

Fastcampus

Frontend Dev SCHOOL

colaborate with git, websocket

WebSocket

Request & Response

Communication channels

- simplex
- half-duplex
- full-duplex

simplex

단방향통신

| 데이터를 전송하는 방향이 정해져있는 방식

half-duplex

반이중통신

| 전송의 방향은 양방향이나 전송 순간에는 한쪽에서만 전송가능한 방식

full-duplex

양방향통신

| 동시에 송수신이 가능한 방식

websocket

| 웹사이트가 사용자와 상호작용하기 위해 만들어진 기술

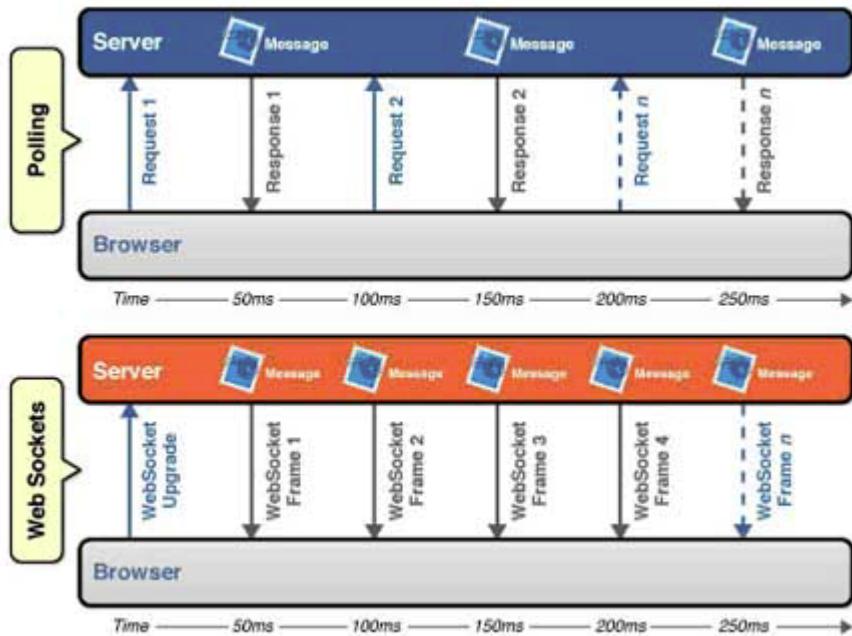
W3C가 API를 관리

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

webSocket

HTML5의 표준 full-duplex 통신 방식

<https://html.spec.whatwg.org/multipage/web-sockets.html#network>

socket.io

```
npm install --save socket.io
```

- 실시간 통신기술의 웹 브라우저 호환성 문제 해결을 위한 프로젝트
- IE6 부터 최신 브라우저까지 지원
- WebSocket, Flash Socket, AJAX Long Polling, AJAX Multipart Streaming, Forever iframe, JSONP Polling 기술 모두 포함
- 브라우저에 따라 최적화된 기술 사용
- 일관성있는 문법과 API로 개발 가능

<https://caniuse.com/#search=web%20sockets>

chat.ejs

```
<style>
  * { margin: 0; padding: 0; box-sizing: border-box; }
  body { font: 13px Helvetica, Arial; }
  form { background: #000; padding: 3px; position: fixed; }
  form input { border: 0; padding: 10px; width: 90%; margin: 0; }
  form button { width: 9%; background: rgb(130, 224, 255); }
  #messages { list-style-type: none; margin: 0; padding: 0; }
  #messages li { padding: 5px 10px; }
  #messages li:nth-child(odd) { background: #eee; }
</style>

<ul id="messages"></ul>
<form action="">
  <input id="m" autocomplete="off" /><button>Send</button>
</form>
```

connection

```
var http = require("http").Server(app);
var io = require("socket.io")(http);

io.on("connection", function(socket){
    console.log("a user connected");
});

http.listen(8080, function(){
    console.log("Chat Server is running at localhost:8080");
});
```

```
<script src="/socket.io/socket.io.js"></script>
<script>
    var socket = io();
</script>
```

disconnect

```
socket.on("disconnect", function(){
    console.log("user disconnected");
});
```

emit message

```
socket.on("chat message", function(msg){  
    console.log("message: " + msg);
```

```
<script src="http://code.jquery.com/jquery-1.11.1.js"></script>  
  
var socket = io();  
$("form").submit(function(){  
    socket.emit("chat message", $("#m").val());  
    $("#m").val("");  
    return false;  
});
```

broadcast

- unicast : 1:1. 출발지와 목적지가 정해진 전송
- broadcast : 네트워크에 접속된 모든 기기에 정보를 전송
- multicast : 네트워크에 접속된 기기 중 선택하여 전송

broadcast

```
io.emit("chat message", msg);
```

```
socket.on("chat message", function(msg){
    $("#messages").append($(".<li>").text(msg));
});
```

Software Engineering

Software Release Life Cycle

초기 개발단계부터 마지막 출시까지 주기를 나타냄

Testing and development period

Pre-alpha

- 테스트 이전까지 진행되는 요구사항 분석, 설계, 개발, 유닛 테스트를 포함
- 핵심 기능이 동작하기 시작한 상태

Alpha

- 소프트웨어 테스트를 시작하는 첫 단계
- 회사 내부 테스터를 통해 진행하며, 피드백을 통해 소프트웨어를 개선함

Beta

- 외부에 직, 간접으로 오픈하여 테스트를 진행
 - 오픈 베타: 일반 유저에 모두 오픈하여 기능을 제공함.
 - 클로즈드 베타: 신청자 중 일부에 접근권한을 부여하고, 테스트를 진행함.

RC(Release Candidate)

- 정식 제품이 될 가능성이 있는 베타버전. 심각한 문제가 없다면 이들 중 하나가 정식 버전이 됨.

Release period

RTM(Release to Manufacturing)

- 소프트웨어를 유저에게 제공될 준비가 완료된 상태

GA(General Availability)

- 소프트웨어를 유저가 이용 가능한 상태

Queue



Queue

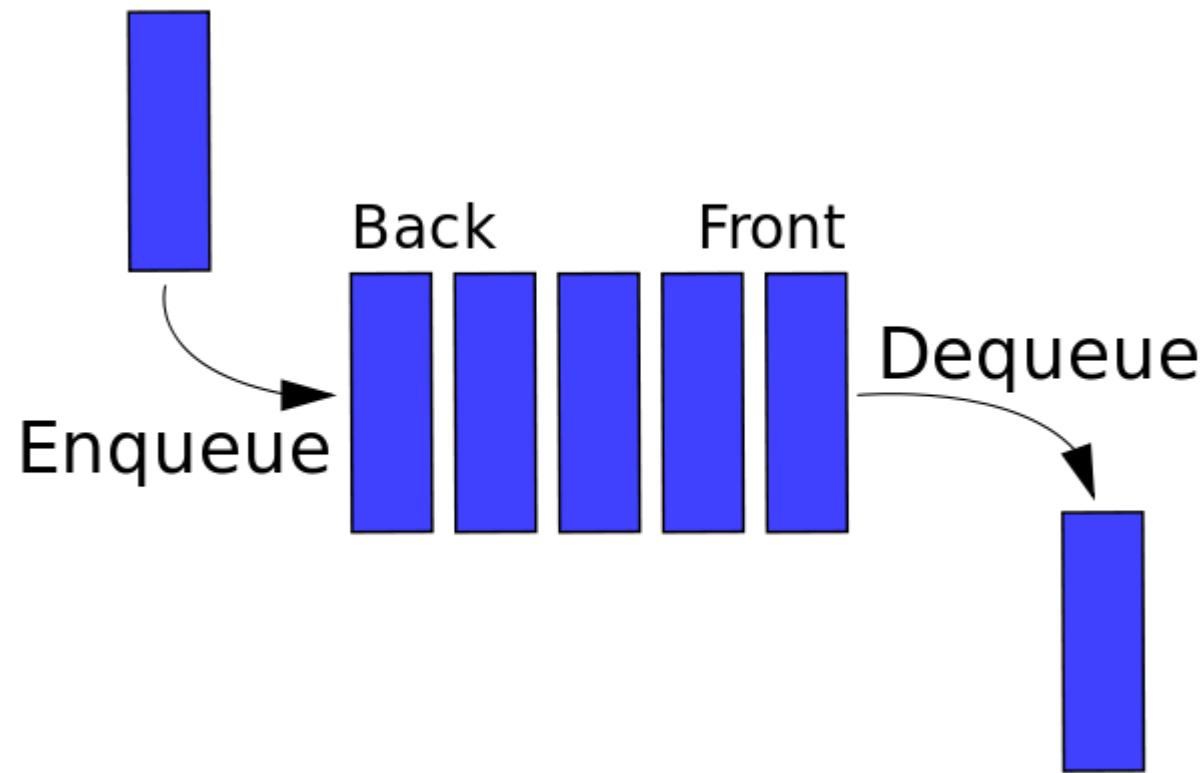
a queue is a particular kind of abstract data type or collection in which the entities in the collection are kept in order.

FIFO

First In First Out

Enqueue & Dequeue

- Enqueue: addition of entities to the rear terminal position
- Dequeue: removal of entities from the front terminal position



Let's Create Queue class

```
function Queue() {  
    //properties, methods  
    var items = [];  
}
```

Enqueue & Dequeue

```
function Queue() {  
    //properties, methods  
    this.enqueue = function(element) {  
        items.push(element);  
    };  
    this.dequeue = function() {  
        return items.shift();  
    };  
}
```

front & isEmpty

```
function Queue() {
    //underneath Enqueue & Dequeue
    ...
    this.front = function() {
        return items[0];
    };
    this.isEmpty = function() {
        return items.length == 0;
    };
}
```

clear & size & print

```
function Queue() {
    //underneath front & isEmpty
    ...
    this.clear = function() {
        items = [];
    };
    this.size = function() {
        return items.length;
    };
    this.print = function() {
        console.log(items.toString());
    };
}
```

Let's Enqueue with Queue class

```
> var queue = new Queue();
> console.log(queue.isEmpty());

> queue.enqueue("Fast");
> queue.enqueue("Campus");
> queue.enqueue("School");

> queue.print();
> console.log(queue.size());
> console.log(queue.isEmpty());
```

Let's Dequeue with Queue class

```
> queue.dequeue();
> queue.dequeue();
> queue.print();
```

Create Stack(), QueueWithStack()

- 3 members 1 repo
- pull 받아 사용해야 하며, 피처 브랜치에서 작업해야 합니다.
- Requirement
 - stack
 - push, pop
 - peek, isEmpty
 - size, print
 - queue with stack
 - enqueue, dequeue

Create Queue with 2 Stacks

```
function Queue_with_stack( ) {  
    var inBox = [];  
    var outBox = [];  
  
    this.enqueue = function( num ) {  
        inBox.push( num );  
    };  
  
    this.dequeue = function( ) {  
        if (outBox.length > 0) {  
            return outBox.pop();  
        }  
  
        while(inBox.length > 1) {  
            outBox.push( inBox.pop() );  
        }  
  
        return inBox.pop();  
    };  
}
```

Gulp



gulp.js

The streaming build system

Gulp is..



Task runner

- 매우 귀찮은 루틴한 작업들을 자동화 할 수 있는 툴
- 현재 2735 + a 개의 패키지가 존재
 - 따라서 필요한 기능을 골라 설치할 필요가 있음!!

task flow

코드작성 – JS test(jshint) – JS Minify – JS Merge(concat) – CSS
Minify – CSS Merge – 결과물

```
$ npm install gulp --global  
$ npm install gulp --save-dev
```

```
$ touch gulpfile.js

var gulp = require("gulp");

//hello라는 gulp task를 등록
gulp.task("hello", function () {
    return console.log("hello gulpworld");
});

$ gulp hello
```

gulp 기본 문법

- `gulp.task` : gulp의 작업단위
- `gulp.src` : gulp 실행의 대상
- `gulp.dest` : gulp 실행 후 목적지
- `gulp.watch` : 변화 감지 후 자동 실행

기본값 설정하기

```
$ gulpfile.js

var gulp = require("gulp");

//hello라는 gulp task를 등록
gulp.task("hello", function () {
    return console.log("hello gulpworld");
});

gulp.task("default", ["hello"]);

$ gulp
```

우선순위 설정하기

```
$touch gulpfile.js

var gulp = require("gulp");

//hello라는 gulp task를 등록
gulp.task("hello", function () {
    return console.log("hello");
});

gulp.task("gulpworld", ["hello"], function () {
    return console.log("gulpworld");
});

gulp.task("default", ["gulpworld"]);

$ gulp
```

자주쓰는 목적지 설정하기

```
var publicPath = {  
    src : './public/src/',  
    dest : './public/dist/'  
};
```

uglify(gulp-uglify) : js uglify

```
gulp.task("uglify", function(){
    pump([
        gulp.src(publicPath.src + 'js/uglify.js'),
        uglify(),
        gulp.dest(publicPath.dest + 'js/')
    ]);
});
```

gulp-concat : js concatenate

```
gulp.task("concat", function(){
  pump([
    gulp.src([publicPath.src + 'js/concat1.js', publicPath.src + 'js/concat2.js'],
      {buffer: false})
    .pipe(concat('concatenated.js'))
    .pipe(gulp.dest(publicPath.dest + 'js/'))
  ]);
});
```

gulp-imagemin : image minify

```
gulp.task("imagemin", function(){
  pump([
    gulp.src(publicPath.src + 'img/*.jpg'),
    imagemin(),
    gulp.dest(publicPath.dest + 'img/')
  ]);
});
```

css minify(gulp-clean-css) : css minify

```
gulp.task("cleancss", function(){
  pump([
    gulp.src(publicPath.src + 'css/minify.css'),
    cleancss(),
    gulp.dest(publicPath.dest + 'css/')
  ]);
});
```

gulp-sass : convert .scss to .css

```
gulp.task("sass", function(){
  pump([
    gulp.src(publicPath.src + 'sass/*.scss'),
    sass().on('error', sass.logError),
    gulp.dest(publicPath.dest + 'css/')
  ]);
});
```

gulp-concat-css : concatenate css files

```
gulp.task("concatcss", function(){
  pump([
    gulp.src([publicPath.src + 'css/concat1.css', pu
      concat('concatenated.css'),
      gulp.dest(publicPath.dest + 'css/')
  ]);
});
```

clean(del)

```
gulp.task("clean", function(){
    return del.sync([publicPath.dest + 'js/*.js', publicPath
});
```

watch

```
gulp.task("watch", function(){
    gulp.watch("public/src/*.js", ["uglify"]);
});

gulp.task("default", ["uglify", "watch"]);
```