

# 디지털컨버전 스 기반 UXUI Front 전문 개발자 양성과정

강사 - Innova Lee(이상훈)

[gcccompil3r@gmail.com](mailto:gcccompil3r@gmail.com)

학생 - JungHyun

LEE(이정현)

akdl911215@naver.com

## 자바스크립트 - 동적타이핑

자바스크립트는 **loosely typed**(느슨한 타입) 언어, 혹은 **Dynamic**(동적) 언어입니다. 그 말은, 변수의 타입을 미리 선언할 필요가 없다는 뜻입니다. 타입은 프로그램이 처리되는 과정에서 자동으로 파악될 것입니다. 또한 그 말은 같은 변수에 여러 타입의 값을 넣을 수 있다는 점입니다.

### typeof 연산자

**undefined** : 변수가 정의되지 않거나 값이 없을 때

**number** : 데이터 타입이 수일 때

**string** : 데이터 타입이 문자열일 때

**boolean** : 데이터 타입이 불리언일 때

**object** : 데이터 타입이 함수, 배열 등 객체일 때

**function** : 변수의 값이 함수일 때

**symbol** : 데이터 타입이 심볼일 때

### = () => 함수

위와 같은 화살표 함수는 무명 함수를 생성하는 방법 중의 하나입니다. 원래는 기본적으로 메모리 할당 작업을 해주어야 하는데 **= () =>** 동적으로 알아서 할당해라는 뜻을 내포하고 있습니다. 결론적으로 의미는 **function ConstTest() {** 와 동일합니다. 차이가 있다면 명시적으로 할당하는 방식이 아니라 알아서 동적 할당 합니다.

화살표 함수의 형태상 특징은 다음과 같습니다.

1. 함수 내용이 한줄인 경우 함수내용을 감싸는 **{}**를 사용하지 않아도 됩니다.
2. **{}**가 없는 경우 해당 함수의 실행결과를 자동으로 이전합니다.
3. 함수 내용이 한줄 이상인 경우 **return** 을 사용해서 결과를 리턴합니다.
4. 파라미터가 한 개인 경우 파라미터를 감싸는 **()**를 생략할 수 있습니다.

```
// 함수 표현
filteredArray = myArray.filter(function(element){
  return element > 2;
});

// 화살표 함수
filteredArray = myArray.filter(element => element > 2);
```

## 동등 연산자 ( == )

동등 연산자는 두 피연산자의 자료형이 같지 않은 경우 같아지도록 변환하고, 엄격 비교를 수행합니다. 피연산자가 모두 객체라면, **JavaScript**는 내부 참조를 보고, 둘 다 메모리의 같은 객체를 바라보고 있는지 판별합니다.

### 구문

```
x == y
```

### 예제

```
1 == 1           // true
"1" == 1         // true
1 == '1'         // true
0 == false       // true
0 == null        // false

0 == undefined  // false
null == undefined // true
```

## 부등 연산자 ( != )

부등 연산자는 두 피연산자가 같지 않은 경우 참을 반환합니다. 피연산자의 자료형이 일치하지 않는 경우 적절한 자료형으로의 변환을 시도합니다. 피연산자가 모두 객체라면, **JavaScript**는 내부 참조를 보고, 서로 메모리의 다른 객체를 바라보고 있는지 판별합니다.

### 구문

```
x != y
```

### 예제

```
1 != 2           // true
1 != "1"         // false
1 != '1'         // false
1 != true        // false
0 != false       // false
```

## 일치 연산자 ( === )

일치 연산자는 자료형 변환 없이 두 연산자가 엄격히 같은지 판별합니다.

### 구문

```
x === y
```

### 예제

```
3 === 3 // true
3 === '3' // false
```

## 불일치 연산자 ( !== )

일치 연산자는 두 연산자가 같지 않거나, 같은 자료형이 아닐 때 참을 반환 합니다.

### 구문

```
x !== y
```

### 예제

```
3 !== '3' // true
4 !== 3 // true
```

## 초과 연산자 ( > )

초과 연산자는 왼쪽 피연산자가 오른쪽 피연산자보다 큰 경우 참을 반환합니다.

### 구문

```
x > y
```

### 예제

```
4 > 3 // true
```

## 이상 연산자 ( $\geq$ )

이상 연산자는 왼쪽 피연산자가 오른쪽 피연산자보다 크거나 같으면 참을 반환합니다.

### 구문

```
x >= y
```

### 예제

```
4 >= 3 // true
3 >= 3 // true
```

## 미만 연산자 ( $<$ )

미만 연산자는 왼쪽 피연산자가 오른쪽 피연산자보다 작은 경우 참을 반환합니다.

### 구문

```
x < y
```

### 예제

```
3 < 4 // true
```

## 이하 연산자 ( $\leq$ )

이하 연산자는 왼쪽 피연산자가 오른쪽 피연산자보다 작거나 같으면 참을 반환합니다.

### 구문

```
x <= y
```

### 예제

```
3 <= 4 // true
```

console.log 에 대해서

console 객체에는 log 메소드 말고도, 다양한 메소드들이 많이 존재합니다. log, dir, count, time, timeEnd 등등 있다.

## log

```
1  var a = 1;
2  var b = 'hello';
3  var c = true;
4  console.log(a); // 하나만 로그
5  console.log(a, b, c); // 여러 개를 동시에 로그
6  console.log('%d는 숫자 %s는 문자열', a, b); // c 언어처럼 로그
```

마지막 용법처럼 C언어의 printf처럼 %d 와 %s를 뒤에 입력한 인자로 치환이 가능합니다.

log와 같은 기능을 가진 warn, info, error도 사용가능합니다

```
1  console.log('기본');
2  console.info('정보');
3  console.warn('경고');
4  console.error('에러');
```

## 호이스팅 (Hoisting)

호이스팅은 모든 변수가 프로그램 시작시 함수 안에 있는 선언들을 모두 끌어올려서 해당 함수 유효 범위의 최선두에 선언하는 형식으로 동작하는 것을 의미합니다. 좀 더 쉽게 말하자면 **var**는 변수 선언과 값의 할당이 통합됩니다. 반면 **let**은 변수 선언과 값의 할당이 분리되어 있습니다.

그러다보니 **var**는 변수 생성시 이름이 같은게 있으면 **Mangling**이라는 방식을 통해서 변수의 이름을 바꿔버립니다.

반면 **let**은 선언과 할당이 분리되어 있다보니 왜 같은것을 선언하냐면서 문법 오류가 나는 것이다.

함수 안에 있는 선언들을 모두 끌어올려서 해당 함수 유효 범위의 최상단에 선언하는 것을 말합니다