TIL (6.30) js 책 학습.

① function ⇒ 익명함수

js에서는 함수는 저장하고 변수처럼 사용가능

→ first citizen.

함수는 인자로 막 전달 할 수 있고, 간결성↑

array.filter(function(v, i){ return }).—.—

익명함수!

⊕ 이런식으로 call back 사용가능

※ 단 한번만 쓸

익명함수는 재귀를 쓰면 이상(?) 해진다

```
function recursive(){ recursive(); }
```

if ~~함수명~~ 변수명이 바뀌면?

```
① var a = function(){ return a(); };
② var β = a;
③ a = {};
④ β();
```

Warning!? ← ④

① 에서 이름이 없으므로 a()를 호출

② a를 β에 담음

③ a가 사라짐

④ ERROR!?

해결책 ⊗ this

↙

this.recursive()!

↙

얘는 함수명, 즉 담기는 애가 recursive가 아니면 문제가 생김

↳ 해결책

✓ 이름있는 익명함수

↳
```
var x = function name(){
    return name();
}
```

↙ 이름. scope가 딱 여기

→ 다른곳에 name이 존재해도 상관 x?

⇒ JS에 재귀를 쓰는 경우가 있는지 모르지만, 안 사용하는 편

+ callee ← 얘는 실행중인 함수를 가르킴

arguments.callee(—);

↳ 재귀랑 같은 의미

(!) 쓰는 것 추천하진 x

↳ 여기까지가 재귀

Memoizing ? → 뭔진 알아. 근데 실제로 쓰나?

우리팀은 안쓰는데.. ?

값이 바뀌든 없어? ⇒ 이부분은 이야기를 해보자

? array에 method를 추가하는게 의미 있냐? (p 96)

→ Array.prototype.push - 이렇게 쓰는데, 이거 쓰려면 내부구현을 알아야 할듯 → length가 올라가는데...

⊕ arguments를 통해 overloading은 흉내 낼수 있음

→ 개인적으로 이건 overloading이라 생각하진 X

↳ 이건 보기도 혼란함

+ 비교방법

tip function Name.length → 함수에 정의된 인자 수

arguments length 호출된 수

→ 기본적으로 js는 overload를 지원하지 X

```
function x(a,b) —
function a(a) —
```

← 얘가 호출됨

책에서는 (오류)는 뭔

→ addMethod(obj, name, function( ))

↳ 곧 보아야 한것 (이해 X)

fn.argument 가 쓰여서..

혹시 → this.argument 인가?

→ 브라우저마다 | typeof fn == 'function' |

↳ 다르게 동작 할수도..

Object.prototype.toString(◯) == '[object Function]'

↓

이렇게도 가능.. (흔히?)

⇒ 재밌어. (4장 끝)