

# | new 연산자와 생성자 함수 |

| 이은주

2023.04.25

# 목 차


1  
객체 리터럴과 생성자 함수

2  
생성자 함수  
3  
생성자 함수 알고리즘

4  
메소드 추가  
5  
new를 사용하지 않았다면

# 객체 리터럴과 생성자 함수

## 객체 리터럴



```
1 let user1 = {  
2     name : 'lee',  
3     age : 30,  
4 }  
5 let user2 = {  
6     name : 'eun',  
7     age : 20,  
8 }  
9 let user3 = {  
10    name : 'joo',  
11    age : 40,  
12 }
```

## 생성자 함수



```
1 function User(name, age) {  
2     this.name = name;  
3     this.age = age;  
4 }  
5 let user1 = new User('lee', 30)  
6 let user2 = new User('eun', 20)  
7 let user3 = new User('joo', 40)
```

# 생성자 함수



```
1 function User(name, age) {  
2   this.name = name;  
3   this.age = age;  
4 }  
5 let user1 = new User('lee', 30)  
6 let user2 = new User('eun', 20)  
7 let user3 = new User('joo', 40)
```

생성자 함수는 함수 이름이  
첫글자가 대문자로 시작합  
니다.

## 생성자 함수



```
1  function User(name, age) {  
2    this.name = name;  
3    this.age = age;  
4  }  
5  let user1 = new User('lee', 30)  
6  let user2 = new User('eun', 20)  
7  let user3 = new User('joo', 40)
```

반드시 new 연산자를 붙여  
실행합니다.

# 생성자 함수의 알고리즘



```
1  function User(name, age) {  
2    // this = {}; 빈 객체 만들기  
3  
4    this.name = name;  
5    this.age = age; // this에 프로퍼티 추가  
6  
7    // return this; // 반환  
8  }  
9  
10 let user1 = new User('lee', 30)
```

## 메소드 추가



```
1  function User(name, age) {  
2    this.name = name;  
3    this.age = age;  
4    this.text = function () {  
5      console.log(this.age);  
6    };  
7  }  
8  let user1 = new User("lee", 30);  
9  user1.text();  
10
```

## new를 사용하지 않았다면



```
1 function User(name, age) {  
2   this.name = name;  
3   this.age = age;  
4 }  
5 let user1 = new User('lee', 30)  
6 let user2 = User('eun', 20)  
7 let user3 = new User('joo', 40)
```

---

```
> user1
```

```
< ▶ User {name: 'Lee', age: 30}
```

---

```
> user2
```

```
< undefined
```

---