

혼자 공부하는 자바스크립트

Chapter 01 자바스크립트 개요와 개발환경 설정



학습 목표

- **CHAPTER 01: 자바스크립트 개요와 개발환경 설정**
 - 자바스크립트 개발환경 설치와 자바스크립트 프로그래밍 기본 용어 학습
- **CHAPTER 02: 자료와 변수**
 - 프로그램 개발의 첫걸음. 자료형과 변수 학습
- **CHAPTER 03: 조건문**
 - 프로그램의 흐름을 변화시키는 요소. 조건문의 종류를 알아보고 사용 방법을 이해
- **CHAPTER 04: 반복문**
 - 배열의 개념과 문법을 익혀 while 반복문과 for 반복문 학습
- **CHAPTER 05: 함수**
 - 다양한 형태의 함수를 만들기와 매개변수를 다루는 방법 이해
- **CHAPTER 06: 객체**
 - 객체의 속성과 메소드, 생성, 관리하는 기본 문법 학습
- **CHAPTER 07: 문서 객체 모델**
 - DOMContentLoaded 이벤트를 사용한 문서 객체 조작과 다양한 이벤트의 사용 방법 이해
- **CHAPTER 08: 예외 처리**
 - 구문 오류와 예외를 구분하고, 예외 처리의 필요성과 예외를 강제로 발생시키는 방법을 이해
- **CHAPTER 09: 클래스**
 - 객체 지향을 이해하고 클래스의 개념과 문법 학습
- **CHAPTER 10: 리액트 라이브러리**
 - 리액트 라이브러리 사용 방법과 간단한 애플리케이션을 만드는 방법 학습



Contents

- CHAPTER 01: 자바스크립트 개요와 개발환경 설정

SECTION 1-1 자바스크립트의 활용

SECTION 1-2 개발환경 설치와 코드 실행

SECTION 1-3 알아두어야 할 기본 용어



CHAPTER 01 자바스크립트 개요와 개발환경 설정

자바스크립트 개발환경 설치와 자바스크립트 프로그래밍 기본 용어 학습

SECTION 1-1 자바스크립트의 활용(1)

- 자바스크립트(JavaScript)는 웹 브라우저에서 사용하는 프로그래밍 언어
- 자바스크립트로 할 수 있는 것들
 - 웹 클라이언트 애플리케이션 개발
 - 초기의 웹은 변하지 않는 정적인 글자로 이뤄진 커다란 책 → 자바스크립트가 나오며 웹 문서의 내용을 동적으로 바꾸거나 사용자의 마우스 클릭과 같은 이벤트 처리가 가능
 - 웹 서버 애플리케이션 개발
 - 기존 웹 개발에는 2 가지 이상의 프로그래밍 언어가 필요
 - 웹 클라이언트 애플리케이션을 자바스크립트로 개발하고, 웹 서버 애플리케이션은 C#, 자바(Java), 루비(Ruby), 파이썬(Python) 등
 - 2009년에 Node.js가 등장하면서 자바스크립트만으로 웹 서버 애플리케이션 개발이 가능해짐
 - Node.js의 장점
 - 처리 속도가 빠름
 - 빠른 배포, 업그레이드 작업이 가능
 - 타 언어에 비해 배우기 쉽고, 생산성이 좋음
 - 프론트엔드와 백엔드 기술의 통합

SECTION 1-1 자바스크립트의 활용(2)

- 자바스크립트로 할 수 있는 것들
 - 모바일 애플리케이션 개발
 - 페이스북의 리액트 네이티브(React Native) : 자바스크립트만으로 모든 운영체제에서 빠르게 작동하는 네이티브 애플리케이션 작성 가능
 - 안드로이드폰은 자바/코틀린(Kotlin), 아이폰은 스위프트(Swift) 프로그래밍 언어로 개발

SECTION 1-1 자바스크립트의 활용(3)

◦ 자바스크립트의 종류

- 1990년대 중반부터 자바스크립트가 많은 곳에서 사용되자 유럽컴퓨터제조협회(ECMA)는 자바스크립트를 ECMAScript라는 이름으로 표준화
- 2000년대 중반부터 자바스크립트가 많은 곳에서 널리 사용되며, 자바스크립트의 문법이 급속도로 발전

| ECMAScript | 버전 표준 발표 시기 |
|-----------------|-------------|
| ECMAScript 1 | 1997년 6월 |
| ECMAScript 2 | 1998년 6월 |
| ECMAScript 3 | 1999년 12월 |
| ECMAScript 4 | 2008년 10월 |
| ECMAScript 5 | 2009년 12월 |
| ECMAScript 2015 | 2015년 6월 |
| | |
| ECMAScript 2020 | 2020년 6월 |

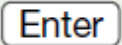
- ▲ ECMAScript 6부터는 발표 연도를 사용해서 ECMAScript 2015와 같이 버전을 부르는 경우가 일반적

SECTION 1-2 개발환경 설치와 코드 실행(1)

- 개발환경에는 코드를 작성하는 **텍스트 에디터**와 코드를 실행하는 **코드 실행기**가 필요
 - 텍스트 에디터: 비주얼 스튜디오 코드(Visual Studio Code) 사용
 - 코드 실행기: 구글 크롬 웹 브라우저를 사용
- 구글 크롬 설치하기
- 비주얼 스튜디오 코드 설치하기
 - 비주얼 스튜디오 홈페이지: <https://code.visualstudio.com>
 - 한국어 언어팩 설치하기: Extensions > korean 검색 후 설치

SECTION 1-2 개발환경 설치와 코드 실행(2)

- 코드 실행하기(1): 구글 크롬 콘솔에서 실행하기
 - 01: 구글 크롬의 주소창에 about:blank를 입력해 크롬이 기본적으로 제공하는 빈 페이지로 들어가기
 - 02: 단축키 Ctrl + Shift + I (또는 F12)를 눌러 개발자 도구를 실행하고 [Console] 탭을 클릭
 - 03: 코드를 입력하고 Enter 키를 누르면 곧바로 코드 실행을 확인
 - > console.log("Hello JavaScript...!") → [Enter]
 - 04: 코드의 실행 결과가 다음과 같이 나오는 것을 확인

> console.log("Hello JavaScript...!")  입력한 코드

Hello JavaScript...!  console.log()로 출력된 내용

undefined  해당 줄의 결과

SECTION 1-2 개발환경 설치와 코드 실행(3)

- 코드 실행하기(2): 파일 만들고 저장해 실행하기

- 1단계: HTML 페이지 생성하기

- 01: 비주얼 스튜디오 코드 메뉴에서 [파일] - [새 파일]을 선택해서 새 파일 생성

- 02: 생성한 파일을 곧바로 저장합니다. 메뉴에서 [파일] - [저장], 폴더를 지정하고 test.html이라는 이름으로 저장

- 2단계: HTML 페이지 작성하기

- 01: 새 창에 html이라고 입력하는 중에 다음과 같이 자동 완성이 나타나면 [html:5]를 선택하고 [Enter]

- 02: [html:5]를 선택했을 때 자동 완성되는 코드 확인

- 03: 생성된 HTML 페이지를 다음과 같이 간략하게 만들어서 사용

- 04: 자바스크립트를 사용하기 위해 기본 HTML 페이지의 <head> 태그 사이에 <script> 태그를 삽입하고 <script> 태그 사이에 자바스크립트 코드를 입력

```
<script> console.log('console test'); <script>
```

- 3단계: HTML 페이지 실행하기

- 01: test.html 파일을 크롬 브라우저에 드래그&드롭하여 출력됨을 확인

- 02: 비주얼 스튜디오 코드 "실행 및 디버그" 메뉴 실행


[좀 더 알아보기①] 오류를 확인하는 방법

- 내가 무엇을 잘못 입력했는지 알아내는 방법과 찾는 방법

```
<script>  
  alrt('Hello World')  
</script>
```



alert를 alrt로 잘못 입력했다고 가정

- 01: 현재 상태에서 코드를 실행해보면 아무 것도 출력되지 않음
- 02: 크롬에서 코드를 실행한 후 마우스 오른쪽 버튼을 클릭해 [검사]를 선택
- 03: 개발자 도구 오른쪽 위에 × 표시가 되어 있는 붉은색 원  (자바스크립트 코드 등에 오류가 발생했을 때 출력되는 아이콘) 아이콘을 클릭하거나 개발자 도구의 [Console] 탭을 클릭
- 04: 'Uncaught ReferenceError: alrt is not defined'라는 오류 출력, 어떤 오류인지 확인. test.html : 6은 오류가 발생한 위치. [test.html : 6]을 클릭하면 오류가 발생한 위치로 이동
- 05: 붉은색 밑줄이 표시되어 있어 쉽게 오류를 찾을 수 있음

- 처음 자바스크립트를 공부할 때 자주 접하는 오류
 - ReferenceError: 예외 처리
 - SyntaxError: 구문 오류

[좀 더 알아보기②] 자바스크립트 표준 스타일

- 코딩 스타일 또는 코딩 컨벤션
 - 들여쓰기 2개와 4개
 - 따옴표는 한 종류로 통일(' 권장)
 - 중괄호 입력 방식
 - 키워드 뒤에 공백
- 의도하지 않은 오류 방지
- 가독성 증대 -> 유지보수 비용 감소

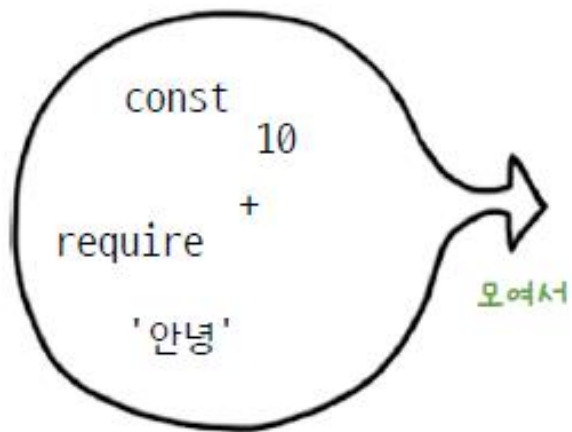
```
if (condition) {  
    // 코드1  
    // 코드n..  
}  
  
// bad  
if (n < 0)  
    alert(`Power ${n} is not supported`);  
  
// not bad  
if (n < 0) alert(`Power ${n} is not supported`);  
  
// good  
if (n < 0) {  
    alert(`Power ${n} is not supported`);  
}
```

- [추천] 에어비앤비 JavaScript 스타일 가이드 (<https://github.com/tipjs/javascript-style-guide>)

SECTION 1-3 알아두어야 할 기본 용어(1)

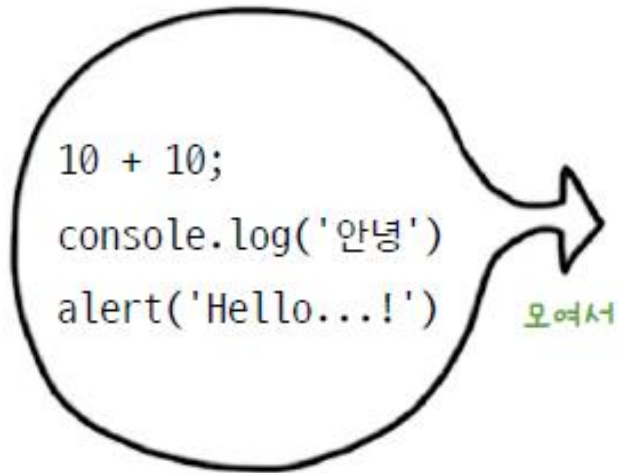
표현식

값을 만들어 내는 간단한 코드



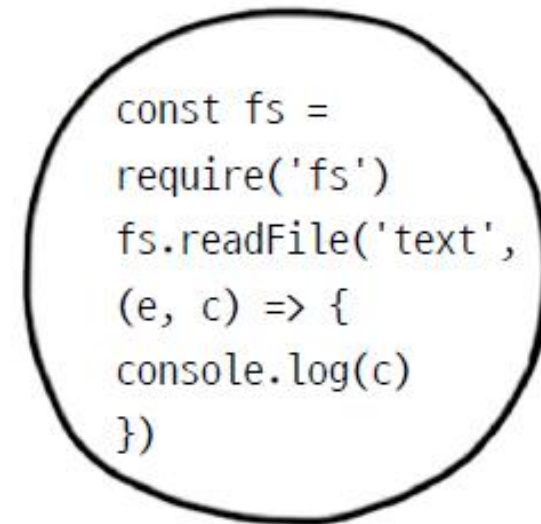
문장

표현식이 하나 이상 모인 것



프로그램

문장이 모인 것



SECTION 1-3 알아두어야 할 기본 용어(2)

- 표현식과 문장
 - 표현식: 자바스크립트에서 값을 만들어내는 간단한 코드
 - 문장: 하나 이상의 표현식이 모여 문장(statement)을 구성. 문장 끝에는 마침표를 찍듯이 세미콜론(;) 또는 줄바꿈을 넣어서 문장의 종결을 나타냄
 - 프로그램: 줄바꿈으로 문장을 구분해 코드를 작성
- 키워드: 자바스크립트가 처음 만들어질 때 정해놓은 특별한 의미가 있는 단어

| | | | |
|------------|---------|----------|----------|
| let | break | case | catch |
| class | const | continue | debugger |
| default | delete | do | else |
| export | extends | finally | for |
| function | if | import | in |
| instanceof | new | return | super |
| switch | this | throw | try |
| typeof | var | void | while |
| with | yield | | |

SECTION 1-3 알아두어야 할 기본 용어(3)

- 식별자: 프로그래밍 언어에서 이름을 붙일 때 사용하는 단어. 주로 변수명이나 함수명 등으로 사용
 - 키워드 사용 불가
 - 숫자로 시작 불가
 - 특수 문자는 _와 \$만 허용
 - 공백 문자를 포함할 수 없음
- 식별자를 만드는 일반적인 관례
 - 클래스(Chapter 9-1 참조)의 이름은 항상 대문자로 시작
 - 변수(Chapter 2-2 참조)와 인스턴스(Chapter 09-1 참조), 함수(Chapter 05-1 참조), 메소드(Chapter 06-1 참조)의 이름은 항상 소문자로 시작
 - 여러 단어로 이루어진 식별자는 각 단어의 첫 글자를 대문자 ex) class MyArray {}, let simpleString;
- 식별자의 종류

| 구분 | 단독으로 사용 | 다른 식별자와 사용 |
|--------------|---------|------------|
| 식별자 뒤에 괄호 없음 | 변수 | 속성 |
| 식별자 뒤에 괄호 있음 | 함수 | 메소드 |

Rectangle.width;

Rectangle.area();

SECTION 1-3 알아두어야 할 기본 용어(4)

- 주석: 프로그램 코드를 설명할 때 사용하며 프로그램 진행에는 전혀 영향을 주지 않음
 - HTML 태그 주석: <!-- -->로 문자열을 감싸 생성
 - 자바스크립트 주석
 - [방법1] //를 입력하는 것으로 한 줄 주석을 표현(// 뒤의 문장은 실행되지 않음)
 - [방법2] /*와 */를 입력하여 여러 줄 주석을 표현(/*와 */ 사이에 있는 모든 문장은 실행되지 않음)

```
<script>
// 주석은 코드 실행에 아무 영향을 미치지 않습니다.
/*
alert('Hello JavaScript')
alert('Hello JavaScript')
alert('Hello JavaScript')
*/
</script>
```


[좀 더 알아보기] 영어와 프로그래밍 언어

- 영어의 기본 형식

I love you → 주어 + 동사(일반 동사 또는 be 동사) + 목적어

- 프로그래밍 언어의 기본적인 형식

i.love(you) → 주어 + 동사(함수) + 목적어(매개변수)

- console.log() 메소드의 형식

console.log('Hello JavaScript...!') → 주어 + 동사(함수) + 목적어(매개변수)

- 프로그래밍 언어의 명령 표현

love(you) → 동사(함수) + 목적어(매개변수)

- alert() 함수의 형식

alert('Hello JavaScript...!') → 동사(함수) + 목적어(매개변수)

[요점 정리]

- 5가지 키워드로 정리하는 핵심 포인트
 - **표현식**이란 값을 만들어내는 간단한 코드
 - **문장**이란 하나 이상의 표현식이 모여 구성되는 것으로, 코드를 읽어 들이는 기본 단위
 - **키워드**란 프로그래밍 언어가 처음 만들어질 때 정해진 특별한 의미가 있는 단어
 - **식별자**란 변수나 함수에 이름을 붙일 때 사용하는 단어
 - **주석**은 프로그램 코드를 설명하는 문장으로, 프로그램 진행에는 전혀 영향을 주지 않음