



# 생성자 함수



- 생성자 함수의 개념을 안다.
  - 생성자 함수에서 활용되는 변수 설정방법을 안다.
  - 생성자 함수에서 쓰이는 기본구조를 사용하고 적절하게 활용할 수 있다.
    - 변수활용 : **this**의 이용방법을 안다.
    - 메서드:**action** 활용 방법을 이해한다.
-



생각해봅시다! :

---



# 자바스크립트 현재 어디즈음:

- 개요
  - 기본문법
  - 조건반복문
  - 객체
  - 함수
  - **생성자 함수**
  - 브라우저 객체 모델
  - 문서 객체 모델
  - 이벤트
-



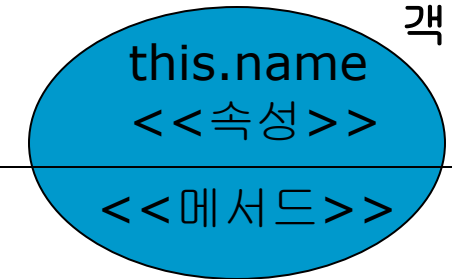
## 생성자? :

- 생성자는 객체지향 프로그램에서 객체를 생성하는 단위 메서드를 말한다.
  - **new**를 함께 사용해서 호출한다.
  - **new** 생성자명();
- 자바스크립트에서 생성자 함수 선언과 호출
  - **function** 생성자명(){ 속성, 메서드}
  - 함수명과 동일한 선언규칙을 사용하나, 객체로 쓰이는 차이점 있다.
  - **ex) function Student(){ 속성, 메서드}**
  - **var** 참조변수 = **new** 생성자명();
  - **ex) var student01 = new Student();**



## 생성자 함수의 속성정의 :

- 모든 객체는 속성값을 가지고 활용할 수 있다. 속성은 모든 객체에 있다.
  - function 생성자함수(){
    - **this**. 속성=초기값;
  - }
  - 속성명을 지정할 때, **this** 는 해당객체의 전역변수임을 나타낸다. 현재 객체(this)의 속성값 name.
  - ex)
  - function Student(){
  - **this**.name="이정희";
  - }





## 함수 객체 호출해서 활용 :

### ■ 기본 형식

- `var 참조변수 = new 생성자함수();`
- cf) 참조변수 : 객체를 호출하는 변수를 참조변수라고 한다..
- 생성자함수에 포함된 변수를 활용하는 방법.
  - `참조변수.속성` : 객체 안에 포함된 속성..
- ex) `var student01 = new Student();`
- `student01.name;`
- `console.log( "이름:" + student01.name );`
- `Student` 안에 `name` 속성이 있기에 호출 가능



- 객체 함수
  - 가전제품이라는 객체를 만들고, 속성값으로 종류와 가격을 선언한다.
- 객체 호출
  - 출력 : 가장 좋아하는 가전제품으로 @@@ 이며, 가격은 @@@입니다. `alert, console.log`





## 생성자를 통한 초기값 설정 :

- `var student1 = new Student();`
- 객체마다 다른 속성의 값을 초기에 setting 해야 한 필요성이 있다..
- 객체는 구조(속성,메서드)를 만들어 놓고, 데이터를 다르게 처리해야지 의미가 있다.
- 외부에 있는 데이터를 초기값으로 입력이 가능한 구조로 만들어야 한다.
  - 생성의 입력값을 정의해 놓고, 정의한 입력값이 전역변수에 할당하면, 데이터가 변경하는 객체가 만들어 질 수 있다.




## 변경하는 속성값 정의 :


### ■ 형식

- function 생성자명(입력값1, 입력값2, ..... )
- function Student( name, kor, eng, math ){
  - this.name= name;
  - this가 붙어 있는 전역변수의 name이고, this붙지 않는 것은 외부에서 받은 지역변수 name을 말한다.
- }

### ■ 객체 호출과 활용. 입력값1 = 데이터1;

- var 참조변수 = new 생성자명( 데이터1, 데이터2);
- ex) var stu01 = new Student("이신영",70,80,90);
- ex) var stu02 = new Student("홍길동",90,70,80);
- name = "이신영";
- this.name=name;으로 인해서 전역변수에 " 이신영"
- stu01.name; stu02.name; //여러객체를 활용이가능

- 
- 함수 객체
    - CoffeeShop
  - 입력값 : 생성자의 입력값..
    - 주문커피 종류, 가격, 갯수
  - 함수 객체 내용 변수
    - 종류, 가격, 갯수, 총계(가격\*갯수)
  - 호출처리. : 주문한 커피는 @@@, 가격 @@  
갯수 @@ 총 @@@ 입니다.
    - coffeeOrder01
    - coffeeOrder02
-



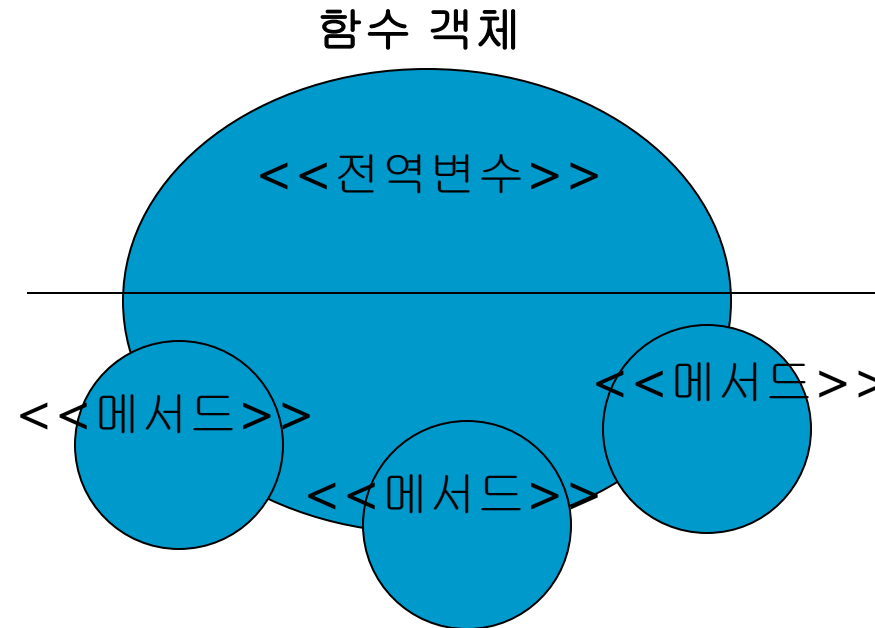
```
function CoffeeShop(kind, price, cnt){  
    this.kind=kind;  
    this.price=price;  
    this.cnt = cnt;  
    this.tot=price*cnt;  
}  
  
var coffeeOrder01 = new CoffeeShop("아메리카노",3000,2);  
var coffeeOrder02 = new CoffeeShop("카푸치노",4000,3);  
console.log("주문한 커피는 "+coffeeOrder01.kind+  
", 가격"+coffeeOrder01.price+",갯수"+coffeeOrder01.cnt  
+",총"+coffeeOrder01.tot);  
console.log("주문한 커피는 "+coffeeOrder02.kind+  
", 가격"+coffeeOrder02.price+",갯수"+coffeeOrder02.cnt  
+",총"+coffeeOrder02.tot);
```

---



# 메서드 정의하여 활용하기 :

- 객체는 고정된 변수와 이를 활용하는 메서드로 크게 나누어 진다.
- 메서드 기능 : 함수가 객체에 소속되어 있는 것
  - 입력값
  - 프로세스 처리(연산, 조건), 공통 출력
  - return값



```
function Student(name){  
    this.name=name;  
    this.show = function( ..... ){  
        console.log("이름"+ this.name);  
        return XXXX;  
    }; --메서드 정의  
}
```



## 객체에서 메서드 활용 :

- function Person(name){
- this.name=name;   // 데이터 할당
- this.show = function( **fname** ){   // 함수 할당
- console.log("이름"+ this.name);
- console.log("친구이름"+ **fname**);
- return **XXXX**;
- }; --메서드 정의
- }
- 호출... : var 객체참조 = new 생성자();
- 객체참조.**메서드 ( ..... )**;
- ex) var stu01 = new Student("홍길동");
- stu01 **.show( "이진영"**);   //stu01.show;(X)



## 객체와 함수 활용하기(확인예제) :

- 변수들 출력하기..
  - 함수객체 : ChildFriends
  - 생성 입력값 : 어린 시절 친구들 입력(3명)
  - 메서드 printAll 정의 하고..
    - console.log 통해서
    - 추억의 친구들 @@@, @@@, @@@
- 계산기 만들기(숙제)
  - 함수객체 : Calculator
  - 입력값 : num01, num02 3, 4
  - 메서드 : plus(), minus(), multi(), div(); 를 통해서 각 기능별로 출력처리 ex) @@@ + @@@ = @@@ 3 + 4 = 7 return 값은 결과값 (7)
  - ex) var result = cal01.plus();



## 함수객체 이해 예제 :

- 자판기!!! (설계)
    - 생성자 입력 : 판매물건
    - 전역변수 : 현재 총 금액
    - 메서드 : insertMoney(입력되는 money)
      - 현재 입력된 총금액 : @@@
    - 메서드 : clickButton()
      - 금액이 작으면 금액 부족하다.
      - return 음료
    - 메서드 : restMoney()
      - return 잔액
-





# 함수객체 이해(소스) 예제 :

## ■ 자판기!!! (설계)

```
- function SellerBox(product, price){
  ■ this.product = product; // 물건명
  ■ this.price = price; // 물건가격
  ■ this.insTotMn=0;
  ■ this.insertMoney=function(inMoney){
    - this.insTotMn+=inMoney; // 누적처리..
    - console.log("###"+this.product+"자판기###");
    - console.log("현재 입력된 금액"+ this.insTotMn);
  ■ }
  ■ this.clickButton=function(){
    - var retVal=""; // 금액부족하면 "" return
    - if( this.insTotMn>= this.price){
      ■ this.insTotMn -= this.price; //잔액을 계산..
      ■ // this.insTotMn = this.insTotMn- this.price;
      ■ console.log("구매 후, 남은 잔액은 "+ this.insTotMn);
      ■ retVal="커피가 나왔습니다";
    - }else{
      ■ console.log("잔액이 부족합니다");
    - }
    - return retVal;
  ■ }
  ■ this.restMoneyBtn=function(){
    - return this.insTotMn;
  ■ }
- }
- var sellingBox01 = new SellerBox("커피", 1000);
- sellingBox01. insertMoney(500);
- sellingBox01. insertMoney(500);
- sellingBox01. insertMoney(500);
- var resultMsg=sellingBox01.clickButton(); //return값을 처리..화면출력(X)
- console.log("자판기 박스에서 나온 결과:"+resultMsg);
- var restMoney=sellingBox01.restMoneyBtn();
- console.log("나머지 돈은:"+ restMoney );
```



- 객체
  - DailyMoneyPocket (지갑) – 하루동안 입출금
  - 초기입력값 : 날짜, 소유자이름, 초기금액
  - 메서드 :
    - spendMoney(지출내역, 지출금액)
    - @@@에 @@@ 지출
    - 지출금액이 현재 잔액 크면 잔액부족
      - return 잔액..

```
function DailyMoneyPocket(date, owner, initMn){
    this.date =date;
    this.owner = owner;
    this.totMony=initMn; // 현잔액을 초기에 입력 처리..
    this.spendMoney=function(spendCont, spendMn){
        console.log("###"+this.owner+"님의 "+this.date+"일 지갑 ###");
        if( this.totMony>=spendMn){
            this.totMony-=spendMn;
            console.log(spendCont+"에 "+spendMn+"원 지출");
        }else{
            console.log("잔액 부족");
        }
        return this.totMony;
    }
}

var person01 = new DailyMoneyPocket("4/4", "홍길동",20000);
person01.spendMoney("버스비용",2400);
person01.spendMoney("지하철",1200);
person01.spendMoney("점심식사 ",6000);
person01.spendMoney("저녁식사 ",6000);
person01.spendMoney("버스",2400);
var rest=person01.spendMoney("지하철" ,1200);
console.log("남은 잔액:"+rest);
```

## ■ 기본 객체 선언..

- function Emp(empno, ename,sal){
  - this.empno=empno;
  - this.ename=ename;
  - this.sal=sal;
- }

## ■ 객체 배열..

- var empList=[];
- empList[0] = new Emp(1000,"홍길동",3000);
- empList[1] = new Emp(1001,"신길동",3500);
- for(var idx=0;idx< empList.length;idx++){
  - console.log(empList[idx].empno+":")
  - empList[idx].ename+":")
  - empList[idx].sal
- }



## 배열객체의 메서드 :

```
function Student(name){  
  this.name=name;  
}
```

- 배열 객체에서 활용되는 키워드.
  - var list =[];
  - list.push( new Student("홍길동") );
  - list[0]= new Student("홍길동");
    - 객체를 배열에 등록처리..
  - list.length : 배열의 크기.
    - for( var idx=0;idx< list.length ;idx++)
      - list[ idx ].name : 홍길동 데이터가 있음
  - index

0	1	2	3	4
---	---	---	---	---

    - list[0] → new Student("홍길동")



## 배열처리 for2:

- for( **idx** in 배열객체){
  - 배열객체[ **idx** ].속성
- }
- var books =[];
- books.push( new Book("쉬운자바") );
- books.push( new Book("자바스크립트") );
- for( **idx** in books ){
  - books[ **idx** ].show();
- }

```
function Book(name){  
    this.name=name;  
    this.show=function(){  
        console.log(this.name);  
    };  
}
```



## 정리 및 과제 :

- 생성자란 무엇인지 알는가?
  - 객체의 구성요소 **2**가지가 무엇인지 아는가?
  - 메서드의 주요 기능 요소 **3**가지를 나열해 보자.
  - 다양한 객체를 프로그램으로 작성하여 활용한다. **ex) SuperMarket, Bank**
  - 객체 배열을 활용해 보자..
    - 성적관리..
-



**quiz:**

---





**다음 장은 입니다 !**

---