Funtional Programing

Chapter 3. **자료 구조는 적게, 일은 더 많이**

map

Example

```
Observers.map((observer: Observer) =>
observer.triggerAction(actionType, this.state));
```

Example

```
// markers
Array.from(this.$base.find("._item"))
  .map((item, index) => ({
    index,
    lat: Number($(item).data("lat")),
    lng: Number($(item).data("lng")),
    title: $(item).data("title"),
    subTitle: $(item).data("subtitle"),
    icon: $(item).data("icon"),
  }))
  .map(({ index, lat, lng, title, subTitle, icon }: Marke
    new Marker({
      map: this.map.getMap(),
      markerOption: {index, lat, lng, title, subTitle, ic
      template: this.$markerTemplate.html(),
      state: this.state,
    })
));
```

reduce

mergeObjects

```
export const mergeObjects = <T>(base: T, ...targets: any[
  return targets.reduce((prev, next) => ({
    ...prev,
    ...next,
  }), base);
};
```

flatten

```
export const flatMerge =
(params: object[]): object => {
  return params.reduce((reduced: object, param: object) =>
    ...param,
    ...reduced,
  }), {});
};
```

pipe

```
export function pipe(...fns) {
  return param => fns.reduce(
    (result, fn) => fn(result),
    param
  )
}
```

getPairWithInitialValue

객체를 순회하며, 객체의 key 에 initialValue 를 덮어씌운다

```
/**
 * @param base key: value 쌍으로 이루어진 객체
 * @param initialValue 각 key 에 할당할 값
 */
export const getPairWithInitialValue =
  (base: object, initialValue: any): object => {
    return Object.keys(base).reduce((acc, key) => {
      acc[key] = initialValue;

      return acc;
    }, {});
};
```

filter

includes

```
export const includes = (array: any[], target: any): bool
  return array.filter((item) => item === target).length >
};
```

Recursive

DeepFlatten

```
function flatten(arr) {
  return function f(arr, newArr) {
    arr.forEach(function(v)) {
      Array.isArray(v) ? f(v, newArr) : newArr.push(v);
    });
    return newArr;
  }(arr, []);
}
```

Chapter 3. **자료 구조는 적게, 일은 더 많이**

끝.