

□ 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 내부)에 정의되어야 한다
- ⑨ 객체 내에 특정 프로퍼티 존재 여부를 확인하려면 (in)연산자를 사용한다
- ⑩ (클래스)로 객체를 생성할 경우 반드시 new 연산자가 있어야 한다

(2) 리터럴 표기법으로 book 객체를 생성하는 문장을 선택하시오

답:1번

- 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

답: 2번,4번

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

(4) 생성자 함수를 사용하여 객체를 정의하는 문장을 선택하시오

답:2번

- 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('홍길동전',10000);
```

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

==풀이==

1. 클래스로 생성할 경우 new 를 붙이지 않으면 오류가 생긴다.

생성자 함수는 new 를 생략하면 일반함수로 동작, new 를 붙이면 객체 (...더쓰기)

(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.time=10;
```

```
obj1.message='시간';
```

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

==풀이==

```
console.log(age in obj1); -> age is not defined : 답
```

```
console.log('age' in obj1); -> false
```

(8) 객체 생성과 메소드 호출을 참고하여 Book class를 작성하시오

```
const book = new Book('홍산', '김훈');
```

```
book.bwrite(); //객체 프로퍼티 값을 웹브라우저로 출력
```

==풀이==

```
class Book{
```

```

    constructor(name,author){
    this.name=name;
    this.author=author;
    }
    bwrite(){
    document.write( this.name+ ' , '+ this.author);
    }
}

```

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

```

for(const property in book){
    console.log(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

```

시간은 임의로 설정
 웹브라우저 출력

[소스]

```

<script>
    let music={
        name:'이소라',
        music_name:'바람이 분다',
        playtime:3.5,
        play: function(cnt){
            for(var i=1 ; i<=cnt ; i++){
                document.write('가수 : '+this.name+', 제목 : '+this.music_name+', 재생시간 : '+this.playtime*i+' => '+i+' 번째 재생<br>');
            }
        }
    }
    music.play(5);
</script>

```

[실행 결과]

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

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

127.0.0.1:5501 내용:
차량 번호와 주행거리를 입력하세요.
더 이상 없으면 '완료'를 입력하세요

50서1234 150

확인 취소

결과를 출력합니다

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

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.

차량번호 주행거리 사이에만 공백

[소스]

```
<script>

function Car(num,km){
    this.num=num;
    this.km=km;
}

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

Car.prototype.toString=function(){
    return "차량번호 : "+this.num+"    주행거리 : "+this.km;
}

var i=0;
let car=[];
//let array=new Array();
while(true){
```

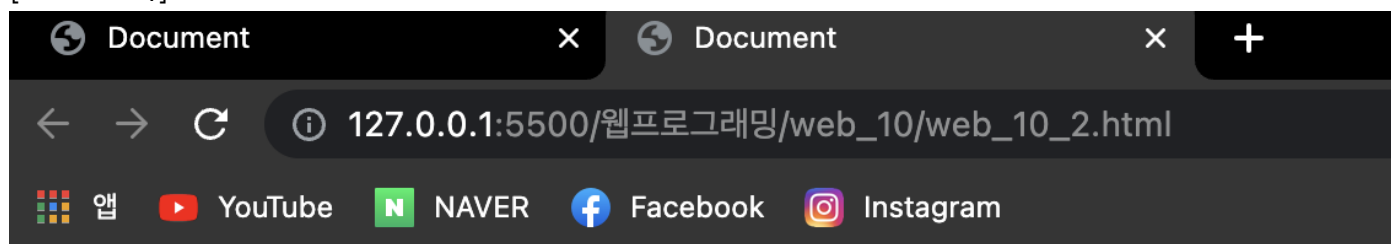
```

var str=prompt("차량 번호와 주행거리를 입력해주세요.\n 더 이상 없으면 '완료'를 입력하세요.");
if(str=="완료"){
//return array;

    break;
}
var arr=str.split(" ");
car[i]=new Car(arr[0],parseInt(arr[1]));
i=i+1;
}
document.write("<h1>결과를 출력합니다.</h1> <br>");
document.write("<hr><br>");
for(var i=0 ; i<car.length ; i++){
//for (item of array)
//    item.toString() 출력
    car[i].addMileage(30);
    document.write(car[i].toString()+"<br>");
}
</script>

```

[실행 결과]



결과를 출력합니다.

차량번호 : 30가3040 주행거리 : 60
 차량번호 : 39가4938 주행거리 : 50

(3) 다음과 같은 속성과 메소드로 구성되는 클래스 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]

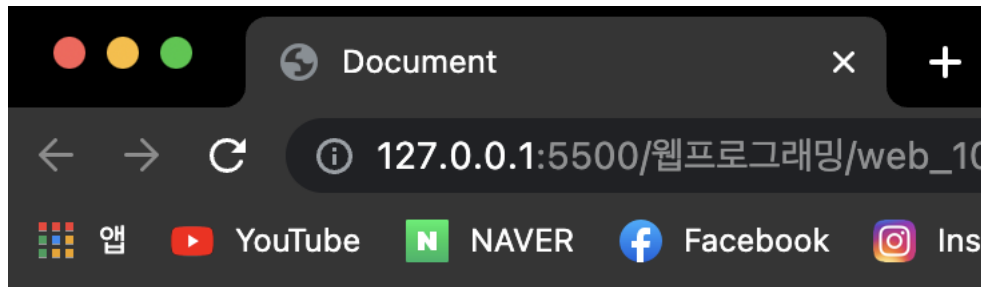
```
<script>
```

```
class Account{
    constructor(account_holder,balance){
        this.account_holder=account_holder;
        this.balance=balance;
    }
    deposit(count){
        document.write(count+" 예금후 상태 입니다.<br>");
        this.balance=this.balance+count;
    }
    withdraw(count){
        document.write(count+"을 인출하려고 합니다.<br>");
        if((this.balance-count)<0){
            document.write("잔액부족 : "+(count-this.balance));
        }
        else{
            this.balance=this.balance-count;
        }
    }
    display(){
        document.write("예금주 : "+this.account_holder+"<br>");
        document.write("현재 잔액 : "+this.balance+"<br><br>");
    }
}

const account=new Account('스크립트',50000);
document.write("현재 상태 입니다.<br>");
account.display();
account.deposit(50000);
```

```
account.display();
account.withdraw(1000000);
</script>
```

[실행 결과]



현재 상태 입니다.
예금주 : 스크립트
현재 잔액 : 50000

50000 예금후 상태 입니다.
예금주 : 스크립트
현재 잔액 : 100000

1000000을 인출하려고 합니다.
잔액부족 : 900000

- (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

애는 검사->콘솔창에서 결과 확인

[소스]

```
<script>

class Corona{
  constructor(vaccine,numOfInocul,phoneNum){
    this.vaccine=vaccine;
    this.numOfInocul=parseInt(numOfInocul);
    this.phoneNum=phoneNum;
  }
  isFinished(){
    if(this.numOfInocul==2){
      return '접종 완료';
    }
    else if(this.numOfInocul==1){
      return '추가 1 회';
    }
    else{
      return '미접종';
    }
  }
  addShot(){
    if(this.numOfInocul!=2){
      this.numOfInocul=this.numOfInocul+1;
    }
  }
  changeTel(value){
    this.phoneNum=value;
  }
  ToString(){
    return "백신종류 : "+this.vaccine+", 연락처 : "+this.phoneNum+", 접종 현황 : "+this.isFinished();
  }
}

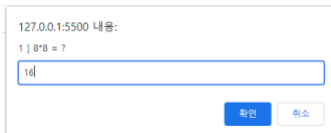
let corona=new Corona('화이자',0,'010-3022-3237');
console.log(corona.ToString());
corona.addShot();
console.log(corona.ToString());
console.log("연락처 변경 후 출력");
corona.changeTel('010-3222-3333');
console.log(corona.ToString());
```


[실행 결과]

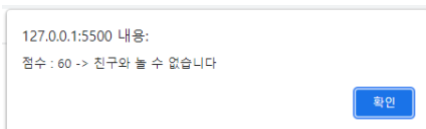
백신종류 : 화이자, 연락처 : 010-3022-3237, 접종 현황 : 미접종
백신종류 : 화이자, 연락처 : 010-3022-3237, 접종 현황 : 추가 1회
연락처 변경 후 출력
백신종류 : 화이자, 연락처 : 010-3222-3333, 접종 현황 : 추가 1회

(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.

구구단 랜덤 생성

[소스]

```
class Gugudan {
    constructor(total) {
        this.total=total*10;
    }

    Total() {

        if (this.total >= 90)
            return '점수 : ' + this.total + ' -> 친구와 놀아도 됩니다.';
        else if (this.total >= 80)
            return '점수 : ' + this.total + ' -> 한번 더 연습하세요.';
        else if (this.total >= 70)
            return '점수 : ' + this.total + ' -> 두번 더 연습하세요.';
        else
            return '점수 : ' + this.total + ' -> 친구와 놀 수 없습니다.';
    }
}
```

```

var count=0;
for (var i = 0; i < 10; i++) {
    randomNum1 = Math.floor(Math.random() * 9 + 1);
    randomNum2 = Math.floor(Math.random() * 9 + 1);

    var answer = prompt(i+1 + "]" + randomNum1 + "*" + randomNum2 + " = ? ");
    // ==: 내용만 비교 , ===: 타입까지 비교
    if (Number(answer) == randomNum1 * randomNum2)
        count = count + 1;
}
const gugudan = new Gugudan(count);
alert(gugudan.Total());

```

[실행 결과]

127.0.0.1:5500 내용:

1]1*5 = ?

취소

확인

127.0.0.1:5500 내용:

점수 : 100 -> 친구와 놀아도 됩니다.

확인