**1. 객체 생성 방법**

**- 객체 리터럴**

: 가장 일반적인 객체 생성 방법은 객체 리터럴을 사용하는 것이다.

텍스트, 스크린샷, 폰트, 디자인이(가) 표시된 사진

자동 생성된 설명

**- ‘new Object()’ 구문**

: **‘Object’ 생성자**를 사용하여 객체를 생성할 수도 있다.

텍스트, 스크린샷, 폰트이(가) 표시된 사진

자동 생성된 설명

**- 생성자 함수**

: 생성자 함수는 **객체의 청사진을 정의**하는 데 사용된다.

텍스트, 스크린샷, 폰트이(가) 표시된 사진

자동 생성된 설명

**- ‘Object.create()’ 메서드**

: ‘Object.create()’ 메서드는 주어진 프로토타입 객체를 가지는 새 객체를 생성한다.

텍스트, 스크린샷, 폰트이(가) 표시된 사진

자동 생성된 설명

**1.2. 객체 속성**

**- 속성 접근**

: 객체의 속성은 점 표기법 또는 대괄호 표기법을 사용하여 접근할 수 있다.

텍스트, 스크린샷, 폰트, 디자인이(가) 표시된 사진

자동 생성된 설명

**- 속성 추가 및 수정**

: 객체의 속성은 **동적으로 추가하거나 수정**할 수 있다.

텍스트, 스크린샷, 폰트이(가) 표시된 사진

자동 생성된 설명

**- 속성 삭제**

: **‘delete’ 연산자**를 사용하여 객체의 속성을 삭제할 수 있다.

텍스트, 스크린샷, 폰트이(가) 표시된 사진

자동 생성된 설명

**1.3. 객체 메서드**

: 객체 메서드는 객체의 속성으로 정의된 함수이다.

텍스트, 스크린샷, 폰트이(가) 표시된 사진

자동 생성된 설명

**1.4. 객체 속성의 다양한 기능**

**- 계산된 속성 이름 (Computed Property Names)**

: 객체 리터럴 안에서 **대괄호를 사용**하여 **속성 이름을 동적으로 정의**할 수 있다

. 텍스트, 스크린샷, 폰트이(가) 표시된 사진

자동 생성된 설명

**- 속성 축약 (Property Shorthand)**

: 객체 리터럴에서 **변수명을 속성명으로 사용**할 때, 축약된 문법을 사용할 수 있다.

텍스트, 스크린샷, 폰트이(가) 표시된 사진

자동 생성된 설명

**1.5. 객체의 메서드**

**- ‘Object.keys()’**

: 객체의 **열거 가능한 속성 이름을 배열로 반환**한다.

텍스트, 스크린샷, 폰트이(가) 표시된 사진

자동 생성된 설명

**- ‘Object.values()’**

: 객체의 **열거 가능한 속성 값을 배열로 반환**한다.

텍스트, 스크린샷, 폰트이(가) 표시된 사진

자동 생성된 설명

**- ‘Object.entries()’**

: 객체의 **열거 가능한 속성 [key, value] 쌍을 배열로 반환**한다.

텍스트, 스크린샷, 폰트이(가) 표시된 사진

자동 생성된 설명

**- ‘Object.assign()’**

: 하나 이상의 소스 객체로부터 타깃 객체에 **속성을 복사**한다.

텍스트, 폰트, 스크린샷이(가) 표시된 사진

자동 생성된 설명

**- ‘Object.freeze()’**

: 객체를 동결하여 **속성을 추가, 삭제, 수정할 수 없게** 만든다.

텍스트, 스크린샷, 폰트이(가) 표시된 사진

자동 생성된 설명

**- ‘Object.seal()’**

: 객체를 밀봉하여 기존 속성을 수정할 수 있지만, 새로운 속성을 추가하거나 기존 속성을 삭제할 수 없게 만든다.

텍스트, 스크린샷, 폰트이(가) 표시된 사진

자동 생성된 설명

**1.6. 객체의 프로토타입**

: JavaScript는 프로토타입 기반 상속을 사용합니다. 객체는 다른 객체로부터 속성과 메서드를 상속받을 수 있다.

**- 프로토타입 체인**

: 객체는 **‘\_\_proto\_\_’** 속성을 통해 프로토타입 객체에 접근할 수 있으며, **프로토타입 체인을 통해 상속된 속성이나 메서드를 사용**할 수 있다.

텍스트, 스크린샷, 폰트이(가) 표시된 사진

자동 생성된 설명

**- ‘Object.create()’를 통한 상속**

: ‘Object.create()’ 메서드를 사용하여 객체를 생성하면 **특정 객체를 프로토타입으로 설정**할 수 있습니다.

텍스트, 스크린샷, 폰트, 디자인이(가) 표시된 사진

자동 생성된 설명

**- 생성자 함수와 프로토타입**

: 생성자 함수를 사용하여 **객체를 생성**할 때, **프로토타입을 통해 메서드를 공유**할 수 있다.

텍스트, 스크린샷, 폰트이(가) 표시된 사진

자동 생성된 설명

**2. 배열**

**-. 배열 요소 접근**

: 배열은 각 요소마다 고유한 인덱스를 가지며, 이를 사용하여 요소에 접근할 수 있다.

인덱스는 0부터 시작하며, 대괄호 표기법을 사용하여 요소에 접근한다.

텍스트, 폰트, 스크린샷이(가) 표시된 사진

자동 생성된 설명

**- 배열의 길이**

: 배열의 길이는 **length** 속성을 통해 알 수 있다.

텍스트, 폰트, 스크린샷이(가) 표시된 사진

자동 생성된 설명

**- 배열 순회(반복)**

**: ‘for’ 문**이나 **‘forEach()’ 메서드** **등**을 사용하여 배열을 순회할 수 있다.

* ‘for’ 문을 사용한 순회

텍스트, 폰트, 스크린샷이(가) 표시된 사진

자동 생성된 설명

* ‘forEach()’ 메서드를 사용한 순회

텍스트, 폰트, 스크린샷이(가) 표시된 사진

자동 생성된 설명

**- 배열 추가, 삭제, 수정**

: JavaScript는 배열을 동적으로 다룰 수 있도록 다양한 메서드를 제공한다,

- **요소 추가**: ‘push()’, ‘unshift()’

* **‘push()’**: 배열의 끝에 요소를 추가한다.
* **‘unshift()’**: 배열의 시작에 요소를 추가한다.

텍스트, 스크린샷, 폰트이(가) 표시된 사진

자동 생성된 설명

**- 요소 삭제**: ‘pop()’, ‘shift()’

* **‘pop()’**: 배열의 마지막 요소를 제거한다.
* **‘shift()’**: 배열의 첫 번째 요소를 제거하고 나머지 요소를 앞으로 이동한다.

텍스트, 폰트, 스크린샷이(가) 표시된 사진

자동 생성된 설명

- **요소 수정**: 인덱스 직접 접근

텍스트, 폰트, 스크린샷이(가) 표시된 사진

자동 생성된 설명

**- 배열 메서드**

: 배열을 효율적으로 조작할 수 있다.

* **concat()**: 배열을 결합하여 새로운 배열을 생성한다.
* **slice()**: 배열의 일부를 추출하여 새로운 배열을 생한다.
* **join()**: 배열 요소를 문자열로 결합한다.
* **indexOf()**, **lastIndexOf()**: 요소의 인덱스를 검색한다.
* **includes()**: 요소가 배열에 포함되어 있는지 확인한다.
* **filter()**: 주어진 조건에 맞는 요소만 추출하여 새로운 배열을 생성한다.
* **map()**: 배열의 각 요소에 함수를 적용하여 새로운 배열을 생성한다.
* **reduce()**, **reduceRight()**: 배열의 요소를 하나의 값으로 축소한다.
* **sort()**: 배열을 정렬한다.
* **reverse()**: 배열의 순서를 반대로 변경한다.

**3. 클래스**

: 클래스는 **‘class’** 키워드를 사용하여 정의된다. 클래스 정의는 **클래스 이름과 중괄호 내부에 멤버 변수와 메서드를 포함**한다.

텍스트, 스크린샷, 폰트이(가) 표시된 사진

자동 생성된 설명

**- 메서드 정의 (Method Definition)**

: 클래스 내부에 메서드를 정의할 수 있다. **메서드는 클래스의 인스턴스에서 호출될 수 있는 함수**이다.

텍스트, 스크린샷, 폰트이(가) 표시된 사진

자동 생성된 설명

**- 생성자 메서드 (Constructor Method)**

: 생성자 메서드는 **클래스가 인스턴스화 될 때 호출**되는 특별한 메서드이다. **constructor** 키워드를 사용하여 정의한다.

텍스트, 스크린샷, 폰트이(가) 표시된 사진

자동 생성된 설명

**- 인스턴스 생성 (Instance Creation)**

: 클래스를 사용하여 인스턴스를 생성할 때는 **‘new’** 키워드를 사용

텍스트, 폰트, 스크린샷, 그래픽이(가) 표시된 사진

자동 생성된 설명

**- 클래스 상속 (Class Inheritance)**

: 클래스는 상속을 통해 **다른 클래스로부터 속성과 메서드를 상속받을 수 있다**. 상속은 **‘extends’** 키워드를 사용하여 정의된다.

텍스트, 스크린샷, 폰트이(가) 표시된 사진

자동 생성된 설명

**- 정적 메서드 (Static Method)**

: 정적 메서드는 클래스의 **인스턴스 없이 호출될 수 있는 메서드**이다. 정적 메서드는 **‘static’** 키워드를 사용하여 정의된다.

텍스트, 폰트, 스크린샷, 디자인이(가) 표시된 사진

자동 생성된 설명

**- 접근 제어자 (Access Modifiers)**

: **‘#’**으로 시작하는 이름을 가진 변수를 사용하여 **'비공개' 변수를 표현**하는 것으로 이러한 변수에는 **클래스 외부에서 직접 접근할 수 없다**.

>> JavaScript 클래스에서는 접근 제어자를 명시적으로 지원하지는 않지만, 널리 사용되는 관례이다.

텍스트, 스크린샷, 폰트, 디자인이(가) 표시된 사진

자동 생성된 설명

**- 클래스의 인스턴스 여부 확인 (Instanceof Operator)**

**: ‘instanceof’** 연산자를 사용하여 **객체가 특정 클래스의 인스턴스인지 확인**할 수 있다.

텍스트, 폰트, 스크린샷이(가) 표시된 사진

자동 생성된 설명

**4. 입출력**

: JavaScript에서 주로 브라우저나 Node.js 환경에서 처리된다. 각 환경마다 조금씩 차이가 있을 수 있다.

**- 브라우저 환경에서의 입력과 출력**

* **출력 (브라우저 콘솔에 출력)**

: 브라우저 콘솔에 출력할 때는 **‘console.log()’** 메서드를 사용한다.

폰트, 그래픽, 그래픽 디자인, 타이포그래피이(가) 표시된 사진

자동 생성된 설명

* **입력 (사용자로부터 데이터 입력 받기)**

: 브라우저에서는 사용자로부터 데이터를 입력 받는 방법이 다소 제한적이다.

주로 **HTML 폼 요소를 사용하여 데이터를 입력** 받고, **JavaScript로 이를 처리**한다.

텍스트, 스크린샷, 폰트이(가) 표시된 사진

자동 생성된 설명

* HTML 폼 요소를 사용하여 사용자로부터 데이터를 입력 받고, JavaScript를 사용하여 이를 처리한다.

**- Node.js 환경에서의 입력과 출력**

: Node.js 환경에서는 **터미널을 통해 데이터를 입력** 받고, **콘솔을 통해 데이터를 출력**한다.

* **출력 (터미널 콘솔에 출력)**

: Node.js 환경에서 콘솔에 출력할 때도 **‘console.log()’** 메서드를 사용한다.



* 입력 (사용자로부터 데이터 입력 받기)

: Node.js에서는 readline 모듈을 사용하여 사용자로부터 데이터를 입력 받는다.

텍스트, 스크린샷, 폰트이(가) 표시된 사진

자동 생성된 설명

* 사용자에게 터미널에서 질문을 하고, 사용자가 답변을 입력하면 그 내용을 콘솔에 출력한다.