VanillaJS 정리

1. 객체 정리
    - 객체
        - javascript에서 객체를 크게 나누면 3가지가 있다.
            - 1) 네이티브 객체
                - ECMAScript 사양에 정의된 객체(native object)
                - Object, String, Number, Boolean, Array, Function 등
            - 2) 호스트 객체
                - ECMAScript에는 정의되어 있지 않지만 자바스크립트 실행환경에 정의된 객체(host object)
                - 아래와 같은 객체들이 클라이언트 자바스크립트에 정의된 호스트 객체 이다.
                - 브라우져 객체 : Window, Navigator, History, Location …
                - DOM에 정의되어 있는 객체
                - Ajax를 위한 XmlHttpRequest 객체
                - HTML5 API
            - 3) 사용자 정의 객체
                - 사용자(개발자)가 정의한 자바스크립트 코드를 실행한 결과로 생성된 객체
        - javascript 프로그래밍으로 객체를 생성할때는 크게 5가지가 있다.
        - 기본적으로 이 4가지 형태의 객체의 의미를 알고 있으면 큰 틀에서 자바스크립트의 객체를 이해하는데는 도움이 된다.
            - 1) 리터럴 형태의 객체 var card = {suit:"하트",rank:"A"};
            - 2) 생성자 형태의 객체
                - function Card(suit, rank){
                - this.suit = suit;
                - this.rank = rank;
                - }
            - 3) 내장 객체 Object, String, Array …
            - 4) 함수 객체 function sqaure(x) {return x * x;}
            - 5) class로 객체 선언하기 **[es6추가]**
                - class Card{
                - constructor(suit,rank){
                - this.suit = suit;
                - this.rank = rank;
                - }
                - showCard(){
                - console.log(this.suit+"/"+this.rank);
                - }
                - }
        - 객체 리터럴 : 값을 선언하는 형태로 객체 생성하기
            - - 리터럴로 객체 생성하기
            - - var card = {suit:"하트",rank:"A"};
            - - var card = {"suit":"하트","rank":"A"}
            - - card.suit -> 하트 O
            - - card["rank"] -> A O

- - card.color -> undefiened
- - var obj = {}; -> 빈객체 선언
-
- - 객체 리터럴로 메소드 선언하기
- - var circle = {
- center : {x:1, y:0},
- radius : 25,
- area : function(){
- return Math.PI * this.radius * this.radius;
- }
- }

■ 생성자

- javascript에는 클래스가 없고, 생성자라는 함수로 객체를 생성한다.
- new 연산자로 객체를 생성할 것이라고 기대하고 만든 함수를 생성자라고 부른다.
- 생성자 이름은 관례적으로 그것이 생성자임을 알리기 위해 첫 글자를 대문자로 쓰는 파스칼 표기법을 사용한다.
- 생성자 안에서 'this.property이름'에 값을 대입하면 그 이름을 가진 property에 값이 할당된 객체가 생성된다. 이때 this는 생성자가 생성하는 객체를 가진다.
  - function Card(suit, rank){
  - this.suit = suit;
  - this.rank = rank;
  - }
  - var card = new Card(suit, rank);
- 그리고 생성자로 생성한 것을 똑같은 효과를 내는 형태로 '객체 리터럴'로 표현할 수도 있다.
  - var card = {};
  - card.suit = "하트";
  - card.rank = "A";
- 생성자와 new 연산자로 생성한 객체를 그 생성자의 인스턴스라고 한다.
- 인스턴스에는 실체라는 뜻이 있다.
- 객체지향 언어에서는 인스턴스는 클래스를 생성한 실체를 뜻한다.
- 이때 클래스는 객체를 실체로 만들기 위한 설계도고 그 설계도로 생성한 실체가 바로 인스턴스다.
- 자바스크립트에는 클래스가 없다.
- 따라서 생성자 자체가 함수 객체라는 실체이다.
- 하지만 생성자가 클래스처럼 객체를 생성하는 역할을 담당하고 있으므로 생성자로 생성한 객체도 인스턴스라고 부르는 것이 관례이다.
- 생성자의 역할
  - 객체를 생성하고 값을 초기화 하는 역할을 한다.
  - 생성자를 사용하면 이름은 같지만 프로퍼티 값이 다른 객체(인스턴스) 여러개를 간단히 생성할 수 있다.
    - var card1 = new Card("하트","A");
    - var card2 = new Card("클럽","K");
    - var card3 = new Card("스페이드","2");

■ 내장 객체(built in object, 기본객체)

- javascript에서는 처음부터 사용할 수 있는 내장객체(built in object, 기본객체)가 제공된다.

- 내장 객체는 자바스크립트 언어의 뼈대를 구성한다.
- Object : 자사스크립트의 최상위 객체
- String : 문자열을 처리하는 객체
- Number : 숫자를 처리하는 객체
- Boolean : 논리값을 처리하는 객체
- Array : 배열 데이터를 처리하는 객체
- ...

■ 함수(function)
- 함수 선언문으로 함수 정의하기
  - function sqaure(x) {return x * x;}
- 함수 리터럴로 정의하는 방법
  - var square = function(x) {return x * x;}
- Function 생성자로 정의하는 방법
  - var square = new Function("x", "return x * x");
- 화살표 함수(arrow function) 표현식으로 정의하는 방법
  - var sqaure = x => x * x;
- 즉시 실행 함수
  - (function(){...})();
  - 즉시 실행 함수에도 인자값을 넘길 수 있다.
    - (function(a,b){...})(1,2);
  - 즉시 실행 함수에도 이름을 붙일 수 있지만, 함수 내부에서만 유효하다.
    - (function fact(n){
    - if(n<=1) return 1;
    - return n*fact(n-1);
    - })(5); // 120
- 함수 정의식을 함수값으로 만들 수 있다.
  - +function(){...}()
  - +function(){
  - return 1+1;
  - }()