

# Fastcampus

Web Programming & Frontend Dev SCHOOL

REST API, Socket(TCP, websocket)

# API

# API

## Application Program Interface

- 응용프로그램에서 사용할 수 있도록 운영체제나 프로그래밍 언어가 제공하는 기능을 제어할 수 있게 만든 인터페이스
- [Windows API](#)
- [python/C API](#)

# Web API

웹서버 혹은 웹브라우저를 위한 API

## 웹 개발 패턴의 변화

- 1991 ~ 1999: Sir Timothy John "Tim" Berners-Lee가 하이퍼텍스트 기반의 프로젝트를 제안한 이후 정적인 컨텐츠들을 중심으로 한 웹 기술이 발달
- 1999 ~ 2009: Linux, Apache, Mysql, Php 중심의 동적인 서버, 정적인 클라이언트 모델이 지속됨
- 2010 ~ 현재: javaScript!! (Dynamic Web Client)

## 웹 개발 패턴의 변화

```
<html>
<head></head>
<body>
<h1>Static Header</h1>
<div>Static Contents</div>
</body>
</html>
```

- 1991 ~ 1999: Sir Timothy John "Tim" Berners-Lee가 하이퍼텍스트 기반의 프로젝트를 제안한 이후 정적인 컨텐츠들을 중심으로 한 웹 기술이 발달

## 웹 개발 패턴의 변화

```
<html>
<head></head>
<body>
<h1>{% Dynamic Header %}</h1>
<div>{% Dynamic Contents %}</div>
</body>
</html>
```

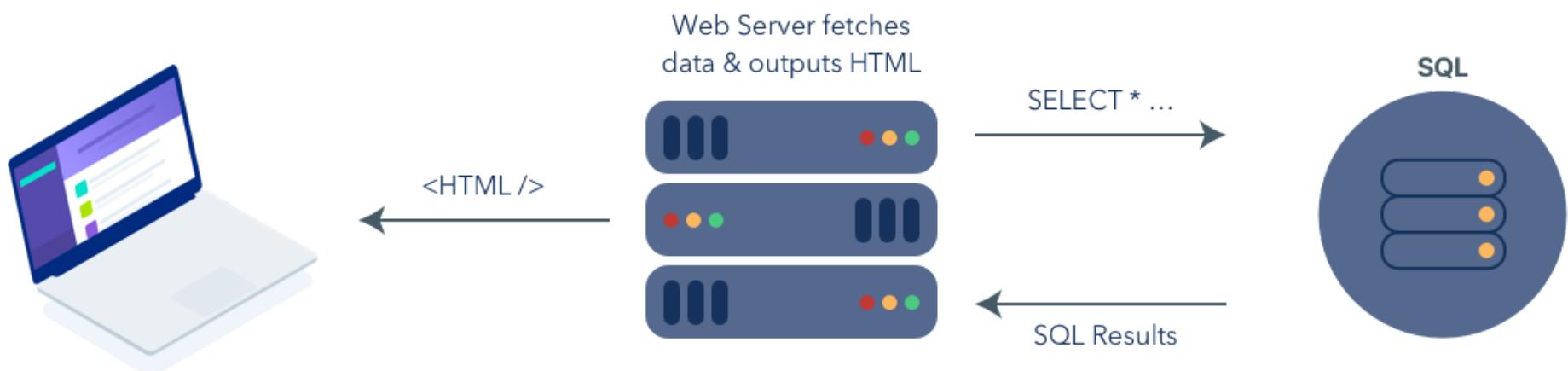
- 1999 ~ 2009: Linux, Apache, Mysql, Php 중심의 동적인 서버, 정적인 클라이언트 모델이 지속됨

# 웹 개발 패턴의 변화

```
<html>
<head>
<script src="https://unpkg.com/vue"></script>
</head>
<body>
<h1>{{ header }}</h1>
<div id="app">
  {{ message }}
</div>
<script>
var app = new Vue({
  el: '#app',
  data: {
    message: '안녕하세요 Vue!'
  }
})
</script>
</body>
</html>
```

- 2010 ~ 현재: JavaScript!! (Dynamic Web Client)

## Before REST



Before REST

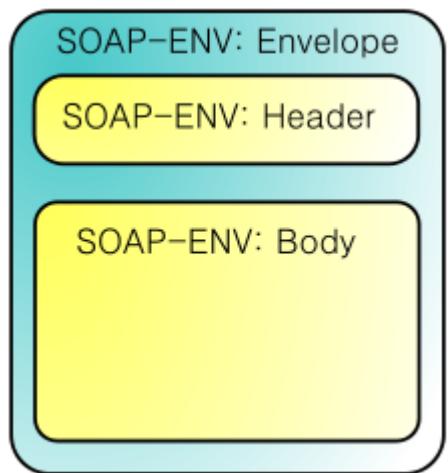


# SOAP

Simple Object Access Protocol

XML 파일 포맷을 활용해 데이터를 주고 받기 위한 시도

## xml architecture



## xml

```
<?xml version="1.0" encoding="UTF-8"?>
<breakfast_menu>
<food>
    <name>Belgian Waffles</name>
    <price>$5.95</price>
    <description>
        Two of our famous Belgian Waffles with plenty of real maple s
    </description>
    <calories>650</calories>
</food>
<food>
</food>
</breakfast_menu>
```

# REST API

RE presentational S tate T ransfer

A pplication P rogramming I nterface

Resource - URI

Verb - HTTP method

Representations - 표현

# So, REST is

| HTTP URI + HTTP method

[Yahoo Finance](#)  
[json api](#)

# Roy Fielding



- 2000년 UC Irvine의 박사 학위 논문 "Architectural Styles and the Design of Network-based Software Architectures" 발표

# Characteristics of REST

- 범용성(HTTP가 가능하면 OK)
- 리소스 중심 API 명세(URI를 읽는 것으로 이해 가능)
- Stateless(클라이언트의 상태를 신경쓰지 않음)

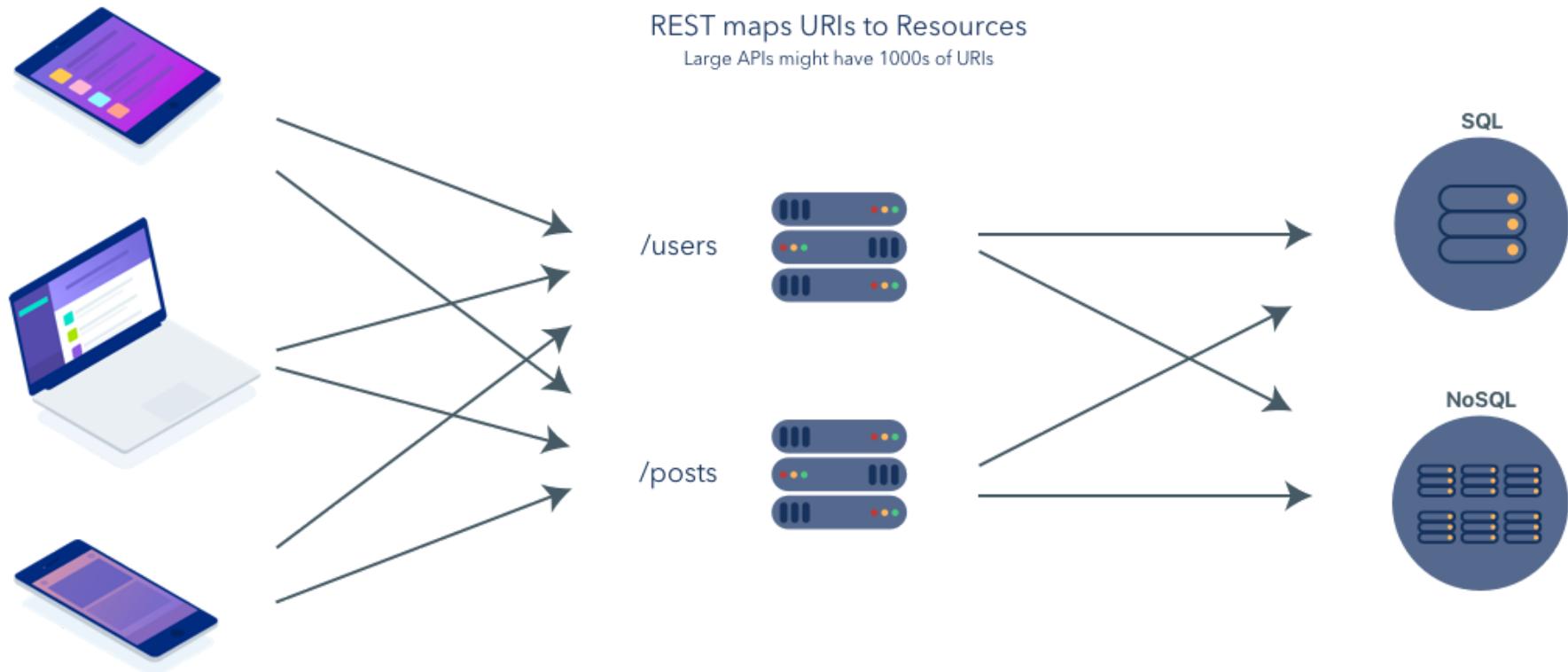
# pros and cons of REST

pros:

- 스펙없이 기존의 HTTP를 이용해 요청을 처리할 수 있다.

cons:

- 사용할 수 있는 메소드가 4개다
- 표준이 없다



# CRUD

Create

Read

Update

Delete

# HTTP Response code

200, 201 - Success

400, 404 - Not found

403 - Forbidden

500 - Server error

HTTP	REST	Status Code
GET	read	200 OK
POST	update	200 OK
PUT	create	201 CREATED
DELETE	delete	200 OK

404 Not Found

500 Internal Server Error

```
app.get('/users', (req, res)=>{
    res.json(db.users);
});

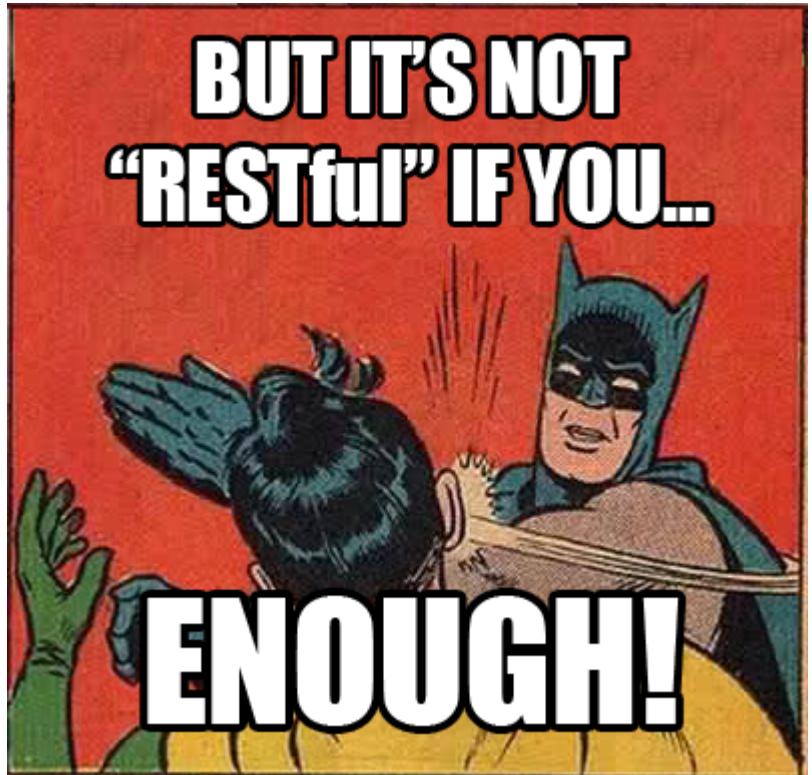
app.get('/users/:id', (req, res)=>{
    let user = db.users.find(user => user.id == req.params.i
    if (user) {
        res.json(user);
    } else {
        res.status(404).end();
    }
});

app.post('/users', ...);

app.delete('/users/:id', ...);
```

## REST API 설계시 주의할 점

- 버전관리 <https://api.foo.com/v1/bar>
- 명사형 사용 <https://foo.com/showid/> --> <https://foo.com/user/>
- 반응형 <https://foo.com/m/user/>, <https://m.foo.com/user/> (x)
- 언어코드 <https://foo.com/kr/>, <https://kr.foo.com/> (x)
- 응답상태 코드 (200, 400, 500)



# RESTful API Documentation

- 기존의 정의된 메소드를 사용하세요(200, 403, ..)
- 사용자가 정의하는 방식으로 작성하세요
- URI를 너무 강조하지 마세요
- 문서화도구(swagger 등)을 너무 믿지 마세요

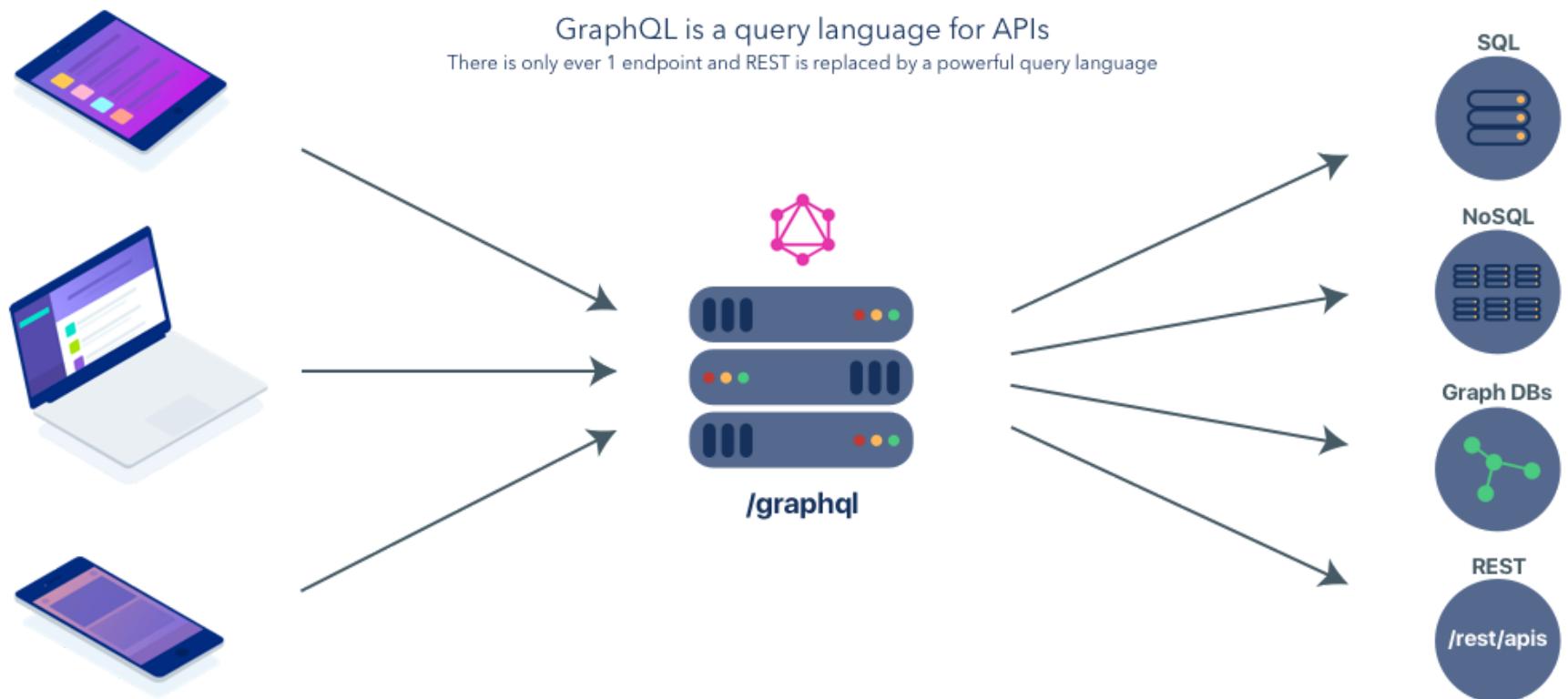
ex) <https://www.vmware.com/support/developer/vas/rest-api-1.1.0.RELEASE/index.html#4>

# After REST API

# GraphQL

- Open-sourced by Facebook
- Alternative to REST for building APIs
- create strong contract between Client and Server

# GraphQL



# Querying with GraphQL

```
query MoviesAndActors {  
  movies {  
    title  
    image  
    actors {  
      image  
      name  
    }  
  }  
}
```

# schema of GraphQL

```
schema {
  query: Query
}

type Query {
  movies: [Movie]
  actors: [Actor]
  movie(id: Int!): Movie
  actor(id: Int!): Actor
  searchMovies(term: String): [Movie]
  searchActors(term: String): [Actor]
}
```

```
type Movie {
    id: Int
    title: String
    image: String
    release_year: Int
    tags: [String]
    rating: Float
    actors: [Actor]
}

type Actor {
    id: Int
    name: String
    image: String
    dob: String
    num_credits: Int
    movies: [Movie]
}
```

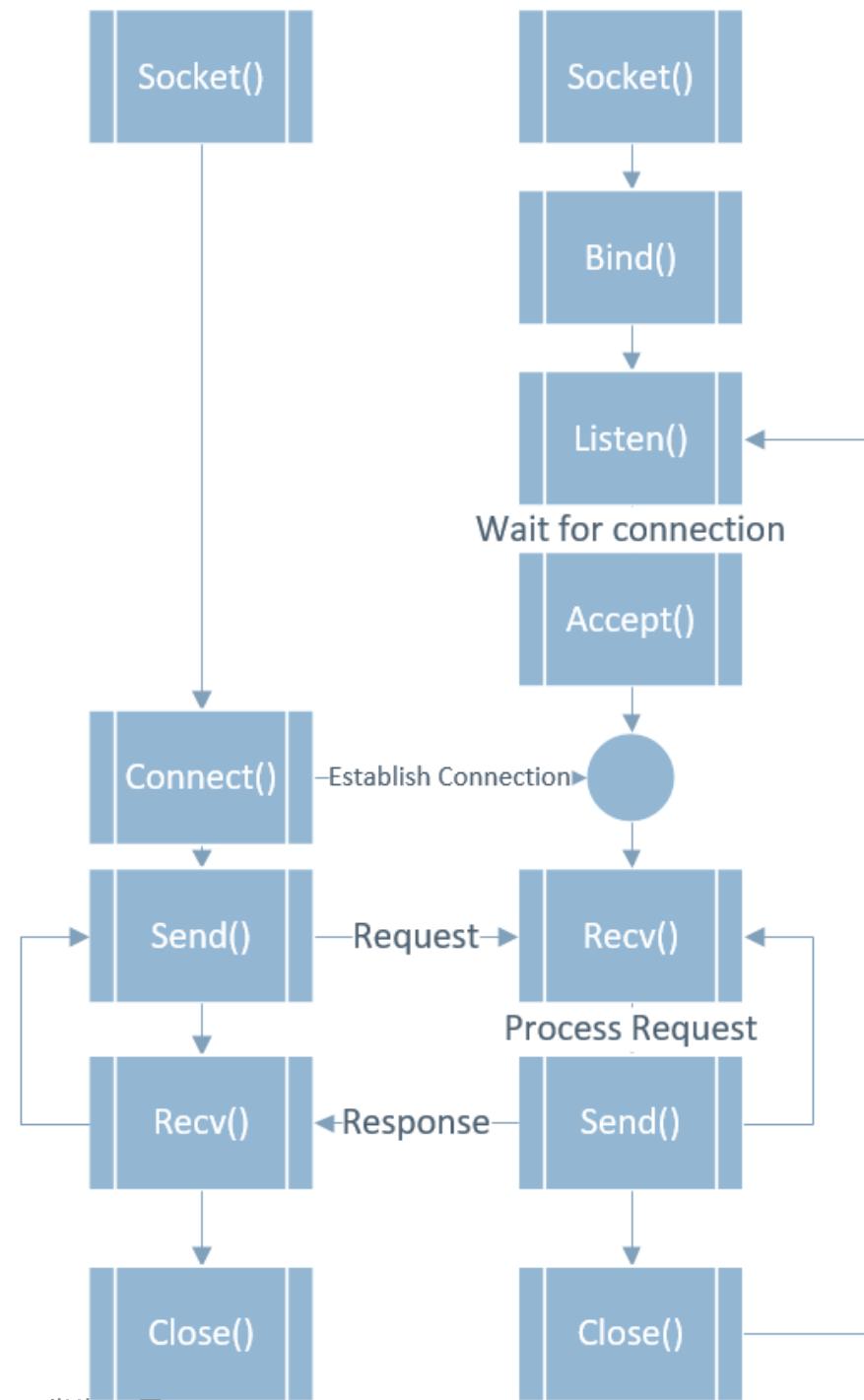
# Socket

# Socket

- Virtual End Point where entities can perform inter-process communication.

## So, Socket is ..

▶ 떨어져 있는 두 컴퓨터를 연결해주는 과정



## parameters

socket family(family) - AF\_INET, AF\_UNIX, AF\_BLUETOOTH

socket type(type) - SOCK\_STREAM, SOCK\_DGRAM

**socket 통신도 통신이므로.. 통신보안!!**

TLS(Transport Layer Security) - 프로토콜에 의한 암호화

SSL(Secure Socket Layer) - 포트에 의한 암호화

# Websocket

# Websocket

| 웹사이트가 사용자와 상호작용하기 위해 만들어진 기술

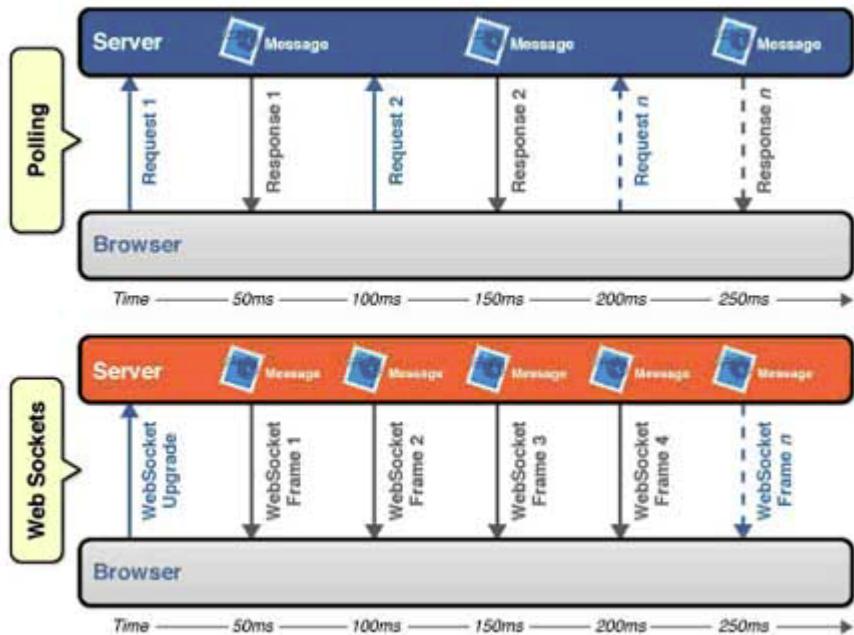
W3C가 API를 관리

port:80, HTTP1.1

# Before Websocket

- HTTP Request, Response
- Hidden Frame
- Long Polling

# Polling vs Websocket



## Differences between Socket, Websocket

Socket - HTTP run over TCP/IP

Websocket - run from web browser

# Socket.io

browser 와 상관없이 js를 이용해 실시간 통신을 지원

- 브라우저와 웹 서버의 종류와 버전을 분석해 가장 적절한 기술로 통신
- WebSocket, FlashSocket, AJAX Long Polling, AJAX Multi part Streaming, IFrame, JSONP Polling을 추상화한 기술
-