**11/15 김민식 OJT Day 11**

**Html/css/javascript/jquery 기본교육**

**JavaScript 재입문하기(JS 튜토리얼)**

[생활코딩 - 자바스크립트(JavaScript) 기본 | 학습 페이지 (inflearn.com)](https://www.inflearn.com/course/%EC%A7%80%EB%B0%94%EC%8A%A4%ED%81%AC%EB%A6%BD%ED%8A%B8-%EC%96%B8%EC%96%B4-%EA%B8%B0%EB%B3%B8/unit/2456?tab=curriculum) – 강의 학습

18. 객체지향 프로그래밍 소개

* OOP(Object Oriented Programming) : 프로그램을 구성하고있는 로직을 상태와 행위로 이루어진 객체로 만드는 것
  + 객체를 분리 (그룹핑) 하여 재활용성을 높임
  + 추상화 (abstract) – 해결해야될 문제를 단순화
  + 부품화 – 메소드는 부품화의 예, 코드양을 극적으로 감소 가능
  + 은닉화(Information Hiding), 캡슐화(encapsulation) – 내부의 동작 방법을 숨기고, 사용자에게 부품의 사용방법만 노출
  + 인터페이스 – 잘만들어진 부품은 부품과 부품을 서로 교환 가능, 인터페이스는 연결점

19. 생성자와 new

* 자바스크립트는 prototype – based programming
* 객체 생성 방법

        var person = {}; //생성방법 1

        person.name = 'kms';

        person.introduce = function(){

            return 'My name is '+this.name;

        }

        document.write (person.introduce());

        var person = { //생성방법 2

            'name' : 'egoing',

            'introduce' : function(){

                return 'My name is '+this.name;

            }

        }

* + 객체 내의 변수를 프로퍼티 (ex. person.name)
  + 객체 내의 함수를 메소드 (ex. person.introduce)
* 생성자와 new
  + 자바스크립트는 함수형 언어
  + 생성자 : 객체를 만드는 역할을 하는 함수
    - 자바는 클래스 안에 생성자안에 소속, 자바스크립트는 생성자는 함수 자체
    - 함수를 호출할 때 new를 붙이면 새로운 객체를 만든 후에 이를 리턴

        function Person(){};

        var p = new Person();

        p.name = 'egoing';

        p.introduce = function(){

            return 'My name is '+this.name;

        }

* + - 생성자의 재사용 예

       function Person(name){

        this.name = name;

        this.introduce = function(){

            return 'My name is '+this.name;

        }

       }

       var p1 = new Person('kms');

       document.write(p1.introduce()+"<br/>");

       var p2 = new Person('minsik');

       document.write(p2.introduce());

* + - 객체를 새로 생성할 때마다 초기화 (Init, Initialize)

20. 전역객체

* 모든 객체는 전역객체의 프로퍼티
* 모든 전역변수와 함수는 사실 window 객체의 프로퍼티. 객체를 명시하지 않으면 암시적으로 window의 프로퍼티로 간주

        var o = {'func':function(){

            alert('hello');

        }}

        o.func();

        window.o.func();

21. this

* 함수와 this
  + This는 함수 내에서 함수 호출 맥락(context)를 의미 (가변적)

        function func(){

            if(window === this){

                document.write("window === this");

            }

        }

        func(); //this는 전역객체인 window와 같다

* 메소드와 this
  + 객체의 소속인 메소드의 this는 그 객체를 가르킨다.
* 생성자와 this

        var funcThis = null;

        function Func(){

            funcThis = this;

        }

        var o1 = Func();

        if(funcThis === window){

            document.write('window </br>');

        }

        var o2 = new Func();

        if(funcThis === o2){

            document.write('o2 </br>');

        }

* 객체로서 함수
  + Apply와 this

        var o = {};

        var p = {};

        function func(){

            switch(this){

                case o:

                    document.write('o<br />');

                    break;

                case p:

                    document.write('p<br />');

                    break;

                case window:

                    document.write('window<br />');

                    break;

            }

        }

        func();

        func.apply(o);

        func.apply(p);

22. 상속 (inheritance)

* 객체의 로직을 그대로 물려 받는 또 다른 객체를 만들 수 있는 기능
  + 기본 사용법

        function Person(name){

            this.name = name;

        }

        Person.prototype.name = null;

        Person.prototype.introduce = function(){

            return 'My name is '+this.name;

        }

        function Programmer(name){

            this.name = name;

        }

        Programmer.prototype = new Person();

        var p1 = new Programmer('kms');

        document.write(p1.introduce()+"<br/>");

* + 응용 방법

        function Person(name){

            this.name = name;

        }

        Person.prototype.name = null;

        Person.prototype.introduce = function(){

            return 'My name is '+this.name;

        }

        function Programmer(name){

            this.name = name;

        }

        Programmer.prototype = new Person();

        Programmer.prototype.coding = function(){

            return "hello world";

        }

        function Designer(name){

            this.name = name;

        }

        Designer.prototype = new Person();

        Designer.prototype.design = function(){

            return "beautiful!";

        }

        var p1 = new Programmer('kms');

        document.write(p1.introduce()+"<br/>");

        document.write(p1.coding()+"<br />");

        var p2 = new Designer('msk');

        document.write(p2.introduce()+"<br/>");

        document.write(p2.design()+"<br />");

* Prototype
  + 한국어로는 원형
  + 프로토타입을 사용해서 상속 사용 가능
* Prototype chain
  + 다수의 프로토타입의 흐름

        function Ultra(){}

        Ultra.prototype.ultraProp = true;

        function Super(){}

        Super.prototype = new Ultra();

        function Sub(){}

        Sub.prototype = new Super();

        sub.prototype.ultraProp = 2;

        var o = new Sub();

        console.log(o.ultraProp);

23. 표준 내장객체의 확장

* 표준 내장 객체(Standard Built-in Object) : 자바스크립트가 기본적으로 가지고 있는 객체
  + Object, Function, Array, String, Boolean, Number, Math, Date, RegExp + 호스트환경에서 제공하는 API
  + 대비되는 용어 : 사용자 정의 객체
* 배열의 확장

        // function getRandomvalueFromArray(arr){

        //     var index = Math.floor(arr.length\*Math.random());

        //     return arr[index];

        // }

        // console.log(getRandomvalueFromArray(arr));

        Array.prototype.random = function(){

             var index = Math.floor(this.length\*Math.random());

             return this[index];

        }

        var arr = new Array('seoul','new york', 'ladarkh','pusan','tsukuba');

        console.log(arr.random());

24. Object

* Object 객체는 객체의 가장 기본적인 형태를 가지고 있는 객체
* 자바스크립트의 모든 객체는 Object 객체를 상속 받는다.
* Object API
  + MDN – JavaScript – JavaScript Reference – Object
    - Prototype의 유무에 따른 차이
      * Prototype이 없는 예 Object자체가 생성자, new를 안해도 됨(ex. Object.key)
      * Prototype이 있는 예, 생성자 생성 해야함 (ex. x.toString());

        //Object.keys() : 객체가 가지고 있는 키값들을 리턴

        var o = {"name":"kms","age":"29","city":"seoul"};

        console.log('Object.keys(o)',Object.keys(o));

        //name,age,city가 출력 (Key 값)

        //Object.prototype.tostring()

        var o = new Object();

        console.log(o.toString()); // [object Object]

        var a = [1,2,3];

        console.log(a.toString()); // 1, 2, 3

* Object의 확장

        Object.prototype.contain = function(needle){

            for(var name in this){

                if(this[name] === needle){

                    return true;

                }

            }

            return false;

        }

        var o = {'name':'msk','city':'seoul'}

        console.log(o.contain('msk'));

        var a = ['msk','jys','leezche'];

        console.log(a.contain('leezche'));

* + Object의 확장 위험
    - 모든객체에 영향을 끼침

25. 데이터 타입

* 원시 데이터 타입 (기본)
  + 숫자, 문자열, 불리언, null, undefined
* 객체 데이터 타입 (참조)
* 래퍼 객체
  + Str.length의 . 은 Object Access Operator(객체 접근 연산자)
  + Wrapper(래퍼 객체) : 원시 데이터 형을 객체처럼 다룰 수 있도록 하기 위한 객체
    - 문자열 -> String
    - 숫자 -> Number
    - 불리언 - > Boolean

26. 참조 (Reference)

* 복제 : 전자화된 시스템의 가장 중요한 특징
* 참조 : 데이터가 원시형이면 그 안에는 실제 데이터가 들어있고, 객체면 변수 안에는 데이터에 대한 참조 방법이 들어있다.

        var a = {'id':1};

        var b = a;

        b.id = 2;

        console.log(a.id); // 2

* 함수와 참조

[자바스크립트 기초 강좌 : 100분 완성 - YouTube](https://www.youtube.com/watch?v=KF6t61yuPCY) –강의학습

1. 변수

* Let과 const
  + Let : 한번선언 후 내용을 바꿀수 있다 (but 같은이름을 let으로 두번이상 설정 불가)
  + Const : 절대로 바뀌지 않는 상수, 수정 X
* 변수는 문자와 숫자, $와 \_만 사용
* 첫글자는 숫자 X
* 예약여 사용 X
* 가급적 상수는 대문자
* 변수명은 읽기 쉽고 이해 쉽게

1. Alert, prompt, confirm

* Alert : 알려줌, prompt : 입력 받음, confirm : 확인받음
  + 단점 : 스크립트 일시 정지, 스타일링 X, 장점 : 간단

[자바스크립트 중급 강좌 : 140분 완성 - YouTube](https://www.youtube.com/watch?v=4_WLS9Lj6n4)

* Computed property (계산된 프로퍼티)

        let a = 'age';

        const user = {

            'name' : 'Miek',

            [a] : 30 // age : 30

        }

* Object Method (객체 메소드)
  + Object.assign() : 객체 복제 (객체를 복제하여 합침)
  + Object.keys() : 키들을 배열로 반환
  + Object.values() : 값들을 배열로 반환
  + Object.entries: 키/값들을 배열로 반환
  + Object.fromEntries() : 키/값 배열을 객체로 반환
* Symbol (심볼)
  + Symbol : 유일한 식별자를 만들 때 사용 // new를 붙이지 않음 , 유일성 보장

질문사항

1. Domain, DTO, VO 차이 \_ nShop code
2. ~~MyBatis\_ xml파일 위치 \_ nShop code~~
3. .ajax 처리방법 (json파일의 위치 or dataflow) \_nShop/views/admin/common/code/insert.jsp
4. 시도해볼만한 자격증