generator

ECMAScript 2015

S67 일정 : 2016/9/22 - 16/10/27
시간 : 매주 목요일 저녁 8~10시
장소 : 네이버 D2 STARTUP FACTORY

projectBS x Bsidesoft x RoasteryKay x NAVER D2

Abstract Loop

기존의 루프는 추상화가 불가함. 루프의 추상화?

기존에는 루프 전체의 실행여부를 결정만 추상화할 수 있음.

```
const elLoop = (el, f)=>{
    const stack = [];
    do{
        f(el);
        if(el.firstElementChild) stack.push(el.firstElementChild);
        if(el.nextElementSibling) stack.push(el.nextElementSibling);
    }while(el = stack.pop());
};

const elLoopWithFilter = (el, run, filter)=>{
    const stack = [];
    do{
        if(filter(el)) run(el);
        if(el.firstElementChild) stack.push(el.firstElementChild);
        if(el.nextElementSibling) stack.push(el.nextElementSibling);
    }while(el = stack.pop());
};

const elLoopWithFilter = (el, run, filter)=> elLoop(el, el=>filter(el) && run(el));

[1,2,3].forEach(v=>console.log(v));
```

Abstract Loop

Generator는 루프구조의 추상화를 가능하게 함

```
const elLoop = function*(el) {
    const stack = [];
    do{
        yield(el);
        if(el.firstElementChild) stack.push(el.firstElementChild);
        if(el.nextElementSibling) stack.push(el.nextElementSibling);
    }while(el = stack.pop());
};

for(const el of elLoop(document.getElementById('a'))){
    if(el.tagName == 'article' && el.innerHTML.startWiths('projectA')) return el;
}
```

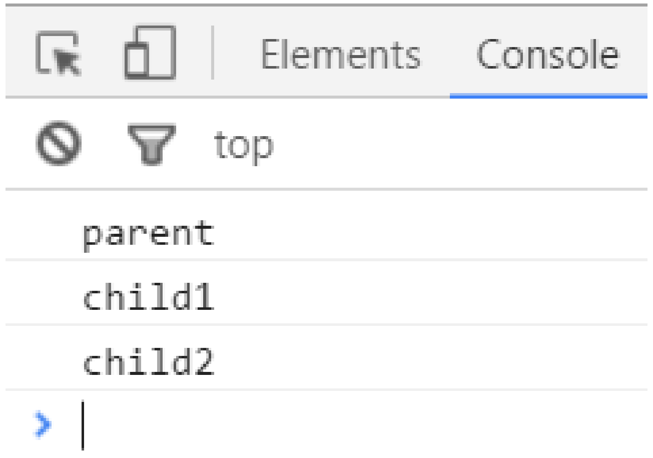
Abstract Loop

composite, visitor, iterator, decorator, cor 등의 복합적인 루프를 모두 추상화하여 for of로 노출

```
const is = (v, cls)=>{
    if(!(v instanceof cls) throw 'invalid type';
};
const Composite = class{
    constructor(title){
        this.title = title;
        this.children = new Set();
    }
    add(child, type = is(child, Composite)){
        this.children.add(child);
    }
    *operation(){
        yield this.title;
        for(const c of this.children) yield* c.operation();
    }
    [Symbol.iterator](){
        return this.operation();
    }
}
```

```
let P = new Composite('parent');
P.add(new Composite('child1'));
P.add(new Composite('child2'));

for(const title of P) console.log(title);
```

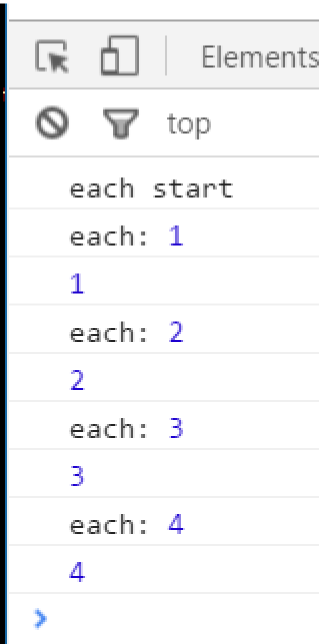


```
parent
child1
child2
>
```

Lazy Loop(Loop to Value)

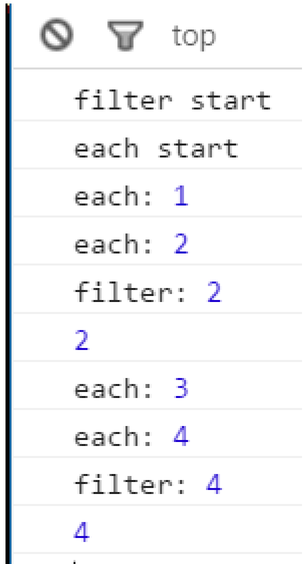
루프를 지연하여 필요한 만큼만 루프를 돌면서 문제를 해결하고 루프가 시작되기 전에는 부하를 걸지 않음

```
const each = function*(arr) {
    console.log('each start');
    for(const v of arr.slice(0)) {
        console.log('each:', v);
        yield v;
    }
};
for(const v of
    each([1,2,3,4])
)   console.log(v);
```



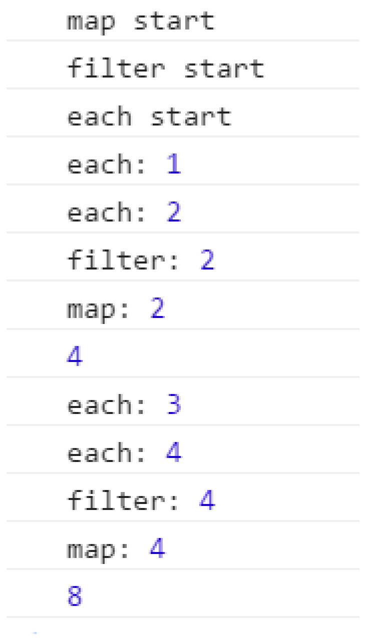
```
top
each start
each: 1
1
each: 2
2
each: 3
3
each: 4
4
>
```

```
const filter = function*(e, f) {
    console.log('filter start');
    for(const v of e) {
        if(f(v)) {
            console.log('filter:', i);
            yield v;
        }
    }
}
for(const v of
    filter(each([1,2,3,4]), v=>(v%2 == 0))
) console.log(v);
```



```
top
filter start
each start
each: 1
each: 2
filter: 2
2
each: 3
each: 4
filter: 4
4
```

```
const map = function*(e, f) {
    console.log('map start');
    for(const v of e) {
        console.log('map:', v);
        yield f(v);
    }
}
for(const v of
    map(
        filter(each([1,2,3,4]), v=>(v%2 == 0)),
        v=>v*2)
) console.log(v);
```



```
map start
filter start
each start
each: 1
each: 2
filter: 2
map: 2
4
each: 3
each: 4
filter: 4
map: 4
8
```

lazy chaining

```
const lazy = (_=>{
    const gene = function*(iter) {for(const v of iter) yield v; };
    const filter = function*(g, f) {for(const v of g) if(f(v)) yield v;};
    const map = function*(g, f) {for(const v of g) yield f(v);};
    const Lazy = class{
        constructor(iter) {this.seed = gene(iter); }
        [Symbol.iterator] () {return this.seed; }
        filter(f) {
            this.seed = filter(this.seed, f);
            return this;
        }
        map(f) {
            this.seed = map(this.seed, f);
            return this;
        }
    };
    return v=>new Lazy(v);
})();
```

```
for(const v of lazy([1,2,3,4])) console.log(v);

for(const v of
    lazy([1,2,3,4])
        .filter(v=>v % 2 == 0)
) console.log(v);

for(const v of
    lazy([1,2,3,4])
        .filter(v=>v % 2 == 0)
        .map(v=>v*2)
) console.log(v);
```