10.21

**객체**

- object. **속성** + **함수**로 이루어짐. (특징, 특색 / 하는 일)

- 어떻게 만드는가?

-- 리터럴 방식

-- 객체생성자 함수 방식

-- prototype 방식

- 객체 데이터

        // 자료를 다루는 방법 : 배열, 객체 데이터(JSON)

        // 객체 데이터

        const product={

            name: `건조망고`,

            type: `당절임`,

            weight: 3,

            arr: [1,2,3,4]

        }   // object로 인식 됨. name (key) : `건조망고` (value)

            // 배열자체가 객체 데이터에 들어갈 수도 있음.

            // 데이터 보기

            console.log(product);

            console.log(product.type); //객체데이터에서 type에 해당하는 데이터를 꺼내 볼 수 있음.

            //데이터 바꾸기

            product.name = `바나나` //객체데이터의 name에 해당하는 데이터를 바꿀 수 있음.

            console.log(product.name); // 건조망고 -> 바나나

            // 배열 데이터 꺼내오기

            const result = product.arr;

            console.log(`result = ${result}`);

            console.log(result[1]);//arr의 2번째 애를 가져오고 싶은 경우

            //데이터를 넣고 싶을 때

            product.date = '2022.10.21'

            console.log(product);

            //데이터를 삭제하고 싶을 때

            delete product.weight;

            console.log(product);

        //리터럴 방식 6-1-1.html

        객체 = "속성 + 메소드(함수)"

        const pet = {

            name : `구름`, // 속성 (key : name / value : 구름)

            eat : function(food) {

                console.log(`구름이 ${food}밥 먹는다`); //function을 이용한 메소드 적용.

            }

        }

        console.log(pet.name); // 객체의 이름 가져옴.

        pet.eat(); // 객체 pet의 eat function 사용 : 구름이 밥먹는다

        pet.eat(`츄르`); // 객체 pet의 function(추르) 사용 : 구름이 츄르밥 먹는다

        // 속성 나이, 종 추가 해보자

        const pet = {

            name : `구름`, // 속성 (key : name / value : 구름)

            age : `5살`,

            gender : `M`,

            eat : function(food) {

                console.log(`구름이 ${food}밥 먹는다`); //function을 이용한 메소드 적용.

                return (food + '밥을 먹는다')

            },

            sleep : function (where) {

                console.log(`구름이 ${where}에서 잔다`);

                return (where + '에서 잔다')

            }

        }

        console.log(`이름은 ${pet.name}, 나이는 ${pet.age}, 성별은${pet.gender}. ${pet.sleep('소파')}`);

        // function이 log가 이미 있어서 여기 줄에서 에러가 남. 즉 return을 이용하여 특정 값을 내보내주고, 그 값으로 씀.

        // 화살표 함수로 만들어보기

        eat : (food) => {

            console.log(`구름이 ${food}밥 먹는다`);

            return (food + '밥을 먹는다')

        },

        sleep : (where) => {

            console.log(`구름이 ${where}에서 잔다`);

            return (where + '에서 잔다')

        }

    // 2. 객체 생성자 방식

        // 객체 : 속성, 메소드

        function Pet(namevalue) //객체를 만드는 함수이기 떄문에 이름을 대문자로 시작함

        {

            // 속성 만들기

            this.name = namevalue

            // this? 현재 객체를 의미함.

            // 즉, Pet라는 객체 안에 있는 name이다를 의미함.

            // 메소드 만들기

            this.eat = function(food) {

                console.log(`${this.name}이 ${food} 먹는다`);

            }

        }

        //객체 생성

        /\* 왜 객체를 쓰는가? 코드의 생산성을 높이기 위하여.

        여러개의 변수를 만들고 객체를 사용하면 생산성이 높아진다? \*/

         /\* 변수의 종류가 많을때, 그때마다 일일이 함수를 만들 수 없으니

        거기서 이름만 변경해서 사용할 수 있도록 하기 위해 객체를 선언함\*/

        const pet = new Pet(); // 변수 만들고 new+객체명 선언

const pet = new Pet("구름");

        const pet2 = new Pet("별"); // this.name 에 '별'이 할당 됨.

        const pet3 = new Pet("달");

        const pet4 = new Pet("해");

        /\* 객체가 생성 될 때 어떤 일이 벌어지는가?

        ☆ 속성에 값이 할당된다 ☆

        pet, pet2, pet3, pet4라는 4개의 방이 생성 됨.

         \*/

        pet.eat(`밥`)

        pet2.eat(`밥`)

        pet3.eat(`밥`)

        pet4.eat(`밥`)

        /\* pet2기준, 이미 this.name에 "별" 이라는 이름이 저장되어있는

        상태에서 eat함수에 `밥`이라는 값이 들어가므로

        "별이 밥 먹는다" 라는 결과값이 나옴. \*/

function Pet(namevalue,age) //객체를 만드는 함수이기 떄문에 이름을 대문자로 시작함

        {

            // 속성 만들기

            this.name = namevalue

            // this? 현재 객체를 의미함.

            // 즉, Pet라는 객체 안에 있는 name이다를 의미함.

            this.age = age // new pet 옆에 새로 들어오는 값을 age로 받음

            // 메소드 만들기

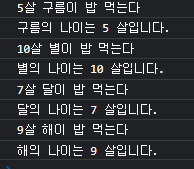
            this.eat = function(food) {

                console.log(`${this.age}살 ${this.name}이 ${food} 먹는다`);

            }

const pet = new Pet("구름", 5);

        const pet2 = new Pet("별", 10); // this.name 에 '별'이 할당 됨.

pet.eat(`밥`)

        pet.age1()

pet2.eat(`밥`)

        pet2.age1()

        pet3.eat(`밥`)

        pet3.age1()

        pet4.eat(`밥`)

        pet4.age1()

// 3. prototype 방식 만들기

       // 이번에도. 앞으로도. 항상. 객체 = 속성 + 메소드

       function Pet(nameVal,ageVal) {

        //속성

        this.name = nameVal;

        this.age = ageVal;

       }

       /////////////function 밖에다 만드는 메소드 (차이점!)

       // 객체명.prototype = function(){}

       Pet.prototype.eat = function (food){

        console.log(`${this.name}이/가 ${food} 먹는다.`);

       }

       Pet.prototype.agePrint = function(){

        console.log(`${this.name}이/가 ${this.age} 살이다.`);

       }

       const pet = new Pet('구름',7);

       pet.eat("밥");

       pet.agePrint();

자료 객체형

        let str = "제 이름은" // 자료도 객체화 되어있고, 배열로 할 수 있다.

        let num = 1.111111111111111

        console.log(num.toFixed(2)); // 레퍼런스에 있는대로 숫자 함수를 바로 사용할 수 있다.

        for (let i=0 ; i<str.length; i++){

            console.log(str[i]);

    Number('33')

    const f = new Number(273.4234243211) // 숫자를 쓰기 위해 쓰는 객체.

    console.log(f.toFixed(2)); //소숫점 2자리까지 남기고 자름

    const d = new Date()

    console.log(d); //현재 날짜. '5'라고 나옴.

    console.log(d.getDay()); //현재 날짜. '5'라고 나옴. (일],월.화.수.목.금,토)

    console.log(d.getHours());

    console.log(d.getFullYear());

        let str = "the orange test WWW" // 자료도 객체화 되어있고, 배열로 할 수 있다.

        // const word = str.split(' ') // 단어를 잘라서 배열형태로 만들어줌. (빈칸 한 칸)

       // const word = str.split() //통째로 배열형태로 만들어줌. (빈칸 x)

        const word = str.split('') //하나씩 배열형태로 만들어줌. ('')

        // 결국 배열형태로 return함.

객체 데이터 심화 & 레퍼런스 사이트 활용 객체 <-> 문자열

        // 배열안에 객체데이터 두개를 넣는다.

        const product2 = [{

            name: `건조망고`,

            type: `당절임`,

            weight: 3

        },{

            name: `양파`,

            type: `초절임`,

            weight: 5

        }]

        console.log(JSON.stringify(product2)); // 객체의 모든 데이터를 문자열로 바꿔줌.

        // 문자열을 객체로 바꾸고 싶은 경우

        let product3 = JSON.stringify(product2); // 문자열

        console.log(product3);

        console.log(JSON.parse(product3)); // 객체화

        console.log(product2);

        console.log(product2[0]); // 첫번째 배열 가져올 수 있음

        console.log(product2[1]); // 두번째 배열 가져올 수 있음