❒ 개념 확인

(1) 괄호 안을 채워 넣으시오

① 자바 스크립트 객체는 키와 값으로 구성된 (프로퍼티)들의 집합이다
② 자바 스크립트 객체의 프로퍼티 값이 함수일 경우 일반 함수와 구분하기 위해 (메소드)라고 부른다
③ 자바 스크립트 객체의 프로퍼티 키는 빈 문자열을 포함하는 모든 (문자열)또는 심볼값을 사용한다
④ 프로퍼티 또는 메소드명 앞에 작성하는 (this)는 생성자 함수가 생성할 인스턴스를 의미한다
⑤ 생성자 함수를 사용한 객체 생성시 (new) 키워드를 사용한다
⑥ 프로퍼티 값을 읽기 위해 대괄호 표기법을 사용할 경우 대괄호 내에 들어가는 프로퍼티 키는 반드시 (문자열) 이어야 한다
⑦ 생성자 함수 프로토타입을 사용할 경우 내부에는 (프로퍼티)만 존재한다.
⑧ 클래스에서 인스턴스 프로퍼티는 반드시 (constructor) 에 정의되어야 한다
⑨ 객체 내에 특정 프로퍼티 존재 여부를 확인하려면 (new)연산자를 사용한다
⑩ (클래스)로 객체를 생성할 경우 반드시 new 연산자가 있어야 한다

(2) 리터럴 표기법으로 book 객체를 생성하는 문장을 선택하시오

① let book={title:'js', price:30000}
② let book={title='js', price=3000}
③ let book={title='js'; price=3000}
④ let book=[title:'js', price:30000]

(3) 2번에서 생성된 book 객체에 접근하는 방법을 모두 선택하시오

① book[title]
② book.title
③ book->title
④ book['title']

(4) 생성자 함수를 사용하여 객체를 정의하는 문장을 선택하시오

① let Book = function(title, price){
this.title=title; this.price=price;
}

② function Book(title, price){
this.title=title; this.price=price;
}

③ let Book = (title, price) => {
this.title=title; this.price=price;
}

④ function Book(title, price){
this.title=title; this.price=price;

```
}
Book.prototype.total=title;
```

(5) 4번의 생성자 함수를 사용하여 객체를 생성하는 문장을 제시하시오. 단, 매개값은 임의로 정할 것

==풀이==

```
let Book = new Book('Function',1000);
```

(6) 생성자 함수와 클래스로 객체를 생성하는 경우 차이점은 무엇인가?

==풀이==

생성자 함수는 객체를 생성시 new 키워드를 사용하여 연산자가 없을시에 일반함수로 동작하는 것을 방지한다.

클래스는 객체 생성시 반드시 new를 사용하여 생략시에 오류가 나는 것을 방지한다.

(7) 질문에 답하시오

① Object 생성자 함수를 사용하여 빈 객체를 생성하는 문장을 제시하시오. 단 객체명은 obj1

==풀이==

```
let obj1 = new Object();
```

② 1에서 생성된 객체에 다음과 같은 프로퍼티를 추가하고 임의의 값으로 초기화 한다.

time(자료타입 number), message(자료타입 string)

==풀이==

```
obj1.number = 'time';
obj1.string = 'message';
```

③ console.log(age in obj1); 실행 결과를 제시하시오.

==풀이==

(8) 객체 생성과 메소드 호출을 참고하여 Book class를 작성하시오

```
const book = new Book('흑산', '김훈');
book.bwrite();  //객체 프로퍼티 값을 웹브라우저로 출력
```

==풀이==

```
class Book {
    constructor(title, author) {
        this.title = title;
        this.author = author;
    }
    bwrite() {
        Document.write(`title : ${this.title}₩n author : ${this.author}`);
}
```

(9) 8에서 생성된 객체의 모든 프로퍼티를 순회하면서 출력하는 문장을 작성하시오. 힌트)for~in

==풀이==

```
For(const property in book) {
```

```
        Document.write(`${property} : ${book[property]}`);
    }
```

❑ 개념 활용 응용 프로그래밍

(1) 다음과 같은 속성과 메소드로 구성되는 객체를 제시된 방법으로 생성하고 결과를 확인하세요

- 속성 : 가수 이름, 곡명, 재생시간
- 메소드 : play(cnt) – cnt 횟수만큼 반복 재생
- 객체 생성 방법
  - 객체 리터럴

```
가수 : 이소라, 제목: 바람이 분다, 재생시간 : 3.5 => 1 번째 재생
가수 : 이소라, 제목: 바람이 분다, 재생시간 : 7 => 2 번째 재생
가수 : 이소라, 제목: 바람이 분다, 재생시간 : 10.5 => 3 번째 재생
가수 : 이소라, 제목: 바람이 분다, 재생시간 : 14 => 4 번째 재생
가수 : 이소라, 제목: 바람이 분다, 재생시간 : 17.5 => 5 번째 재생
```

[소스]

```
<script>
    let song= {
        artist:'이소라',
        title:'바람이분다',
        time:3.5,
        play(cnt) {
            for(let i=1; i<=cnt; i++) {
                document.write(`가수 : ${this.artist}, 제목 : ${this.title}, 재생시간 :
${this.time*i} => ${i} 번째재생 <br>`);
            }
        }
    }
    song.play(5);
</script>
```

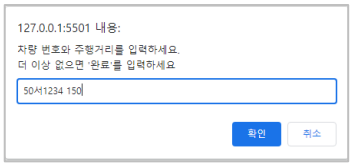
[실행 결과]

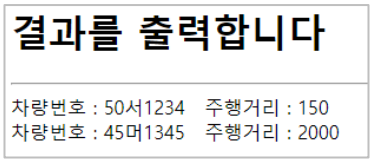
```
가수 : 이소라, 제목 : 바람이분다, 재생시간 : 3.5 => 1 번째재생
가수 : 이소라, 제목 : 바람이분다, 재생시간 : 7 => 2 번째재생
가수 : 이소라, 제목 : 바람이분다, 재생시간 : 10.5 => 3 번째재생
가수 : 이소라, 제목 : 바람이분다, 재생시간 : 14 => 4 번째재생
가수 : 이소라, 제목 : 바람이분다, 재생시간 : 17.5 => 5 번째재생
```

(2) 다음과 같은 속성과 메소드로 구성되는 객체를 생성하는 프로그램을 생성자 함수 프로토타입을 사용하여 구현한 후 제시된 결과처럼 동작할 수 있도록 프로그램을 작성하시오

– 속성 : 차량번호, 주행거리 //입력(사이 공백) split

– 메소드 : 주행거리를 dist 만큼 증가시키는 addMileage(dist) 메소드, 반환값 없음
차량번호와 주행거리를 문자열로 반환하는 toString()





힌트1) 데이터 입력은 prompt()함수를 사용하고 차량번호와 주행거리는 공백으로 구분한다
힌트2) 입력된 데이터는 split() 함수를 사용하여 구분한 후 객체 초기화에 사용한다
힌트3) 초기화된 객체는 Array에 저장한다.

[소스]

```
    <script>
        let dist=0;
        function Car(carNum, Range) {
                this.carNum = carNum;
                this.Range = Range;
        }

        Car.prototype.addMileage = function(dist) {
            this.Range = parseInt(this.Range) + parseInt(dist);
        }

        Car.prototype.ToString = function() {
            return `차량번호 : ${this.carNum}, 주행거리 : ${this.Range}<br>`;
        }

        let ar = [];
        let Array = [];
        while(1) {
            let input = prompt(`차량번호와 주행거리를 입력하세요 더이상 없으면 '완료'를
입력하세요`);
            if(input == '완료') {
                break;
            }
            ar = input.split(' ');
            Array.push(new Car(ar[0],ar[1]));
        }

        document.write(`<h2>결과를 출력합니다</h2><hr>`);
        for(let i=0; i<Array.length; i++) {
            document.write(`${Array[i].ToString()}`);
        }
        document.write('<hr><h2>주행거리를 4 만큼 증가</h2>');
        for(let i=0; i<Array.length; i++) {
            Array[i].addMileage(4);
            document.write(`${Array[i].ToString()}`);
        }
    </script>
```

[실행 결과]

결과를 출력합니다

차량번호 : 50서1234, 주행거리 : 150
차량번호 : 45머1345, 주행거리 : 2000

주행거리를 4만큼 증가

차량번호 : 50서1234, 주행거리 : 154
차량번호 : 45머1345, 주행거리 : 2004

(3) 다음과 같은 속성과 메소드로 구성되는 클래스ㅊ Account를 만들고 제시된 결과처럼 실행되는 프로그램을 작성하세요.

- 속성 : 예금주, 잔액
- 메소드

- 매개변수로 받은 값 만큼 잔액을 증가하는 deposit(매개변수) 메소드, 반환값 없음
- 매개변수로 받은 값 만큼 잔액을 감소하는 withdraw(매개변수) 메소드, 반환값 없으며 잔액이 적으면 "잔액부족" 출력
- 예금주와 잔액을 출력하는 display() 메소드, 매개변수 없음

현재 상태 입니다
예금주 : 스크립트
현재 잔액 : 50000

50000 예금 후 상태 입니다
예금주 : 스크립트
현재 잔액 : 100000

1000000을 인출하려고 합니다
잔액 부족 : 900000

[소스]

```
<script>
    class Account {
        constructor(name, money) {
            this.name = name;
            this.money = money;
        }

        deposit(plus) {
            this.money = parseInt(this.money) + parseInt(plus);
            document.write(`${plus}원 예금 후 상태입니다.<br> 예금주 : ${this.name} <br>현재
잔액 : ${this.money}<br><br>`);
        }

        withdraw(minus) {
            if(this.money < minus) {
                document.write(`${minus}를 인출하려합니다.<br>잔액부족 :
${this.money}<br><br>`);
            }
            else {
                this.money = parseInt(this.money) - parseInt(minus);
                document.write(`${minus}를 인출하려합니다.<br>예금주 : ${this.name}<br>현재
잔액 : ${this.money}<br><br>`);
            }
        }
```

```
        display() {
            return `현재상태입니다.<br> 예금주 : ${this.name}<br> 현재 잔액 :
${this.money}<br><br>`;
        }
    }

    const a = new Account('박신우', 20000);
    document.write(a.display());
    a.withdraw(50000);
    a.deposit(10000);
    a.withdraw(20000);
</script>
```

[실행 결과]

현재상태입니다.
예금주 : 박신우
현재 잔액 : 20000

50000를 인출하려합니다.
잔액부족 : 20000

10000원 예금 후 상태입니다.
예금주 : 박신우
현재 잔액 : 30000

20000를 인출하려합니다.
예금주 : 박신우
현재 잔액 : 10000

(4) 다음과 같은 속성과 동작을 갖는 대상을 자바스크립트 객체로 구현하고 테스트 하시오. 단, 클래스로 구현하고 테스트 결과는 console.log()를 사용하여 처리하시오.

```
백신종류 : 화이자,  연락처 : 010-2312-8723 접종현황: 미 접종
백신종류 : 화이자,  연락처 : 010-2312-8723 접종현황: 추가 1회
연락처 변경 후 출력
백신종류 : 화이자,  연락처 : 010-6543-7968 접종현황: 추가 1회
```

속성	값
백신	모더나, 화이자
접종 횟수	0
연락처	010-2193-5234
동작	내용
isFinished()	접종 횟수가 2이면 '접종 완료', 1이면 '추가 1회', 0이면 '미 접종' 반환
addShot()	접종 회수를 +1 증가, 만약 접종 회수가 2이면 증가 없음
changeTel(value)	연락처를 value값으로 변경

[소스]

```
<script>
    class medi {
        constructor(label, count, phone) {
            this.label = label;
```

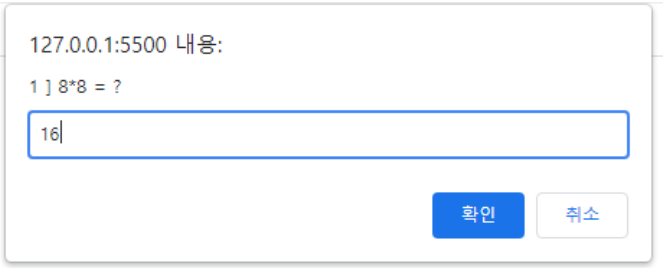
```
            this.count = count;
            this.phone = phone;
        }

        isFinished() {
            if(this.count==2) {
                return `접종 완료`;
            }
            else if(this.count==1) {
                return `추가 1 회`;
            }
            else {
                return `미 접종`;
            }
        }

        addShot() {
            if(this.count>=2) {
                return 0;
            }
            else {
                this.count++;
            }
        }

        changeTel(value) {
            this.phone = value;
            console.log('연락처 변경 후 출력');
            console.log(this.toString());
        }

        toString() {
            return `백신종류 : ${this.label}, 연락처 : ${this.phone}, 접종현황 :
${this.isFinished()}`;
        }
    }

    const m1 = new medi('화이자', 0, '010-2312-4444');
    const m2 = new medi('모더나', 2, '010-3231-5678');

    console.log(m1.toString());
    m1.addShot();
    console.log(m1.toString());
    m1.changeTel('010-2222-3333');
</script>
```

[실행 결과]

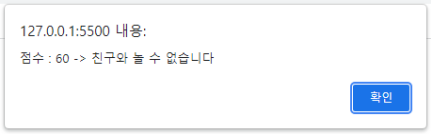
```
백신종류 : 화이자, 연락처 : 010-2312-4444, 접종현황 : 미 접종
백신종류 : 화이자, 연락처 : 010-2312-4444, 접종현황 : 추가 1회
연락처 변경 후 출력
백신종류 : 화이자, 연락처 : 010-2222-3333, 접종현황 : 추가 1회
```

(5) 2학년 조카의 구구단 학습 도우미 프로그램을 제시된 결과처럼 실행되도록 프로그램하세요.

- 1~9사이에 생성된 난수를 입력창에 제시된 결과처럼 출력하고, 답을 입력 받는다(10번 반복)



- 맞춘 회수에 10을 곱하여 점수를 계산한다.
- 계산된 점수가 90이상이면 '친구와 놀아도 됩니다', 80 이상이면 '한번 더 연습하세요', 70 이상이면 '두번 더 연습하세요', 70미만이면 '친구와 놀 수 없습니다'를 알림창으로 출력



- Gugudan 클래스를 정의하여 사용하도록 한다.

[소스]

```
<script>
    class Gugudan {
        constructor(num1, num2) {
            this.num1 = num1;
            this.num2 = num2;
        }

        problem() {
            return `${this.num1} * ${this.num2}`;
        }

        grading(result) {
            if(result>90) {
                return '친구와 놀아도 됩니다.';
            }
            else if(result>80) {
                return '한번 더 연습하세요.';
            }
            else if(result>70) {
                return '두번 더 연습하세요.';
            }
            else {
                return '친구와 놀 수 없습니다.'
            }
        }
    }3

    let count=0;
    const Gugu = new Gugudan();
    for(let i=1; i<11; i++) {
        const Gugu = new Gugudan(Math.floor(Math.random()*9+1),
Math.floor(Math.random()*9+1));
        let input = prompt(`${i}) ${Gugu.problem()} = ?`);32
        let answer = Gugu.num1 * Gugu.num2;
        if(input==answer) {
            count++;
```

```
        }
    }

    let output = alert(`점수 : ${count*10} -> ${Gugu.grading(count*10)}`);
</script>
```

[실행 결과]