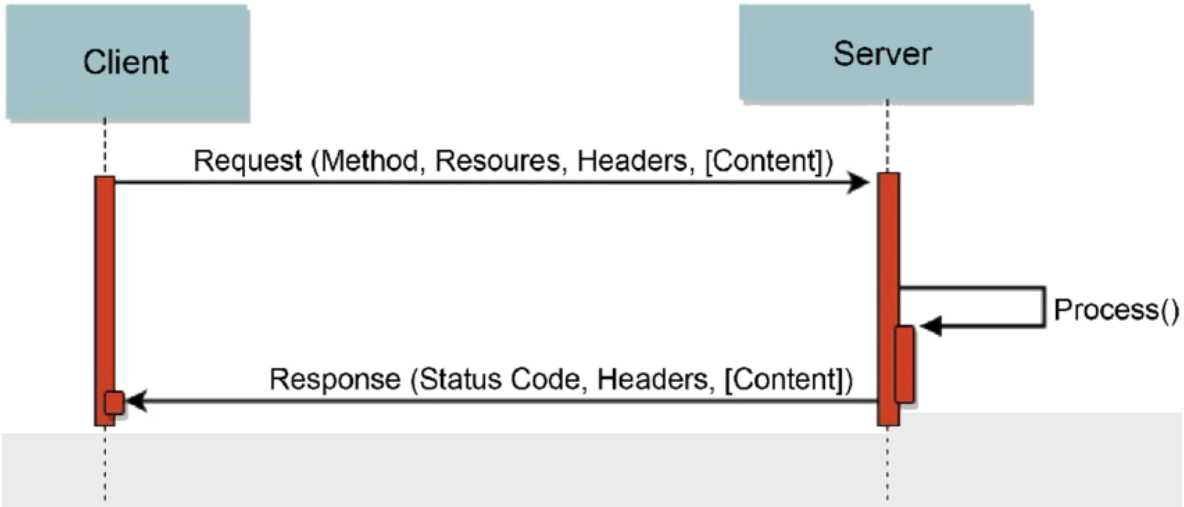
2주차. 실시간 네트워크 통신

> API의 의미를 이해하고 최근 각광받는 REST API에 대해 알아보자

01. 양방향 네트워크 방식

1. 단방향 통신

- 브라우저에서 url을 통해 접속하거나, 버튼을 통해 데이터를 요청/전송하는 경우 http규약을 따라 통신이 시작된다.
- 이때 통신규약을 잘 지키면 SERVER에서는 `성공메시지(200)`을 브라우저를 통해 CLIENT에게 전송해주고 우리에게 웹 페이지를 보여주는 것이다.
- 이때 서버에서는 해당문서에 필요한 모든 정보들을 한 꺼번에 보내준다.
  - html, css, javascript, img, data 등등..
- 때문에 우리가 웹페이지를 보다가 어떤 버튼을 누르면 화면이 다시 처음부터 로딩되는 것을 볼 수 있다.
- 우리가 django로 진행했던 pjt03을 보면, 새로운 페이지에 들어갈 때마다, 생성/수정/삭제 등의 동작을 진행할 때마다 CLIENT와 SERVER는 웹 브라우저를 통해 마치 처음 만난 사람마냥 이 작업을 반복한다.
- 우리가 지난 주에 학습했던 http는 이 통신을 인터넷에서는 어떤 방식으로 하기로 미리 정해둔 약속인 것이다.



2. 실시간 양방향 통신

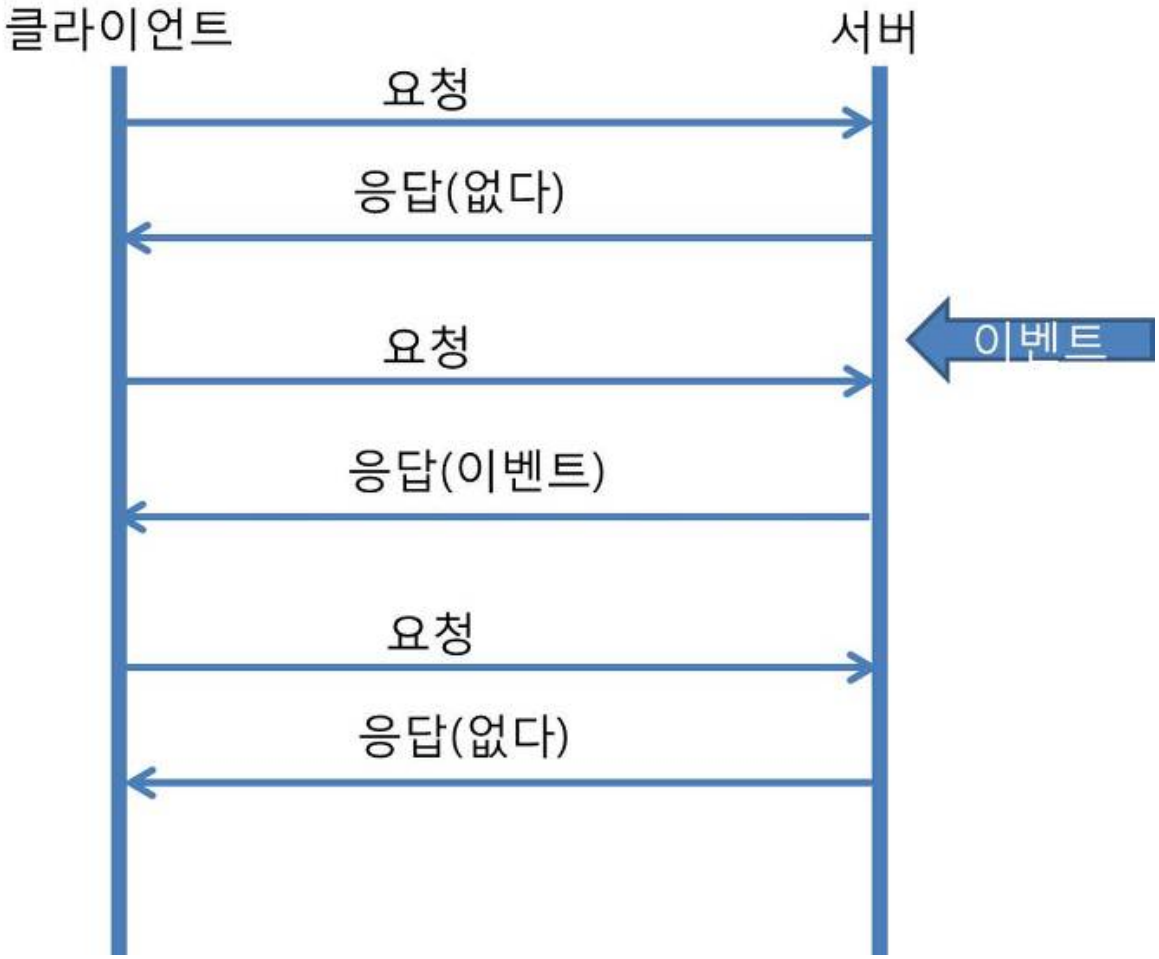
- 실시간 양방향 통신 개념을 알아보자.
- 비유를 들자면, 우리 집과 앞집에 도둑이 자주 드는 우범지역이다. 우리가 앞집과 별로 친하지 않아서 아주 가끔씩 서로 괜찮은지 안부를 묻는다. 이런 경우 그때마다 초인종을 누르고 내가 누구인지 신원을 확인한 후 문을 열어주고 소통을 할 것이다.
- 하지만, 앞집에 엄마가 살아서 서로 문제가 없는지 수시로 문을 두드린다. 그러려면, 나는 우리집 대문 열고 초인종 누르고 안에서 누구인지 확인하러 움직이고, 엄마는 찾아온게 나인지 확인해서 신원 확인되면 대문 열어주고 하는 이러한 과정을 거쳐야 하는데 이는 매우 비효율적일 것이다.
- 그렇다면 이런 과정의 비효율을 줄이기 위한 방법은 무엇이 있을까? 해서 나온 것이 연결형 통신의 개념이다.

```
(앞집, 엄마 = 'SERVER', '아들바보')
(우리집,  나 = 'CLIENT', '효자')
```

1. 나(CLIENT): "변화(EVENT)가 발생했는지 아닌지 모르겠으니 매일 1시간 간격으로 주기적으로 앞집을 살펴봐야지"
2. 나(CLIENT): "힘드니까 6시간에 한 번씩만 갔다와야겠다. 대신 간 김에 한 5분정도 문 열고 기다렸다가 변화(EVENT)가 발생했으면 그 소식을 알아와야지."
3. 나(CLIENT): "아니 엄마 이럴바에는 문 열어두자. 차라리 엄마가 변화(EVENT)가 발생할때마다 소리지르면 내가 찾아갈게"

1번: 폴링 (Polling, Ajax) - "스포츠 문자중계"

- 페이스북의 방식은 새로운 컨텐츠를 계속 보내준다기보다 사용자가 새로고침을 눌렀을 때에 신규 이벤트를 서버로 부터 받아오지만, "좋아요" 버튼을 누르면 화면이 새로고침 되지 않고 바로 데이터가 전송되죠.
- 클라이언트가 평범한 http request를 계속 날려서 이벤트 내용을 전달 받는 방식이다.
- 기존의 http request 방식을 약간의 트릭을 써서 실시간 소통 처럼 활용하는 방식인 것이다.
- 가장 손쉬운 방식이지만 http request에 응답하기 위해 지속적으로 connection을 맺고 끊는 것이 부담스럽다는게 단점이다
- 또, 클라이언트 수가 많아질 수록 부담이 계속해서 가중된다.
- 예를 들어 1초에 2회씩 요청을 보낸다고 생각해보자. 이 웹서비스에 100명의 클라이언트가 접속한다면, 서버는 1초에 200회 접속량을 감당해야 한다.

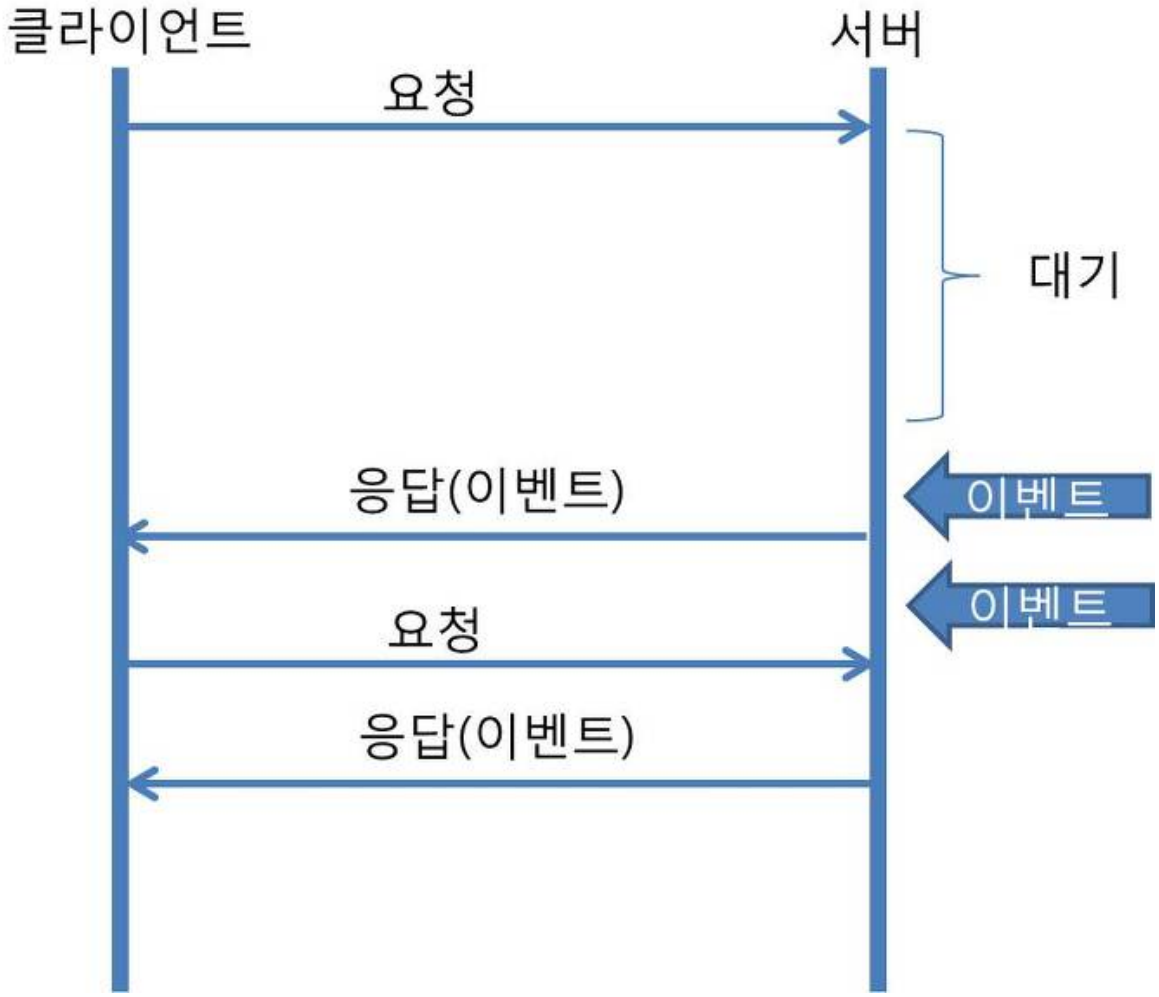


2번: 롱 폴링 (Long Polling) - "그룹채팅"

- 서버에 부담이되는 http request수를 줄이고 대신 대기 시간을 갖도록 한다.
- 때문에 서버 부담이 줄어든다는 장점이 있다.
- 어떤 이벤트가 새로 발생해서 CLIENT에게 넘겨줄 EVENT가 생기면, 그때 RESPONSE를 주면서 Connection을 끊는다.
- 그룹채팅 역시 서버에 상당히 부하가 많이 갈 것으로 느껴지지만, 실제로 CLIENT가 타이핑을 하는 시간이 수 초 정도 있고 그룹채팅의 인원수 제한도 있으므로, 롱- 폴링 방식을 사용할 수 있는 수준이다.

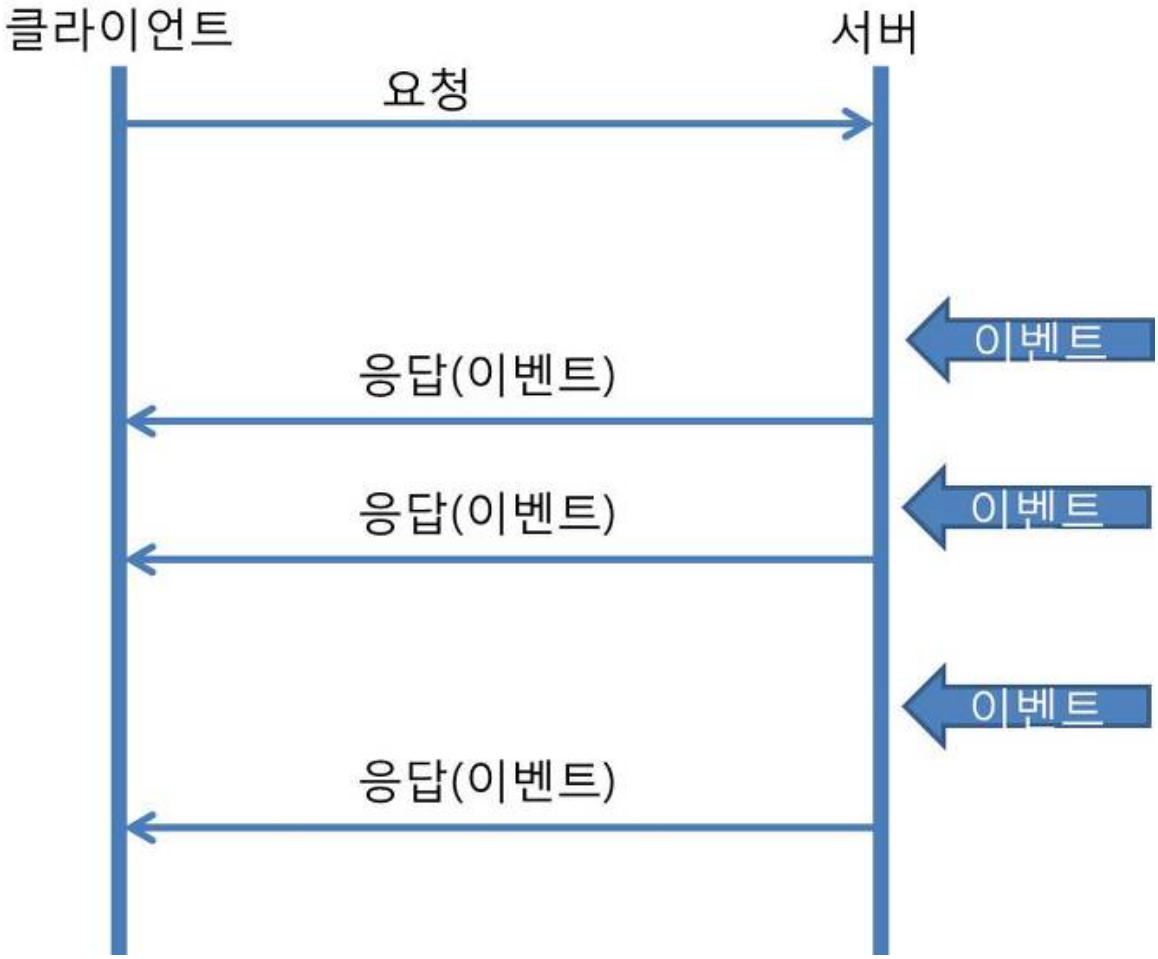
<단점>

- 하지만, 연결이 유지된다는 특성 때문에 동시 접속자를 많이 가지게 될 수 있습니다. 따라서 웹서버의 최대요청한계(Max REQUEST Limit)을 넘겨 연결이 끊어질 위험이 있습니다.
-



3번. Streaming - "유튜브 동영상 스트리밍 서비스"

- long polling과 마찬가지로 클라이언트에서 서버로 일단 http request를 날린다.
- 하나의 웹 요청에 대해 웹 접속을 계속 열어두고, 새로 이벤트가 발생하면 발생할 때마다 부분적인 응답으로 브라우저로 보내는 방식
- 서버에서 클라이언트로 이벤트를 전달할 때 해당 요청을 끊지 않고 필요한 메시지만 보내기를 (flush) 반복하는 방식이다.
- 서버는 지속적인 업데이트를 위해 무한정(또는 일정 시간 동안) 요청을 대기시키며, "chunked" 기반 메시지를 이용하여 응답 시 연결을 유지 시키는 방식이다.
- long polling에 비해 서버에서 메시지를 보내고도 다시 http request 연결을 하지 않아도 되어 부담이 경감될 것으로 보인다.



1~3번 까지는 진정한 의미에서의 양방향 통신이라기보다는, 기존 클라이언트 서버간 http request를 활용한 흉내내기라 볼 수 있다.

다음 나올 웹소켓 방식은 html5와 함께 나온 기술로 진정한 의미의 양방향 통신으로 명성을 떨치고 있다.

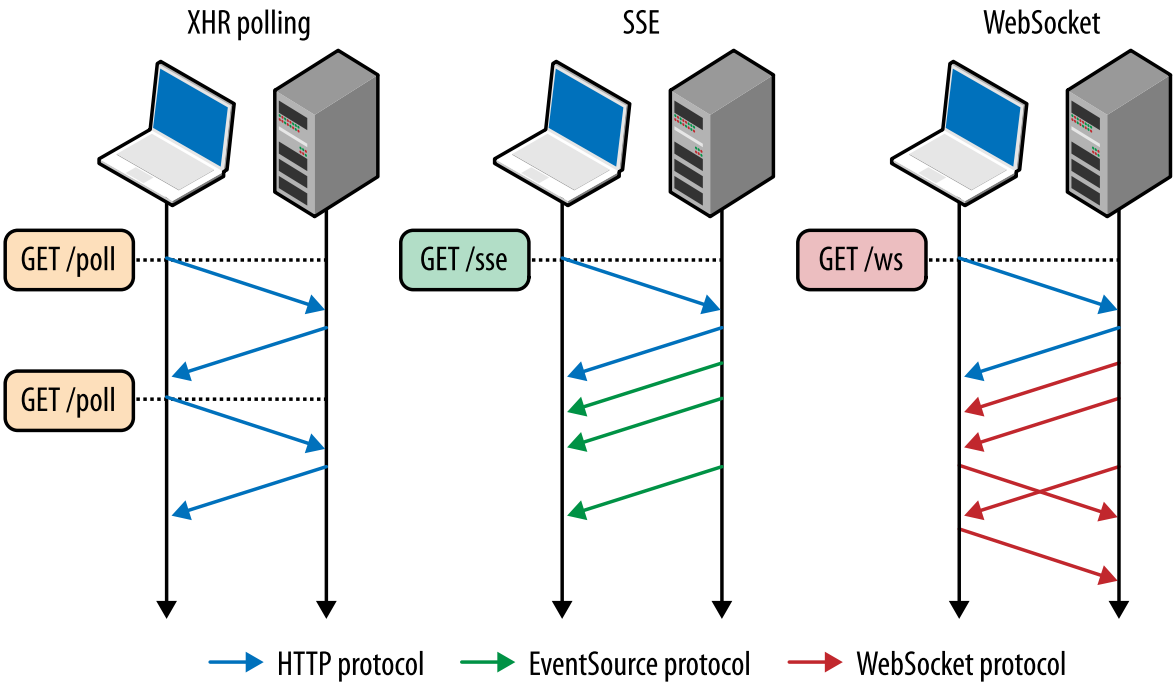
4. 웹소켓 방식 (Websocket) - "유튜브 실시간 라이브 방송"

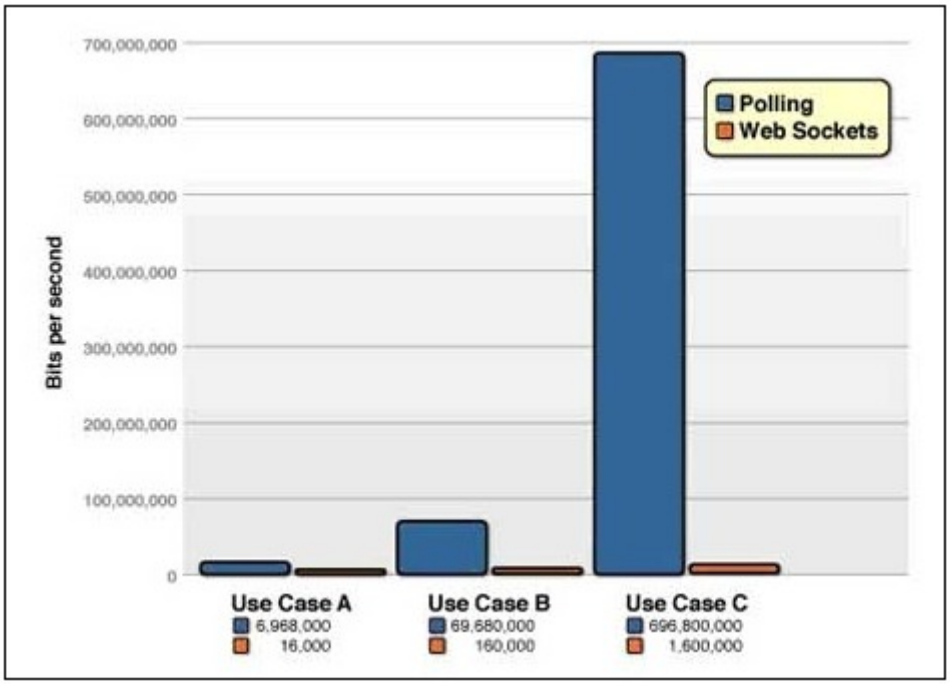
- 실시간 양방향 통신의 끝판왕이다.
- 앞의 1, 2, 3번 방식은 모두 어찌됐든 ClENT로 부터 http request 를 서버가 받은 다음 데이터를 전송해줄 수 있었다. 하지만, 웹소켓은 이 장벽을 넘어섰다.
- 다시말해, 클라이언트의 요청 없이도 서버에서 클라이언트로도 요청/데이터 전송등이 가능한 양방향 통신인 것이다.
- HTML5에서는 이 웹소켓 연결을 HTTP방식으로 지원한다. 때문에 기존 웹에 접근하듯이 80, 443 포트로 접속하므로 추가적인 방화벽을 열거나 하는 귀찮은 작업을 하지 않아도 양방향 통신이 가능하다. http의 규격인 CORS적용 이나 인증 과정을 기존과 동일하게 가져갈 수 있다는 장점이 있다.

  <단점>
- 하지만, 단점이 있는데 이 Websocket을 지원하지 않는 브라우저에서는 화면이 나타나지 않는다는 것이다!

  <단점 해결>

- 이 단점을 해결하는 방법은, 해당 브라우저(a.k.a X읍스플로러)에서는 앞의 1,2,3번 방법을 활용해서 화면을 rendering 하고 지원하는 브라우저에서는 웹소켓 방식을 사용하여 실시간 소통을 가능하도록 하는 것이다.





REFERECES

[WebSocket 실시간 개발하기]

[WebSocket과 Socket.io]

[[WS Protocol] HTML5 WebSocket(웹 소켓) 기본 예제 및 설명]

[HTTP에서부터 WEBSOCKET까지]

[AJAX란 무엇인가?]

[실시간 데이터 송신의 진화 : 폴링 방식, 롱폴링 방식, 웹소켓]

코멧(Comet) #2 – Ajax 폴링(Ajax polling) 채팅방 예제로 배우기

코멧(Comet) #3 – Ajax 롱폴링(Ajax Long polling) 채팅방 예제로 배우기

WebSocket

Longpolling PUSH SERVER