6주차 JavaScript

상태	승인
설명	JS 시작 , 변수, 자료형

자바스크립트(JavaScript)

자바스크립트를 활용하면 웹페이지에 동적인 요소들을 채워 넣어 줄 수 있다.

자바스크립트는 웹 브라우저만 만드는 역할을 하는 것은 아니며, `node.js`를 활용하면 `서버사이트`나 `데스크탑 앱`, `모바일 앱`또한 만들 수 있다.

💡 **Node.js ❓**

> Node.js는 Chrome V8 JavaScript 엔진으로 빌드 된 JavaScript 런타임이다.

즉, 노드를 통해 **다양한 자바스크립트 애플리케이션을 실행**할 수 있으며, 서버를 실행하는 데 제일 많이 사용된다.

Node.js는 JavaScript를 서버에서도 사용할 수 있도록 만든 프로그램이다.
Node.js는 V8이라는 JavaScript 엔진 위에서 동작하는 자바스크립트 런타임(환경)이다.
Node.js는 서버사이트 스크립트 언어가 아니다. 프로그램(환경)이다.
Node.js는 웹서버와 같이 확장성 있는 네트워크 프로그램을 제작하기 위해 만들어졌다.

말은 조금 어렵지만 쉽게 생각해보자.

`JavaScript`는 `C` `Python` `Java`와 같은 프로그래밍 언어지만, `JavaScript`는 이름에서도 알 수 있듯이 스크립트 언어이다. 즉, 원래 `JavaScript`는 웹 브라우저(크롬,사파리 등)이 없으면 사용할 수 없는 프로그램이다.

`Node.js`를 활용하면 `JavaScript`를 웹 브라우저에서 독립시켜 사용하는것이 가능하다.

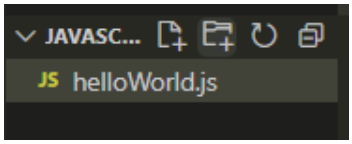
하지만 본 교재에서는 `node.js`를 사용해서 실습하진 않을 것이며 웹 브라우저를 통해서 `JS`를 다루면서 실습을 진행할 예정이다.

HelloWorld 출력하기

모든언어의 기초가 되는 HelloWorld를 출력해보는 예제를 실행해보자.

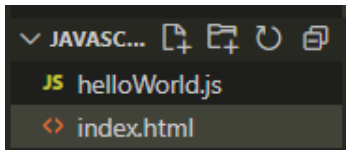
`JS`또한 `CODEPEN`과 `VSCODE` 둘 모두로 실습을 진행해 볼 수 있다.

vscode



폴더를 하나 만든 후, helloWorld.js라는 파일을 하나 만들어보자.

```
//helloWorld.js 내용
console.log("helloWorld");
```

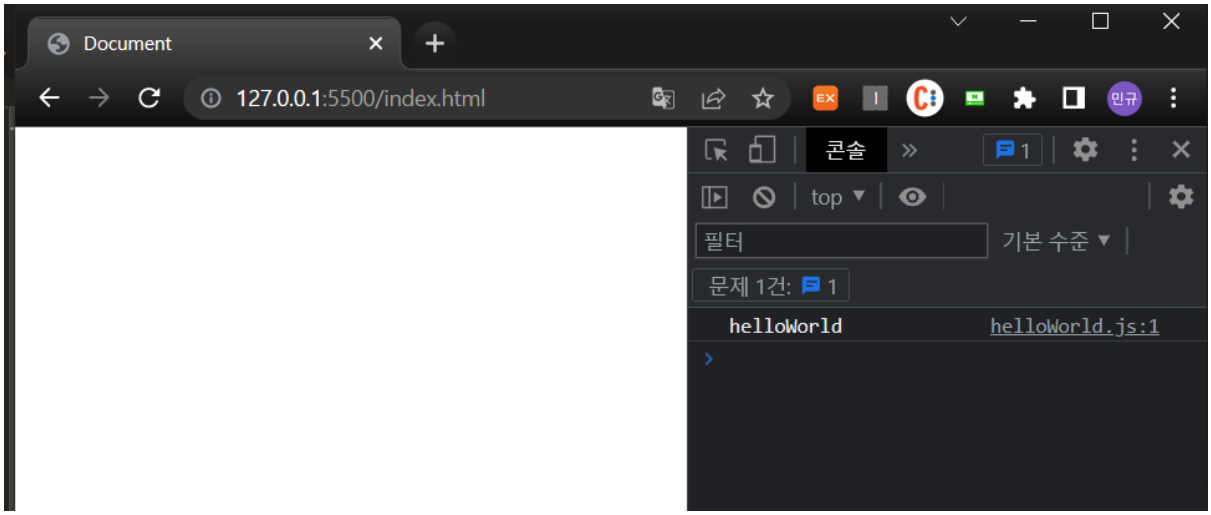


그 후 `index.html` 파일도 만들어준다.

`script` 태그를 활용하면 `html`에서 만들어준 `js`파일과 연동이 가능하다.

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <meta http-equiv="X-UA-Compatible" content="IE=edge" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <title>Document</title>
  </head>
  <body></body>
  <script src="./helloWorld.js"></script>
</html>
```

이후 `html`파일을 실행시킨 후, 나온 페이지에서 개발자옵션(크롬기준 f12)를 눌러보자.



`개발자옵션` - `콘솔` 칸에 들어가면 JS에서 작성했던

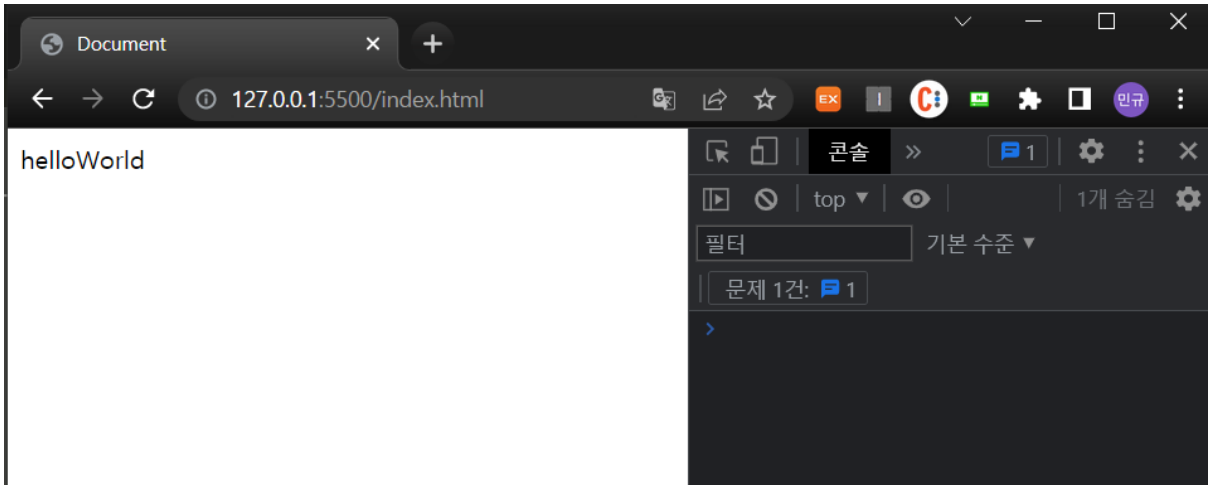
```
console.log('helloWorld');
```

가 나옴을 확인할 수 있다.

또한 `script`태그 안에 바로 JS를 작성할 수도 있다.

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <meta http-equiv="X-UA-Compatible" content="IE=edge" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <title>Document</title>
  </head>
  <body></body>
  <script>
    document.write("helloWorld");
  </script>
</html>
```

위 파일을 실행시키면



다음과 같이 나온다.

`console.log()` 는 `()` 안의 내용을 콘솔 창에 띄우는 역할을 하고

`document.write()` 는 `()`안의 내용을 웹 브라우저에 띄우는 역할을 한다

`alert()` 는 `()` 안의 내용을 알림창으로 띄워주는 역할을 한다.

💡 문제 1

`alert()` 를 사용해서 “hellowWorld” 를 알림창을 통해 띄워보자.

💡 실습 시 주의점!

본 교재의 예시로 나오는 코드들은 이제 모두 `<script>` 태그 안에 존재하게 되니 이를 생각하고 실습을 진행하자.

변수와 상수

대다수의 `JS` 애플리케이션은 사용자나 서버로부터 입력받은 정보를 처리하는 방식으로 동작한다.

1. 온라인 쇼핑몰 – 판매 중인 상품이나 장바구니 등의 정보
2. 채팅 애플리케이션 – 사용자 정보, 메시지 등의 정보

`변수`는 이러한 정보를 저장하는 용도로 사용된다.

변수

`변수`는 데이터를 저장할 때 쓰이는 '이름이 붙은 저장소' 이다.

자바스크립트에선 `let` 키워드를 사용해 변수를 생성한다.

아래 는 변수를 선언하는 예시이다.

```
let message;
```

할당 연산자 `=`를 사용하면 변수 안에 데이터를 저장할 수 있다.

```
let message;

message = 'Hello'; // 문자열을 저장
```

데이터를 저장한 이후에는, 아래와 같이 변수에 저장된 값을 사용할 수 있다.

```
let message;
message = 'Hello!';

alert(message); // 변수에 저장된 값을 출력
```

아래와 같이 변수 선언과 값 할당을 한 줄에 작성할 수도 있다.

```
let message = 'Hello!'; // 변수를 정의하고 값을 할당

alert(message); // Hello!
```

💡 `let` vs `var` ❓

`let` 대신 `var`

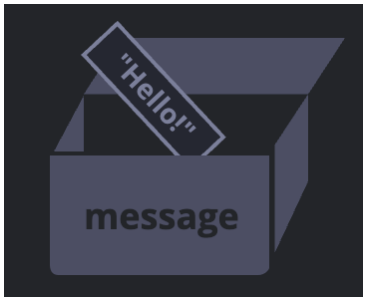
만들어진 지 오래된 스크립트에서 `let` 대신 `var`라는 키워드를 발견하는 경우가 있다.

```
var message = 'Hello';
```

`var`는 `let`과 *거의* 동일하게 동작한다. `var`도 `let`처럼 변수를 선언하는 데 쓰인다. 다만 `var`는 '오래된' 방식이다. 두가지 방식은 약간의 차이점이 있지만, 현재 단계에서는 중요하지 않기에 var 대신 let을 사용하면 된다고 생각하고 넘어가자

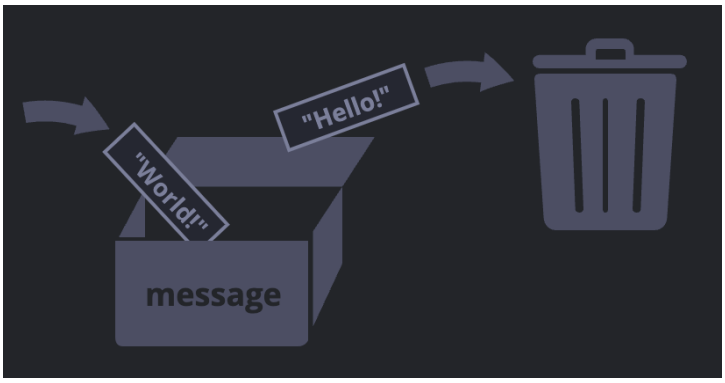
현실속의 비유

'상자' 안에 데이터를 저장하는데, 이 상자에는 특별한 이름표가 붙어 있다고 상상해 보자.



예를 들어, 변수 `message`는 `message`라는 이름표가 붙어있는 상자에 `"Hello!"`라는 값을 저장한 것이라고 생각해보자.

상자 속엔 어떤 값이든지 넣을 수 있으며 원하는 만큼 값을 변경하는 것도 가능하다.




```
let message;

message = 'Hello!';

message = 'World!'; // 값 변경

alert(message);
```

값이 변경되면, 이전 데이터는 변수에서 제거되며, 변수 두개를 선언하고 한 변수의 값을 복사하는 것도 가능하다.

```
let Hello = 'Hello world!';

let message;

// Hello의 'Hello world' 값을 message에 복사
message = Hello;// 이제 두 변수는 같은 데이터를 가진다
alert(Hello); // Hello world!
alert(message); // Hello world!
```

💡 변수를 두번 선언하면 ❓

변수는 한번만 선언해야 하며, 같은 변수를 여러 번 선언하면 에러가 발생한다.

```
let message = "This";

// 'let'을 반복하면 에러가 발생
let message = "That"; // SyntaxError: 'message' has already been declared
```

따라서 변수는 한번만 선언하고, 그 변수를 사용할때는 `let` 키워드 없이 사용해야 한다.

변수 명명 규칙

자바스크립트에선 변수 명명 시 두 가지 제약 사항이 있다.

1️⃣ 변수명에는 오직 문자와 숫자, 그리고 기호 `$`와 `_`만 들어갈 수 있다.

2️⃣ 첫 글자는 숫자가 될 수 없다.

다음은 유효한 변수명의 예시이다.

```
let userName;
let test123;
```

💡 카멜 표기범(camelCase) ❓

카멜표기법은 여러 단어를 조합하여 변수명을 만드는 방법으로

`myVeryLongName`

과 같이 첫 단어를 제외한 각단어의 첫 글자를 대문자로 작성하는 방식이다.

아래는 유효한 변수명에 관한 예시이다. 조금 특이하게 `$`와 `_` 를 변수명으로 사용할 수 있다.

```
let $ = 1; // '$'라는 이름의 변수를 선언
let _ = 2; // '_'라는 이름의 변수를 선언

alert($ + _); // 3
```

아래는 잘못된 변수명의 예시이다.

```
let 1a; // 변수명은 숫자로 시작하면 안됨

let my-name; // 하이픈 '-'은 변수명에 올 수 없음.
```

대·소문자 구별

`apple`와 `AppLE`은 서로 다른 변수이다.

예약어

예약어 목록에 있는 단어들은 변수로 사용할 수 없다. 이 단어들은 자바스크립트 내부에서 이미 사용 중이기 때문이다. 예를들어 `let`, `class`, `return`, `function` 와같은 키워드들은 변수이름으로 사용할 수 없다.

따라서 아래 코드는 에러를 발생시키게 된다.

```
let let = 5; // 'let'을 변수명으로 사용할 수 없으므로 에러!
let return = 5; // 'return'을 변수명으로 사용할 수 없으므로 에러!
```

상수

변화하지 않는 변수를 선언할 땐, `let` 대신 `const`를 사용한다.

```
const myBirthday = '18.04.1982';
```

이렇게 `const`로 선언한 변수를 '`상수(constant)`'라고 부르며 상수는 재할당이 불가능하다.

```
const myBirthday = '18.04.1982';

myBirthday = '01.01.2001'; // error, can't reassign the constant!
```

변숫값이 절대 변경되지 않을 것이라 확신하면, 값이 변경되는 것을 방지하면서 다른 개발자들에게 이 변수는 상수라는 것을 알리기 위해 `const`를 사용하는것이 좋다.

대문자 상수

상수가 사용되는 가장 대표적인 예시는 기억하기 힘든 값을 변수에 할당하여 별명으로 사용하는 것이다. 보통 이런경우, 대문자와 밑줄로 구성된 이름으로 작성한다.

예시로 웹에서 사용하는 색상 표기법인 `16진수 컬러 코드`에 대한 상수를 한번 만들어보자.

```
const COLOR_RED = "#F00";
const COLOR_GREEN = "#0F0";
const COLOR_BLUE = "#00F";
const COLOR_ORANGE = "#FF7F00";

// 색상을 고르고 싶을 때 별칭을 사용할 수 있게 되었다.
let color = COLOR_ORANGE;
alert(color); // #FF7F00
```

대문자로 상수를 만들어 사용하면 다음과 같은 장점이 있다.

1️⃣ `COLOR_ORANGE`는 `"#FF7F00"`보다 기억하기가 훨씬 쉽다.

2️⃣ `COLOR_ORANGE`를 사용하면 `"#FF7F00"`를 사용하는 것보다 오타를 낼 확률이 낮다.

3️⃣ `COLOR_ORANGE`가 `#FF7F00`보다 훨씬 유의미하므로, 코드 가독성이 증가한다.

바람직한 변수명

변수에 관한 매우 중요한 사실이 한 가지 더 있다. 변수의 이름은 최대한 간결하고, 명확하게 짓는 것이 좋다. 변수의 이름을 짓는것 또한 굉장히 어려운데, 규칙이 정해져있는건 아니지만 아래의 규칙을 따른다면 비교적 의미있는 변수명을 만들기 쉽다.

아래는 변수 명명 시 참고하기 좋은 규칙이다.

1️⃣ `userName` 이나 `shoppingCart`처럼 사람이 읽을 수 있는 이름을 사용하자.

2️⃣ 무엇을 하고 있는지 명확히 알고 있지 않을 경우 외에는 줄임말이나 `a`, `b`, `c`와 같은 짧은 이름은 피하는 것이 좋다.

3️⃣ 최대한 서술적이고 간결하게 명명할것. `data`와 `value`는 나쁜 이름의 예시가 된다. 해당 변수명만 보면 어떤 내용이 들어있는지 파악하기 어렵기 때문

4️⃣ 자신만의 규칙을 만들어서 지켜볼 것 예를들어 무언가를 다루는 코드를 작성할때 `handle`과 `control` 두가지 단어가 있지만, 그 중 한가지만 사용하기로 본인만의 규칙을 만드는 것

요약

`var`, `let`, `const`를 사용해 변수를 선언할 수 있고 해당 변수에는 `데이터(값)`을 저장할 수 있다.

1️⃣ `let` – 변수 선언 키워드이다.

2️⃣ `var` – 오래된 변수 선언 키워드로 잘 사용하지 않는다.

3️⃣ `const` – `let`과 비슷하지만, 변수의 값을 변경할 수 없다.

4️⃣ 변수명은 변수가 담고 있는 것이 무엇인지 쉽게 알 수 있도록 지어져야 한다.

문제 1 - 변수 가지고 놀기

1. `admin`과 `name`이라는 변수를 선언하세요.
2. `name`에 값으로 `"John"`을 할당해 보세요.
3. `name`의 값을 `admin`에 복사해 보세요.
4. `admin`의 값을 `alert` 창에 띄워보세요. "John"이 출력되어야 합니다.

정답

```
let admin, name; // 변수 두 개를 동시에 선언할 수 있습니다.

name = "John";

admin = name;

alert( admin ); // "John"
```

문제2 - 올바른 이름 선택하기

1. 현재 우리가 살고있는 `행성(planet)`의 이름을 값으로 가진 변수를 만들어보세요. 변수 이름은 어떻게 지어야 할까요?
2. 웹사이트를 개발 중이라고 가정하고, 현재 접속 중인 `사용자(user)`의 `이름(name)`을 저장하는 변수를 만들어보세요. 변수 이름은 어떻게 지어야 할까요?

정답

```
let ourPlanetName = "Earth";
let currentUserName = "John";
```

정해진 정답은 없지만, 위와같은 형식으로 정할 수 있다.

자료형

어떠한 변수에 `데이터(값)`을 넣었을 때 해당 `값`의 `Type` 을 자료형이라고 한다. 자바스크립트에는 여덟 가지 기본 자료형이 있으며 하나씩 알아보자.

자바스크립트의 변수는 자료형에 관계없이 모든 데이터일 수 있으며 따라서 변수는 어떤 순간에 문자열일 수 있고 다른 순간엔 숫자가 될 수도 있다.

```
// no error
let message = "hello";
message = 123456;
```

이처럼 자료의 타입은 있지만 변수에 저장되는 값의 타입은 언제든지 바꿀 수 있는 언어를 '**동적 타입(dynamically typed)**' 언어라고 부른다.

숫자형

```
let n = 123;
n = 12.345;
```

정수 및 부동소수점 숫자(floating point number)를 나타내며 곱셈 `*`, 나눗셈 `/`, 덧셈 `+`, 뺄셈 `-` 등의 연산이 가능하다.

💡 `Infinity` `-Infinity` `Nan` ❓

숫자형을 가지고 계산을 하다보면 위와같은 값이 얻어지는 경우가 있다.

`Infinity`는 어떤 숫자보다 더 큰 특수 값, 즉 무한대를 나타낸다.

```
alert( 1 / 0 ); // 무한대
```

`Infinity`를 직접 참조할 수도 있다.

```
alert( Infinity ); // 무한대
```

`NaN`은 계산 중에 에러가 발생했다는 것을 나타내주는 값이다. 부정확하거나 정의되지 않은 수학 연산을 사용하면 계산 중에 에러가 발생하는데, 이때 `NaN`이 나오게 된다.

```
alert( "숫자가 아님" / 2 ); // NaN, 문자열을 숫자로 나누면 오류가 발생합니다.
```

`NaN`은 여간해선 바뀌지 않는다. `NaN`에 어떤 추가 연산을 해도 결국 `NaN`이 반환된다.

```
alert( "숫자가 아님" / 2 + 5 ); // NaN
```

BigInt

내부 표현 방식 때문에 자바스크립트에선 `(253-1)`(`9007199254740991`) 보다 큰 값 혹은 `-(253-1)` 보다 작은 정수는 '숫자형'을 사용해 나타낼 수 없다. 대부분의 상황에서는 문제가 되지 않지만 간혹 아주 큰 숫자가 필요한 경우에 사용할 수 있다.

```
// 끝에 'n'이 붙으면 BigInt형 자료입니다.
const bigInt = 1234567890123456789012345678901234567890n;
alert(bigInt)
```

문자형

자바스크립트에선 문자열(string)을 따옴표로 묶어서 표현한다.

```
let str = "Hello";
let str2 = 'Single quotes are ok too';
let phrase = `can embed another ${str}`;
```

따옴표는 세 종류가 존재한다.

1️⃣ 큰따옴표: `"Hello"`

2️⃣ 작은따옴표: `'Hello'`

3️⃣ 역 따옴표(백틱, backtick): `` `Hello` ``

큰따옴표와 작은따옴표는 '기본적인' 따옴표로, 둘의 차이는 없다고 생각하면 된다.

`역 따옴표`로 변수나 표현식을 감싼 후 `${…}` 안에 넣어주면, 아래와 같이 원하는 변수나 표현식을 문자열 중간에 손쉽게 넣을 수 있으니 알아두면 굉장히 편리하다.

```
let name = "John";

// 변수를 문자열 중간에 삽입
alert( `Hello, ${name}!` ); // Hello, John!

// 표현식을 문자열 중간에 삽입
alert( `the result is ${1 + 2}` ); // the result is 3
```

`${…}` 안에는 `name` 같은 변수나 `1 + 2` 같은 수학 관련 표현식을 넣을 수 있으며 더 복잡한 무언가도 넣을 수 있다. 큰따옴표나 작은따옴표를 사용하면 활용할 수 없는 방식이다.

```
alert( "the result is ${1 + 2}" ); // the result is ${1 + 2} (큰따옴표는 확장 기능을 지원하지 않습니다.)
```

불린형(Boolean)

불린형(논리 타입)은 `true`와 `false` 두 가지 값밖에 없는 자료형으로 `true`는 긍정, `false`는 부정을 의미한다.

```
let nameFieldChecked = true; // 네, name field가 확인되었습니다(checked).
let ageFieldChecked = false; // 아니요, age field를 확인하지 않았습니다(not checked)
```

불린값은 비교 결과를 저장할 때 사용되곤 한다.

```
let isGreater = 4 > 1;

alert( isGreater ); // true (비교 결과: "yes")
```

불린 자료형은 단독적으로 사용되기보다는 연산자와 연동되어 사용되곤 하니 추후에 자세히 알아보자.

`null` VS `undefind` ❓

`null` 값은 지금까지 소개한 자료형 중 어느 자료형에도 속하지 않는 값으로 존재하지 않는 , 비어있는 값을 표현하는데 사용된다.

```
let age = null;
```

`undefined` 값도 `null` 값처럼 자신만의 자료형을 형성하며, 값이 할당되지 않은 상태를 표시한다.

```
let age;
alert(age); //undefined 출력
```

두 자료형이 비슷해 보이지만 보통 `null`은 개발자가 의도적으로 값을 비워두는 경우 사용되며, `undefined`는 프로그램에서 비어있음을 자동적으로 할당해주는 용도로 사용된다.

typeof

`typeof` 연산자는 인수의 자료형을 반환해준다.

```
typeof undefined // "undefined"

typeof 0 // "number"

typeof 10n // "bigint"

typeof true // "boolean"

typeof "foo" // "string"
```

요약

자바스크립트에는 여덟 가지 기본 자료형이 존재한다.

`숫자형` – 정수, 부동 소수점 숫자 등의 숫자를 나타낼 때 사용한다.

`bigint` – 길이 제약 없이 정수를 나타낼 수 있다.

`문자형` – 빈 문자열이나 글자들로 이뤄진 문자열을 나타낼 때 사용한다.

`불린형` – `true`, `false`를 나타낼 때 사용한다.

`null` – `null` 값만을 위한 독립 자료형이다. `null`은 알 수 없는 값을 나타낸다.

`undefined` – `undefined` 값만을 위한 독립 자료형이다. `undefined`는 할당되지 않은 값을 나타낸다.

배우지 않은 자료형들은 추후에 소개해보도록 하겠다

문제 - 문자열 따옴표

아래 코드 결과를 예측해보자.

```
let name = "Ilya";

alert( `hello ${1}` ); // ?
alert( `hello ${"name"}` ); // ?
alert( `hello ${name}` ); // ?
```