객체

객체는 중괄호 `{}` 안에 키(key)와 값(value) 쌍으로 이루어져 있는 데이터 구조입니다. 값에는 문자열, 숫자뿐만 아니라 함수(메소드)도 들어갈 수 있죠. 중요한 부분은 객체를 생성하고 접근하는 방법입니다.

객체 생성과 접근:

1. **객체 생성**:

```
const person = {
    name: 'Black',
    age: 32,
    greet: function() {
        console.log('Hello');
    }
};
```

여기서 객체는 `name`, `age`, `greet`라는 속성을 가지고 있고, 이 속성 중 `greet`은 함수입니다.

2. **객체 접근**:
   - `person.name`은 'Black'을 반환하고,
   - `person.age`는 32를 반환합니다.
   - `person.greet()`를 호출하면 "Hello"라는 메시지가 출력됩니다.

`this` 키워드:

- `this`는 객체를 가리킵니다. 메소드 내에서 사용될 때는 그 메소드를 호출한 객체를 가리키죠. 예를 들어, `person.greet()` 내부에서 `this.name`을 호출하면 `person` 객체의 `name` 속성인 'Black'을 반환하게 됩니다.

1. `this`의 세 가지 상황

자바스크립트에서 `this`는 문맥에 따라 달라지며, 크게 세 가지로 나눌 수 있습니다.

- **일반 함수 호출**:
  - `this`는 전역 객체를 가리킵니다. 브라우저 환경에서는 `window` 객체를 가리키죠.
  - 예시:

```
function showThis() {
    console.log(this);  // window 객체
}
showThis();
```

- **메소드 호출**:
  - 객체의 메소드 내에서 `this`는 그 **객체 자체**를 가리킵니다.
  - 예시:

```
const person = {
    name: 'Alice',
    greet: function() {
        console.log(this.name);  // 'Alice'
    }
};
person.greet();  // this는 person 객체를 가리킴
```

- **생성자 함수에서**:
  - 생성자 함수 내에서 `this`는 새롭게 만들어진 **인스턴스 객체**를 가리킵니다.
  - 예시:

```
function Person(name) {
    this.name = name;
}
const p = new Person('Bob');
console.log(p.name);  // 'Bob'
```

2. 프로토타입 (Prototype)

프로토타입은 자바스크립트의 객체 상속을 담당하는 중요한 개념입니다. 모든 객체는 부모 객체로 **프로토타입**을 가지고 있으며, 자바스크립트에서 모든 함수와 객체는 기본적으로 프로토타입을 상속받습니다. 이를 통해 공통된 메소드를 모든 객체가 공유할 수 있습니다.

- **프로토타입 정의**:

```
function Animal(name) {
    this.name = name;
}
Animal.prototype.speak = function() {
    console.log(`${this.name} makes a noise.`);
};

const dog = new Animal('Dog');
dog.speak();  // 'Dog makes a noise.'
```

- **프로토타입의 역할**:
  - `Animal` 생성자 함수는 `speak` 메소드를 직접 가지고 있지 않지만, 프로토타입을 통해 `speak` 메소드에 접근할 수 있습니다.
  - 자바스크립트는 객체가 특정 메소드를 가지고 있지 않으면, 그 객체의 프로토타입을 통해 메소드를 찾습니다.

3. 프로토타입 체인 (Prototype Chain)

자바스크립트에서 객체는 프로토타입을 통해 부모 객체의 메소드나 속성에 접근할 수 있습니다. 만약 특정 객체에 없는 메소드를 호출하려 하면, 자바스크립트는 그 객체의 프로토타입에서 메소드를 찾습니다. 이것이 **프로토타입 체인**입니다.

- **예시**:

```
console.log(dog.hasOwnProperty('name'));  // true
console.log(dog.hasOwnProperty('speak'));  // false
// dog 객체에는 speak가 없지만, 프로토타입 체인을 통해 Animal.prototype.speak을 찾음
```

결론

`this`와 프로토타입은 자바스크립트의 객체지향 개념에서 중요한 요소입니다. `this`는 함수 호출 방식에 따라 가리키는 대상이 다르고, 프로토타입은 자바스크립트에서 상속과 코드 재사용을 가능하게 합니다. 이 두 가지 개념은 복잡하지만, 실습과 예제를 통해 천천히 익숙해질 수 있습니다.

이해가 어려운 부분은 실습을 통해 다시 한 번 복습해보시면 도움이 될 거에요!