2018년 인턴사원 교육

JavaScript

GS ITM

GS ITM

1. 변수

- 변수는 데이터를 저장하는 장소
- 변수는 데이터를 읽고 쓰고 할 수 있는 장소

```
var valname = '값';
```

var 는 변수 선언을 의미하는 키워드

break	else	instanceof	true
case	false	new	try
catch	finally	null	typeof
continue	for	return	var
default	function	switch	void
delete	if	this	while
do	in	throw	with

abstract	enum	int	short
Boolean	export	interface	static
byte	extends	long	super
char	final	native	synchronized
class	float	package	throws
const	goto	private	transient
debugger	implements	protected	volatile
double	import	public	

1. 변수

- 변수이름 만들 때 주의사항
  - 숫자로 시작 x
  - 대소문자 구분
  - 소문자로 시작
  - 상수일 경우 모두 대문자
  - 여러단어 조합일 경우 CamelCase 작성
  - 예약어(키워드) 사용 금지
  - $기호 _기호 쓸수 있음.

1. 변수

- 변수 데이터 종류
  - 숫자 : 정수형 10진수, 정수형 16진수(0x로 시작 0~9,A~F 주로 색상), 실수형
  - 문자형
  - 논리형 : true/false 1/0
  - undefined : 변수 초기화 하지 않은 상태

    함수 인자값 없이 호출

```
function myFunc(data1) {
    console.log("data1= " + data1);
}

myFunc("value1");
myFunc();
```

    존재하지 않는 객체의 프로퍼티에 접근

```
function MyClass() {
    this.name = "ddandongne";
}

var test1 = new MyClass();
console.log("test1.name = " + test1.name);
console.log("test1.userName = " + test1.userName);
```

  - null : 이 변수에 클래스의 인스턴스를 대입할 것

```
var data3 = null;
......
data3 = new MyClass();
```

1. 변수

- 변수 값 저장

```
var data1 = 10;
var data2 = data1;
var data3 = data1 + 20 + data2;

var data1 = 10;
var data2 = 10;
var data3 = 10 + 20 + 10;
```

- 변수 값 변경

```
var data1 = 10;
data1 = 20;
data1 = 30;
```

문제) 1과 2이 차이점은 무엇일까?

```
1. var data1 = 10;
   var data1 = 20;
   var data1 = 30;

2. var data1 = 10;
   data1 = 20;
   data1 = 30;
```

1. 변수

- 변숫값이 자동으로 읽혀지는 경우

  - 우측에 변수를 두는 경우

```
var data1 = 10;
var data2 = data1;
```

  - 함수 호출 시 변수를 매개변수 값으로 사용하는 경우

```
function test1(userName) {
    console.log("userName = " + userName);
}

var name = "ddandongne";
test1(name);
```

  - 연산자와 함께 사용하는 경우

```
var data1 = 10;
var data2 = 20;
var data3 = data1 + data2;
```

문제) 어떻게 해석될까?

```
var data1 = 10;
var data2 = 20;
var data3 = data1 + data2;
console.log(data3);
```

1. 변수

- 변수를 활용한 데이터 중복 제거 및 재사용

```
document.write("GSITM님 안녕하세요. 인턴교육에 오신걸 환영합니다.", "<br>");
document.write("GSITM님 안녕하세요. 인턴교육에 오신걸 환영합니다.", "<br>");
document.write("GSITM님 안녕하세요. 인턴교육에 오신걸 환영합니다.", "<br>");
document.write("GSITM님 안녕하세요. 인턴교육에 오신걸 환영합니다.", "<br>");
document.write("GSITM님 안녕하세요. 인턴교육에 오신걸 환영합니다.", "<br>");
```

```
var name = "이득영";
document.write(name + "님 안녕하세요. 인턴교육에 오신걸 환영합니다.", "<br>");
document.write(name + "님 안녕하세요. 인턴교육에 오신걸 환영합니다.", "<br>");
document.write(name + "님 안녕하세요. 인턴교육에 오신걸 환영합니다.", "<br>");
document.write(name + "님 안녕하세요. 인턴교육에 오신걸 환영합니다.", "<br>");
document.write(name + "님 안녕하세요. 인턴교육에 오신걸 환영합니다.", "<br>");
```

- 변수에 들어간 값 확인

```
var name = "이득영";

alert(name);
console.log(name);
document.write(name);
```

1. 변수

- 주석

  - 한 줄 주석

```
// 사용자 나이를 저장하는 변수
var age = 10;
// 사용자 몸무개를 저장하는 변수
var weight = 55.2;
// 사용자 이름을 저장하는 변수
var name = "이득영";
```

  - 여러 줄 주석

```
/*
 * 설명 :  화면에 사용자 정보를 출력하는 함수.
 * 매개변수 :
 *  age : 나이
 *  weight : 몸무게
 *  name : 사용자 이름.
 *
 * 리턴값 :
 *  없음.
 */
function showInfo(age, weight, name) {
    document.write("age=" + age, "weight=" + weight, "name=" + name);
}
```

1. 변수

- 배열
  - 변수에 여러 개의 데이터를 담기 위한 데이터 형

```
var userName = ['철수', '영희', '아영', '정희', '선희', '영자', '병철'];
```

```
document.write('userName[0] = ' + userName[0], '<br>');
document.write('userName[1] = ' + userName[1], '<br>');
document.write('userName[2] = ' + userName[2], '<br>');
document.write('userName[3] = ' + userName[3], '<br>');
document.write('userName[4] = ' + userName[4], '<br>');
document.write('userName[5] = ' + userName[5], '<br>');
document.write('userName[6] = ' + userName[6], '<br>');
```

```
for(var i=0;i<userName.length;i++)
document.write("userName["+i+"] = "+userName[i],"<br>");
```

1. 변수

- 변수 종류
  - 전역변수 : 어디서나 접근 가능한 변수
  - 지역변수 : 특정 영역에서만 사용가능, 주로 함수내부에서 사용
  - 매개변수(파라미터) : 함수외부에서 함수내부로 데이터를 전달하기 위한 변수
  - 멤버변수(프로퍼티) : 클래스 내부에서 만들어지고, 주로 객체에서 사용하는 정보 담는 변수

```
var globalV = "전역변수";

window.onload = function() {
    global2 = "전역변수";
}

function func1() {
    var local1 = "지역변수";
}
```

```
var globalV = "전역변수";

window.onload = function() {
    var local1 = "지역변수";
    func1(100, 200);
}
function func1(num1, num2) {
    var local1 = "지역변수";

    document.write("매개변수 num1=" + num1 + ", num2=" + num2);
}
```

```
function MyClass() {
    this.name = "멤버변수";
}

MyClass.prototype.showName = function() {
    document.write("name = ", this.name);
}
var objClass = new MyClass();
objClass.showName();
```

2. 기본 연산자

- 숫자 연산자

  + , - , *, / , %

- 문자 연산자

  +

- 복합 연산자

  +=, -=, *=, /=, %=

- 증감 연산자

  ++, -- (전위 연산자, 후위 연산자)

- 연산자 우선순위

#1. 초보 함수

- 함수 사용 전

```
var dan = 3
document.write(dan + " * 2 = " + (dan * 2), "<br>");
document.write(dan + " * 3 = " + (dan * 3), "<br>");
document.write(dan + " * 4 = " + (dan * 4), "<br>");
document.write(dan + " * 5 = " + (dan * 5), "<br>");
document.write(dan + " * 6 = " + (dan * 6), "<br>");
document.write(dan + " * 7 = " + (dan * 7), "<br>");
document.write(dan + " * 8 = " + (dan * 8), "<br>");
document.write(dan + " * 9 = " + (dan * 9), "<br>");
```

- 함수 사용 후

```
function print99DAN(dan) {
    document.write(dan + " * 1 = " + (dan * 1), "<br>");
    document.write(dan + " * 2 = " + (dan * 2), "<br>");
    document.write(dan + " * 3 = " + (dan * 3), "<br>");
    document.write(dan + " * 4 = " + (dan * 4), "<br>");
    document.write(dan + " * 5 = " + (dan * 5), "<br>");
    document.write(dan + " * 6 = " + (dan * 6), "<br>");
    document.write(dan + " * 7 = " + (dan * 7), "<br>");
    document.write(dan + " * 8 = " + (dan * 8), "<br>");
    document.write(dan + " * 9 = " + (dan * 9), "<br>");
}

print99DAN(3);
```

#2. 초보 클래스

```
function Calculator(){
  this.add = function(a,b){
      alert("두 수의 합은 "+(a+b)+"입니다.");
  }
  this.sub =function(a,b){
      alert("두 수의 차는 "+(a-b)+"입니다.");
  }
  this.mul=function(a,b){
      alert("두 수의 곱은 "+(a*b)+"입니다.");
  }
  this.div=function(a,b){
      alert("두 수의 나눈 값은 "+(a/b)+"입니다.");
  }
}
```

```
// 인스턴스 생성
var cal1 = new Calculator();
```

```
//접근연산자(.)를 사용한 메서드 접근
cal1.add(10,20);
```

3. 형변환

- 암시적 형변환

  - 자바스크립트에 의해 자동으로 형변환

  숫자형 + 문자형 = 문자형

  불린형 + 문자형 = 문자형

  불린형 + 숫자형 = 숫자형

```
var a = "30";
var result = 1 + a + 10;
console.log(result);
```

- 명시적 형변환

  - 개발자에 의해 수동적으로 형변환

```
var value = '123.456';

// 정수로 변환
console.log(parseInt(value));
// 실수로 변환
console.log(parseFloat(value));
// 정수,실수 모두로 변환
console.log(Number(value));

// 문자로 변환
console.log(String(value));

// 실수 문자로 변환
var value2 = 123.456;
console.log(value2.toFixed(2));
```

4. 조건문 if

```
if(여러분의 로또 번호==이번 주 로또 당첨 번호){
    난 집을 사겠다;
    // 여러분은 로또에 당첨된다면 뭘 할 건가요?
}
```

```
if(자바스크립트마스터여부=="성공"){
    연봉 500원 인상을 위한 보스와의 단판승부하러 가기
}
```

```
if(보스와의 단판승부=="성공"){
    //기존연봉=기존연봉+500;
    기존연봉+=500;
}
else{
    회사 알아보기;
    밥 대신 라면만 먹기;
}
```

4. 조건문 if

- 비교 연산자

  - 두 값을 비교할 때 사용

  >, <, ==, ===, !=, !==, >=, <=

- 논리 연산자

  - 여러 개의 비교 연산자를 묶을 때 사용

  &&, ||

- 조건부 연산자

  - 2단계 if를 간결하게 표현할 때 사용

  (조건식) ? 실행구문1 : 실행구문2;

  → 조건식이 true이면 실행구문1 실행, 그렇지 않으면 실행구문2 실행

5. 조건문 switch

```
var luckyValue = window.prompt("두근! 두근! 행운의 번호를 고르세요.");
if(luckyValue=="3")
    document.write("당첨! 냉장고 ");
else if(luckyValue=="2")
    document.write("당첨! 세탁기");
else if(luckyValue=="1")
    document.write("당첨! TV");
else
    document.write("꽝입니다.");
```

```
var luckyValue = window.prompt("두근! 두근! 행운의 번호를 고르세요.");

switch(luckyValue){
    case "3" :
        document.write("당첨! 냉장고<br>");
        break;
    case "2" :
        document.write("당첨! 세탁기<br>");
        break;
    case "1" :
        document.write("당첨! TV<br>");
        break;
    default :
        document.write("꽝입니다.<br>");
}
```

- if와 switch의 차이점
  - if 조건식에 비교연산자 사용 시 switch 변경 불가
  - if 조건식에 == (일치) 의 경우에만 switch 사용 가능

6. 반복문 for

- 특정구문을 여러 번 반복해서 실행할 때 사용하는 제어문
- 가장 일반적으로 사용하는 반복문
- 반복 횟수가 정해진 경우 주로 사용

- 단일 for문

```
for(초기값; 조건식; 증감) {
  실행구문;
  .........
}
```

```
for (var i = 1; i <= 10; i++) {
    document.write(i + "<br>");
}
```

- 다중 for문

```
for(초기값; 조건식; 증감) {
  실행구문;
  .........
  for(초기값; 조건식; 증감) {
    .........
  }
}
```

```
for (var i = 1; i <= 10; i++) {
    document.write(i + "<br>");
    for (var j = 1; j <= 10; j++) {
        document.write(j + "<br>");
    }
}
```

6. 반복문 for

- continue

```
for (var i = 1; i <= 10; i++) {
    continue;
    document.write(i + "<br>");
}
document.write("최종 i=" + i + "<br>");
```

- break

```
for (var i = 1; i <= 10; i++) {
    break;
    document.write(i + "<br>");
}
document.write("최종 i=" + i + "<br>");
```

6. 반복문 for

- 실습1 for문을 이용해서 실행화면처럼 출력해 주세요.

  실행화면:

```
*          *****
**         ****
***        ***
****       **
*****      *
```

- 실습2 for문을 이용하여 구구단을 출력해 주세요

  예) 2 * 1 = 2

- 실습3 배열의 총 합을 구해 보세요

  배열 10,20,30,40,50

- 실습4 1 부터 100까지의 총 합을 출력해 주세요

  1 + 2 + 3 + 4 ..... + 100 = ?

7. 반복문 while

- 패스워드가 1234 이면 "환영합니다." 라는 메시지를 출력한 후 멈추고 그렇지 않으면 "패스워드를 잘못 입력했습니다." 라고 출력한 후 계속해서 패스워드를 입력 받기 위해서 for문을 사용해 보세요

```
for(var i=0;i<10000;i++){
    var value = window.prompt("패스워드를 입력해주세요.");
    if(value=="1234"){
        alert("환영합니다.")
        break;
    }
    else{
        alert("잘못 입력했습니다. 다시 입력해주세요.")
    }
}
```

- while문 이용

```
while (true) {
    var value = window.prompt("패스워드를 입력해주세요.");

    if (value == "1234") {
        alert("환영합니다.");
        break;
    } else {
        alert("잘못 입력했습니다.다시 입력해주세요.")
    }
}
```

- while문 사용
  - 무한반복 처리
  - 파일 읽기
  - 파일 쓰기
  - 파일 전송
  - DB 데이터 출력

7. 반복문 while

- 실습1 for문을 while문으로 변경해 보세요

```
var dan=3;
for(var i=1;i<=9;i++){
   document.write(dan+"*"+i+"="+(dan*i)+"<br>");
}
```

- 실습2 end가 입력될 때까지 숫자를 계속해서 입력받아 다음과 같이 출력되게 만들어 주세요.

```
var value = window.prompt(i + "번째 입력 ");
```

8. 함수 기초

- 이름을 5번 출력

  출력결과

  1. ddandongne
  2. ddandongne
  3. ddandongne
  4. ddandongne
  5. ddandongne

```
document.write("1. ddandongne<br>");
document.write("2. ddandongne<br>");
document.write("3. ddandongne<br>");
document.write("4. ddandongne<br>");
document.write("5. ddandongne<br>");
```

반복문 활용

```
for(var i=1;i<=5;i++)
    document.write(i+". ddandongne<br>");
```

8. 함수 기초

- 함수 쉽게 만들기

```
document.write("안녕 하세요. 환영합니다.", "<br>");
document.write("안녕 하세요. 환영합니다.", "<br>");
document.write("안녕 하세요. 환영합니다.", "<br>");
```

```
function 함수이름(){
    실행구문;(포장할 내용을 여기에 작성해 주세요.)
}
```

```
function hello(){
    document.write("안녕하세요. 환영합니다.","<br>");
}
```

```
function hello(){
    document.write("안녕하세요. 환영합니다.","<br>");
}
hello();
hello();
hello();
```

8. 함수 기초

- 다음 내용을 printerStar 라는 함수로 만들어 보세요

```
var star = "";
for (var i = 1; i <= 5; i++) {
    for (var m = 0; m < i; m++) {
        star += "*";
    }
    star += "<br>";
}
document.write(star);

var star = "";
for (var i = 1; i <= 5; i++) {
    for (var m = 0; m < i; m++) {
        star += "*";
    }
    star += "<br>";
}
document.write(star);
```

단계01: 함수로 포장할 내용(중복코드 또는 로직)찾기

단계02: 빈 함수 만들기

단계03: 중복코드 포장하기

단계04: 함수 호출

8. 함수 기초

- 지역변수와 전역변수

```
var name="test1";
function func1(){
    var name="test2";
    document.write("2. name = "+name,"<br>");
}

function func2(){
    var name="test3";
    document.write("3. name = "+name,"<br>");
}

function func3(){
    name="test20";
    document.write("4. name = "+name,"<br>");
}

document.write("1. name = "+name,"<br>");
func1();
func2();
func3();
document.write("5. name = "+name,"<br>");
```

결과 :

1. name = test1
2. name = test2
3. name = test3
4. name = test20
5. name = test20

8. 함수 기초

- 지역변수의 생명주기

```
function func1(){
    var name="gsitm";
    document.write("1. name = "+name);
}
func1();

// name은 지역변수이기 때문에 전역에서 사용할 수 없어요.
document.write("2. name = "+name);
```

실행 결과:

1. name = gsitm
2. name = undefined

8. 함수 기초

- 예제

```
var a = 10;
var b = 100;
function func1(){
    var b = 20;
    document.write("1. a = "+a+"<br>");
    document.write("2. b = "+b+"<br>");
    a = 50;
}
func1();

document.write("3. a = "+ a+"<br>");
document.write("4. b = "+b+"<br>");
```

풀이:

1. a = 10 (전역변수)
2. b = 20 (지역변수)
3. a = 50 (전역변수)
4. b = 100 (전역변수)

8. 함수 기초

- 함수 만들때 주의사항
  - 변수 이름 만드는 것처럼 함수 이름 역시 몇가지 지켜야할 규칙이 있습니다
  - 결론을 미리 이야기 하자면 함수이름은 일반 변수이름 짓는 방법과 동일하다고 보면 됩니다.

  1. 숫자로 시작하면 안되요.
  2. 대소문자 구분: name과 Name은 완전히 다른 함수입니다.
     함수 이름이 대문자로 해도 되지만, 일반적으로 함수이름이 대문자로 시작하는 경우
     클래스를 나타냅니다.
  3. 낙타 표기법(camelcase): 여러 단어가 조합되는 경우 아래와 같이 해주세요.

8. 함수 기초

- 예제1. 구구단 출력을 함수로 만들기

  이 코드를 print99dan이라는 함수를 만들어 포장해 주세요.

```
for(var i=2;i<=9;i++){
    document.write(i+"단 출력","<br>");
    for(var m=1;m<=9;m++){
        document.write(i+"*"+m+"="+(i*m),"<br>");
    }
    document.write("<br>");

}
```

- 예제2. 사칙연산 계산기 만들기

  실행 예:

```
document.write("1 결과 = "+calculator("+", 20, 10), "<br>");
document.write ("2 결과 = "+calculator("-", 20, 10), "<br>");
document.write ("3 결과 = "+calculator("*", 20, 10), "<br>");
document.write ("4 결과 = "+calculator("/", 20, 10), "<br>");
document.write ("5 결과 = "+calculator("%", 20, 10), "<br>");
```

실행 결과:

1 결과 = 30
2 결과 = 10
3 결과 = 200
4 결과 = 2
5 결과 = 지원하지 않는 연산자입니다.

9. 함수 기능

- 중복코드 제거 및 코드 재사용성

예제 01: 다음 내용은10, 15, 19가 홀수인지 짝수인지 출력하는 예제입니다.

함수를 이용해서 깔끔하게 만들어 주세요

```
// 중복코드 또는 재사용 코드 찾아 함수로 포장하기
function checkEvenOdd() {
    var n1 = 10;
    document.write(n1 + "은 ");
    if (n1 % 2)
        document.write("홀수입니다.<br>");
    else
        document.write("짝수입니다.<br>");
}
```

```
// 변경되는 부분을 매개변수로 만들기
function checkEvenOdd(value) {

    document.write(value + "은 ");
    if (value % 2)
        document.write("홀수입니다.<br>");
    else
        document.write("짝수입니다.<br>");
}
```

```
var n1 = 10;
document.write(n1 + "은 ");
if (n1 % 2)
    document.write("홀수입니다.<br>");
else
    document.write("짝수입니다.<br>");

var n2 = 15;
document.write(n2 + "은 ");
if (n2 % 2)
    document.write("홀수입니다.<br>");
else
    document.write("짝수입니다.<br>");

var n3 = 19;
document.write(n3 + "은 ");
if (n3 % 2)
    document.write("홀수입니다.<br>");
else
    document.write("짝수입니다.<br>");
```

10. 함수 중급

- 변수에 함수를 저장한 경우

```
function hello(name){
    document.write(name+"님 환영합니다.");
}
hello("GSITM");
var func = hello;
func("GSITM");
```

- 매개변수 값으로 함수를 사용한 경우

```
function hello1(){
    alert("hello.");
}
function hello2(){
    alert("안녕하세요.");
}
function execute(func){
    func();
}
execute(hello1);
execute(hello2);
```

10. 함수 중급

- 리턴값으로 함수를 사용한 경우

```
function createHello(){
    function hello(user){
        document.write(user+"님 방문을 환영합니다.");
    }
    return hello;
}

var result = createHello();
result("GSITM");
```

10. 함수 중급

- 함수 만드는 방법 3가지

1. 리터럴 방식

```
var hello = function(name){
    alert(name+"님 환영합니다.");
}
hello("GSITM");
```

2. 일반 방식

```
function hello(name){
    alert(name+"님 환영합니다.");
}
hello("GSITM");
```

3. 객체 방식

```
var hello2 = new Function("name", "alert(name+'님 환영합니다.');");
hello2("GSITM");
```

- 익명 함수
  - 리터럴 방식으로 만들어진 이름없는 함수
  - 함수를 재사용 하지 않을 경우 사용

10. 함수 중급

- 중첩 함수

  - 함수 내부에서만 사용하는 기능을 중첩함수로 만들어서 사용

```
function calculator(op, num1, num2) {
    var result = "";

    switch(op) {
    case "+" :
        result = num1 + num2;
        break;
    case "-" :
        result = num1 - num2;
        break;
    case "*" :
        result = num1 * num2;
        break;
    case "/" :
        result = num1 / num2;
        break;

    default :
        result = "지원하지 않는 연산자입니다";
    }
    return result;
}
```

```
function calculator(op, num1, num2) {
    var result = "";

    switch (op) {
        case "+" :
            result = add(num1, num2);
            break;
        case "-" :
            result = sub(num1, num2);
            break;
        case "*" :
            result = mul(num1, num2);
            break;
        case "/" :
            result = div(num1, num2);
            break;

        default :
            result = "지원하지 않는 연산자입니다";
    }
    return result;

    function add(num1, num2) {
        return num1 + num2;
    }

    function sub(num1, num2) {
        return num1 - num2;
    }

    function mul(num1, num2) {
        return num1 * num2;
    }

    function div(num1, num2) {
        return num1 / num2;
    }
}
```

10. 함수 중급

- 1,2,3 번에는 어떤 값이 출력될까?

```
var a=10;
var b=20;
var c=30;
function outer_func(){
    var b=200;
    var c=300;
    function inner_func(){
        var c=3000;
        document.write("1. a = "+a+"<br>");
        document.write("2. b = "+b+"<br>");
        document.write("3. c = "+c+"<br>");
    }
    inner_func();
}
outer_func();
```

10. 함수 중급

- 콜백함수

  - 함수 내부의 처리결과 값을 함수 외부로 내보낼 때 사용 (return 문과 비슷)

함수실행시 출력

1. 두수의 합은 00입니다.
2. 정답은 00입니다.

```
function calculator(op, num1,num2){
    var result="";

    switch(op) {
        case "+" :
            result = num1 + num2;
            break;
        case "-" :
            result = num1 - num2;
            break;
        case "*" :
            result = num1 * num2;
            break;
        case "/" :
            result = num1 / num2;
            break;

        default :
            result = "지원하지 않는 연산자입니다";
    }

    document.write("두 수의 합은"+result+"입니다.", "<br>");
}

calculator("+", 10,20);
```

10. 함수 중급

```
function calculator(op, num1, num2, callback){
    var result="";

    switch(op) {
        case "+" :
            result = num1 + num2;
            break;
        case "-" :
            result = num1 - num2;
            break;
        case "*" :
            result = num1 * num2;
            break;
        case "/" :
            result = num1 / num2;
            break;

        default :
            result = "지원하지 않는 연산자입니다";
    }

    callback(result);
}

function print1(result){
    document.write("두 수의 합은 = "+ result+"입니다.", "<br>");
}

function print2(result){
    document.write("정답은 ="+ result+"입니다.<br>");
}
calculator("+", 10,20, print1);
calculator("+", 10,20, print2);
```

- return VS 콜백함수

  - 단순한 처리는 리턴을 이용하는게 더 효율적

  - 콜백함수는 처리부분이 구현부분과 분리되어야 하는 경우나,
  구현부분은 하나로 하고 처리부분을 다양하게 만든 후 실행 시에
  연결해서 사용하는 경우 더 적합적

10. 함수 중급

- 동기 vs 비동기

```
alert("안녕하세요");
document.write("alert() 동작이 끝나야 이 내용은 실행됩니다.");
```

```
var count=1;

setInterval(function(){
    document.write("2. count = "+count);
},3000);

document.write("1. ajax() 동작이 모두 끝나지 않았어도 바로 실행됩니다.");
```

10. 함수 중급

- 클로저

  - 함수 내부에 만든 지역변수가 사라지지 않고 계속해서 값을 유지하고 있는 상태

```
// 다음 예제를 실행하면 1,2,3은 어떤 값이 출력될까요?
// 일반 함수인 경우
// 클로저를 사용한 경우
function createCounter(){
    var count=0;
    function addCount(){
        count++;
        return count;
    }
    return addCount;
}

var counter = createCounter();

document.write("1. count = " + counter(),"<br>");
document.write("2. count = " + counter(),"<br>");
document.write("3. count = " + counter(),"<br>");
```

```
$("#btnStart").click(function () {
    start();
    document.write("시작합니다.");
});

function start() {
    var count = 0;
    setInterval(function () {
        count++;
        document.write(count);
    }, 1000);
}
```

GS ITM

10. 함수 중급

- 클로저를 사용하면 좋은 점

  - 연관 있는 변수와 기능(중첩

  함수 내부에 데이터가 만들어

  가능 (private)

음.

데이터 생성

```
$(document).ready(function(){
    // 탭메뉴 코드가 동작할 수 있도록 tabMenu() 함수 호출
    tabMenu("#tabMenu1 li");
    tabMenu("#tabMenu2 li");
});


function tabMenu(selector){
    // 선택 한 탭메뉴를 저장할 변수
    var $selectMenuItem =null;

    // 메뉴 항목에 클릭 이벤트 등록
    $(selector).click(function(){

        // 기존 선택 메뉴 항목이 있으면 비 선택 상태로 만들기
        if($selectMenuItem!=null){
            $selectMenuItem.removeClass("select");
        }

        // 클릭한 메뉴 항목을 신규 선택 메뉴 항목으로 저장
        $selectMenuItem = $(this);
        // 선택 상태로 만들기
        $selectMenuItem.addClass("select");

    })
}
```

11. 타이머 함수

- setInterval() : 일정 시간마다 주기적으로 특정 구문을 실행하는 기능
- setTimeout() : 일정 시간이 지난 후 특정 구문을 딱 한번 실행하는 기능
- clearInterval() : 실행 중인 타이머 함수를 멈추는 기능

```
setInterval(function(){
    // 값 증가
    count++;
    // 값을 출력
    console.log(count);
},1000);
```

```
setTimeout(function () {
    console.log("안녕하세요. 환영합니다.")
}, 3000);
```

```
var $output = $("#output");
var count = 0;
var timerID=0;
timerID = setInterval(function() {
    // 값 증가
    count++;
    // 값을 출력
    $output.text(count);
}, 1000);


$("#stop").click(function() {
    // 여기에 풀이를 입력해주세요.
    clearInterval(timerID);
});
```

12. Math 클래스

- 랜덤 숫자

```
var value = parseInt(Math.random() * 50) + 50;
```

- 작은 값, 큰 값

```
var value = Math.min(최소값, 비교값);
```

```
var value = Math.max(최대값, 비교값);
```

- 예제1. 10에서 100 사이의 값이면 입력받은 숫자 값을 출력하고

  10보다 작은 경우 10을, 100보다 큰 경우 100으로 출력

```
var value = window.prompt("숫자를 입력해주세요.", 0);
```

```
if(value<10)
    value =10;

if(value>100)
        value=100;

alert(value);
```

```
value = Math.min(100,Math.max(10,value));
alert(value);
```

- 올림값, 내림값 구하기

  floor : 내림값, ceil : 올림값

13. String 클래스

- 프로퍼티

  - lenth : 문자열 개수

```
var str = window.prompt("문자를 입력해 주세요");
alert("문자열길이는 = "+str.length);
```

- 메서드

  - charAt(n) : n번째 문자 구하기

```
var ch = str.charAt(index);
```

  예제1. 입력받은 문자열을 1초마다 한 문자씩 출력하기

  - concat(str) : 문자열 뒤쪽에 str 연결하기

```
var strs = '문자열'.concat('추가');
```

  - indexOf : 문자열 위치 찾기 (찾지 못하면 -1)

```
var index = data.indexOf("sample");
```

  - slice : 문자열 자르기 (시작,끝 위치)
  - substr : 문자열 자르기 (시작, 길이)
  - replace : 다른문자열로 변경

```
var data = "안녕하세요 GSITM님 자바스크립트에 온 걸 환영합니다.";
var result = data.replace("GSITM", "인턴");
```

  - match(reg) : 정규표현식(reg)을 활용한 문자열 검색
  - split(str) : 문자열을 str로 분할해서 배열로 생성
  - toLowerCase() : 소문자로 변환
  - toUpperCase() : 대문자로 변환
  - trim() : 좌우공백 제거

13. String 클래스

- 예제1. 앞쪽 공백 문자를 제거하는 함수

```
// 앞쪽 공백 문자를 제거하는 함수
function ltrim(str) {
    // 문자열 값이 없는 경우
    if (str.length <= 0)
        return "";
    // 첫 번째 문자가 공백이 아니라면 검사할 필요 없이 입력값으 그대로 리턴
    var firstCh = str.charAt(0);
    if (firstCh != " ") {
        return str;
    }
    // 공백이 끝나는 위치 찾기
    for (var index = 1; index < str.length; index++) {
        // 공백 문자가 아닐때까지 찾기
        var ch = str.charAt(index);
        if (ch != " ") {
            break;
        }
    }
    //index 위치에서 마지막 위치 까지의 문자열 잘라내기
    var result = str.slice(index, str.length);
    return result;
}
```

13. String 클래스

- 예제2. 오른쪽 공백 문자를 제거하는 함수

```
// 오른쪽 공백 문자를 제거하는 함수
function rtrim(str){
    // 문자열 값이 없는 경우
    if(str.length<=0)
        return "";
    // 마지막 번째 문자가 공백이 아니라면 검사할 필요 없이 입력값으 그대로 리턴
    var firstCh= str.charAt(str.length-1);
    if(firstCh!=" "){
        return str;
    }
    // 공백이 끝나는 위치 찾기
    for(var index=str.length-1;index>=0;index--){
        // 공백 문자가 아닐때까지 찾기
        var ch = str.charAt(index);
        if(ch!=" "){
            index++;
            break;
        }
    }
    //0에서 index위치 까지의 문자열 잘라내기
    var result = str.slice(0, index);
    return result;
}
```

13. String 클래스

- 예제3. 세자리수 마다 콤마를 추가하는 기능

```
// 세 자릿수 마다 콤마(,) 추가 하는 기능
function won(value) {
    // 문자열 길이가 3과 같거나 작은 경우 입력 값을 그대로 리턴
    if (value.length <= 3) {
        return value;
    }
    // 3단어씩 자를 반복 횟수 구하기
    var count = Math.floor((value.length - 1) / 3);
    // 결과 값을 저정할 변수
    var result = "";
    // 문자 뒤쪽에서 3개를 자르며 콤마(,) 추가
    for (var i = 0; i < count; i++) {
        // 마지막 문자-3 에서 마지막 문자 자르기
        var length = value.length;
        var strCut = value.substr(length - 3, length);
        // 입력값 뒷쪽에서 3개의 문자열을 잘라낸 나머지 값으로 입력값 갱신
        value = value.slice(0, length - 3);
        // 콤마(,) + 신규로 자른 문자열 + 기존 결과 값
        result = "," + strCut + result;
    }
    // 마지막으로 루프를 돌고 남아 있을 입력값을 최종 결과 앞에 추가
    result = value + result;
    // 최종값 리턴
    return result;
}
```

14. Date 클래스

- 현재 시간 출력

```
var objDate = new Date(); // Date클래스의 인스턴스 생성
var hours = objDate.getHours();   // 시간
var minutes = objDate.getMinutes(); // 분
var seconds = objDate.getSeconds(); // 초
var millseconds = objDate.getMilliseconds();  // 밀리초
```

- 현재 일자 출력

```
// Date클래스의 인스턴스 생성
var objDate = new Date();
// 년 구하기
var year = objDate.getFullYear();
// 월 구하기
var month = objDate.getMonth();
// 일 구하기
var date = objDate.getDate();
// 요일 구하기
var day = objDate.getDay();
```

14. Date 클래스

- 예제1. 시계 만들기

```
function clock() {
    // 현재 시간 출력
    function showTime() {
        var objDate = new Date();
        var hours = objDate.getHours();
        var minutes = objDate.getMinutes();
        var seconds = objDate.getSeconds();

        var time = hours + ":" + minutes + ":" + seconds;
        $clock.html(time);
    }

    // 시작하자마 시간 출력
    showTime();

    // 0.5초마다 시간 출력
    setInterval(function() {

        showTime();

    }, 500)
}
```

15. Array 클래스

- 배열을 만드는 방법

```
// 리터럴 방식
var users = ["user1", "user2", "user3", "uer4"];

// 배열 클래스 방식
var users = new Array('user1', 'user2', 'user3', 'uer4');
```

- 배열 개수 : `alert(users.length);`
- 특정 위치의 배열 요소 접근하기 : `var menuItem = users[0];`
- 배열을 문자열로 만들기 : `var result = users.join("-");`
- 문자열을 배열로 만들기 : `var arrUser = result.split('-');`
- 특정 위치에 배열 요소 추가하기
  - 배열 마지막에 추가 : `users.push("new");`
  - 배열 첫번째 위치에 추가 : `users.unshift("new");`
  - 배열 n번째 위치에 추가 : `users.splice(2, 0, "new");` // start, deleteCount, 추가요소
- 배열의 인덱스 검색 : indexOf

15. Array 클래스

- 특정 위치의 배열 요소 삭제하기
  - 첫번째 요소 삭제 : `users.shift();`
  - 마지막 요소 삭제 : `users.pop();`
  - n번째 요소 삭제 : `users.splice(2, 1);`
- 문자열 정렬
  - 오름차순 : `users.sort();`
  - 내림차순 :

```
users.sort(function(a, b) {
    return b > a;
});
```

  - 숫자 오름차순 :

```
var aryData = [5, 2, 8, 9, 3, 6, 4, 1, 77];
aryData.sort(function(a, b) {
    return a-b;
});
```

- 예제1. 배열의 총합 구하기

```
// 배열의 총 합 구하기
var result = 0;
for (var i = 0; i < aryData.length; i++) {
    result += aryData[i];
}

// 총합 출력
alert("총합 = " + result);
```

#1. 그외

- use strict : 엄격 모드 (경고가 더 많고 더 명확하게 작성해야 함)
- 세미콜론 : 일반적인 문은 세미콜론으로 닫음. 블록인 경우는 예외

#2. 예외 처리

```
function getMonthName (mo) {
    mo = mo-1;
    var months = ["Jan","Feb","Mar","Apr","May","Jun","Jul","Aug","Sep","Oct","Nov","Dec"];
    if (months[mo] != null) {
        return months[mo];
    } else {
        throw "InvalidMonthNo";
    }
}

try {
    monthName = getMonthName(-1);
}
catch (e) {
    monthName = "unknown";
    console.log(e);
} finally {
    console.log('완료');
}
```

#3. JSON

- 자바스크립트 객체 표기법 – 데이터를 저장하는 평범한 텍스트 형식

```
var jsons =
{
    "first": "Jane",
    "last": "Porter",
    "married": true,
    "born": 1890,
    "friends": ["Trazan", "Cheeta"]
}

console.log(jsons);
```

- JSON.stringify : 자바스크립트 값 value를 JSON형식 문자열로 변환

```
function relacer(key, value) {
    if(typeof value === 'number') {
        value = 2 * value;
    }
    return value;
}
console.log(JSON.stringify({foo:1, bar:[2, 8]}, relacer));
```

- JSON.parse : JSON데이터를 자바스크립트 값으로 변환

```
console.log(JSON.parse('{"foo":2,"bar":[4,16]}'));
```