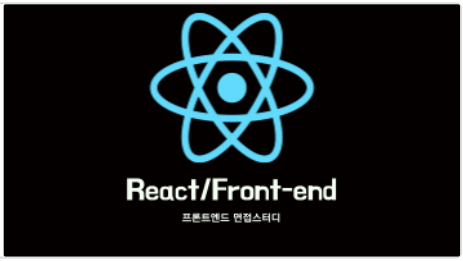
[Javascript] 불변성이란 무엇인가(상)



리액트를 공부해본 당신....

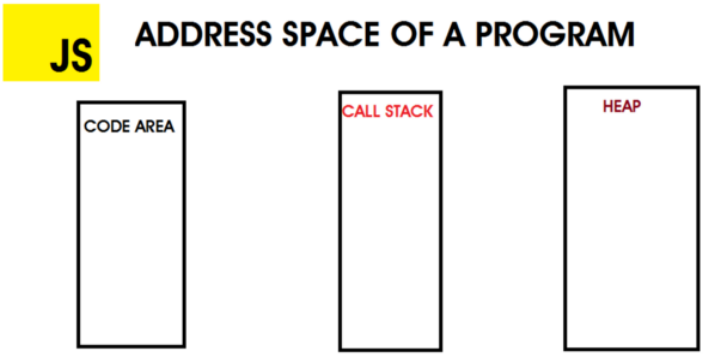
리액트가 지향하는 개념 중 하나인 **"불변성"**에 대해 알고 계시나요?

저는 useState, spread 연산자를 사용하면 되는 건줄로만 알고 있었는데... 정확히는 ***"왜?🤔"*** 사용하는지 모르겠더라구요...!!!!!

그래서 이번 시리즈는 **"리액트의 불변성"** 에 대해 알아보도록 하겠습니다.

그 전에 Javascript의 메모리 구조와 데이터 타입을 통해서 정확히 "불변성"이 무엇인지, "불변성"은 어떻게 지키는 것인지 알아보는 시간을 갖도록 하겠습니다! *(그래서 상 편 ㅋㅅㅋ)*

1. JS의 메모리 구조와 데이터 타입에 대해 알아보자..



Javascript 엔진은 **Call Stack**과 **Heap** 두 가지의 메모리 공간을 가집니다.

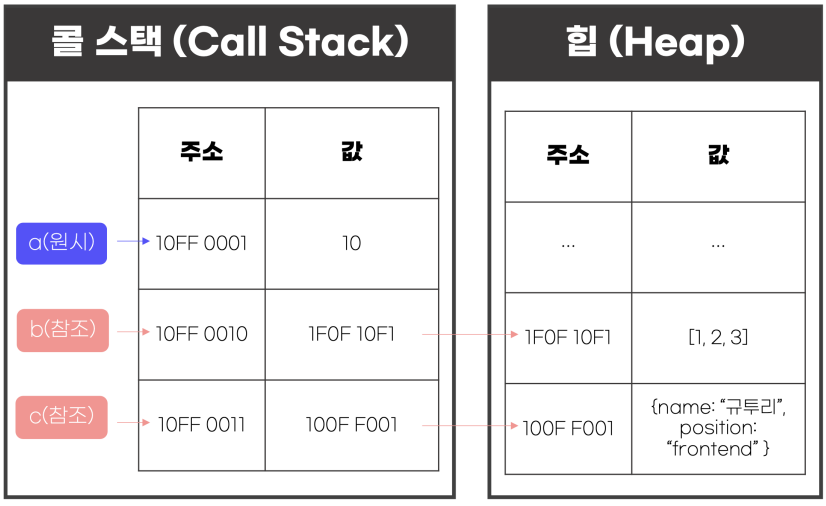
더보기

또, Javascript는 **원시타입(primitive type)**과 **참조타임(reference type)** 두 가지의 데이터 타입을 가집니다.

더보기

보통 **스택 메모리에 원시타입이, 힙 메모리에 참조타입이 저장 및 할당**됩니다.

아래의 그림을 보며 설명을 해보자면...



Number인 변수 a는 10 이라는 값을 콜 스택의 변수값에 10을 그대로 저장을 합니다.

반면, Array와 Object인 변수 b와 c의 경우 저장방식이 다릅니다.

변수 b와 c의 경우 실제의 값은 힙 메모리에 저장이 되며, 힙 메모리의 주소가 스택 메모리에 저장이 되는 것입니다.

정리하자면,

원시 타입의 경우 **스택 메모리에 값이 그대로 저장**이 되며,

참조 타입의 경우 **실제 값은 힙 메모리에 저장**이 되며, **저장된 힙 메모리의 주소값이 스택 메모리에 참조되어 저장**이 됩니다.

2. 타입별 변수 할당과 재할당에 대하여...

그래서 Javascript의 타입과 메모리가 불변성이랑 무슨 상관이 있느냐.....

아래 예시들을 보며 다시 알아봅시다!

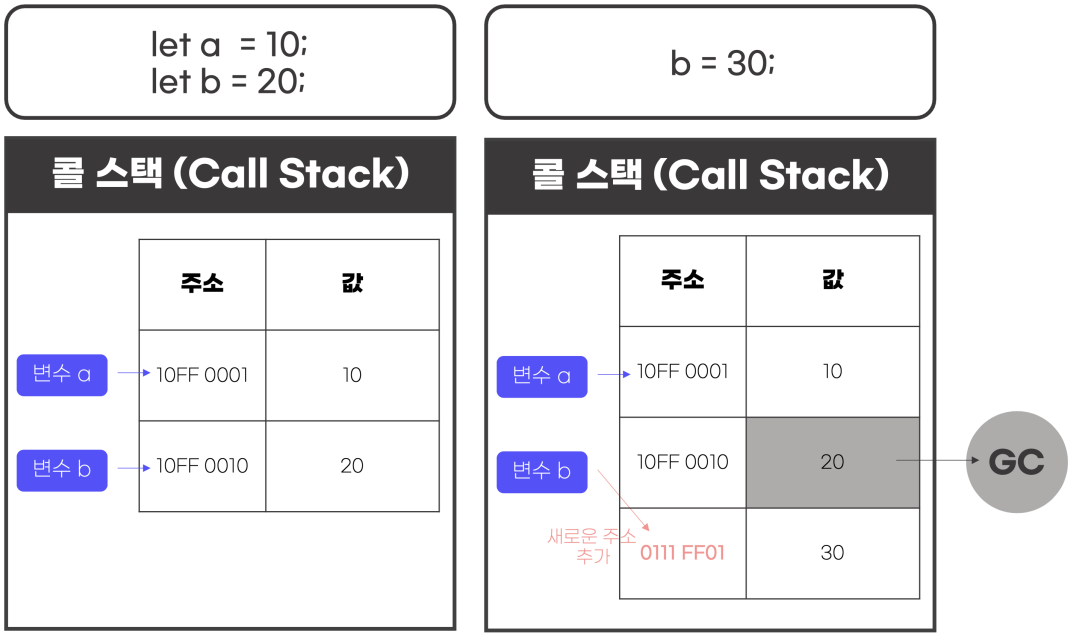
2-1. 원시 타입

각각의 Number 변수 a와 b가 있습니다.

이때, 저는 a는 10, b는 20이라고 할당하도록 하겠습니다.

Number는 원시타입이기 때문에 해당 변수값은 스택 메모리에 저장이 되겠죠?

그렇다면 b의 값을 30으로 변경하면 어떻게 될까요...? 🤔



오른쪽 그림의 변수 b를 자세히 봐주세요!

변수 b에 새로운 값 30을 재할당을 한다면.. 뭔가 b의 변수 값이 있던 자리에 쇽샥하고 값만 바뀔줄 알았나요? (네, 제가 그랬습니다)

원시타입의 변수는 변수값을 변경하라는 명령을 받았을 때, **기존 콜스택의 값을 변경하지 않고 새로운 주소를 추가**합니다!

추가된 주소는 변수 b가 바라보는 새로운 주소가 되며, 그 주소에 새로운 변수 값 30이 할당됩니다.

그렇다면 원래 값 20이 있던 주소는 어떻게 될까요..? 😢

더이상 **참조되지 않는 데이터는 가비지 컬렉터에 의해 적절한 시점에서 메모리가 해제**됩니다. (GC는 정말 똑똑하군요)

변수 b를 보면서 확인했지만, 원시타입의 경우 원본 값이 변하는 것이 아니라, 새로운 값이 추가되는 것을 확인했습니다.

이것이 우리가 그토록 궁금해하던 **"불변성"**입니다.

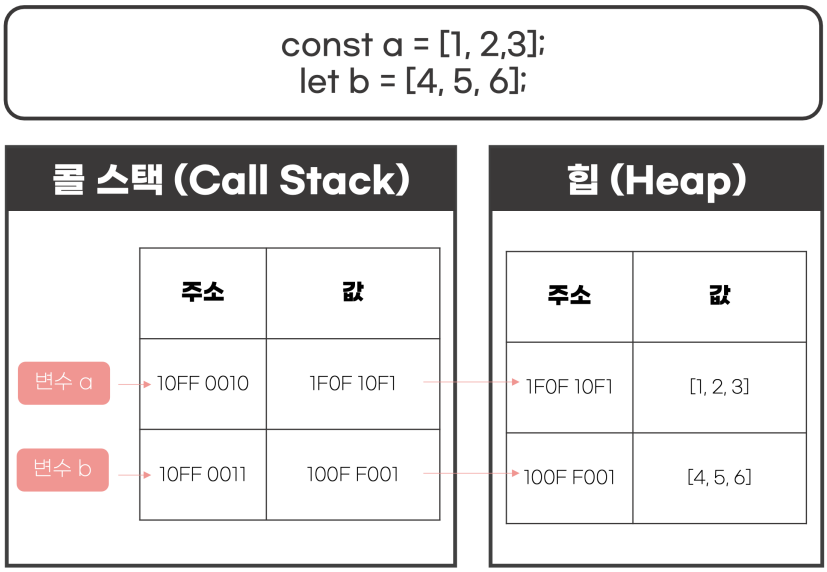
> 💡 즉,
>
> **불변성이란 메모리 영역에서 값이 변하지 않는 것**

2-2. 참조 타입

그렇다면 참조타입의 재할당은 어떻게 이루어질까요?

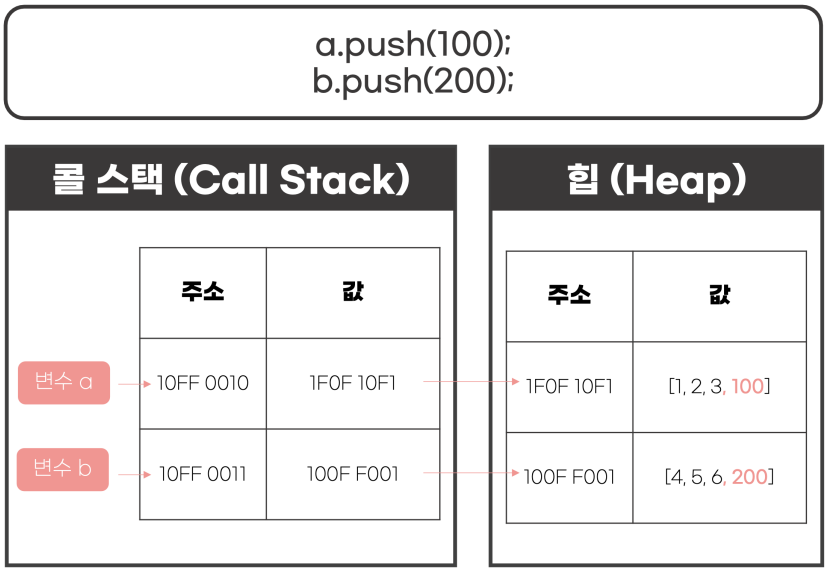
참조타입 Array 변수 a와 b가 있습니다.

앞서 말했듯이 실제 값은 힙 메모리에 저장이 되며, 그 주소값이 스택 메모리에 참조되는 것을 확인할 수 있습니다.



여기서 그렇다면 a 와 b에 새로운 값을 push 해볼까요?

a에는 100을 b에는 200을 추가해 기존의 배열과 다르게 변경해보도록 하겠습니다.



원시 타입과 마찬가지로 불변성이 지켜졌나요?

대답은 ***No!*** 입니다.

실제로는 변수 a, b가 바라보고 있는 스택 메모리의 값은 변경이 되지 않았지만, **힙 메모리에 있는 데이터 값이 변경이 되었기 때문에 불변성이 유지가 되지 않았습니다.**

즉, Javascript의 참조타입 데이터는 원시타입과 다르게 **동적**입니다.

우리는 동적인 참조타입을 주의하기 위해 불변성을 지켜야합니다.

아래 코드를 보면서 이해를 해볼까요?

```
const obj1 = {name: "규투리", position: "backend"};
const obj2 = obj1;

// obj1 변경
obj1.position = "frontend";

console.log(obj2.position)// frontend
```

우리는 obj1의 값만 변경했을 뿐인데, obj2.position 을 출력해보니 바뀐 obj1.position 이 출력되는 것을 확인할 수 있습니다.

다시 말하자면, obj1와 obj2가 **힙 메모리에 있는 같은 객체를 가리키고 있기에 서로 영향을 주는 상황**이 되는 것입니다.

3. 결론

원시 타입의 경우 값을 재할당할 시 새로운 메모리가 할당되어 스택 메모리의 주소값이 감지가 되기 때문에 불변성을 지킬 수 있지만,

참조 타입의 경우 메모리가 동적으로 변하기 때문에 불변성을 지키기 어렵습니다.

> 결론적으로
>
> **참조 타입은 서로에게 영향**
>
> **최대한 불변성을 유지**

그렇다면 다음 시간에 위의 개념을 가지고 리액트에서의 불변성에 대해 알아보도록 하겠습니다!😉

0. 들어가기 전

함수형 프로그래밍의 특징들에서 리액트와 같이 “불변성”을 지향한다.

함수형 프로그래밍의 특징 중 하나가 순수함수를 사용하는 것

▼ 순수함수란?

- 동일한 매개변수를 넣었을 대 동일한 리턴값을 출력하는 함수
- 외부의 값을 변경하는 사이드 이펙트가 일어나지 않는 조건을 지키는 함수를 뜻한다.

→ 여기서 외부의 값을 변경하지 않는다는 것이 **불변성**과 연관이 되어 있다.

1. React의 불변성이란?

불변성 : 값이나 상태를 변경할 수 없는 것

불변성의 장점

1. 복잡한 특징들을 단순화할 수 있다
2. 변화를 감지한다.

3. React에서 다시 렌더링하는 시기를 결정할 수 있다.