**ch 01. Class**

1. 실습 준비

* vs Code 다운로드 후 설치(https://code.visualstudio.com)
* nodejs 다운로드 후 설치(<https://nodejs.org/ko>)
* node –v 로 버전 확인
* npm –v로 버전 확인

1. 객체의 얕은 복사 vs 깊은 복사

let memberArray = ['egoing', 'graphittie', 'leezhce'];

var copyArray = [...memberArray]; // 배열 복사(깊은 복사)

var copyArray = Array.from(memberArray);

copyArray[0] = '홍길동'; copyArray[1]='복사함'; copyArray[2]='카피함';

console.group('memberArray for-loop');

for(var i=0 ; i<memberArray.length ; i++){

    console.log(i+'번째 ☞ ' , memberArray[i]);

}

console.groupEnd('memberArray loop');

console.group('copyArray forEach');

copyArray.forEach((data, idx)=>{

    console.log(idx, '번째 : ', data);

});

console.groupEnd('copyArray forEach');

var memberObject = {

    manager:'egoing',

    developer : 'graphittie',

    designer : 'leezhce'

};

// var copyObject = Object.assign({}, memberObject); // 복사

//var copyObject = Object.assign({id:1}, memberObject);

var copyObject = { ...memberObject};

copyObject['manager'] = '홍길동';

console.group('memberObject for-loop');

for(var key in memberObject){

    console.log(key, ' : ', memberObject[key]);

}

console.groupEnd('memberObject for-loop');

console.group('copyObject loop');

for(var key in copyObject){

    console.log(key, ' : ', copyObject[key]);

}

console.groupEnd('copyObject loop');

1. 단순 객체 생성

let kim = {

    name : 'Kim GilDong',

    first : 10,

    second : 20,

    third : 30,

    sum : function(){

        return this.first + this.second+this.third

    },

};

let lee = {

    name : 'Kim GilDong',

    first : 10,

    second : 20,

    thirth:10,

    sum : function(){

        return this.first + this.second + this.thirth;

    },

};

console.log('kim.sum() :', kim.sum());

console.log('lee.sum() :', lee.sum());

var thatDay = new Date('2022-01-01');

console.log('년도 :', thatDay.getFullYear());

console.log('월 :', thatDay.getMonth()+1);

console.log(thatDay);

console.log('Date : ', Date);

1. 생성자 함수를 이용한 객체 생성

function Person(name, first, second, third){

    this.name = name;

    this.first = first;

    this.second = second;

    this.third = third;

}

Person.prototype.sum = function(){

    return this.first + this.second + this.third;

};

console.log('Person() :', Person());

console.log('new Person() :', new Person());

var kim = new Person('홍길동', 10,10,20);

var lee = new Person('이승엽', 90,90,99);

console.group('생성자이용');

console.log(kim.sum());

console.log(lee.sum());

console.groupEnd('생성자이용');

1. class를 이용한 객체 생성

// class는 ECMA scipt 6 이상을 지원하는 브라우저에서 가능

// https://caniuse.com/에서 Can I use [ES6 classes]입력해서 확인

// https://babeljs.io javaScript 컴파일러

class Person {

    constructor(name, first, second){

        console.log('생성자 호출됨');

        this.name = name;

        this.first = first;

        this.second = second;

    }

    sum(){

        return this.first + this.second;

    }

}

var kim = new Person('kim', 10, 20);

kim.sum = function(){

    return 'kim의 합 : ' + (this.first + this.second)

}

var lee = new Person('lee', 100, 300);

console.log("kim :", kim);

console.log('kim sum : ' + kim.sum());

console.log('lee sum : ' + lee.sum());

1. class를 이용한 상속

class Person{

    constructor(name, first, second){

        this.name = name;

        this.first = first;

        this.second = second;

    }

    sum(){

        return this.first+this.second;

    }

}

var kim = new Person('kim', 20, 30);

var lee = new Person('lee', 100, 20);

console.log('kim sum :', kim.sum());

console.log('lee sum :', lee.sum());

class PersonPlus extends Person{

    constructor(name, first, second, third){

        super(name, first, second); // 부모클래스의 생성자함수

        this.third = third;

    }

    sum(){

        return super.sum()+this.third; // 부모클래스의 sum()함수 + third

    }

    avg(){

        return this.sum()/3;

    }

}

var pp = new PersonPlus('홍', 100,200,300);

console.group('pp');

console.log('pp : ', pp);

console.log('pp의 합계 :', pp.sum());

console.log('pp의 평균 :', pp.avg());

console.groupEnd('pp');

1. 객체의 상속

* \_\_proto\_\_ 이용

var superObj = {superVal : 'super'};

var subObj = {subVal : 'sub'};

// 객체끼리도 상속관계를 맺을 수 있다.

subObj.\_\_proto\_\_ = superObj; // 표준으로 인정받지는 못하나 브라우저에서는 제공

console.log(subObj);

console.log('subObj.subVal :', subObj.subVal);

console.log('subObj.superVal :', subObj.superVal);

subObj.superVal = 'sub';

console.log('subObj.superVal 수정 후 subObj.superVal :', subObj.superVal);

console.log('subObj.superVal 수정 후 superObj.superVal :', superObj.superVal);

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <meta http-equiv="X-UA-Compatible" content="IE=edge">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <title>Document</title>

    <script src="1-5\_proto대체-inheritance.js"></script>

</head>

<body>

</body>

</html>

* Object.create() 이용

var superObj = {superVal : 'super'};

//var subObj = {subVal : 'sub'};

//subObj.\_\_proto\_\_ = superObj; // 표준으로 인정받지는 못하나 브라우저에서는 제공

var subObj = Object.create(superObj); // 이 방법이 더 추천

subObj.subVal = 'sub';

debugger;

console.log(subObj);

console.log('subObj.subVal :', subObj.subVal);

console.log('subObj.superVal :', subObj.superVal);

subObj.superVal = 'sub';

console.log('subObj.superVal 수정 후 subObj.superVal :', subObj.superVal);

console.log('subObj.superVal 수정 후 superObj.superVal :', superObj.superVal);

var kim = {

    name:'kim',

    first : 10,

    second : 20,

    sum : function(){return this.first+this.second;},

};

// var lee = {

//     name : 'lee',

//     first:100,

//     avg : function(){

//         return this.sum()/2;

//     }

// };

// lee.\_\_proto\_\_ = kim;

var lee = Object.create(kim); // 표준은 아니나 거의 모든 브라우저에서 지원됨

lee.name = 'lee';

lee.first = 100;

lee.avg = function() {

    return this.sum()/2;

};

console.log(lee.name + '의 lee.sum() :', lee.sum());

console.log(lee.name + '의 lee.avg() :', lee.avg());

console.log('kim.sum() :', kim.sum());

1. call과 bind

var kim = {name:'kim', first:10, second:20};

var lee = {name:'lee', first:10, second:10};

first = 100;

second = 200;

function sum(prefix='=>') {

    return prefix + (this.first + this.second);

}

// console.log(sum()); 과 동일

console.log('외부 변수 합 :',sum.call());   // this는 window

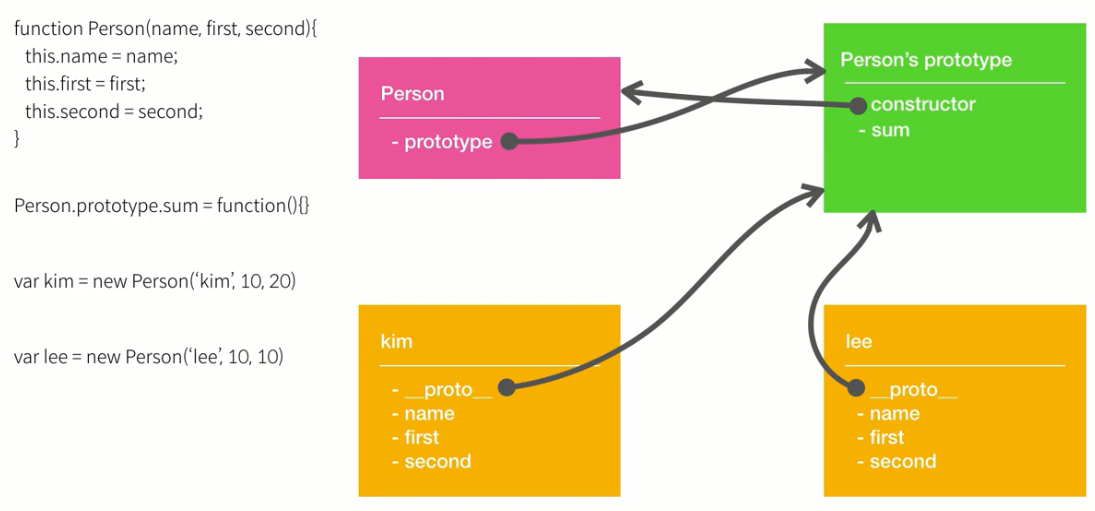
console.log('kim 합 :', sum.call(kim, '~~')); // this는 kim이 됨

console.log('lee 합 :', sum.call(lee));

// call과 apply는 비슷

var kimSum = sum.bind(kim, '->');

console.log(kimSum());



1. constructor를 이용한 상속

// class를 통한 상속이 더 나아

// class Person{

//     constructor(name, first, second){

//         this.name = name;

//         this.first = first;

//         this.second = second;

//     }

//     sum(){

//         return this.first + this.second;

//     }

// }

// class PersonPlus extends Person{

//     constructor(name, first, second, third){

//         super(name, first, second);

//         this.third = third;

//     }

//     sum(){

//         return super.sum()+this.third;

//     }

//     avg(){

//         return this.sum()/3;

//     }

// }

// 아래의 예제는 생성자함수를 통한 상속

function Person(name, first, second){

    this.name = name;

    this.first = first;

    this.second = second;

}

Person.prototype.sum = function(){

    return this.first + this.second;

};

function PersonPlus(name, first, second, third){

    Person.call(this, name, first, second);

    this.third = third;

}

//PersonPlus.prototype.\_\_proto\_\_ = Person.prototype;

PersonPlus.prototype = Object.create(Person.prototype);

PersonPlus.prototype.constructor = PersonPlus;

PersonPlus.prototype.avg = function(){

    // return (this.first + this.second + this.third)/3;

    return this.sum()/3;

};

var kim = new PersonPlus('kim', 10, 20, 30);

console.log('kim.sum() :', kim.sum());

console.log('kim.avg() :', kim.avg());

console.log('kim.constructor :', kim.constructor);

1. react 문법에 다가가기

class App extends Component{

    render(){

        return (

            <div>

                <h1>WEB</h1>

                <ol>

                    <li>html</li>

                    <li>css</li>

                    <li>js</li>

                </ol>

                Hello, Web!

            </div>

        )

    }

}