함수(Function)

함수(Function)

- 어떤 목적을 위해 만들어진 코드를 하나로 묶어 반복되는 작업을 효율적으로 관리
- 함수 정의 / 함수 호출 / 함수 인자 / 함수 종료 및 리턴값

함수 정의

- 선언적 함수와 익명 함수, 두개 모두 기능에는 전혀 차이가 없음
- 익명함수는 반드시 함수를 호출하는 코드보다 먼저 작성이 되어야 함 (**함수 호이스팅 X**)

```
<script>
// 선언적 함수(이름을 선언)
function sum () {
        var a = 10;
        var b = 20;
        var c = a + b;
};
</script>
```

```
<script>
// 익명 함수(이름 없는 함수를 변수에 대입)
var sum = function () {
        var a = 10;
        var b = 20;
        var c = a + b;
};
</script>
```

함수(Function)

함수 호출

- 함수 정의만 해서는 절대로 실행되지 않음
- 함수 정의되어 메모리에 담겨있을뿐, 실제로 실행되는 것이 아님
- 정의된 함수를 수행하려면 반드시 함수를 호출해야 함

```
<script>
var sum = function () {
            var a = 10;
            var b = 20;
            var c = a + b;
};
// 함수 호출
sum();
</script>
```

함수(Function)

함수 인자

- 함수를 선언할 때 () 안에 들어가는 여러가지 변수를 재료라고 생각하면 됨

```
<script>
var sum = function (a, b) {
            var c = a + b;
            console.log(c);
};
sum(10, 20);
</script>
```

함수(Function)

함수 종료 및 리턴값

```
<script>
var sum = function (a, b) {
            var c = a + b;
            return c; // 리턴값이 있는 함수
            return false; // 리턴값이 없는 함수
};
sum(10, 20);
</script>
```

함수(Function)

즉시 실행 함수

- 함수를 정의함과 동시에 바로 실행하는 함수 (익명 함수를 응용한 형태)
- 함수가 선언되자마자 실행되게 만든 즉시 실행 함수의 경우, 같은 함수를 다시 호출할 수 없다.
- 최초 한 번의 실행만을 필요로 하는 초기화 코드 부분 등에 사용할 수 있다.

```
<script>
(function (name) {
        console.log('This is the immediate function -> ' + name);
})('foo');
</script>
```

함수(Function)

즉시 실행 함수

- jQuery 나 기타 프레임워크 소스들에서 사용됨
- 즉시 실행 함수를 사용하는 이유는 자바스크립트의 변수 유효 범위 특성 때문
- 기본적으로 자바스크립트는 변수를 선언할 경우 프로그램 전체에서 접근할 수 있는 전역 유효 범위를 가지게 된다. 그러나 함수 내부에서 정의된 매개변수와 변수들은 함수 코드 내부에서만 유효할 뿐 함수 밖에서는 유효하지 않다. (var 문을 사용해서 정의해야 함. 그렇지 않으면 함수 내의 변수라도 전역 유효 범위를 갖게 됨)
- 달리 말하면, 함수 외부의 코드에서 함수 내부의 변수를 액세스하는 게 불가능 함
- 전역 네임스페이스를 더럽히지 않으므로, 충돌 같은 문제를 방지할 수 있음

```
<script>
(function (win, undefined) {
})(window);
</script>
```

함수(Function)

함수 주요 메서드

- apply - 해당 함수가 지칭하는 this 를,
  인자로 들어온 object 로 변경하게 함.
  : https://www.w3schools.com/js/js_function_apply.asp
- call - 해당 함수가 지칭하는 this 를,
  인자로 들어온 object 로 변경하게 함.
  : https://www.w3schools.com/js/js_function_call.asp

```
<script>
var person = {
        name : 'John',
        fullName : function () {
                return this.name;
        }
};
var myObject = {
        name : 'you'
};
person.fullName.call(myObject); // return 'you'
console.dir(Function.prototype); //메서드종류확인
</script>
```

타이머

타이머

- 지정된 시간마다 함수를 호출할 수 있는 window 객체 메서드
- setTimeout / clearTimeout / setInterval / clearInterval

setTimeout

- setTimeout('실행할 함수', 대기시간) // 대기 시간이 지난 후 1번만 코드 실행

clearTimeout

- setTimeout 을 해제

```
<script>
var gallery = function () {
        console.log('다음 사진');
};
var clearId = setTimeout(gallery, 1000);
</script>
```

```
<script>
// setTimeout 을 해제
clearTimeout(clearId);
</script>
```

타이머

setInterval

- setInterval(‘실행할 함수’, 대기시간) // 대기 시간이 지난 후 코드 반복 실행

clearInterval

- setInterval 을 해제

```
<script>
var count = 1;
var gallery = function () {
        console.log(‘다음 사진’);
        if (count == 5) {
                clearInterval(clearId);
        }
        count += 1;
};
var clearId = setInterval(gallery, 1000);
</script>
```

Thank you.

HIVELAB

Question.