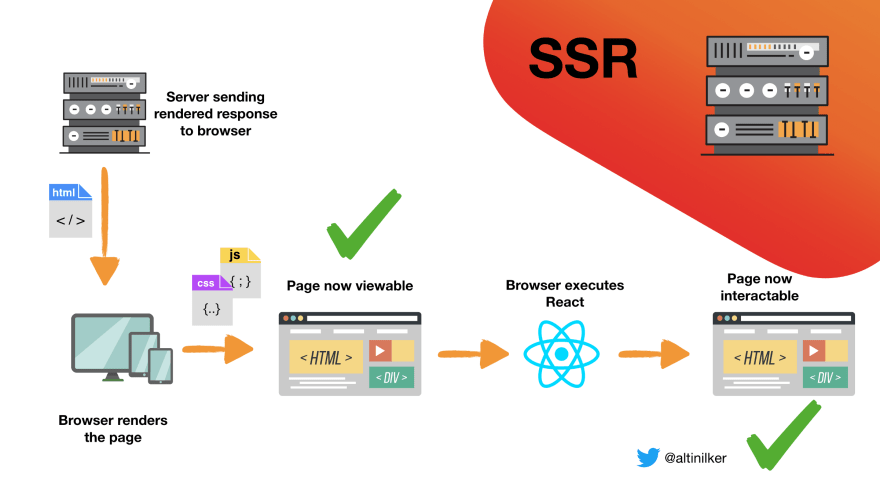
**1. 서버사이드 렌더링이란 무엇인가**

**SSR (Server-Side rendering)**

출처: https://dev.to/itnext/server-side-rendering-with-react-redux-and-react-router-2en7

-  User가 페이지에 처음 접근하였을때, 페이지를 서버측에서 먼저 간단하게 띄워주고, 완전하게 렌더링된 페이지를 클라이언트에게 다시 보내주는 기법이다.

- 사용자의 요청시(인터렉션이 있을 때)마다 새로고침이 일어나며, 서버에 새로운 페이지에 대한 요청을 한다.

장점 :

**SEO (검색엔진 최적화)**

- 리액트, 혹은 view 렌더링이 자바스크립트 위주로 돌아가는 프로젝트는, 자바스크립트 엔진이 돌아가지 않으면 원하는 정보를 표시해주지 않는다. 이렇게 클라이언트 렌더링만 될 경우엔 검색엔진 크롤러가 어플리케이션이 지닌 데이터들을 제대로 수집하지 못해 빈 페이지로 인식한다. 검색엔진에 제대로 노출이 되지 않으니 웹페이지에의 유입이 줄어드는 결과가 발생한다.

- 구글 검색엔진은 크롤러에 자바스크립트 엔진이 내장되어 있어서 상관없지만 네이버, 다음 등의 검색엔진이 사이트를 제대로 크롤링 하게 지원해야 한다면, 서버사이드 렌더링을 따로 구현해야 한다.

- SSR은 view를 서버에서 전부 렌더링해 내려주어 html에 모든 컨텐츠가 저장되어 있다. -> 검색엔진에 최적화가 된다.

**성능 개선**

- 첫 렌더링된 html을 클라이언트에게 전달을 해주기때문에 view를 보기까지 초기 로딩속도가 빠르다. JS파일을 불러오고 렌더링 작업이 완료되기 전에도 유저가 사이트의 컨텐츠를 이용 할 수 있어다. JS파일을 전부 다운받기 전까지는 제대로 동작하지 않지만, 사용자는 빠르다 느낄 수 있다.

**보안**

사용자에 대한 정보를 서버 측에서 세션으로 관리를 해 CSR방식보다 안전하다.

단점 :

**프로젝트의 복잡도**

- 단순히 렌더링만 하는것은 그렇게 큰 문제는 아니지만, 리액트 라우터, 리덕스 등의 라이브러리와 함께 연동해서 사용하면서, 서버에서 데이터를 가져와서 렌더링을 해줘야하는 경우엔 조금 어려워질수도 있다.

**성능 악화의 가능성**

- 클라이언트에서 초기렌더링을 서버측에서 대신 하여 클라이언트의 부담을 없애주기는 하지만, 렌더링을 위한 서버 자원을 사용해 불필요하게 트래픽을 낭비할 수 있다. 서버사이드 렌더링을 하게 될 때는, ReactDOMServer.renderToString

함수를 사용하게 되는데, 이 함수는 동기적으로 작동한다. 그래서, 렌더링하는동안은 이벤트루프가 막히게 된다. 그 뜻은, 렌더링이 되는 동안은, 다른 작업을 못한다는 뜻이다. 만약 나의 프로젝트를 렌더링하는데 250ms가 걸린다면 1초에 4개의 요청밖에 처리하지 못한다는 의미이다. 치명적이다.

(하지만 동일한 페이지를 특정기간 동안 캐싱하여 성능을 최적화 하거나, 써드파티 라이브러리를 통하여 비동기식으로 작동하게끔 코드를 작성하여 이에 대처할 수 있다.)

**도움이 된 정보들**

칼럼:

<https://velopert.com/3425>

[react-router :: 3장. 서버사이드 렌더링 | VELOPERT.LOG](https://velopert.com/3425" \t "_blank)

[이 튜토리얼은 3개의 포스트로 나뉘어진 이어지는 강좌입니다. 목차를 확인하시려면 여기를 참고하세요. 3장. 서버사이드 렌더링 이번 장에서는 리액트 어플리케이션을 서버사이드 렌더링 하는](https://velopert.com/3425" \t "_blank)

[velopert.com](https://velopert.com/3425" \t "_blank)

<https://medium.com/@donggyu9410/%EC%99%9C-%EC%84%9C%EB%B2%84-%EC%82%AC%EC%9D%B4%EB%93%9C-%EB%A0%8C%EB%8D%94%EB%A7%81%EC%9D%B4-%ED%95%84%EC%9A%94%ED%95%A0%EA%B9%8C%EC%9A%94-eb41a594f94b>

[왜 서버 사이드 렌더링이 필요할까요?](https://medium.com/@donggyu9410/%EC%99%9C-%EC%84%9C%EB%B2%84-%EC%82%AC%EC%9D%B4%EB%93%9C-%EB%A0%8C%EB%8D%94%EB%A7%81%EC%9D%B4-%ED%95%84%EC%9A%94%ED%95%A0%EA%B9%8C%EC%9A%94-eb41a594f94b" \t "_blank)

[SPA 을 개발할때에 있어서 왜 서버 사이드 렌더링이 필요하고, 서버 사이드 렌더링 부재시의 문제점과, 서버 사이드 렌더링의 기본적인 동작 방식에 대해서 알아봐요.](https://medium.com/@donggyu9410/%EC%99%9C-%EC%84%9C%EB%B2%84-%EC%82%AC%EC%9D%B4%EB%93%9C-%EB%A0%8C%EB%8D%94%EB%A7%81%EC%9D%B4-%ED%95%84%EC%9A%94%ED%95%A0%EA%B9%8C%EC%9A%94-eb41a594f94b" \t "_blank)

[medium.com](https://medium.com/@donggyu9410/%EC%99%9C-%EC%84%9C%EB%B2%84-%EC%82%AC%EC%9D%B4%EB%93%9C-%EB%A0%8C%EB%8D%94%EB%A7%81%EC%9D%B4-%ED%95%84%EC%9A%94%ED%95%A0%EA%B9%8C%EC%9A%94-eb41a594f94b" \t "_blank)

<https://brownbears.tistory.com/411>

[서버 사이드 렌더링(SSR), 클라이언트 사이드 렌더링(CSR)](https://brownbears.tistory.com/411" \t "_blank)

[SSR 이란? Server Side Rendering 의 약어로써 단어 그대로 서버에서 렌더링을 작업합니다. 기존에 존재하던 방식으로 사용자가 웹페이지에 접근할 때, 서버에 페이지에 대한 요청을 하며 서버에서는 ht](https://brownbears.tistory.com/411" \t "_blank)

[brownbears.tistory.com](https://brownbears.tistory.com/411" \t "_blank)

<https://medium.com/@donggyu9410/%EA%B0%80%EC%9E%A5-%EC%89%AC%EC%9A%B4-%EB%B0%A9%EB%B2%95%EC%9C%BC%EB%A1%9C-%EB%A6%AC%EC%95%A1%ED%8A%B8%EC%97%90%EC%84%9C-%EC%84%9C%EB%B2%84%EC%82%AC%EC%9D%B4%EB%93%9C-%EB%A0%8C%EB%8D%94%EB%A7%81-%EC%9D%B4%ED%95%B4%ED%95%98%EA%B8%B0-966702610664>

[가장 쉬운 방법으로 리액트에서 서버사이드 렌더링 이해하기](https://medium.com/@donggyu9410/%EA%B0%80%EC%9E%A5-%EC%89%AC%EC%9A%B4-%EB%B0%A9%EB%B2%95%EC%9C%BC%EB%A1%9C-%EB%A6%AC%EC%95%A1%ED%8A%B8%EC%97%90%EC%84%9C-%EC%84%9C%EB%B2%84%EC%82%AC%EC%9D%B4%EB%93%9C-%EB%A0%8C%EB%8D%94%EB%A7%81-%EC%9D%B4%ED%95%B4%ED%95%98%EA%B8%B0-966702610664" \t "_blank)

[서버 사이드 렌더링이란 페이지를 서버측에서 먼저 간단하게 띄워주고, 완전하게 렌더링된 페이지를 클라이언트에게 다시 보내주는 기법을 말합니다.](https://medium.com/@donggyu9410/%EA%B0%80%EC%9E%A5-%EC%89%AC%EC%9A%B4-%EB%B0%A9%EB%B2%95%EC%9C%BC%EB%A1%9C-%EB%A6%AC%EC%95%A1%ED%8A%B8%EC%97%90%EC%84%9C-%EC%84%9C%EB%B2%84%EC%82%AC%EC%9D%B4%EB%93%9C-%EB%A0%8C%EB%8D%94%EB%A7%81-%EC%9D%B4%ED%95%B4%ED%95%98%EA%B8%B0-966702610664" \t "_blank)

[medium.com](https://medium.com/@donggyu9410/%EA%B0%80%EC%9E%A5-%EC%89%AC%EC%9A%B4-%EB%B0%A9%EB%B2%95%EC%9C%BC%EB%A1%9C-%EB%A6%AC%EC%95%A1%ED%8A%B8%EC%97%90%EC%84%9C-%EC%84%9C%EB%B2%84%EC%82%AC%EC%9D%B4%EB%93%9C-%EB%A0%8C%EB%8D%94%EB%A7%81-%EC%9D%B4%ED%95%B4%ED%95%98%EA%B8%B0-966702610664" \t "_blank)

<https://junsday.tistory.com/44>

[서버사이드렌더링(Server-Side-Rendering)](https://junsday.tistory.com/44" \t "_blank)

[서버사이드렌더링(Server-Side-Rendering)이란? - Rendering : 어떠한 웹 페이지 접속시, 그 페이지를 화면에 그려주는 것(요청받은 내용을 브라우저 화면에 표시하는것) - 로더가(Loader)가 서버로부터 정보](https://junsday.tistory.com/44" \t "_blank)

[junsday.tistory.com](https://junsday.tistory.com/44" \t "_blank)

**2. SPA란 무엇이고 어떤 장점이 있는가.**

### SPA (Single-Page Application)

- 단일 페이지로 구성된 웹 어플리케이션을 말한다.

|  |  |
| --- | --- |
| https://k.kakaocdn.net/dn/9T0NW/btqFrDK19LT/nc6Wfx5w3TyfsyhtDRLKv1/img.png | |
| [출처: https://www.digitalclaritygroup.com/single-page-application-make-sense/](https://www.digitalclaritygroup.com/single-page-application-make-sense/) | |
| 화면 구성에 필요한 모든 HTML을 클라이언트가 갖고 있고 서버사이드에는 필요한 데이터를 요청하고 JSON으로 받기 때문에 기존의 어플리케이션에 비해 화면을 구성하는 속도가 빠르다. (CSR(Client-side rendering) 방식) | 기존 어플리케이션은 화면이동 시에 화면 이동에 필요한 HTML을 서버사이드에서 받아서 처음부터 다시 로딩하기 때문에 시간이 걸린다. |

장점:

- 하나하나 화면 전체를 렌더링 할 필요가 없기 때문에 화면 이동이 빠르다.

- 화면에 필요한 부분의 데이터만 받아서 렌더링 하기 때문에 처리과정이 효율적이다.

단점 :

-  유저가 실제로 방문하지 않을수도 있는 페이지의 렌더링과 관련한 스크립트도 불러오기 때문에, 앱의 규모가 커지면 자바스크립트 파일 사이즈가 너무 커져 초기 로딩 속도가 길어진다.

(이는 Code Splitting 을 사용하여 라우트 별로 파일들을 나눠, 트래픽과 로딩속도를 개선 할 수 있다.)

- 어플리케이션을 구현하는데 보다 시간이 걸리며 복잡하다.

**도움이 된 정보들**

칼럼:

<https://velopert.com/3417>

[react-router :: 1장. 리액트 라우터 사용해보기 | VELOPERT.LOG](https://velopert.com/3417" \t "_blank)

[이 튜토리얼은 3개의 포스트로 나뉘어진 이어지는 강좌입니다. 목차를 확인하시려면 여기를 참고하세요. SPA 란? Single Page Application (싱글 페이지 어플리케이션) 의 약자입니다. 말 그대로, 페이지](https://velopert.com/3417" \t "_blank)

[velopert.com](https://velopert.com/3417" \t "_blank)

<https://velog.io/@josworks27/SPA-%EA%B0%9C%EB%85%90>

[SPA(Single Page Application)에 기초 개념](https://velog.io/@josworks27/SPA-%EA%B0%9C%EB%85%90" \t "_blank)

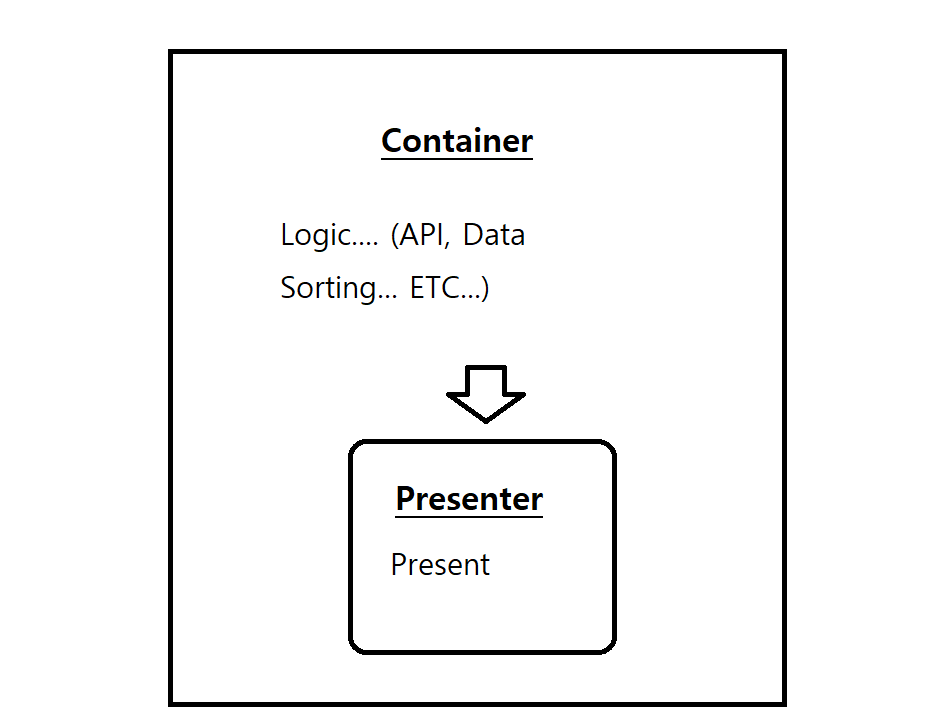
[단일 페이지로 구성된 웹 어플리케이션을 말한다. SPA는 화면이동 시에 필요한 데이터를 서버사이드에서 HTML으로 전달받지 않고(서버사이드 렌더링 X), 필요한 데이터만 서버로부터 JSON으로 전달](https://velog.io/@josworks27/SPA-%EA%B0%9C%EB%85%90" \t "_blank)

[velog.io](https://velog.io/@josworks27/SPA-%EA%B0%9C%EB%85%90" \t "_blank)

**3. 리액트에 사용되는 디자인패턴은 어떤 것들이 있는가. 각각 설명하시오.**

**Container Presenter Pattern**

데이터처리와 데이터출력을 분리 (logic과 presenter를 분리)



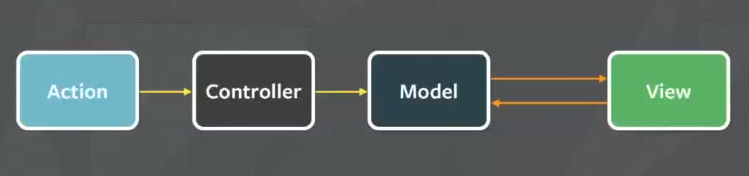
## **Container Component**

1. Redux action dispatch를 담당한다.
2. dispatch에는 Ajax call이 될 수 있고, store update가 될 수 있다.
3. Redux에 선언된 Action들을 import받고 dispatch 함수를 반환하여 CC의 props로 내려준다.

## **Presentational Component**

1. Redux와 Connect 하지는 않고 모두 CC로부터 props를 받아 처리한다.
2. FC도 처리해야하는 View state가 많거나, 자체적으로 React-Router (Sub Header)가 필요한 경우 규모가 커지는 경우가 있다.
3. 정확하게 FC는 View 용도로만 사용을 한다.

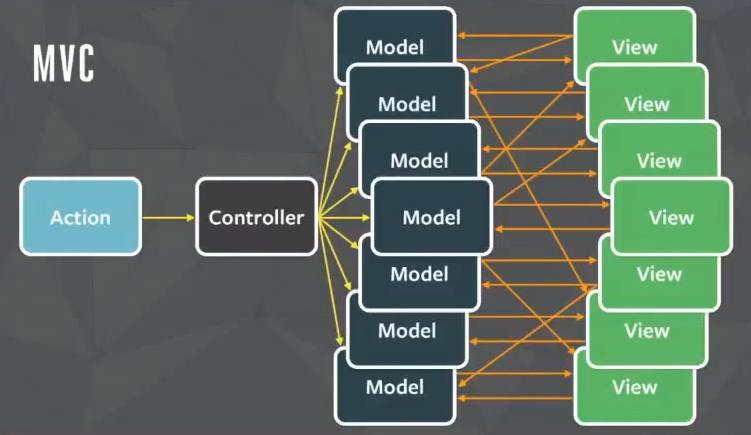
### MVC 디자인 패턴



**-**  Controller, Model, View 3가지 개념으로 이루어져 있다.

- **Action** 이 입력되면, **Controller**은 **Model**이 지니고 있는 데이터를 조회하거나 업데이트하며, 이 변화는 **View** 에 반영되는 구조이다. 또한, **View**에서 **Model**의 데이터에 접근할 수도 있다.

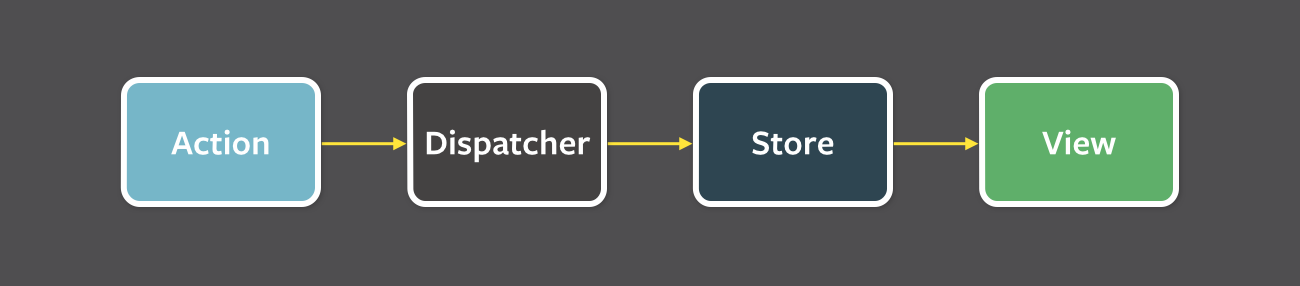
Model과 View가 늘어난다면?



어떤 모델이 뷰를 건드리고, 그 뷰가 어떤 또 다른 모델을 건드리고.. 무한반복.. 화살표들을 본다고 해서 그걸 발견해내기 쉽지 않다.

### FLUX 패턴

위 문제를 해결하기 위해서 FLUX 라는 디자인 패턴이 만들어졌다.



- 시스템에서 어떠한 **Action** 을 받았을 때, **Dispatcher**가 받은 **Action**들을 통제하여 **Store**에 있는 데이터를 업데이트하고, 변동된 데이터가 있으면 **View** 에 리렌더링한다.

- **View**에서 **Dispatcher**로 **Action**을 보낼 수도 있다.



**Dispatcher**은 작업이 중첩되지 않도록 어떤 **Action**이 **Dispatcher**를 통하여

**Store**에 있는 데이터를 처리하고, 그 작업이 끝날 때 까지 다른 **Action**들을 대기시킨다.

<https://velopert.com/1225>

<https://medium.com/@lyhlg0201/about-frontend-react-design-pattern-2c387a2ae4fb>