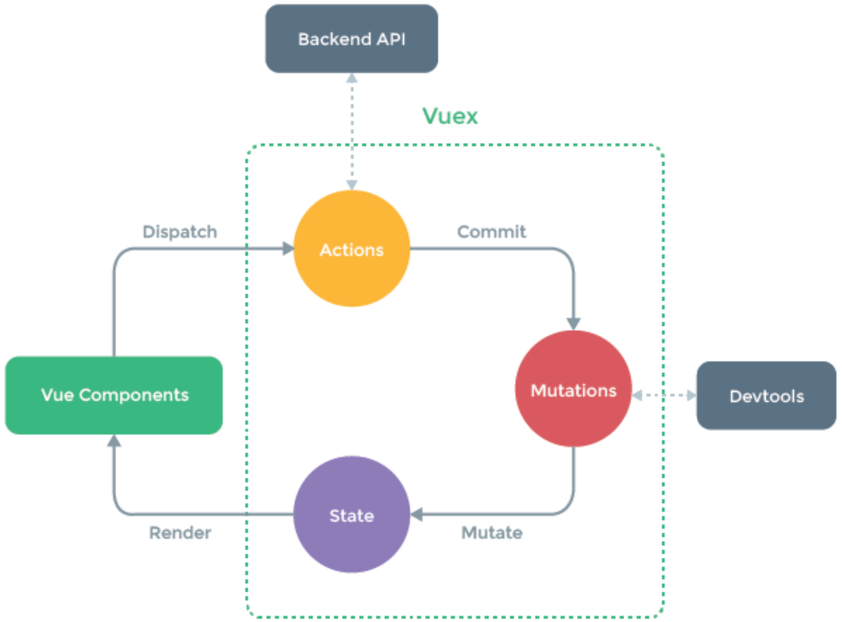

Backend API
Vuex
Dispatch
Actions
Commit
Vue Components
Mutations
Devtools
State
Render
Mutate

Vuex

```
import Vue from 'vue'
import Vuex from 'vuex'

Vue.use(Vuex)
```

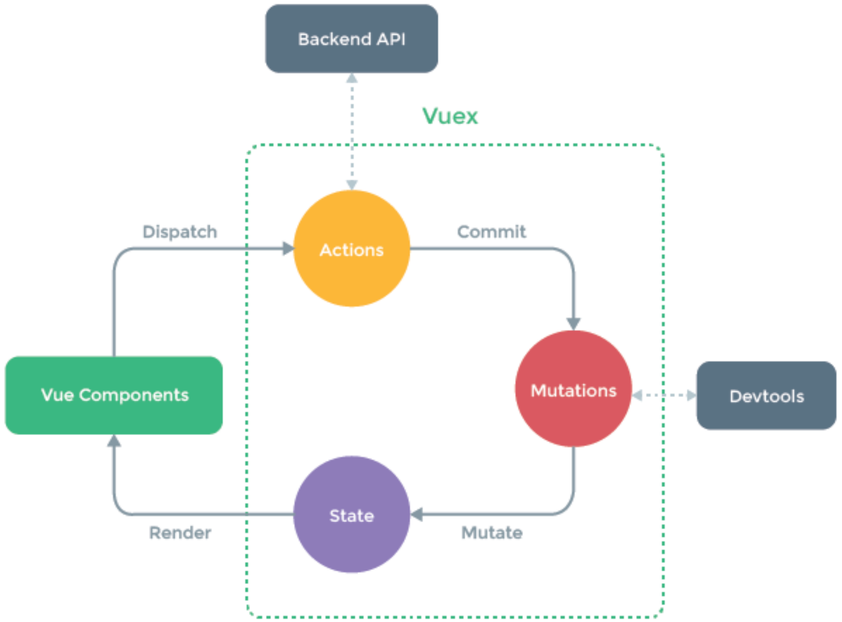
Vuex

```
// store.js
export default new Vuex.Store({
  state: {
    count: 0
  },
  mutations: {
    incr: (state) => ++state.count,
    decr: (state) => --state.count
  }
})
```



Vuex

```
import store from '../store'
export default {
  computed: {
    count: () => store.state.count
  },
  methods: {
    incr: () => store.commit('incr'),
    decr: () => store.commit('decr')
  }
}
```



Vuex

```
// main.js
import store from ‘./store’

new Vue({
  ...
  store
})
```

Vuex

```
export default {
  computed: {
    count () { return this.$store.state.count }
  },
  methods: {
    incr () { this.$store.commit('incr') },
    decr () { this.$store.commit('decr') }
  }
}
```

Vuex - State, Getters

```
// store.js
export default new Vuex.Store({
  state: {
    tweets: [···],
    currentUser: {···}
  },
  getters: {
    myTweets: (state) =>
      state.tweets.filter((it) => it.owner === state.currentUser.uid)
  }
})
```

```
export default {
  computed: {
    tweets () { return this.$store.getters.myTweets }
  }
}
```

Vuex - Mutations, Actions

```
// store.js
export default new Vuex.Store({
  ...
   mutations: {
     addBy: (state, payload) => { state.count += payload.value }
   },
   actions: {
     incr: (ctx) => ctx.commit('addBy', { value: 1 })
   }
})
```

```
export default {
  methods: {
    incrValue () {
      this.$store.dispatch('incr')
    }
  }
}
```

Vuex - Helper

```
export default {
  computed: {
    count () { return this.$store.state.count },
    tweets () { return this.$store.getters.myTweets }
  },
  methods: {
    decr () { this.$store.commit('decr') },
    incr () { this.$store.dispatch('incr') }
  }
}
```

mapState, mapGetters

mapMutations, mapActions

Vuex - Helper

```
export default {
  computed: {
    ...mapState(['count']),
    ...mapGetters({ tweets: 'myTweets' })
  },
  methods: {
    ...mapMutations(['decr']),
    ...mapActions(['incr'])
  }
}
```

mapState, mapGetters

mapMutations, mapActions

Homework

- Upload Photo
- Tweets in Home page

Section 6

— Production —

Environment

Development

- API_URL: 'http://localhost:9000'
- DEBUG: true
- FEATURE1: true
- FEATURE2: true

Staging

- API_URL: 'https://staging.mysite.com/api'
- DEBUG: true
- FEATURE1: true
- FEATURE2: true

Production

- API_URL: 'https://mysite.com/api'
- DEBUG: false
- FEATURE1: true
- FEATURE2: false

Makefile

```
$ npm run dev
$ npm run build
...
// package.json
"scripts": {
  "dev": "node build/dev-server.js",
  "build": "node build/build.js",
  "unit": "karma start test/unit/karma.conf.js --single-run",
  "e2e": "node test/e2e/runner.js",
  "test": "npm run unit && npm run e2e",
  "lint": "eslint --ext .js,.vue src test/unit/specs test/e2e/specs"
}
```

Makefile

```
dev:
	node build/dev-server.js

build-production: clean
	node build/build.js

build-staging: clean
	node build/build-staging.js

clean:
	rm -rf dist

use-production:
	firebase use default

use-staging:
	firebase use staging

deploy:
	firebase deploy

production: build-production use-production deploy

staging: build-staging use-staging deploy
```

```
$ make dev
$ make staging
$ make production
```

Makefile

```
SRC = index.js db.js

build: clean $(SRC)


clean:
    rm -rf build/


%.js: src/%.js .babelrc
    mkdir -p build/$(@D)
    babel $< -o build/$@
```

```
$ make build
# or
$ make
```

Deploy to Firebase

```
// firebase.json
{
  "database": {
    "rules": "database.rules.json"
  },
  "hosting": {
    "public": "dist",
    "rewrites": [
      {
        "source": "**",
        "destination": "/index.html"
      }
    ]
  }
}
```

```
$ firebase deploy
$ firebase deploy --only hosting
$ firebase deploy --only database
```

Deploy to Firebase

```
// .firebaserc
{
  "projects": {
    "default": "twitty",
    "staging": "twitty-staging"
  }
}
```

```
$ firebase use default
$ firebase use staging
```

NGINX

```
server {
    listen       80;
    server_name  localhost;

    location / {
        root   /usr/share/nginx/html;
        index  index.html;
        try_files $uri /index.html;
    }
}
```

Docker

```
# Dockerfile
FROM nginx:latest
COPY dist /usr/share/nginx/html
COPY nginx.conf /etc/nginx/nginx.conf
```

Docker

```
$ docker build -t twitty .
$ docker run --rm -p 8080:80 twitty

$ docker run --name twitty -d -p 8080:80 twitty
```

Homework

- Website link