Javascript

Ⅰ. 개발환경 설정과 시작하기 전에

1. Visual Studio Code 설치

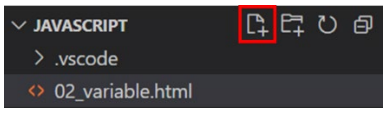
```
https://code.visualstudio.com/
```

2. Visual Studio Code Extensions 설치

- HTML Snippets
- HTML CSS Support
- JavaScript (ES6) code snippets
- Live Server
- Open In Default Browser

2. 기본 코딩 작성법

1) HTML5 형식의 문서 생성



① 왼쪽 그림과 같이 탐색기 패널의 폴더 이름 오른쪽의 "새파일" 아이콘을 선택한 후 확장자가 html인 새파일을 생성

② 편집창에 느낌표(!) 하나를 입력하고 엔터

③ <html lang="en">을 <html lang="ko">로 수정

2) 자바스크립트 기본 형식

① HTML문서안에 삽입

```
<head>
    <script ~~language="javascript"~~ src="js파일의 경로 및 파일명">
        자바스크립트 코드
    </script>
</head>
```

② 외부 파일 로드

```
<head>
    <script src="js파일의 경로 및 파일명"></script>
```

```
</head>
```

③ 태그에 직접 작업

```
<input type="button" onclick="console.log('Hello. Javascript!!!');">
```

Ⅱ. Javascript Basic

1. 변수 (Variable)

어렵게 생각하지 말자. 변수는 로컬 컴퓨터의 메모리 상에 작성된 Javascript가 실행되는 동안 임시로 데이터를 저장하고 있는 이름이 주어진 메모리 공간이라고 생각하면 된다. 이때 저장된 데이터는 항상 마지막에 저장된 데이터만 유지된다.

1) 변수의 선언

기본적으로 Javascript에서는 변수를 선언하지 않고 바로 사용도 가능하다. 다만, 함수와 함께 사용이 될 경우 변수를 선언한 것과 선언하지 않고 사용할 경우엔 차이가 발생한다.

```
var 변수명; //변수를 선언만 한 경우
var 변수명2=data; //변수의 선언과 초기값(data)를 함께 지정하여 선언한 경우
```

2) 전역변수

- HTML문서가 브라우저에 로딩되어 있는 동안 변수의 데이터가 유지됨.
- HTML문서 내 어디서든 변수를 사용 가능함.

3) 지역변수

- 사용자 정의 함수 내에서만 데이터가 유지되며 함수가 종료되면 변수의 값을 상실함.
- 함수 바깥에선 지역변수를 사용 할 수 없음

.

2. 데이터 타입

데이터 타입이란 자료(data)의 형식을 의미하는 것으로, 문자, 문자열, 숫자, 참(true)과 거짓(false) 등이 대표적인 데이터 타입이다.
다만, Javascript에서는 변수 선언에 있어 데이터 타입을 지정하지 않으며, 변수에 서로 다른 데이터 타입의 데이터를 저장 할 수도 있다.

1) 개발에 있어 알아두면 도움이 되는 데이터 타입

- null : 일반적으로 선언이 되지 않은 변수의 반환 값.

```
<script>
    console.log(test); //null 로  error
</script>
```

위 test는 변수의 선언이 되지 않아 null로 출력의 대상이 되지 않아 error.

- undefined : 변수의 선언은 되었으나 값이 할당되지 않은 경우.

```
<script>
    var test;
    console.log(test); //null 로  error
</script>
```

위 test는 변수로 선언이 되었으나 값이 할당되지 않아 undefined를 출력함.

3. 연산자

1) 산술연산자

```
+, -, *, /, %
```

```
<script>
        console.log(5+3); //덧셈
        console.log(5-3); //뺄셈
        console.log(5*3); //곱셈
        console.log(5/3); //나눗셈
        console.log(5%3); //나머지
    </script>
```

[질문]

```
<script>
    console.log(5*4/2); //result : ?
    console.log(5/4*2); // result : ?
    console.log(5/2%2); // result : ?
    console.log(5%3*2); // result : ?
</script>
```

2) 산술 연산자(증감 연산자)

```
++, --
```

[질문]

```
<script>
    var num=5;
    console.log(num++); //result : 출력(?), num(?)
    console.log(++num); //result : 출력(?), num(?)
    console.log(num--);   //result : 출력(?), num(?)
    console.log(--num);   //result : 출력(?), num(?)
</script>
```

3) 할당 연산자

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

[질문]

```
<script>
    var num;
    console.log(num=5);          //result : 출력(?), num(?)
    console.log(num+=5);         //result : 출력(?), num(?)
    console.log(num-=2);         //result : 출력(?), num(?)
    console.log(num*=3);         //result : 출력(?), num(?)
    console.log(num/=2);         //result : 출력(?), num(?)
    console.log(num%=5);         //result : 출력(?), num(?)
</script>
```

4) (문자열)연결 연산자 : +

```
<script>
    var num=10;
     //연결 연산자는 문자열과 숫자를 순서대로 나열한다.
    console.log("홍길동은 " + num + "살이다.");
```

```
    console.log("홍길동은 " + num + 5 + "살이다.");
    console.log(num + 10 + "개의 사과");
    console.log("홍길동은 " + (num + 5) + "살이다.");
</script>
```

[질문]

```
<script>
    var num=10;
     //연결 연산자는 문자(열)과 숫자를 순서대로 나열한다.
    console.log("홍길동은 " + num + "살이다.");
    console.log("홍길동은 " + num + 5 + "살이다."); //result : ?

    console.log(num + 10 + "개의 사과"); //result : ?
    console.log("홍길동은 " + (num + 5) + "살이다."); //result : ?
</script>
```

5) 비교 연산자

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

```
<script>
    var num=10;
    console.log(num==10);       //result : ?
    console.log(num!=10);       //result : ?
    console.log(num===10);      //result : ?
    console.log(num==="10");    //result : ?
    console.log(num!=="10");    //result : ?
</script>
```

6) 논리 연산자

||, &&, !

```
<script>
    var num=10;
```

```
    var sum; //선언만 하고 값을 할당하지 않음(undefined)
    console.log(true || false || false); //하나 이상이 true 이면 결과값은 true
    console.log(true && true && false); //모두 true 여야만 true
    console.log(true && !(true && false)); //값을 반대로
</script>
```

[질문]

```
    <script>
        var num=10;
        var sum; //선언만 하고 값을 할당하지 않음(undefined)
        console.log(true || false || false); //하나 이상이 true 이면 결과값은 true
        console.log(true && true && false); //모두 true 여야만 true
        console.log( true && !(true && false) ); //값을 반대로
        console.log(true || false && true); //result : ?
        console.log(true && false || true); //result : ?
        console.log(sum); //result : ?
        console.log( !sum ); //result : ?
    </script>
```

4. 함수

특별한 목적의 작업을 수행하도록 설계된 독립적인 블록으로 그 내부에 변수를 선언할 수 있으며, 매개변수로 값을 전달 할 수 있다.
또한 함수를 호출(실행)한 곳으로 특정 값(data)를 반환 할 수 있다.

1) 함수 선언문

```
function 함수명 ( [매개변수1, 매개변수 2 ... ( {
    실행문;
    [ return 값; ]
}
```

```
const 함수명 = function ( [매개변수1, 매개변수 2 ... ( {
```

```
    실행문;
    [ return  값; ]
}
```

```
<script>
    function cmd_1() {
        console.log("함수 선언 1");
    }
    const cmd_2 = function () {
        var sum;
        var num=5;
        sum = num + 7;
        console.log("num : " + num + ", sum : " + sum);
    }
    cmd_1(); //함수 cmd-1 호출
    cmd_2(); //함수 cmd-1 호출
    </script>
```

2) 지역변수와 전역변수

① 전역변수와 지역변수

전역변수는 함수 바깥쪽에서 선언된 변수로 모든 함수에서 사용이 가능하며, 변수의 데이터는 최종적으로 저장된 데이터가 문서를 벗어나기 이전가지 유지된다.

지역변수는 선언된 함수 안에서만 사용이 가능하여 변수의 생명은 함수 시작부터 종료 까지다.

함수 내에서 선언되지 않은 변수는 전역변수로 보면 된다.

```
<script>
    var sum=0;
    const cmd_1 = function() {
        var num=5;
        sum = num + 7;
        console.log("num : " + num + ", sum : " + sum);
```

```
    }
    const cmd_2 = function() {
        console.log("sum : " + sum); //sum(?)
    }
    cmd_1(); //함수 cmd-1 호출
    cmd_2(); //함수 cmd-1 호출
    //console.log("num : " + num + ", sum : " + sum); //num(?), sum(?)
    </script>
```

② 함수내에서 선언되지 않은 변수 : 전역변수

```
<script>
    //var sum=0;
    const cmd_1 = function() {
        var num=5;
        sum = num + 7;
        console.log("num : " + num + ", sum : " + sum);
    }
    const cmd_2 = function() {
        console.log("sum : " + sum); //sum(?)
    }
    cmd_1(); //함수 cmd_1 호출
    cmd_2(); //함수 cmd_1 호출
</script>
```

③ 전역변수와 동일한 이름의 지역변수

```
    <script>
        var sum=0;
        const cmd_1 = function() {
            var num=5;
            var sum=10;
            sum = num + 7;
```

```
            //parent.sum = num + 7;
            console.log("num : " + num + ", sum : " + sum);
        }
        const cmd_2 = function() {
            console.log("sum : " + sum); //sum(?)
        }
        cmd_1(); //함수 cmd_1 호출
        cmd_2(); //함수 cmd_1 호출
    </script>
```

3) 매개변수

매개변수는 함수의 외부에서 보내오는 값(데이터)을 받을 수 있는 함수내에서 사용 되는 임시 메모리 공간이며, 함수 내에서는 지역변수처럼 사용이 가능하다.

```
<script>
    var sum=0;
    const cmd_1 = function(n) {
        sum = n + 5;
        cmd_2(5); //함수 cmd-1 호출
    }
    const cmd_2 = function(num) {
        sum += num;
        console.log("num : " + num + ", sum : " + sum);
    }
    cmd_1(10); //함수 cmd-1 호출
</script>
```

4) 함수 값의 반환

함수내에서 만들어진 값을 함수를 호출한 곳으로 반환하여 전역변수의 사용을 줄일 수 있음.

```
<script>
    var cmd1;
    const cmd_1 = function(n) {
        var sum;
        sum = n + 5;
        return sum;
    }
     const cmd_2 = function(num) {
        var sum=10;
        sum += num;
        console.log("num : " + num + ", sum : " + sum);
    }
    num1=cmd_1(10); //함수  cmd-cmd-1 호출
    cmd_2(num1)
</script>
```