

클래스의 기본 문법

class

getter와 setter

computed property

class field

class

객체 지향 프로그래밍에서 특정 객체를 생성하기 위해 변수와 메소드를 정의하는 일종의 틀.



```
1  class MyClass {  
2      // 여러 메서드를 정의할 수 있음  
3      constructor() { ... }  
4      method1() { ... }  
5      method2() { ... }  
6      method3() { ... }  
7      ...  
8  }
```

class



```
1  class User {  
2      constructor(name) {  
3          this.name = name;  
4      }  
5  
6      sayHi() {  
7          alert(this.name);  
8      }  
9  }  
10  
11 // 사용법:  
12 let user = new User("John");  
13 user.sayHi();  
14  
15 // User가 함수라는 증거  
16 alert(typeof User); // function
```

new User("John") 호출 시

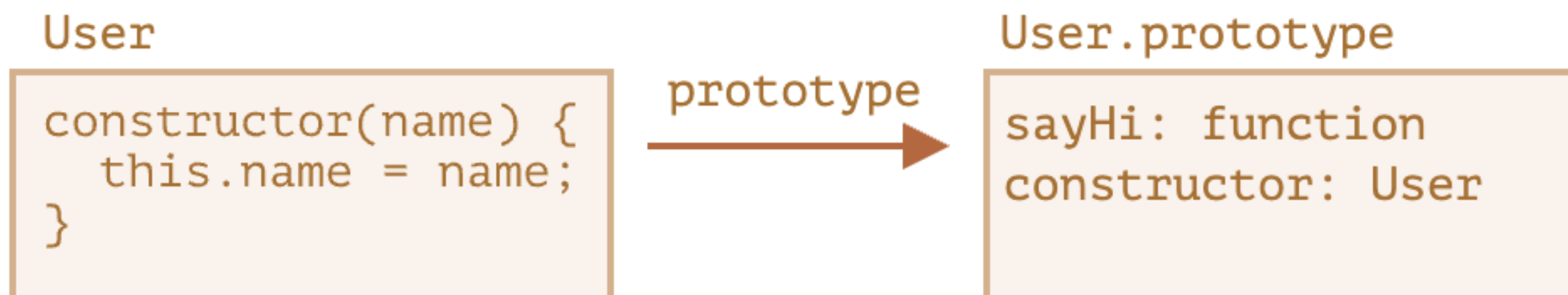
1. 새로운 객체(인스턴스)가 생성
2. 넘겨받은 인수와 함께 constructor가 자동으로 실행
이때 인수 "John"이 this.name에 할당

class

```
1  class User {  
2      constructor(name) {  
3          this.name = name;  
4      }  
5  
6      sayHi() {  
7          alert(this.name);  
8      }  
9  }  
10  
11 // 사용법:  
12 let user = new User("John");  
13 user.sayHi();  
14  
15 // User가 함수라는 증거  
16 alert(typeof User); // function
```


class User {...} 가 하는 일

1. constructor의 코드를 본문으로 갖는 User라는 함수를 생성
2. 클래스 내에서 정의한 메서드 sayHi를 User.prototype에 저장



class 클래스 표현식

함수처럼 클래스도 다른 표현식 내부에서 정의, 전달, 반환, 할당 가능



```
1 let User = class {  
2     sayHi() {  
3         alert("안녕하세요.");  
4     }  
5 };
```

getter setter

객체처럼 클래스도 **getter**나 **setter**를 지원

- constructor 메서드가 실행될 때 setter를 활성화
- 다른 메서드와 같이 User.prototype에 정의

```
1  class User {  
2    //생성자 메서드  
3    constructor(name) {  
4      // setter를 활성화합니다.  
5      this.name = name;  
6    }  
7  
8    //getter 메서드 - 획득  
9    get name() {  
10     return this._name;  
11   }  
12  
13   //setter 메서드 - 지정  
14   set name(value) {  
15     if (value.length < 4) {  
16       alert("이름이 너무 짧습니다.");  
17       return;  
18     }  
19     this._name = value;  
20   }  
21 }  
22  
23 let user = new User("박지민");
```

getter setter

```
6      }
7
8      //getter 메서드 - 획득
9      get name() {
10         return this._name;
11     }
12
13     //setter 메서드 - 지정
14     set name(value) {
15         if (value.length < 4) {
16             alert("이름이 너무 짧습니다.");
17             return;
18         }
19         this._name = value;
20     }
21 }
22
23 let user = new User("보라");
24 alert(user.name); // 보라
25
26 user = new User(""); // 이름이 너무 짧습니다.
27
```

객체처럼 클래스도 **getter**나 **setter**를 지원

- constructor 메서드가 실행될 때 setter를 활성화
- 다른 메서드와 같이 User.prototype에 정의
- **setter** - user.name 을 실행할 때 실행되는 코드
- **getter** - new User(value)를 실행할 때 실행되는 코드

computed property

```
1 class User {  
2     //이런식으로 해도 가능!!  
3     ["say" + "Hi"]() {  
4         alert("Hello");  
5     }  
6 }  
7  
8 new User().sayHi(); //사용가능
```

리터럴을 사용해 만든 객체처럼 클래스도
계산된 프로퍼티(**computed property**) 지원

- [...] 를 이용하여 메서드 이름 지정가능

class field

```
1 class User {  
2     // constructor 안이 아닌 그냥 밖에 '<프로퍼티 이름> = <값>'  
3     name = "보라";  
4  
5     //아울러 클래스 필드엔 복잡한 표현식이나 함수 호출 결과를 사용할 수 있습니다.  
6     prColor = prompt("좋아하는 색상을 알려주세요.", "연두");  
7 }  
8  
9 let user = new User(); //이때 prompt 나옴  
10 alert(user.name); // 보라  
11 alert(User.prototype.name); // undefined  
12 alert(user.prColor); // 연두
```

클래스 필드(class field) 문법

- 어떤 종류의 프로퍼티도 클래스에 추가가능
- constructor 메서드가 아닌 그냥 밖에 지정
- 구식에서는 지원은 안할 가능성이 있다

class field

```
1 class User {  
2     // constructor 안이 아닌 그냥 밖에 '<프로퍼티 이름> = <값>'  
3     name = "보라";  
4  
5     //아울러 클래스 필드엔 복잡한 표현식이나 함수 호출 결과를 사용할 수 있습니다.  
6     prColor = prompt("좋아하는 색상을 알려주세요.", "연두");  
7 }  
8  
9 let user = new User(); //이때 prompt 나옴  
10 alert(user.name); // 보라  
11 alert(User.prototype.name); // undefined  
12 alert(user.prColor); // 연두
```

클래스 필드(class field) 문법

- 어떤 종류의 프로퍼티도 클래스에 추가가능
- constructor 메서드가 아닌 그냥 밖에 지정
- 구식에서는 지원은 안할 가능성이 있다

class field**동적인 this 문제 해결****일반함수**

```
1 class Button {
2   constructor(value) {
3     this.value = value;
4   }
5   click() {
6     alert(this.value);
7   }
8 }
9
10 let button = new Button("안녕하세요.");
11
12 setTimeout(button.click, 1000); // undefined
```

화살표함수

```
1 class Button {
2   constructor(value) {
3     this.value = value;
4   }
5   click = () => {
6     alert(this.value);
7   };
8 }
9
10 let button = new Button("안녕하세요.");
11
12 setTimeout(button.click, 1000); // 안녕하세요.
```

- 메서드를 이벤트 리스너로 설정해야 할 때 특히 유용하다.
- 화살표 함수로 만들 경우, 각 Button 객체마다 독립적인 함수를 만들어주고
이 함수의 this를 해당 객체에 바인딩하여 this엔 항상 의도한 값이 들어가게 된다.

THANK YOU