

Fastcampus

Web Programming & Frontend Dev SCHOOL

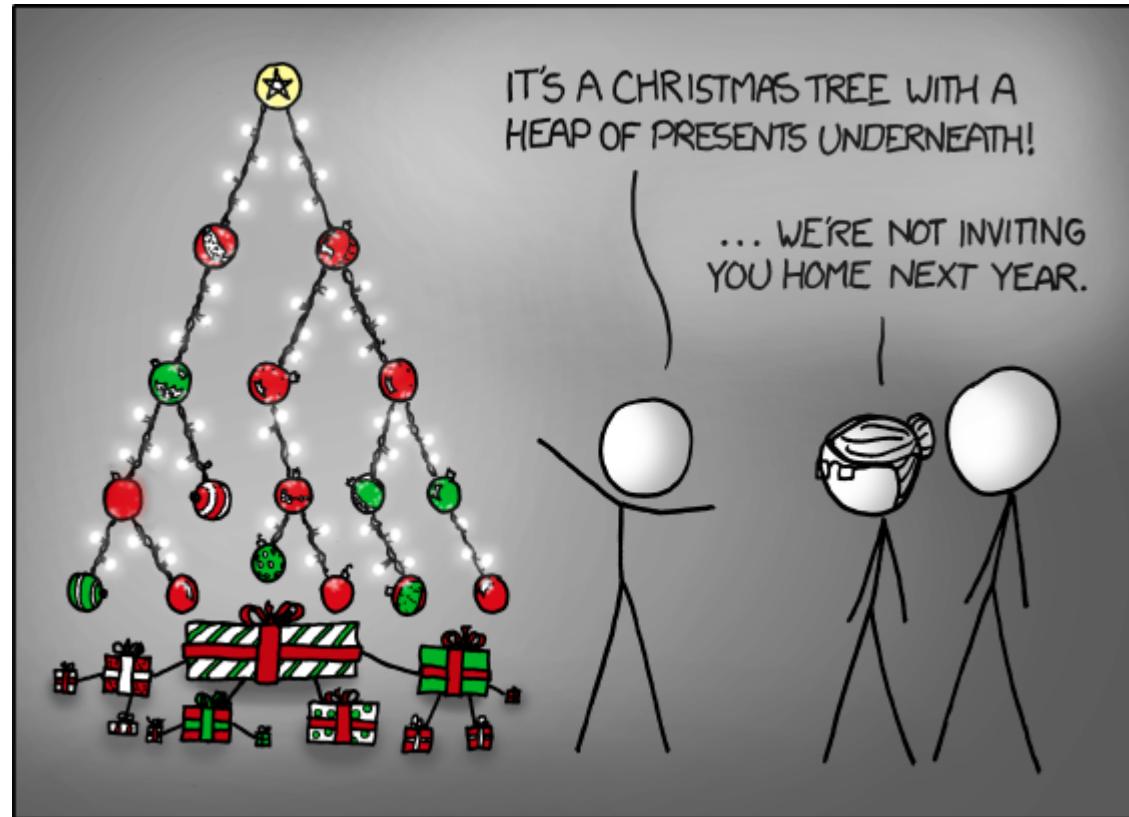
Intro to Computer Science

Data Structure

Data Structure

Data structure is a particular way of organizing data in a computer so that it can be used efficiently.

So, Data Structure is..



Data Structures in Web Development

Array & Hash(Dictionary) - indexing post

```
in RDB  
[articleId, title, body, userId, view]
```

```
[  
  {  
    userId: 1,  
    articleId: 1,  
    view: 100,  
    title: "sunt aut facere repellat provident occaecati exco  
    body: "quia et suscipit suscipit recusandae consequuntur  
  },  
  ...  
]
```

Data Structures in Web Development

Tree - DOM rendering performance, reply

```
<html>
<head></head>
<body>
<h1></h1>
<p></p>
</body>
</html>
```

Data Structures in Web Development

- Binary Tree Search
 - Queue(BFS, Breadth First Search)
 - Stack(DFS, Depth First Search)

We'll Learn about..

- Stack
- Queue
- Linked-list
- Tree

Stack



Stack

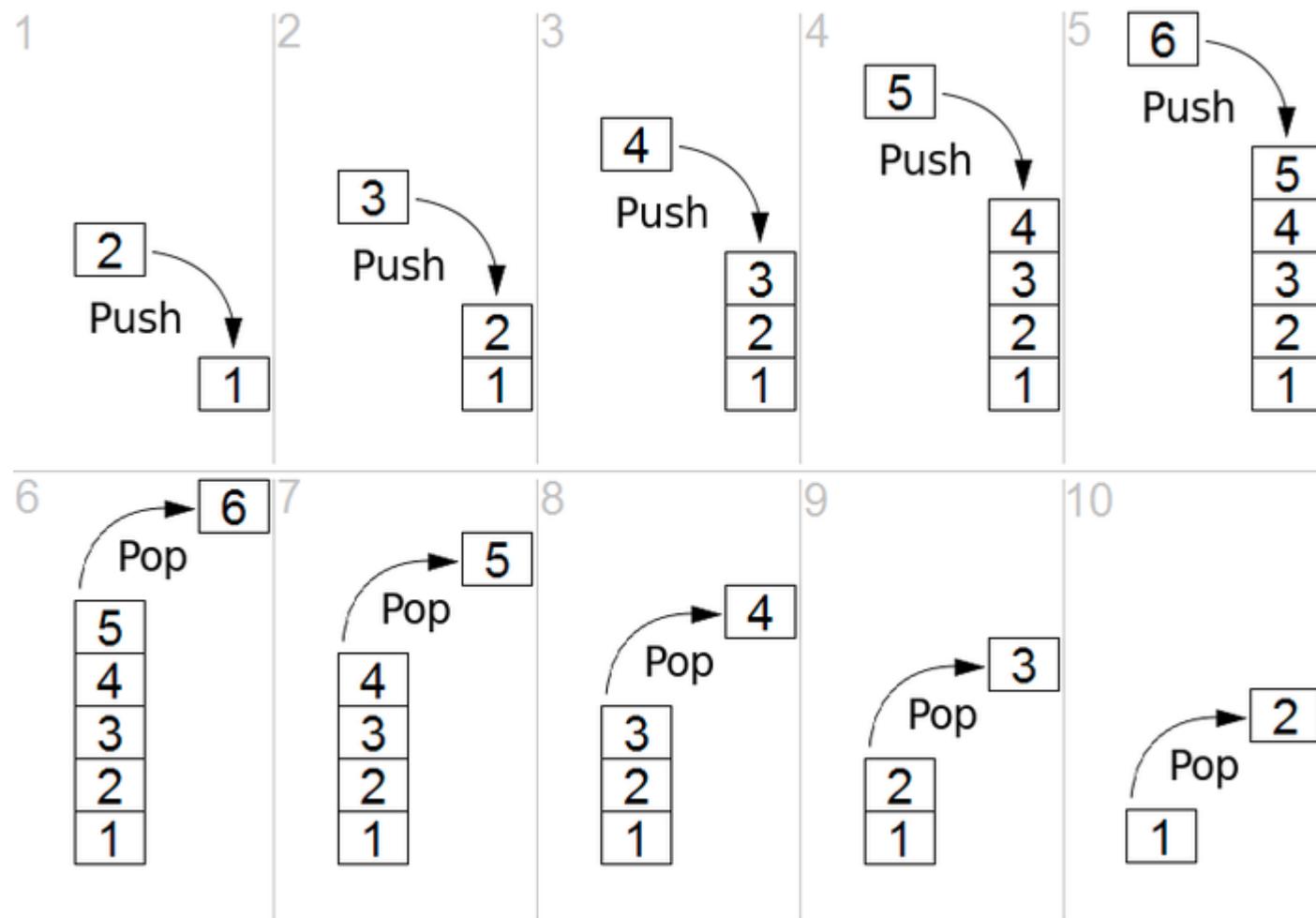
A stack is an abstract data type that serves as a collection of elements, with two principal operations.

- push: which adds an element to the collection
- pop: which removes the most recently added element that was not yet removed

LIFO

| Last In, First Out





Let's Create Stack class

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

push & pop

```
function Stack() {  
    //properties, methods  
    var items = [];  
  
    this.push = function(element){  
        return items.push(element);  
    };  
  
    this.pop = function(){  
        return items.pop();  
    };  
}
```

peek & isEmpty

```
function Stack() {
    //underneath push & pop
    ...
    this.peek = function(){
        return items[items.length-1];
    };
    this.isEmpty = function(){
        return items.length == 0;
    };
}
```

size & clear & print

```
function Stack() {
    //underneath peek & isEmpty
    ...
    this.size = function(){
        return items.length;
    };

    this.clear = function(){
        items = [];
    };

    this.print = function(){
        console.log(items.toString());
    };
}
```

Let's push with Stack class

```
> var stack = new Stack();
> console.log(stack.isEmpty());

> stack.push(5);
> stack.push(2);
> stack.push(8);

> console.log(stack.peek());

> stack.push(11);

> console.log(stack.size());
> console.log(stack.isEmpty());
```

Let's pop with Stack class

```
> stack.pop();
> stack.pop();

> console.log(stack.size());
> stack.print();
```

Let's make binaryConverter with Stack

case: $10_{(10)}$ to $1010_{(2)}$

```
10 / 2 == 5 rem == 0
5 / 2 == 2 rem == 1
2 / 2 == 1 rem == 0
1 / 2 == 0 rem == 1
```

push remainders ==> [0, 1, 0, 1]

output = pop remainders

Let's make binaryConverter with Stack

```
function divideBy2 (decimal) {  
    var remStack = new Stack(),  
        rem,  
        binaryString = '';  
}
```

Let's make binaryConverter with Stack

```
function divideBy2 (decimal) {
    var remStack = new Stack(),
        rem,
        binaryString = '';
    while (decimal > 0) {
        rem = Math.floor(decimal % 2);
        remStack.push(rem);
        decimal = Math.floor(decimal / 2);
    }
}
```

[Math.floor](#)

Let's make binaryConverter with Stack

```
function divideBy2 (decimal) {
    var remStack = new Stack(),
        rem,
        binaryString = '';

    while (decimal > 0) {
        rem = Math.floor(decimal % 2);
        remStack.push(rem);
        decimal = Math.floor(decimal / 2);
    }

    while (!remStack.isEmpty()) {
        binaryString += remStack.pop().toString();
    }
}
```

Queue

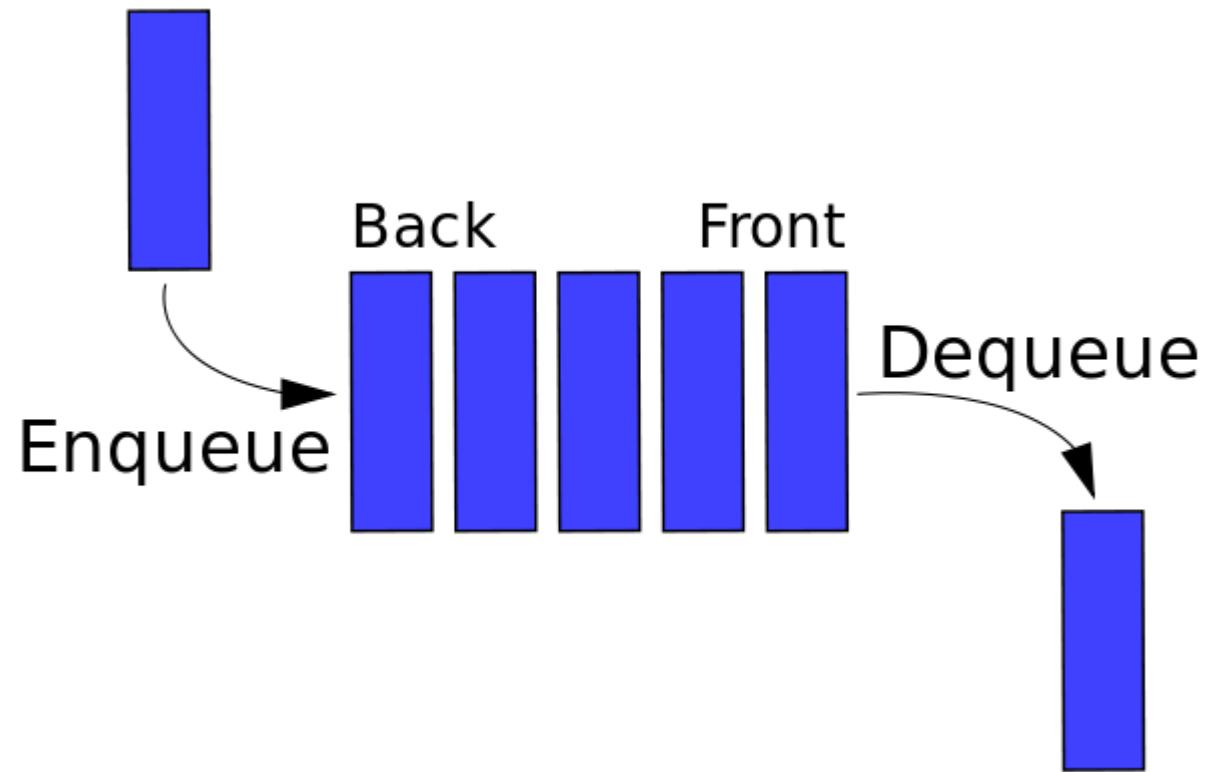


Queue

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

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 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();
}
}
```

Queue can make

- Priority Queue
- Circular Queue

Priority Queue

```
function PriorityQueue() {  
    var items = [];  
  
    ...  
}
```

```
function QueueElement (element, priority){  
    this.element = element;  
    this.priority = priority;  
}  
  
this.enqueue = function(element, priority){  
    var queueElement = new QueueElement(element, priority);  
  
    if (this.isEmpty()) {  
        items.push(queueElement);  
    } else {  
        var added = false;  
        for (var i=0; i<items.length; i++) {  
            if (queueElement.priority < items[i].priority) {  
                items.splice(i,0,queueElement);  
                added = true;  
                break;  
            }  
        }  
        if (!added) {  
            items.push(queueElement);  
        }  
    }  
};
```

splice

```
this.dequeue = function(){
    return items.shift();
};

this.front = function(){
    return items[0];
};

this.size = function(){
    return items.length;
};
```

Gulp



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/')
  ]);
});
```