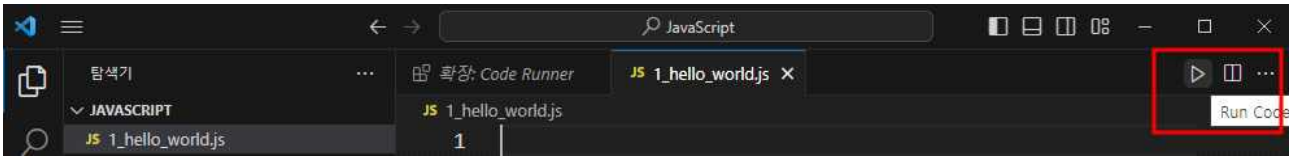
JavaScript

1장. 자바스크립트에 필요한 IDE 설치와 JavaScript 기초

1. Visual Studio Code 설치

① plug-in 설치

- code runner(Control +Alt + L)
- live server



2. 프로젝트 폴더 생성 후 깃허브와 연결

3. 기본 출력과 주석 처리하기

▶ 1_hello_world.js

```
// 코멘트를 작성하는 첫번째 방법입니다.
console.log('Hello World')

/**
 * 이렇게 하면 여러줄에 걸쳐서
 * 코멘트를 작성할 수
 * 있습니다.
 * */

console.log('Hello', 'World')
```

4. 변수의 사용 및 선언과 할당

① 변수선언

- let, const

② 선언과 할당

- 선언 : 변수를 선언하는 것
- 할당 : 선언된 변수에 값을 넣는 것

③ 변수 naming 규칙

- 일반적으로 영어와 숫자를 사용
- 특수기호는 달러($)와 언더스코어(_) 만 사용 가능

- 숫자로 변수명을 시작할 수 없다.
- 키워드를 변수명으로 사용할 수 없다.

ex> var const = '';

④ Naming Convention
- camel case : 첫문자 소문자 뒤는 단어 시작마다 첫글자를 대문자로(namingConvention)
- snake case : 전체 소문자를 사용하며 단어별로 _ 로 연결(naming_convention)
- pascal case : 매 단어 첫자를 대문자로 사용(Naming Convention)

※ 자바스크립트를 비롯한 대부분 개발툴에서 camel case를 사용함.

▶ 2_varialbe.js

```
/**
 * 변수의 사용
 * 1. var : 더 이상 쓰지 않는다.
 * 2. let
 * 3. const
 */

var name = '김형민'
console.log(name)
var age = 22
console.log(age)

let ive = '아이브'
console.log(ive)

ive = '안유진'
console.log(ive)

let girlGroup
// undefined 출력(선언되었으나 할당되지 않았다.)
console.log(girlGroup)

const newJeans = '뉴진스'
console.log(newJeans)
newJeans = '뉴 진스'
```

```
김형민
22
아이브
안유진
undefined
뉴진스
k:₩lect₩$_Front_End₩JavaScript₩1_basics₩2_variable.js:25
newJeans = '뉴 진스'
           ^

TypeError: Assignment to constant variable.
k:₩lect₩$_Front_End₩JavaScript₩1_basics₩2_variable.js:21
newJeans = '뉴 진스'
           ^

TypeError: Assignment to constant variable.
```

5. Data Type

여섯개의 Primitive Type과 한 개의 Object 타입이 있다.

1) Number : 숫자
2) String : 문자열
3) Boolean : 논리값(true, false)
4) undefined
5) null
6) Symbol
7) Object : 객체(Function, Array, Object)

※ 각 변수의 type을 확인하는 법 : typeof

① Number Type

▶ 3_1_number_test.js

```
const age = 32
const tempature = -10
const pi = 3.14
console.log(typeof age)
console.log(typeof tempature)
console.log(typeof pi)
```

```
console.log('--------------------')

const infinity = Infinity  // 양의 무한대 값
const nInfinity = -Infinity // 음의 무한대 값
console.log(typeof infinity)
console.log(typeof nInfinity)
```

② String Type

```
※※ Template Literal

1) Escape Character
 ⓐ new Line : ₩n
 ⓑ tab : ₩t
 ⓒ ₩(Back_Slash)는 두번 입력해서 사용

2) 백틱(`) 사용
 ⓐ 여러 줄에 걸쳐 출력할 때 매우 유용하다.

3) Template Literal 내에서의 변수 사용방법
 ⓐ 백틱 내에 변수명을 ${변수명}으로 감싸 처리한다.
```

▶ 3_2_string_test.js

```
const codeTest = '코드테스트'
console.log(codeTest)

// 문자열 안에 작은 따옴표와 큰 따옴표 처리하기
const ive = "'아이브' 장원영"
console.log(ive)

// Escape Character
const iveYuJin = '아이브 ₩n안유진'
console.log(iveYuJin)

const iveWonYoung = '아이브 ₩t장원영'
console.log(iveWonYoung)

const iveGaEul = '아이브 ₩' 가을'
console.log(iveGaEul)
```

```
console.log('--------------------')

// Template Literal에서 백틱(`)으로 변수 사용
const groupName = '아이브'
console.log(groupName + '안유진')
console.log(`${groupName} 안유진`)
```

③ Boolean Type

▶ 3_3_boolean_test.js

```
const isTrue = true
const isFalse = false

console.log('True : ' + isTrue)
console.log('False : ' + isFalse)
```

④ undefined Type : 사용자가 직접 값을 초기화하지 않았을 때 지정되는 값.
(직접 undefined로 값을 초기화 하는 것은 ○○다)

▶ 3_4_undefined_test.js

```
let noInit = undefined
let noInit2

console.log(noInit)
console.log(noInit2)

console.log(typeof noInit2)
```

⑤ null Type : undefined와 마찬가지로 값이 없다는 뜻이나, 자바스크립트에서는 개발자가 명시적으로 없는 값으로 초기화할 때 사용된다.

☞ 영국의 컴퓨터 과학자 Tony Hoare가 만든 개념
- 프로그래밍 언어인 C언어에서는 생성된 메모리의 주소를 '포인터'라는 것이 가리키게 되어 포인터를 통해 데이터를 가져올 수 있게 된다. 그러나, 이때 포인터가 아무것도 가리키고 있지 않다는 것을 나타내기 위해 NULL이라는 개념을 사용하게 되는 것
- null 참조를 만든 것은 10억 달러의 실수 → 2009년 소프트웨어 컨퍼런스(null 만든 본인 이야기)

▶ 3_5_null_test.js

```
let init = null
console.log(init)
console.log(typeof init)

// 약간의 버그개념으로 이해해야 함.

null
object
```

⑥ Symbol Type
- 유일무이한 값을 선언할 때 사용
- 다른 Primitive Type과는 다르게 Symbol 함수를 호출해서 만든다.

▶ 3_6_Symbol_test.js

```
const test1 = '1'
const test2 = '1'
console.log(1 == '1')
console.log(test1 == test2)
console.log(test1 === test2)

console.log('-------------------------')

const symbol1 = Symbol('1')
const symbol2 = Symbol('1')
console.log(symbol1 === symbol2)
console.log(symbol1 === test1)
```

⑦ Object Type
- Map 자료구조 처럼 키 : 값의 쌍으로 이루어져 있다.

▶ 3_7_object_test.js

```
const dictionary = {
  red : '빨강색',
  orange : '주황색',
  yellow : '노랑색',
}

console.log(dictionary)
```

```
console.log(dictionary.red)
console.log(dictionary['yellow'])
console.log(typeof dictionary)

------------------------------------------------------------------

{ red: '빨강색', orange: '주황색', yellow: '노랑색' }
빨강색
노랑색
object
```

⑧ Array Type
- 값을 리스트로 나열 할 수 있는 타입

▶ 3_7_array_test.js

```
const iveMemberArray = [
  '안유진',
  '장원영',
  '레이',
  '이서',
  '가을',
  '리즈',
]

console.log(iveMemberArray)

/**
 * Array 안에는 index(0부터 시작) 가 들어있어서
 * index 값을 이용해 다양한 처리를 할 수 있다.
 *  */

console.log(iveMemberArray[5])
console.log(iveMemberArray[3])

// 배열에 값을 할당하는 방법
iveMemberArray[0] = '아이브'
console.log(iveMemberArray)
console.log(typeof iveMemberArray)
```

▷ typing 종류

ⓐ static typing : 변수 선언 시 변수에 타입을 명시한다
ⓑ dynamic typing : 변수 선언시 type을 명시하지 않아도 변수에 값이 할당될 때 값에 의해 타입을 추출한다.

6. 호이스팅(Hoisting)

- 모든 선언된 변수가 최상단으로 이동되는 것처럼 느껴지는 현상
- var 키워드는 변수 호이스팅을 막지 못한다.
- 변수 선언 후 할당이라는 안정적 개발환경을 만족시키기 위해서는 const 및 let 키워드 사용을 권장.

▶ 4_hoisting.js

```
console.log(name)
// ReferenceError 오류가 발생하지 않음!!
var name = '김형민'
console.log(name)

// let과 const도 hoisting 처리 된다.

console.log(yuJin)
//ReferenceError: Cannot access 'yuJin' before initialization
let yuJin = '안유진'
```

7. 연산자(Operators)

1) 산술연산자

▶ 5_operators.js

```
// 1) 산술연산자
console.log(10 + 10)
console.log(10 - 10)
console.log(10 / 10)
console.log(10 * 10)
console.log(10 % 10)  // 나머지를 구하는 연산자
console.log(10 % 3)
console.log(10 * (10+10))

console.log('---------------------------')
```

```
// 2) 증감연산자
let number = 1
number++
console.log(number)

number--
console.log(number)

console.log('---------------------------')

let result = 1
console.log(result)
result = number++
// ① result = number
// ② number = number + 1
console.log(result, number) // 1 2 출력

result = number --
// ① result = number
// ② number = number - 1
console.log(result, number) // 2 1 출력

result = ++ number
// ① number = number + 1
// ② result = number
// 현대에서 거의 사용하지 않음.(뒤통수 맞을 수 있음.)
console.log(result, number)

console.log('---------------------------')
// 숫자가 아닌 타입에 +, - 사용하기
let sample = '99'
console.log(+sample)
console.log(typeof +sample)

console.log(sample)
console.log(typeof sample)

sample = true
console.log(+sample)
console.log(typeof +sample)
```

```
// 실제 true = 1로, false = 0으로 표현되기도 한다.

sample = '안유진'
console.log(+sample)
// NaN : Not a Number

console.log('---------------------------')

// 3) 할당연산자(assignment operator)
number = 100
console.log(number)
number += 10
console.log(number)
number -= 10
console.log(number)
number *= 10
console.log(number)
number /= 10
console.log(number)
number %= 10
console.log(number)

console.log('---------------------------')

// 4) 비교연산자
// ① 값의 비교
console.log(5 == 5)
console.log(5 == '5')
console.log(0 == '')
console.log(true == 1)
console.log(false == 0)
console.log(true == '1')

console.log('---------------------------')

// ② 값과 타입을 같이 한꺼번에 비교
console.log(5 === 5)
console.log(5 === '5')

console.log('---------------------------')
```

```
/**
 * ③ 관계연산자
 * - 값의 비교 : > , < , >= , <=
 * - 값과 타입을 같이 한꺼번에 비교 : !==
 */

// ④ 삼항조건연산자
console.log(10 > 0 ? '10이 0보다 크다' : '10이 0보다 작다.')

console.log('---------------------------')

// ⑤ 논리연산자(&& , ||)
console.log(true && true)
console.log(true && false)

console.log(true || true)
console.log(true || false)

console.log(10>1 && 20>2);
console.log(10>1 || 20>2);

/**
 * ⑥ 단축평가(short circuit evaluation) - 실무서 많이 씀
 * - 논리곱(&&) 연산자
 * 피연산자를 왼쪽에서 오른쪽으로 확인하며 하나라도 falsy인 경우, 해당 falsy 값을 반환한다.
 * 처음부터 끝까지 모든 피연산자가 truthy인 경우, 마지막 truthy 값을 반환한다.
 * - 논리합(||) 연산자
 * 피연산자를 왼쪽에서 오른쪽으로 확인하며 하나라도 truthy인 경우, 해당 truthy 값을 반환한다.
 * 처음부터 끝까지 모든 피연산자가 falsy인 경우, 마지막 falsy 값을 반환한다.
 */

console.log('---------------------------')
console.log(true || '아이브')
console.log(false || '아이브')

console.log(false && '아이브')
console.log(true && '아이브')

console.log(true && true && '아이브')
```

```
console.log(true && false && '아이브')

// ⑦ 지수연산자
console.log(2 ** 2)
console.log(10 ** 3)

// ⑧ null 연산자 : 좌측값이 null 혹은 undefined 이면 우측값 반환
console.log('--------------------------')
let name
console.log(name)

name = name ?? '김형민'
console.log(name)

name = name && '아이브'
console.log(name)
```