JavaScript

: 사용자 동작에 반응해서 동적인 효과를 만들기 위한 기술

웹 요소를 제어하고 다양한 라이브러리사용, 웹 어플리케이션, 서버를 구성하고 서버용 프로그램을 만들 수 있음

1) 웹 요소를 제어한다.
- 웹 요소를 가져와서 필요에 따라 스타일을 변경하거나 움직이게
- 웹 사이트 ui부분에 많이 활용

2) 다양한 라이브러리를 사용할 수 있다.
- 라이브러리와 프레임워크

3) 웹 애플리케이션 만든다 .

4) 서버를 구성하고 서버용 프로그램을 만들 수 있다.
- node.js : 프론트엔드 개발에 사용하던 자바스크립트를 백엔드에서도 사용할 수 있게 만든 프레임워크

- 작성하는 방법
  - <script></script> 태그 사이에 자바스크립트 소스 작성
  - 웹 문서 어디든 위치 할 수 있지만 주로 </body> 태그 앞에 작성
  - 외부스크립트파일 연결 <script src =”외부스크립트파일경로”></script>

- 자료형태

종류		설명	예시
기본유형	숫자형	따움표없이 숫자로만 표기	var birthYear=2000;

종류		설명	예시
	문자열(string)	작은 따움표(’) 나 큰 따움표(”)로 묶어서 나타낸다	var greeting=”Hello!”; var birthYear=”2000”;
	논리형(boolean)	참과 거짓이라는 2가지 값만 있음	var isEmpty = true;
복합유형	배열	하나의 변수에 여러개의 값을 지정	var seasons=[’봄’, ‘여름’, ‘가을’, ‘겨울’];
	객체	함수의 속성을 함께 포함	var date = new Date()
특수유형	undefinted	자료형이 지정되지 않았을 때의 상태	
	null	값의 유효하지 않았을 때의 상태	

Console에 typeof (자료형태 궁금한거 적기)

- 연산자
  (1) = 할당 연산자
  (2) x = x+10 → x += 10 으로 쓸 수 있음
  (3) x = x*5 → x *= 5

- 비교
  ☑️ 같은지 비교
  (1) 3 == “3” → true 자료형과 상관없이 같기 때문에 true
  (2) 3 === ”3” → false 자료형까지 같이 비교하기 때문에 false

  ☑️ 다른지 비교
  (1) 3 ! = “3” → false
  (2) 3 ! == “3” → true 자료형까지 비교 함

- 조건문 (if, switch)

```
<!DOCTYPE html>
<html lang="ko">
<head>
    <meta charset="UTF-8">
```

```
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title> 조건문 만들기</title>
</head>
<body>
    <script>
        var numberOne=prompt("숫자를 입력하세요.");
        var numberTwo=prompt("숫자를 입력하세요.");

        if(numberOne !== null && numberTwo !== null) {
            if(numberOne < 10 || numberTwo < 10) {
                alert("둘 중의 하나가 10보다 작습니다.");
            }
            else{
                "10보다 작은 숫자가 없습니다."
            }
        }
    </script>

</body>
</html>
```

```
<!DOCTYPE html>
<html lang="ko">

<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Switch 연습하기</title>
</head>

<body>
    <script>
        var session = prompt("관심 수업을 선택해 주세요. 1. 마케팅 2. 개발 3. 디자인");

        switch (session) {
            case "1": document.write("마케팅 수업은 201호에서 진행됩니다.");
                break;

            case "2": document.write("개발 수업은 202호에서 진행됩니다.");
                break;

            case "3": document.write("디자인 수업은 203호에서 진행됩니다.");
                break;

            default: alert("진행되지 않는 수업입니다. 1. 마케팅, 2. 개발, 3. 디자인 수업만 수강가능
합니다.")
        }


    </script>
</body>
```

```
</html>
```

- 반복문 (for, while, do-while)

(1) for 문

for(초기값; 조건; 증감식;) {
}

- 초기값 : 변수 초기화 (초기값은 0이나 1부터 시작)
- 조건 : 명령을 반복하기 위해 조건을 체크 이 조건을 만족해야 그 다음에 오는 명령을 실행할 수 있음
- 증감식 : 명령을 반복한 후 실행

```
<!DOCTYPE html>
<html lang="ko">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>반복문</title>
</head>
<body>
    <script>
        var i;
        var sum =0;

        for(i=0; i<6; i++) {
            sum += i
        }

        document.write("1부터 5까지의 합은" + sum);
    </script>
</body>
</html>
```

(2) while / do-while

- while : 조건을 체크하고 true 라면 { } 안의 명령 실행
→ false 라면 명령은 한 번도 실행되지 않을 수 있음

```
while(조건){
     실행할 명령
    }
```

- do~while : 일단 명령을 한번 실행 한 후 조건 체크
→ true 라면 { } 안의 명령 실행, false 라면 { } 을 빠져 나옴
→ false 라도 명령은 최소한 한번은 실행

```
do{
    실행할 명령
} while (조건)
```

- 지역변수 / 전역변수

① **var는 함수 레벨 스코프(Function-level scope)**
: 함수 내에서 선언된 변수는 **함수 내에서만 유효**하며 함수 외부에서는 참조할 수 없다. 즉, **함수 내부에서 선언**한 변수는 지역 변수이며 함수 외부에서 선언한 변수는 모두 전역 변수이다.

② **let과 const는 블록 레벨 스코프(Block-level scope)**
: 모든 코드 블록(함수, if 문, for 문, while 문, try/catch 문 등) 내에서 선언된 변수는 **코드 블록 내에서만 유효**하며 코드 블록 외부에서는 참조할 수 없다. 즉, **코드 블록 내부**에서 선언한 변수는 지역 변수이다.

- **Hoisting : 호이스팅은 코드가 실행하기 전** 함수 안에 있는 선언들을 모두 끌어올려서 해당 함수 유효 범위의 최상단에 선언하는 것을 말한다.

✔️ 호이스팅의 대상
> var 변수 선언과 함수선언문에서만 호이스팅이 일어난다.
> var 변수/함수의 선언만 위로 끌어 올려지며, 할당은 끌어 올려지지 않는다.
> let/const 변수 선언과 함수표현식에서는 호이스팅이 발생하지 않는다.
이

- 함수
  ⇒ 를 이용해서 쉽게 함수 생성 가능

```
<!DOCTYPE html>
<html lang="ko">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>function</title>
</head>
<body>
    <script>
        var hi = function(){ /*const 상수*/
            alert("안녕")
        }

        // const hi = () => {alert("안녕하세요")};

        hi();

    </script>

</body>
</html>
```

- 이벤트
  - 웹 브라우저나 사용자가 행하는 동작
  - 웹 문서 영역 안에서 이루어지는 동작만 가리킴
  - 주로 마우스나 키보드를 사용할 때 웹 문서를 불러올 때, 폼에 내용을 입력할 때 발생

  ✅ 이벤트 처리기
  : 이벤트가 발생하였을 때 처리하는 함수 (이벤트 핸들러 event handler 라고도 함)

  이벤트가 발생한 html태그에 이벤트 처리기를 직접 연결
  → <태그 on 이벤트명 = “함수명”>

* 예시

```

```

* 마우스 이벤트

종류	설명
click	사용자가 html요소를 클릭할 때 이벤트 발생
dblclick	사용자가 htmld요소를 더블클릭할때 발생
mousedown	사용자가 요소위에서 마우스 버튼 눌렀을 때 발생
mousemove	사용자가 요소위에서 마우스포인터 움직일때 발생
mouseover	마우스 포인터가 요소 위로 옮겨질 때 이벤트 발생
mouseout	마우스 포인터가 요소를 벗어날 때 이벤트 발생
mouseup	사용자가 요소 위에 놓인 마우스 버튼에서 손을 뗄 때 이벤트 발생

*키보드 이벤트

종류	설명
keydown	사용자가 키를 누르는 동안 이벤트 발생
keypress	사용자가 키를 눌렀을 대 이벤트 발생
keyup	사용자가 키에서 손을 뗄 때 이벤트 발생

*문서 로딩 이벤트

종류	설명
abort	문서가 완전히 로딩되기 전에 불러오기를 멈췄을 때 이벤트 발생
error	문서가 정확히 로딩되지 않았을 때 이벤트 발생
load	문서 로딩이 끝나면 이벤트가 발생
resize	문서 화면 크기가 바뀌었을 때 이벤트 발생
scroll	문서화면이 스크롤 되었을 때 이벤트 발생
unload	문서에서 벗어날때 이벤트 발생

* 폼이벤트

종류	설명
blur	폼 요소에 포커스를 잃었을 때 이벤트 발생
change	목록이나 체크 상태 등이 변경되면 이벤트 발생 <input> <select> <textarea> 태그에서 사용

종류	설명
focus	폼 요소에서 포커스가 놓였을 때 이벤트 발생 <label><select><textarea> <button> 태그에서 사용
reset	폼이 리셋되었을 때 발생
submit	submit 버튼 클릭 시 이벤트 발생