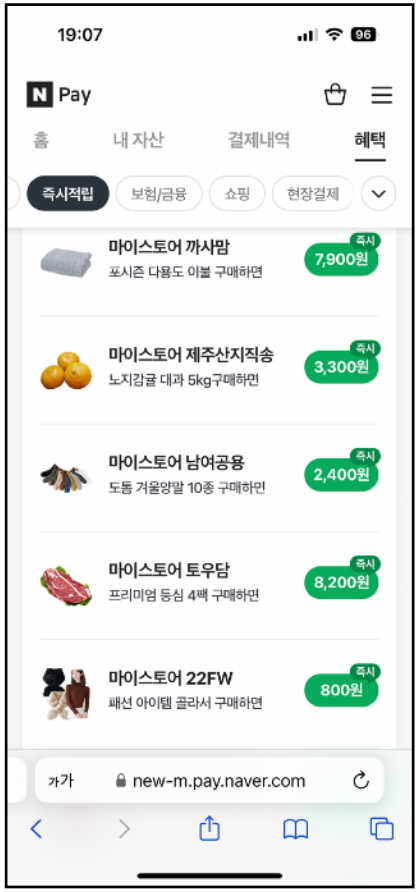
JSFuck

CONTENTS

00

쥽쥽봇 근황

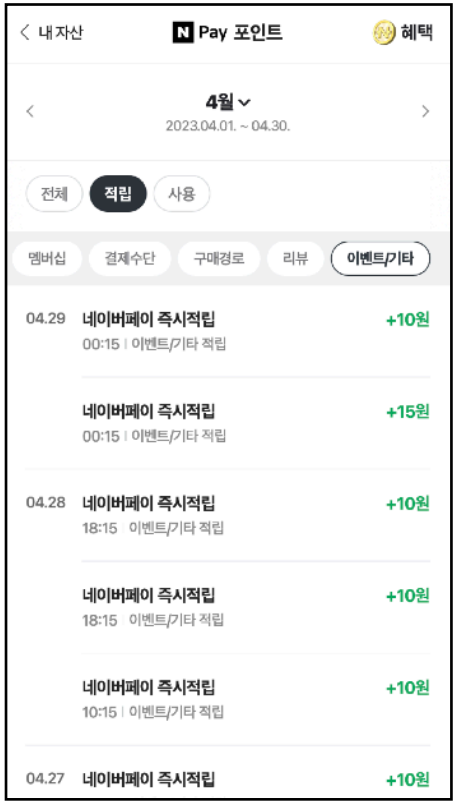
네이버페이 줍줍봇

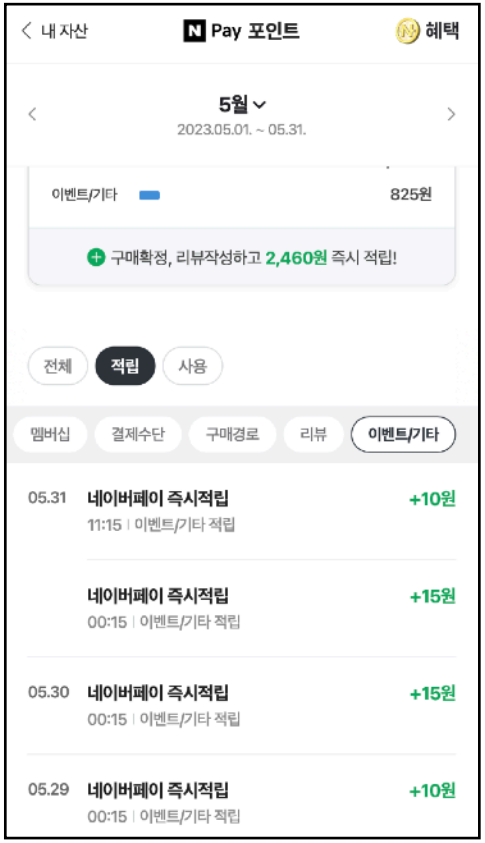


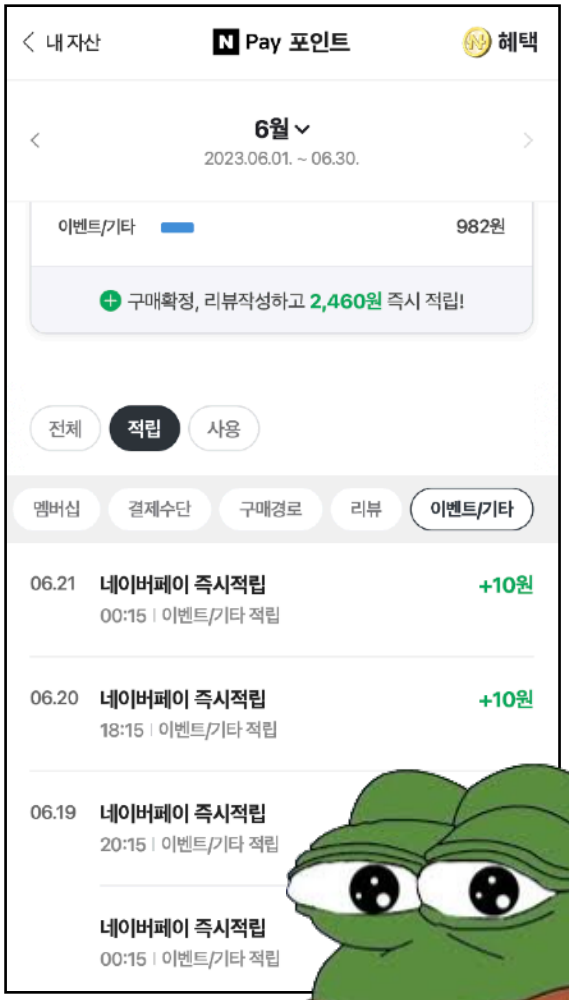

```python
def naver_click(conn, url):
    print("# naver_click : " + url)

    cursor = conn.cursor()
    sql = f"select count(1) from npay_clicker where url = '{url}'"
    cursor.execute(sql)
    rs = cursor.fetchone()
    if rs[0] == 0:
        response = requests.get(url=url, headers=HEADER, cookies=NAVER_COOKIES)
```

- # 아직도 잘 돌아가고 있습니다.







인증 토큰이 아직까지(왜??) 먹히고 있음 (8개월 넘음)

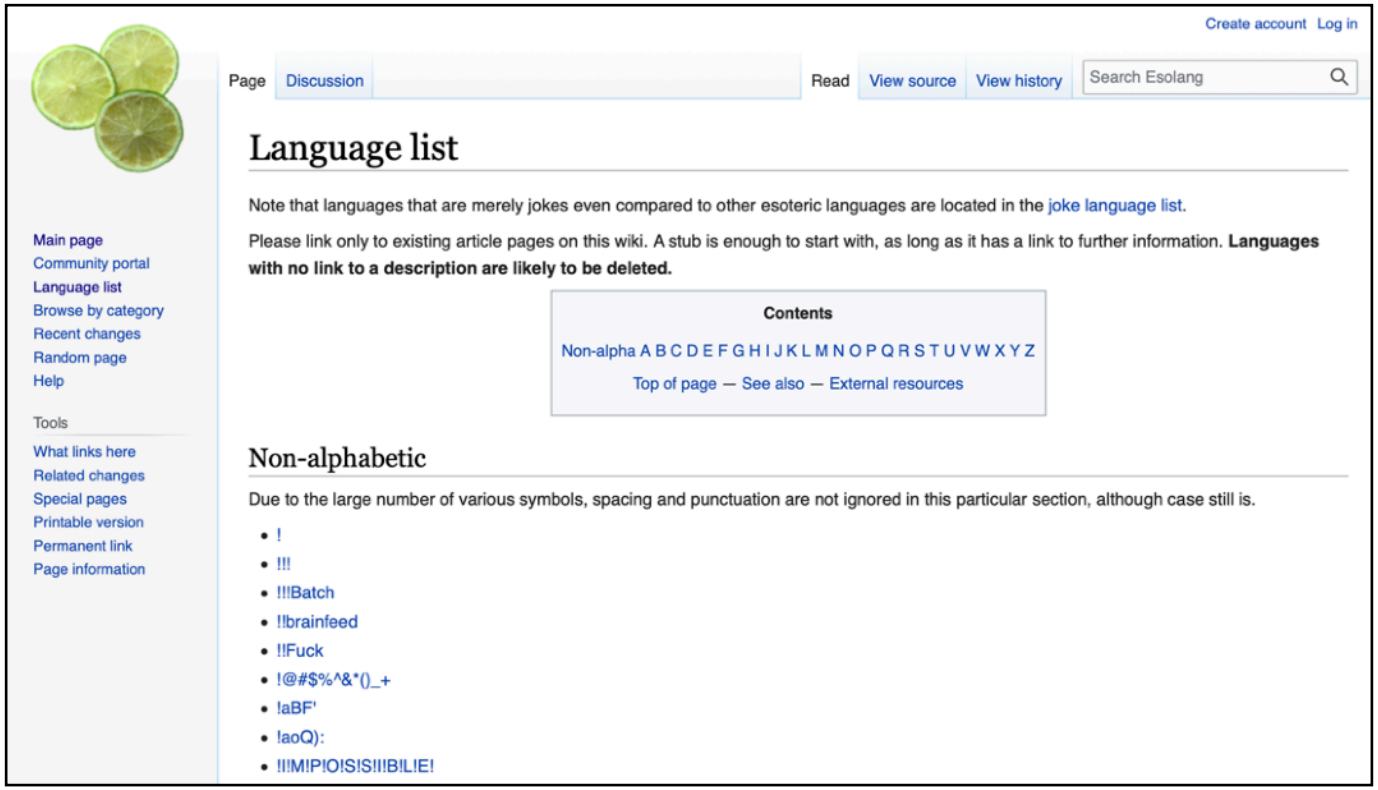
01

난해한 프로그래밍 언어

개요

- Esoteric(난해한) programming language / Esolang
- 일부러 사용하기 어렵게 만든 언어
- 언어의 한계를 시험하기 위해서
- 장난으로
- 기존 언어와 완전히 다른 언어를 만들고 싶어서

- 전용 Wiki도 있음



https://esolangs.org

BrainFuck

- 8개의 문자로만 이루어진 언어
- + -] [> < , .
- 명령어

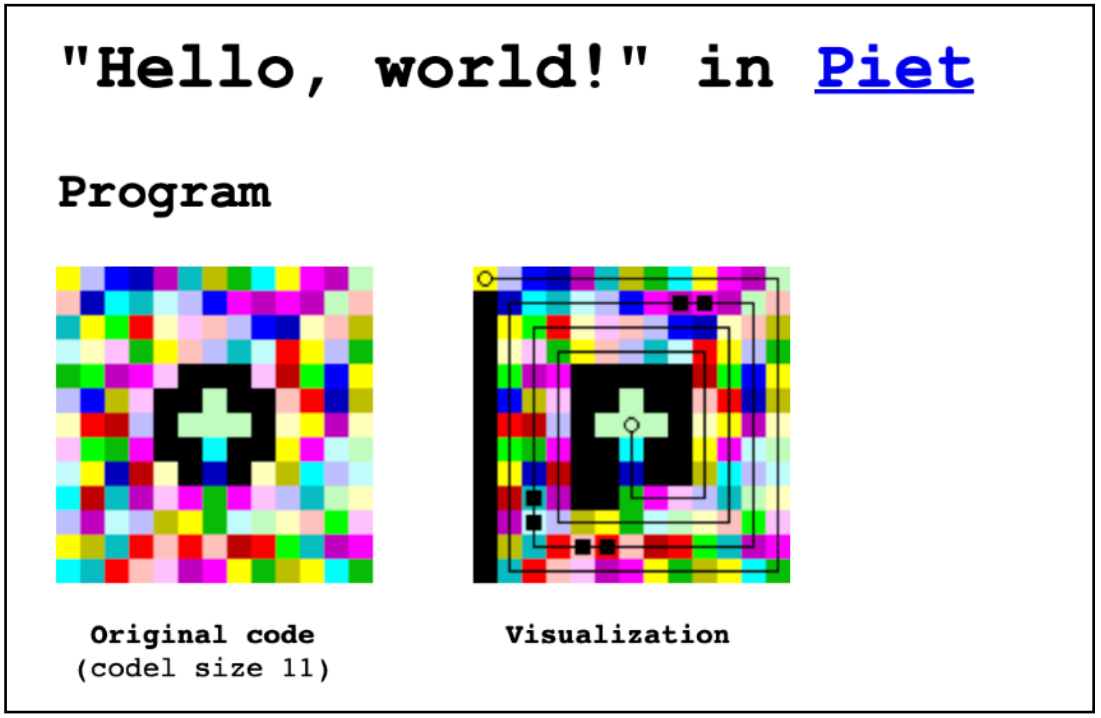
>	포인터 증가
<	포인터 감소
+	포인터가 가리키는 바이트의 값을 증가
-	포인터가 가리키는 바이트의 값을 감소
.	포인터가 가리키는 바이트 값을 아스키 코드 문자로 출력한다.
,	포인터가 가리키는 바이트에 아스키 코드 값을 입력한다. 쉽게 말해서 입력 받는 역할이다.
[	포인터가 가리키는 바이트의 값이 0이 되면 짝이 되는]로 이동한다. 의사코드로는 `while(*ptr != 0) {...}` 이다.
]	포인터가 가리키는 바이트의 값이 0이 아니면 짝이 되는 [로 이동한다.
EOF(\0)	문자로 드러나지 않는 토큰이며 프로그램을 종료한다.

- # BrainFuck - Hello, World!

http://esoteric.sange.fi/brainfuck/impl/interp/i.html

Piet

- 몬드리안의 그림을 보고 만든 언어, RGB값으로 코드를 작성



https://www.dangermouse.net/esoteric/piet.html

난해한 혀엉... 언어

Hello, world!

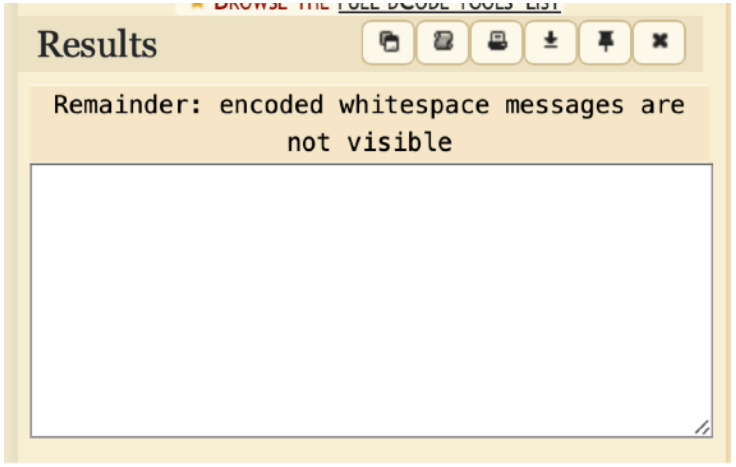
```
혀어어어어어어엉..... 핫. 혀엉..... 흑... 하앗... 흐윽... 형.  하앙.
혀엉.... 하앙... 흐윽... 항. 항. 형... 하앙. 흐으윽... 형... 흡... 혀엉..
하아아앗. 혀엉.. 흡... 흐읍... 형.. 하앗. 하아앙... 형... 하앙... 흐윽...
혀어어엉.. 하앙. 항. 형... 하앙. 혀엉.... 하앙. 흑... 항. 형... 흡  하앗.
혀엉..... 흑. 훗
```

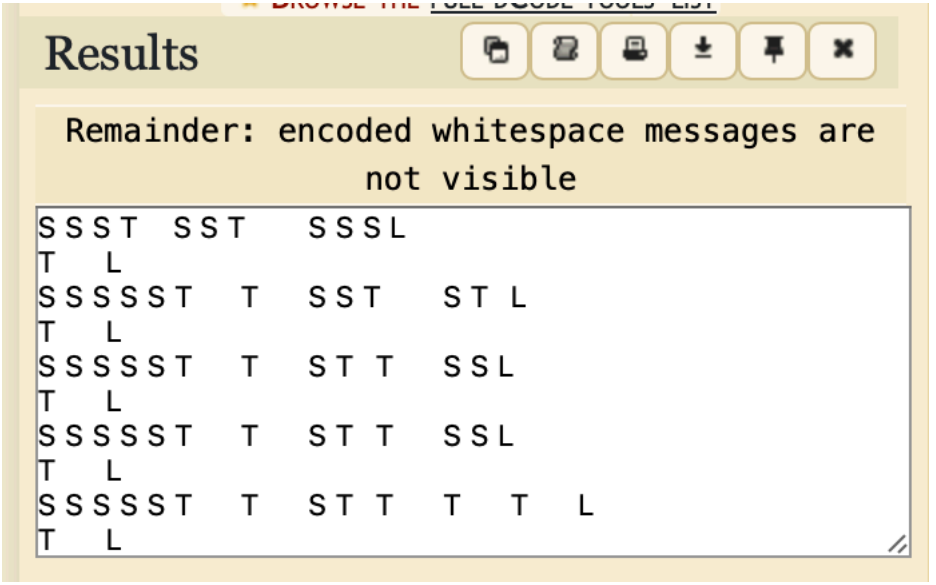
- `형`, `혀엉`, `혀어엉`, `혀어어엉` ... : 글자 수와 마침표 개수를 곱한 값을 현재 스택에 저장합니다.
  - 예를 들어 `혀어엉....`은 `12`를 현재 스택에 넣습니다.
  - `혀`와 `엉` 사이에 `엉`을 제외한 한글 음절 문자를 추가적으로 넣어 글자 수를 늘릴 수 있습니다.

https://gist.github.com/xnuk/d9f883ede568d97caa158255e4b4d069

Whitespace

- 공백문자(스페이스, 탭, 개행문자 등)으로만 만들어진 언어





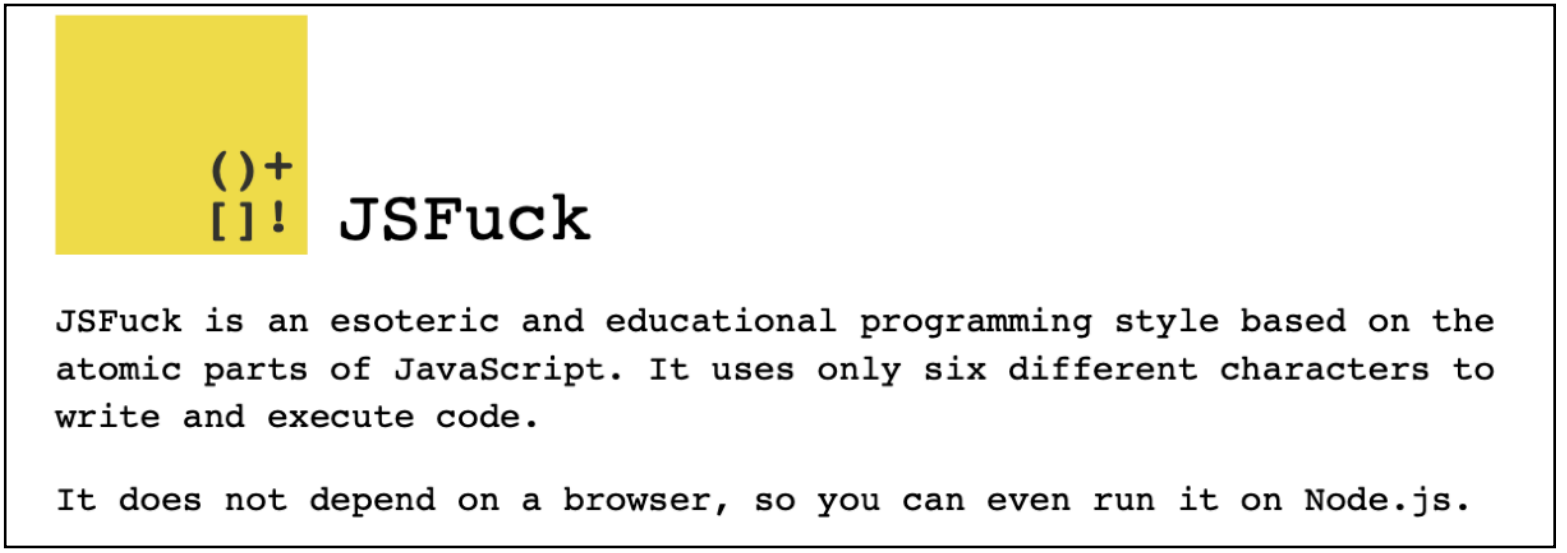
https://www.dcode.fr/whitespace-language

02

JSFuck

JS Fuck

- https://jsfuck.com
- 총 6개의 문자만으로 JavaScript를 구현하는 '난해한 프로그래밍 스타일'
- [ ] ( ) ! +



예제

alert(1) Encode Eval Source Run In Parent Scope

```
(![]+[])[+!+[]]+(![]+[])[!+[]+!+[]]+(!![]+[])[!+[]+!+[]+!+[]]+(!![]+
[])[+!+[]]+(!![]+[])[+[]]+([][(![]+[])[+[]]+(![]+[])[!+[]+!+[]]+(!![]+
[])[+!+[]]+(!![]+[])[+[]]]+[])[+!+[]+[!+[]+!+[]+!+[]]]+[+!+[]]+([+
[]]+![]+[][(![]+[])[+[]]+(![]+[])[!+[]+!+[]]+(![]+[])[+!+[]]+(!![]+[])
[+[]]])[!+[]+!+[]+[+[]]]
```

297 chars Run This

03

실전 예제

- **사전 지식 1 : 문자열을 배열처럼 index로 접근할 수 있다.**

  - "abc"[0] = "a"
  - "abc"[1] = "b"
  - "abc"[2] = "c"

- **사전 지식 2 : 오브젝트 간의 + 연산시 형변환이 이뤄진다.**

(원시 자료형 > .valueOf() > .toString() 순으로 형변환 시도)

- 1 + "a" = "1" + "a" = "1a"
- 1 + true = 1 + 1 = 2
- "a" + [] = "a" + "" = "a"

- **사전 지식 3 : unary plus 연산자 (단항 + 연산자)도 형변환 된다.**

(ToNumber() > .valueOf() > .toString() 순으로 형변환 시도)

- +"1" = 1
- +"" = 0
- +true = 1
- +[] = +"" = 0

■ 사전 지식 4 : Truthy, Falsy

• boolean 값이 기대되는 상황에서

0, "", null, undefined, NaN은 **false**로 취급, 그 외는 **true**로 취급

if (null) {
  console.log(true)
} else {
  console.log(false)
}
 	
-> 결과는 false
	if ([]) {
	  console.log(true)
	} else {
	  console.log(false)
	}
	
	-> 결과는 true

- **"b" + "a" + + "a" + "a" = ?**

"b" + "a" + + "a" + "a"

↳ + "a" => NaN

↳ "b" + "a" + **NaN** + "a"

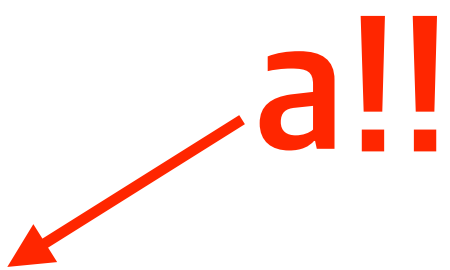
↳ **"baNaNa"**

실전 1 : []()!+ 만으로 숫자 만들기

- [] → [] (이 자체로는 비어있는 배열)
- ![] → false
- +![] → +false → 0
- !![] → true
- +!![] → 1
- +!![] + (+!![]) → 2 또는 !+[]+!+[]

실전 2 : []()!+ 만으로 "a" 만들기

- [] → []
- ![] → false
- ![] + [] → false + "" → "false"
- "false"[1] → (![] + [])[1] → **(![] + [])[+!![]]** → **"a"**



- **참고 자료 : jsfuck의 MAPPING 정보**

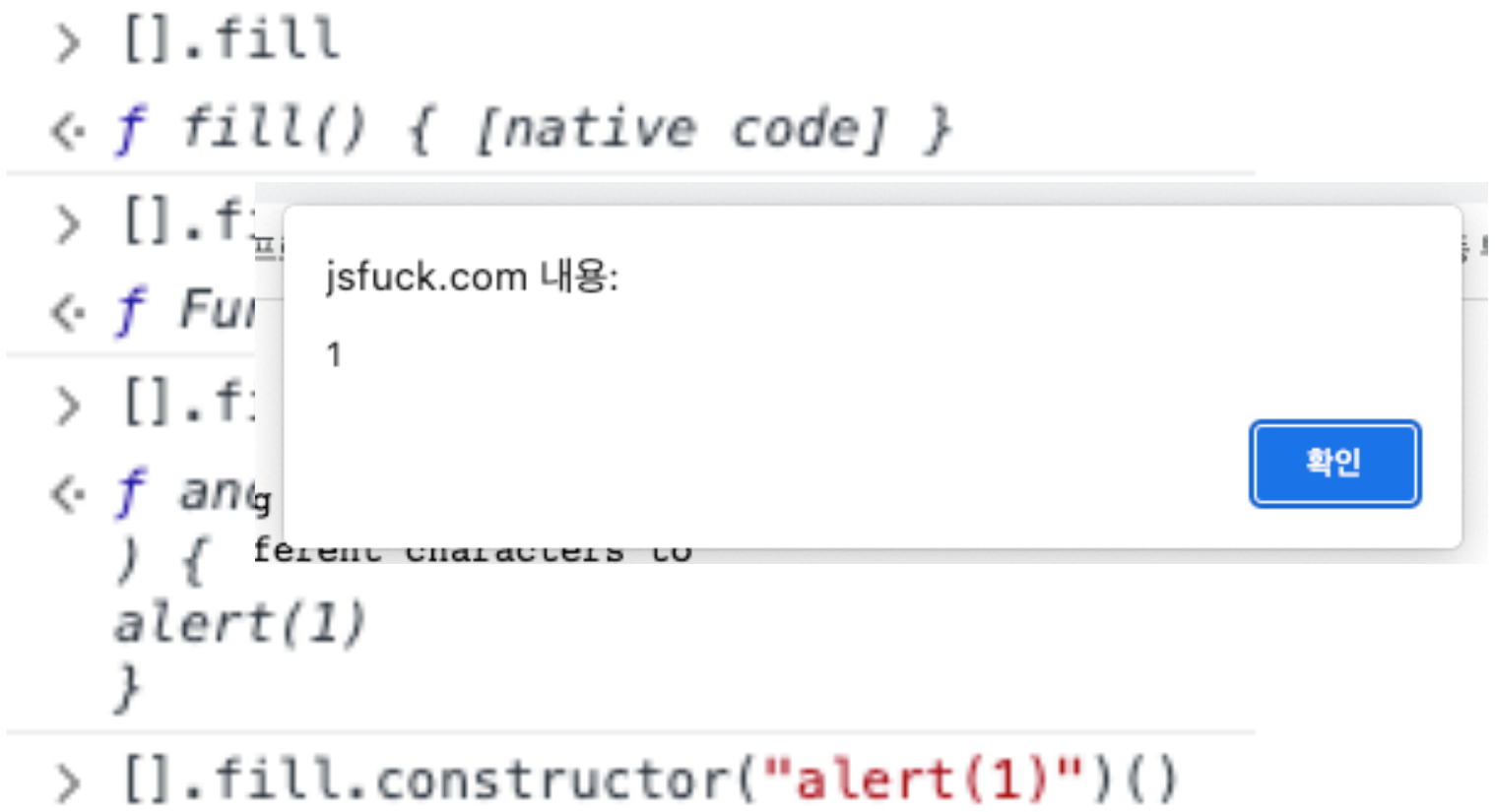
jsfuck / jsfuck.js

Code | Blame | 352 lines (300 loc) · 9.88 KB

```
24  const MAPPING = {
25    'a':   '(false+"")[1]',
26    'b':   '([]["entries"]()+"")[2]',
27    'c':   '([]["flat"]+"")[3]',
28    'd':   '(undefined+"")[2]',
29    'e':   '(true+"")[3]',
30    'f':   '(false+"")[0]',
31    'g':   '(false+[0]+String)[20]',
32    'h':   '(+(101))["to"+String["name"]](21)[1]',
33    'i':   '([false]+undefined)[10]',
34    'j':   '([]["entries"]()+"")[3]',
35    'k':   '(+(20))["to"+String["name"]](21)',
```

https://github.com/aemkei/jsfuck/blob/main/jsfuck.js

- **실전 3 : 함수 호출하기 (1)**


> [].fill
f fill() { [native code] }
> [].f
f Fu
> [].f
f an
) {
alert(1)
}
> [].fill.constructor("alert(1)")()
jsfuck.com 내용:
1
확인

▪ 실전 3 : 함수 호출하기 (2)

- [].fill.constructor("alert(1)")()

↓

- [["fill"]["constructor"]("alert(1)")()

실전 3 : 함수 호출하기 (3)

- []["fill"]["constructor"]("alert(1)")()

↓

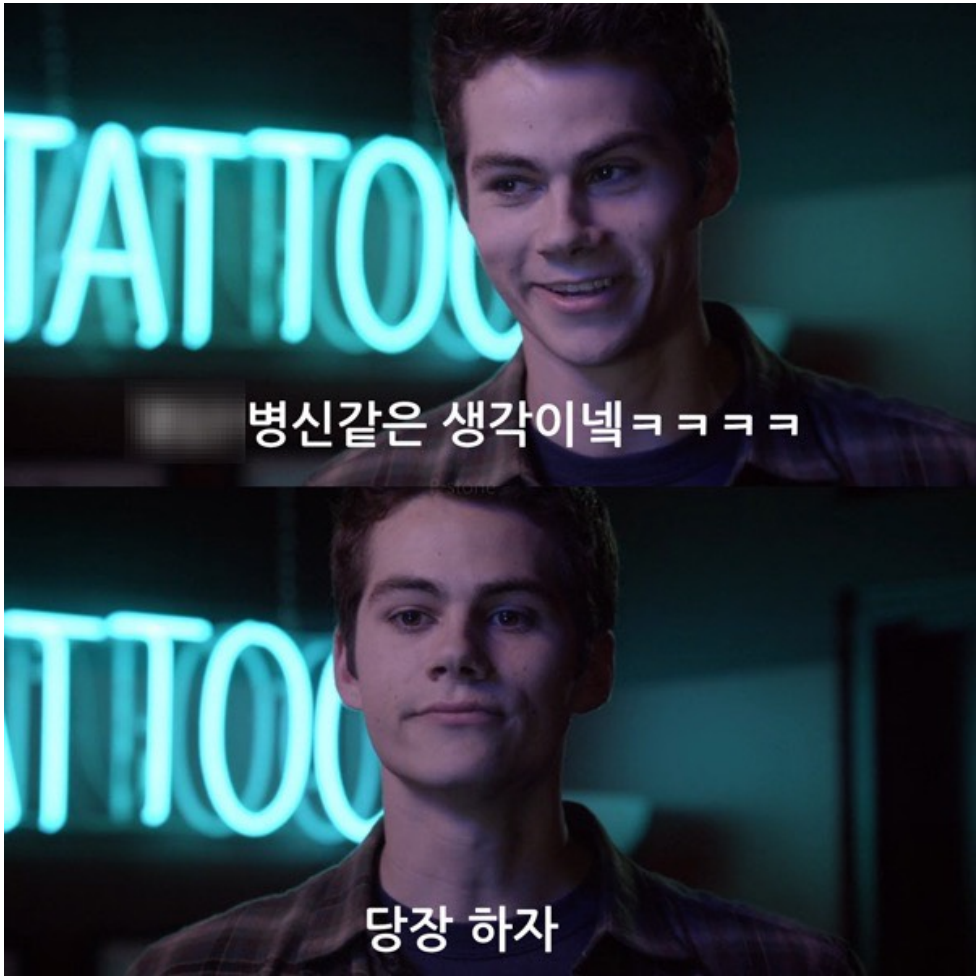
```
(![]+[])[+!+[]]+(![]+[])[!+[]+!+[]]+(!![]+[])[!+[]+!+[]+!+[]]+
(!![]+[])[+!+[]]+(!![]+[])[+[]]+([][(![]+[])[+[]]+(![]+[])[!+[]
+!+[]]+(![]+[])[+!+[]]+(!![]+[])[+[]]]+[])[+!+[]+[!+[]+!+[]+!
+[]]]+[+!+[]]+([+[]]+![]+[][(![]+[])[+[]]+(![]+[])[!+[]+!+[]]+
(![]+[])[+!+[]]+(!![]+[])[+[]]])[!+[]+!+[]+[+[]]]
```

04

마치며

- **이런 걸 왜 하죠?**

재밌잖아요.



감사합니다.