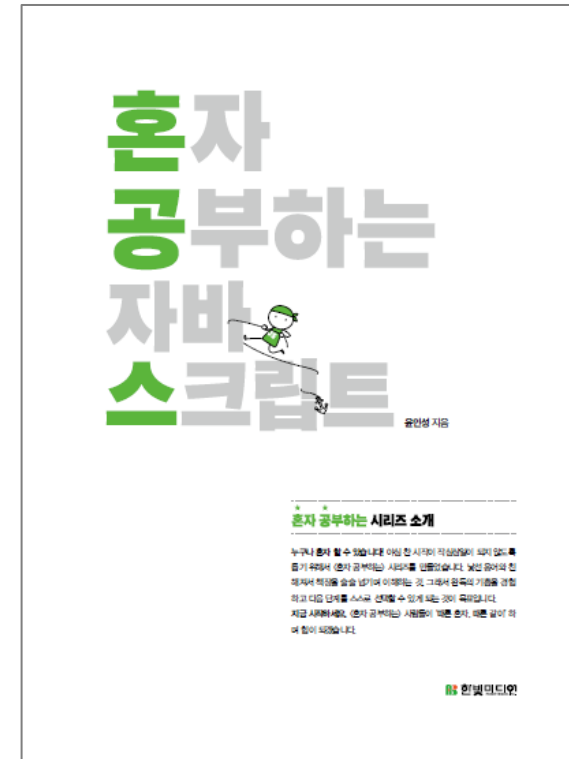


▶ Chapter 01 자바스크립트 개요와 개발환경 설정

혼자 공부하는 자바스크립트



이 책의 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 만들기과 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

- 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 → 자바스크립트의 등장으로 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.js(JavaScript 실행 환경: RunTime)가 등장하면서 자바스크립트만으로 웹 서버 애플리케이션 개발이 가능해짐
 - Node.js의 장단점(Non-Blocking, Single-Thread)
 - 웹 서버 애플리케이션을 개발할 때 꼭 필요한 간단한 모듈만 제공하므로 데이터 처리와 예외 처리 등이 다소 복잡
 - 빠른 속도로 서버 구매 비용과 유지 비용이 1/10 수준

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
 - 데스크톱 애플리케이션 개발
 - NW.js('노드웹킷 제이에스') : JavaScript와 HTML 그리고 Node 통합 환경을 제공
 - 일렉트론은 아톰 셸(Atom shell)이 발전한 것 : 깃허브(GitHub)용 자바스크립트 개발 전용 텍스트 에디터
 - 일렉트론 프레임워크를 사용하여 개발된 애플리케이션:
 - 마이크로소프트의 비주얼 스튜디오 코드(Visual Studio Code), 디스코드 (Discord) 클라이언트, slack, skype
 - Github 데스크톱 클라이언트, 워드프레스(Wordpress) 데스크톱 클라이언트, 몽고디비 (MongoDB),
 - 데이터 관리 도구 컴파스(Compass) 등
 - 데이터베이스 관리
 - MongoDB: 데이터베이스 관리에 자바스크립트를 활용하는 대표적인 NoSQL 데이터베이스

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 1-1-1 자바스크립트의 추가 기능(1/6)

- ECMAScript 2022(ES2022)표준에 변화된 내용 – 2022.06.22

1. Class Fields (클래스 필드)

- 접근 제어자가 필요한 JavaScript의 한계를 극복하고 클래스 선언과 관련된 코드들의 가독성과 지역성을 높이기 위해
 - Private 접근제어자 추가
 - Public 및 static 필드 선언방식 개선
 - Static 초기화 블록 추가

```
class NewClass {  
  // 클래스 내부에 바로 퍼블릭 필드 선언 가능  
  publicField = 0;      이전 퍼블릭 필드 설정 시 생성자내에서만 가능  
  publicMethod() {}    -> this.publicField = 0;  
  
  // 프라이빗 지원. 변수나 메소드 앞에 해시(#)를 붙임  
  #privateField = '비밀!';      이전 프라이빗 지원 안함  
  #privateMethod() {}  
  
  // 접근자도 프라이빗하게  
  get #privateGetter(){}      이전 메소드는 클래스 바디에서만 선언 가능  
  set #privateSetter(value){}
```


SECTION 1-1-1 자바스크립트의 추가 기능(2/6)

- ECMAScript 2022(ES2022)표준에 변화된 내용 – 2022.06.22

1. Class Fields (클래스 필드)

```
// 정적 필드도 클래스 내부에 선언 가능
static staticField = '정적필드';
static staticMethod(){}
```

이전 정적필드와 메소드는 클래스 외부에서만 설정 가능

```
// 정적 초기화블럭 선언 가능
static {
  console.log('정적 초기화 블럭에서 한 번만 실행됨');
  window.getPrivateField = (myClass) => myClass.#privateField;
}
```

```
// 위 내용의 조합도 가능
static #staticPrivateField = '정적 프라이빗 필드';
static get #staticPrivateMethod(){}
```

```
}
```

SECTION 1-1-1 자바스크립트의 추가 기능(3/6)

- ECMAScript 2022(ES2022)표준에 변화된 내용 – 2022.06.22
 2. Class Fields + Private field check (클래스 필드 + 프라이빗 필드 검사)
 - in 연산자를 활용한 프라이빗 필드 체크 가능

```
class OldClass {  
  #field;  
  // 예외를 이용해야만 프라이빗 필드 접근 가능  
  static isMyField(myClass) {  
    try {  
      myClass.#field;  
      return true;  
    } catch {  
      return false;  
    }  
  }  
}
```

```
class NewClass {  
  #field;  
  
  static isMyField(myClass) {  
    // in 연산자 사용  
    return #field in myClass;  
  }  
}
```

in 연산자를 사용하여 프라이빗 필드 접근

SECTION 1-1-1 자바스크립트의 추가 기능(4/6)

- ECMAScript 2022(ES2022)표준에 변화된 내용 – 2022.06.22

3. Regexp Match Indices (정규표현식 플래그 d)

- d 플래그를 추가하여 패턴과 일치하는 문자의 시작과 끝 위치 정보를 획득 가능

```
const target = "xaaaz"
const regExp = /a+(?<Z>z)?/g;

const m1 = regExp.exec(target);

console.log(m1);          // (2) ['aaaz', 'z', index: 1, input: 'xaaaz', groups{Z: 'z'}]
console.log(regExp.lastIndex); // 5
```

```
const target = "xaaaz"
const regExp = /a+(?<Z>z)?/dg;    d flag를 추가하여 group 캡처의 인덱스 정보 획득 가능

const m1 = regExp.exec(target);
```

```
console.log(m1);          // (2) ['aaaz', 'z', index: 1, input: 'xaaaz', groups{Z: 'z'}]
// indices: [[1,5], [4,5], groups: {z: [4,5]}]
```

```
const matchedStr = target.slice(...m1.indices[0]);
const matchedZGroupStr = target.slice(...m1.indices.groups["Z"]);
```

indices의 첫 번째 배열['aaaz']은 전체 문자열의 시작, 종료 인덱스이며
두 번째 배열['z']은 캡처링 group의 시작, 종료 인덱스를 의미함

```
console.log(matchedStr);    // aaaz
console.log(matchedZGroupStr); // z
```

SECTION 1-1-1 자바스크립트의 추가 기능(5/6)

- ECMAScript 2022(ES2022)표준에 변화된 내용 – 2022.06.22

4. Top-level await (모듈 최상위 레벨의 await 호출 가능)

- 비동기 처리를 위해 사용되는 **async/await**는 한 세트(async 내부에서 await 사용)이기 때문에, **await** 혼자서는 동작이 불가능하나 ES2022부터 최상위 레벨에서 **await** 을 단독 사용할 수 있도록 함
- Top-level에서 **await** 을 사용함으로써 Top-level **await** 모듈이 **async** 함수처럼 동작하게 함

5. Array.prototype.at()

- 배열에 Negative Indexing 을 가능하게 해주는 at()

```
const Kim = ["Abc", "Def", "Ghi"];
```

```
console.log(Kim[Kim.length-1]);  
console.log(Kim.slice(-1)[0]);  
console.log(Kim.reverse()[0]);
```

Ghi

Ghi

Ghi

```
console.log(Kim.at(-1));
```

Abc

```
console.log(Kim.at(-2));
```

Def

```
console.log(Kim.at(-4));
```

undefined

reverse() 로 인하여 Kim.at(-1)이 Abc 가 됨

SECTION 1-1-1 자바스크립트의 추가 기능(6/6)

- ECMAScript 2022(ES2022)표준에 변화된 내용 – 2022.06.22

6. Object.hasOwn() 허용

- Object.prototype.hasOwnProperty() 의 문제점을 해결하고 보다 간결하게 사용하기 위해 고안
- Overriding에 의한 property shadowing이 발생할 경우 문제 발생
- Object.prototype을 상속받지 못할 경우 문제 발생

```
const person = {  
  |  lname: "Kim"  
}  
  
const hasOwnProperty = Object.prototype.hasOwnProperty;  
console.log(hasOwnProperty.call(person, "lname")); // return true  
  
console.log(Object.hasOwn(person, "lname")); // return true, 2022년 타입
```

7. Error Cause

- Error Cause 는 에러의 원인을 파악하기 쉽도록 추가한 기능

```
const newError = new Error(ErrorMessage, {cause: ErrorCause}); ErrorCause 타입은 모든 타입  
  
console.log(newError.cause); // ErrorCause, 2022년 타입
```

SECTION 1-1-1 자바스크립트의 추가 기능(1/2)

- ECMAScript 2023(ES2023)표준에 변화된 내용 – 2023.04.

1. findLast(), findLastIndex() (배열 뒤에서부터 찾기)

```
<script>
  const isEven = (number) => number % 2 === 0;
  const numbers = [1, 2, 3, 4];

  console.log(numbers.find(isEven));           // 2
  console.log(numbers.findIndex(isEven));      // 1
  console.log(numbers.findLast(isEven));       // 4
  console.log(numbers.findLastIndex(isEven));  // 3
</script>
```

2. HashBang Comments (유닉스 쉘 스크립트에서 사용되는 문법) 적용 - Js 파일의 맨 처음이 #! 로 시작할 경우 해당 라인을 무시

SECTION 1-1-1 자바스크립트의 추가 기능(2/2)

- ECMAScript 2023(ES2023)표준에 변화된 내용 – 2023.04.

3. WeakMap Keys에 심벌 적용 가능 – 이전에는 객체만 새로 생성이 가능했으나 심벌에도 적용 가능

```
<script>
  const weak = new WeakMap();
  const keyObj = {};
  const key = Symbol("ref");
  weak.set(keyObj, "value");           // value
  weak.set(key, "ECMAScript 2023");   // ECMAScript 2023
  console.log(weak.get(keyObj));
  console.log(weak.get(key));         // 2023부터 가능
</script>
```

4. toReversed(), toSorted(), toSpliced(), with() 추가 (배열을 복사 후 작업)

```
<script>
  const original = [1, 4];
  const spliced = original.toSpliced(1,2,3,5);
  // 인덱스 1부터 2개 삭제 3, 5 추가
  console.log(original);           // [1, 4]
  console.log(spliced);             // [1, 3, 5]
</script>
```

```
<script>
  const original = [1, 2, 2, 4];
  const withThree = original.with(2,3);
  // 중복 제거 후 추가
  console.log(original);           // [1, 2, 2, 4]
  console.log(withThree);          // [1, 2, 3, 4]
</script>
```

[좀 더 알아보기] 모바일 애플리케이션의 종류

- 네이티브 앱: 제조사가 추천하는 프로그래밍 언어를 사용해서 만들어진 애플리케이션
 - 아이폰: 오브젝티브-C(Objective-C)
 - 안드로이드폰: 자바(Java) 프로그래밍
- 모바일 웹 앱
 - 웹사이트 화면을 애플리케이션으로 감싸기만 해서 보여줌
- 하이브리드 앱
 - 스마트폰의 기능과 웹 페이지를 연결할 수 있는 층을 설치해서 웹사이트가 스마트폰의 기능을 활용
 - 쿠팡, 위메프 등의 쇼핑 애플리케이션 – 유지보수가 쉬우나 오프라인 작동 불가
- 프로그레시브 웹 앱
 - 하이브리드 앱과 유사하나 다운로드나 인스톨 과정 없이 기본 앱처럼 사용 가능
 - 오프라인 사용 가능하며 디바이스들의 하드웨어기능도 사용 가능, 다양한 플랫폼에서 실행 가능
 - 리액트(react)나 앵귤러(Angular)만 있으면 개발 가능
- 리액트 네이티브
 - 하나의 프로그램을 만들어서 여러 프로그램으로 만들어주는 크로스 플랫폼 엔진 또는 프레임워크
 - 페이스북, 인스타그램, 핀터레스트, 디스코드, 스카이프 등

[마무리] 모바일 애플리케이션의 종류

- 3가지 키워드로 정리하는 핵심 포인트

- 자바스크립트란 웹 브라우저에서 작동하는 프로그래밍 언어
- ECMAScript란 유럽컴퓨터제조협회에서 표준화한 자바스크립트의 공식 명칭
- 웹 애플리케이션이란 기존의 웹 페이지보다 많은 기능을 구현한 웹 페이지

- 확인 문제

1. 인터넷을 돌아다니면서 보았던 쉽게 사용할 수 있고, 기능이 많다고 느꼈던 웹 사이트를 5개 적어 보기
2. Statcounter에서 책을 보고 있는 현재 시점의 웹 브라우저 점유율(Browser Market Share Worldwide)을 확인

[참조] Statcounter 통계 페이지

<http://gs.statcounter.com/browser-version-market-share/desktop/south-korea>

SECTION 1-2 개발환경 설치와 코드 실행(1)

- 개발환경에는 코드를 작성하는 텍스트 에디터와 코드를 실행하는 코드 실행기가 필요
 - 이 책에서는 텍스트 에디터는 비주얼 스튜디오 코드 Visual Studio Code를 코드 실행기는 구글 크롬 웹 브라우저를 사용
- 구글 크롬 설치하기
- 비주얼 스튜디오 코드 설치하기
 - 비주얼 스튜디오 홈페이지: <https://code.visualstudio.com>
 - 한국어 언어팩 설치하기

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 (알파벳 '아이')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...!") Enter ➡ 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

- 3단계: HTML 페이지 실행하기

- 01: 생성한 test.html 파일 실행 준비

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

```
1 <!DOCTYPE html>
2 <html lang="en">
3 <head>
4   <meta charset="UTF-8">
5   <meta http-equiv="X-UA-Compatible" content="IE=edge">
6   <meta name="viewport" content="width=device-width, initial-scale=1.0">
7   <title>Document</title>
8   <script>
9     |   alert('Hello World')
10  </script>
11 </head>
12 <body>
13
14 </body>
15 </html>
```


[좀 더 알아보기①] 오류를 확인하는 방법

- 내가 무엇을 잘못 입력했는지 알아내는 방법과 찾는 방법

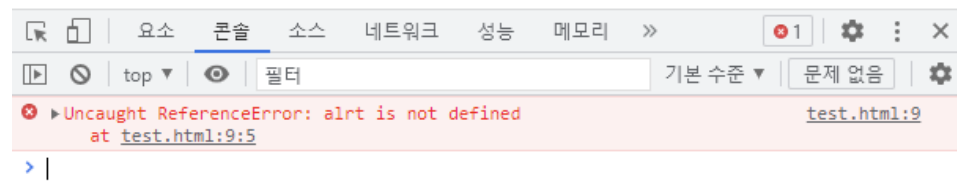
```
<script>
  alrt('Hello World')
</script>
```



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 : 9은 오류가 발생한 위치. [test.html : 9]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
 - 중괄호 입력 방식
 - 키워드 뒤에 공백

[마무리]

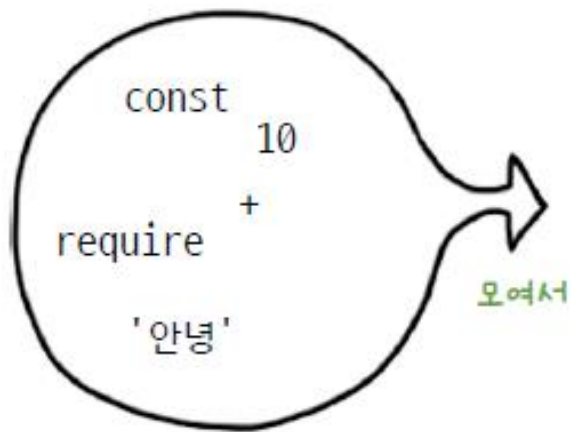
- 3가지 키워드로 정리하는 핵심 포인트
 - 개발환경이란 개발을 할 수 있는 환경을 의미
 - 텍스트 에디터란 코드를 작성할 수 있는 프로그램이며, 이 교재에서는 텍스트 에디터로 비주얼 스튜디오 코드를 사용
 - 구글 크롬 개발자 도구란 구글 크롬이 개발자를 위해 오류 확인 등의 기능을 제공하는 도구
- 확인 문제
 1. 구글 크롬 개발자 도구의 콘솔을 실행하고 다음 명령을 입력했을 때 나오는 결과를 적어 보기(코드를 하나 실행할 때 여러 줄의 출력이 나오는 경우 모두 적어야 함)
 - > "안녕하세요"
 - > console.log("안녕하세요")
 - > "안녕하세요"
 2. 비주얼 스튜디오 코드에 다음 소스 코드를 입력하고 ex01.html로 저장한 후 화면에 나오는 결과를 적어 보기

```
<body>
<script>
document.body.innerHTML = "<h1>안녕하세요</h1>"
</script>
</body>
```

SECTION 1-3 알아두어야 할 기본 용어(1)

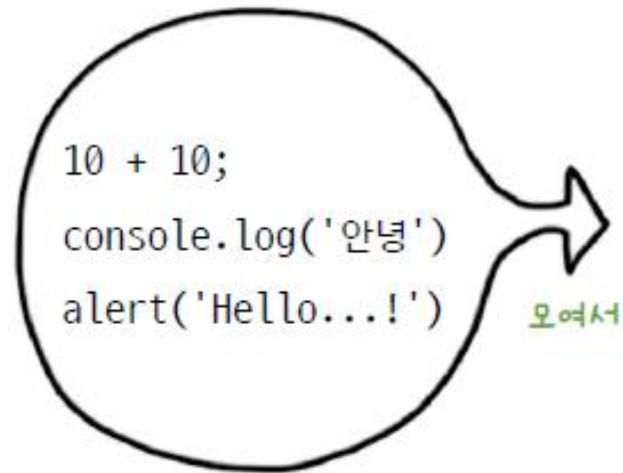
표현식

값을 만들어 내는 간단한 코드



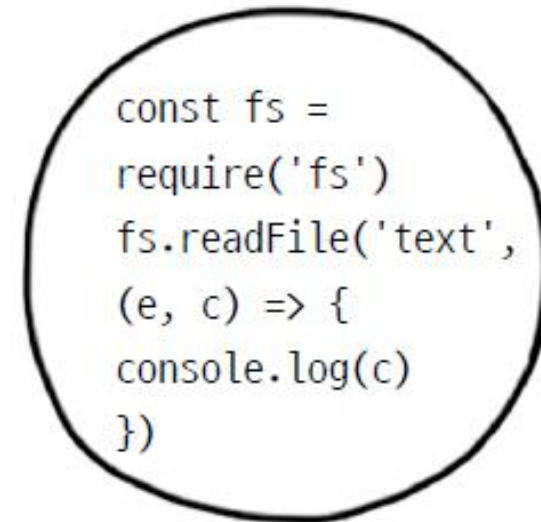
문장

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

await	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	let	static

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(camel 표기법)
- 식별자의 종류

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

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>
```

SECTION 1-3 알아두어야 할 기본 용어(5)

- 출력: 자바스크립트는 다른 프로그래밍 언어와 비교해서 출력 방법이 많고 복잡한 편
 - 간단한 표현식 결과 확인하기
 - 01: 구글 크롬의 주소창에 about:blank를 입력해 빈 페이지로 들어가 단축키 Ctrl + Shift+ I (알파벳 '아이')를 눌러서 개발자 환경을 띄우기
 - 02: about:blank에서 [Console] 탭을 클릭해 구글 크롬 개발자 도구에 진입. 이곳에 어떤 값을 입력하면 곧바로 그 결과가 출력
 - 경고창에 출력하기
 - 개발 전용 에디터를 사용할 때의 출력하는 방법
 - alert() 함수를 사용하여 웹 브라우저에 경고창을 띄우기
 - 콘솔에 출력하기
 - console.log() 메소드 사용

```
<script>  
alert('Hello JavaScript...!')  
</script>
```

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

[마무리①]

- 5가지 키워드로 정리하는 핵심 포인트
 - 표현식이란 값을 만들어내는 간단한 코드
 - 문장이란 하나 이상의 표현식이 모여 구성되는 것으로, 코드를 읽어 들이는 기본 단위
 - 키워드란 프로그래밍 언어가 처음 만들어질 때 정해진 특별한 의미가 있는 단어
 - 식별자란 이름을 붙일 때 사용하는 단어
 - 주석은 프로그램 코드를 설명하는 문장으로, 프로그램 진행에는 전혀 영향을 주지 않음
- 확인 문제
 1. 다음 단어 중 식별자로 사용할 수 있는 것은 O표, 식별자로 사용할 수 없는 것은 X표
① a () ② hello () ③ 10times () ④ _ () ⑤ \$ ()
 2. console.log()에서 console은 다음 중 무엇일까?
①키워드 ②식별자 ③연산자 ④메소드
 3. console.log()에서 log는 다음 중 무엇일까?(중복 선택 가능)
①키워드 ②식별자 ③연산자 ④메소드

[마무리②]

- 확인 문제

4. 여러 단어로 이루어진 식별자 만들기 (식별자를 만드는 일반적인 관례 참조)

① we are the world ()

② create output ()

③ create request ()

④ init server ()

⑤ init matrix ()

5. 다음 코드를 입력해보고 어떤 오류가 뜨는지 확인하기

① `konsole.log('안녕하세요')`

② `+++ 1 ++ 2 + 3`

③ `console.log)`