처음부터 시작하는 웹 프론트엔드 기초

4. Javascript 문법 : 변수와 자료형

Javascript

- HTML과 CSS로 구성된 웹 페이지를 동적으로 만들어주는 언어

- HTML은 웹 사이트에서 화면에 표시되는 정보
- CSS는 구체적으로 어떤 스타일로 요소가 표시 되는지를 정하는 규격

=> 위 둘은 정적인 것만 할 수 있는 언어

- 단순히 규격을 나타내는 HTML과 CSS와 달리, 변수와 함수 등이 존재하는 프로그래밍 언어

Javascript 사용법

1. HTML 문서 내 사용

2. 외부 파일 불러오기

Javascript 사용법 : HTML 문서 내 기술

- <script> 태그 사용
  - html 문서 내에서 Javascript 코드를 사용 가능하게 해주는 태그

- <head></head> , <body></body> 양쪽에 모두 있어도 상관 없습니다.

```
<script>
  alert('hello world');
</script>
```

index.html > ...

```
1   <!DOCTYPE html>
2   <html lang="ko">
3     <head>
4       <title>Document</title>
5       <script>
6         alert("hello world");
7       </script>
8     </head>
9     <body>
10      <p>자바스크립트 예시</p>
11    </body>
12  </html>
```

Javascript 사용법 : HTML 문서 내 기술

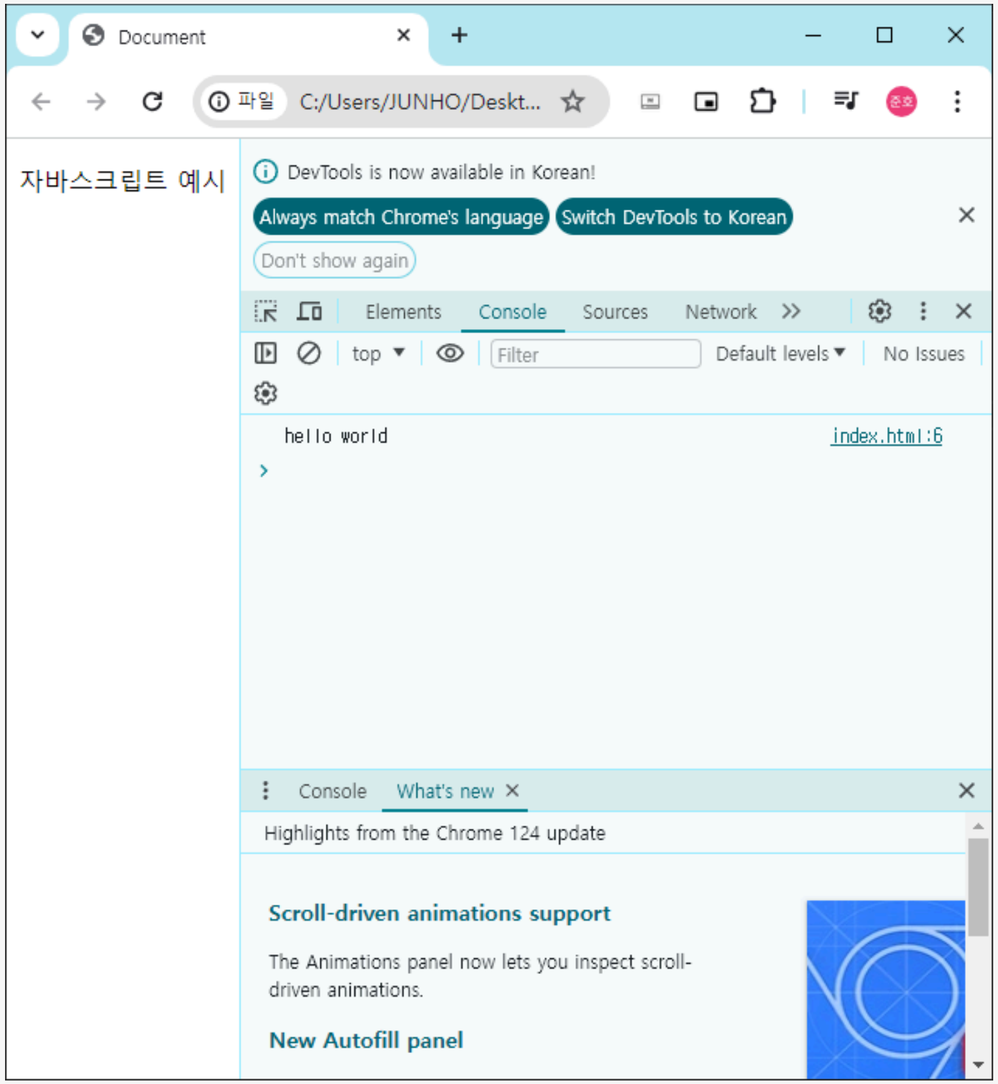
- 출력문 : Alert 창을 이용한 출력
  - 웹브라우저에서 오픈되는 조그만 경고창(alert)을 이용한 출력
  - alert(...);
  - 보통 프로그램에서 에러, 경고, 사용자 입력을 위해 사용

```
<script>
  alert('hello world');
</script>
```

Javascript 사용법 : HTML 문서 내 기술

- 출력문 : 브라우저 콘솔창을 이용한 출력
  - 웹브라우저의 콘솔창에 결과가 나타남
  - console.log(...);
  - F12 키를 눌러 개발자 도구 열기
    - -> console 태그 클릭!

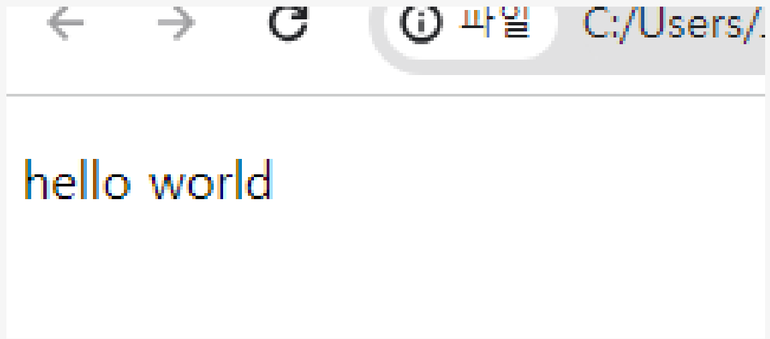
```
<script>
  console.log('hello world');
</script>
```



Javascript 사용법 : HTML 문서 내 기술

- 출력문 : innerHTML
  - HTML 문서의 특정요소를 찾아 해당 콘텐츠를 대체해 출력 합니다.
  - HTML DOM을 활용해 HTML 요소 핸들링
  - innerHTML에 대한 자세한 사항은 다음 시간에!

```
<script>
  document.getElementById("result").innerHTML = 'hello world';
</script>
```



index.html > ...

```
1   <!DOCTYPE html>
2   <html lang="ko">
3     <head>
4       <title>Document</title>
5     </head>
6     <body>
7       <p id="result"></p>
8       <script>
9         document.getElementById("result").innerHTML = "hello world";
10      </script>
11    </body>
12  </html>
```

Javascript 사용법 : 외부 파일 불러오기

- HTML 문서 외부의 별도의 파일을 불러와 사용하는 방식
  - css는 link 태그를 사용해서
  - javascript는 script 태그의 src 속성을 사용해서!

```
<link rel="stylesheet" href="style.css" />
```

```
<script src="../script.js"></script>
```

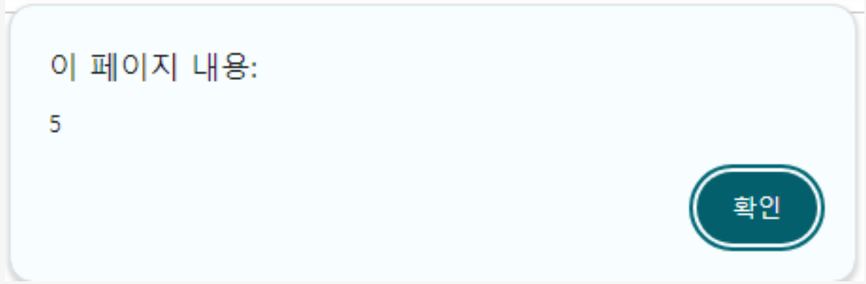
Javascript 문법 : 변수

- 변수란?
  - 데이터를 저장하고 사용하기 위한 값을 저장하는 메모리 공간

```
let x = 5;
const pi = 3.14;
var y = 10;
```

index.html > ...

```
<!DOCTYPE html>
<html lang="ko">
  <head>
    <title>Document</title>
    <script>
      var x = 5;
      alert(x);
    </script>
  </head>
  <body></body>
</html>
```



Javascript 문법 : 변수

- 변수 선언 : 변수 타입 + 변수 이름 ;
- 자바스크립트의 실행문은 세미콜론 (;) 으로 구분
- 변수 할당 : 변수 이름 = 값 ;
- 선언과 할당 동시 가능

```
var i;  // 변수 선언
i = 0 ; // 변수 할당
var sum = 0;  // 변수 선언과 초기화
name = "javascript";  // 선언되지 않은 변수는 전역 변수가 됨
```

Javascript 문법 : 변수

- 변수 선언
  - 변수 선언에는 3가지 방법이 있습니다.
    - var
    - let
    - const

```
let x = 5; // 변수 x를 선언하고 5를 할당
const y = 3.14; // 상수 y를 선언하고 3.14를 할당
var z = 'hello world'; // 변수 z를 선언하고 문자열'hello world'를 할당
```

Javascript 문법 : 변수

- 변수 선언
  - var (Variable) : 함수 범위 지역변수. 함수 외부에서는 전역 변수.
    - 재선언 가능
    - 업데이트 가능

```
var greeter = "hey hi";
var greeter = "say Hello instead";
```

```
var greeter = "hey hi";
greeter = "say Hello instead";
```

Javascript 문법 : 변수

- 변수 선언
  - let : 블록 범위 {...} 내에서만 사용 가능한 변수
    - 재선언 불가능
    - 업데이트 가능

```
let greeting = "say Hi";
greeting = "say Hello instead";
```

```
let greeting = "say Hi";
let greeting = "say Hello instead"; // error
```

Javascript 문법 : 자료형

- 변수 선언
  - const (Constant) : 블록 범위 내 사용 가능 변수. 일정한 상수 값 유지.
    - 재선언 불가능
    - 업데이트 불가능

```
const greeting = "say Hi";
greeting = "say Hello instead"; // error
```

```
const greeting = "say Hi";
const greeting = "say Hello instead"; // error
```

Javascript 문법 : 변수

- 지역 함수? 전역 함수?
- 변수의 scope?
- hoisting?
- 블록 범위?

지금은 알 지 못해도 괜찮다! 천천히 넘어가면서 배워보자!

Javascript 문법 : 자료형

- 변수에 할당하는 값의 타입
- 기본 자료형 (Primitive)
  - **숫자(Number)**
  - **문자열(String)**
  - **불리언(Boolean)**
  - **null: 값이 없음을 나타냄**
  - **undefined: 값이 할당되지 않은 변수가 가지는 값**

Javascript 문법 : 자료형

- 변수에 할당하는 값의 타입
- 객체 자료형 (Object)
  - **객체(Object)**
  - **배열(Array)**
  - **함수(Function)**

Javascript 문법 : 자료형

- 숫자(Number)
  - 정수 , 부동소수점 등등 숫자 자료형을 표현

```
let num1 = 10;
let num2 = -8;
let num3 = 0.5;
let num4 = -3.2;
```

Javascript 문법 : 자료형

- 숫자(Number) : 연산자
  - 숫자형 자료형에는 연산자를 사용할 수 있다

+	덧셈
-	뺄셈
*	곱셈
/	나눗셈
%	나머지

Javascript 문법 : 자료형

- 숫자(Number) : 연산자 예시

```
let num1 = 4;
let num2 = 2;

// 더하기(+)
let sum = num1 + num2;
console.log(sum); // 출력: 6

// 빼기(-)
let difference = num1 - num2;
console.log(difference); // 출력: 2

// 곱하기(*)
let product = num1 * num2;
console.log(product); // 출력: 8

// 나누기(/)
let quotient = num1 / num2;
console.log(quotient); // 출력: 2

// 나머지(%)
let remainder = num1 % num2;
console.log(remainder); // 출력: 0
```

Javascript 문법 : 자료형

- 문자열(String)
  - 텍스트 데이터를 나타내는 데 사용
  - 작은 따옴표(')나 큰 따옴표(")로 둘러싸인다.

```
let greeting = "Hello, World!";
let message = 'This is a message.';
```

Javascript 문법 : 자료형

- 문자열(String) : 이스케이프 문자
  - 줄바꿈을 표현하거나 따옴표를 쓰려면?
  - 아래처럼 코드를 작성하면 오류가 발생

```
var myString = "first line
second line
third line";
```

```
var yourString = """;
```

Javascript 문법 : 자료형

- 문자열(String) : 이스케이프 문자
  - 줄바꿈을 표현하거나 특수문자를 표현하기 위해 사용

\n	줄바꿈
\t	수평 탭
\\	문자 “\”
\'	작은 따옴표(')
\"	큰 따옴표(")

Javascript 문법 : 자료형

- 문자열(String) : 이스케이프 문자 예시

```
var myString = "first line\nsecond line\nthird line";
```

```
var yourString = "\"";
```

Javascript 문법 : 자료형

- 문자형 변수 : 문자열 연결
  - 문자열끼리 이어 붙일때 + 연산자를 사용

  - ‘문자열 + 숫자’ 처럼 다른 타입의 변수를 이어 붙여도, 변수를 강제로 **문자열로** 변형하여 이어 붙임

```
let str1 = "a";
let str2 = "b";
let result = str1 + str2;
console.log(result); // 출력: "ab"
```

```
let myAge = 20;
let message = "My age is " + myAge;
console.log(message); // 출력: "My age is 20"
```

Javascript 문법 : 자료형

- Boolean
  - 참(True) 또는 거짓(False) 두 가지 값을 나타내는 자료형
  - 주로 조건문이나 논리 연산에서 사용
  - 대소문자가 구분되니 true , false 소문자로 사용하는 것에 주의!

```
let isTrue = true;
let isFalse = false;
```

Javascript 문법 : 자료형

- 배열(Array)
  - 여러 값을 순서대로 저장하는 자료구조
  - 배열은 []나 new Array()로 생성
  - 하나의 배열에 서로 다른 타입의 변수가 들어갈 수 있다.

```
var emptyArray = [];
var oddNumbers = [1, 3, 5, 7, 9];
var fruits = ['apple', 'banana', 'orange'];
var evenNumbers = new Array(2, 4, 6, 8, 10);
```

Javascript 문법 : 자료형

- 배열(Array)
  - 배열의 각 요소에 접근하기 위해서는 해당 요소의 index 를 활용한다.
  - 인덱스는 0부터 시작
    - 첫 번째 요소는 인덱스 0에, 두 번째 요소는 인덱스 1에 위치
  - 인덱스에 접근하려면 배열 뒤에 [] 사용

```
console.log(oddNumbers[0]); // 출력: 1
console.log(fruits[1]); // 출력: 'banana'
```

Javascript 문법 : 자료형

- 배열 : 메소드
  - method : 객체 자료형에서 객체가 수행할 수 있는 동작이나 기능
  - . 을 사용하여 해당 메소드의 동작을 수행

Javascript 문법 : 자료형

```
var colors = ['red', 'blue'];
```

- 길이

```
console.log(colors.length);
// 2
```

- 맨 뒤에 항목 추가

```
colors.push('green');
// ['red', 'blue', 'green']
```

- 맨 뒤의 항목 제거

```
colors.pop(); // 함수의 반환값: green
// 배열의 값: ['red', 'blue']
```

Javascript 문법 : 자료형

- 객체(Object)
  - key(키) - value(값) 형태의 쌍인 속성(property)들의 집합
  - 중괄호 { } 를 사용하여 생성
  - : 과 ,를 이용해 중괄호 내부에서 키-값 쌍을 설정

```
let person = {
    name: "John",
    age: 30
};
```

Javascript 문법 : 자료형

```
let person = {
    name: "John",
    age: 30
};
```

- 객체(Object)
  - 객체에 접근하려면 인덱스 접근자 [ ] 를 사용

```
console.log(person['name']); // "John"
```

  - 인덱스를 활용하여 새로운 속성을 만들 수도 있음

```
person['city'] = 'New York';
console.log(person['city']); // "New York"
```

  - . 접근자를 사용하여 key에 접근할 수 있음

```
console.log(person.age); // 30
```

Javascript 문법 : 자료형

```
let person = {
    name: "John",
    age: 30
};
```

- 객체(Object)
  - 객체에 접근하려면 인덱스 접근자 [ ] 를 사용

```
console.log(person['name']); // "John"
```

  - 인덱스를 활용하여 새로운 속성을 만들 수도 있음

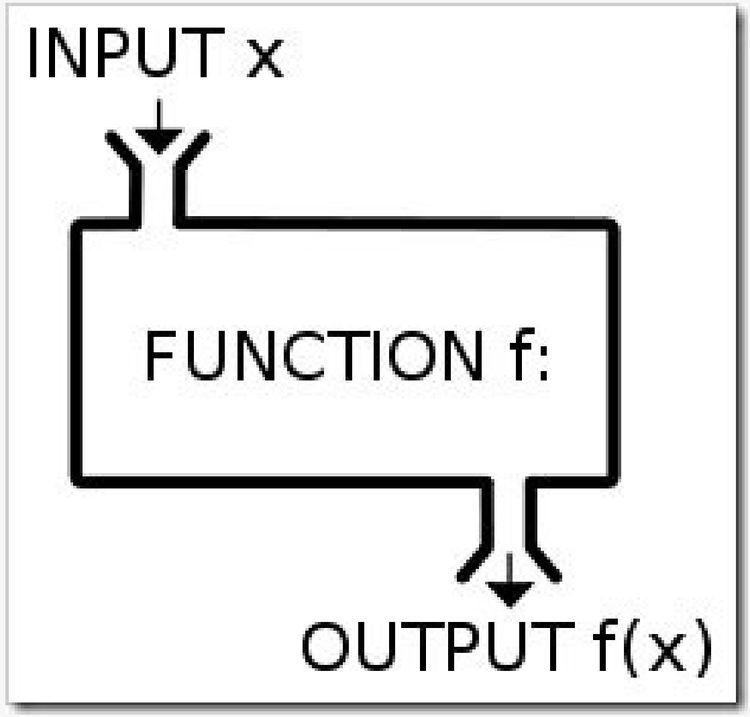
```
person['city'] = 'New York';
console.log(person['city']); // "New York"
```

  - . 접근자를 사용하여 key에 접근할 수 있음

```
console.log(person.age); // 30
```

Javascript 문법 : 자료형

- 함수(function)



Javascript 문법 : 자료형

- 함수(function)
  - 특정 작업이나 계산을 수행하는 코드 블록
  - function 키워드로 함수를 선언하며, 매개변수를 받아, return을 통해 값을 반환

Javascript 문법 : 자료형

- 함수 선언
  - function 함수이름 (매개변수1 , 매개변수2 ….) { …} ;
  - 함수를 선언하는 것 만으로는 실행되지 않음!

```
function plus(a, b) {
  var sum = a + b;
  return sum;
}
```

Javascript 문법 : 자료형

- 함수 호출
  - 해당 함수의 코드를 실행하는 것을 의미
  - 함수를 호출할 때에는 함수 이름과 괄호를 사용하여 호출
  - 함수이름 (매개변수1값 , 매개변수2값, ...);

```
plus(3, 5);
```

Javascript 문법 : 자료형

- 매개변수 없이 함수를 만들 수 있다!

```
function sayHello() {
    console.log("안녕하세요!");
}

sayHello(); // "안녕하세요!"
```

Javascript 문법 : 자료형

- Scope(범위)
  - 변수가 접근 가능한 범위를 나타냄
    - 전역(global) scope
    - 지역(local) scope
      - 함수 scope
      - 블록 scope

Javascript 문법 : 자료형

- var 의 scope
  - 함수 외부에서 선언될 때의 범위는 전역 scope
    - 전체 윈도우 상에서 사용 가능
  - 함수 내에서 선언될 때는 함수 scope.
    - 해당 함수 내에서만 사용하고 접근 가능

```
var tester = "hey hi";

function newFunction() {
    var hello = "hello";
}

console.log(tester); // "hey hi"
console.log(hello); // error
```

Javascript 문법 : 자료형

- let 과 const 의 scope
  - 함수 외부에서 선언
    - 해당 <script> 태그 내, 또는 해당 자바스크립트 파일에서만 접근 가능
  - 블록 scope
    - 중괄호로 감싸진 { } 안에 있는 해당 블록 범위 내에서만 사용 가능

Javascript 문법 : 자료형

- let 과 const 의 scope

```
function greet() {
    function sayHello() {
        let hello = "say Hello";
        console.log(hello); // "say Hello"
    }

    sayHello(); // "say Hello"

    console.log(hello); // error
}

greet();
```

마무리

- 다음시간 : 제어문 + DOM
- 과제는 없습니다
- 5월 8일 세션 정상 진행
  - 다음 주에 동방에서 봅시다!
- CSS 과제 : 5월 8일 세션 시작 전까지