

Vue Router

Bok, Jong Soon
javaexpert@nate.com
<https://github.com/swacademy/Vue.js>

SPA

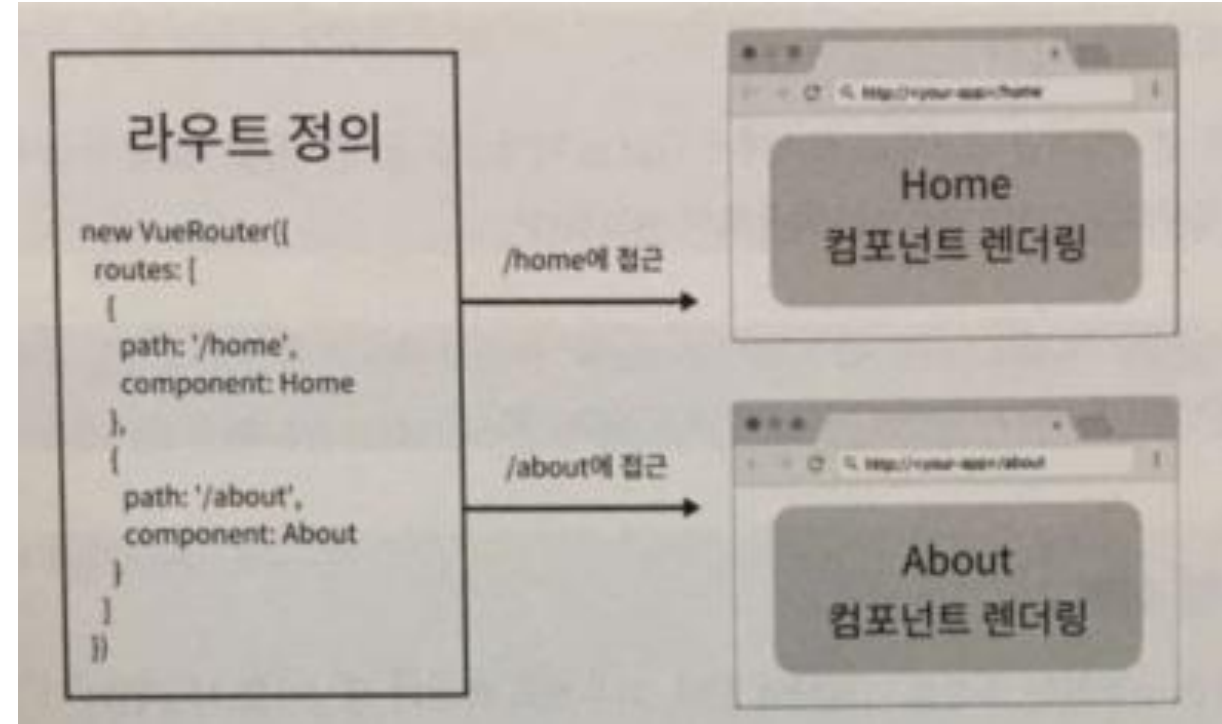
- 처음 HTML Page 하나를 Load한 다음, 그 이후 사용자의 요청에 따라 Ajax로 정보를 받아오면서 동적으로 Page를 Update하는 Web Application.
- 일반적인 Web Application은 Page 이동 시 대상 URL을 Server에 요청해서 전체 Page를 Loading한다.
- SPA는 Page 이동을 Client에서 처리한다.
- Page 이동 시 Ajax를 사용해서 필요한 때에 Data를 받아와서 View를 화면에 표시한다.
- 전체 Page에 해당하는 HTML을 모두 받아오는 데 필요한 Overhead가 줄어들기 때문에 Application의 속도가 향상되며, 더욱 매끄러운 User Experience를 제공할 수 있다.

SPA (Cont.)

- SPA를 구현하기 위해 고려할 사항
 - Client-side에서 History를 관리하는 Page 이동(Routing 관리)
 - 비동기로 Data 받아오기
 - View Rendering
 - Module화된 Code 관리
- Router 또는 Routing Library Module이 이런 기능들을 제공한다.
- SPA로 만들면 Page가 이동할 때 Cross Fade나 Slide 등의 부드러운 Animation을 재생할 수도 있다.

Vue Router

- Vue Library를 이용하여 Single Page Application을 구현할 때 사용하는 Library.
- Component와 URL을 연결해 주는 기능
- <https://router.vuejs.org/>
- <https://github.com/vuejs/vue-router>





Lab : Vue Routing Concept



Vue Router (Cont.)

■ Vue Router Installation

- CDN 방식
- NPM 방식

■ CDN 방식

```
<script src="https://unpkg.com/vue-router/dist/vue-router.js"></script>
```

```
<script src="https://unpkg.com/vue@2.6.10/dist/vue.js"></script>  
<script src="https://unpkg.com/vue-router@3.1.3/dist/vue-router.js"></script>
```

■ NPM 방식

```
npm install vue-router
```

Vue Router 등록

- Vue Router를 설치한 후, 다음과 같은 코드를 통해 Router Instance를 생성하고 Vue Instance에 등록한다.

```
<script>
  //Router Instance 생성
  let router = new VueRouter({
    //Router Options
  })

  //Vue Instance에 Router Instance 등록
  let vm = new Vue({
    el : '#app',
    router : router
  })
</script>
```

Vue Router Options

- Vue Router를 등록 후, Router에 Option을 정의한다.
- vue-router가 이동할 수 있는 Router 객체 Literal 배열로 전달.
- Router 객체 Literal 배열 → Routing Table

- Syntax

{

path : <URL 경로>,

component : <Route에 응답할 Vue Component

}

path 속성은 /로 시작하는 URL경로

Vue Router Options (Cont.)

- 대부분의 SPA에서는 아래와 같이 2개 Option을 지정한다.

- routes

- Routing 할 URL과 Component 값 지정

- mode

- URL의 Hash 값 제거 속성

URL 경로의 중간에 해쉬(#)문자 포함,
HTML5 History 기능 지원
오래된 IE는 history mode 지원 안함.

```
let router = new VueRouter({  
  mode : 'history',  
  routes : [  
    { path : '/login', component : LoginComponent},  
    { path : '/home', component : HomeComponent},  
  ]  
})
```



Lab : 간단한 Vue Routing





Lab : @Vue/cli로 Vue Routing 하기

Vue Router Options (Cont.)

■ 이름 있는 Router

- Route에는 고유한 이름을 붙일 수 있다.
- URL이 복잡한 Application에서 Code를 간단하게 작성가능.
- 매개변수를 가진 Navigation을 만들 때 이름이 필요.

```
const router = new VueRouter({  
  routes : [  
    {  
      name : 'product',    //Route에 이름 붙이기  
      path : '/product',  
      component : Product  
    }  
  ]  
})
```



Lab : Named Routing



Vue Router Options (Cont.)

■ 요청 매개변수와 Pattern Matching

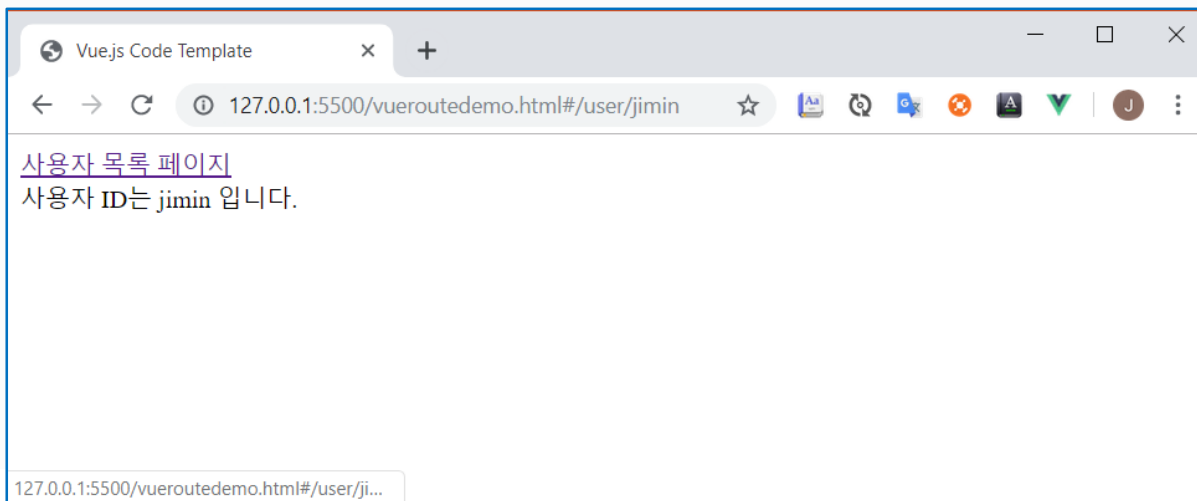
- SPA는 접근 대상 URL의 Pattern Matching을 통해 Parameter를 전달하는 경우가 많다.
- 사용자 이름 또는 상품 ID와 같은 변할 수 있는 URL을 매칭시킬 때는 요청 매개변수 사용한다.
- ex) 사용자 상세 정보 Page → **/user/:userid**
- URL 경로에 **:**을 붙여 Pattern 작성.
- URL에서 이 Pattern과 일치하는 Parameter는 Component에서 **\$route.params**의 속성 중 Pattern에 사용한 Parameter 이름과 같은 속성명으로 접근 가능.
- 오른쪽의 예처럼 **id**라는 이름을 붙인 요청 매개변수는 아래와 같이 사용한다.

this.\$route.params.id

```
const router = new VueRouter({
  routes : [
    {
      path : '/product/:id', // URL을 통해 매개변수 받기
      component : Product
    }
  ]
})
```

Vue Router Options (Cont.)

```
11 <body>
12   <div id="app">
13     <router-link to="/user/jimin">사용자 목록 페이지</router-link>
14     <router-view></router-view>
15   </div>
16   <script>
17     var router = new VueRouter({
18       routes : [
19         {
20           //Pattern Matching에 사용되는 Pattern은 :(콜론)으로 시작
21           path : '/user/:userid',
22           component : {
23             template : '<div>사용자 ID는 {{ $route.params.userid }} 입니다.</div>'
24           }
25         }
26       ]
27     });
28
29     var vm = new Vue({
30       router : router
31     }).$mount('#app');
32   </script>
```





Lab : 요청 매개변수와 Pattern Matching



Vue Router Options (Cont.)

■ query

- URL query는 아래와 같이 사용한다.

`this.$route.query`

Vue Router Options (Cont.)

■ meta 필드

- Page를 볼 때 인증이 필요한지 등의 Route 고유 정보도 설정 가능.

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

Vue Router Options (Cont.)

■ redirect

- 특정 경로로 Redirect할 수 있다.

```
const router = new VueRouter({
  routes : [
    // 'a' 에서 'b' 로 Redirect 하기
    {
      path : '/a',
      redirect : '/b'
    },
    // 이름을 지정해서 Redirect 하기
    {
      path : '/a',
      redirect : {
        name : 'foo'
      }
    }
  ]
})
```

router-view

- Browser의 주소 창에서 URL이 변경되면, 앞에서 정의한 **routes** 속성에 따라 해당 Component가 화면에 나타난다.
- 이 때 나타나는 지점이 Template의 **<router-view>**이다.

```
<div id="app">  
  <router-view></router-view>  
  <!-- LoginComponent 또는 HomeComponent -->  
</div>
```

- 앞에서 정의한 Routing Option 기준으로 */login*은 LoginComponent를 */home*은 HomeComponent를 화면에 표시한다.

router-link

- 일반적으로 Web Page에서 Page 이동을 할 때는 사용자가 URL을 다 입력해서 이동하지 않는다.
- 이 때 화면에서 특정 Link를 Click해서 Page를 이동할 수 있게 해줘야 하는데 그게 바로 **<router-link>** 이다.
- 기본적으로 **to** 속성에 경로 지정.
- 문자열 전달하기

```
<router-link to="이동할 URL"></router-link>
```

```
<div>  
  <router-link to="/login"></router-link>  
</div>
```

router-link (Cont.)

- Template Literal 사용하기

```
<router-link :to="`/product/${ id }`"></router-link>
```

- 기본적으로 **<a>** tag로 만들어진다.
- 하지만, **tag** 속성을 지정하면 다른 tag로도 변경 가능.

```
<router-link to="/product" tag="button"></router-link>
```

router-link (Cont.)

- 객체 형식으로 지정하기
 - **name** : Route 이름
 - **path** : Route 경로
 - **params** : 요청 매개변수 객체
 - **query** : 쿼리 객체

```
<!-- 동작 가능-->
<router-link :to="{path : '/product' }"></router-link>
<!-- query 동작 가능-->
<router-link :to="{path : '/product', query:{ page : 1} }"></router-link>
<!-- params 때문에 동작 불가능-->
<router-link :to="{path : '/product', params: { id : 1} }"></router-link>
<!-- params가 있는 경우는 name을 지정해야 동작 가능-->
<router-link :to="{name: 'product', params : {id : 1} }"></router-link>
```

Hash Mode vs History Mode

- Vue Router는 Hash와 History 두 가지 Routing Mode가 있다.
- 기본은 Hash Mode(**#**)이다.
- Hash Mode
 - SPA가 개발되기 시작하면서 고안한 기법
 - 페이지가 이동하는 것처럼 보이지만 실제로는 페이지가 이동하는 것이 아님.
 - Hash 값의 변경에 따라 현재 Route를 감지하기 때문에 Browser에서 History API를 지원하지 않아도 가능.
 - Server에서도 별도의 설정을 할 필요도 없다.
 - 하지만, URL에 **#**가 붙는 내부 Page Link 기능을 사용할 수 없다.
 - <http://localhost:8080#hello> → ``
 - <http://localhost:8080/#/example/page>
 - 최근에는 HTML5 History API를 사용하여 URL 변경 방법을 선호

URL에 Hash 붙이지 않기 (Cont.)

■ History Mode

- HTML5의 History API의 **pushState()**를 사용하는 방법
 - URL을 변경하고 Browser의 History를 남기지만 실제로 Page는 이동하지 않는 기능.
- URL에서 Hash(#)를 제거하기 위한 Mode
- Browser를 Refresh하거나 직접 Page를 열었을 때, Page를 찾을 수 없는 문제가 발생
- <http://localhost:8080/example/page>
- Server에서 별도의 설정 필요.

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

Routing 전용 속성

■ \$router

- VueRouter Instance 자체를 가리킨다.
- `this.$router`

■ \$route

- Matching된 Route 정보가 들어오는 객체.
- `this.$route`
- 객체 정보들

```
{  
  fullPath, // '/'로 시작하는 전체 경로  
  hash,     // URL의 '#' 뒤에 연결되는 문자열  
  matched,  // 상위 Route를 포함한 중첩된 모든 Route 배열  
  meta,     // Route 메타 정보  
  name,     // Route 이름  
  params,   // Route 매개변수 객체  
  path,     // Route 경로  
  query,    // URL의 '?'에 이어지는 객체 정보  
}
```

동적 Routing

- SPA는 `router-link` element의 `to` 속성값이 정적인 경로에만 적용.
- 동적 Routing을 위해서는 `v-bind:to` directive 사용 필요.

Lab : 동적 Routing



Active Link Highlight

- **<route-link>**를 사용해서 만든 link는 자동으로 다음과 같은 class가 적용되어 Active 상태 여부를 판단 가능.
 - **.router-link-exact-active** : 전체가 매치되는 Route
 - **.router-link-active** : 매치한 경로를 포함하는 Route
- Highlight 전용 Style
 - .router-link-active {**
 - background: #e25193;**
 - }**

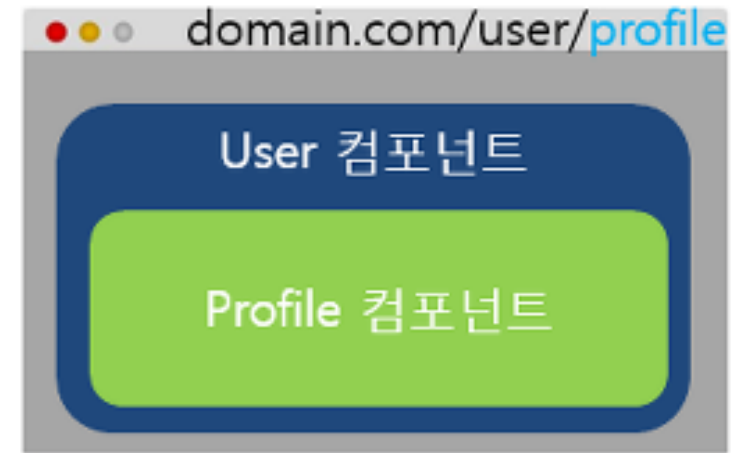
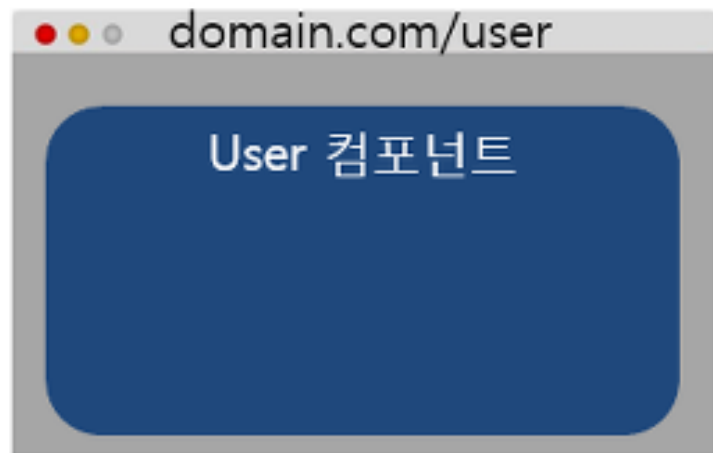
Vue.js Books

- [Go to Vue.js 첫 걸음](#)
- [Go to Vue.js 입문](#)
- [Go to 고양이도 할 수 있는 Vue.js](#)

Vue 3

Nested Route

- 여러 단계로 중첩된 Route
- 어떤 Component 안에 들어있는 Component에 대한 Route.
- Router로 Page 이동할 때 최소 2개 이상의 Component를 화면에 나타낼 수 있다.
- 상위 Component 1개에 하위 Component 1개를 포함하는 구조



Nested Router (Cont.)

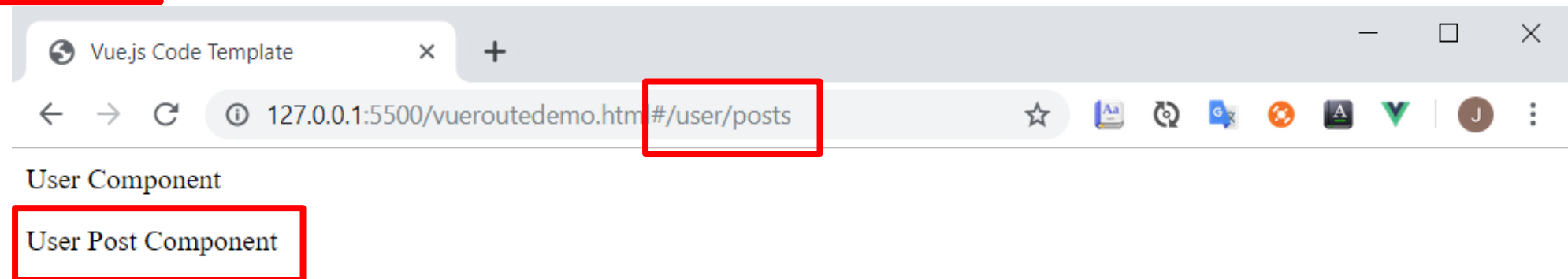
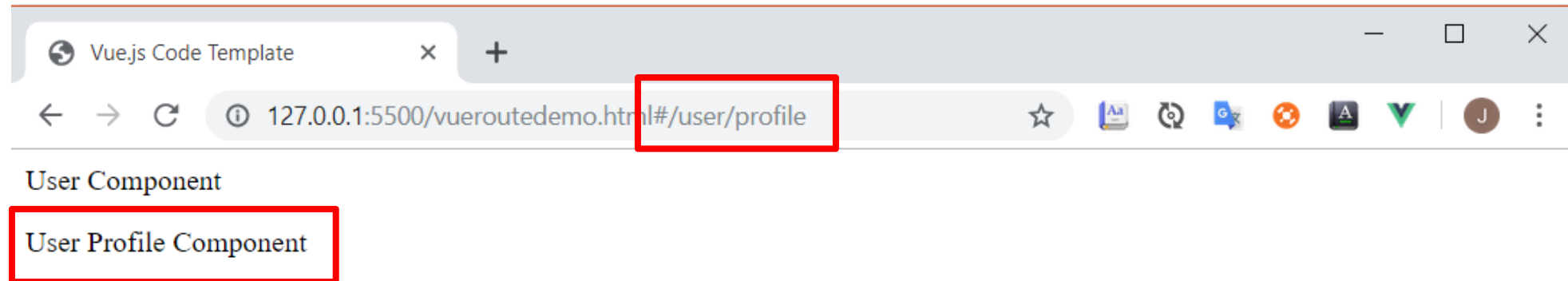
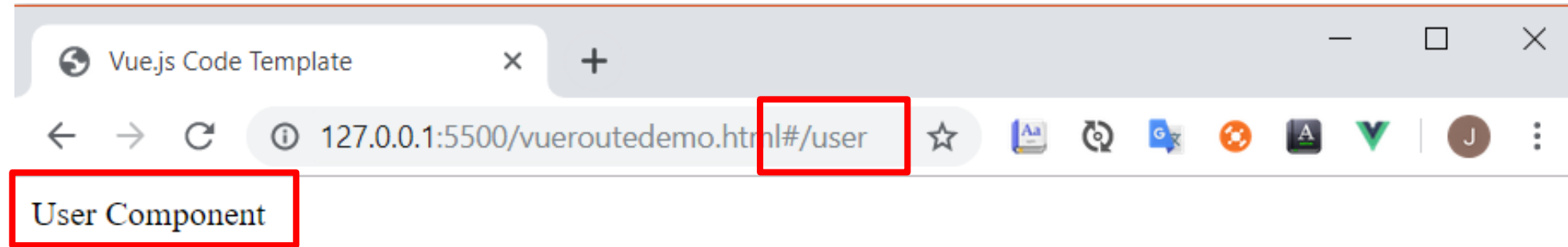
```
8     <script src="https://unpkg.com/vue@2.6.10/dist/vue.js"></script>
9     <script src="https://unpkg.com/vue-router/dist/vue-router.js"></script>
10 </head>
11 <body>
12     <div id="app">
13         <router-view></router-view>
14     </div>
15     <script>
16         var User = {
17             template : `
18                 <div>
19                     User Component
20                     <router-view></router-view>
21                 </div>
22             `,
23         };

```

Nested Router (Cont.)

```
24     var UserProfile = { template : '<p>User Profile Component</p>' };
25     var UserPost = { template : '<p>User Post Component</p>' };
26     var router = new VueRouter({
27         routes : [
28             {
29                 path : '/user',
30                 component : User,
31                 children : [
32                     {
33                         path : 'posts',
34                         component : UserPost
35                     },
36                     {
37                         path : 'profile',
38                         component : UserProfile
39                     }
40                 ]
41             }
42         ]
43     });
```


Nested Router (Cont.)





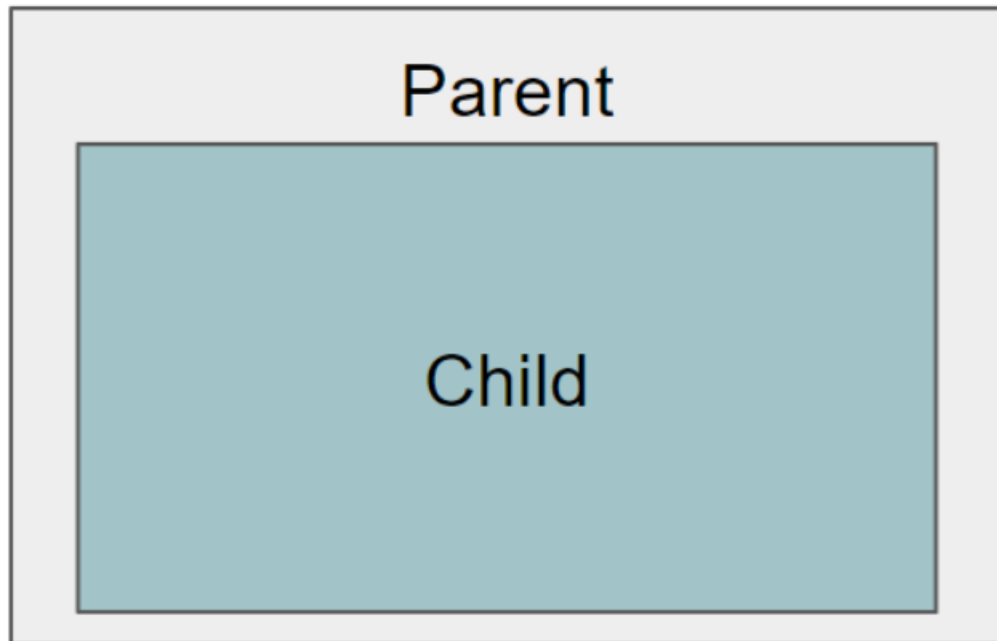
Lab : Nested Route



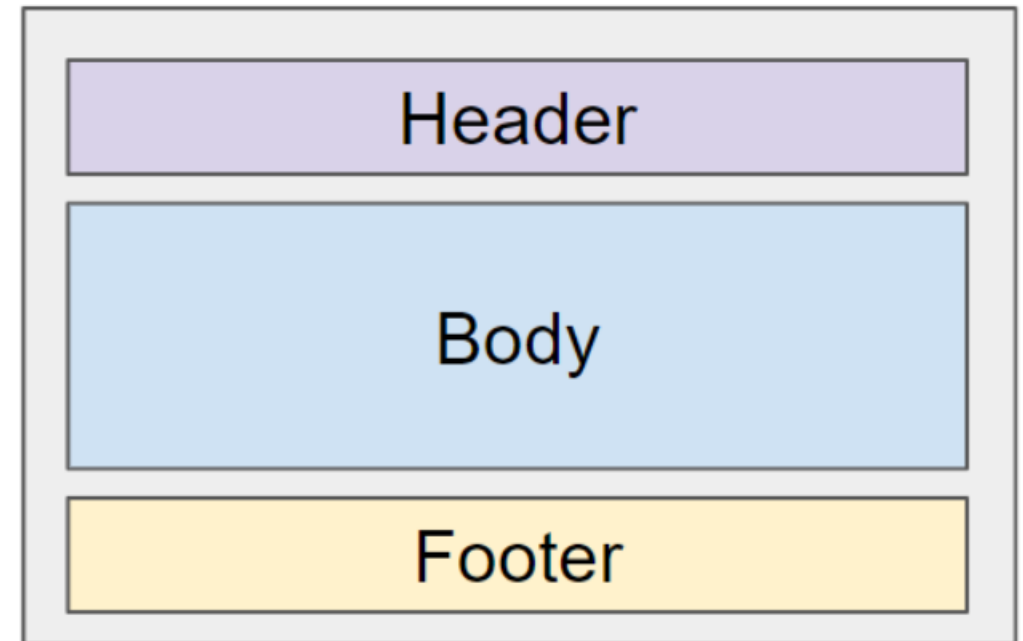
Named View

- 특정 Page로 이동했을 때 여러 개의 Component를 동시에 표시하는 Routing 방식
- 여러 개의 View를 하나의 View에서 보여주는 형태

네스티드 라우터 구조

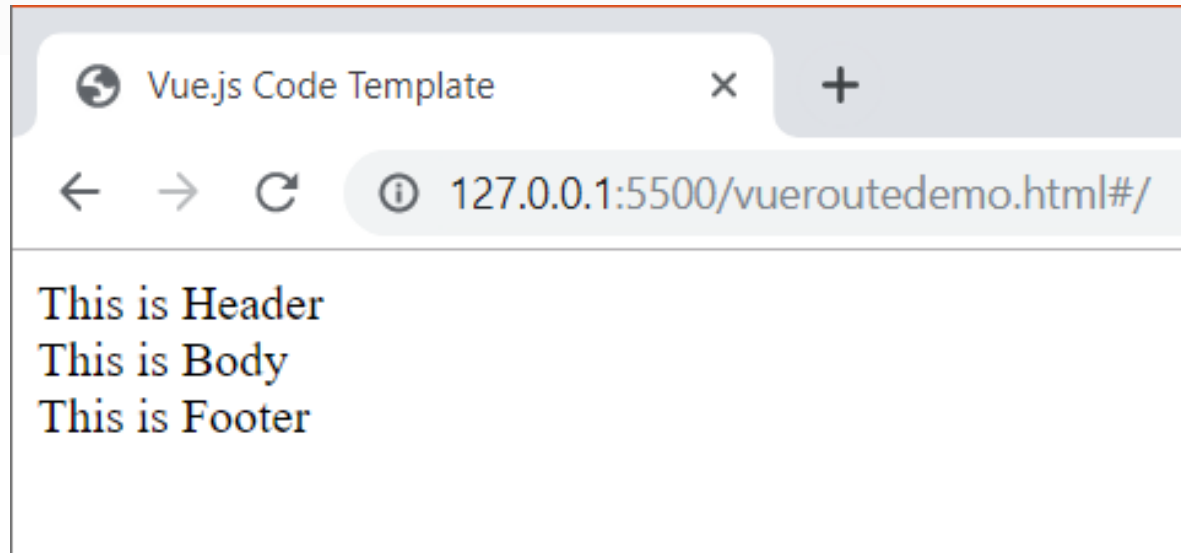


네임드 뷰 구조



Named View (Cont.)

```
8     <script src="https://unpkg.com/vue@2.6.10/dist/vue.js"></script>
9     <script src="https://unpkg.com/vue-router/dist/vue-router.js"></script>
10 </head>
11 <body>
12     <div id="app">
13         <router-view name='header'></router-view>
14         <router-view></router-view> <!--name 0/ 없는 경우는 default -->
15         <router-view name='footer'></router-view>
16     </div>
```



Named View (Cont.)

```
17     <script>
18         var Body = { template : '<div>This is Body</div>' };
19         var Header = { template : '<div>This is Header</div>' };
20         var Footer = { template : '<div>This is Footer</div>' };
21         var router = new VueRouter({
22             routes : [
23                 {
24                     path : '/',
25                     components : {
26                         default : Body,
27                         header : Header,
28                         footer : Footer
29                     }
30                 }
31             ]
32         });
33
34         var vm = new Vue({
35             router : router
36         }).$mount('#app');
37     </script>
```



Lab : Named View



Programmatic Navigation

- Vue Router가 router-link Component를 사용하여 anchor tag를 만드는 일반적인 방식이 아닌 Router 객체의 method를 사용하여 Routing 하는 방식
- 프로그래밍 방식이라고 함.
- 특정 처리가 끝난 후 Page를 이동시키거나 할 때 Vue Router Instance의 다음의 method를 사용한다.
- **router.push()**
 - History entry 추가하기
- **router.replace()**
 - History entry 수정하기
- **router.go()**
 - Browser level에서 Page 이동하기

Programmatic Navigation (Cont.)

■ router.push()

- 다른 경로로 이동시 사용
- 이동하면서 새로운 경로를 Browser의 History에 저장하기 때문에 사용자가 뒤로 가기 버튼을 눌렀을 때 이전 경로로 다시 돌아갈 수 있다.
- **router-link** Component를 사용자가 Click할 때, **Vue Router**가 내부적으로 호출하는 Method.
- 따라서, **push()**를 사용하여 이동하는 것과 **router-link** Component를 Click해서 이동하는 것은 같은 동작이다.
- **this.\$router.push('/product')**
- **this.\$router.push({ path : '/about' })**
- **this.\$router.push({ name : 'product', params : { id : 1 } })**
- **router.push({path: '/contacts', query: {pageno:1, pagesize:5 } })**

Programmatic Navigation (Cont.)

■ `router.replace()`

- replace 속성 설정시
 - `<router-link to="/product" replace>`
- `window` 객체의 `history.replaceState()`처럼 history entry를 만드는 대신 현재 history entry를 수정한다.
- `replace()`를 사용하여 Route를 변경하게 되면, 이후 사용자가 **뒤로 가기** 버튼을 눌러도 이전의 Route로 돌아갈 수 없다.

Programmatic Navigation (Cont.)

■ `router.go()`

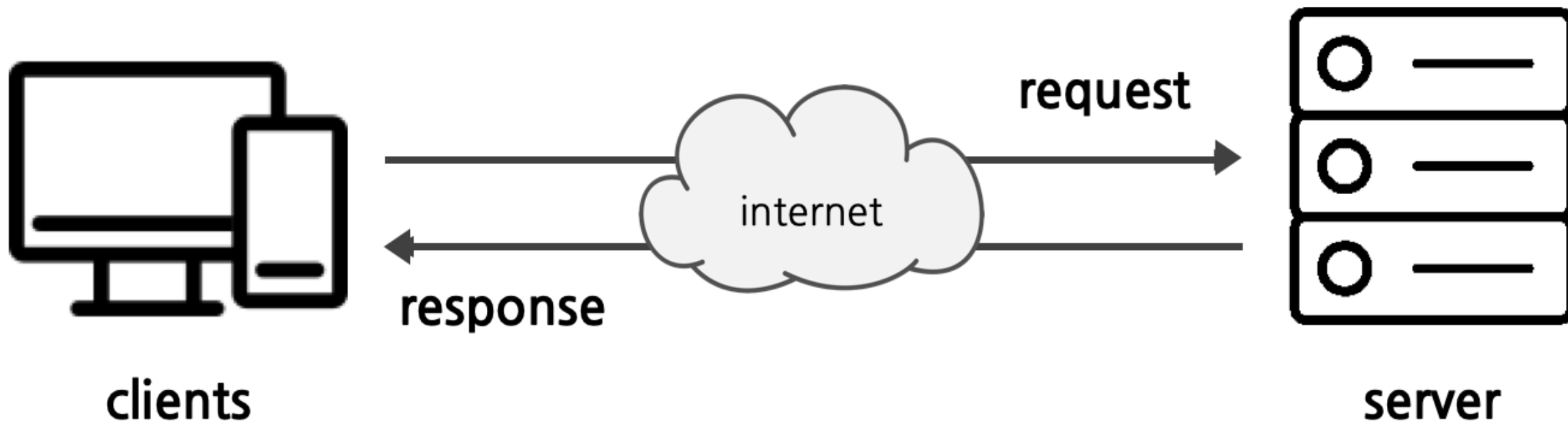
- 현재 쌓여있는 History stack에서 앞 또는 뒤로 이동할 수 있는 method.
- 마치 Browser의 **뒤로가기**, **앞으로 가기** 기능과 동일
- `$router.go(1)` // 한 단계 앞으로 이동
- `$router.go(-1)` // 한 단계 뒤로 이동
- `$router.go(3)` // 3 단계 앞으로 이동



Lab : Programmatic Navigation



Vue HTTP 통신



Vue HTTP 통신 (Cont.)

■ Vue Resource

- <https://github.com/pagekit/vue-resource>
- 처음에는 공식 Vue Library였으나 2016년 말 공식적으로 Evan You가 지원을 중단하기로 결정.
- 기존에 관리했던 PageKit 팀으로 이전됨.
- CDN
 - `<script src="https://cdn.jsdelivr.net/npm/vue-resource@1.5.1"></script>`

Vue HTTP 통신 (Cont.)

```
8     <script src="https://unpkg.com/vue@2.6.10/dist/vue.js"></script>
9     <script src="https://cdn.jsdelivr.net/npm/vue-resource@1.5.1"></script>
10  </head>
11  <body>
12    <div id="app">
13      <button v-on:click="getdata">Get User</button>
14    </div>
15    <script>
16      let vm = new Vue({
17        methods : {
18          getdata : function(){
19            this.$http.get('https://jsonplaceholder.typicode.com/users')
20              .then(function(response){
21                console.log(response);
22                //console.log(JSON.parse(response.data));
23              })
24              .catch(function(error){
25                console.log(error);
26              });
27          }
28        }
29      }).$mount('#app')
30    </script>
```

Vue HTTP 통신 (Cont.)

Get User

The screenshot shows a web browser's developer console with the following content:

- Elements | **Console** | Sources | Network | Performance | Memory | Application | »
- top | Filter | Default levels ▾
- You are running Vue in development mode. vue.js:9058
Make sure to turn on production mode when deploying for production.
See more tips at <https://vuejs.org/guide/deployment.html>
- Live reload enabled. vueroutedemo.html:59
- vueroutedemo.html:21
- ▼ q ⓘ
 - body: Array(10)
 - ▶ 0: {id: 1, name: "Leanne Graham", username: "Bret", email: "Sincere@april.biz", address: {...}, ...}
 - ▶ 1: {id: 2, name: "Ervin Howell", username: "Antonette", email: "Shanna@melissa.tv", address: {...}, ...}
 - ▶ 2: {id: 3, name: "Clementine Bauch", username: "Samantha", email: "Nathan@yesenia.net", address: {...}, ...}
 - ▶ 3: {id: 4, name: "Patricia Lebsack", username: "Karianne", email: "Julianne.OConner@kory.org", address: {...}, ...}
 - ▶ 4: {id: 5, name: "Chelsey Dietrich", username: "Kamren", email: "Lucio_Hettinger@annie.ca", address: {...}, ...}
 - ▶ 5: {id: 6, name: "Mrs. Dennis Schulist", username: "Leopoldo_Corkery", email: "Karley_Dach@jaspe.io", address: {...}, ...}
 - ▶ 6: {id: 7, name: "Kurtis Weissnat", username: "Elwyn.Skiles", email: "Telly.Hoeger@billy.biz", address: {...}, ...}
 - ▶ 7: {id: 8, name: "Nicholas Runolfsson", username: "Maxime_Nienow", email: "Sherwood@rosamond.me", address: {...}, ...}
 - ▶ 8: {id: 9, name: "Glenn Reichert", username: "Delphine", email: "Chaim_McDermott@dana.io", address: {...}, ...}
 - ▶ 9: {id: 10, name: "Clementina DuBuque", username: "Moriah.Stanton", email: "Rey.Padberg@karina.biz", address: {...}, ...}
 - length: 10
 - __proto__: Array(0)
 - bodyText: "[{"id": 1, "name": "Leanne Graham", "username": "Bret", "email": "Sincere@april.biz", "address": {"street": "Kulas Street", "suite": "890", "city": "New York", "zipcode": "10001"}, "phone": "(555) 592-3400"}]"
 - headers: I {map: {...}}
 - ok: true
 - status: 200
 - statusText: ""
 - url: "https://jsonplaceholder.typicode.com/users"
 - data: (...)
 - __proto__: Object

Vue HTTP 통신 (Cont.)

■ Axios

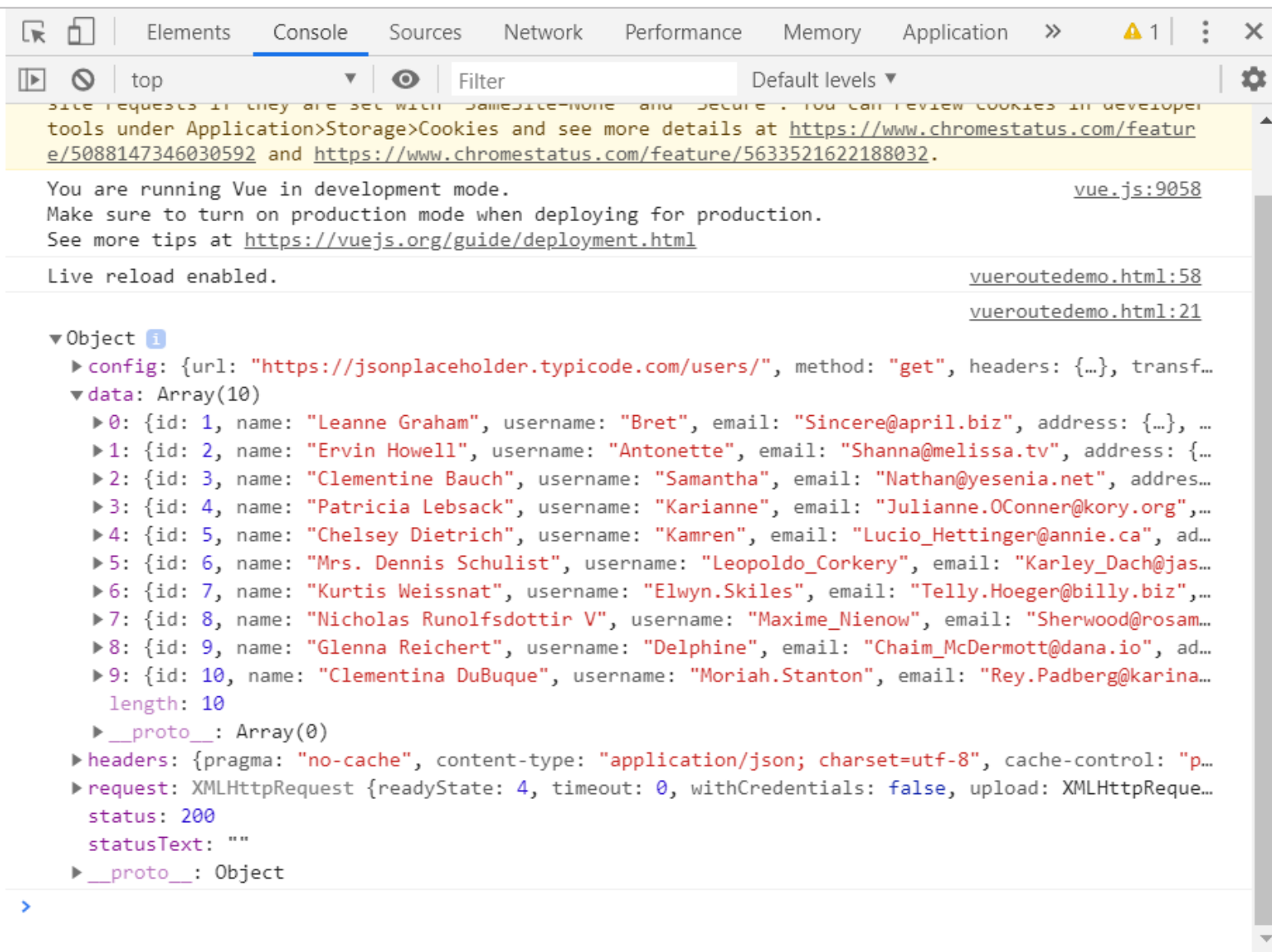
- Promise based HTTP client for the browser and node.js
- Vue에서 가장 많이 사용하고 있는 HTTP 통신 Library.
- Vue에서 권고하는 HTTP 통신 Library
- Promise 기반
- 문서화가 잘되어 있고 API가 다양함.
- <https://github.com/axios/axios>
- 설치
 - CDN과 NPM
- CDN
 - `<script src="https://unpkg.com/axios/dist/axios.min.js"></script>`
- NPM
 - `$ npm install axios`

Vue HTTP 통신 (Cont.)

```
8     <script src="https://unpkg.com/vue@2.6.10/dist/vue.js"></script>
9     <script src="https://unpkg.com/axios/dist/axios.min.js"></script>
10  </head>
11  <body>
12    <div id="app">
13      <button v-on:click="getdata">Get User</button>
14    </div>
15    <script>
16      let vm = new Vue({
17        methods : {
18          getdata : function(){
19            axios.get('https://jsonplaceholder.typicode.com/users')
20              .then(function(response){
21                console.log(response);
22              })
23              .catch(function(error){
24                console.log(error);
25              });
26          }
27        }
28      }).$mount('#app')
29    </script>
```

Vue HTTP 통신 (Cont.)

Get User



The screenshot shows a web browser's developer console with the 'Console' tab selected. The console displays several messages from a Vue.js application:

- A yellow banner message about cookies and development mode.
- A message: "You are running Vue in development mode. Make sure to turn on production mode when deploying for production. See more tips at <https://vuejs.org/guide/deployment.html>"
- A message: "Live reload enabled."
- A log entry for an object, expanded to show details of a GET request to `https://jsonplaceholder.typicode.com/users/`.

The expanded log entry shows the following structure:

- `config`: {url: "https://jsonplaceholder.typicode.com/users/", method: "get", headers: {...}, transf...}
- `data`: Array(10)
 - 0: {id: 1, name: "Leanne Graham", username: "Bret", email: "Sincere@april.biz", address: {...}, ...}
 - 1: {id: 2, name: "Ervin Howell", username: "Antonette", email: "Shanna@melissa.tv", address: {...}, ...}
 - 2: {id: 3, name: "Clementine Bauch", username: "Samantha", email: "Nathan@yesenia.net", address: {...}, ...}
 - 3: {id: 4, name: "Patricia Lebsack", username: "Karianne", email: "Julianne.OConner@kory.org", ...}
 - 4: {id: 5, name: "Chelsey Dietrich", username: "Kamren", email: "Lucio_Hettinger@annie.ca", address: {...}, ...}
 - 5: {id: 6, name: "Mrs. Dennis Schulist", username: "Leopoldo_Corkery", email: "Karley_Dach@jas...}
 - 6: {id: 7, name: "Kurtis Weissnat", username: "Elwyn.Skiles", email: "Telly.Hoeger@billy.biz", ...}
 - 7: {id: 8, name: "Nicholas Runolfsson", username: "Maxime_Nienow", email: "Sherwood@rosam...}
 - 8: {id: 9, name: "Glenna Reichert", username: "Delphine", email: "Chaim_McDermott@dana.io", address: {...}, ...}
 - 9: {id: 10, name: "Clementina DuBuque", username: "Moriah.Stanton", email: "Rey.Padberg@karina...}
- `length`: 10
- `__proto__`: Array(0)
- `headers`: {pragma: "no-cache", content-type: "application/json; charset=utf-8", cache-control: "p..."}
- `request`: XMLHttpRequest {readyState: 4, timeout: 0, withCredentials: false, upload: XMLHttpRequest...}
- `status`: 200
- `statusText`: ""
- `__proto__`: Object