MEGA BRAIN
IJU Web Application Developer Society

자료구조와 자료형

3조 최원석, 문상원

3조는 이렇게 스터디 했습니다!

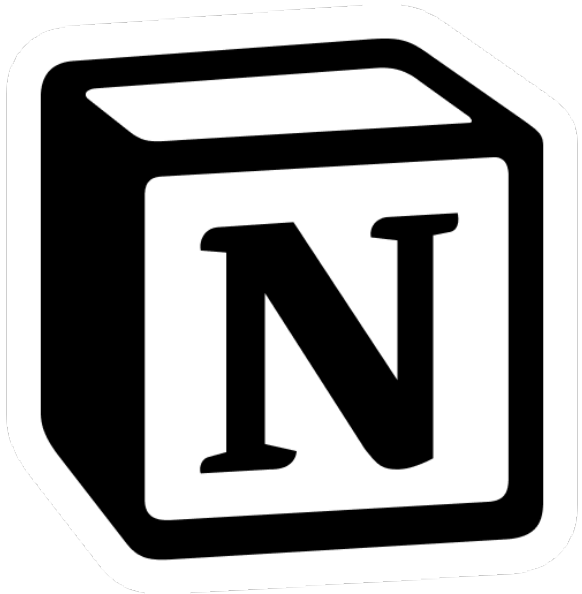
1. 아래와 같이 파트를 나뉘어 따로 공부함
2. 공부 한 뒤, 만나서 각자 했던 공부내용을 설명하며 스터디 진행
3. 책으로 이해하기 어려운 내용은 레퍼런스를 참조해서라도 무조건 이해하고 넘어가기
4. JS를 처음 배우는 입장에서 스터디 로그를 작성, 작성 중 기본에서 긁어올게 있으면 가져와 내용을 더하여 이해도 증진

| 최원석 회원

1. 원시값의 메서드
2. 숫자형
5. 배열과 메서드
6. Iterable 객체
9. Object.keys, values, entries
10. 구조 분해 할당

| 문상원 회원

3. 문자열
4. 배열
7. 맵과 셋
8. 위크맵과 위크셋
11. Date 객체와 날짜
12. JSON메서드



+

자료구조와 자료형

Status Empty

Assign 문상원 Wonseok Choi

Team 3조

Add a property

박영건 May 6
LGTM🙂

Add a comment...

1. JS/자료구조와 자료형 - 1. 원시값의 메서드 + 원시값과 객체
2. JS/자료구조와 자료형 - 2. 숫자형
3. JS/자료구조와 자료형 - 3. 문자열
4. JS/자료구조와 자료형 - 4. 배열
5. JS/자료구조와 자료형 - 5. 배열과 메서드
6. JS/자료구조와 자료형 - 6. iterable 객체
7. JS/자료구조와 자료형 - 7. 맵과 셋
8. JS/자료구조와 자료형 - 8. 위크맵과 위크셋
9. JS/자료구조와 자료형 - 9. Object.keys, values, entries
10. JS/자료구조와 자료형 - 10. 구조 분해 할당
11. JS/자료구조와 자료형 - 11. Date 객체와 날짜

1. 원시값의 메서드, 객체

JavaScript는 원시값과 객체, 두 가지 형태의 형이 있습니다.
그러면, 원시값과 객체. 이것은 무엇이며, 차이는 무엇일까요?

정말 쉽게 풀어 말 하면 원시값은 수정이 안되는 값이며,
객체는 변경 가능한 값입니다.

하지만 둘의 전달 방법은 완전히 틀립니다.
이에 대하여 자세히 다뤄보겠습니다.

💡 짚고 넘어갑시다. JS의 자료형

1. `boolean` : 너무나 친숙한 부울린! 즉, true or false 두 가지의 값을 가질수 있습니다.

2. `undefined` : 값을 할당하지 않는 변수입니다.

3, 4. `number`과 `bigint`
`number`은 -2^53 -1부터 2^53-1까지 표현할수 있습니다.
부동소수점 숫자 외에도 +Infinity, -Infinity, NaN (Not a Number) 등을 가질 수 있습니다.

`bigint`은 임의 정밀도로 정수를 나타 낼 수 있습니다. `number`의 범위를 넘어서는 큰 정수를 안전하게 저장하고 연산 할 수 있습니다. (big int라고 외우면 편할 것 같습니다!)

추가로 `number`과 `bigint`는 혼합해 연산이 불가능합니다.

5. `string` : 텍스트 데이터를 나타낼 때 사용합니다. 16비트 부호 없는 정수 값 요소로 구성된 집합입니다. 재밌는 점은 C언어같은 언어와는 달리 JS에서의 string 문자열은 불변합니다. = 문자열을 생성한 후 바꾸는 것이 불가능합니다.

1. 원시값의 메서드, 객체

원시값 = 원시 타입의 값 (Immutable value)
예) string, boolean, null과 JS를 처음 접하는 개발자에게는 생소한
symbol, number, bigint, undefined 이렇게 총 7가지가 존재함

•**원시값은 선언하면 변경이 불가능한 고유값입니다.**

•원시 값을 변수에 할당하면 변수에는 실제 값이 저장됩니다.

•원시 값을 다른 변수에 할당하면 원본 원시 값이 복사되어 전달됩니다. (Pass by value)

쉽게 말해, 아예 다른 주소로 값이 '복사' 됩니다. (복사된 값은 원본 값과 관계가 없는 완전히 다른곳으로 할당되게 되며, 수정해도 서로 영향을 주고 받지 않습니다.)

1. 원시값의 메서드, 객체

객체 = 객체 타입의 값 (Mutable value)

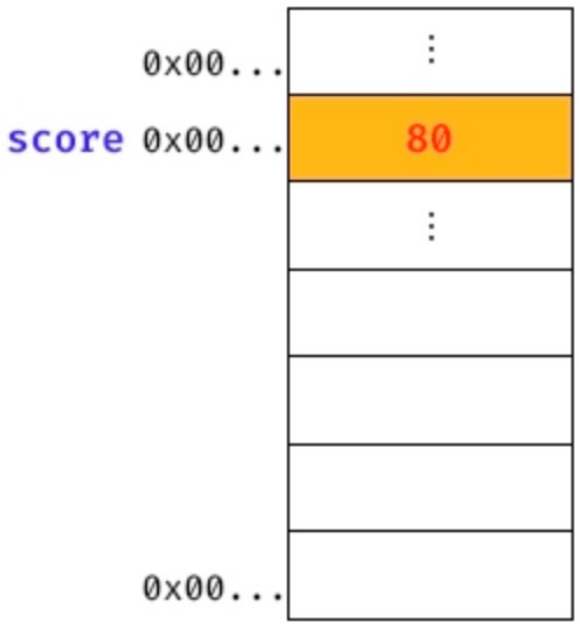
•객체 타입의 값, 즉 객체는 변경 가능한 값입니다.

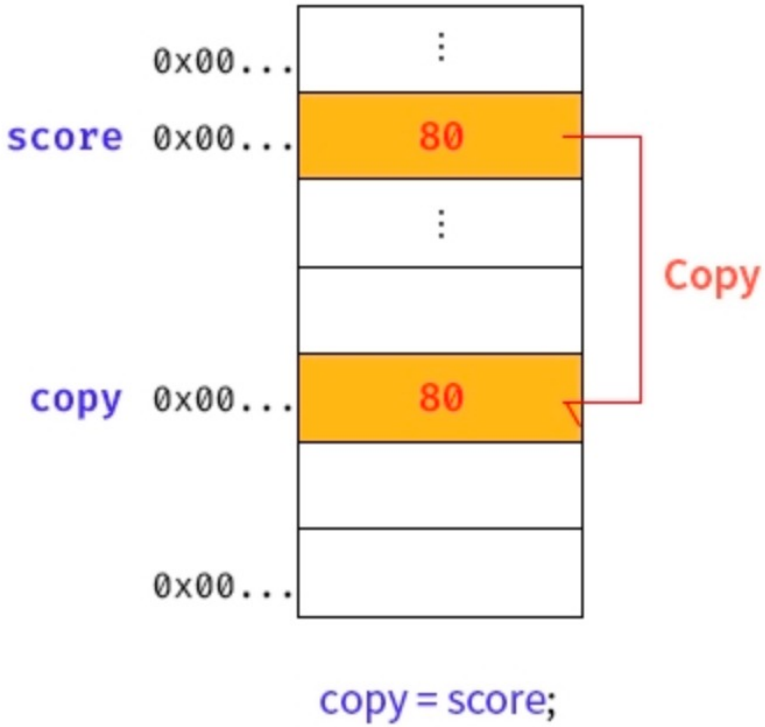
•객체를 변수에 할당하면 변수에는 참조 값이 저장됩니다.

•객체를 가르키는 변수를 다른 변수에 할당하면 원본의 참조 값이 복사되어 전달됩니다. (Pass by reperence)

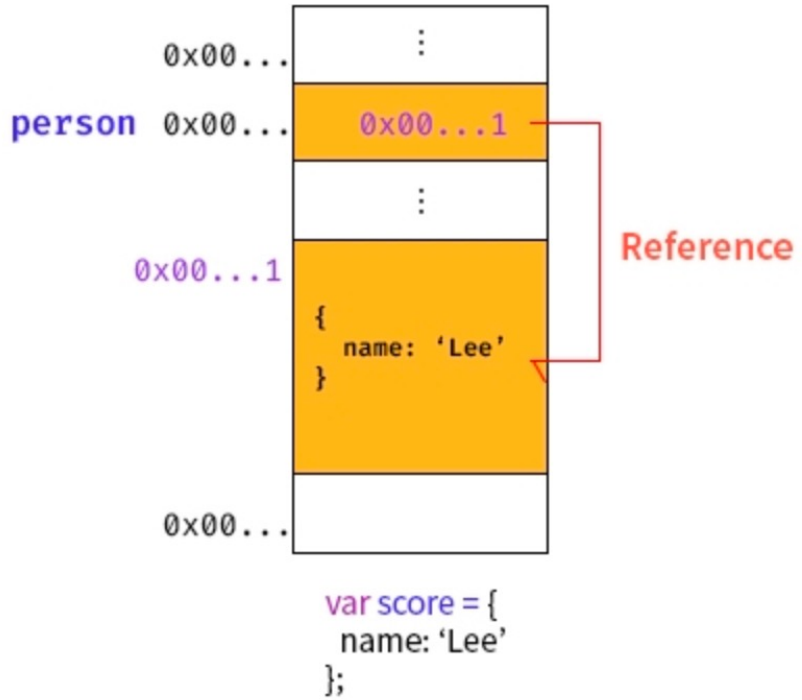
•여기서 말하는 참조 값이란, 생성된 객체가 저장된 메모리 공간의 주소 그 자체입니다. (원시값과는 다르게 객체는 복사된 값을 수정하면 원본값과 서로 영향을 주고 받습니다.)

1. 원시값의 메서드, 객체





원시값 (고유값)



객체 (변경 가능 값)

var, let, const (변수 선언 방식)

var (중복 선언해도 오류가 안나므로 이제는 사용 안함)

let : 재선언은 불가능하나, 변수에 재할당이 가능합니다. (원시 데이터 타입)

```
let name = 'wonseok'
    console.log(name)
      // let형 name = 'wonseok'이 선언됩니다.

let name = 'sangwon'
    console.log(name)
      // 재선언으로 인하여 SyntaxError이 나게 됩니다.

name = 'react'
    console.log(name) //react
      // 하지만 let은 이런식으로 재할당이 가능하죠.
```

```
const name = 'wonseok'
    console.log(name)
      // const형 name = 'wonseok'이 선언됩니다.

const name = 'sangwon'
    console.log(name)
      // 재선언으로 인하여 SyntaxError이 나게 됩니다.

name = 'react'
    console.log(name) //react
      // const는 재할당도 불가능하죠.
```

const : 변수 재선언, 변수 재할당 모두 불가능합니다. (객체)

2. 숫자형

•모던(현대의) 자바스크립트는 숫자를 나타내는 두가지 자료형을 지원합니다.

1.일반적인 숫자는 64비트 형식의 배정밀도 부동소수점 숫자 (IEEE-754)에 저장됩니다.
2.임의의 길이를 가진 정수는 bigint 숫자로 나타낼수 있습니다.

•숫자를 입력하는 다양한 방법

```
let billion = 1000000000;
```

```
let billion = 1e9; // 10억, 1과 9개의 0
alert( 7.3e9 ); // 73억 (7,300,000,000)

//1e3 = 1 * 1000
//1.23e6 = 1.23 * 1000000
```

💡 즉, 'e'는 e 왼쪽의 수에 e 오른쪽에 있는 수만큼의 10의 거듭제곱을 곱합니다!

```
let ms = 0.000001;
```

```
let ms = 1e-6; // 1에서 왼쪽으로 6번 소수점 이동

// 10을 세 번 거듭제곱한 수로 나눔
1e-3 = 1 / 1000 (=0.001)
// 10을 여섯 번 거듭제곱한 수로 나눔
1.23e-6 = 1.23 / 1000000 (=0.00000123)
```

💡 이렇게 'e' 우측에 음수가 있으면, 이 음수의 값만큼 10을 거듭제곱한 수로 나누는것을 의미합니다.

•어림수 구하기

`Math.floor`

소수점 첫째 자리에서 내림(버림)
즉, 3.1은 3이 되며, -1.1은 -2가 됩니다.

`Math.round`

소수점 첫째 자리에서 반올림
즉, 3.1은 3, 3.6은 4, -1.1은 -1이 됩니다.

`Math.ceil`

소수점 첫째 자리에서 올림
즉, 3.1은 4, -1.1은 1이 됩니다.

`Math.trunc` (Internet Exploler에서는 미지원)

소수부를 무시합니다.
3.1은 3이 되고, -1.1은 -1이 됩니다.

각 내장 함수의 차이를 표로 나타내면 다음과 같습니다.

	`Math.floor`	`Math.ceil`	`Math.round`	`Math.trunc`
`3.1`	`3`	`4`	`3`	`3`
`3.6`	`3`	`4`	`4`	`3`
`-1.1`	`-2`	`-1`	`-1`	`-1`
`-1.6`	`-2`	`-1`	`-2`	`-1`

•부정확한 계산

숫자는 내부적으로 `64비트 형식 IEEE-754`로 표현되기 때문에,
숫자를 저장하려면 정확히 64비트가 필요합니다.

그런데, 숫자가 너무 커지면 64비트 공간이 넘쳐 `Infinity`로 처리됩니다.

0.1 + 0.2 == 0.3은 True일까요? 놀랍게도, False입니다.

```
alert( 0.1 + 0.2 == 0.3 ); // false
```

실제 결과는..

```
alert( 0.1 + 0.2 ); // 0.30000000000000004
```

•부정확한 계산

JS에 극한 되어있는 문제가 아닌 다른 언어 (C, Java, PHP, Ruby) 등에서도 이슈가 되는 현상이며,
이러한 문제를 해결 할 방법 중 가장 신뢰적인 방법은 `toFixed(n)` 메소드를 이용하는 것 입니다.

위 메소드를 이용하여, 어림수를 만들어 정확하게 계산하는것이죠.

```
let sum = 0.1 + 0.2;
alert( sum.toFixed(2) ); // 0.30
```

다만 주의 할 점은, toFixed 메소드는 항상 문자열을 반환한다는 점 입니다.
숫자형으로 강제 형 변환을 하려면 '단항 덧셈 연산자'를 사용해주세요.

```
let sum = 0.1 + 0.2;
alert( +sum.toFixed(2) ); // 0.3
```

•그 외 기타 수학적 함수 (내장 객체 Math)

1. `Math.random()` : 0과 1사이의 난수를 반환합니다. (단, 1은 제외)

```
alert( Math.random() ); // 0.1234567894322
alert( Math.random() ); // 0.5435252343232
```

2. `Math.max(a, b, c...)` / `Math.max(a, b, c...)` : 인수 중 최대/최솟값을 반환합니다.

```
alert( Math.max(3, 5, -10, 0, 1) ); // 5
alert( Math.min(1, 2) ); // 1
```

3. `Math.pow(n, power)` : n을 power번 거듭제곱한 값을 반환합니다.

```
alert( Math.pow(2, 10) ); // 2의 10제곱 = 1024
```

9. Object.keys, values, entries

일반 객체 형식 자료구조 변환에 어려움을 겪을때 이 삼형제가 유용하게 사용된다.
일반 객체를 순회(iteration)할때 유용하게 사용되는 메소드이다.

객체 = 요소의 집합, 요소 = 키와 값으로 구성

Object.keys(obj) //객체의 키만 담은 배열을 반환합니다.

Object.values(obj) //객체의 값만 담은 배열을 반환합니다.

Object.entries(obj) //[키, 값] 쌍을 담은 배열을 반환합니다.

```
const user = {
  name: "Daisy",
  food: "pizza"
};
```

- Object.keys(user) = `["name", "food"]`
- Object.values(user) = `["Daisy", "pizza"]`
- Object.entries(user) = `[ ["name","Daisy"], ["food","pizza"] ]`

이렇게 일반 객체에서 배열로 변환함으로써 배열의 편리한 메소드를 사용할수 있는것이 장점이다!

더불어 알아두면 좋은것, Object.fromEntries

`Object.entries(obj)` 은 객체를 [키, 값] 쌍을 담은 배열으로 반환한다. 그럼 이렇게 변환된 배열을 다시 객체로 바꿀 순 없을까? 이때 사용하는 메서드가 `Object.fromEntries(array)` 이다.

예시)

```
const user = {
  name: "Daisy",
  food: "pizza"
};

const updatedUser = Obejct.fromEntries(Object.entries(user).map(([key,value])
 => [key, value + "updated"]));
```

결과)

`{name: "Daisyupdated", food: "pizzaupdated"}`

이렇게 객체를 배열로 변환 한 뒤 배열의 메소드를 이용하여 데이터를 원하는 대로 변경한 뒤 다시 객체로 변환 할 수도 있다.

경청해주셔서 감사합니다.