INTERPARK

2017 NOV.

{ .JS }

JavaScript Lab

전지호

creator

{ Content }

자바스크립트 맛보기

JavaScript Lab 1

01. use strict 엄격모드

> 파일 전체 또는 특정함수를 엄격한 모드로 실행

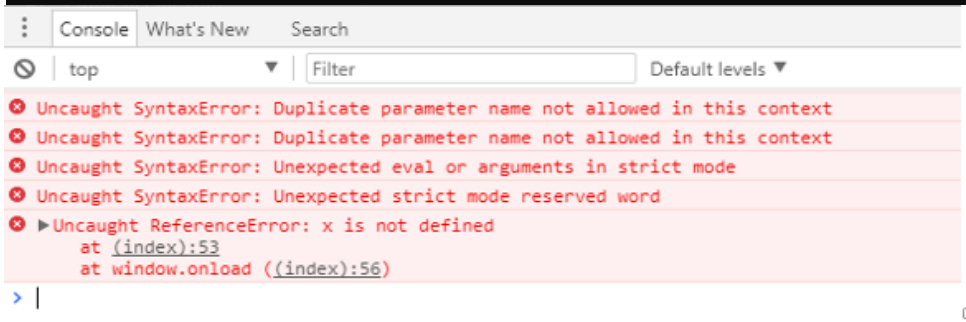
> 이를 통해 의도치 않은 실수에 대한 예외 발생

```
(function (){

    "use strict";

    test = "not definded";
    alert(test);
    //Duplicate parameter name not
    //allowed in this context
    function testFunc(p1, p1) {};
    //Duplicate parameter name not
    //allowed in this context
    var eval = 3.14;
    //Unexpected eval or arguments
    //in strict mode
    eval ("var x = 2");
    //Unexpected eval or arguments
    //in strict mode
    alert(x);
    //x is undefined
    var interface = "test";

})();
```



01. use strict 엄격모드

> strict and non strict file 병합 금지
bundling시 병합될 가능성 존재 -> bundling 분리 필요

> IIFE(Immediately Invoked Function Expression)으로 wrapping 후 병합

```
1
2   //파일전체 xxx.js 파일 최상단에 정의
3   "use strict";
4
5   //특정함수에 적용
6   //Non-strict code...
7
8   (function(){
9   "use strict";
10  //Define your library strictly
11  })();
12
13  //Non-strict code...
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
```

02. 암묵적 타입 변경

> 3 + true = 4

> 1000 + “5000” = “10005000”

> “17” * 3 = 51

> if, ||, && 연산자는 대부분의 자바스크립트 값 true 처리

> 0, -0, "", NAN, null, undefined 6개는 false 처리

> if (typeof x == "undefined") document.write(" undefined ");

03. ===

> "==" 값 체크, "===" 자료형까지 체크

> form.day.value == today.getdate() (형변환 발생)

> +form.day.value === today.getdate() (명시적 형변환)

> 명시적 형변환으로 가독성증가, 예외제거

04. ; (세미콜론)

> Javascript에선 문장을 종료하는 세미콜론 생략가능

> 하지만, 세미콜론 생략의 위험성을 반드시 인지하고 올바르게 사용해야함.

```
1   //1. 세미콜론은 } 토큰 전이나 줄의 마지막
2   //또는 프로그램의 마지막 전에만 추론되어 삽입
3
4   function Point(x,y){
5       this.x = x || 0
6       this.y = y || 1
7       return this.x * this.y
8   }
9
10  function area(r){ r= +r; return Math.PI * r * r}
11
12  function area(r){ r= +r return Math.PI * r * r}
13
14  //2. 세미콜론은 다음 토큰이 파싱될 수 없을때(오류발생)
15  //에만 추론되어 삽입
16
17  function Sum(a,b) { return a + b }
18
19  //a = Math.abs(Sum(1,-3));
20
21  //CASE 1 => ?
22  a = Math.abs
23  (Sum(1,-3));
24
25  //CASE 2 => a = b Sum(1,-3)
26  a = b
27  Sum(1,-3)
28
29
30
31
32
```

04. ; (세미콜론)

> 선언문이
([+ - / 로 시작할 때 주의

```
//3. 선언문이 ( [ + - / 로 시작할 때 주의

//CASE 1
a =
b["r", "g", "b"].forEach(function(key){
background[key] = foreground[key] / 2
});

/*
a = b["r", "g", "b"].forEach(function(key){
background[key] = foreground[key] / 2
});
*/

//CASE 2
a = b
/Error/i.test(str) && fail();

/*
나눗셈 연산자;;;
a = b / Error/i.test(str) && fail();
*/
```

04. ; (세미콜론)

> 번들링 시 주의

```
//4. 스크립트 번들링 시 주의

//파일1.js
(function(){
  //...
})()
//파일2.js
(function(){
  //...
})()
//번들링 후 나 파싱 후
(function(){
  //...
})()
(function(){
  //...
})()

//방어코드 삽입필요
//파일1.js
;(function(){
  //...
})()
//파일2.js
;(function(){
  //...
})()
```

04. ; (세미콜론)

> return, throw, break, continue ++, -- 뒤에 새로운 행 입력 시 주의

```
//5. return, throw, break, continue, ++, --
//바로 뒤에 새로운 행 입력 시 주의

//CASE 1 : return {};
return
{};

//파싱 후
return;
{}
;

//CASE2 : ++ 접두어? 접미어?
a
++
b

//파싱 후
a;
++b;
```

04. ; (세미콜론)

> 세미콜론은 for반복문의 구분자 나 빈 선언문으로 절대 삽입 안됨

```
//6. 세미콜론은 for반복문의 구분자 나
//빈 선언문으로 절대 삽입 안됨

for(var i = 0; total =1
  i < n
  i++
){

  total *= 1

}
```