

# 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.

# Vue Router

- Vue Library를 이용하여 Single Page Application을 구현할 때 사용하는 Library.
- Vue Router Installation
  - CDN 방식
  - NPM 방식
- CDN 방식

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

- NPM 방식

```
npm install vue-router
```

# Vue Router 등록

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

```
// 라우터 인스턴스 생성
var router = new VueRouter({
  // 라우터 옵션
})

// 인스턴스에 라우터 인스턴스를 등록
new Vue({
  router: router
})
```

# Vue Router Options

- Vue Router를 등록 후, Router에 Option을 정의한다.
- 대부분의 SPA에서는 아래와 같이 2개 Option을 필수로 지정한다.
  - **routes**
    - Routing 할 URL과 Component 값 지정
  - **mode**
    - URL의 Hash 값 제거 속성

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

## 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>** 입니다.

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

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



# Lab

```
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-link to="/top">최상위 페이지</router-link>
14         <router-link to="/users">사용자 목록 페이지</router-link>
15         <router-view></router-view>
16     </div>
```

# Lab (Cont.)

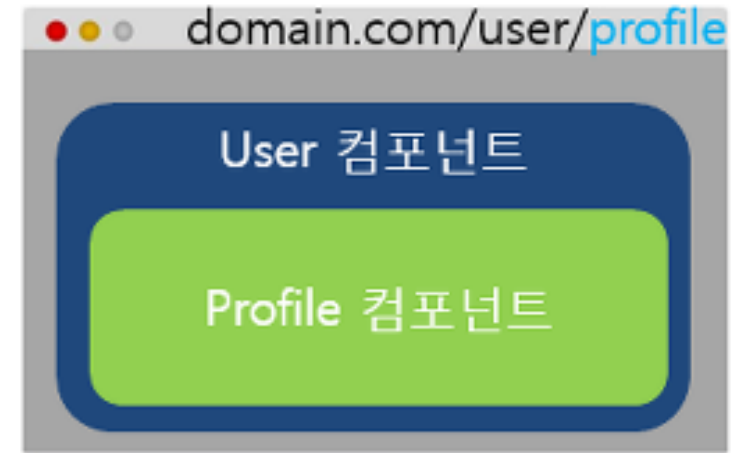
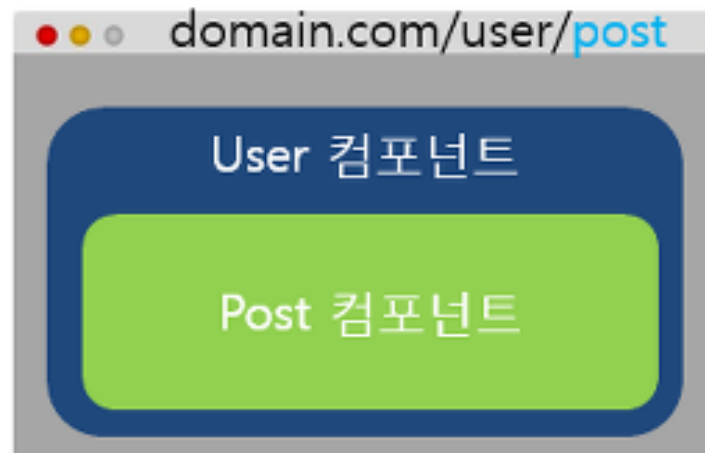
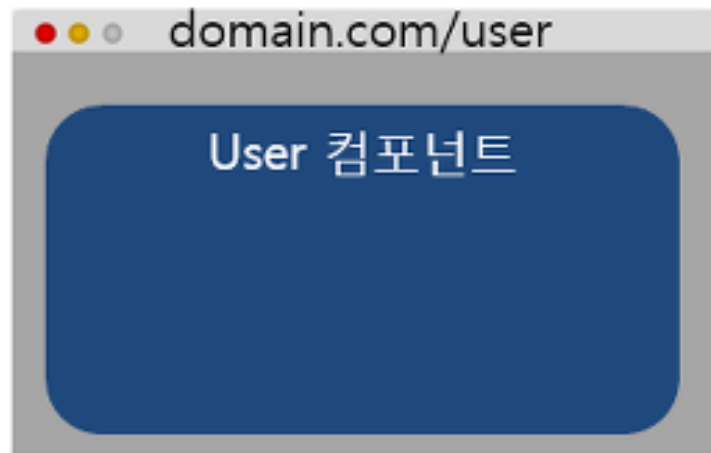
```
17 <script>
18   //Route 옵션을 지정해서 라우터 인스턴스를 생성
19   let router = new VueRouter({
20     routes : [
21       {
22         path : '/top',
23         component : {
24           template : '<div>최상위 페이지</div>'
25         }
26       },
27       {
28         path : '/users',
29         component : {
30           template : '<div>사용자 목록 페이지</div>'
31         }
32       }
33     ]
34   })
35   let app = new Vue({
36     router : router
37   }).$mount('#app');
38 </script>
```

← → ↻ ⓘ 127.0.0.1:5500/vuerouter.html#/users

최상위 페이지 사용자 목록 페이지  
사용자 목록 페이지

# Nested Router

- Router로 Page 이동할 때 최소 2개 이상의 Component를 화면에 나타낼 수 있다.
- 상위 Component 1개에 하위 Component 1개를 포함하는 구조



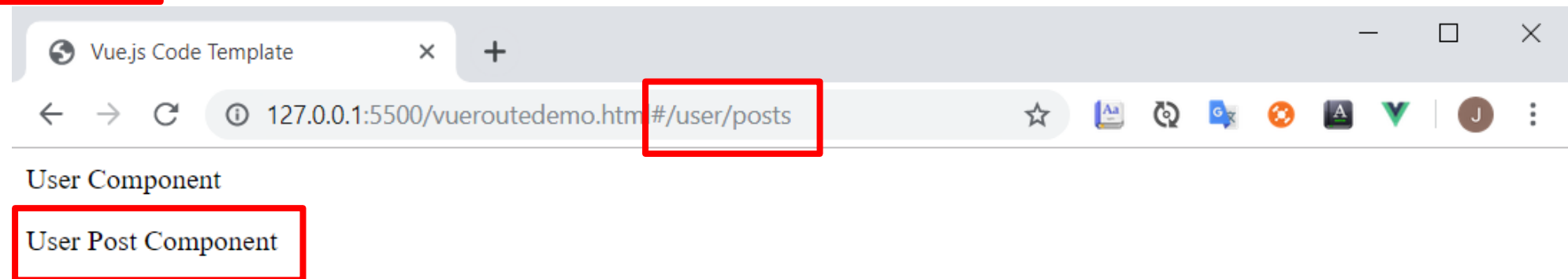
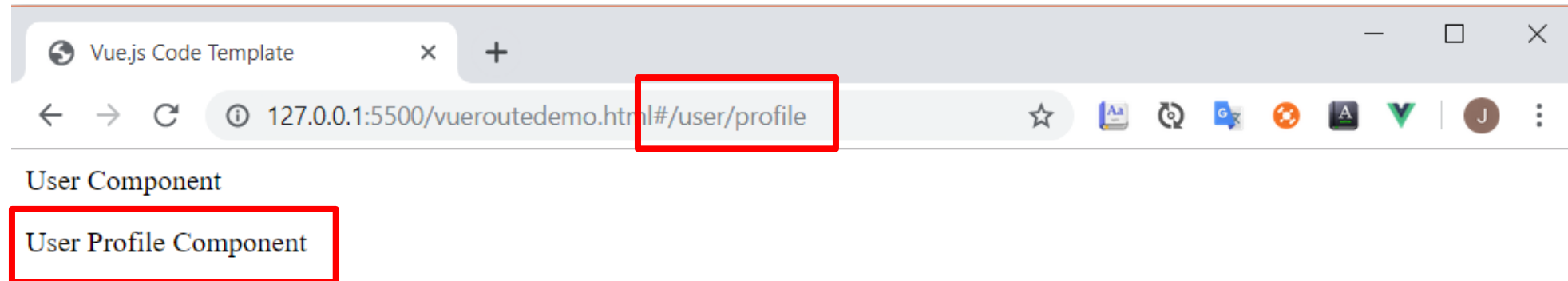
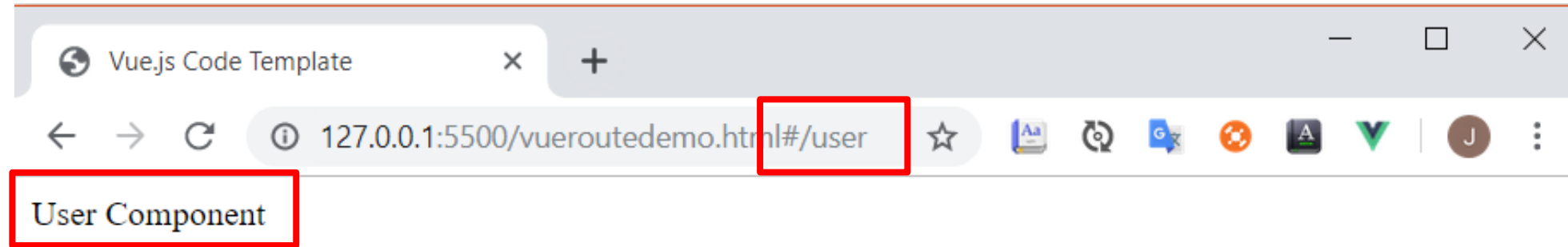
## Nested Router (Cont.)

```
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     });
```

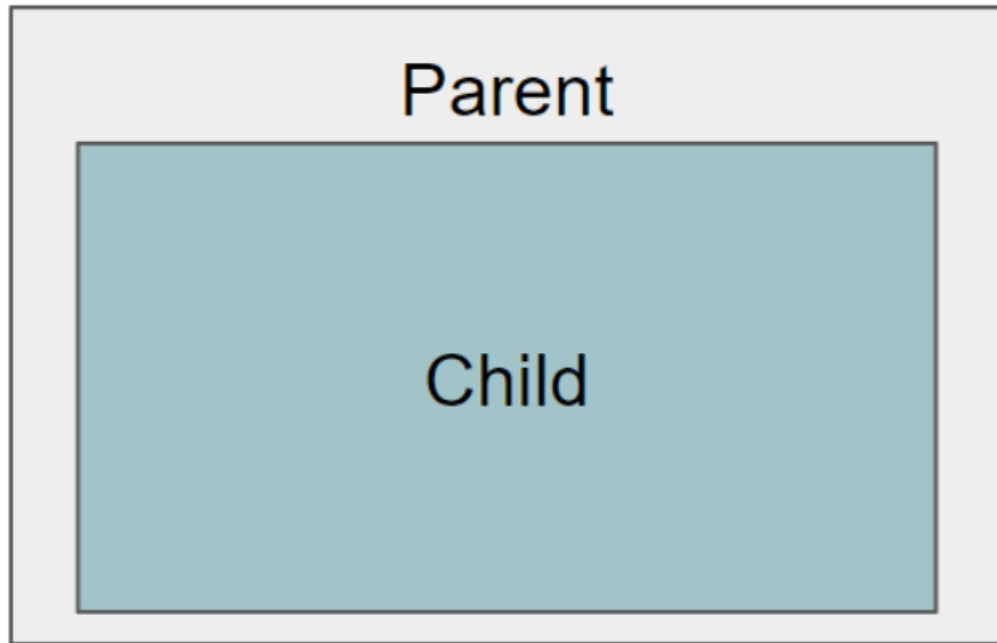
# Nested Router (Cont.)



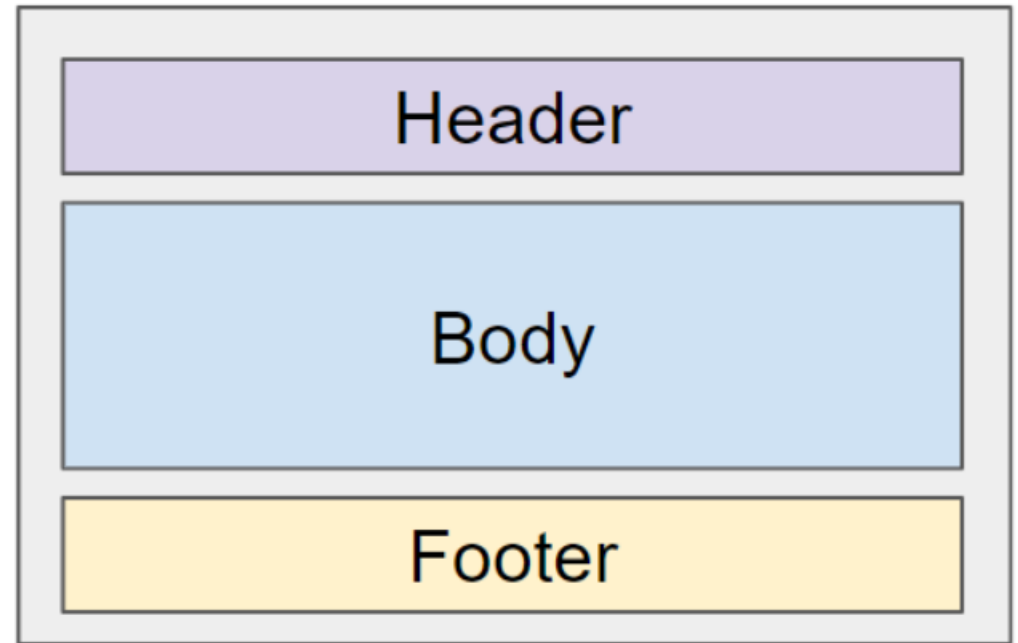
# Named View

- 특정 Page로 이동했을 때 여러 개의 Component를 동시에 표시하는 Routing 방식

네스티드 라우터 구조

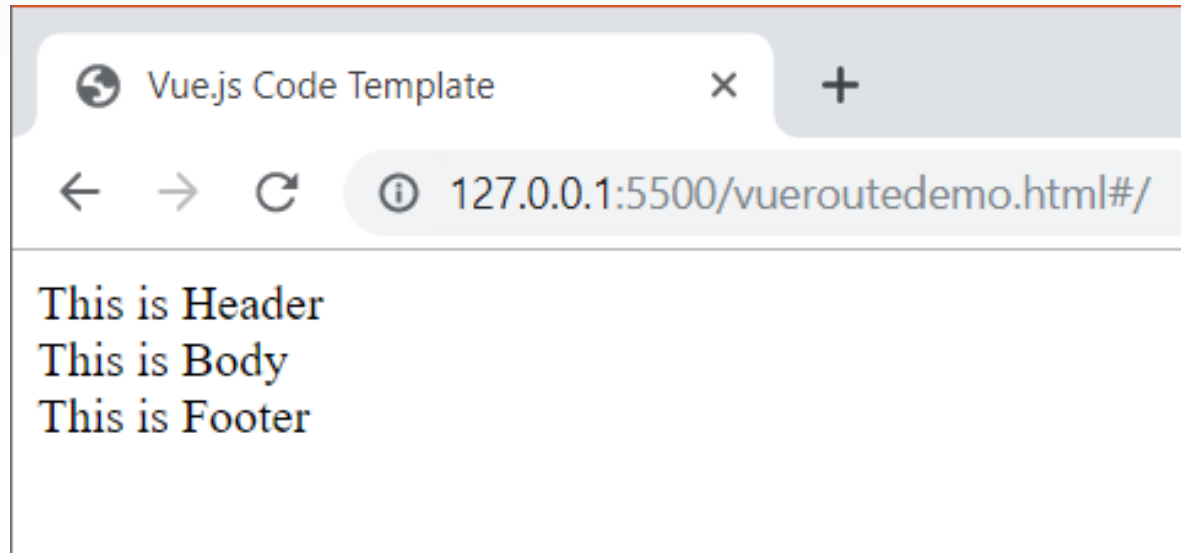


네임드 뷰 구조



## 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.)

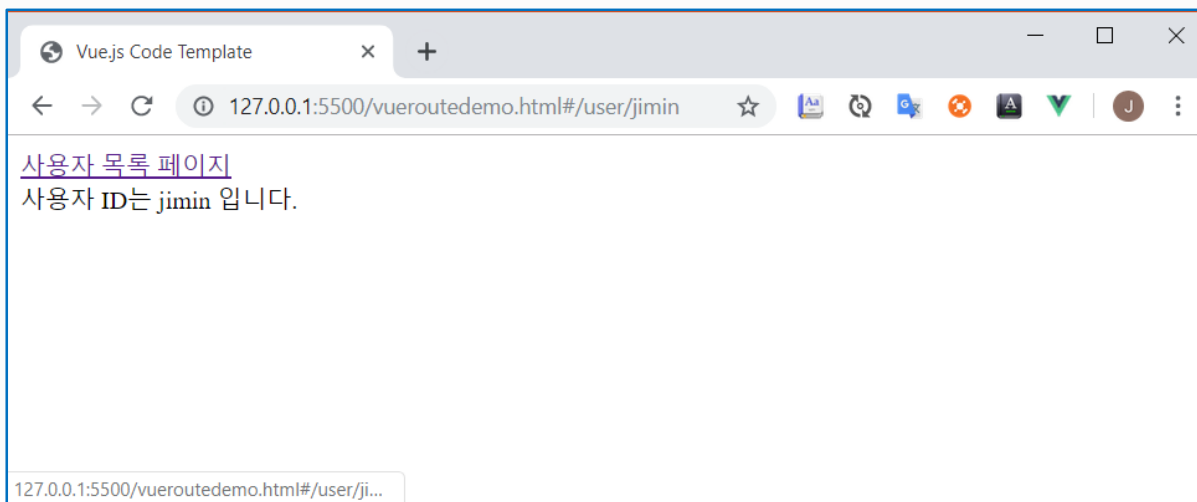
```
17     <script>
18         var Body = { template : '<div>This is Body</div>' };
19         var Header = { template : '<div>This is Header</div>' };
20         var Footer = { template : '<div>This is Footer</div>' };
21         var router = new VueRouter({
22             routes : [
23                 {
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>
```

# URL Parameter를 처리하는 방법과 Pattern Matching

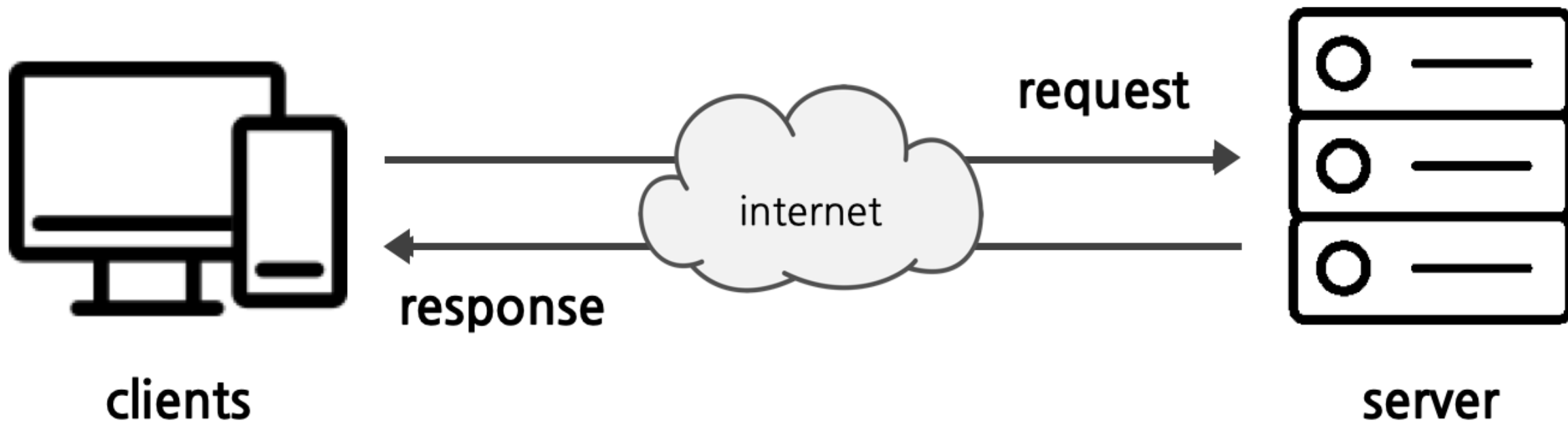
- SPA는 접근 대상 URL의 Pattern Matching을 통해 Parameter를 전달하는 경우가 많다.
- 사용자 상세 정보 Page → **/user/:userid**
- URL 경로에 **:**을 붙여 Pattern 작성.
- URL에서 이 Pattern과 일치하는 Parameter는 Component에서 **\$route.params**의 속성 중 Pattern에 사용한 Parameter 이름과 같은 속성명으로 접근 가능.

# URL Parameter를 처리하는 방법과 Pattern Matching (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>
```



# 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         var 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 'Console' tab selected. The console displays a message about running Vue in development mode, followed by a successful GET request to a JSONPlaceholder API endpoint. The response is a JSON object with a 'body' array containing 10 user objects, a 'status' of 200, and a 'url' of 'https://jsonplaceholder.typicode.com/users'.

```
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

▼ 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": "Apt. 960", "city": "New York", "zipcode": "10014-2947"}, "phone": "(555) 930-9291"}]"
  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`

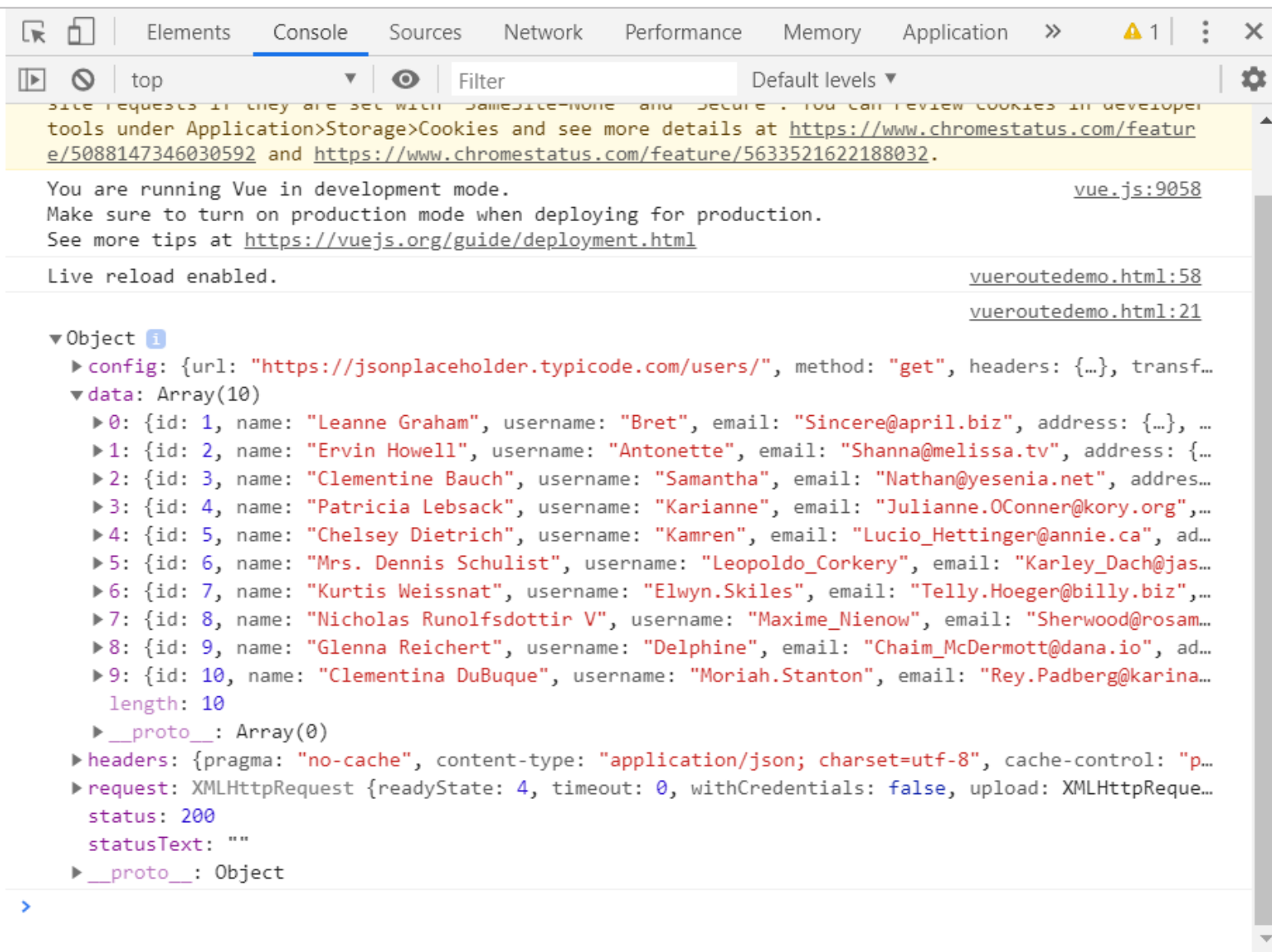


# Vue HTTP 통신 (Cont.)

```
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      var 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>
30  </body>
```

# 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 the Vue.js development environment, including a warning about cookies, a message about development mode, and a confirmation that live reload is enabled. Below these, a log entry for an HTTP GET request is expanded, showing the request configuration and a successful response with a 200 status code. The response data is an array of 10 user objects from the JSONPlaceholder API.

```
Object
  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", 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@jas..."}
    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@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
```