자바스크립트 기본

Hello, world!

- script 태그를 활용하여 자바스크립트 코드 작성 가능

```
<script>
    alert('Hello, world!');
</script>
```

- type 속성 (type="text/javascript")
  - 문서는 텍스트로 이루어져있으며 내용은 자바스크립트라는 의미의 MIME 타입
  - CSS의 경우 MIME 타입을 text/css로 설정
- type 과 language 속성은 필수가 아님

외부 스크립트 활용

- 자바스크립트 코드를 분리하고 싶은 경우, 파일에 코드를 작성한 후 **스크립트 태그의 src 속성을 이용**하여 파일 불러오기 가능

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

- src에 붙은 경로는 사이트의 루트에서부터 파일이 위치한 절대 경로를 나타냄
- URL 전체를 속성으로 사용할 수도 있음

```
<script
src="https://cdnjs.cloudflare.com/ajax/libs/lodash.js/4.17.11/lodash.js">
</script>
```

- 복수개의 스크립트 파일 혹은 여러 스크립트 태그를 삽입 가능

```
<script src="/path/to/script1.js"></script>
<script src="/path/to/script2.js"></script>
<script>
    // ... 내부 스크립트 1 ...
</script>
<script>
    // ... 내부 스크립트 2 ...
</script>
```

- 스크립트가 길어지면 별개의 분리된 파일로 만들어 저장하는 것이 성능상의 이유로 좋음 (캐시를 사용하여 브라우저에 사본 저장)
- 태그 내부에 이벤트 핸들러 관련 속성(onclick, onkeypress 등)을 정의하며 자바스크립트 코드 삽입 가능
  - 그러나 html 태그와 자바스크립트 코드가 섞여 유지보수가 어려워지므로 비권장되는 방법

```
<button onclick="alert('click')">click me</button>
<input type="text"
    onkeypress="console.log('keypress')"
    onchange="console.log('change')" />
```

코드 구조

- 문(statement)은 어떤 작업을 수행하는 문법 구조(syntax structure)와 **명령어(command)**를 의미
- 주석

```
// 한 줄 주석
/*
    여러 줄 주석
*/
```

변수 상수

- **let 키워드**를 이용하여 변수 생성 가능

```
let message = "Hello";

// 에러 발생
// Uncaught SyntaxError: Identifier 'message' has already been declared
let message = "World";
```

- 가독성을 위해 가급적 한 줄에는 하나의 변수를 작성

변수 명명 규칙

1. 변수명에는 오직 문자와 숫자, 그리고 기호 `$`와 `_`만 들어갈 수 있습니다.
2. 첫 글자는 숫자가 될 수 없습니다.
3. 대소문자 구별
4. 비라틴계 문자(ex: 한자)도 사용 가능 (권장 X)
5. 예약어 사용 불가
6. 여러 단어를 조합하여 변수명을 만들 땐 **카멜 표기법(camelCase)** 사용 권장

상수

- **const 키워드**를 이용하여 정의
- 처음 값 대입 이후 값 변경 불가
- 상수는 **대문자와 밑줄로 구성된 이름으로 명명**을 권장

자료형

- 자바스크립트의 변수는 **자료형에 관계없이 모든 데이터 대입 가능** (자료형에 대한 제약이 적음)
  - **동적 타입 언어**(JS, 파이썬) <=> 정적 타입 언어(JAVA, C)
    - 정적 타입 언어의 경우 변수나 상수를 선언하는 시점에 자료형이 고정됨
- 여덟 가지 자료형이 존재
  - **숫자형**
    - 정수와 실수 모두 포함 (다른 언어(ex: C)처럼 구분 X)
    - Infinity
      - 0으로 나눌 경우 다른 언어와 같이 예외처리 되지 않고 무한대 상수값을 반환
      - https://stackoverflow.com/questions/8072323/best-way-to-prevent-handle-divide-by-0-in-javascript
    - NaN
      - 숫자로 표현이 불가능한 값인 경우 Not a Number 상수값 반환
- **문자열**
  - 큰따옴표, 작은따옴표, 역따옴표(백틱) 사용 가능
  - 역따옴표 내부에서는 **${...}안에 표현식(expression) 사용** 가능
  - 한 글자만 저장하는 글자형(char) 타입 존재하지 않음
- **불린형**
- **null**
  - 존재하지 않는(nothing) 값, 비어 있는(empty) 값, 알 수 없는(unknown) 값을 나타내는 데 사용
- **undefined**
  - **"값이 할당되지 않은 상태"**를 나타낼 때 사용
  - 변수는 선언했지만, 값을 할당하지 않았다면 해당 변수에 undefined 값이 할당됨
  - 함수에서 값을 반환하지 않는 경우 undefined 값을 반환함
  - undefined 값을 직접 할당(대입)하는 걸 권장하진 않음, 만약 **값을 모르거나, 값이 없음을 명시해야 한다면 null 값을 할당**
- **객체(object)**
  - 원시(primitive) 자료형과는 다르게 데이터 컬렉션이나 복잡한 개체 표현 가능 (다양한 데이터를 집합으로 표현 가능하다고 이해)
- **객체외에는 모두 원시 자료형**

- **typeof 연산자**로 자료형을 확인 가능
  - typeof x 또는 typeof(x) 형태로 사용

```
typeof undefined // "undefined"
typeof 0 // "number"
typeof true // "boolean"
typeof "foo" // "string"
typeof Math // "object"
typeof null // "object"
typeof alert // "function"
```

- null은 별도의 **고유한 자료형을 가지는 특수 값으로 객체(object)가 아니지만**, 하위 호환성을 유지하기 위해 이런 오류를 수정하지 않고 남겨둔 상황임을 유의
- **함수형은 별도로 존재하지 않음**, 함수는 내부적으로 **객체**로 취급됨 (하지만 typeof 연산자를 사용할 경우 "function" 반환)

- (*) 자바스크립트 언어에서 모든 원시 자료형(number, bigint, string, boolean, undefined, null, symbol)이 아닌 값들은 **모두 객체로 취급**됨

alert, prompt, confirm을 이용한 상호작용

- alert
  - 함수가 실행되면 사용자가 '확인(OK)' 버튼을 누를 때까지 메시지를 보여주는 모달창이 계속 떠있음
    - '모달'이란 단어엔 페이지의 나머지 부분과 상호 작용이 불가능하다는 의미가 내포, 따라서 확인 버튼을 누를 때까지 모달 창 바깥에 있는 버튼을 누른다든가 하는 행동을 할 수 없음
    - prompt, confirm 함수와는 다르게 반환값이 없음 (undefined 반환)
- prompt
  - 함수가 실행되면 텍스트 메시지와 입력 필드(input field), 확인(OK) 및 취소(Cancel) 버튼이 있는 모달 창을 띄워줌

```
// title : 사용자에게 보여줄 문자열
// default : 입력 필드의 초깃값
prompt(title, [default]);

// ex
let age = prompt('나이를 입력해주세요.', 100);
```

- confirm
  - confirm 함수는 매개변수로 받은 question(질문)과 확인 및 취소 버튼이 있는 모달 창을 보여줌

```
let isBoss = confirm("당신이 주인인가요?");
```

    - 사용자가 확인버튼를 누르면 true, 그 외의 경우는 false를 반환

형 변환

- String 함수를 호출해 전달받은 값을 문자열로 변환 가능
  - false는 문자열 "false"로, null은 문자열 "null"로, undefined는 "undefined" 변환
- Number 함수를 호출해 전달받은 값을 숫자로 변환 가능
  - 숫자 이외의 글자가 들어가 있는 문자열을 숫자형으로 변환하려고 하면, 변환이 불가능하므로 결과는 NaN이 됨

```
Number("임의의 문자열 123");
```

- Boolean 함수를 호출하면 명시적으로 불리언으로의 형 변환을 수행
  - 숫자 0, 빈 문자열(""), null, undefined, NaN과 같이 직관적으로도 "비어있다고" 느껴지는 값(falsy value)들은 false가 되며 그 외의 값(truthy value)은 true로 변환

기본 연산자와 수학

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

- **는 거듭제곱 연산자
- + 연산자를 이용하여 문자열 연결 가능
- 피연산자 중 하나가 문자열이면 다른 하나도 문자열로 변환 (실수 유발 가능)
- + 연산자를 단항 연산자로 사용할 경우 숫자 타입으로 변환 진행

```
// 전혀 권장하지 않지만 동작
+true   // 1
+""     // 0
```

- Number 함수를 호출하여 숫자로 변환하는 것보다 좀 더 짧게 코드를 작성 가능하다는 장점이 있음

```
let apples = "2";
let oranges = "3";

// 이항 덧셈 연산자가 적용되기 전에, 두 피연산자는 숫자형으로 변화합니다.
alert( +apples + +oranges ); // 5
```

- NaN에 대한 숫자 연산은 모두 NaN을 반환 (undefined에 대한 숫자 연산도 NaN을 반환)

```
alert( NaN + 1 )
alert( NaN - 1 )
alert( NaN * 1 )
alert( NaN / 1 )
```

연산자 우선순위

- 후위형, 전위형 증감 연산자의 연산 결과값에 차이가 있음을 주의
- 아래의 경우 증감이 이루어진 후 counter 값이 반환되어 a에 저장됨

```
let counter = 1;
let a = ++counter;

alert(a); // 2
```

- 아래의 경우 증감이 이루어지기 전 counter 값이 반환되어 a에 저장되고 그 이후에 증감 연산이 수행됨

```
let counter = 1;
let a = counter++;

alert(a); // 1
```

- 반환 값을 사용하지 않는 경우라면, 전위형과 후위형엔 차이가 없음

비교 연산자

```
>, >=, <, <=, ==, ===, !=, !==
```

- **엄격한 비교 연산자(strict equality operator)** 사용이 권장됨
  - === 연산자
  - 비교 연산자(==)와 달리 === 연산자는 값의 타입까지 비교함
  - 반대의 케이스는 !== 연산자 사용