10.20

**네이밍 가이드**

: 항상 의미가 있게 지어야 한다.

**1. 카멜 : 변수, 함수명에 주로 쓰임. 앞에는 소문자 뒤에는 대문자**

-userList, newTodoList

**2. 파스칼 : 클래스, 객체명에 주로 쓰임**

- UserList, NewTodoList

3. 스네이크 케이스 : 파일명에 주로 쓰임

user\_list, new\_todo\_list

4. 케밥표기법 : css 클래스 명에 주로 쓰임

user-list, new-todo-list

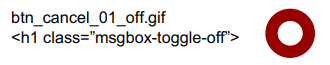
5. 헝가리안 표기법

$userList, $newTodoList

네이밍 기본 규칙

- 시작이름은 영 대문자, 숫자, 특문으로 시작 안함 .

- 네이밍 조합은 외형\_의미\_순서\_상태

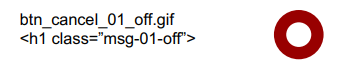


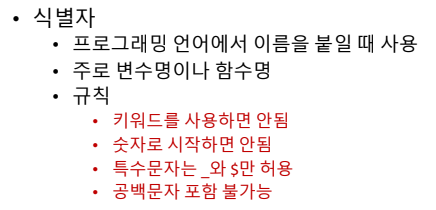
- 하이픈(-)은 CSS 네이밍에 사용하는 것을 권장

- 언더스코어(\_)는 파일, 폴더명 html의 form, input 태그의 name에 사용하는 것을 권장



- class, 파일 등 숫자 사용시 한자리 정수는 사용하지 않음(기본 2자리 정수 사용)





**문제풀이**

**!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!For(const of) / for (const in) 더 잘 기억해두기!!!!!!!!!!!!!!!!!!!!!!!!!**

**알고리즘은 맞췄으나 함수를 몰라서 틀린다는게 화가난다.**

Rest parameter

// sample(1,2,3,4); // 이렇게 매개변수가 바뀌었을 때 여러개를 만들어야 할 경우가 있는데, 그때마다 함수를 복사해서 할건가?

        // 매개변수가 가변적일 때 사용하는 방법

        function sample(...items) {

            console.log(items);

        }

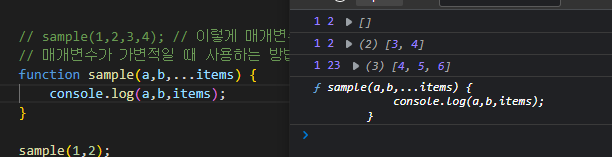
        sample(1,2);

        sample(1,2,3,4);

        sample(1,23,4,5,6);

        console.log(sample);

        //매개변수가 몇개든지 배열로 받아서 출력해줌.



a,b를 지정해줘서 정상적으로 뜨고, 나머지 rest들은 배열로 뜨는 모습.

            /\* rest 이용하여 최소값 구해내는 방법  \*/

            function min(...items) {

                console.log(items);

            let output = items[0];

            for (const item of items) {

                if (output > item){

                    output = item;

                }

            }

            return output;

        }

        console.log(`최솟값 : ${min(535,534522,23,42,1231)}`);

**-심화 : 숫자, 배열형 여부 판단**

          /\* rest 이용하여 최소값 구해내는 방법  \*/

            function min(first, ...rests) {

            let output;

            let items;

                // 받는 매개변수가 숫자형인지 배열인지 판단을 해줘야함. 그거에따라 달라지므로.

                if(Array.isArray(first)){ //배열이면 true, 아니면 false

                    output = first[0];

                    items = first;

                } else if (typeof(first) == 'number') {

                    output = first; // 숫자배열의 '52' 대응

                    items = rests; // 52 제외한 나머지 숫자 대응 (어차피 나머지는 배열이니까)

                }//숫자형인지 여부 확인

            for (const item of items) {

                if (output > item){

                    output = item;

                }

            }

            return output;

        }

        testArr = [535,534522,23,42,1231];

        console.log(`최솟값(숫자) : ${min(535,534522,23,42,1231)}`);

        console.log(`최솟값(배열) : ${min(testArr)}`);

        // 2. spread operator (전개연산자)

            function sample(...items) {

                console.log(items);

            }

            const array = [1,2,3,4];

            console.log(`전개 연산자를 사용하지 않을 때 : ${sample(array)}`);

            // [[1,2,3,4]]

            console.log(`전개 연산자를 사용할 때 : ${sample(...array)}`);

            // [1,2,3,4]

            // undefined 나오는 이유? return 값이 없어서 그럼.

            // 전개연산자를 쓰지 않으면 일반적인 array로 바로 나오지만 (배열안에 배열 형태),

            // 전개연산자를 사용하면 '...' 배열이 전개되어서 나옴

        // 3. 기본 매개변수  (함수의 매개변수 옆에 '='을 추가해서 기본 매개변수를 지정해줌)

            function earnings(name, wage=9000, hours=40) {

                console.log(`${name}님의 급여 정보`);

                console.log(`시급 : ${wage}`);

                console.log(`근무시간 : ${hours}`);

                console.log(`급여 : ${wage \* hours} 원`);

                console.log(`==========`);

            }

            earnings('구름') // wage, hours가 정의되지 않아서 시급, 근무시간이 나오지 않음

            earnings('별',10000) // hours가 정의되지 않아서 급여가 나오지 않음

            earnings('달',10000,52)

            // 기본 매개변수가 지정되어 있어도, 뒤에 추가로 기입된 값이 있다면 그게 더 우선해서 적용됨

        //  4. 구버전 자바스크림트에서 매개변수 구하기

        function sample() {

            console.log(arguments); //매개변수를 배열로 받아옴.

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

                console.log(`${i} 번째 : ${arguments[i]}`);

            }

        }

        //rest 파라미터 생기기 전에 주로 씀.

        sample(1,2)

        sample(1,2,3)

        sample(1,2,3,4)

**디버깅 하는 방법**

함수 (계속)

- 콜백함수 : 매개변수로 함수를 전달.

    <script>

        function callThreetimes(callback){

            for(let i =1; i<4; i++){

                callback(i)

            }

        }

        function print(i){

            console.log(`${i}번째 함수 호출`)

        }

        callThreetimes(print); // 그동안 매개변수에 단순 숫자만 넣었다면, 함수를 넣을수가 있어짐

        // 매개변수에 함수를 주고, 그 함수를 이용해 반복 시키는 것. 이것이 콜백함수

    </script>

        // function print(i){

        //     console.log(`${i}번째 함수 호출`)

        // }

        callThreetimes(function(i){console.log(`${i}번째 함수 호출`)});

        //이것처럼 매개변수에 함수를 통으로 넣어줄수도 있음

        //forEach()

        const numbers = [273,52,103,32,57]

        numbers.forEach(function (value, index, array) {

            // console.log(`${array}`);

            console.log(`${index}번째 : ${value} `);

        });

        // forEach = 배열의 value, index를 가져와주면 numbers에 있는 배열을 가져와 뽑아낼 수 있음.

        // for문과 동일하게 움직이는 느낌임.

// map()

        const numbers = [273,52,103,32,57]

        let result = numbers.map(function(value, index, array){

            console.log(`${index}번째 : ${value} `);

            return value\*value;

        })

        // map : 리턴을 해서 새로운 배열을 만들 수 있음.

        // 데이터들을 다 묶어서 배열로 만들어 주는 느낌.

        console.log(result);

       const numbers = [0, 273,52,103,32,57]

//filter

        const evennumbers = numbers.filter(function(value){

            return value % 2=== 0

        }) /\* 조건을 걸어서 걸러주는 역할을 하는 filter \*/

        // 단순히 데이터만 보내주는게 아니라, 조건이 참일때 값을 보낼 수 있음.

                    // return값이 참일때만 해당 데이터를 모아준다.

        console.log(`결과 : ${evennumbers}`);

타이머 함수

        // setTimeout(function()){

        // setTimeout() -> setInterval or clearInterval 형태로 진행

        setTimeout(()=>{

            console.log(`2초후에 실행됩니다`)

        },2000)

        let cnt = 0;

        let id = setInterval(()=> {

            console.log(`2초 마다 실행됩니다 ${cnt}`);

            cnt++

        },2000)

        setTimeout(() => {

           console.log(`타이머 종료`);

           clearInterval(id) //clearinterval에 의해 setinterval 완전히 종료됨.

        }, 5000);

**// 복습 및 응용, 2초뒤 시작 이후 2초마다 반복, 3회반복시 종료하는 루틴**

        setTimeout(() => {

            console.log(`2초뒤 시작`);

        }, 2000);

        let reps = 0;

        let repeat = setInterval(() => {

            console.log(`2초마다 반복 ${reps}`);

            reps++;

            if(reps == 3){

                console.log(`${reps}회차이므로 종료`);

                clearInterval(repeat)

            }

        }, 2000);

즉시 실행 함수

      // 라이언 개발자

      function menu() {

        num += 100;

        alert(num);

      }

      menu();

      // 어피치 개발자

      function menu() {

        alert(num);

      }

      // 원래는 이따구로 한거를

      // 라이언 개발자

      (function () {

        var num = 100;

        function menu() {

          num += 100;

          alert(num);

        }

        menu();

      })(); // 울타리를 쳐서 보호해 줌

호이스팅

// 비익명 함수는 호이스팅이 되지만, 익명함수는 호이스팅이 안됨.

      // 위에서부터 순서대로 코딩을 시키는 자바스크립트와 달리, 호이스팅을 사용하면

      // 변수를 정의하지 않고 함수를 돌려도 작동이 가능하다 (호이스팅)

      //

      // function a() {

      //   document.write("haha");

      // }

      // a();

      b();

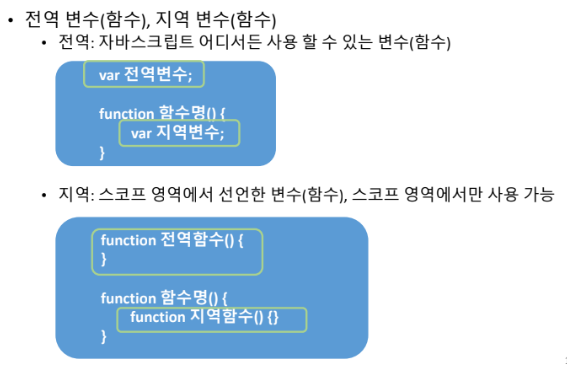
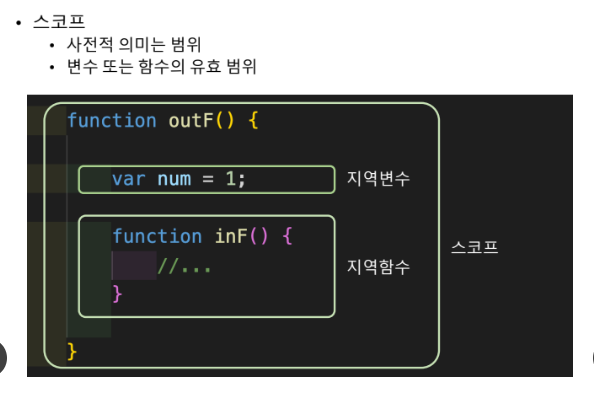
        // b사용하기 전에 선언을 해야하는데, 선언이 밑에 있어서 적용이 안됨.

      const b = function () {

        document.write("baba");

      };

스코프 : 변수, 함수의 유효범위



전역변수, 지역변수

            var num = 1; // 전역변수

            //함수선언

            function love(){

              var a = num+1; //지역변수

              console.log('a:',a);

            }

            //지역변수는 선언 된 해당 함수를 벗어났을 때, 사용할 수 없어짐

  function love(){

      str ='산';    //전역변수 (var을 쓰지 않으면 애가 전역변수로 인지함.)

      var sum ='강'; //지역변수



-메소드 체인

        let numbers = [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]

        // 메소드 체인

        let result = numbers.filter(function(value){

            console.log(value);

            return value % 2 === 0;

        })

        .map(function (value) {

            return value\*2

        }) //.을 붙여서 function을 이중으로 적용함. 적용된 결과값을 result로 보냄. 메소드 체인

        console.log(result);