
JavaScript



목차

a table of contents

- 1 자바스크립트 기초
- 2 자료와 변수
- 3 조건문
- 4 반복문
- 5 함수

목차

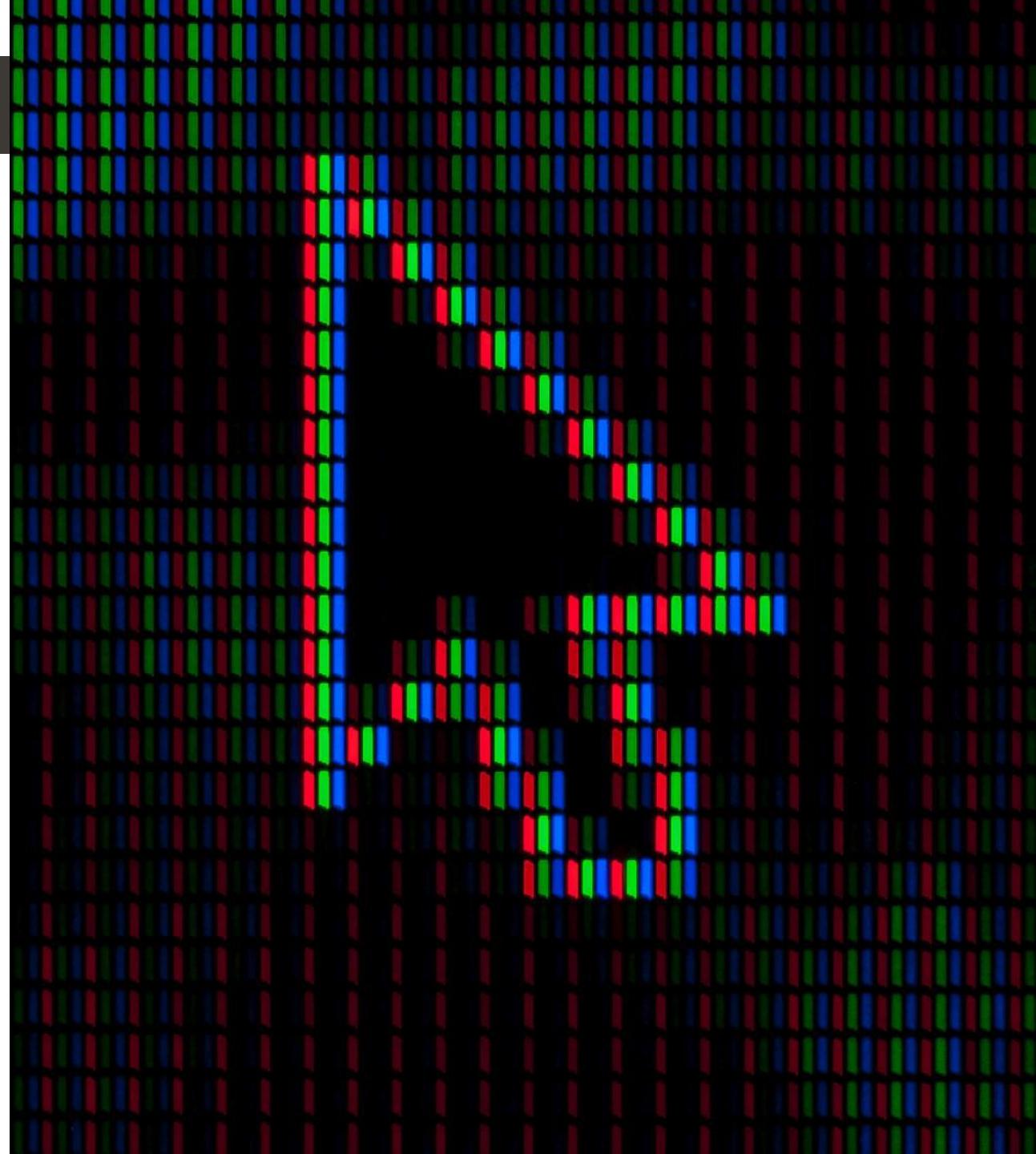
a table of contents

6 객체

7 문서 객체 모델

8 예외 처리

9 클래스



- 1 자바스크립트 기초**
- 2 자료와 변수
- 3 조건문
- 4 반복문
- 5 함수

A close-up photograph of a person's hands holding a white smartphone. The person is wearing a white button-down shirt. A white rectangular overlay box is positioned in the center of the image, containing the text.

JavaScript 활용



vs



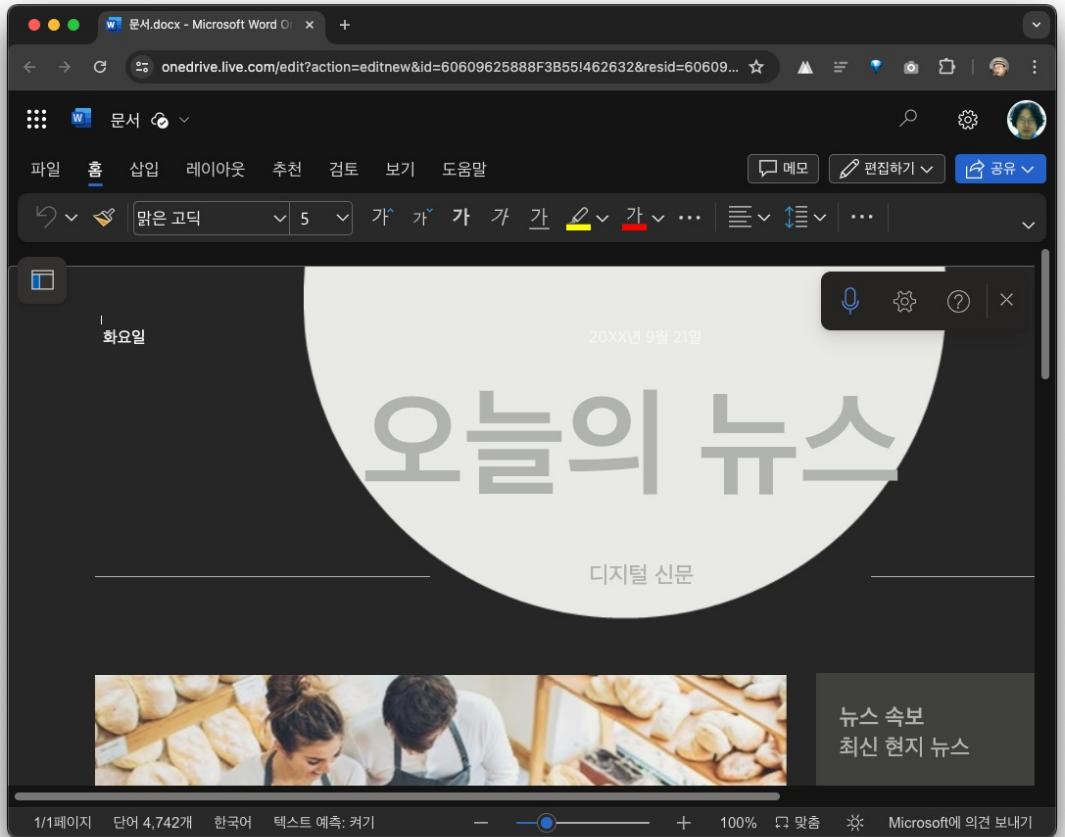
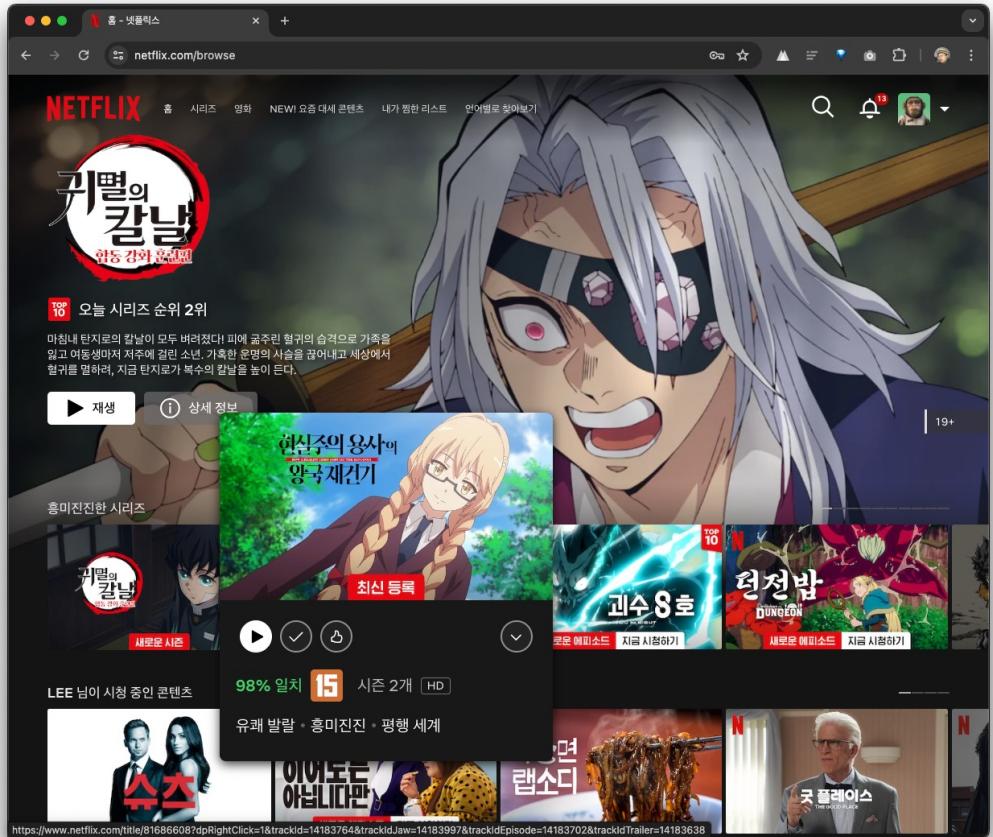
웹 클라이언트 애플리케이션 개발

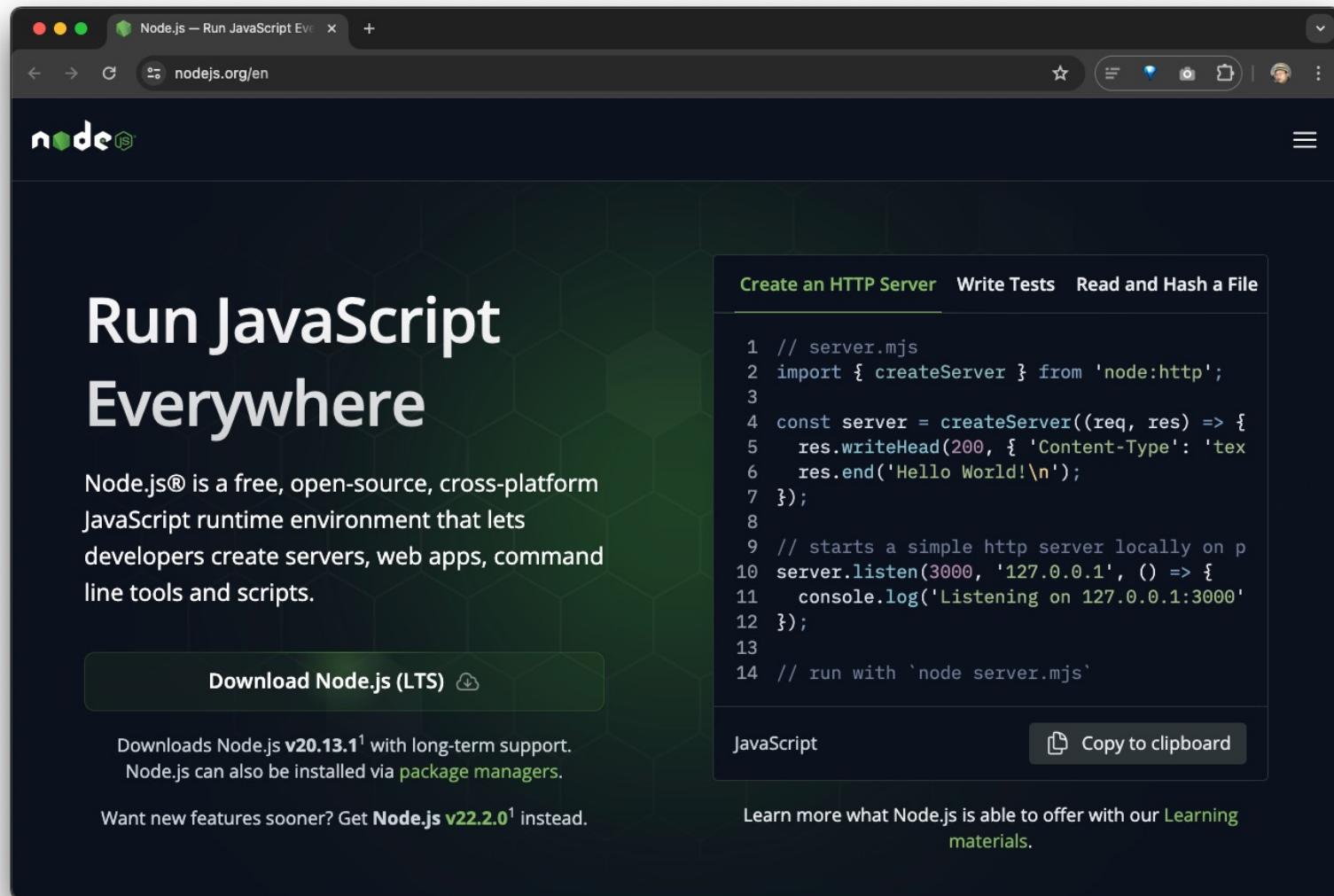
The screenshot shows the classic Yahoo! homepage with a red banner at the top. Below it, there's a search bar with a dropdown menu for 'advanced search' and 'most popular'. A sidebar on the left lists various services: New!, Shop, Find, Connect, Organize, Fun, and Info. The main content area features a 'Yahoo! Games' section with a Rubik's cube graphic, a 'Personal Assistant' section with a 'Sign up' button, and a 'In The News' section with a list of headlines. At the bottom, there's a 'Yahoo! Personals' section with a photo of a couple and a 'Find My Match!' button, along with a 'Web Site Directory' and sections for Business & Economy, Computers & Internet, Regional, Society & Culture, and Shopping.

The screenshot shows the Altavista homepage with its signature blue mountain logo and the tagline 'THE SEARCH COMPANY'. A search bar at the top has dropdown menus for 'any language' and 'Search'. Below the search bar, there are links for 'Help', 'Customize Settings', and 'Family Filter is off'. The main content area includes a 'Search for:' field, a 'Search Assistant | Advanced Search' link, and several categorized links: Shopping, Tools, News, Web Site Hosting, Arts & Entertainment, Music, Business Center, and Autos. There are also sections for People & Chat and a link to download a trial version of AltaVista Enterprise Software.

Part 1

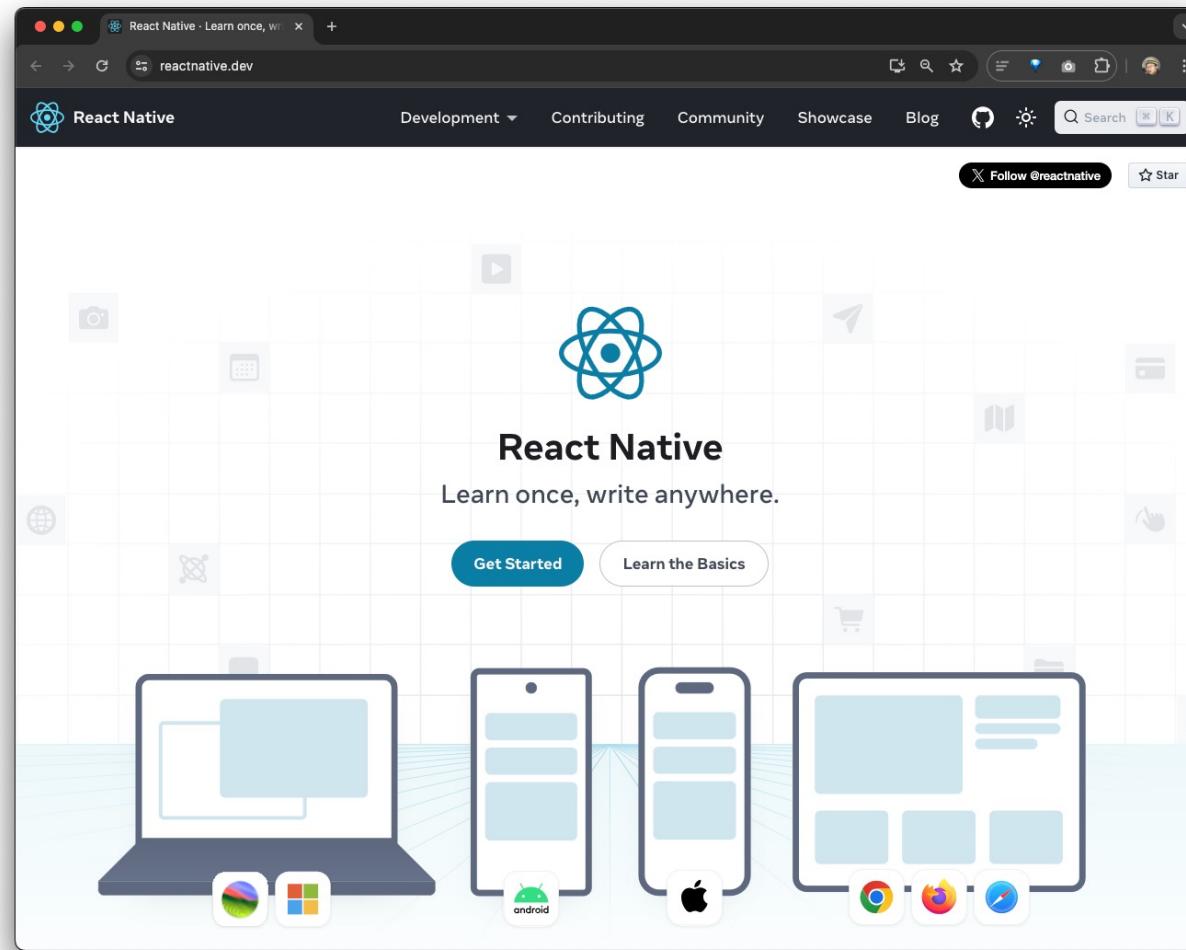
웹 클라이언트 애플리케이션 개발





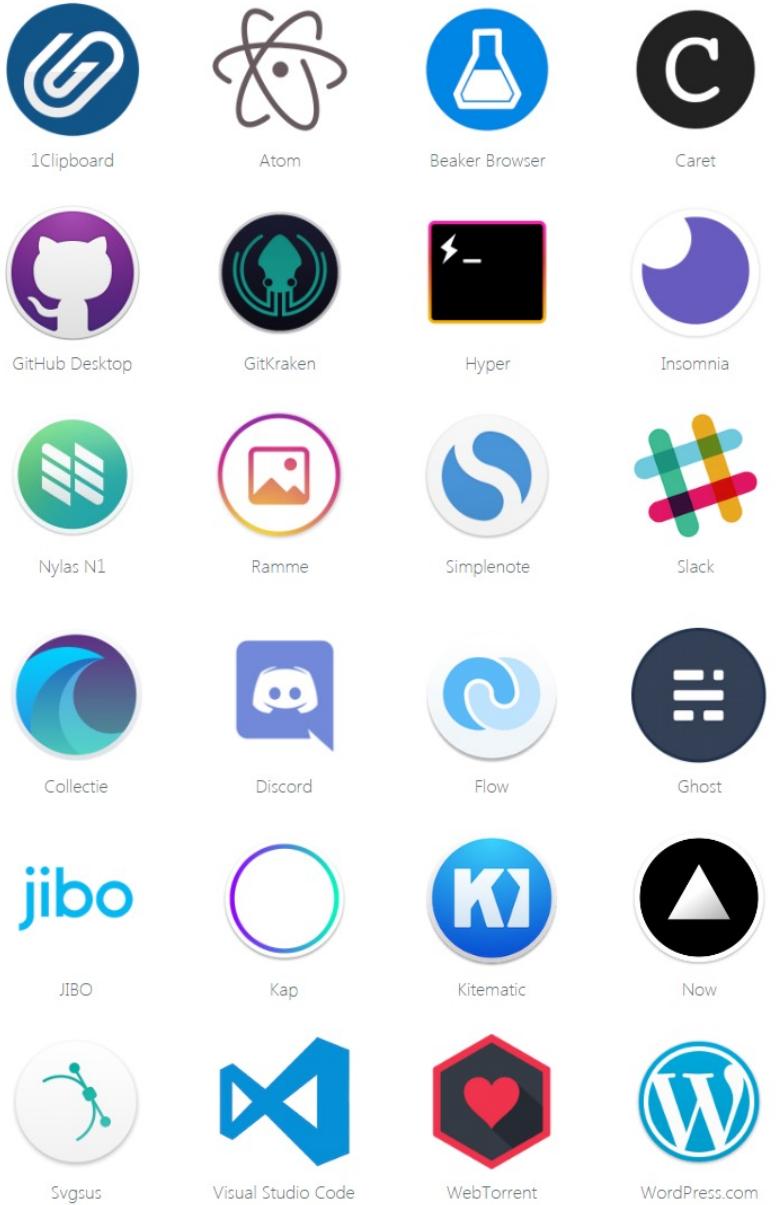
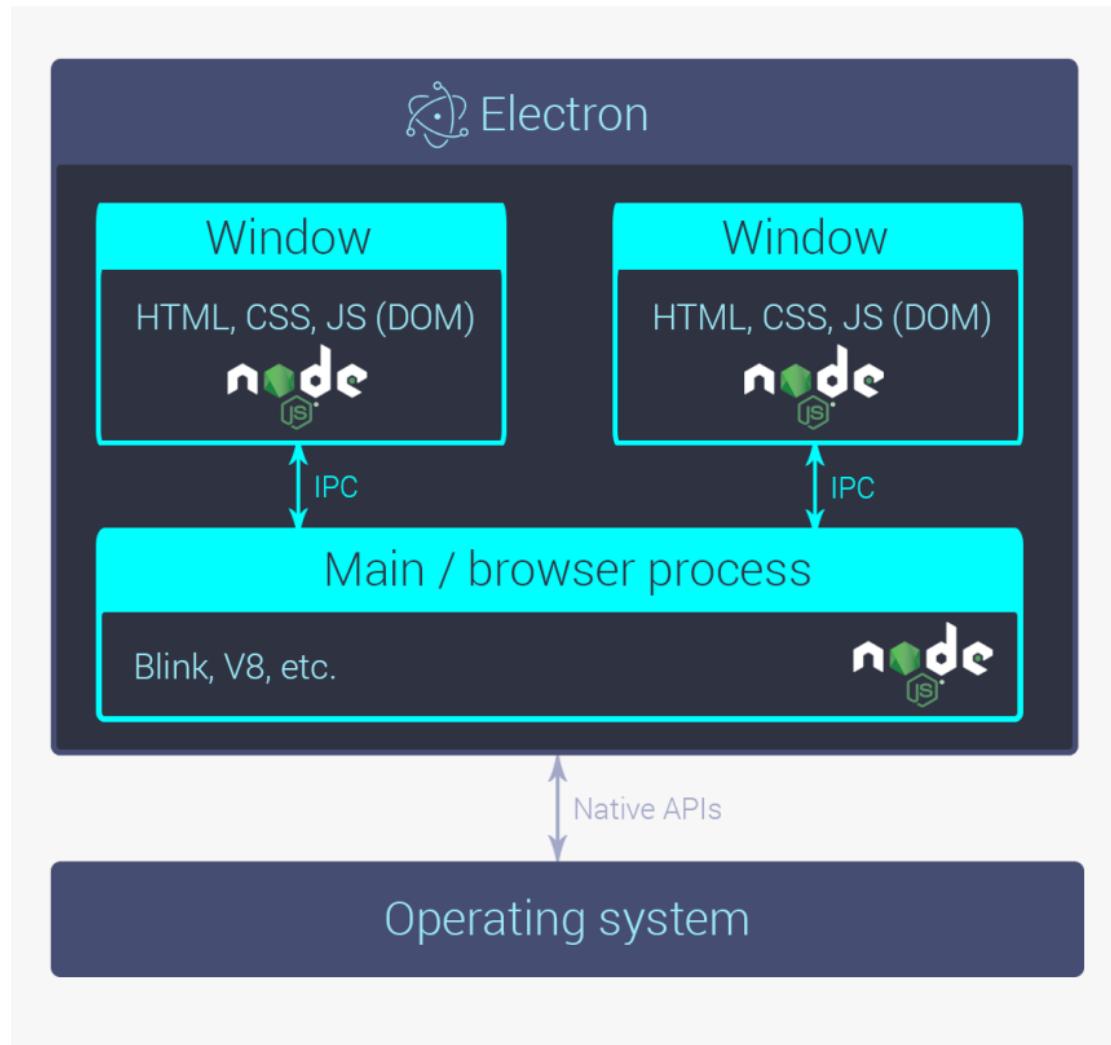
Part 1

모바일 애플리케이션 개발



Part 1

데스크톱 애플리케이션 개발



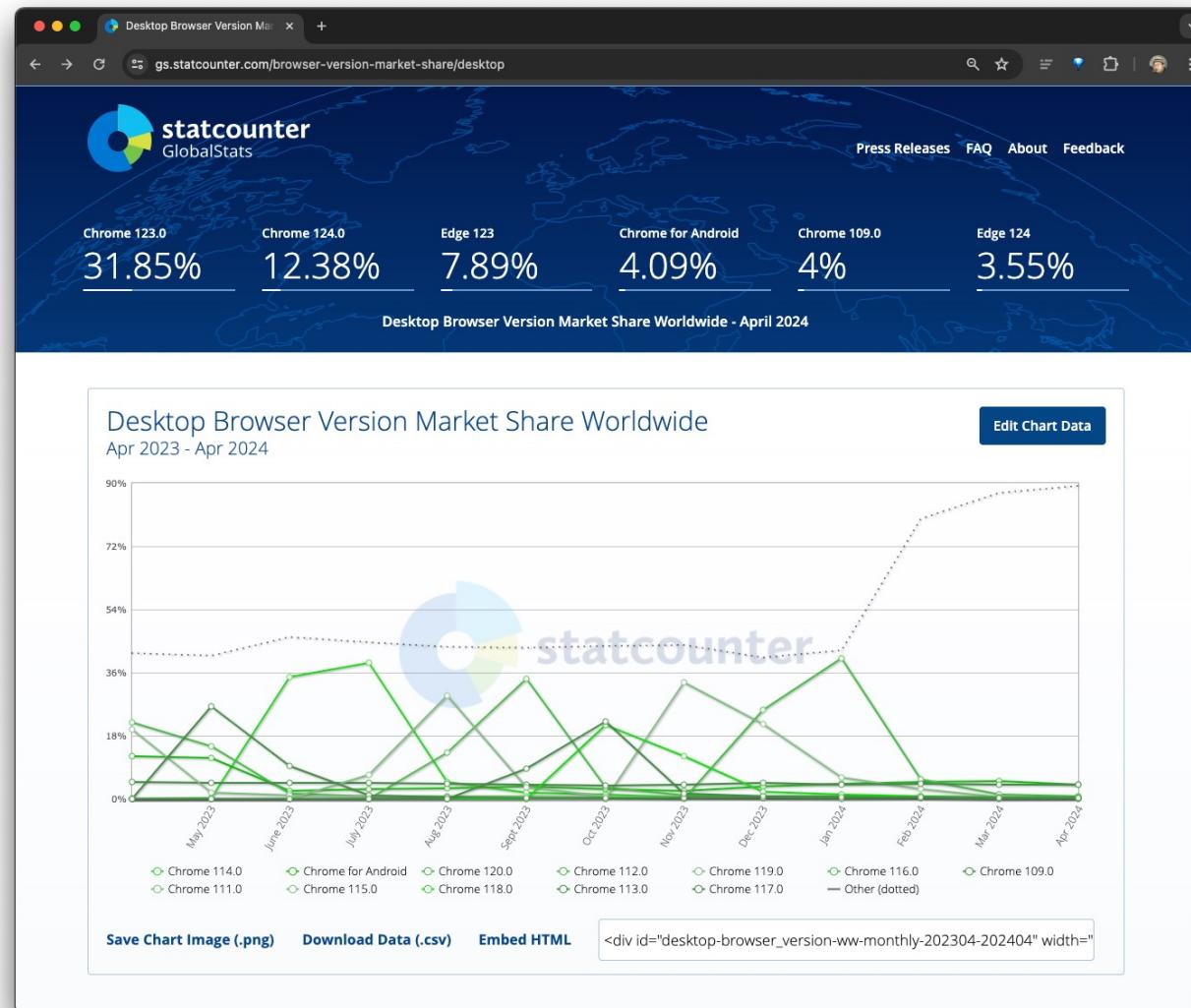
The screenshot shows the MongoDB website homepage. At the top, there's a navigation bar with links for Products, Resources, Solutions, Company, Pricing, a search bar, language selection (Eng), Support, Sign In, and a prominent green "Try Free" button. The main headline reads "Built by developers, for developers" followed by the subtext "The document data model maps to how you think and code." Below this, there are three vertical columns: "Model", "Query", and "Optimize". The "Query" column contains text about using an expressive query API and performing semantic search. To the right of these columns is a code editor window showing a snippet of JavaScript code for a vector search. The code uses the MongoDB aggregate pipeline to perform a vector search on an "orders" collection. A "Contact Us" button is located at the bottom right of the code editor.

```
0 // Query on embedded documents
1
2 const cursor =
3 db.collection('orders').aggregate([
4   "$vectorSearch": {
5     "index" : "default",
6     "path" : "name_embedding",
7     "queryVector" : query_embedding,
8     "numCandidates" :150,
9     "limit" :10
10 });
11
```

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 2023	2023년 6월

Part 1

자바스크립트의 종류



Part 1

모바일 애플리케이션의 종류



네이티브 앱

VS



크로스 플랫폼

VS



하이브리드 앱

VS



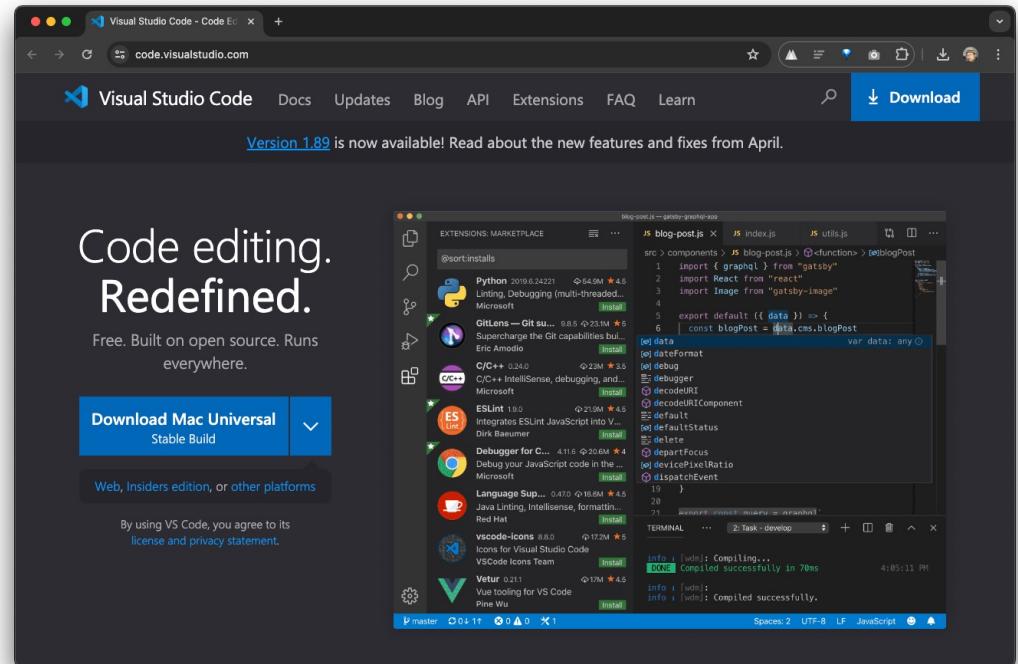
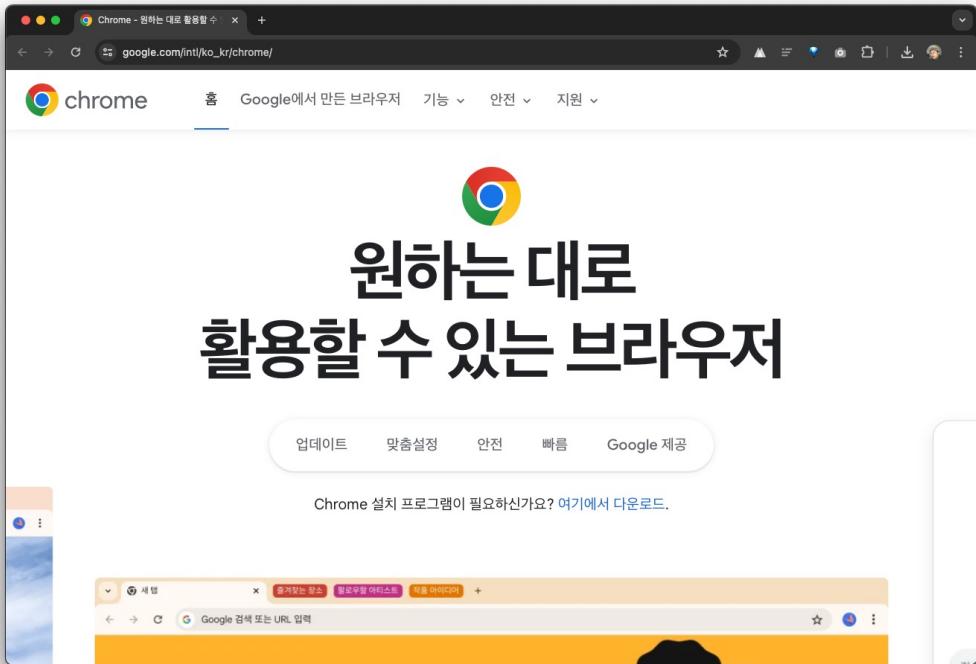
웹 앱

A close-up photograph of a person's hands holding a white smartphone. The person is wearing a white button-down shirt. The background is blurred.

개발환경 설정

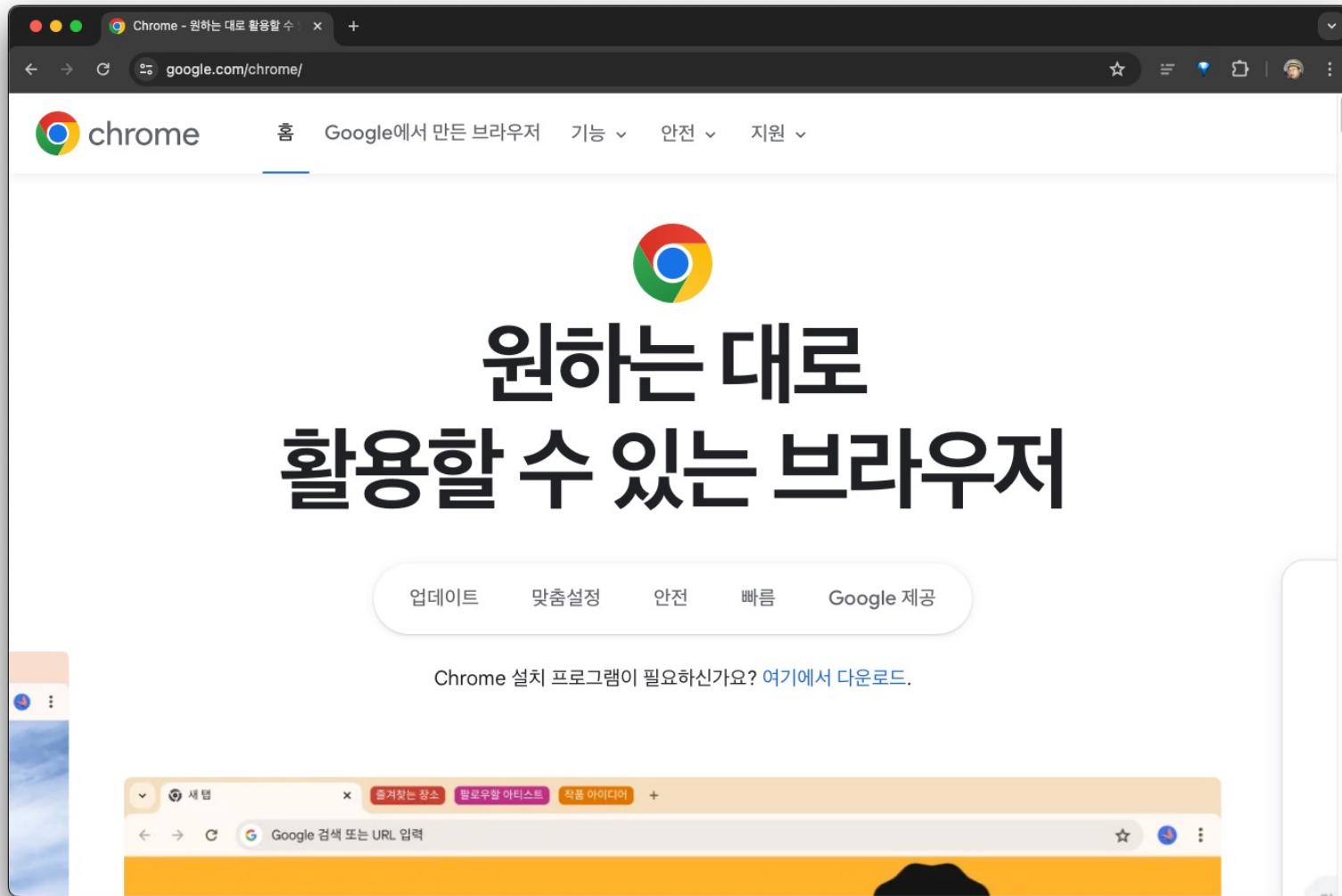
Part 1

개발환경 설치와 코드 실행



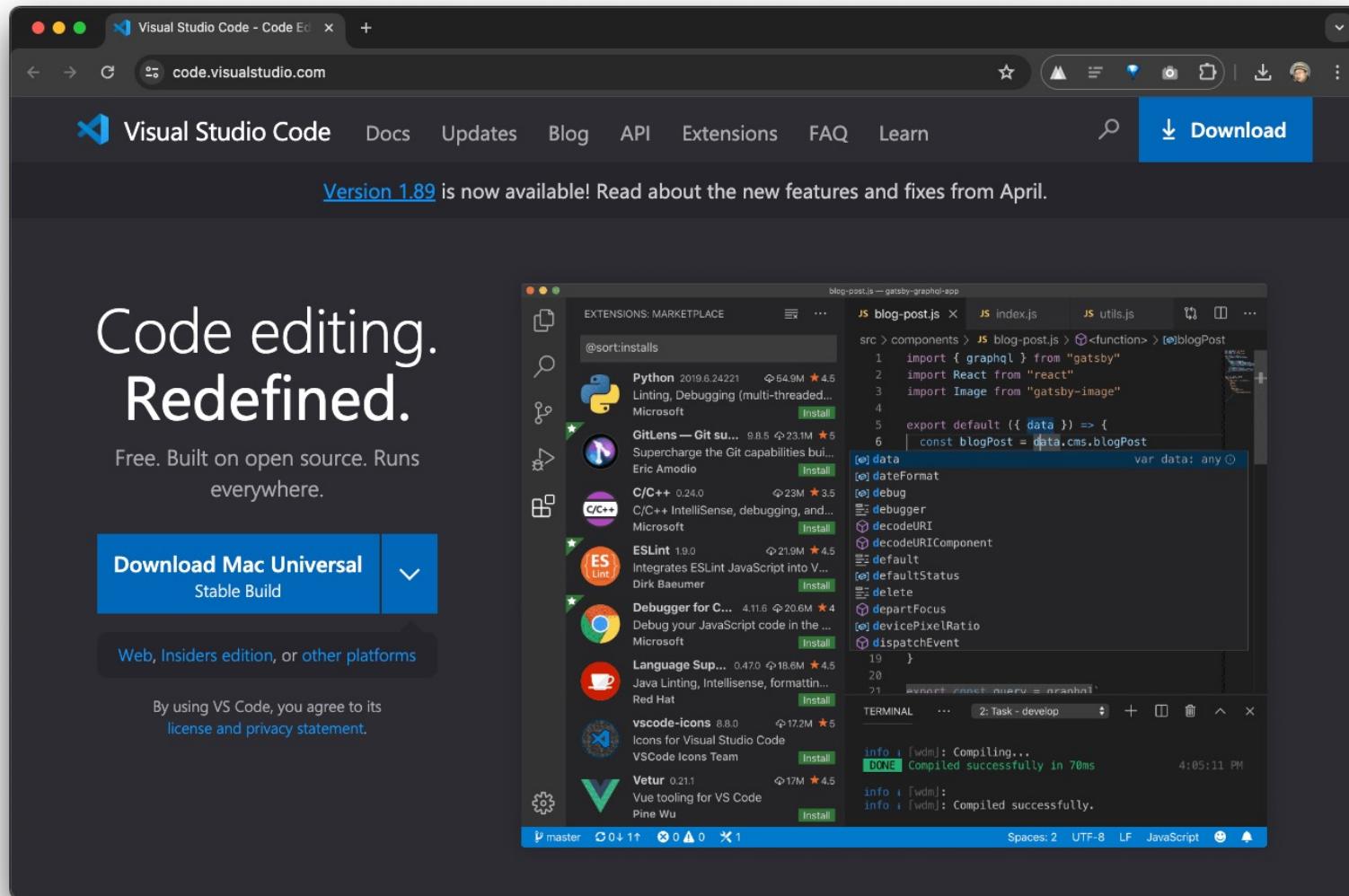
Part 1

구글 크롬 설치



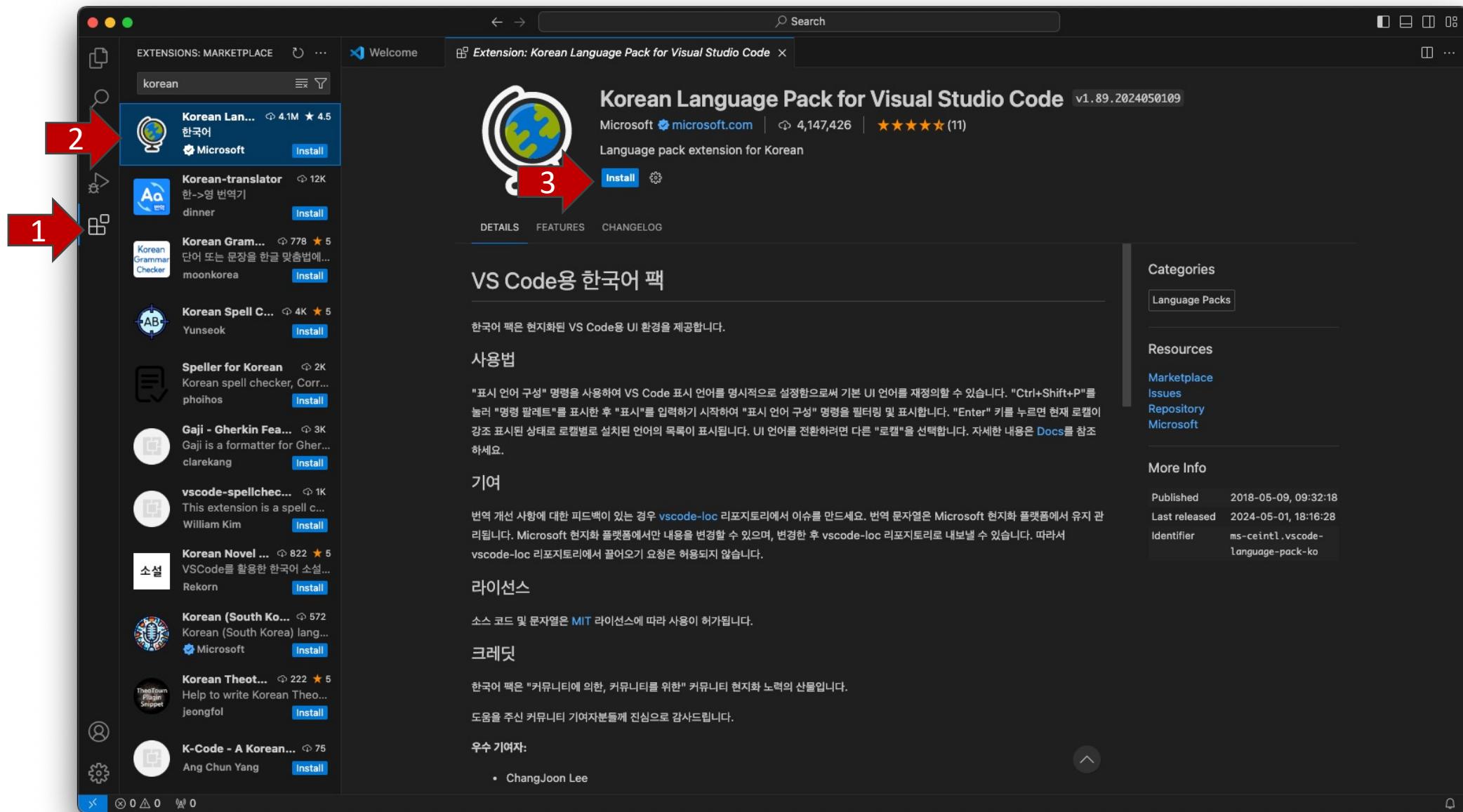
Part 1

Visual Studio Code 설치



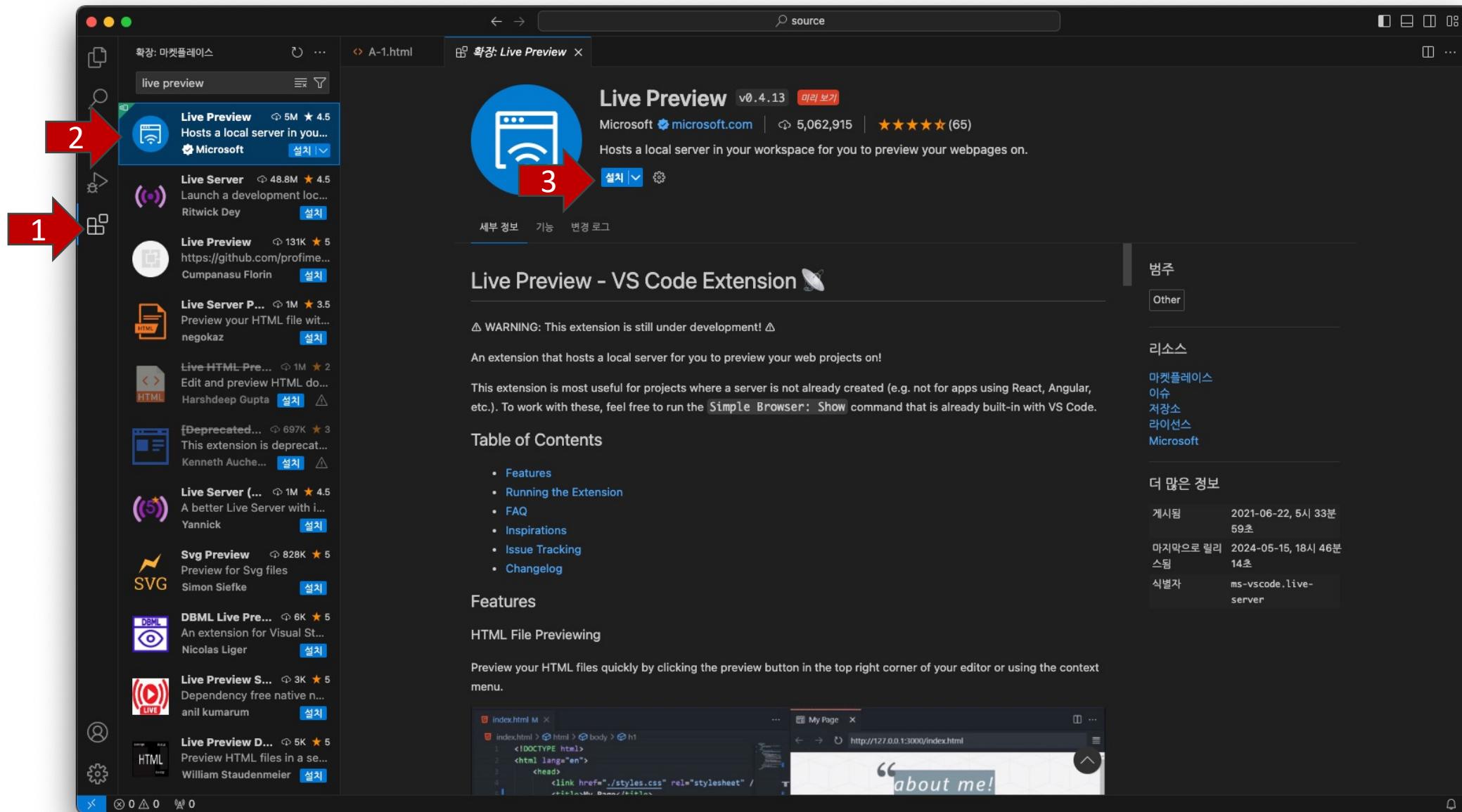
Part 1

Visual Studio Code 플러그인 설치



Part 1

Visual Studio Code 플러그인 설치



Part 1

HTML 페이지 작성

The screenshot shows a dark-themed code editor interface. In the top left, there are window control buttons (red, yellow, green) and a search bar labeled "source". On the far right, there are icons for zooming and closing. The left sidebar contains icons for file operations (new, open, save, etc.), search, and other tools. A file list on the left shows "HTMLPage.html" with a preview icon and a count of 1. The main editor area has the following content:

```
1 html
```

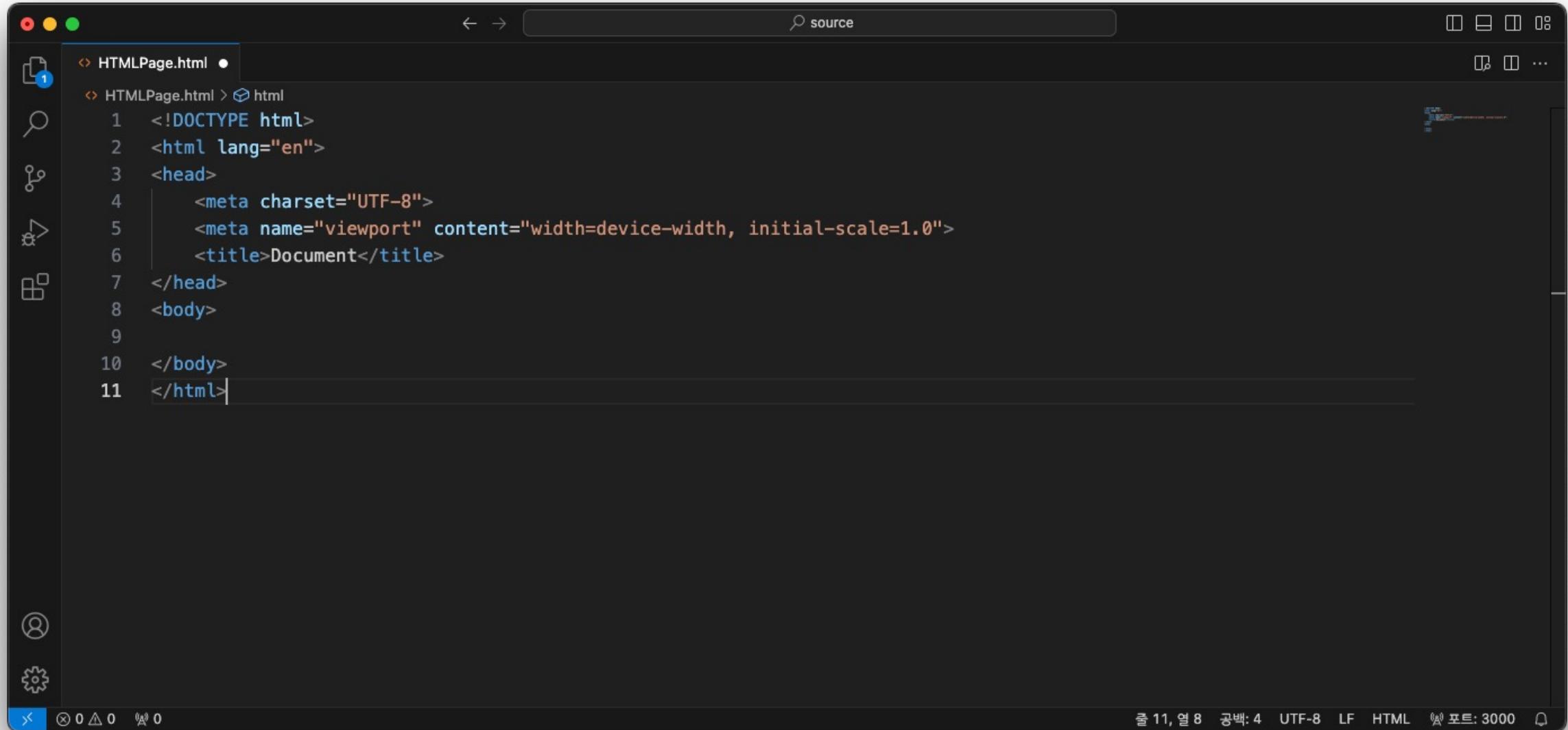
Below the content, a dropdown menu titled "Emmet Abbreviation" is open, showing three options:

- html
- html:5
- html:xml

The bottom status bar displays file information: 줄 1, 열 5, 공백: 4, UTF-8, LF, HTML, 포트: 3000, and a bell icon.

Part 1

HTML 페이지 작성



The screenshot shows a dark-themed code editor window with a sidebar on the left containing various icons for file operations like new file, save, search, and settings. The main area displays an HTML document titled 'HTMLPage.html'. The code is as follows:

```
1  <!DOCTYPE html>
2  <html lang="en">
3  <head>
4      <meta charset="UTF-8">
5      <meta name="viewport" content="width=device-width, initial-scale=1.0">
6      <title>Document</title>
7  </head>
8  <body>
9
10 </body>
11 </html>
```

The status bar at the bottom indicates the file is saved (S), has 0 changes (△ 0), and is 0 bytes (0). It also shows the current date (Jul 11, 2018), time (8:44 PM), encoding (UTF-8), line separator (LF), file type (HTML), and port (3000).

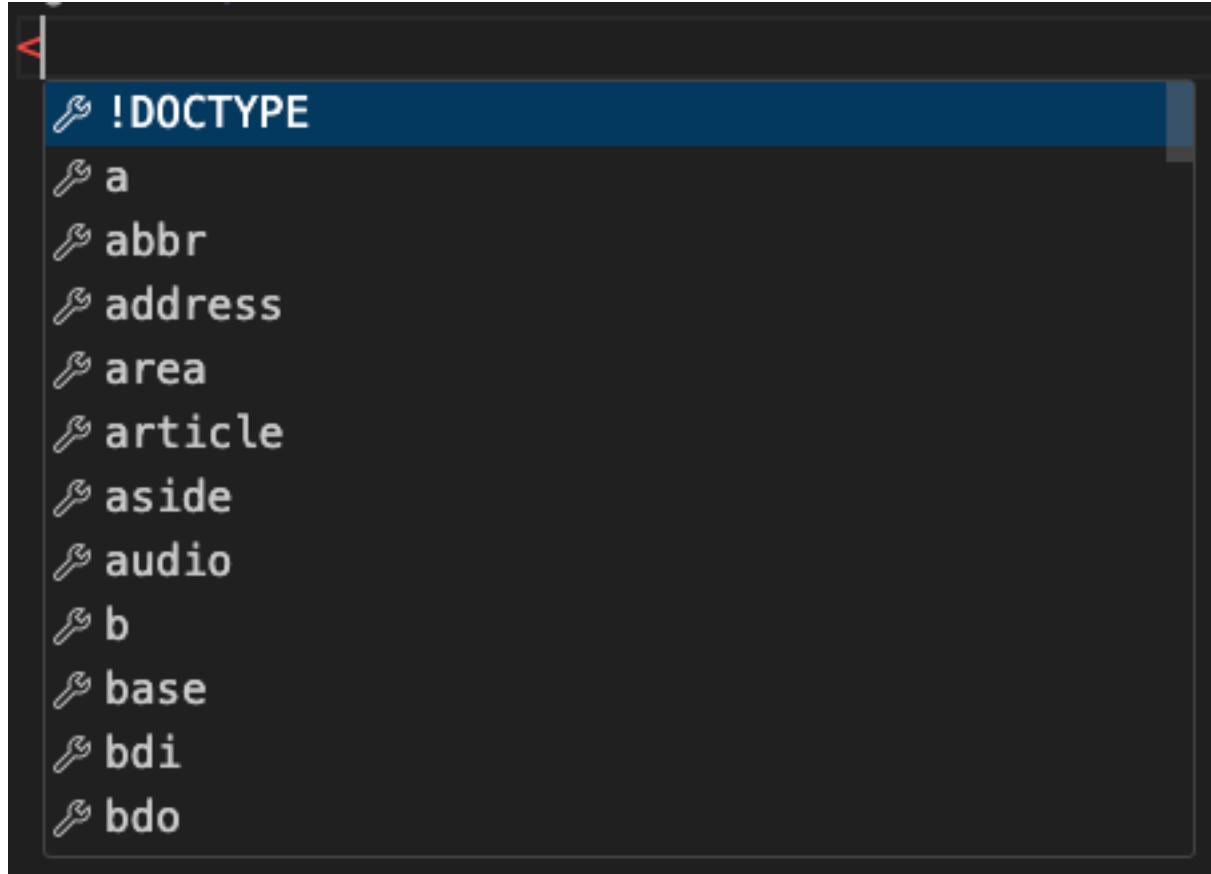
Part 1

HTML 페이지 작성

The screenshot shows a dark-themed web browser window. On the left, the file tree displays a single file named "HTMLPage.html". The main content area shows the source code of the file:

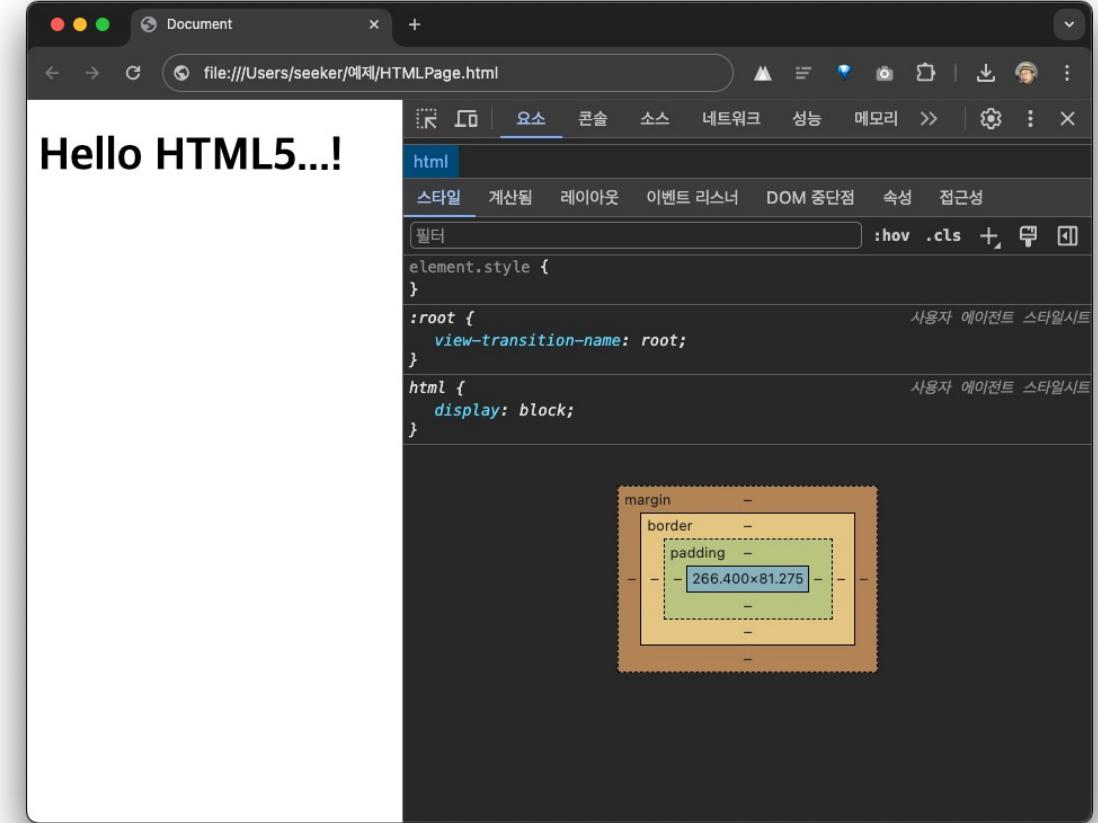
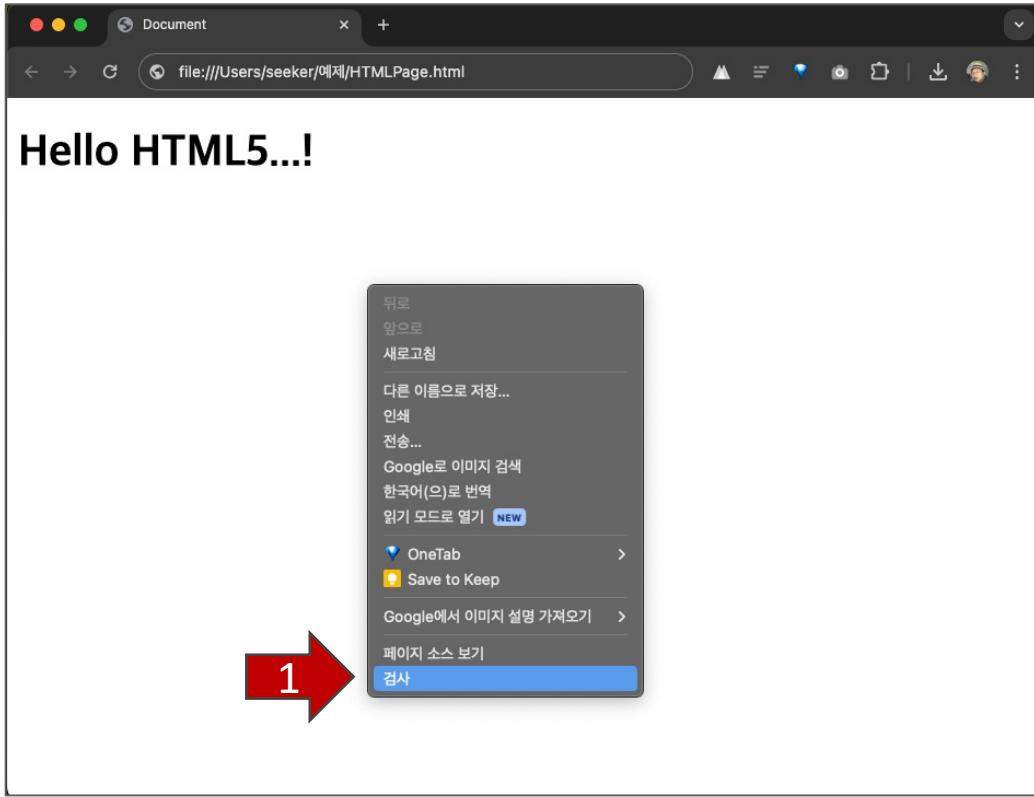
```
1  <!DOCTYPE html>
2  <html>
3  <head>
4  |   <title>Document</title>
5  </head>
6  <body>
7  |   <h1>Hello HTML5...!</h1>
8  </body>
9  </html>
```

On the right, the browser's status bar indicates the URL `http://127.0.0.1:3000/HTMLPage.html`. The rendered output of the page is displayed in the main pane, showing the text "Hello HTML5...!" in a large, bold, black font.



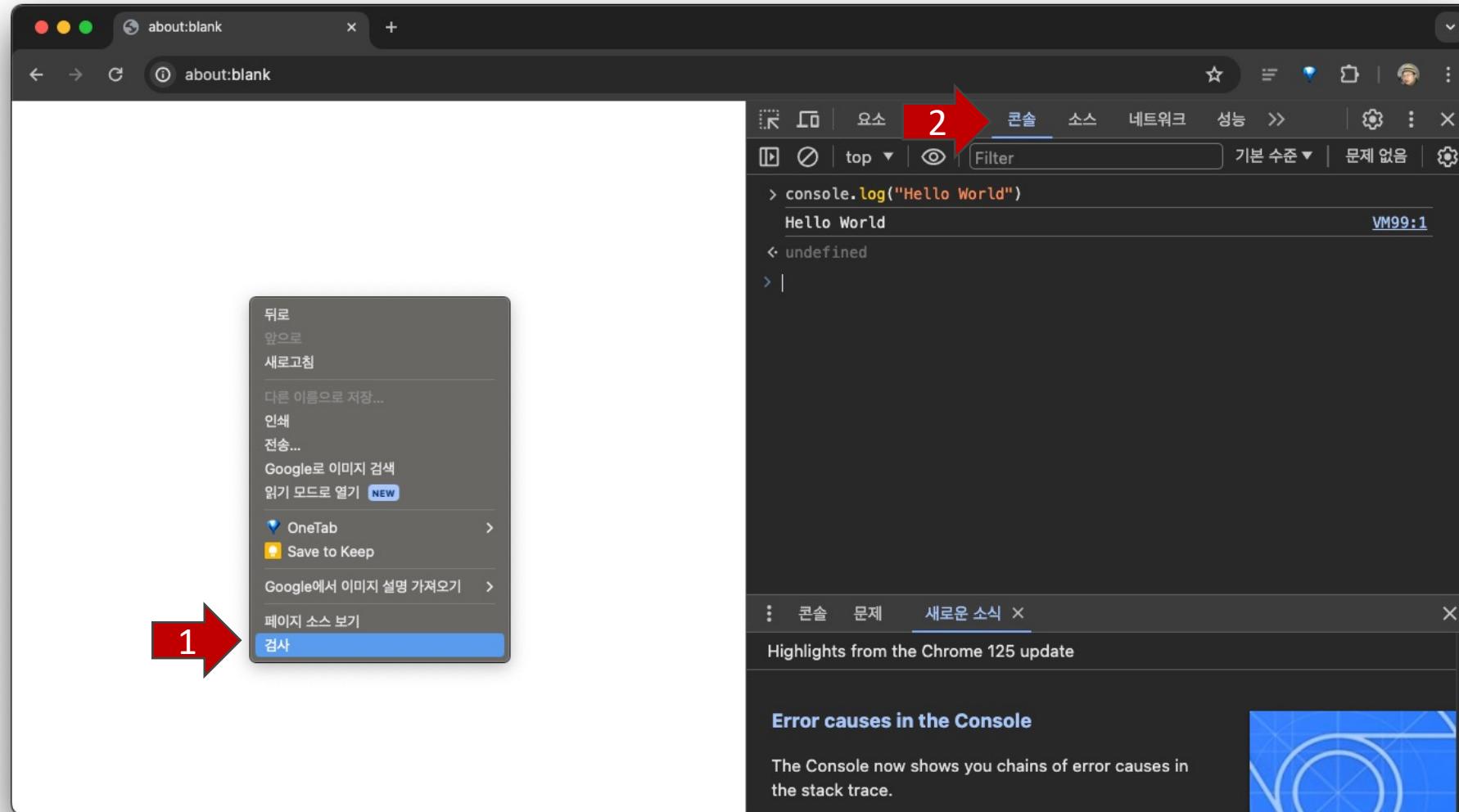
Part 1

구글 크롬 개발자 도구



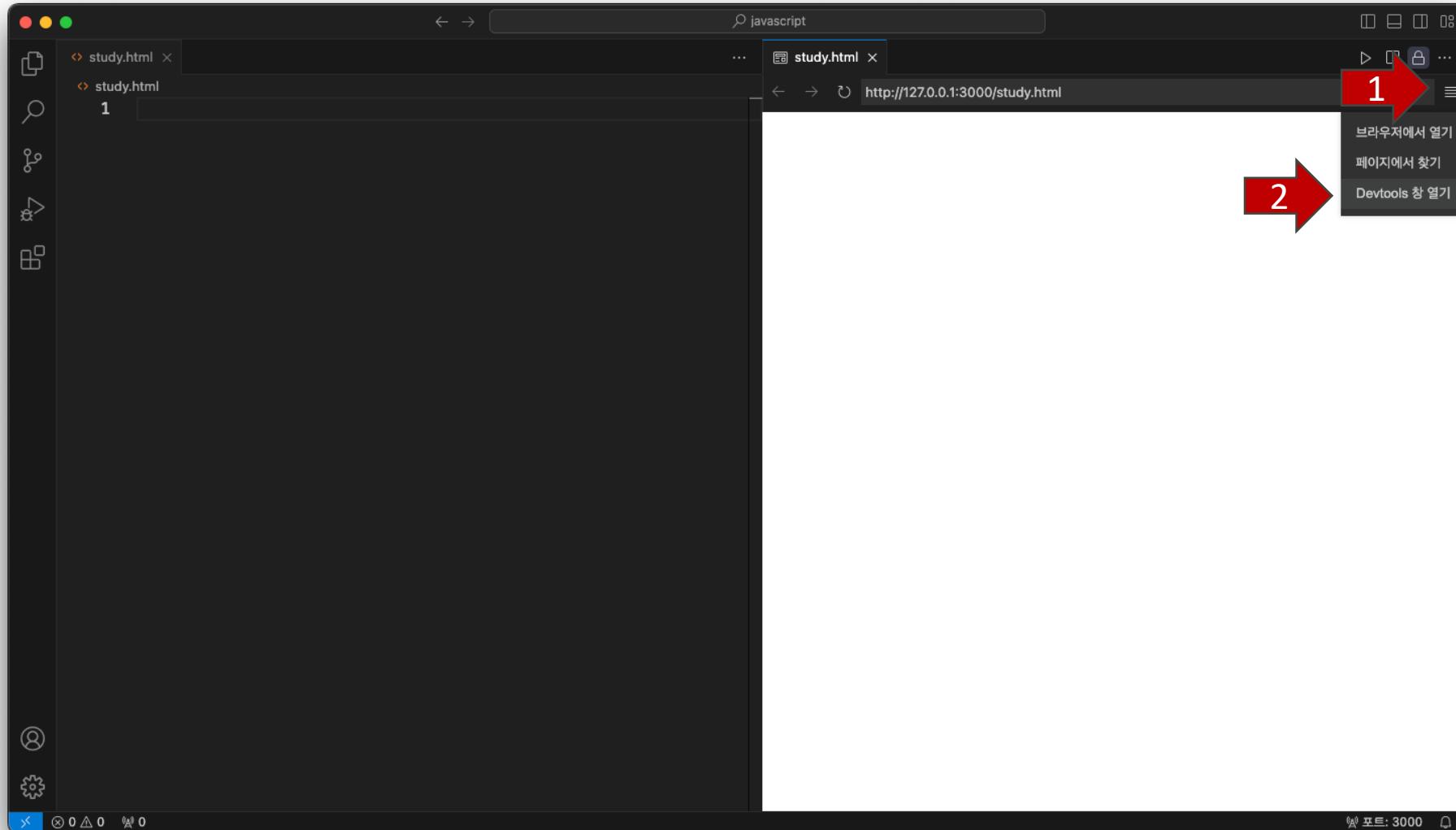
Part 1

구글 크롬 개발자 도구



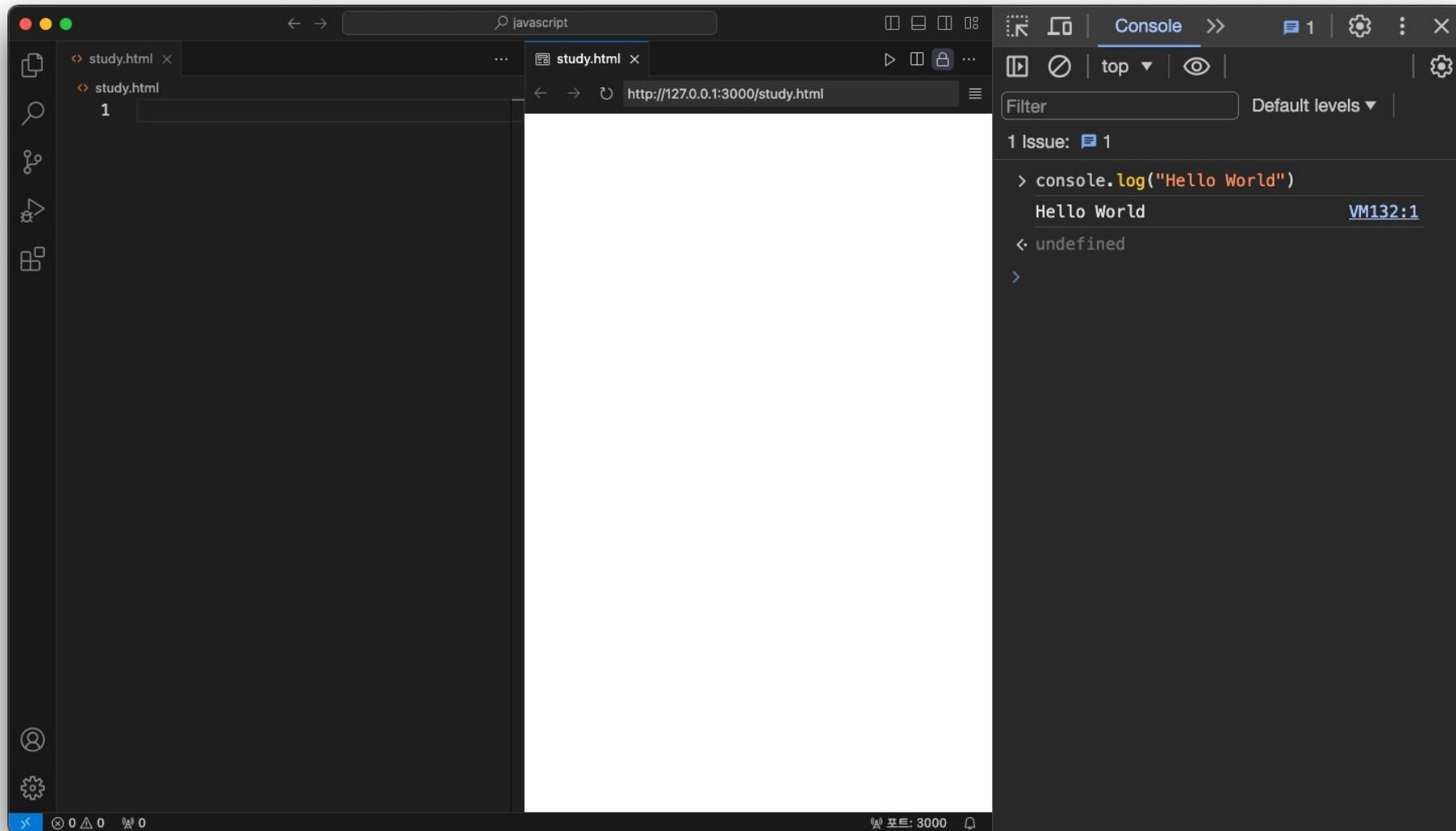
Part 1

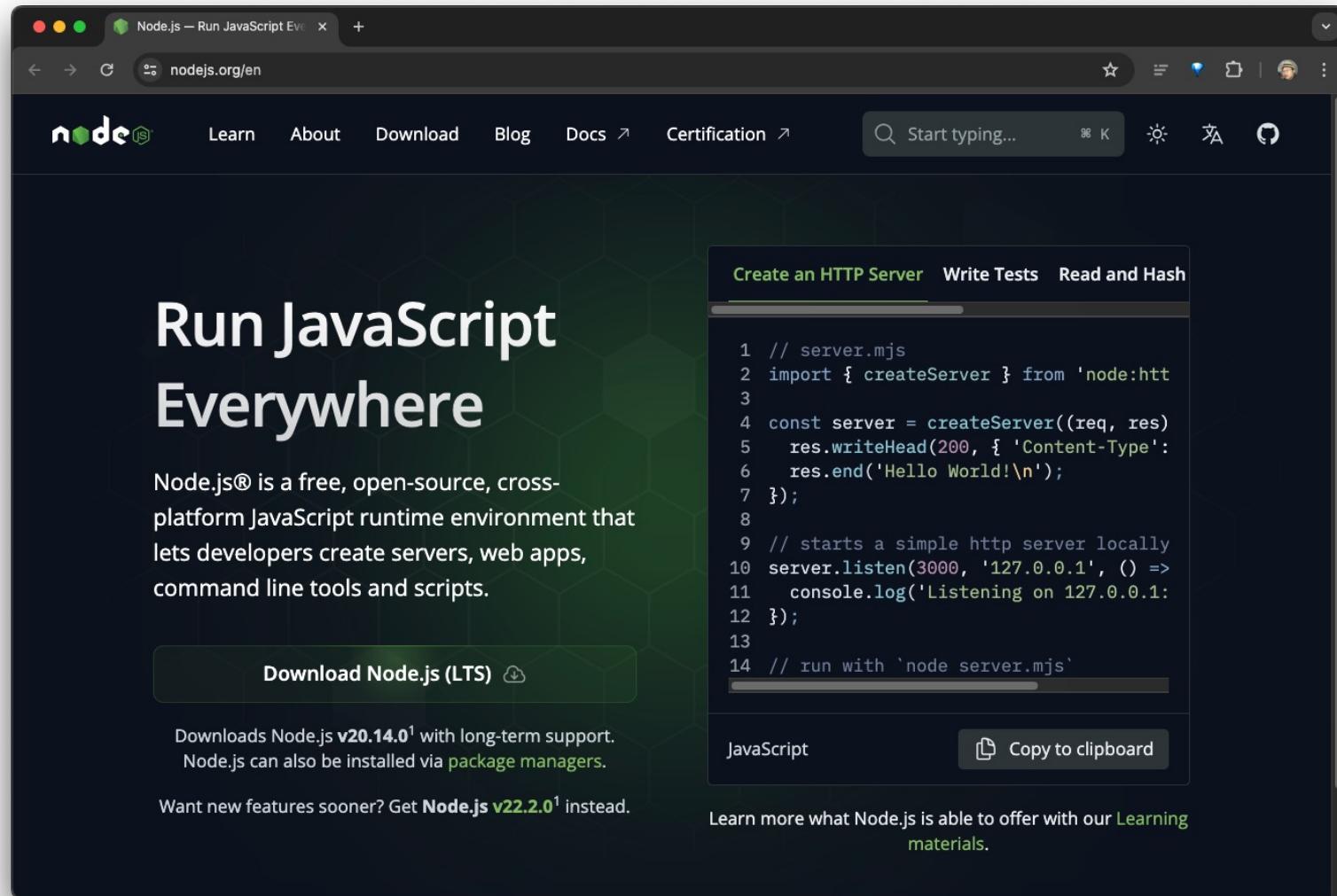
Live Preview 개발자 도구



Part 1

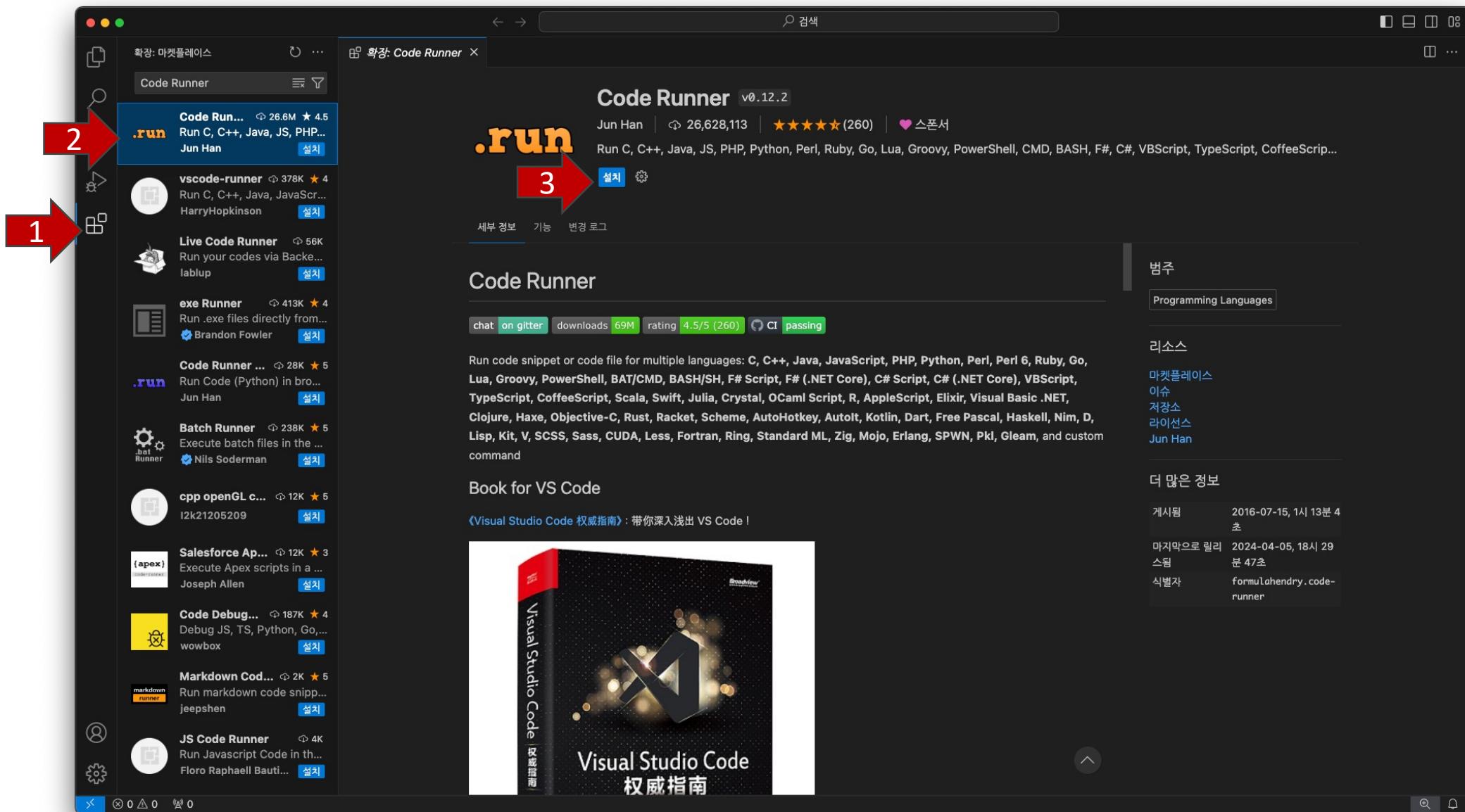
Live Preview 개발자 도구





Part 1

Visual Studio Code 플러그인 설치



Part 1

JavaScript 작성

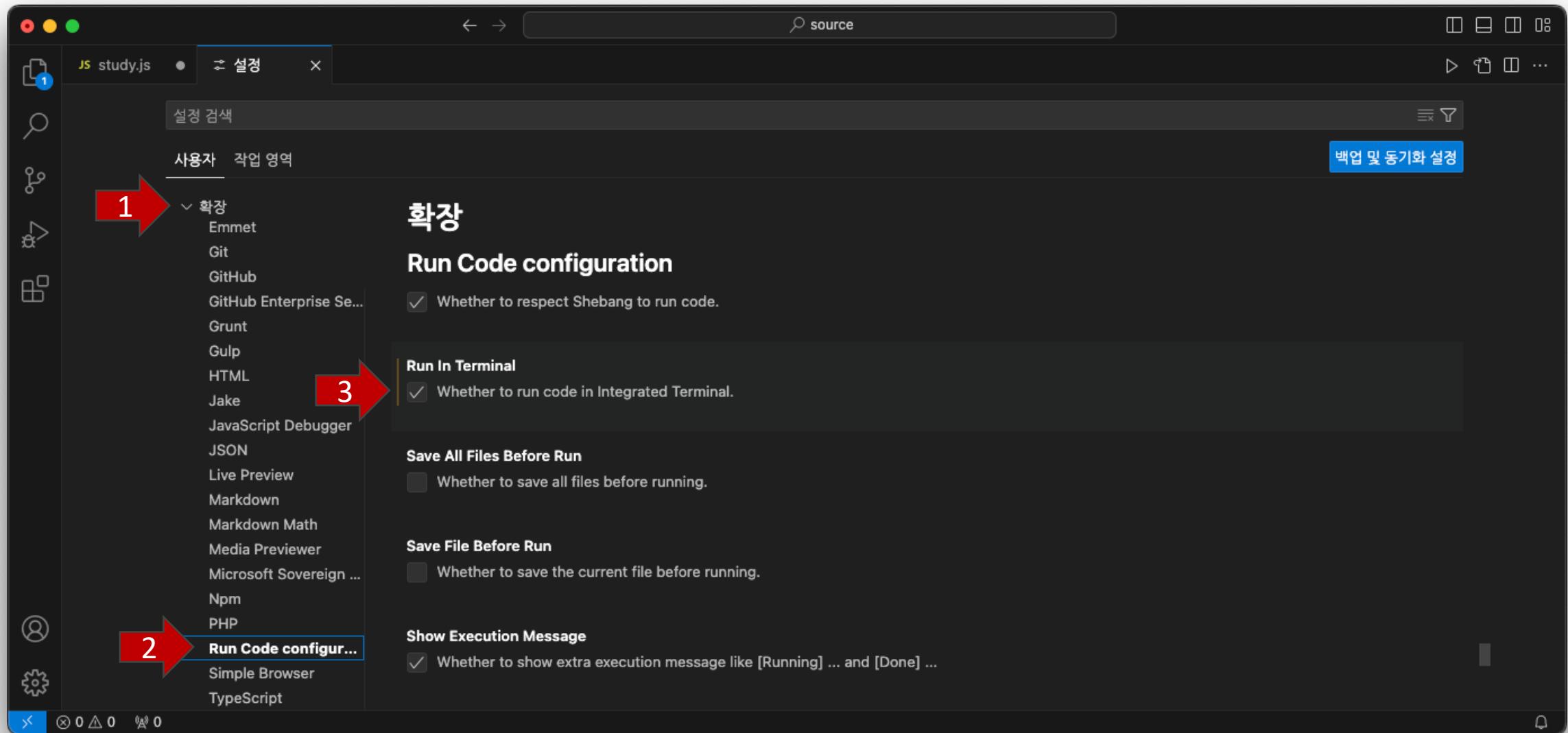
```
study.js
JS study.js
1 console.log('Hello World!')

문제   출력   디버그 콘솔   터미널   포트
[Running] node "/Users/seeker/javascript/study.js"
Hello World!

[Done] exited with code=0 in 0.088 seconds
```

Part 1

Code Runner 환경설정 – 터미널 활성화



Part 1

터미널에서 JavaScript 실행

The screenshot shows a macOS terminal window with a dark theme. The window title is "javascript". In the top-left corner, there are icons for file operations (New, Open, Save, etc.). The main area contains a code editor with a single file named "study.js" containing the code:

```
1 console.log('Hello World!')
```

. Below the code editor is a tab bar with tabs for 문제, 출력, 디버그 콘솔, 터미널, and 포트. The terminal pane displays the command `node "/Users/seeker/javascript/study.js"` followed by its output: `seeker@Mini javascript % node "/Users/seeker/javascript/study.js"` and `Hello World!`. The bottom status bar shows file statistics: 줄 1, 열 28, 공백: 4, UTF-8, LF, {}, JavaScript.

Part 1

JavaScript 오류 확인

The screenshot shows a macOS terminal window with a dark theme. The title bar says "javascript". The main pane displays the following text:

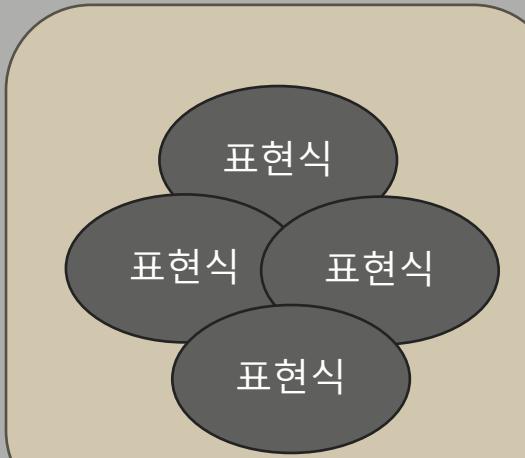
```
[Running] node "/Users/seeker/javascript/study.js"
/Users/seeker/javascript/study.js:1
console.log('Hello World!'
^~~~~~
SyntaxError: missing ) after argument list
    at wrapSafe (node:internal/modules/cjs/loader:1281:20)
    at Module._compile (node:internal/modules/cjs/loader:1321:27)
    at Module._extensions..js (node:internal/modules/cjs/loader:1416:10)
    at Module.load (node:internal/modules/cjs/loader:1208:32)
    at Module._load (node:internal/modules/cjs/loader:1024:12)
    at Function.executeUserEntryPoint [as runMain] (node:internal/modules/run_main:174:12)
    at node:internal/main/run_main_module:28:49

Node.js v20.13.1
[Done] exited with code=1 in 0.085 seconds
```

The terminal window has several icons in its sidebar: a file icon, a search icon, a magnifying glass, a gear icon, and a user icon.

A close-up photograph of a person's hands holding a white smartphone. The person is wearing a white button-down shirt. A large, semi-transparent white rectangular box is overlaid on the center-left portion of the image. Inside this box, the Korean text "기본 용어" is displayed in a bold, black, sans-serif font.

기본 용어



문장



문장

프로그램

The screenshot shows a dark-themed code editor interface with a sidebar on the left containing icons for file operations, search, and settings. The main area displays a file named 'study.js' with the following content:

```
study.js > ...
1 10 + 20 + 30          //표현식
2 var hello = '안녕' + '하세요' //표현식
3 alert(hello)           //표현식
4
5 //세미콜론을 사용한 문장 구분
6 10 + 20 + 30;var hello = '안녕' + '하세요';alert(hello)
7
8 //줄바꿈을 사용한 문장 구분
9 10 + 20 + 30
10 var hello = '안녕' + '하세요'
11 alert(hello)
12
13 //세미콘론과 줄바꿈을 사용한 문장 구분
14 10 + 20 + 30;
15 var hello = '안녕' + '하세요';
16 alert(hello);
17
```

The right side of the interface shows a terminal window with the following output:

```
[Running] node "/Users/seeker/javascript/study.js"
[Done] exited with code=0 in 0.094 seconds
```

The bottom status bar indicates the file has 17 lines, is in UTF-8 encoding, and uses LF line endings. It also shows icons for Prettier and a code analysis tool.

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
true	false	null	as
from	get	of	set
target			

Part 1

식별자

The screenshot shows a dark-themed code editor interface with two tabs open: `study.js` and `study2.js`. The search bar at the top is set to `javascript`.

study.js:

```
1 //식별자로 사용 가능한 단어
2
3 let alpha = 10;
4
5 let alpha10 = 10;
6
7 let _alpha = 10;
8
9 let $alpha = 10;
10
11 let Alpha = 10;
12
13 let ALPHA = 10;
14
```

study2.js:

```
1 //식별자로 사용 불가능한 단어
2
3 let break = 10; //키워드라서 안됨
4
5 let 273break = 10; //숫자로 시작해서 안됨
6
7 let has space = 10; //공백을 포함해서 안됨
8
```

The code editor includes standard UI elements like file, search, and settings icons on the left, and status indicators at the bottom.

식별자 > 식별자를 만드는 일반적인 관례

- **클래스**의 이름은 항상 대문자로 시작합니다.
- **변수와 인스턴스, 함수, 메소드**의 이름은 항상 소문자로 시작합니다.
- 여러 단어로 이루어진 **식별자**는 각 단어의 첫 글자를 대문자로 합니다.

will out → willOut

will return → willReturn

I am a boy → IAmABoy

Part 1

식별자 > 식별자의 종류

The screenshot shows a browser's developer tools interface with the 'Console' tab selected. The left sidebar lists files: 'study.html' (selected) and 'study.html'. The main area displays the following JavaScript interactions:

```
> alert('Hello World') //함수
< undefined
> Array.length //속성
< 1
> let input = 10 //변수
< undefined
> Math.PI //속성
< 3.141592653589793
> Math.abs(-273) //메소드
< 273
>
```

The console output includes color-coded tokens: orange for strings ('Hello World'), yellow for properties ('length'), blue for variables ('input'), purple for functions ('Math.PI'), and green for numbers ('273').

Part 1

주석

The screenshot shows a browser developer tools window with two tabs: 'javascript' and 'study.html'. The 'javascript' tab contains the following code:

```
study.html <script>
1  // 주석
2  /*
3   * 주석
4   */
5  // 주석
6  /*
7   * 주석
8   */
9  // 주석
10 /*
11  */
12 // 주석
13 // 주석
14 // 주석
15 // 주석
16 // 주석
17 // 주석
18 // 주석
```

The 'study.html' tab shows the rendered HTML page at <http://127.0.0.1:3000/study.html>. The page content is blank, indicating that the comments have been stripped or are not rendered.

Part 1

출력

The screenshot shows a browser developer tools interface with the 'Console' tab selected. A search bar at the top contains the text 'javascript'. The main area displays a stack trace:

```
> 10
< 10
> 10 + 10
< 20
> 10 * 10
< 100
> |
```

The left sidebar shows the file 'study.html' with line 1 selected. The bottom status bar indicates the file is '줄 1, 열 1' (Line 1, Column 1), encoding is 'UTF-8', and port is '3000'. There are also icons for Prettier and a lock.

Part 1

출력 > 경고창에 출력하기

The screenshot shows a developer's workspace with two main components:

- Code Editor:** On the left, an IDE window displays the content of an `study.html` file. The code is as follows:

```
1  <!DOCTYPE html>
2  <html>
3  |   <head>
4  |       <title></title>
5  |       <script>
6  |           |   alert("Hello World");
7  |       </script>
8  |   </head>
9  |   <body></body>
10 |</html>
11 |
```

- Browser Preview:** On the right, a browser window titled `study.html` is open at the URL `http://127.0.0.1:3000/study.html`. A modal dialog box is displayed, containing the text "127.0.0.1:3000 내용: Hello World" and a blue "확인" (Confirm) button.

Part 1

출력 > 콘솔에서 출력하기

The screenshot shows a dark-themed code editor interface. On the left, there's a sidebar with various icons for file operations like copy, paste, search, and settings. The main area has tabs for 'study.js' and 'source'. The 'study.js' tab contains the following code:

```
source > JS study.js
1 console.log('Hello World');
```

To the right of the code editor is a terminal window. The title bar of the terminal says 'javascript'. The terminal tabs include '문제', '출력' (Output), '디버그 콘솔' (Debug Console), '터미널' (Terminal), and '포트' (Port). The 'Output' tab is currently selected. It displays the following output from running the script:

```
[Running] node "/Users/seeker/javascript/source/study.js"
Hello World

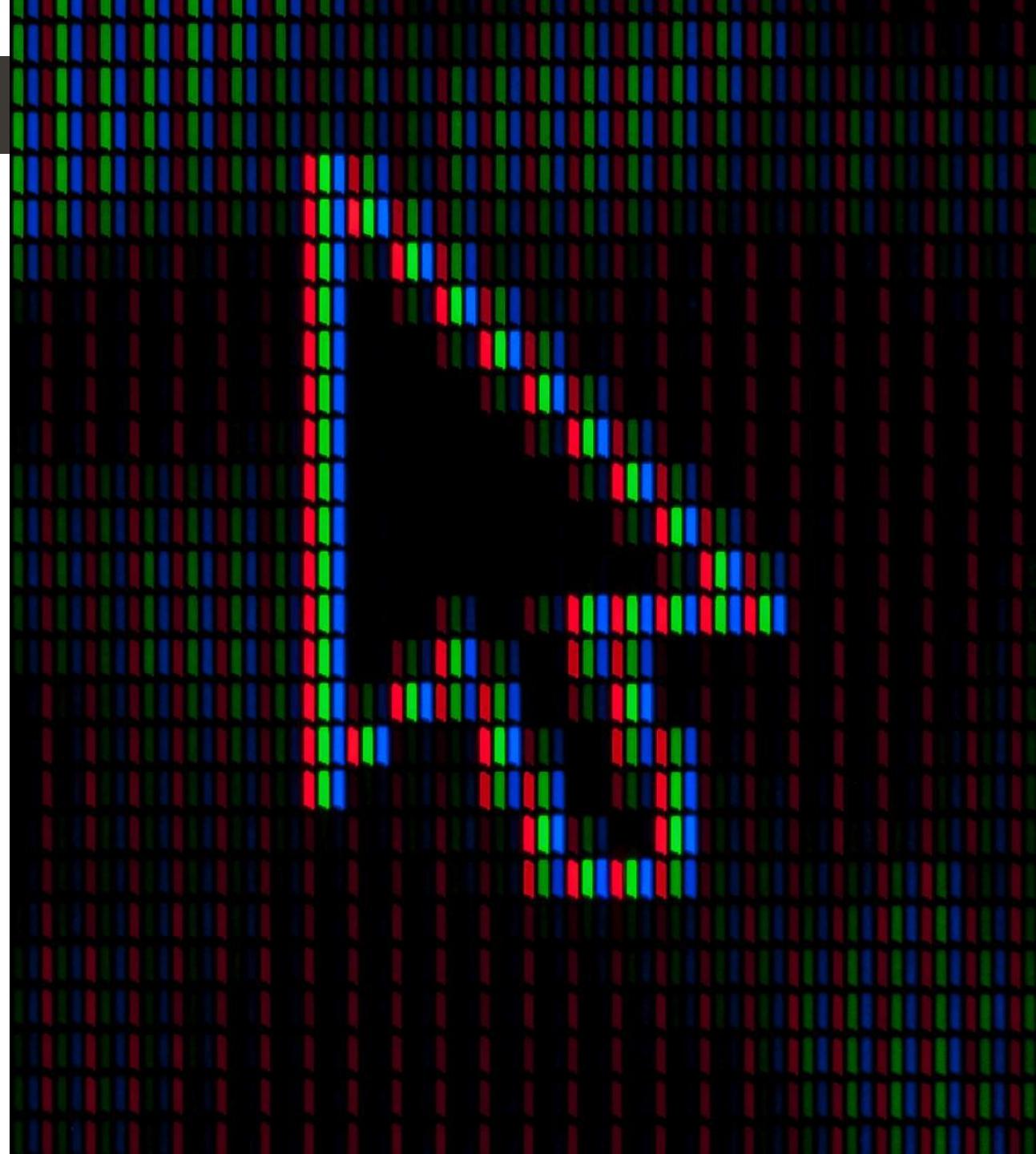
[Done] exited with code=0 in 0.097 seconds
```

At the bottom of the terminal window, there are status indicators for file changes, line numbers, and file type (JavaScript). The status bar also shows the current file is 'study.js'.

목차

a table of contents

- 1 자바스크립트 기초
- 2 자료와 변수
- 3 조건문
- 4 반복문
- 5 함수



A close-up photograph of a person's hands holding a white smartphone. The person is wearing a white button-down shirt. The background is blurred.

기본 자료형

숫자
(number)

문자열
(string)

불
(boolean)

Part 2

문자열 자료형 > 만들기

The screenshot shows a browser's developer tools interface, specifically the Console tab, with a dark theme. In the top navigation bar, the 'Console' tab is selected. Below it, the output area displays several string values:

```
> '안녕하세요'  
< '안녕하세요'  
> "안녕하세요"  
< '안녕하세요'  
>
```

The console also includes standard controls like back and forward arrows, a search bar for 'javascript', and a status bar at the bottom showing file details (줄 1, 열 1), encoding (UTF-8), and port information (포트: 3000). A Prettier icon is also visible in the status bar.

Part 2

문자열 자료형 > 만들기

The screenshot shows a browser's developer tools console tab open, with the search bar set to "javascript". The console output displays several string literals and their corresponding escape sequences:

```
> 'This is "String"'  
< 'This is "String"'  
> "This is 'String'"  
< "This is 'String'"  
> "This is \"String\""  
< 'This is "String"'  
> 'This is \'String\''  
< "This is 'String'"  
> " \ "  
< ' '  
> "\ "  
✖ Uncaught SyntaxError: Invalid or unexpected token VM593:1  
> "\\ \\ \\ \\"  
< '\\ \\ \\ \\'  
> |
```

A red error bar highlights the line "✖ Uncaught SyntaxError: Invalid or unexpected token VM593:1", indicating a parsing error at the end of the previous command. The bottom status bar shows the file is "study.html", has 1 issue, and is using Prettier.

Part 2

문자열 자료형 > 연산

The screenshot shows a browser developer tools interface with the 'Console' tab selected. The search bar at the top contains the text 'javascript'. The left sidebar shows an open file named 'study.html' with one line of code: '1'. The right panel displays the results of a string analysis for the expression '안녕하세요'.

2 Issues: 2

- > "안녕" + "하세요"
- < '안녕하세요'
- > '안녕하세요'[0]
- < '안'
- > '안녕하세요'[1]
- < '녕'
- > '안녕하세요'[2]
- < '하'
- > '안녕하세요'[3]
- < '세'
- > '안녕하세요'[4]
- < '요'
- > |

Bottom status bar: 줄 1, 열 1 | 공백: 4 | UTF-8 | LF | HTML | 포트: 3000 | Prettier

Part 2

문자열 자료형 > 길이 구하기

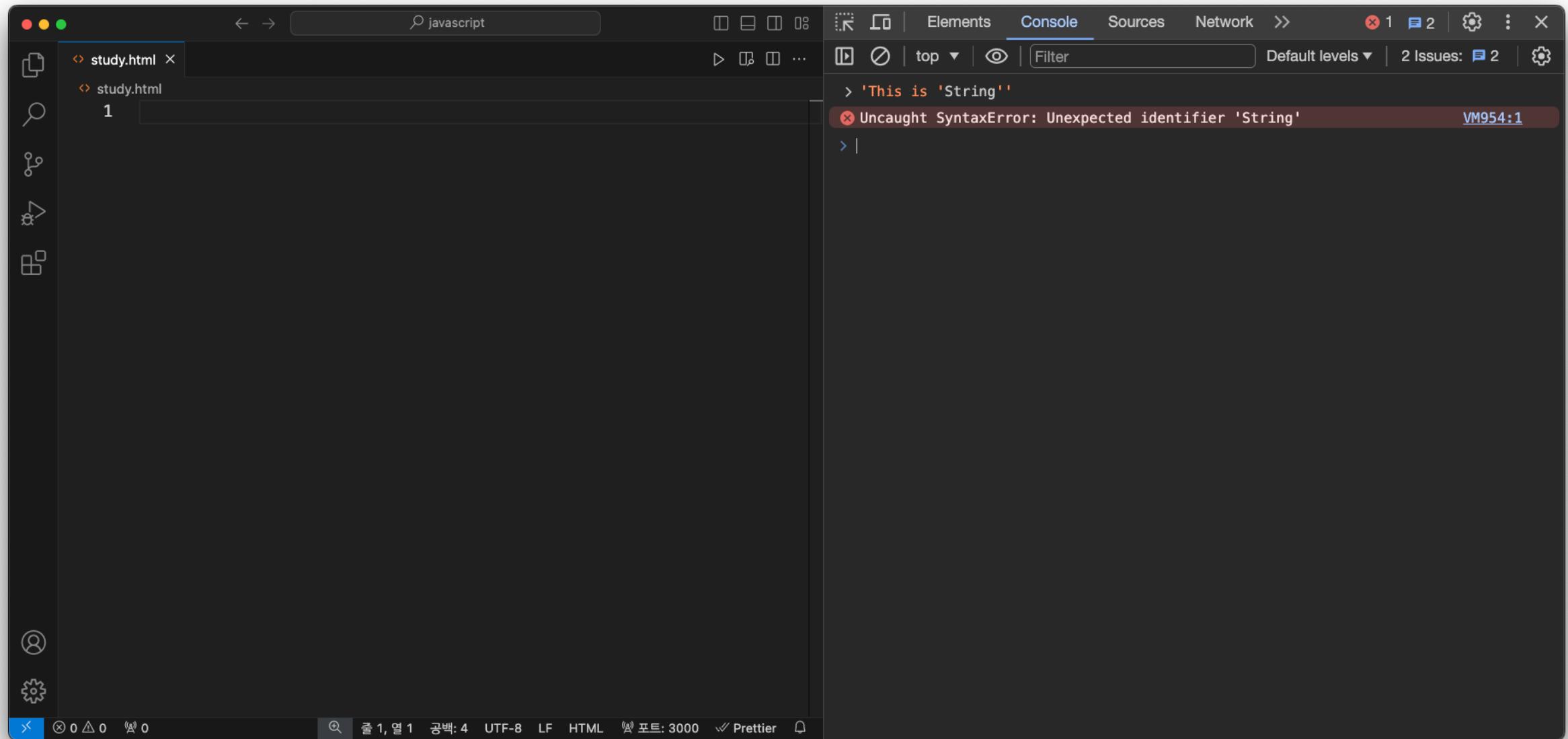
The screenshot shows a browser developer tools interface with the 'Console' tab selected. The left sidebar shows an open file named 'study.html'. The console output displays the following JavaScript code and its results:

```
> "안녕하세요".length
< 5
> "자바스크립트".length
< 6
> "".length
< 0
>
```

The results are color-coded: the strings are in yellow, and their lengths are in purple. The console also includes a search bar at the top labeled 'javascript' and a status bar at the bottom showing file details like '줄 1, 열 1' and encoding 'UTF-8'.

Part 2

문자열 자료형 > 오류



Part 2

숫자 자료형 > 연산자

The screenshot shows a browser's developer tools interface, specifically the Console tab, with a dark theme. On the left, the file tree shows 'study.html' is open. The console output is as follows:

```
> 273  
< 273  
> 52.273  
< 52.273  
> 5 + 3  
< 8  
> 5 * 3  
< 15  
> 5 - 3  
< 2  
> 5 / 3  
< 1.6666666666666667  
> 5 + 3 * 2  
< 11  
> (5 + 3) *2  
< 16  
>
```

The console also displays the following status bar information at the bottom:

- File: study.html
- Line: 줄 1, 열 1
- Encoding: 공백: 4
- Character Encoding: UTF-8
- Line Breaks: LF
- HTML: HTML
- Port: 포트: 3000
- Prettier: Prettier

Part 2

숫자 자료형 > 연산자

The screenshot shows a browser developer tools interface with the 'Console' tab selected. The left sidebar shows an open file named 'study.html'. The console output displays a series of numbers from 0 to 10, each preceded by '> 10 %' and followed by a percentage value (e.g., '2', '3', '4', etc.). The output ends with a final '> |'.

```
> 10 % 2  
< 0  
> 10 % 3  
< 1  
> 10 % 4  
< 2  
> 10 % 5  
< 0  
> 10 % 6  
< 4  
> 10 % 7  
< 3  
> 10 % 8  
< 2  
> 10 % 9  
< 1  
> 10 % 10  
< 0  
> |
```

At the bottom of the console, there are status indicators: '0 △ 0 ⚡ 0' and file information: '줄 1, 열 1 공백: 4 UTF-8 LF HTML 포트: 3000 Prettier'.

Part 2

불 자료형 > 만들기

The screenshot shows a browser's developer tools open to the 'Console' tab. The left sidebar shows an 'Elements' panel with a file named 'study.html' selected. The main area is the 'Console' tab, which displays the following evaluations:

```
> true  
< true  
> false  
< false  
> 52 > 273  
< false  
> 52 < 273  
< true  
> 10 === 10  
< true  
> 10 !== 10  
< false  
> 10 >= 10  
< true  
> 10 <= 10  
< true  
> '가방' > '하마'  
< false  
> |
```

The console also includes a search bar at the top labeled 'javascript' and a status bar at the bottom showing file details: 줄 1, 열 1, 공백: 4, UTF-8, LF, HTML, 포트: 3000, Prettier.

Part 2

불 자료형 > 만들기

The screenshot shows a dark-themed code editor interface. On the left is the file tree, which contains a single file named "study.js". The main area displays the following code:

```
source > study.js
1  if (273 < 52) {
2    console.log('273은 52보다 작습니다.');
3 }
4
5 if (273 > 52) {
6   console.log('273은 52보다 큽니다.');
7 }
8
```

To the right of the code editor is a terminal window titled "javascript". It shows the command "node "/Users/seeker/javascript/source/study.js" and the output of the console logs:

```
[Running] node "/Users/seeker/javascript/source/study.js"
273은 52보다 큽니다.

[Done] exited with code=0 in 0.088 seconds
```

The bottom status bar indicates the file has 8 lines and 4 characters per line, is saved in UTF-8 LF encoding, is a JavaScript file, and is connected to port 3000 via Prettier.

Part 2

불 자료형 > 부정 연산자

The screenshot shows a browser's developer tools open to the 'Console' tab. The left sidebar shows an 'Elements' panel with a file named 'study.html' selected. The main area displays the following JavaScript code and its execution results:

```
> !true          // 피연사진가 true로 1개          -> 단항 연산자
< false
> 10 + 20       // 피연사진가 10과 20으로 2개          -> 이항 연산자
< 30
> true? 10 : 20 // 피연사진가 true, 10, 20으로 3개          -> 삼항 연산자
< 10
>
```

The console output shows the results of various operations: the negation of true (false), the addition of 10 and 20 (30), and a ternary expression where true is evaluated as 10 and false as 20.

Part 2

불 자료형 > 논리합/논리곱 연산자

The screenshot shows a browser's developer tools open to the 'Console' tab. On the left, the file 'study.html' is selected in the project tree. The console displays the following sequence of operations:

```
> true && true  
< true  
> true && false  
< false  
> false && true  
< false  
> false && false  
< false  
> true || true  
< true  
> true || false  
< true  
> false || true  
< true  
> false || false  
< false  
>
```

The console interface includes a search bar at the top labeled 'javascript', tabs for Elements, Console, Sources, Network, and Performance, and various status indicators and settings on the right.

Part 2

자료형 검사

The screenshot shows a browser's developer tools open to the 'Console' tab. The left sidebar shows an 'Elements' panel with a file named 'study.html' selected. The main area displays the following JavaScript console output:

```
> typeof('문자열')
< 'string'
> typeof(273)
< 'number'
> typeof(true)
< 'boolean'
> typeof '문자열'
< 'string'
> typeof 273
< 'number'
> typeof 10 === 'number'
< true
>
```

The browser interface includes standard navigation buttons (back, forward, search), a status bar at the bottom with file information like '줄 1, 열 1', and a bottom toolbar with icons for refresh, search, and other developer tools features.

Part 2

템플릿 문자열

The screenshot shows a browser's developer tools open to the 'Console' tab. On the left, there is a sidebar with various icons and a file tree showing 'study.html' is the active file. The main area is the console window.

```
> console.log('표현식 273 + 52의 값은 ' + (273 + 52) + '입니다.')
표현식 273 + 52의 값은 325입니다. VM293:1
< undefined
> console.log(`표현식 273 + 52의 값은 ${273+ 52}입니다.`)
표현식 273 + 52의 값은 325입니다. VM468:1
< undefined
> |
```

The console output demonstrates two ways of using template literals. The first uses a standard string with concatenation, and the second uses a back-tick (`) to include variable interpolation directly within the string.

Part 2

== 연산자와 != 연산자

The screenshot shows a browser developer tools interface with the 'Console' tab selected. The left sidebar shows an open file named 'study.html'. The console output displays the results of several equality comparisons:

```
> 1 == "1"
< true
> false == "0"
< true
> "" == []
< true
> 0 == []
< true
>
```

The results show that primitive values (1, false, "", 0) are considered equal to their string representations ("1", "0", "", "[]") respectively, while objects are considered unequal to strings.

A close-up photograph of a person's hands holding a white smartphone. The person is wearing a white button-down shirt. The background is blurred.

상수와 변수

Part 2

상수

The screenshot shows a browser's developer tools open to the 'Console' tab. The left sidebar shows an 'Elements' panel with a file named 'study.html' selected. The main area is the 'Console' tab, which displays the following JavaScript session:

```
> const pi = 3.141592
< undefined
> pi
< 3.141592
> const r = 10 //반지름이 10인 상수를 선언
< undefined
> 2 * pi * r //반지름으로 원의 둘레 구하기
< 62.83184
> pi * r * r //반지름으로 원의 넓이 구하기
< 314.1592
>
```

The console output shows the definition of a constant `pi` with the value `3.141592`, its value being `3.141592`, the declaration of a constant `r` with the value `10`, the calculation of the circumference (`2 * pi * r`) resulting in `62.83184`, and the calculation of the area (`pi * r * r`) resulting in `314.1592`.

Part 2

상수 > 오류

The screenshot shows a browser's developer tools open to the 'Console' tab. The left sidebar shows an 'Elements' panel with a file named 'study.html' selected. The main area is the 'Console' tab, which contains the following text:

```
> const name = "name이라는 이름의 상수를 선언해볼게요."
< undefined
> name = "그 값을 변경해볼게요."
✖ > Uncaught TypeError: Assignment to constant variable.
      at <anonymous>:1:6
> const pi
✖ > Uncaught SyntaxError: Missing initializer in const declaration
      VM1885:1
> |
```

The console highlights two errors: one for attempting to assign to a constant variable 'name' and another for missing an initializer in a 'const' declaration for 'pi'. The error for 'pi' is specifically labeled 'VM1885:1'.

Part 2

변수

The screenshot shows a browser developer tools window with the 'Console' tab selected. The left sidebar lists files: 'study.html' (marked with a blue dot) and 'study.html'. The main area displays the following JavaScript code and its execution results:

```
> let pi = 3.141592
< undefined
> pi
< 3.141592
> let r = 10 //반지름이 10인 변수를 선언
< undefined
> 2 * pi * 2 //반지름으로 원의 둘레 구하기
< 12.566368
> pi * r * r //반지름으로 원의 넓이 구하기
< 314.1592
> |
```

The browser status bar at the bottom shows: 줄 1, 열 1 | 공백: 4 | UTF-8 | LF | HTML | 포트: 3002 | Prettier.

Part 2

변수 > 변경

The screenshot shows a browser developer tools interface with the 'Console' tab selected. The left sidebar shows an open file named 'study.html'. The console output is as follows:

```
> let g = 9.8 //중력 가속도
< undefined
> let m = 10 //질량
< undefined
> m * g //힘
< 98
> g = 9.80665
< 9.80665
> m * g
< 98.06649999999999
>
```

The browser status bar at the bottom indicates the file is 'study.html', encoding is 'UTF-8', and port is '3002'. There are also icons for Prettier and a lock.

Part 2

변수 > 오류

The screenshot shows a terminal window with two tabs open, both labeled "study.js". The top tab contains the following code:

```
source > JS study.js > [e] name
1 let name = "name이라는 이름의 변수를 선언합니다"
2 let name = "한 번 더 선언해볼게요"
```

The bottom tab contains the same code:

```
source > JS study.js > [e] nameB
1 let nameA = "name이라는 이름의 변수를 선언합니다"
2 let nameB = "한 번 더 선언해볼게요"
```

To the right of the tabs, the terminal output shows the execution of the script:

```
[Running] node "/Users/seeker/javascript/source/study.js"
/Users/seeker/javascript/source/study.js:3
const name = "한 번 더 선언해볼게요."
^

SyntaxError: Identifier 'name' has already been declared
at wrapSafe (node:internal/modules/cjs/loader:1281:20)
at Module._compile (node:internal/modules/cjs/loader:1321:27)
at Module._extensions..js (node:internal/modules/cjs/loader:1416:10)
at Module.load (node:internal/modules/cjs/loader:1208:32)
at Module._load (node:internal/modules/cjs/loader:1024:12)
at Function.executeUserEntryPoint [as runMain]
(node:internal/modules/run_main:174:12)
at node:internal/main/run_main_module:28:49

Node.js v20.13.1

[Done] exited with code=1 in 0.091 seconds
```

The terminal interface includes standard macOS-style window controls at the top right and various icons along the left edge.

Part 2

변수에 적용할 수 있는 연산자 > 복합 대입 연산자

The screenshot shows a browser developer tools window with the 'Console' tab selected. The left sidebar shows files: 'study.html' (1) and 'study.html'. The console output is as follows:

```
> let value = 10 //value라는 변수를 10으로 선언
< undefined
> value += 10 //value에 10을 더합니다.
< 20
> value -= 10 //value에 10을 뺍니다.
< 10
> value *= 10 //value에 10을 곱합니다.
< 100
> value /= 10 //value에 10을 나눕니다.
< 10
> value %= 7 //value에 7의 나머지를 구합니다.
< 3
> |
```

The console uses color coding for keywords and values. The 'Elements', 'Sources', 'Network', and 'Performance' tabs are also visible at the top.

Part 2

변수에 적용할 수 있는 연산자 > 복합 대입 연산자

The screenshot shows a browser developer tools window with two main panes. The left pane is a code editor for a file named 'study.html'. The right pane is a preview of the browser output.

```
<!DOCTYPE html>
<html>
  <head>
    <title></title>
    <script>
      // 변수를 선언합니다.
      let list = '';

      // 연산자를 사용합니다.
      list += '<ul>';
      list += '  <li>Hello</li>';
      list += '  <li>JavaScript..!</li>';
      list += '</ul>';

      // 문서에 출력합니다.
      document.write(list);
    </script>
  </head>
  <body></body>
</html>
```

The preview pane shows the resulting HTML output:

- Hello
- JavaScript..!

Part 2

변수에 적용할 수 있는 연산자 > 증감 연산자

The screenshot shows a browser's developer tools console tab open, with the URL bar containing "javascript". The console log output demonstrates the behavior of the increment (++) and decrement (--) operators on a variable named "number".

```
> let number = 10
< undefined
> number++ //기존 변수 값에 1을 더합니다. (후위)
< 10
> number
< 11
> ++number //기존 변수 값에 1을 더합니다. (전위)
< 12
> number
< 12
> number--
< 12
> number
< 11
> --number //기존 변수 값에 1을 뺍니다. (후위)
< 10
> number
< 10
>
```

The console output shows the following sequence of operations:

- A variable "number" is initialized to 10.
- The expression `number++` is evaluated. The result is 10, and the variable is updated to 11.
- The expression `++number` is evaluated. The result is 12, and the variable is updated to 12.
- The expression `number--` is evaluated. The result is 12, and the variable is updated to 11.
- The expression `--number` is evaluated. The result is 10, and the variable is updated to 10.

Part 2

변수에 적용할 수 있는 연산자 > 증감 연산자

```
study.js
1 // 변수를 선언합니다.
2 let number = 10;
3
4 // 연산자를 사용합니다.
5 number++;
6
7 // 출력합니다.
8 console.log(number);
9
```

[Running] node "/Users/seeker/javascript/study.js"
11
[Done] exited with code=0 in 0.086 seconds

문서 출력 디버그 콘솔 터미널 포트 Code

줄 9, 열 1 공백: 4 UTF-8 LF {} JavaScript 포트: 3000 Prettier

Part 2

변수에 적용할 수 있는 연산자 > 증감 연산자

```
study.js
1 let number = 10;
2
3 // 출력합니다.
4 console.log(number++);
5 console.log(number++);
6 console.log(number++);

[Running] node "/Users/seeker/javascript/study.js"
10
11
12

[Done] exited with code=0 in 0.091 seconds
```

The screenshot shows a dark-themed code editor interface. On the left, there's a sidebar with various icons for file operations like copy, paste, search, and settings. The main area has tabs for 'study.js' and 'study.js > ...'. The code in 'study.js' is as follows:

```
1 let number = 10;
2
3 // 출력합니다.
4 console.log(number++);
5 console.log(number++);
6 console.log(number++);
```

The right side of the interface shows the terminal or output window. It starts with '[Running] node "/Users/seeker/javascript/study.js"', followed by three lines of output: '10', '11', and '12'. Below the output, it says '[Done] exited with code=0 in 0.091 seconds'. At the bottom of the window, there are status indicators and a toolbar.

Part 2

변수에 적용할 수 있는 연산자 > 증감 연산자

```
study.js
1 let number = 10
2
3 // 출력합니다.
4 console.log(number); number += 1
5 console.log(number); number += 1
6 console.log(number); number += 1
```

[Running] node "/Users/seeker/javascript/study.js"

10
11
12

[Done] exited with code=0 in 0.09 seconds

줄 6, 열 33 공백: 4 UTF-8 LF () JavaScript 포트: 3000 ✓ Prettier

Part 2

변수에 적용할 수 있는 연산자 > 증감 연산자

```
study.js
1 let number = 10;
2
3 // 출력합니다.
4 console.log(++number);
5 console.log(++number);
6 console.log(++number);
7
```

[Running] node "/Users/seeker/javascript/study.js"

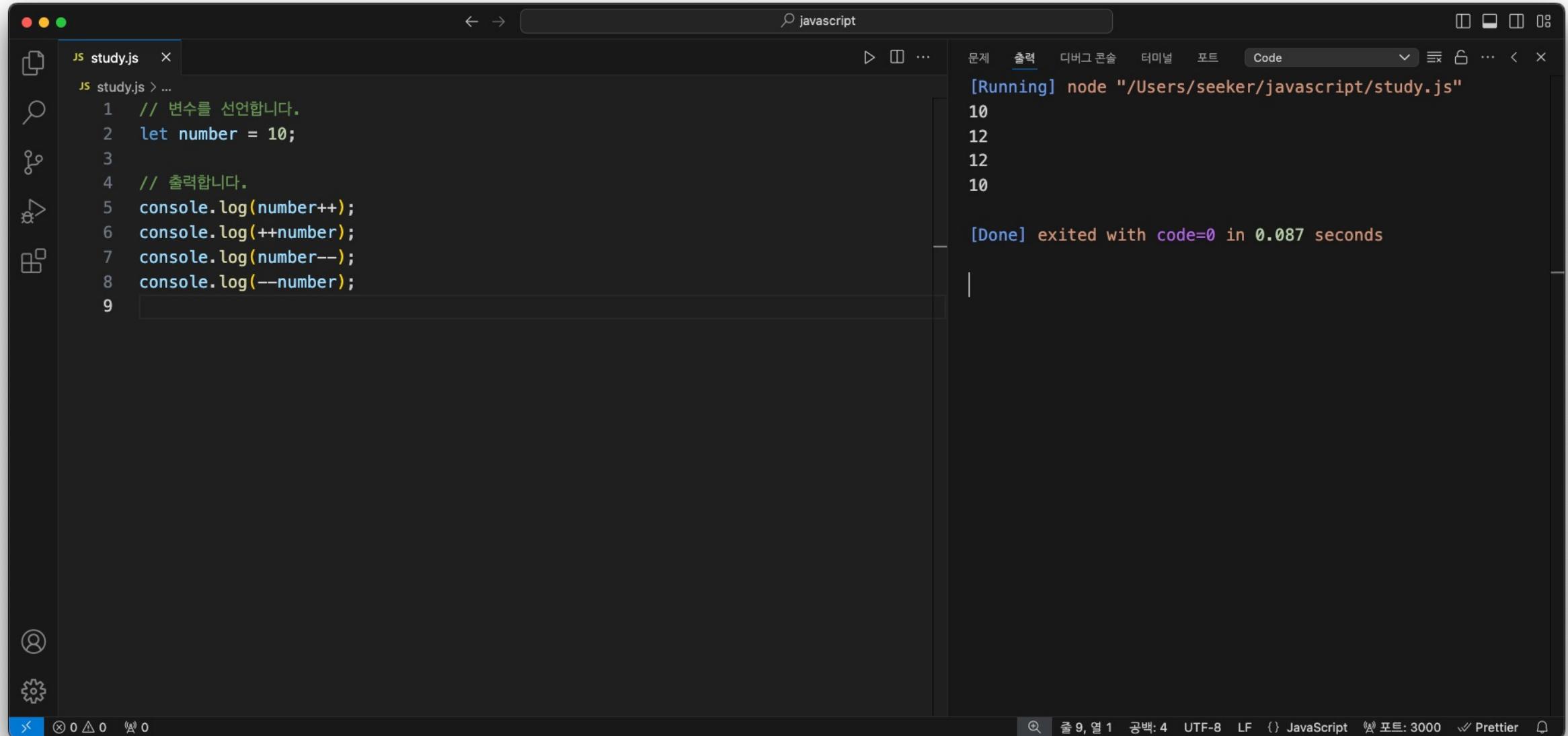
11
12
13

[Done] exited with code=0 in 0.089 seconds

줄 7, 열 1 공백: 4 UTF-8 LF {} JavaScript 포트: 3000 Prettier

Part 2

변수에 적용할 수 있는 연산자 > 증감 연산자



```
study.js
1 // 변수를 선언합니다.
2 let number = 10;
3
4 // 출력합니다.
5 console.log(number++);
6 console.log(++number);
7 console.log(number--);
8 console.log(--number);
9
```

[Running] node "/Users/seeker/javascript/study.js"

```
10
12
12
10

[Done] exited with code=0 in 0.087 seconds
```

줄 9, 열 1 공백: 4 UTF-8 LF () JavaScript 포트: 3000 ✓ Prettier

Part 2

변수에 적용할 수 있는 연산자 > 증감 연산자

```
study.js
1 // 변수를 선언합니다.
2 let number = 10;
3
4 // 출력합니다.
5 console.log(number);
6 number++;
7 number++;
8 console.log(number);
9 console.log(number);
10 number--;
11 number--;
12 console.log(number);
13
```

[Running] node "/Users/seeker/javascript/study.js"

10
12
12
10

[Done] exited with code=0 in 0.096 seconds

줄 13, 열 1 공백: 4 UTF-8 LF () JavaScript 포트: 3000 Prettier

Part 2

undefined 자료형

The screenshot shows a browser's developer tools interface with the 'Console' tab selected. The left sidebar shows an open file named 'study.html'. The console output is as follows:

```
> typeof(abc)
< 'undefined'
> typeof(그냥식별자)
< 'undefined'
> let a
< undefined
> typeof(a)
< 'undefined'
> |
```

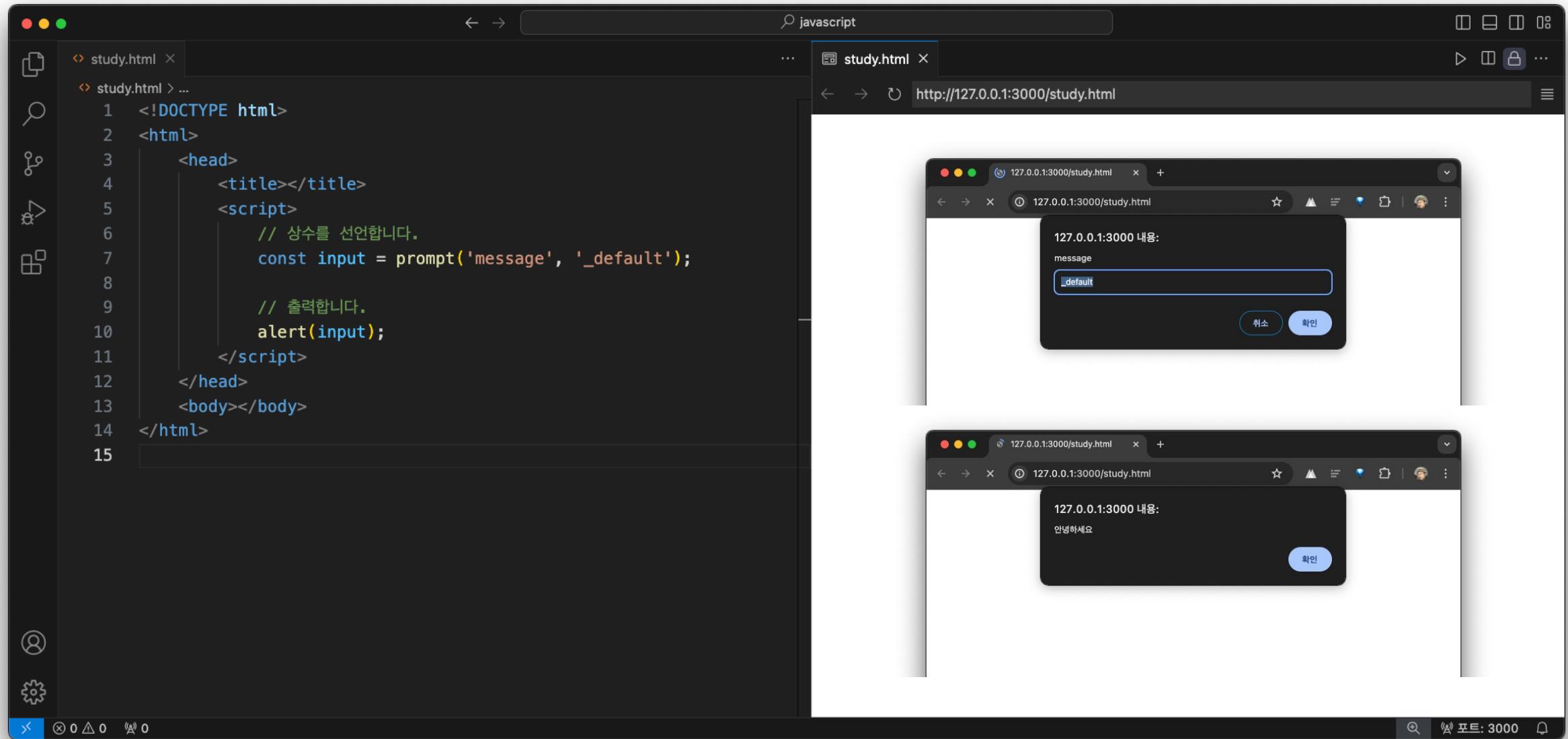
The browser status bar at the bottom indicates: 줄 1, 열 1 공백: 4 UTF-8 LF HTML 포트: 3002 Prettier

A close-up photograph of a person's hands holding a white smartphone. The person is wearing a white lab coat, suggesting a medical or scientific professional. The hands are positioned as if the person is interacting with the device. A large, semi-transparent white rectangular box is overlaid on the center-left portion of the image, containing the Korean text.

자료형 변환

Part 2

문자열 변환



```
<!DOCTYPE html>
<html>
  <head>
    <title></title>
    <script>
      // 상수를 선언합니다.
      const input = prompt('message', '_default');

      // 출력합니다.
      alert(input);
    </script>
  </head>
  <body></body>
</html>
```

The screenshot shows a code editor with a dark theme and a browser window displaying the output of the code. The browser window has two tabs: 'study.html' and 'javascript'. The main content area shows the browser's address bar with 'http://127.0.0.1:3000/study.html'. Below it, there are two screenshots of a web browser window titled '127.0.0.1:3000/study.html'.
The top screenshot shows a 'prompt' dialog box with the message '127.0.0.1:3000 내용:' and the default value '_default'. The bottom screenshot shows an 'alert' dialog box with the message '127.0.0.1:3000 내용:' and the value '안녕하세요'.

Part 2

불 입력

```
study.html
1 <!DOCTYPE html>
2 <html>
3   <head>
4     <title></title>
5     <script>
6       // 상수를 선언합니다.
7       const input = confirm('수락하시겠습니까?');
8
9       // 출력합니다.
10      alert(input);
11    </script>
12  </head>
13  <body></body>
14</html>
15
```

The screenshot shows a code editor with a dark theme. On the left is the code editor interface with a sidebar containing icons for file operations, search, and settings. The main area displays the contents of a file named "study.html". The code itself is as follows:

```
<!DOCTYPE html>
<html>
  <head>
    <title></title>
    <script>
      // 상수를 선언합니다.
      const input = confirm('수락하시겠습니까?');

      // 출력합니다.
      alert(input);
    </script>
  </head>
  <body></body>
</html>
```

To the right of the code editor are two browser windows. Both windows have the URL `http://127.0.0.1:3000/study.html` in their address bars. The top window shows a confirmation dialog box with the message "수락하시겠습니까?" (Do you want to proceed?). The bottom window shows the result of the `confirm` function, which is "true".

Part 2

숫자 자료형으로 변환하기

The screenshot shows a browser developer tools interface with the 'Console' tab selected. The left sidebar shows an open file named 'study.html'. The console output demonstrates various ways to convert strings to numbers:

```
> Number("273")
< 273
> typeof(Number("273"))
< 'number'
> Number("$273")
< NaN
> typeof(Number("$273"))
< 'number'
> Number(true)
< 1
> Number(false)
< 0
>
```

The browser's status bar at the bottom indicates the file is 'study.html', encoding is 'UTF-8', and port is '3002'.

Part 2

숫자 자료형으로 변환하기 > 숫자 연산자 사용

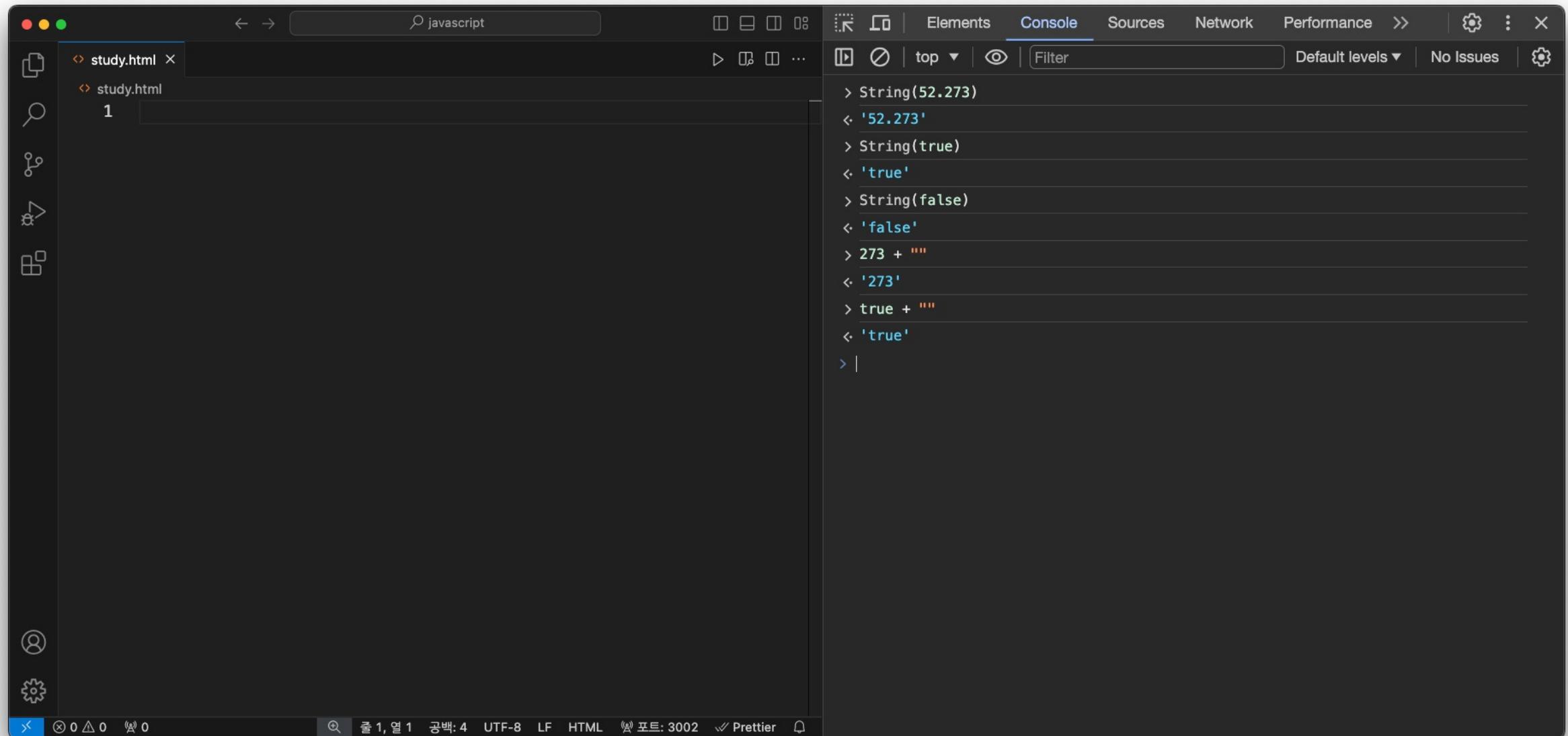
The screenshot shows a browser's developer tools open to the 'Console' tab. On the left, the file 'study.html' is selected in the project tree. The console displays the following JavaScript interactions:

```
> "52" - 0
< 52
> typeof("52" - 0)
< 'number'
> true - 0
< 1
> typeof(true - 0)
< 'number'
> 1 + true
< 2
> 1 + false
< 1
> |
```

The console interface includes a search bar at the top labeled 'javascript', tabs for Elements, Console, Sources, Network, and Performance, and various status indicators and settings on the right.

Part 2

문자열 자료형으로 변환하기



The screenshot shows a browser's developer tools open to the 'Console' tab. The left sidebar displays an 'Elements' panel with a file named 'study.html' selected. The main area is the 'Console' log, which contains the following entries:

```
> String(52.273)
< '52.273'
> String(true)
< 'true'
> String(false)
< 'false'
> 273 + ""
< '273'
> true + ""
< 'true'
> |
```

The console uses color coding for different types: numbers are blue, booleans are red, and strings are green.

Part 2

불 자료형으로 변환하기

The screenshot shows a browser's developer tools interface with the 'Console' tab selected. The left sidebar shows an open file named 'study.html'. The console output displays the following JavaScript code and its results:

```
> Boolean(0)
< false
> Boolean(NaN)
< false
> Boolean("")
< false
> Boolean(null)
< false
> let 변수
< undefined
> Boolean(변수)
< false
> |
```

The code demonstrates various ways to convert values into Boolean types. It shows that Boolean(0), Boolean(""), Boolean(null), and Boolean(undefined) all result in false. The variable '변수' (variable) is declared but has no value, resulting in undefined when passed to Boolean.

Part 2

불 자료형으로 변환하기 > 논리 부정 연산자 사용

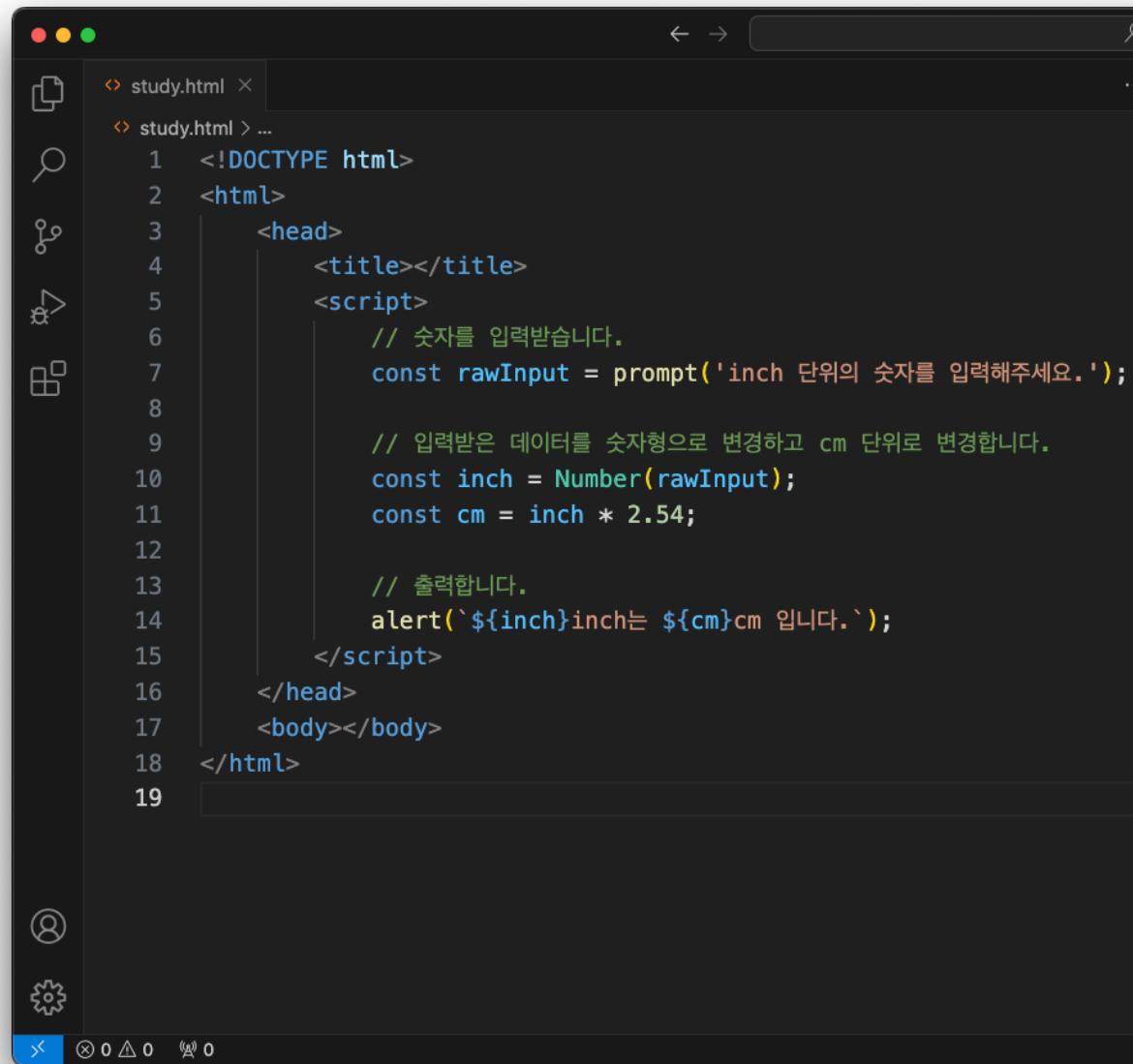
The screenshot shows a browser's developer tools interface with the 'Console' tab selected. The left sidebar shows an open file named 'study.html'. The console output is as follows:

```
> !!273  
< true  
> !!0  
< false  
> !!!'안녕하세요'  
< true  
> !!!  
< false  
> |
```

The console uses color coding where blue indicates variable names and purple indicates boolean values.

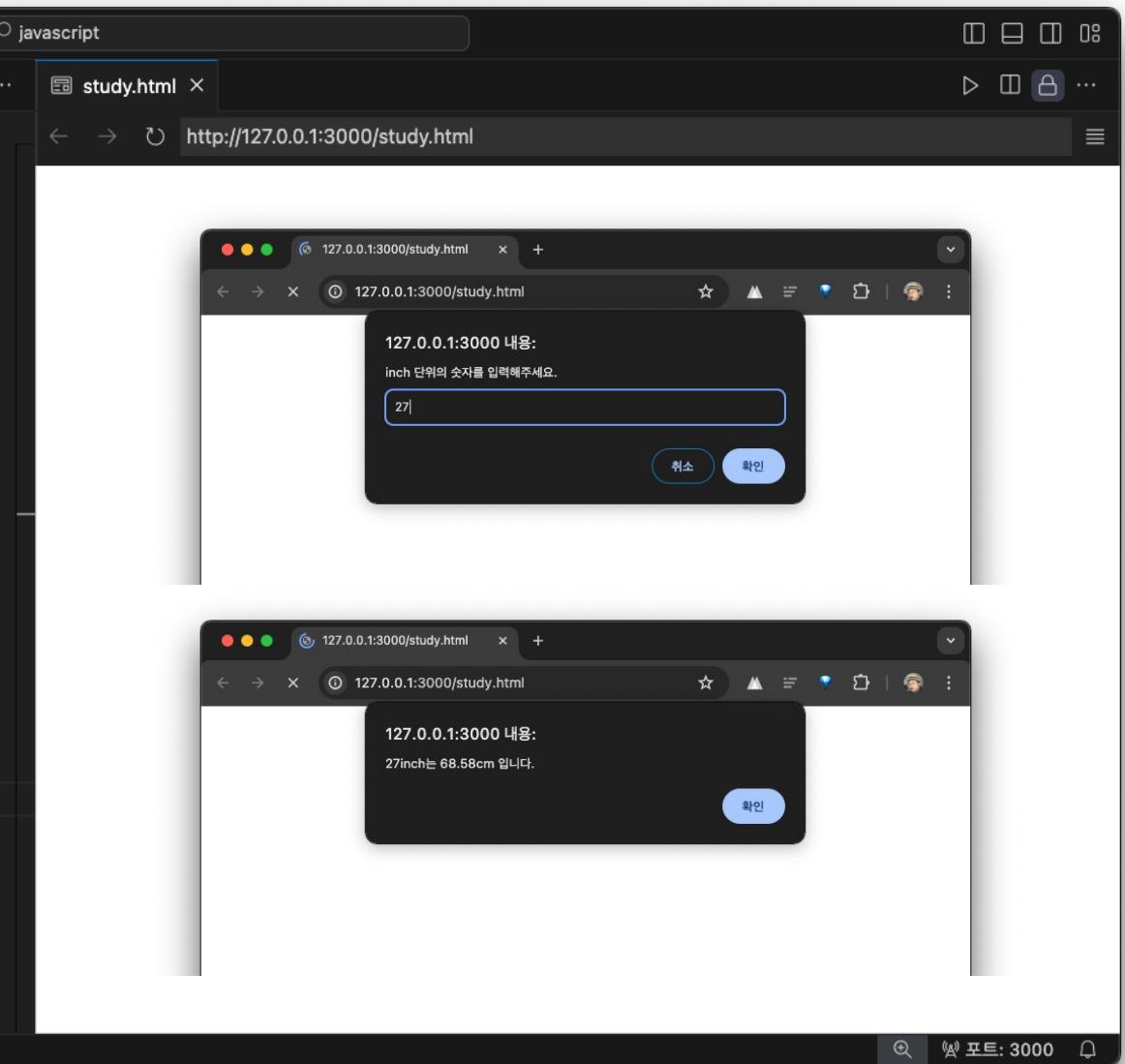
Part 2

inch를 cm 단위로 변경하기



The screenshot shows a code editor window with a file named "study.html". The code contains a script that prompts the user for an inch value, converts it to cm, and then displays the result.

```
1  <!DOCTYPE html>
2  <html>
3      <head>
4          <title></title>
5          <script>
6              // 숫자를 입력받습니다.
7              const rawInput = prompt('inch 단위의 숫자를 입력해주세요.');
8
9              // 입력받은 데이터를 숫자형으로 변경하고 cm 단위로 변경합니다.
10             const inch = Number(rawInput);
11             const cm = inch * 2.54;
12
13             // 출력합니다.
14             alert(`${inch}inch는 ${cm}cm 입니다.`);
15         </script>
16     </head>
17     <body></body>
18 </html>
```



Two screenshots of a web browser window are shown. The top screenshot shows the initial prompt dialog: "127.0.0.1:3000 내용: inch 단위의 숫자를 입력해주세요." with the input field containing "27". The bottom screenshot shows the result dialog: "127.0.0.1:3000 내용: 27inch는 68.58cm 입니다." with the "확인" (Confirm) button highlighted.

목차

a table of contents

- 1 자바스크립트 기초
- 2 자료와 변수
- 3 조건문
- 4 반복문
- 5 함수

A close-up photograph of a person's hands holding a white smartphone. The person is wearing a white button-down shirt. The background is blurred.

if 조건문

Part 3

if 조건문

The screenshot shows a dark-themed code editor interface. On the left is the file tree, with a single file named "study.js" selected. The main area displays the following code:

```
JS study.js ×
JS study.js
1 // if 조건문
2 if (273 < 100) {
3     // 표현식 273 < 100이 참일 때 실행합니다.
4     console.log('273 < 100 => true');
5 }
6
7 // 프로그램 종료
8 console.log('종료');
9
```

The output panel on the right shows the terminal window with the following logs:

```
[Running] node "/Users/seeker/javascript/study.js"
종료

[Done] exited with code=0 in 0.087 seconds
```

At the bottom, there are status icons and text indicating the file has 9 lines, 4 columns, is in UTF-8 encoding, is a JavaScript file, and is on port 3000.

Part 3

if 조건문

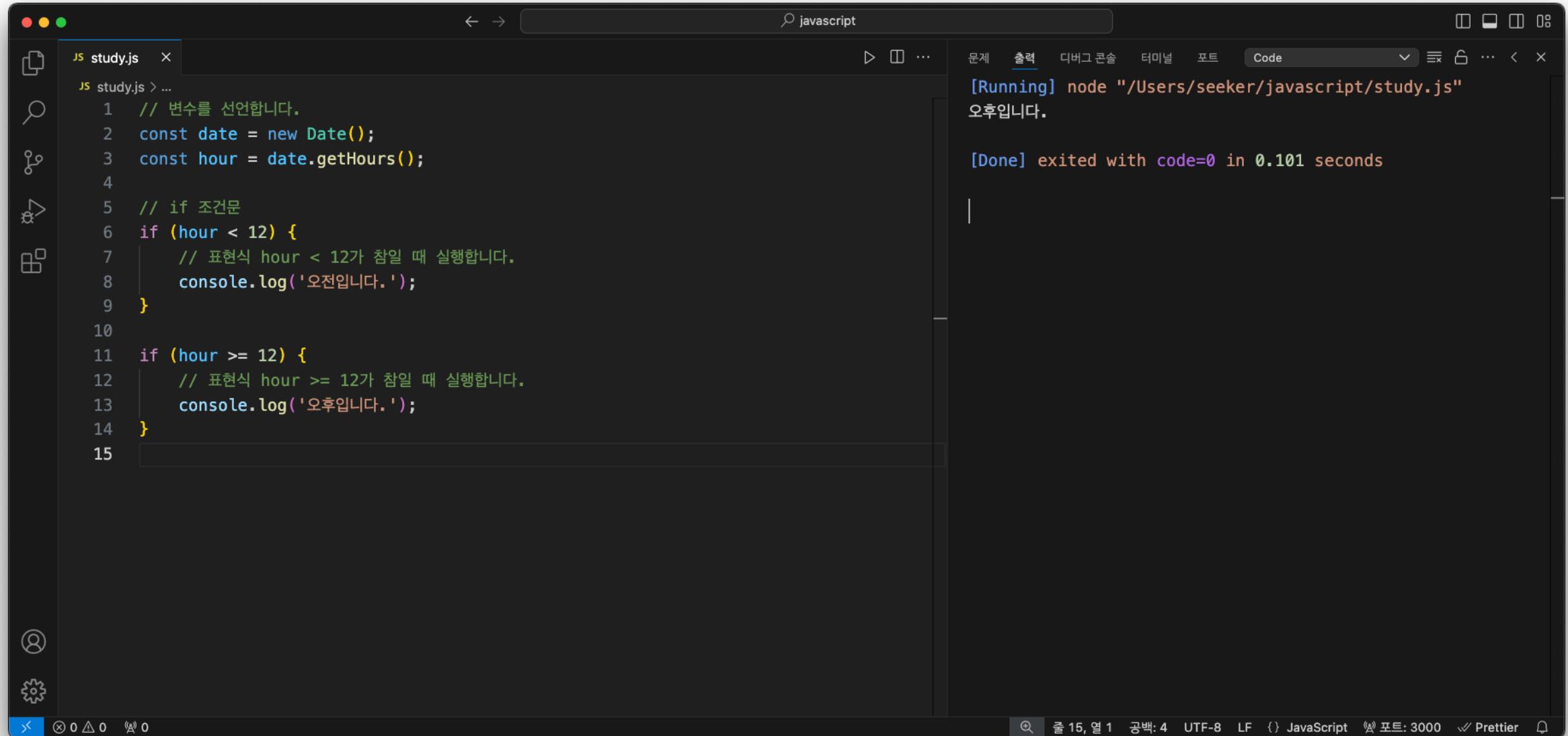
The screenshot shows a browser's developer tools open to the 'Console' tab. On the left, there is a sidebar with various icons and a file tree showing 'study.html' is the active file. The main area is a dark-themed console window with white text. It displays a series of JavaScript commands and their results:

```
> const date = new Date()
< undefined
> date.getFullYear()
< 2024
> date.getMonth() + 1
< 5
> date.getDate()
< 31
> date.getHours()
< 20
> date.getMinutes()
< 51
> date.getSeconds()
< 25
> |
```

The console also includes a search bar at the top labeled 'javascript', a 'Filter' input field, and several tabs: Elements, Console, Sources, Network, and Performance. At the bottom, there are status indicators for tabs, search, and Prettier.

Part 3

if 조건문



The screenshot shows a dark-themed code editor interface with a sidebar on the left containing icons for file operations like Open, Save, Find, and Settings. The main area displays a file named 'study.js' with the following content:

```
study.js > ...
1 // 변수를 선언합니다.
2 const date = new Date();
3 const hour = date.getHours();
4
5 // if 조건문
6 if (hour < 12) {
7     // 표현식 hour < 12가 참일 때 실행합니다.
8     console.log('오전입니다.');
9 }
10
11 if (hour >= 12) {
12     // 표현식 hour >= 12가 참일 때 실행합니다.
13     console.log('오후입니다.');
14 }
15
```

The right side of the interface shows the terminal output. It starts with '[Running]' followed by the command 'node "/Users/seeker/javascript/study.js"', then '오후입니다.' (It's afternoon), and finally '[Done] exited with code=0 in 0.101 seconds'.

At the bottom of the screen, there are several status indicators: a search icon, a file icon, a refresh icon, a zoom icon, a character count of '줄 15, 열 1', a byte count of '공백: 4', a line ending indicator 'LF', a file type indicator 'JavaScript', a port number '포트: 3000', a Prettier icon, and a save icon.

Part 3

if else 조건문

study.js

```
1 // 변수를 선언합니다.
2 const date = new Date();
3 const hour = date.getHours();
4
5 // if 조건문
6 if (hour < 12) {
7     // 표현식 hour < 12가 참일 때 실행합니다.
8     console.log('오전입니다.');
9 } else {
10    // 표현식 hour < 12가 거짓일 때 실행합니다.
11    console.log('오후입니다.');
12 }
13
```

[Running] node "/Users/seeker/javascript/study.js"
오후입니다.

[Done] exited with code=0 in 0.457 seconds

줄 13, 열 1 공백: 4 LF {} JavaScript 포트: 3000 Prettier

Part 3

중첩 조건문

```
study.js
1 // 변수를 선언합니다.
2 const date = new Date();
3 const hour = date.getHours();
4
5 // if 조건문
6 if (hour < 11) {
7     // 표현식 hour < 11가 참일 때 실행합니다.
8     console.log('아침 먹을 시간입니다.');
9 } else {
10    // 표현식 hour < 11가 거짓일 때 실행합니다.
11    if (hour < 15) {
12        // 표현식 hour < 15가 참일 때 실행합니다.
13        console.log('점심 먹을 시간입니다.');
14    } else {
15        // 표현식 hour < 15가 거짓일 때 실행합니다.
16        console.log('저녁 먹을 시간입니다.');
17    }
18 }
19
```

[Running] node "/Users/seeker/javascript/study.js"
저녁 먹을 시간입니다.

[Done] exited with code=0 in 0.094 seconds

문서 출력 디버그 콘솔 터미널 포트 Code

줄 19, 열 1 공백: 4 UTF-8 LF {} JavaScript 포트: 3000 Prettier

Part 3

if else if 조건문

```
study.js > ...
1 // 변수를 선언합니다.
2 const date = new Date();
3 const hour = date.getHours();
4
5 // if else if 조건문
6 if (hour < 11) {
7     // 표현식 hour < 11가 참일 때 실행합니다.
8     console.log('아침 먹을 시간입니다.');
9 } else if (hour < 15) {
10    // 표현식 hour < 11가 거짓이고 표현식 hour < 15가 참일 때 실행합니다.
11    console.log('점심 먹을 시간입니다.');
12 } else {
13    // 표현식 hour < 15가 거짓일 때 실행합니다.
14    console.log('저녁 먹을 시간입니다.');
15 }
16
```

[Running] node "/Users/seeker/javascript/study.js"
저녁 먹을 시간입니다.

[Done] exited with code=0 in 0.097 seconds

문서 출력 디버그 콘솔 터미널 포트 Code

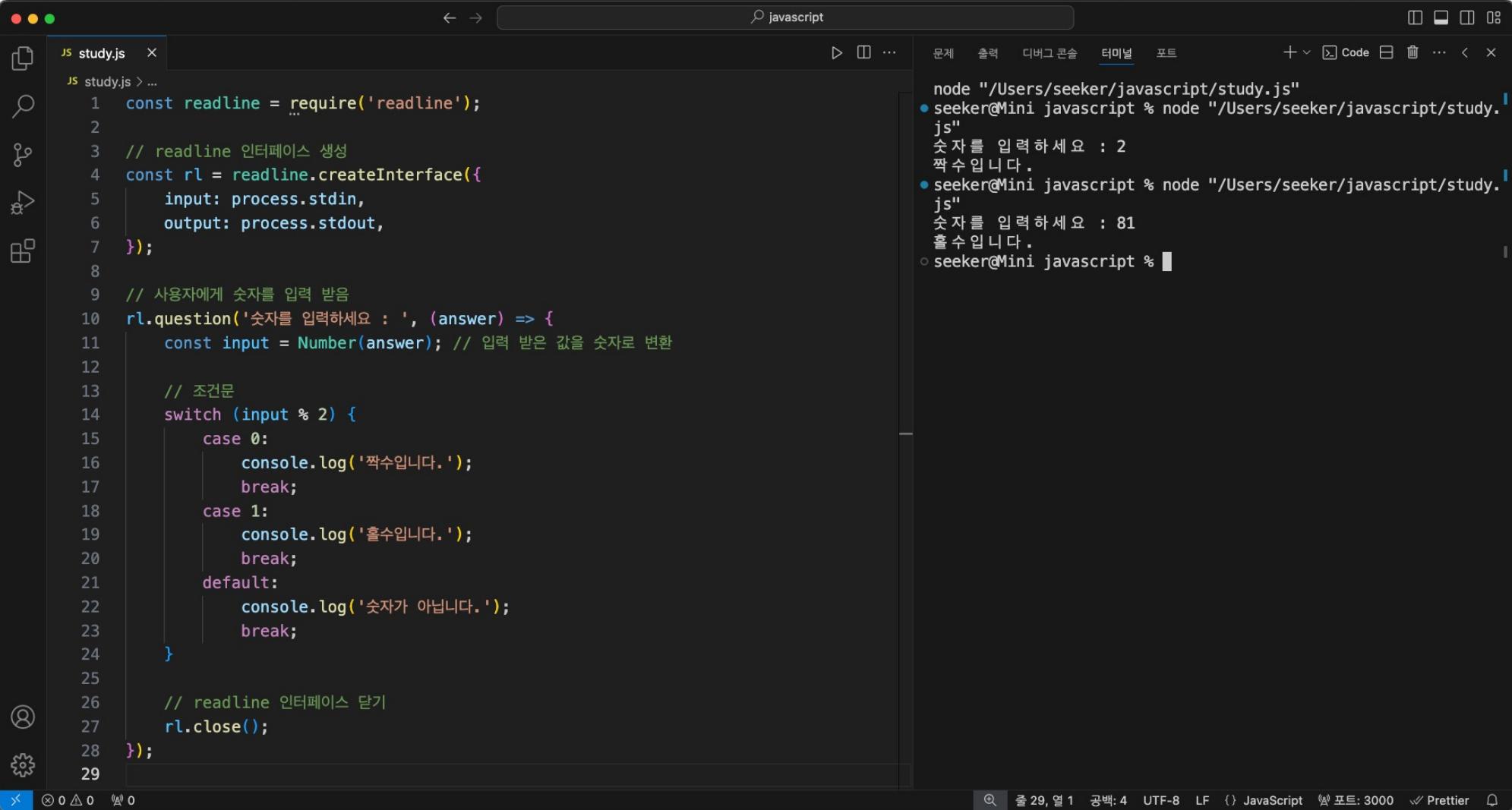
줄 16, 열 1 공백: 4 UTF-8 LF {} JavaScript 포트: 3000 Prettier

A close-up photograph of a person's hands holding a white smartphone. The person is wearing a white button-down shirt. The background is blurred.

switch 조건문

Part 3

switch 조건문

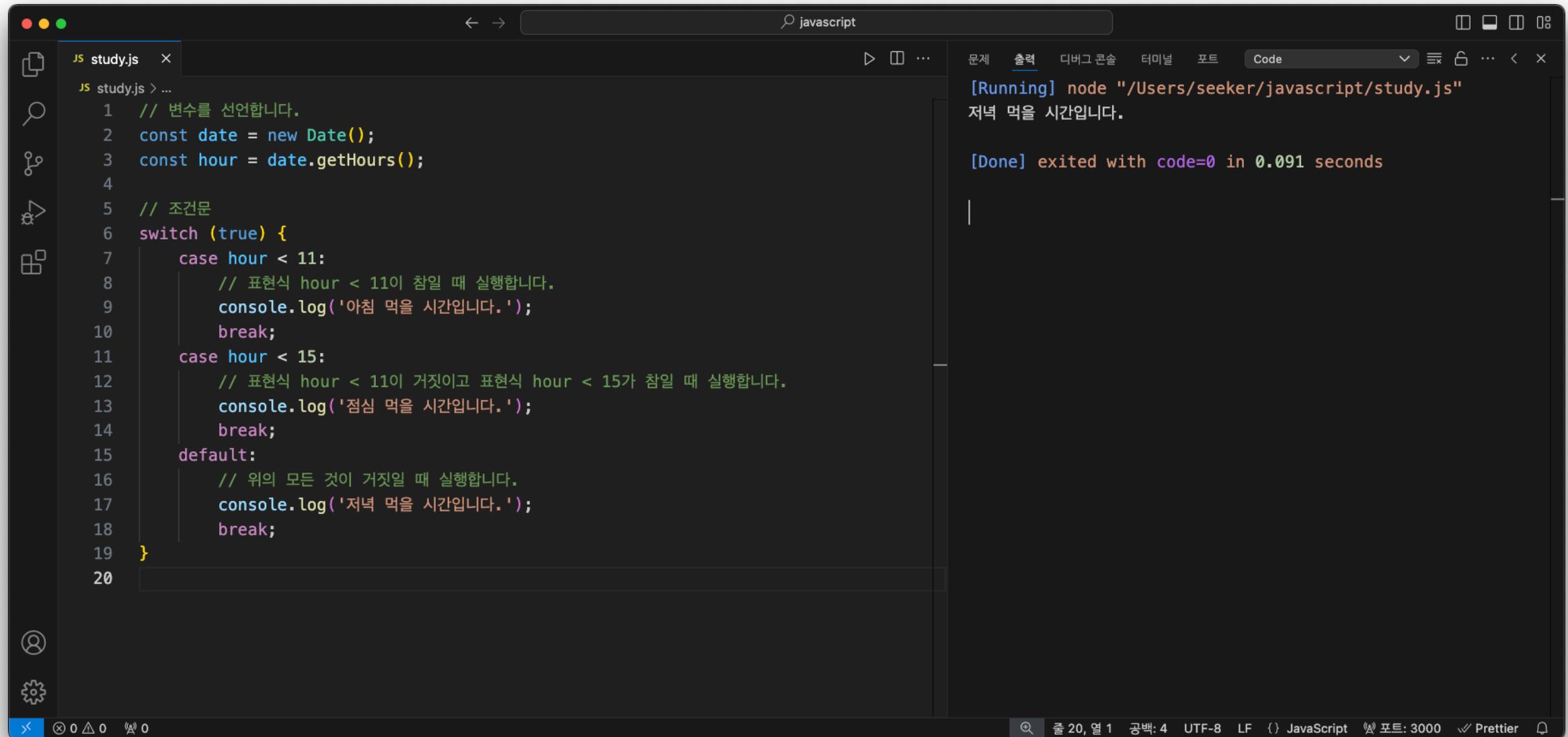


```
study.js > ...
1  const readline = require('readline');
2
3  // readline 인터페이스 생성
4  const rl = readline.createInterface({
5      input: process.stdin,
6      output: process.stdout,
7  });
8
9  // 사용자에게 숫자를 입력 받음
10 rl.question('숫자를 입력하세요 : ', (answer) => {
11     const input = Number(answer); // 입력 받은 값을 숫자로 변환
12
13     // 조건문
14     switch (input % 2) {
15         case 0:
16             console.log('짝수입니다.');
17             break;
18         case 1:
19             console.log('홀수입니다.');
20             break;
21         default:
22             console.log('숫자가 아닙니다.');
23             break;
24     }
25
26     // readline 인터페이스 닫기
27     rl.close();
28 });
29
```

node "/Users/seeker/javascript/study.js"
seeker@Mini javascript % node "/Users/seeker/javascript/study.js"
숫자를 입력하세요 : 2
짝수입니다.
seeker@Mini javascript % node "/Users/seeker/javascript/study.js"
숫자를 입력하세요 : 81
홀수입니다.
seeker@Mini javascript %

Part 3

switch 조건문 > if 조건문으로 변환



```
study.js
1 // 변수를 선언합니다.
2 const date = new Date();
3 const hour = date.getHours();
4
5 // 조건문
6 switch (true) {
7     case hour < 11:
8         // 표현식 hour < 11이 참일 때 실행합니다.
9         console.log('아침 먹을 시간입니다.');
10        break;
11     case hour < 15:
12         // 표현식 hour < 11이 거짓이고 표현식 hour < 15가 참일 때 실행합니다.
13         console.log('점심 먹을 시간입니다.');
14         break;
15     default:
16         // 위의 모든 것이 거짓일 때 실행합니다.
17         console.log('저녁 먹을 시간입니다.');
18         break;
19 }
20
```

[Running] node "/Users/seeker/javascript/study.js"
저녁 먹을 시간입니다.
[Done] exited with code=0 in 0.091 seconds

The screenshot shows a dark-themed code editor window with a tab bar at the top labeled "study.js". The main area contains a block of JavaScript code. The code uses a `switch` statement with a `true` condition to check the hour. It has three cases: one for hours less than 11 (labeled '아침'), one for hours less than 15 (labeled '점심'), and a default case for all other hours (labeled '저녁'). Each case logs a message to the console using `console.log`. The status bar at the bottom indicates the file is running in Node.js, with the output "저녁 먹을 시간입니다." displayed. The status bar also shows the file is a JavaScript file, the code exited successfully, and the execution time was 0.091 seconds.

A close-up photograph of a person's hands holding a white smartphone. The person is wearing a white button-down shirt. The background is blurred.

기타 조건문

Part 3

조건부 연산자

```
study.js > ...
1 const readline = require('readline');
2
3 // readline 인터페이스 생성
4 const rl = readline.createInterface({
5   input: process.stdin,
6   output: process.stdout,
7 });
8
9 // 사용자에게 숫자를 입력 받음
10 rl.question('숫자를 입력해주세요 : ', (answer) => {
11   const number = Number(answer); // 입력 받은 값을 숫자로 변환
12
13   // 조건문
14   const result = number >= 0 ? '0 이상의 숫자입니다.' : '0보다 작은 숫자입니다.';
15   console.log(result);
16
17   // readline 인터페이스 닫기
18   rl.close();
19 });
20
```

寻求者@Mini javascript % node "/Users/seeker/javascript/study.js"
请输入一个数字： 24
0以上的数字。
寻求者@Mini javascript % node "/Users/seeker/javascript/study.js"
请输入一个数字： -75
0比小的数字。
寻求者@Mini javascript %

줄 20, 열 1 공백: 4 LF {} JavaScript 포트: 3000 Prettier

Part 3

짝수와 홀수 구분하기

The screenshot shows a terminal window titled "javascript" running on a Mac OS X system. The window has tabs for "study.js" and "Code". The terminal output shows two runs of the script. In the first run, it asks for a number and correctly identifies 17 as odd and 18 as even. In the second run, it asks for a number and correctly identifies 38 as even and 39 as odd. The script uses the readline module to get input from the user and logs the results to the console.

```
study.js > ...
1 const readline = require('readline');
2
3 // readline 인터페이스 생성
4 const rl = readline.createInterface({
5   input: process.stdin,
6   output: process.stdout,
7 });
8
9 // 사용자에게 숫자를 입력 받음
10 rl.question('정수를 입력해주세요 : ', (answer) => {
11   // 입력이 문자열이므로 다음과 같은 코드를 사용할 수 있습니다.
12   const 입력 = answer;
13   const 끝자리 = 입력[입력.length - 1];
14
15   // 끝자리를 비교합니다.
16   if (끝자리 === '0' || 끝자리 === '2' || 끝자리 === '4' || 끝자리 === '6' || 끝자리 === '8') {
17     console.log(`#${입력}은 짝수입니다.`);
18   } else {
19     console.log(`#${입력}은 홀수입니다.`);
20   }
21
22   // readline 인터페이스 닫기
23   rl.close();
24 });
25
```

seeker@Mini javascript % node "/Users/seeker/javascript/study.js"
정수를 입력해주세요 : 17
17은 홀수입니다.
 seeker@Mini javascript % node "/Users/seeker/javascript/study.js"
정수를 입력해주세요 : 38
38은 짝수입니다.
 seeker@Mini javascript %

Part 3

짝수와 홀수 구분하기

```
study.js > ...
1 const readline = require('readline');
2
3 // readline 인터페이스 생성
4 const rl = readline.createInterface({
5   input: process.stdin,
6   output: process.stdout,
7 });
8
9 // 사용자에게 숫자를 입력 받음
10 rl.question('정수를 입력해주세요 : ', (answer) => {
11   // 입력이 문자열이므로 다음과 같은 코드를 사용할 수 있습니다.
12   const 입력 = answer;
13   const 숫자 = Number(입력);
14
15   if (숫자 % 2 === 0) {
16     console.log(`[${입력}]은 짝수입니다.`);
17   } else {
18     console.log(`[${입력}]은 홀수입니다.`);
19   }
20
21   // readline 인터페이스 닫기
22   rl.close();
23 });
24
```

seeker@Mini javascript % node "/Users/seeker/javascript/study.js"
정수를 입력해주세요 : 17
17은 홀수입니다.
seeker@Mini javascript % node "/Users/seeker/javascript/study.js"
정수를 입력해주세요 : 38
38은 짝수입니다.
seeker@Mini javascript %

줄 24, 열 1 공백: 4 UTF-8 LF () JavaScript 포트: 3000 Prettier

Part 3

학점을 기반으로 별명 붙여주기

```
study.js > ...
1 const readline = require('readline');
2
3 // readline 인터페이스 생성
4 const rl = readline.createInterface({
5   input: process.stdin,
6   output: process.stdout,
7 });
8
9 // 사용자에게 숫자를 입력 받음
10 rl.question('학점을 입력해주세요 : ', (answer) => {
11   const score = Number(answer);
12   if (score === 4.5) {
13     console.log('신');
14   } else if (4.2 <= score && score < 4.5) {
15     console.log('교수님의 사랑');
16   } else if (3.5 <= score && score < 4.2) {
17     console.log('현 체제의 수호자');
18   } else if (2.8 <= score && score < 3.5) {
19     console.log('일반인');
20   } else if (2.3 <= score && score < 2.8) {
21     console.log('일탈을 꿈꾸는 소시민');
22   } else if (1.75 <= score && score < 2.3) {
23     console.log('오락문화의 선구자');
24   } else if (1.0 <= score && score < 1.75) {
25     console.log('불가촉천민');
26   } else if (0.5 <= score && score < 1.0) {
27     console.log('자벌레');
28   } else if (0 <= score && score < 0.5) {
29     console.log('플랑크톤');
30   } else {
31     console.log('시대를 앞서가는 혁명의 씨앗');
32   }
33
34 // readline 인터페이스 닫기
35 rl.close();
36 });
37
```

node "/Users/seeker/javascript/study.js"
seeker@Mini javascript % node "/Users/seeker/javascript/study.js"
학점을 입력해주세요 : 3.5
현 체제의 수호자
seeker@Mini javascript % node "/Users/seeker/javascript/study.js"
학점을 입력해주세요 : 4.2
교수님의 사랑
seeker@Mini javascript %

Part 3

학점을 기반으로 별명 붙여주기

```
study.js
const readline = require('readline');
const rl = readline.createInterface({
  input: process.stdin,
  output: process.stdout,
});

rl.question('학점을 입력해주세요 : ', (answer) => {
  const score = Number(answer);
  if (score === 4.5) {
    console.log('신');
  } else if (4.2 <= score) {
    console.log('교수님의 사랑');
  } else if (3.5 <= score) {
    console.log('현 체제의 수호자');
  } else if (2.8 <= score) {
    console.log('일반인');
  } else if (2.3 <= score) {
    console.log('일탈을 꿈꾸는 소시민');
  } else if (1.75 <= score) {
    console.log('오락문화의 선구자');
  } else if (1.0 <= score) {
    console.log('불가촉천민');
  } else if (0.5 <= score) {
    console.log('자벌레');
  } else if (0 <= score) {
    console.log('풀랑크톤');
  } else {
    console.log('시대를 앞서가는 혁명의 씨앗');
  }
});

rl.close();
});
```

seeker@Mini javascript % node "/Users/seeker/javascript/study.js"
학점을 입력해주세요 : 3.5
현 체제의 수호자
 seeker@Mini javascript % node "/Users/seeker/javascript/study.js"
학점을 입력해주세요 : 4.2
교수님의 사랑
 seeker@Mini javascript %

Part 3

태어난 연도를 입력 받아서 떠 출력하기

```
study.js > ...
1 const readline = require('readline');
2
3 // readline 인터페이스 생성
4 const rl = readline.createInterface({
5   input: process.stdin,
6   output: process.stdout
7 });
8
9 // 사용자에게 숫자를 입력 받음
10 rl.question('태어난 해를 입력해주세요 : ', (answer) => {
11   const year = Number(answer);
12   const e = year % 12;
13
14   let result
15   if (e === 0) { result = '원숭이' }
16   else if (e === 1) { result = '닭' }
17   else if (e === 2) { result = '개' }
18   else if (e === 3) { result = '돼지' }
19   else if (e === 4) { result = '쥐' }
20   else if (e === 5) { result = '소' }
21   else if (e === 6) { result = '호랑이' }
22   else if (e === 7) { result = '토끼' }
23   else if (e === 8) { result = '용' }
24   else if (e === 9) { result = '뱀' }
25   else if (e === 10) { result = '말' }
26   else if (e === 11) { result = '양' }
27   console.log(`${year}년에 태어났다면 ${result} 떠입니다.`);
28
29 // readline 인터페이스 닫기
30 rl.close();
31 });
32
```

seeker@Mini javascript % node "/Users/seeker/javascript/study.js"
태어난 해를 입력해주세요 : 1978
1978년에 태어났다면 말 떠입니다.
seeker@Mini javascript %

줄 32, 열 1 공백: 4 UTF-8 LF {} JavaScript 포트: 3000 Prettier

Part 3

태어난 연도를 입력 받아서 땡 출력하기

```
study.js > ...
1 const readline = require('readline');
2
3 // readline 인터페이스 생성
4 const rl = readline.createInterface({
5   input: process.stdin,
6   output: process.stdout,
7 });
8
9 // 사용자에게 숫자를 입력 받음
10 rl.question('태어난 해를 입력해주세요 : ', (answer) => {
11   const year = Number(answer);
12   const tti = '원숭이,닭,개,돼지,쥐,소,호랑이,토끼,용,뱀,말,양'.split(',');
13
14   console.log(`#${year}년에 태어났다면 ${tti[year % 12]} 땡입니다.`);
15
16   // readline 인터페이스 닫기
17   rl.close();
18 });
19
```

seeker@Mini javascript % node "/Users/seeker/javascript/study.js"
태어난 해를 입력해주세요 : 1978
1978년에 태어났다면 말 땡입니다.
seeker@Mini javascript %

줄 19, 열 1 공백: 4 LF {} JavaScript 포트: 3000 Prettier

목차

a table of contents

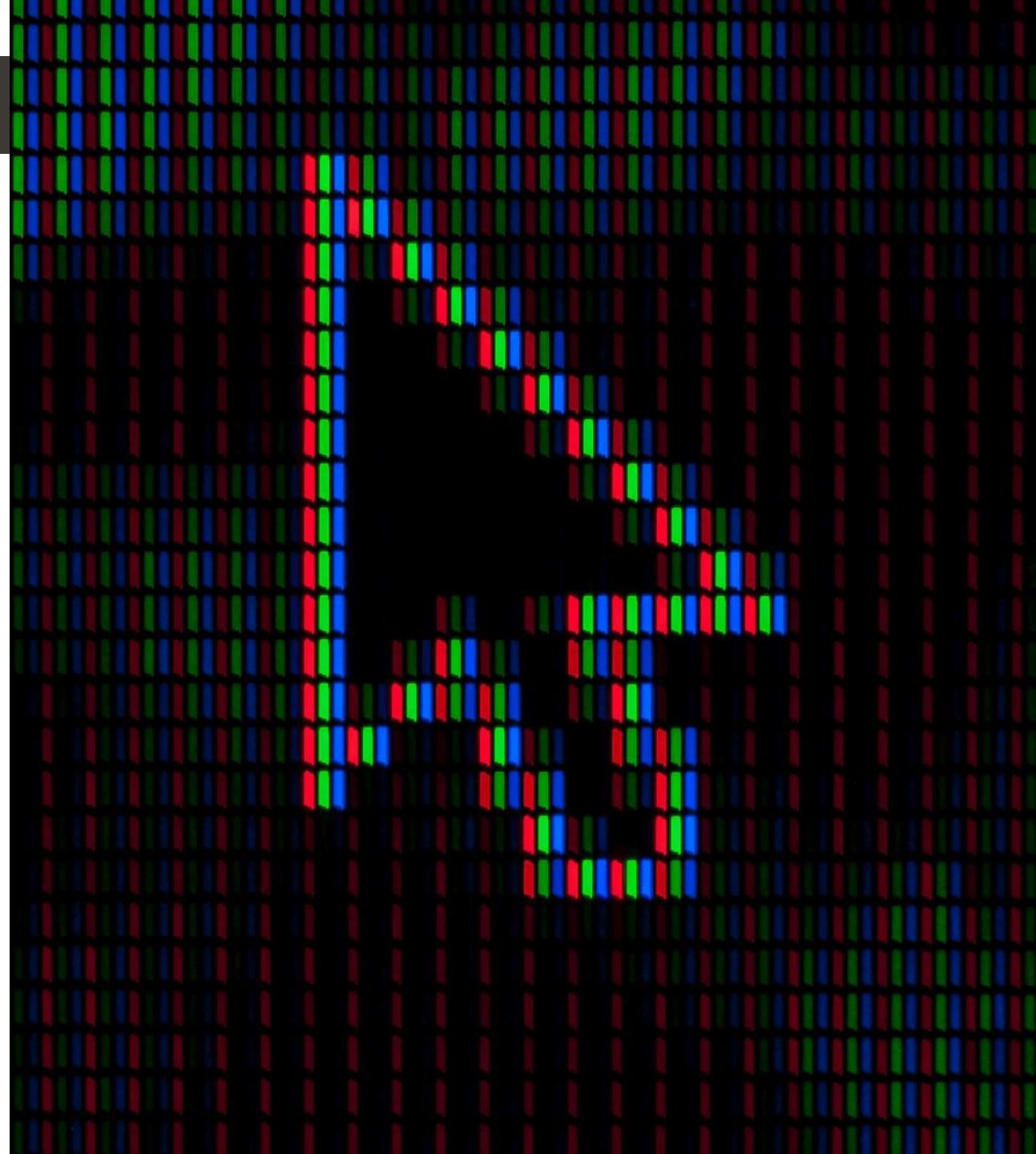
1 자바스크립트 기초

2 자료와 변수

3 조건문

4 반복문

5 함수



A close-up photograph of a person's hands holding a white smartphone. The person is wearing a white button-down shirt. A large, semi-transparent white rectangular box is overlaid on the center-left portion of the image. Inside this box, the Korean characters "백일" (Baek-il) are displayed in a bold, black, sans-serif font.

백일

Part 4

배열

The screenshot shows a browser's developer tools open to the 'Console' tab. The search bar at the top contains the text 'javascript'. The left sidebar has icons for file, search, and settings, with 'study.html' selected. The main console area displays the following JavaScript code and its execution:

```
> let str = '안녕하세요'  
< undefined  
> str[0]  
< '안'  
> str[1]  
< '녕'  
> str[2]  
< '하'  
> str[3]  
< '세'  
> str[4]  
< '요'  
> str[str.length-1]  
< '요'  
>
```

The code defines a string 'str' with the value '안녕하세요' and then iterates through its characters using square bracket notation from index 0 to index 4, followed by the last character at index 4 (length-1).

Part 4

배열 만들기

The screenshot shows a browser developer tools interface with the 'Console' tab selected. The left sidebar lists files: 'study.html X' and 'study.html'. The main area displays the following JavaScript code and its resulting object structure:

```
> const array = [273, 'String', true, function() { }, {}, [273, 103]]  
< undefined  
> array  
< ▾ (6) [273, 'String', true, f, {}, Array(2)] ⓘ  
  0: 273  
  1: "String"  
  2: true  
  3: f ()  
  4: {}  
  5: (2) [273, 103]  
    length: 6  
  ► [[Prototype]]: Array(0)  
> |
```

The array contains six elements: 273, the string "String", the boolean true, a function (f), an empty object {}, and another array [273, 103] which has a length of 6.

Part 4

배열 요소에 접근하기

The screenshot shows a browser's developer tools open to the 'Console' tab. The search bar at the top contains the text 'javascript'. The left sidebar has icons for file, search, and settings, with 'study.html' selected. The main console area displays the following JavaScript code and its execution results:

```
> const numbers = [273, 52, 103, 32]
< undefined
> numbers[0]
< 273
> numbers[1]
< 52
> numbers[1+1]
< 103
> numbers[1*3]
< 32
> numbers
< ▾ (4) [273, 52, 103, 32] i
  0: 273
  1: 52
  2: 103
  3: 32
  length: 4
  ▶ [[Prototype]]: Array(0)
```

The code defines an array named 'numbers' with four elements: 273, 52, 103, and 32. It then demonstrates various ways to access these elements using square bracket notation, including direct indices like 'numbers[0]', 'numbers[1]', and 'numbers[1+1]', as well as more complex expressions like 'numbers[1*3]'. The final line shows the array itself, followed by its length and prototype information.

Part 4

배열 요소 개수 확인하기

The screenshot shows a browser developer tools interface with the 'Console' tab selected. The left sidebar shows an open file named 'study.html'. The console output is as follows:

```
> const fruits = ['배', '사과', '키위', '바나나']
< undefined
> fruits.length
< 4
> fruits[fruits.length - 1]
< '바나나'
> fruits
< ▶ (4) [ '배', '사과', '키위', '바나나' ] ⓘ
    0: "배"
    1: "사과"
    2: "키위"
    3: "바나나"
    length: 4
    [[Prototype]]: Array(0)
>
```

The code defines an array 'fruits' with four elements: 배, 사과, 키위, and 바나나. It then checks the length of the array, which is 4. It also prints the last element at index 3, which is 바나나. Finally, it logs the entire array to the console.

Part 4

배열 뒷부분에 요소 추가하기 > push 메소드 사용

The screenshot shows a browser's developer tools console window. The title bar includes the URL 'javascript' and several tabs: Elements, Console, Sources, Network, and Performance. The 'Console' tab is active. The sidebar on the left lists files: 'study.html X' and 'study.html'. The main area displays the following JavaScript code and its execution:

```
> const todos = ['우유 구매', '업무 메일 확인하기', '필라테스 수업']
< undefined
> todos
<▶ (3) ['우유 구매', '업무 메일 확인하기', '필라테스 수업']
> todos.push('저녁 식사 준비하기')
< 4
> todos.push('피아노 연습하기')
< 5
> todos
<▶ (5) ['우유 구매', '업무 메일 확인하기', '필라테스 수업', '저녁 식사 준비하기', '피아노 연습하기']
    ▶ 0: "우유 구매"
    ▶ 1: "업무 메일 확인하기"
    ▶ 2: "필라테스 수업"
    ▶ 3: "저녁 식사 준비하기"
    ▶ 4: "피아노 연습하기"
    ▶ length: 5
    ▶ [[Prototype]]: Array(0)
```

The console output shows the initial array 'todos' with three items, followed by two calls to the 'push' method adding new items ('저녁 식사 준비하기' and '피아노 연습하기'). The final state of the array is shown with five items, indexed from 0 to 4.

Part 4

배열 뒷부분에 요소 추가하기 > 인덱스 사용

The screenshot shows a browser's developer tools open to the 'Console' tab. On the left, there's a sidebar with icons for file operations and a search bar labeled 'javascript'. The main area displays the following JavaScript code:

```
> const fruitsA = ['사과', '배', '바나나']
< undefined
> fruitsA[10] = '귤'
< '귤'
> fruitsA
< ▶ (11) ['사과', '배', '바나나', empty × 7, '귤']
> const fruitsB = ['사과', '배', '바나나']
< undefined
> fruitsB[fruitsB.length] = '귤'
< '귤'
> fruitsB
< ▶ (4) ['사과', '배', '바나나', '귤']
>
```

The code demonstrates two ways to add elements to the end of arrays. In the first example, `fruitsA[10] = '귤'` is used, resulting in an array of length 11. In the second example, `fruitsB[fruitsB.length] = '귤'` is used, resulting in an array of length 4.

Part 4

배열 요소 제거하기 > 인덱스 사용

The screenshot shows a browser's developer tools interface with the 'Console' tab selected. On the left, there's a sidebar with various icons and a file tree showing 'study.html' is open. The main area is the JavaScript console.

```
> const itemsA = ['사과', '배', '바나나']
< undefined
> itemsA.splice(2, 1) //배열의 2번째 인덱스로부터 1개 요소를 제거
< [ '바나나' ]
> itemsA
< [ '사과', '배' ]
>
```

The code demonstrates using the `splice` method to remove one element from an array. It starts by defining an array `itemsA` with three elements: '사과', '배', and '바나나'. Then, it uses `splice(2, 1)` to remove the element at index 2 ('바나나') and returns the modified array ['사과', '배'].

Part 4

배열 요소 제거하기 > 값 사용

The screenshot shows a browser developer tools window with the 'Console' tab selected. The code in the console is as follows:

```
> const itemsB = ['사과', '배', '바나나']
< undefined
> const index = itemsB.indexOf('바나나')
< undefined
> index
< 2
> itemsB.splice(index, 1) //배열의 2번째 인덱스에 있는 1개의 요소를 제거
< [ '사과', '배' ]
> itemsB
< [ '사과', '배' ]
> itemsB.indexOf('바나나')
< -1
> |
```

The code demonstrates using the `splice` method to remove the element at index 2 from the `itemsB` array. The `indexOf` method is used to find the index of the element '바나나'.

Part 4

배열의 특정 위치에 요소 추가하기

The screenshot shows a browser's developer tools interface with the 'Console' tab selected. On the left, there is a sidebar with various icons and a file tree showing 'study.html' is open. The main area displays the following JavaScript code:

```
> const itemsD = ['사과', '귤', '바나나', '오렌지']
< undefined
> itemsD.splice(1, 0, '양파')
< ▶ []
> itemsD
< ▶ (5) ['사과', '양파', '귤', '바나나', '오렌지']
> |
```

The code demonstrates using the `splice` method to insert the string '양파' (onion) at index 1 of the `itemsD` array. The output shows the original array being modified in place, resulting in a new array with five elements: ['사과', '양파', '귤', '바나나', '오렌지'].

A close-up photograph of a person's hands holding a white smartphone. The person is wearing a white button-down shirt. A large, semi-transparent white rectangular box is overlaid on the center-left portion of the image. Inside this box, the Korean characters "만복문" are displayed in a bold, black, sans-serif font.

만복문

Part 4

for in 반복문

The screenshot shows a dark-themed code editor interface. On the left is the code editor pane with a file named "study.js". The code contains a simple for-in loop that logs each item from an array of todos to the console. The right side shows the terminal pane where the script is running and its output is displayed.

```
study.js > ...
1 const todos = ['우유구매', '업무 메일 확인하기', '필라테스 수업'];
2
3 for (const i in todos) {
4     console.log(`#${i}번째 할 일: ${todos[i]}`);
5 }
6
```

[Running] node "/Users/seeker/javascript/study.js"
0번째 할 일: 우유구매
1번째 할 일: 업무 메일 확인하기
2번째 할 일: 필라테스 수업
[Done] exited with code=0 in 0.101 seconds

At the bottom, there are status icons and text indicating the current file is "study.js", it has 6 lines, and the encoding is UTF-8. It also shows the current mode is "JavaScript".

Part 4

for of 반복문

The screenshot shows a dark-themed code editor interface. On the left is the code editor pane with a file named "study.js". The code contains a simple for-of loop that logs each item in the "todos" array to the console. The output pane on the right shows the results of running the script with Node.js.

```
study.js > ...
1 const todos = ['우유구매', '업무 메일 확인하기', '필라테스 수업'];
2
3 for (const todo of todos) {
4     console.log(`오늘의 할 일: ${todo}`);
5 }
6
```

[Running] node "/Users/seeker/javascript/study.js"
오늘의 할 일: 우유구매
오늘의 할 일: 업무 메일 확인하기
오늘의 할 일: 필라테스 수업
[Done] exited with code=0 in 0.109 seconds

줄 6, 열 1 공백: 4 UTF-8 LF () JavaScript 포트: 3000 Prettier

Part 4

for 반복문

A screenshot of a dark-themed code editor, likely VS Code, demonstrating a for loop. The left sidebar shows icons for file, search, and settings. The main area has tabs for 'study.js' and 'index.html'. The 'study.js' tab contains the following code:

```
study.js > ...
1  for (let i = 0; i < 5; i++) {
2      console.log(`[${i}]번째 반복입니다.`);
3  }
4
```

The right side shows the terminal output:

```
[Running] node "/Users/seeker/javascript/study.js"
0번째 반복입니다.
1번째 반복입니다.
2번째 반복입니다.
3번째 반복입니다.
4번째 반복입니다.

[Done] exited with code=0 in 0.088 seconds
```

The status bar at the bottom shows: ⌛ 0 △ 0 ⌂ 0 콜 4, 열 1 공백: 4 UTF-8 LF {} JavaScript ⌂ 포트: 3000 ✅ Prettier

Part 4

for 반복문

```
study.js > ...
1 let output = 0;
2 for (let i = 1; i <= 100; i++) {
3     output += i;
4 }
5 console.log(`1~100까지 숫자를 모두 더하면 ${output}입니다.`);
6
```

[Running] node "/Users/seeker/javascript/study.js"
1~100까지 숫자를 모두 더하면 5050입니다.

[Done] exited with code=0 in 0.097 seconds

줄 6, 열 1 공백: 4 UTF-8 LF {} JavaScript 포트: 3000 ✓ Prettier

Part 4

for 반복문

The screenshot shows a dark-themed code editor interface with a sidebar on the left containing icons for file operations, search, and settings. The main area displays a file named 'study.js' with the following content:

```
study.js > ...
1 const todos = ['우유 구매', '업무 메일 확인하기', '필라테스 수업'];
2
3 for (let i = 0; i < todos.length; i++) {
4     console.log(`#${i}번째 할 일: ${todos[i]}`);
5 }
6
```

The status bar at the bottom indicates the file is saved, has 6 rows and 1 column, uses UTF-8 encoding, is a JavaScript file, and has port 3000 open.

The right side of the interface shows the output of running the script with Node.js:

[Running] node "/Users/seeker/javascript/study.js"

0번째 할 일: 우유 구매
1번째 할 일: 업무 메일 확인하기
2번째 할 일: 필라테스 수업

[Done] exited with code=0 in 0.086 seconds

Part 4

for 반복문

The screenshot shows a dark-themed code editor interface with a sidebar on the left containing icons for file operations, search, and settings. The main area displays a file named 'study.js' with the following content:

```
study.js > ...
1 const todos = ['우유 구매', '업무 메일 확인하기', '필라테스 수업'];
2
3 for (let i = todos.length - 1; i >= 0; i--) {
4     console.log(`#${i}번째 할 일: ${todos[i]}`);
5 }
6
```

To the right of the code editor is a terminal window titled 'javascript' with the following output:

[Running] node "/Users/seeker/javascript/study.js"
2번째 할 일: 필라테스 수업
1번째 할 일: 업무 메일 확인하기
0번째 할 일: 우유 구매
[Done] exited with code=0 in 0.092 seconds

At the bottom of the terminal window, there are status indicators for file changes, encoding (UTF-8), line endings (LF), file type (JavaScript), port (3000), and Prettier.

Part 4

while 반복문

A screenshot of a dark-themed code editor, likely VS Code, showing a JavaScript file named `study.js`. The code contains a simple `while` loop that logs the indices and values of an array to the console.

```
study.js
1 let i = 0;
2 const array = [1, 2, 3, 4, 5];
3
4 while (i < array.length) {
5     console.log(`${i} : ${array[i]}`);
6     i++;
7 }
8
```

The output window shows the results of the execution:

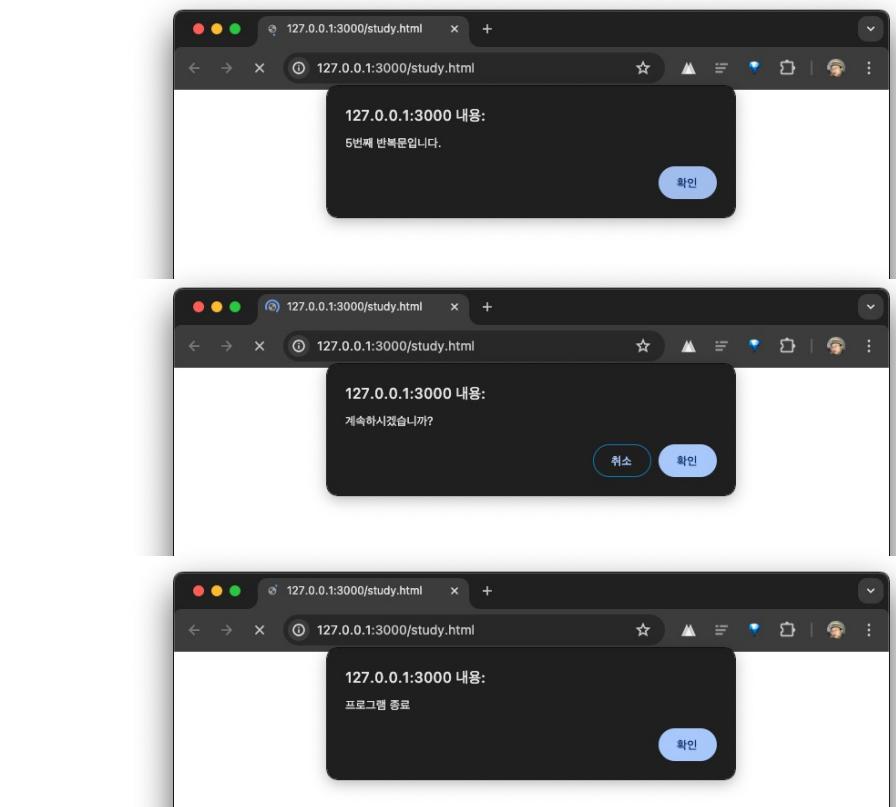
```
[Running] node "/Users/seeker/javascript/study.js"
0 : 1
1 : 2
2 : 3
3 : 4
4 : 5

[Done] exited with code=0 in 0.414 seconds
```

The status bar at the bottom indicates the file is a JavaScript file (JavaScript), has 8 lines and 4 columns, and is using UTF-8 encoding.

Part 4

break 키워드



The screenshot shows a code editor on the left and a browser window on the right. The code editor displays an HTML file named 'study.html' with the following content:

```
<!DOCTYPE html>
<html>
  <head>
    <title></title>
    <script>
      // 반복문
      for (let i = 0; true; i++) {
        alert(i + '번째 반복문입니다.');

        // 진행 여부를 물어봅니다.
        const isContinue = confirm('계속하시겠습니까?');
        if (!isContinue) {
          break;
        }

        // 프로그램의 종료를 확인합니다.
        alert('프로그램 종료');
      }
    </script>
  </head>
  <body></body>
</html>
```

The browser window shows three consecutive alert dialogs. The first dialog displays "127.0.0.1:3000 내용: 5번째 반복문입니다." (Content: 127.0.0.1:3000 내용: 5번째 반복문입니다.). The second dialog displays "127.0.0.1:3000 내용: 계속하시겠습니까?" (Content: 127.0.0.1:3000 내용: 계속하시겠습니까?). The third dialog displays "127.0.0.1:3000 내용: 프로그램 종료" (Content: 127.0.0.1:3000 내용: 프로그램 종료).

Part 4

continue 키워드

The screenshot shows a dark-themed code editor interface. On the left, there's a sidebar with various icons: a file icon, a search icon, a refresh icon, a settings gear icon, and a user profile icon. The main area has a title bar with a back arrow, a forward arrow, and a search field containing "javascript". Below the title bar is a tab bar with "study.js" and "...". The code editor itself displays the following JavaScript code:

```
study.js > ...
1 // 반복문
2 for (let i = 0; i < 5; i++) {
3     // 현재 반복 작업을 중지하고 다음 반복 작업을 수행합니다.
4     continue;
5     console.log(i);
6 }
7
```

To the right of the code editor is an output panel titled "Output" (출력). It shows the command "node '/Users/seeker/javascript/study.js'" being run, followed by the message "[Done] exited with code=0 in 0.082 seconds". At the bottom of the output panel, there are status indicators: a magnifying glass icon, a file icon, a date and time indicator ("줄 5, 일 20"), a character count ("공백: 4"), a line separator ("LF"), a file type indicator ("JavaScript"), a port number ("포트: 3000"), and a Prettier logo.

Part 4

continue 키워드

```
study.js
1 // let 변수를 선언합니다.
2 let output = 0;
3
4 // 반복문
5 for (let i = 1; i <= 10; i++) {
6     // 조건부
7     if (i % 2 === 1) {
8         // 홀수면 현재 반복을 중지하고 다음 반복을 수행합니다.
9         continue;
10    }
11    output += i;
12 }
13
14 // 출력합니다.
15 console.log(output);
16
```

[Running] node "/Users/seeker/javascript/study.js"
30
[Done] exited with code=0 in 0.416 seconds

줄 16, 열 1 공백: 4 LF {} JavaScript 포트: 3000 Prettier

Part 4

중첩 반복문을 사용하는 피라미드

The screenshot shows a code editor interface with a dark theme. On the left is a sidebar with various icons. The main area has a search bar at the top right with the text "javascript". A tab labeled "study.js" is active. The code in the editor is:

```
study.js > ...
1 // let 변수를 선언합니다.
2 let output = '';
3
4 // 중첩 반복문
5 for (let i = 1; i < 10; i++) {
6     for (let j = 0; j < i; j++) {
7         output += '*';
8     }
9     output += '\n';
10}
11
12 // 출력합니다.
13 console.log(output);
14
```

The output in the terminal tab is:

```
[Running] node "/Users/seeker/javascript/study.js"
*
**
***
****
*****
*****
*****
*****
*****
*****
[Done] exited with code=0 in 0.092 seconds
```

At the bottom, there are status indicators: "줄 14, 열 1" (Line 14, Column 1), "공백: 4" (4 spaces), "UTF-8 LF", "JavaScript", "포트: 3000", and "Prettier".

Part 4

중첩 반복문을 사용하는 피라미드

```
study.js
1 // let 변수를 선언합니다.
2 let output = '';
3
4 // 중첩 반복문
5 for (let i = 1; i < 15; i++) {
6     for (let j = 15; j > i; j--) {
7         output += ' ';
8     }
9     for (let k = 0; k < 2 * i - 1; k++) {
10        output += '*';
11    }
12    output += '\n';
13 }
14
15 // 출력합니다.
16 console.log(output);
17
```

[Running] node "/Users/seeker/javascript/study.js"

```
*****
 ****
 *****
 *****
```

[Done] exited with code=0 in 0.089 seconds

목차

a table of contents

- 1 자바스크립트 기초
- 2 자료와 변수
- 3 조건문
- 4 반복문
- 5 함수

A close-up photograph of a person's hands holding a white smartphone. The person is wearing a white button-down shirt. A large, semi-transparent white rectangular box is overlaid on the center-left portion of the image. Inside this box, the Korean text "함수 기본" is displayed in a bold, black, sans-serif font.

함수 기본

$$f(x) = y$$

입력 (input) = x



함수 (Function) : f



출력 (output) = y

Part 5

익명 함수

```
study.js
1 // 변수를 선언합니다.
2 const 함수 = function () {
3     console.log('함수 내부의 코드입니다 ... 1');
4     console.log('함수 내부의 코드입니다 ... 2');
5     console.log('함수 내부의 코드입니다 ... 3');
6     console.log('');
7 };
8
9 // 함수를 호출합니다.
10 함수();
11 함수();
12
13 // 출력합니다.
14 console.log(typeof 함수);
15 console.log(함수);
16
```

[Running] node "/Users/seeker/javascript/study.js"

```
함수 내부의 코드입니다 ... 1
함수 내부의 코드입니다 ... 2
함수 내부의 코드입니다 ... 3

함수 내부의 코드입니다 ... 1
함수 내부의 코드입니다 ... 2
함수 내부의 코드입니다 ... 3

function
[Function: 함수]

[Done] exited with code=0 in 0.101 seconds
```

```
f () {
    console.log('함수 내부의 코드입니다 ... 1');
    console.log('함수 내부의 코드입니다 ... 2');
    console.log('함수 내부의 코드입니다 ... 3');
    console.log('');
}
```

줄 16, 열 1 공백: 4 UTF-8 LF {} JavaScript 포트: 3000 ✓ Prettier

Part 5

선언적 함수

The screenshot shows a dark-themed code editor interface with a terminal window on the right. The code editor has a sidebar with icons for file, search, and settings. The main area shows a file named 'study.js' with the following content:

```
study.js > ...
1 // 함수를 생성합니다.
2 function 함수() {
3     console.log('함수 내부의 코드입니다 ... 1');
4     console.log('함수 내부의 코드입니다 ... 2');
5     console.log('함수 내부의 코드입니다 ... 3');
6     console.log('');
7 }
8
9 // 함수를 호출합니다.
10 함수();
11 함수();
12
13 // 출력합니다.
14 console.log(typeof 함수);
15 console.log(함수);
16
```

The terminal window on the right displays the execution of the script:

```
[Running] node "/Users/seeker/javascript/study.js"
함수 내부의 코드입니다 ... 1
함수 내부의 코드입니다 ... 2
함수 내부의 코드입니다 ... 3

함수 내부의 코드입니다 ... 1
함수 내부의 코드입니다 ... 2
함수 내부의 코드입니다 ... 3

function
[Function: 함수]

[Done] exited with code=0 in 0.624 seconds
```

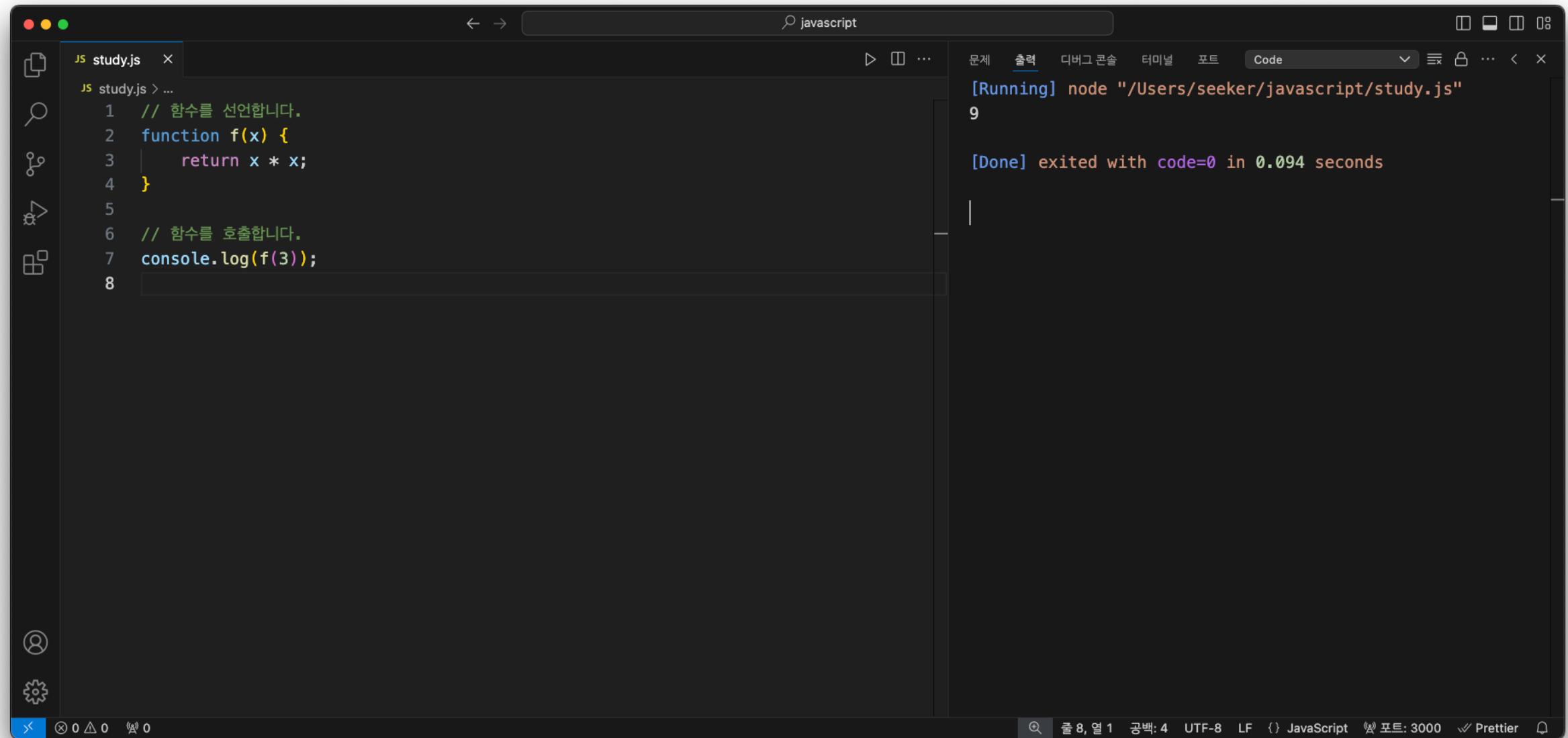
A callout box highlights the function definition from line 2:

```
f 함수() {
    console.log('함수 내부의 코드입니다 ... 1');
    console.log('함수 내부의 코드입니다 ... 2');
    console.log('함수 내부의 코드입니다 ... 3');
    console.log('');
}
```

The bottom status bar shows file information: 줄 16, 열 1, 공백: 4, UTF-8, LF, JavaScript, 포트: 3000, Prettier.

Part 5

매개변수와 리턴 값



```
study.js
1 // 함수를 선언합니다.
2 function f(x) {
3     return x * x;
4 }
5
6 // 함수를 호출합니다.
7 console.log(f(3));
8
```

[Running] node "/Users/seeker/javascript/study.js"
9
[Done] exited with code=0 in 0.094 seconds

줄 8, 열 1 공백: 4 UTF-8 LF {} JavaScript 포트: 3000 ✓ Prettier

Part 5

기본적인 함수 예제 > 윤년 확인

```
study.js
1 function isLeapYear(year) {
2     return (year % 4 === 0 && year % 100 !== 0) || year % 400 === 0;
3 }
4
5 console.log(`2020년은 윤년일까? === ${isLeapYear(2020)} `);
6 console.log(`2010년은 윤년일까? === ${isLeapYear(2010)} `);
7 console.log(`2000년은 윤년일까? === ${isLeapYear(2000)} `);
8 console.log(`1900년은 윤년일까? === ${isLeapYear(1900)} `);
```

[Running] node "/Users/seeker/javascript/study.js"
2020년은 윤년일까? === true
2010년은 윤년일까? === false
2000년은 윤년일까? === true
1900년은 윤년일까? === false
[Done] exited with code=0 in 0.096 seconds

문서 출력 디버그 콘솔 터미널 포트 Code

줄 9, 열 1 공백: 4 UTF-8 LF () JavaScript 포트: 3000 Prettier

Part 5

기본적인 함수 예제 > A부터 B까지 더하기

The screenshot shows a dark-themed code editor interface with a sidebar on the left containing icons for file operations, search, and settings. The main area displays a file named 'study.js' with the following content:

```
study.js > ...
1  function sumAll(a, b) {
2    let output = 0;
3    for (let i = a; i <= b; i++) {
4      output += i;
5    }
6    return output;
7  }
8
9  console.log(`1부터 100까지의 합: ${sumAll(1, 100)}`);
10 console.log(`1부터 500까지의 합: ${sumAll(1, 500)}`);
11
```

The right side of the interface features a toolbar with tabs for '문제', '출력' (Output), '디버그 콘솔' (Debug Console), '터미널' (Terminal), and '포트' (Port). The 'Output' tab is active, showing the command `[Running] node "/Users/seeker/javascript/study.js"`. Below it, two lines of text are displayed: `1부터 100까지의 합: 5050` and `1부터 500까지의 합: 125250`. At the bottom of the output pane, the message `[Done] exited with code=0 in 0.106 seconds` is shown. The bottom status bar includes icons for file operations, a search field, and file statistics like '줄 11, 열 1' and 'UTF-8 LF'.

Part 5

기본적인 함수 예제 > 최소값 구하기

```
study.js
function min(array) {
  let output = array[0];
  for (const item of array) {
    // 현재 output 보다 더 작은 item이 있다면
    if (output > item) {
      // output 값을 item으로 변경
      output = item;
    }
  }
  return output;
}

const testArray = [52, 273, 32, 103, 275, 24, 57];
console.log(`[${testArray}]중에서`);
console.log(`최솟값 = ${min(testArray)}`);

```

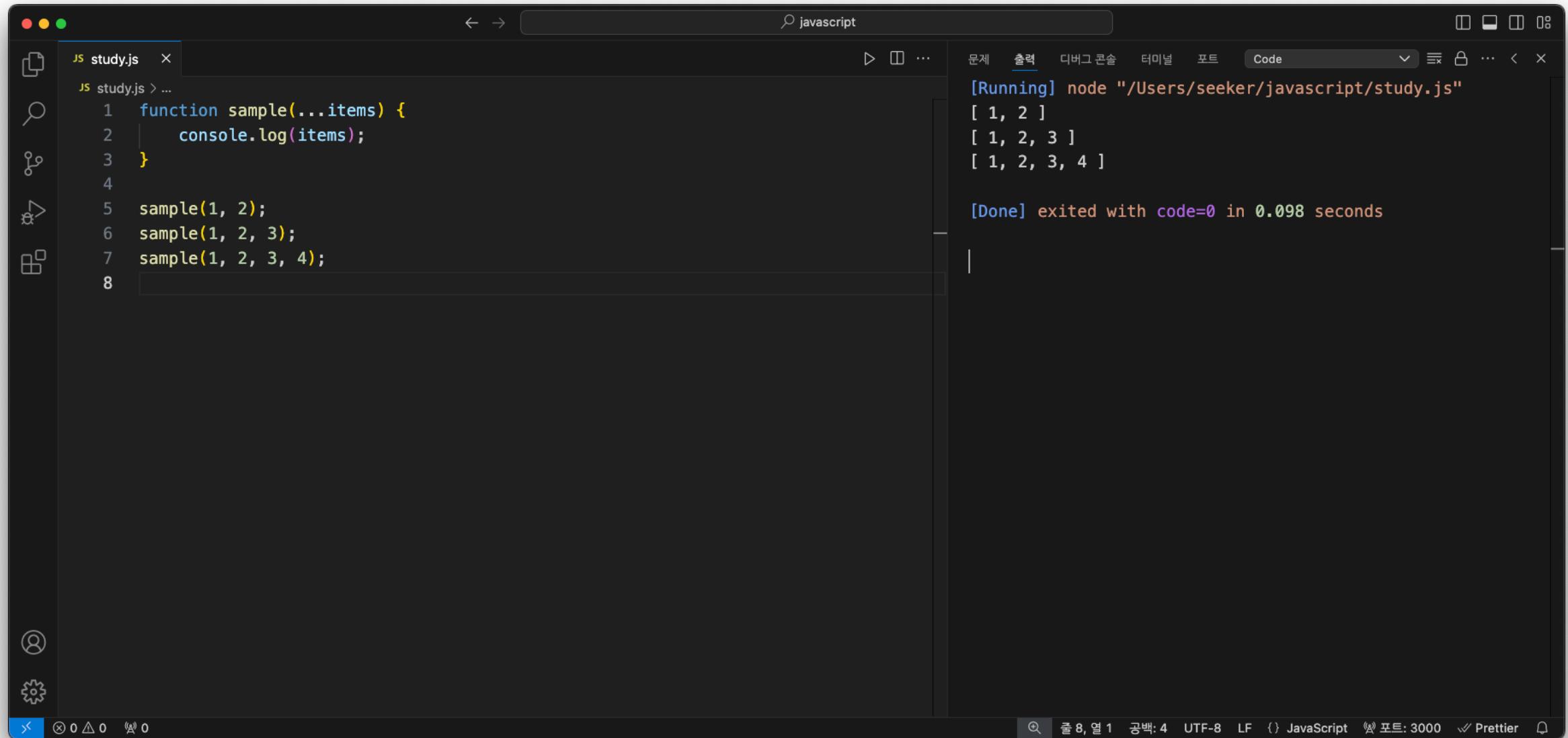
[Running] node "/Users/seeker/javascript/study.js"
52,273,32,103,275,24,57중에서
최솟값 = 24

[Done] exited with code=0 in 0.091 seconds

줄 16, 열 1 공백: 4 UTF-8 LF () JavaScript 포트: 3000 Prettier

Part 5

나머지 매개변수



```
study.js
1 function sample(...items) {
2   console.log(items);
3 }
4
5 sample(1, 2);
6 sample(1, 2, 3);
7 sample(1, 2, 3, 4);
8
```

[Running] node "/Users/seeker/javascript/study.js"
[1, 2]
[1, 2, 3]
[1, 2, 3, 4]
[Done] exited with code=0 in 0.098 seconds

줄 8, 열 1 공백: 4 UTF-8 LF {} JavaScript 포트: 3000 Prettier

Part 5

나머지 매개변수

```
study.js > ...
1 // 나머지 매개변수를 사용한 함수 만들기
2 function min(...items) {
3     // 매개변수 items는 배열처럼 사용합니다.
4     let output = items[0];
5     for (const item of items) {
6         if (output > item) {
7             output = item;
8         }
9     }
10    return output;
11 }
12
13 // 함수 호출하기
14 console.log('min(52,273,32,103,275,24,57)');
15 console.log(`= ${min(52, 273, 32, 103, 275, 24, 57)}`);
16
```

[Running] node "/Users/seeker/javascript/study.js"
min(52,273,32,103,275,24,57)
=24

[Done] exited with code=0 in 0.084 seconds

줄 16, 열 1 공백: 4 UTF-8 LF () JavaScript 포트: 3000 Prettier

Part 5

나머지 매개변수 > 일반 매개변수 조합

The screenshot shows a dark-themed code editor interface. On the left is the code editor pane with a file named "study.js". The code defines a function "sample" that takes three parameters: "a", "b", and "...c". It logs the values of "a", "b", and "c" to the console. Three calls to this function are made: "sample(1, 2)", "sample(1, 2, 3)", and "sample(1, 2, 3, 4)". The output pane on the right shows the results of these executions. The first call logs "1 2 []", the second "1 2 [3]", and the third "1 2 [3, 4]". Below the output, a message indicates "[Done] exited with code=0 in 0.086 seconds". The bottom status bar shows file information like "줄 7, 열 1" and encoding "UTF-8".

```
study.js > ...
1  function sample(a, b, ...c) {
2 |   console.log(a, b, c);
3 }
4 sample(1, 2);
5 sample(1, 2, 3);
6 sample(1, 2, 3, 4);
7
```

[Running] node "/Users/seeker/javascript/study.js"

1 2 []
1 2 [3]
1 2 [3, 4]

[Done] exited with code=0 in 0.086 seconds

줄 7, 열 1 공백: 4 UTF-8 LF {} JavaScript 포트: 3000 Prettier

Part 5

나머지 매개변수

```
study.js > ...
1  function min(first, ...rests) {
2      // 변수 선언하기
3      let output;
4      let items;
5
6      // 매개변수의 자료형에 따라 조건 분기하기
7      if (Array.isArray(first)) {
8          output = first[0];
9          items = first;
10     } else if (typeof first === 'number') {
11         output = first;
12         items = rests;
13     }
14
15     // 이전 절에서 살펴보았던 최솟값 구하는 공식
16     for (const item of items) {
17         if (output > item) {
18             output = item;
19         }
20     }
21     return output;
22 }
23
24 console.log(`min(배열): ${min([52, 273, 32, 103, 275, 24, 57])}`);
25 console.log(`min(숫자, ...): ${min(52, 273, 32, 103, 275, 24, 57)}`);
```

[Running] node "/Users/seeker/javascript/study.js"
min(배열): 24
min(숫자, ...): 24
[Done] exited with code=0 in 0.087 seconds

문서 출력 디버그 콘솔 터미널 포트 Code

출 26, 열 1 공백: 4 UTF-8 LF {} JavaScript 포트: 3000 Prettier

Part 5

전개 연산자

```
study.js > ...
1 // 단순하게 매개변수를 모두 출력하는 함수
2 function sample(...items) {
3     console.log(items);
4 }
5
6 // 전개 연산자 사용 여부 비교하기
7 const array = [1, 2, 3, 4];
8
9 console.log('# 전개 연산자를 사용하지 않은 경우');
10 sample(array);
11 console.log('# 전개 연산자를 사용한 경우');
12 sample(...array);
13
```

[Running] node "/Users/seeker/javascript/study.js"

전개 연산자를 사용하지 않은 경우
[[1, 2, 3, 4]]

전개 연산자를 사용한 경우
[1, 2, 3, 4]

[Done] exited with code=0 in 0.1 seconds

줄 13, 열 1 공백: 4 UTF-8 LF {} JavaScript 포트: 3000 ✓ Prettier

```
study.js
1  function earnings(name, wage = 8590, hours = 40) {
2    console.log(`# ${name} 님의 급여 정보`);
3    console.log(`- 시급: ${wage}원`);
4    console.log(`- 근무 시간: ${hours}시간`);
5    console.log(`- 급여: ${wage * hours}원`);
6    console.log('');
7  }
8
9 // 최저 임금으로 최대한 일하는 경우
10 earnings('구름');
11
12 // 시급 1만원으로 최대한 일하는 경우
13 earnings('별', 10000);
14
15 // 시급 1만원으로 52시간 일한 경우
16 earnings('인성', 10000, 52);
17
```

[Running] node "/Users/seeker/javascript/study.js"

구름 님의 급여 정보
- 시급: 8590원
- 근무 시간: 40시간
- 급여: 343600원

별 님의 급여 정보
- 시급: 10000원
- 근무 시간: 40시간
- 급여: 400000원

인성 님의 급여 정보
- 시급: 10000원
- 근무 시간: 52시간
- 급여: 520000원

[Done] exited with code=0 in 0.417 seconds

Part 5

기본 매개변수

The screenshot shows a dark-themed code editor interface. On the left is a sidebar with various icons for file operations. The main area has a search bar at the top labeled "javascript". A tab labeled "study.js" is active. The code in the editor is:

```
study.js > ...
1  function isLeapYear(year = new Date().getFullYear()) {
2      console.log(`매개변수 year: ${year}`);
3      return (year % 4 === 0 && year % 100 !== 0) || year % 400 === 0;
4  }
5
6  console.log(`올해는 윤년일까? === ${isLeapYear()}`);
7
```

To the right of the code editor is a terminal window titled "Output" which shows the execution of the script:

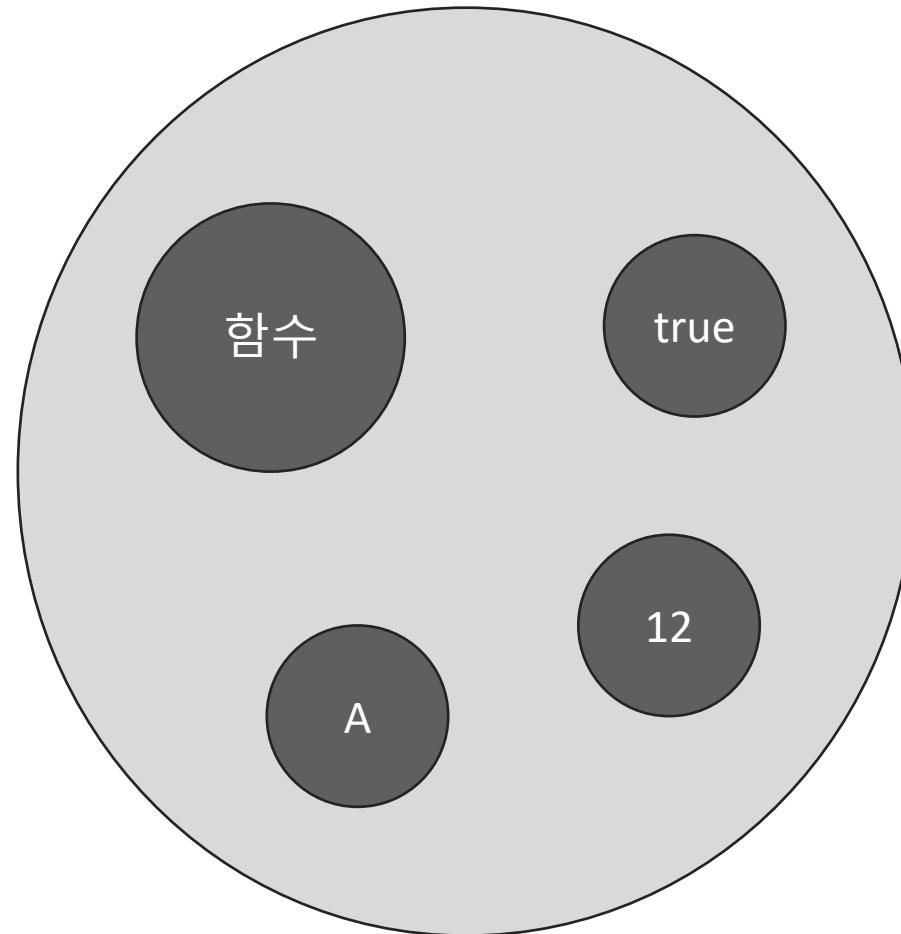
```
[Running] node "/Users/seeker/javascript/study.js"
매개변수 year: 2024
올해는 윤년일까? === true

[Done] exited with code=0 in 0.097 seconds
```

At the bottom of the terminal window, there are status indicators: 콜 7, 열 1, 공백: 4, LF, JavaScript, 포트: 3000, Prettier.

A close-up photograph of a person's hands holding a white smartphone. The person is wearing a white button-down shirt. A large, semi-transparent white rectangular box is overlaid on the center-left portion of the image. Inside this box, the Korean text "함수 고급" is displayed in a bold, black, sans-serif font.

함수 고급



매개변수

Part 5

콜백 함수

```
study.js > ...
1 // 함수를 선언합니다.
2 function callThreeTimes(callback) {
3     for (let i = 0; i < 3; i++) {
4         callback(i);
5     }
6 }
7
8 function print(i) {
9     console.log(`#${i}번째 함수 호출`);
10}
11
12 // 함수를 호출합니다.
13 callThreeTimes(print);
14
```

[Running] node "/Users/seeker/javascript/study.js"

0번째 함수 호출
1번째 함수 호출
2번째 함수 호출

[Done] exited with code=0 in 0.143 seconds

줄 14, 열 1 공백: 4 UTF-8 LF {} JavaScript 포트: 3000 ✓ Prettier

Part 5

콜백 함수

```
study.js
1 // 함수를 선언합니다.
2 function callThreeTimes(callback) {
3     for (let i = 0; i < 3; i++) {
4         callback(i);
5     }
6 }
7
8 // 함수를 호출합니다.
9 callThreeTimes(function (i) {
10     console.log(`#${i}번째 함수 호출`);
11 });
12
```

[Running] node "/Users/seeker/javascript/study.js"

0번째 함수 호출
1번째 함수 호출
2번째 함수 호출

[Done] exited with code=0 in 0.088 seconds

줄 12, 열 1 공백: 4 UTF-8 LF {} JavaScript 포트: 3000 Prettier

Part 5

콜백 함수 활용 > forEach()

The screenshot shows a terminal window with a dark theme. On the left is the code editor pane containing a file named 'study.js'. The code defines an array 'numbers' and uses its 'forEach' method to log each element to the console with a custom message.

```
study.js
1 const numbers = [273, 52, 103, 32, 57];
2
3 numbers.forEach(function (value, index, array) {
4     console.log(` ${index}번째 요소 : ${value}`);
5 });
6
```

The right pane shows the terminal output. It starts with '[Running] node "/Users/seeker/javascript/study.js"', followed by five lines of output: '0번째 요소 : 273', '1번째 요소 : 52', '2번째 요소 : 103', '3번째 요소 : 32', and '4번째 요소 : 57'. Finally, it ends with '[Done] exited with code=0 in 0.086 seconds'.

At the bottom of the terminal window, there are status indicators: a blue progress bar, a refresh icon, and text indicating the current file is 'study.js', has 6 columns and 1 row, is in UTF-8 encoding, is a JavaScript file, and is connected to port 3000 via Prettier.

Part 5

콜백 함수 활용 > map()

```
study.js
1 // 배열을 선언합니다.
2 let numbers = [273, 52, 103, 32, 57];
3
4 // 배열의 모든 값을 제곱합니다.
5 numbers = numbers.map(function (value, index, array) {
6   return value * value;
7 });
8
9 // 출력합니다.
10 numbers.forEach((value, index, array) => {
11   console.log(` ${value} ${index} ${array}`);
12 });
13
14 // 출력합니다.
15 numbers.forEach(console.log);
16
```

[Running] node "/Users/seeker/Edu_Project/동호/최종/javascript/study.js"
74529 0 74529,2704,10609,1024,3249
2704 1 74529,2704,10609,1024,3249
10609 2 74529,2704,10609,1024,3249
1024 3 74529,2704,10609,1024,3249
3249 4 74529,2704,10609,1024,3249
74529 0 [74529, 2704, 10609, 1024, 3249]
2704 1 [74529, 2704, 10609, 1024, 3249]
10609 2 [74529, 2704, 10609, 1024, 3249]
1024 3 [74529, 2704, 10609, 1024, 3249]
3249 4 [74529, 2704, 10609, 1024, 3249]
[Done] exited with code=0 in 0.092 seconds

문서 출력 디버그 콘솔 터미널 포트 Code

줄 16, 열 1 공백: 4 UTF-8 LF {} JavaScript 포트: 3000 Prettier

Part 5

콜백 함수 활용 > map()

```
study.js
1 // 배열을 선언합니다.
2 let numbers = [273, 52, 103, 32, 57];
3
4 // 배열의 모든 값을 제곱합니다.
5 numbers = numbers.map(function (value) {
6   return value * value;
7 });
8
9 // 출력합니다.
10 numbers.forEach((value) => {
11   console.log(` ${value} `);
12 });
13
14 // 출력합니다.
15 numbers.forEach(console.log);
16
```

[Running] node "/Users/seeker/Edu_Project/동호/최종/javascript/study.js"

74529
2704
10609
1024
3249
74529 0 [74529, 2704, 10609, 1024, 3249]
2704 1 [74529, 2704, 10609, 1024, 3249]
10609 2 [74529, 2704, 10609, 1024, 3249]
1024 3 [74529, 2704, 10609, 1024, 3249]
3249 4 [74529, 2704, 10609, 1024, 3249]

[Done] exited with code=0 in 0.128 seconds

문서 출력 디버그 콘솔 터미널 포트 Code

줄 16, 열 1 공백: 4 UTF-8 LF () JavaScript 포트: 3000 Prettier

Part 5

콜백 함수 활용 > filter()

The screenshot shows a terminal window with the following details:

- Title Bar:** javascript
- File:** study.js
- Code Content:**

```
1 const numbers = [0, 1, 2, 3, 4, 5];
2 const evenNumbers = numbers.filter(function (value) {
3     return value % 2 === 0;
4 });
5
6 console.log(`원래 배열: ${numbers}`);
7 console.log(`짝수만 추출: ${evenNumbers}`);
8
```
- Output:**
 - [Running] node "/Users/seeker/javascript/study.js"
 - 원래 배열: 0,1,2,3,4,5
 - 짝수만 추출: 0,2,4
 - [Done] exited with code=0 in 0.094 seconds
- Bottom Status Bar:** 줄 8, 열 1 | 공백: 4 | LF | JavaScript | 포트: 3000 | Prettier

Part 5

화살표 함수

The screenshot shows a dark-themed code editor interface. On the left is the file tree with a single file named 'study.js'. The main area contains the following code:

```
study.js > ...
1 // 배열을 선언합니다.
2 let numbers = [0, 1, 2, 3, 4, 5, 6, 7, 8, 9];
3
4 // 배열의 메소드를 연속적으로 사용합니다.
5 numbers
6   .filter((value) => value % 2 === 0)
7   .map((value) => value * value)
8   .forEach((value) => {
9     console.log(value);
10  });
11
```

The output panel on the right shows the results of the execution:

[Running] node "/Users/seeker/javascript/study.js"

0
4
16
36
64

[Done] exited with code=0 in 0.085 seconds

At the bottom, the status bar shows: 줄 11, 열 1 공백: 4 UTF-8 LF () JavaScript 포트: 3000 Prettier

Part 5

타이머 함수

The screenshot shows a dark-themed code editor window with a sidebar on the left containing icons for file operations, search, and settings. The main area has tabs for 'study.js' and 'javascript'. The code editor displays the following JavaScript code:

```
study.js > ...
1  setTimeout(() => {
2      console.log(`1초 후에 실행됩니다.`);
3  }, 1 * 1000);
4
5  let count = 0;
6  setInterval(() => {
7      console.log(`1초마다 실행됩니다.(${count}번째)`);
8      count++;
9  }, 1 * 1000);
10
```

The output pane on the right shows the results of running the script with 'node' in the terminal:

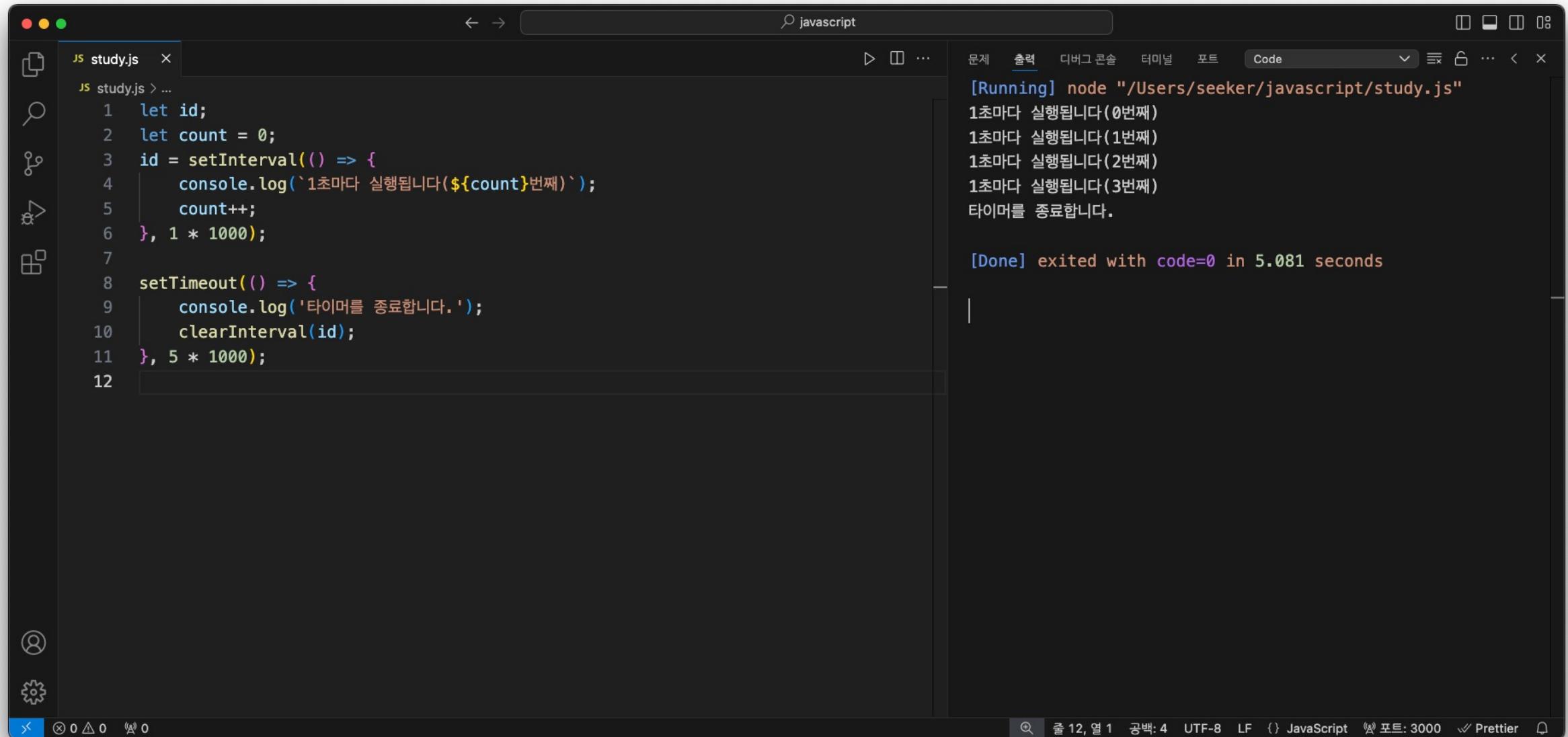
[Running] node "/Users/seeker/javascript/study.js"

- 1초 후에 실행됩니다.
- 1초마다 실행됩니다. (0번째)
- 1초마다 실행됩니다. (1번째)
- 1초마다 실행됩니다. (2번째)
- 1초마다 실행됩니다. (3번째)
- 1초마다 실행됩니다. (4번째)
- 1초마다 실행됩니다. (5번째)
- 1초마다 실행됩니다. (6번째)
- 1초마다 실행됩니다. (7번째)
- 1초마다 실행됩니다. (8번째)
- 1초마다 실행됩니다. (9번째)
- 1초마다 실행됩니다. (10번째)
- 1초마다 실행됩니다. (11번째)
- 1초마다 실행됩니다. (12번째)
- 1초마다 실행됩니다. (13번째)
- 1초마다 실행됩니다. (14번째)
- 1초마다 실행됩니다. (15번째)
- 1초마다 실행됩니다. (16번째)
- 1초마다 실행됩니다. (17번째)
- 1초마다 실행됩니다. (18번째)

At the bottom, there are status indicators for tabs, file count, and terminal output, along with file navigation and search tools.

Part 5

타이머 함수



```
study.js
1 let id;
2 let count = 0;
3 id = setInterval(() => {
4     console.log(`1초마다 실행됩니다(${count}번째)`);
5     count++;
6 }, 1 * 1000);
7
8 setTimeout(() => {
9     console.log('타이머를 종료합니다.');
10    clearInterval(id);
11 }, 5 * 1000);
12
```

[Running] node "/Users/seeker/javascript/study.js"

1초마다 실행됩니다(0번째)
1초마다 실행됩니다(1번째)
1초마다 실행됩니다(2번째)
1초마다 실행됩니다(3번째)
타이머를 종료합니다.

[Done] exited with code=0 in 5.081 seconds

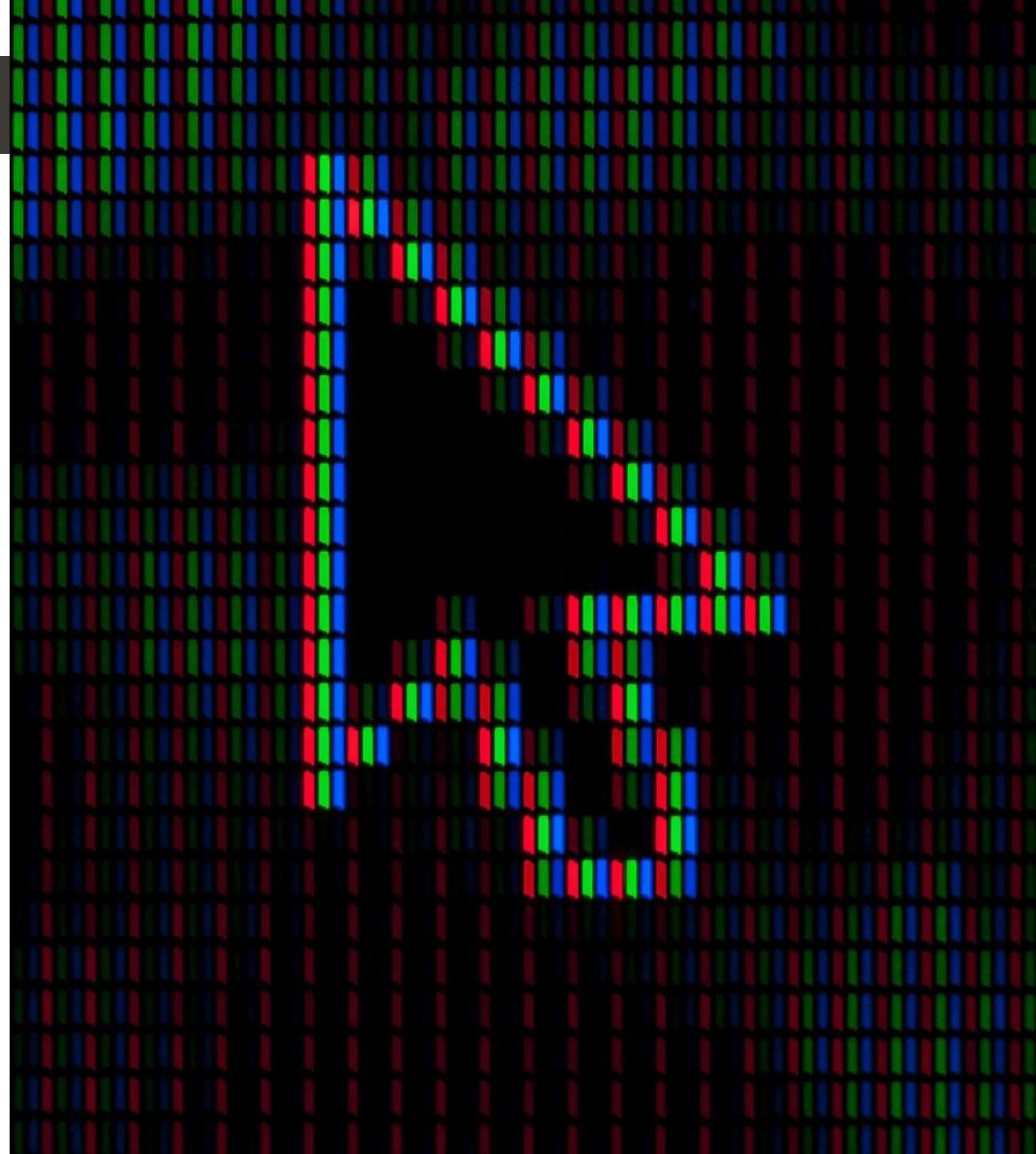
줄 12, 열 1 공백: 4 LF {} JavaScript 포트: 3000 Prettier

6 **객체**

7 문서 객체 모델

8 예외 처리

9 클래스



A close-up photograph of a person's hands holding a white smartphone. The person is wearing a white button-down shirt. A large, semi-transparent white rectangular box is overlaid on the center-left portion of the image. Inside this box, the Korean text "객체 기본" is displayed in a bold, black, sans-serif font.

객체 기본



이름 : 구름
나이 : 7살

속성

산책하기()
밥먹기()

메소드

The screenshot shows a browser developer tools interface with the 'Console' tab selected. The left sidebar lists files: 'study.html X' and 'study.html'. The main area displays the following JavaScript interaction:

```
> typeof([])
< 'object'
> const array = ['사과', '바나나', '망고', '딸기']
< undefined
> array
< - ▼ (4) ['사과', '바나나', '망고', '딸기'] 1
    0: "사과"
    1: "바나나"
    2: "망고"
    3: "딸기"
    length: 4
  ▶ [[Prototype]]: Array(0)
> |
```

The code demonstrates the creation of a new array named 'array' containing four Korean fruit names: '사과', '바나나', '망고', and '딸기'. The 'length' property is shown as 4. The array's prototype is also displayed.

The screenshot shows a browser developer tools window with the 'Console' tab selected. The search bar at the top contains the text 'javascript'. The left sidebar shows an open file named 'study.html' with one line of code: '1'. The main area displays the execution of a JavaScript object definition and its resulting state.

```
> const product = {  
  제품명: '7D 건조 망고', //키와 값 뒤에 쉼표(,)를 넣어 구분  
  유형: '당절임',  
  성분: '망고, 설탕, 메타중아화산나트륨, 치자황색소',  
  원산지: '필리핀'  
}  
< undefined  
> product  
< {제품명: '7D 건조 망고', 유형: '당절임', 성분: '망고, 설탕, 메타중아화산나트륨, 치자황색소', 원  
  산지: '필리핀'} i  
  성분: "망고, 설탕, 메타중아화산나트륨, 치자황색소"  
  원산지: "필리핀"  
  유형: "당절임"  
  제품명: "7D 건조 망고"  
> [[Prototype]]: Object  
> |
```

The console output shows the creation of an object 'product' with properties: 제품명 ('7D 건조 망고'), 유형 ('당절임'), 성분 ('망고, 설탕, 메타중아화산나트륨, 치자황색소'), and 원산지 ('필리핀'). The object is then expanded to show its internal structure, including its prototype as 'Object'.

The screenshot shows a browser's developer tools open to the 'Console' tab. A search bar at the top contains the text 'javascript'. The left sidebar has icons for file, search, and settings, with 'study.html' selected. The main area displays the following JavaScript code and its object structure:

```
> const product = {  
  제품명: '7D 건조 망고', //키와 값 뒤에 쉼표(,)를 넣어 구분  
  유형: '당절임',  
  성분: '망고, 설탕, 메타중아화산나트륨, 치자황색소',  
  원산지: '필리핀'  
}  
< undefined  
> product  
< -> {제품명: '7D 건조 망고', 유형: '당절임', 성분: '망고, 설탕, 메타중아화산나트륨, 치자황색소',  
원산지: '필리핀'}  
> product['제품명']  
< -> '7D 건조 망고'  
> product['유형']  
< -> '당절임'  
> product['성분']  
< -> '망고, 설탕, 메타중아화산나트륨, 치자황색소'  
> product['원산지']  
< -> '필리핀'  
> product.제품명  
< -> '7D 건조 망고'  
> product.유형  
< -> '당절임'  
> product.성분  
< -> '망고, 설탕, 메타중아화산나트륨, 치자황색소'  
> product.원산지  
< -> '필리핀'
```

The screenshot shows a browser developer tools interface with the 'Console' tab selected. On the left, there's a sidebar with various icons and a file tree showing 'study.html' is open. The main area displays the following JavaScript code and its execution results:

```
> const object = {
    number: 273,
    string: '구름',
    boolean: true,
    array: [52, 273, 103, 32],
    method: function () {}
}
< undefined
> object
< > {number: 273, string: '구름', boolean: true, array: Array(4), method: f}
> const pet = {
    name: '구름',
    eat: function (food) {}
}
< undefined
> pet
< > {name: '구름', eat: f} i
    > eat: f (food)
        name: "구름"
    > [[Prototype]]: Object
> pet.eat()
< undefined
> |
```

The code defines two objects: 'object' and 'pet'. The 'object' has properties like number, string, boolean, array, and a method. The 'pet' object has a name and an eat method. The 'eat' method is shown as a function that takes a parameter 'food'. The final line shows the result of calling the 'eat' method on the 'pet' object.

```
study.js
1 // 변수를 선언합니다.
2 const pet = {
3   name: '구름',
4   eat: function (food) {
5     console.log(this.name + '은/는 ' + food + '을/를 먹습니다.');
6   },
7 };
8
9 // 메소드를 호출합니다.
10 pet.eat('밥');
11
```

[Running] node "/Users/seeker/javascript/study.js"
구름은/는 밥을/를 먹습니다.
[Done] exited with code=0 in 0.413 seconds

The screenshot shows a dark-themed code editor interface with a sidebar on the left containing icons for file operations, search, and settings. The main area has a title bar with tabs for 'study.js' and 'javascript'. A search bar at the top right contains the text 'javascript'. The code editor displays the following JavaScript code:

```
study.js > ...
1 // 객체를 선언합니다.
2 const student = {};
3 student.이름 = '윤인성';
4 student.취미 = '악기';
5 student.장래희망 = '생명공학자';
6
7 // 출력합니다.
8 console.log(JSON.stringify(student, null, 2));
9
```

The output panel on the right shows the execution results:

```
[Running] node "/Users/seeker/javascript/study.js"
{
  "이름": "윤인성",
  "취미": "악기",
  "장래희망": "생명공학자"
}

[Done] exited with code=0 in 0.097 seconds
```

At the bottom, status indicators show '줄 9, 열 1' and 'JavaScript'.

The screenshot shows a dark-themed code editor interface with a sidebar on the left containing icons for file operations, search, and settings. The main area has a tab bar with 'study.js' selected. The code in 'study.js' is as follows:

```
study.js > ...
1 // 객체를 선언합니다.
2 const student = {};
3 student.이름 = '윤인성';
4 student.취미 = '악기';
5 student.장래희망 = '생명공학자';
6
7 // 객체의 속성을 제거합니다.
8 delete student.장래희망;
9
10 // 출력합니다.
11 console.log(JSON.stringify(student, null, 2));
12
```

The output pane on the right shows the execution results:

```
[Running] node "/Users/seeker/javascript/study.js"
{
  "이름": "윤인성",
  "취미": "악기"
}

[Done] exited with code=0 in 0.095 seconds
```

At the bottom, there are status indicators for tabs, file count, encoding, and a Prettier button.

```
study.js > ...
1 // 객체를 선언합니다.
2 const pet = {
3     name: '구름',
4     eat(food) {
5         console.log(this.name + '은/는 ' + food + '을/를 먹습니다.');
6     },
7 };
8
9 // 메소드를 호출합니다.
10 pet.eat('밥');
11
```

[Running] node "/Users/seeker/javascript/study.js"
구름은/는 밥을/를 먹습니다.

[Done] exited with code=0 in 0.098 seconds

A close-up photograph of a person's hands holding a white smartphone. The person is wearing a white button-down shirt. The background is blurred.

속성과 메소드

The screenshot shows a browser developer tools window with the 'Console' tab selected. The console output is as follows:

```
> const a = []
< undefined
> a.sample = 10
< 10
> a.sample
< 10
> function b () { }
< undefined
> b.sample = 10
< 10
> b.sample
< 10
> typeof a
< 'object'
> Array.isArray(a)
< true
> typeof b
< 'function'
> |
```

The left side of the image shows the browser's interface with tabs for 'study.html' and 'study.html'. The status bar at the bottom indicates: 줄 1, 열 1 | 공백: 4 | UTF-8 | LF | HTML | 포트: 3002 | Prettier.

The screenshot shows a browser's developer tools with the 'Console' tab selected. The left sidebar displays files: 'study.html x' and 'study.html'. The main area shows the following JavaScript interactions:

```
> const c = 273
< undefined
> c.sample = 10
< 10
> c.sample
< undefined
> const d = '안녕하세요'
< undefined
> d.sample = 10
< 10
> d.sample
< undefined
> const e = true
< undefined
> e.sample = 10
< 10
> e.sample
< undefined
> |
```

The console output shows the results of various assignments and property access operations on variables `c`, `d`, and `e`. The variable `c` is assigned the value 273 and has a property `sample` set to 10. The variable `d` is assigned the string '안녕하세요' and also has a property `sample` set to 10. The variable `e` is assigned the boolean value `true` and has a property `sample` set to 10.

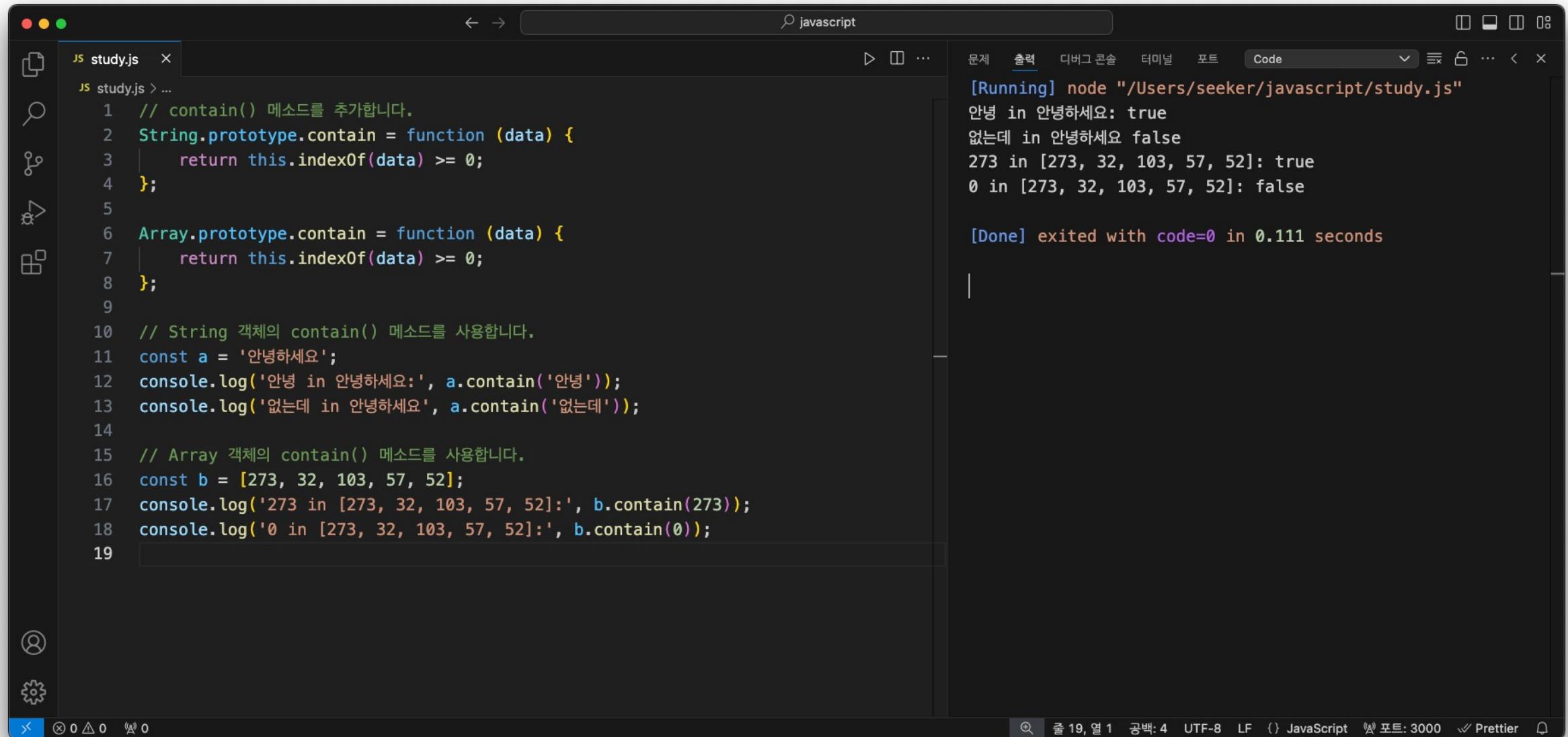
The screenshot shows a browser developer tools interface with the 'Console' tab selected. The left sidebar shows an open file named 'study.html'. The console output is as follows:

```
> const f = new Number(273)
< undefined
> typeof f
< 'object'
> f.sample = 10
< 10
> f.sample
< 10
> f
< Number {273, sample: 10}
> f + 0
< 273
> f.valueOf()
< 273
> |
```

The screenshot shows a dark-themed code editor interface. On the left is the code editor pane with a file named 'study.js'. The code defines a power() method for the Number prototype and then uses it on a Number object 'a'. The right side shows the terminal pane where the script is running and outputting results.

```
study.js > ...
1 // power() 메소드를 추가합니다.
2 Number.prototype.power = function (n = 2) {
3     return this.valueOf() ** n;
4 };
5
6 // Number 객체의 power() 메소드를 사용합니다.
7 const a = 12;
8 console.log('a.power():', a.power());
9 console.log('a.power(3):', a.power(3));
10 console.log('a.power(4):', a.power(4));
11
```

[Running] node "/Users/seeker/javascript/study.js"
a.power(): 144
a.power(3): 1728
a.power(4): 20736
[Done] exited with code=0 in 0.109 seconds



```
study.js
1 // contain() 메소드를 추가합니다.
2 String.prototype.contains = function (data) {
3     return this.indexOf(data) >= 0;
4 };
5
6 Array.prototype.contains = function (data) {
7     return this.indexOf(data) >= 0;
8 };
9
10 // String 객체의 contains() 메소드를 사용합니다.
11 const a = '안녕하세요';
12 console.log('안녕 in 안녕하세요:', a.contains('안녕'));
13 console.log('없는데 in 안녕하세요', a.contains('없는데'));
14
15 // Array 객체의 contains() 메소드를 사용합니다.
16 const b = [273, 32, 103, 57, 52];
17 console.log('273 in [273, 32, 103, 57, 52]:', b.contains(273));
18 console.log('0 in [273, 32, 103, 57, 52]:', b.contains(0));
```

[Running] node "/Users/seeker/javascript/study.js"
안녕 in 안녕하세요: true
없는데 in 안녕하세요 false
273 in [273, 32, 103, 57, 52]: true
0 in [273, 32, 103, 57, 52]: false
[Done] exited with code=0 in 0.111 seconds

The screenshot shows a browser's developer tools open to the 'Console' tab. The left sidebar shows files 'study.html' and 'study.html'. The console output is as follows:

```
> const l = 123.456789
< undefined
> l.toFixed(2)
< '123.46'
> l.toFixed(3)
< '123.457'
> l.toFixed(4)
< '123.4568'
> const m = Number('숫자로 변환할 수 없는 경우')
< undefined
> m
< NaN
> m === NaN
< false
> Number.isNaN(m)
< true
> |
```

The console uses color coding for syntax: purple for keywords like 'const', 'Number', and 'true'; blue for variables like 'l' and 'm'; yellow for method names like 'toFixed' and 'isNaN'; and cyan for strings like '123.456789' and 'NaN'.

String 객체 > trim() : 문자열 양쪽 끝의 공백 없애기

The screenshot shows a browser developer tools console window with the title bar "javascript". The console tab is selected. On the left, there's a sidebar with icons for file, search, and other developer tools. The main area shows the following JavaScript interaction:

```
> const stringA = `메세지를 입력하다보니 앞에 줄 바꿈도 들어가고`;
< undefined
> const stringB = `    앞과 뒤에 공백도 들어가고    `;
< undefined
> stringA
< ' 메세지를 입력하다보니 앞에 줄 바꿈도 들어가고 '
> stringB
< '    앞과 뒤에 공백도 들어가고    '
> stringA.trim()
< '메세지를 입력하다보니 앞에 줄 바꿈도 들어가고'
> stringB.trim()
< '앞과 뒤에 공백도 들어가고'
>
```

The code demonstrates the use of the `trim()` method on two strings. `stringA` contains a newline character at the start, which is removed by `trim()`. `stringB` contains leading and trailing spaces, which are both removed by `trim()`.

String 객체 > split() : 문자열을 특정 기호로 자르기

The screenshot shows a browser developer tools console window with the title bar "javascript". The console tab is selected. The code in the console is as follows:

```
> let input = `일자, 달러, 엔, 유로
02,1141.8, 1097.46,1262.37
03,1148.7,1111.36,1274.65
04,1140.6,1107.81, 1266.58
07,1143.4, 1899.58,1267.8
08,1141.6,1091.97,1261.07`  

< undefined  

> input = input.trim()  

< '일자, 달러, 엔, 유로\n02,1141.8, 1097.46,1262.37\n03,1148.7,1111.36,1274.65\n04,1140.6,1107.81, 1266.58\n07,1143.4, 1899.58,1267.8\n08,1141.6,1091.97,1261.07'  

> input = input.split('\n')  

< (6) ['일자', '달러', '엔', '유로', '02,1141.8, 1097.46,1262.37', '03,1148.7,1111.36,1274.65',  

  > '04,1140.6,1107.81, 1266.58', '07,1143.4, 1899.58,1267.8', '08,1141.6,1091.97,1261.07']  

> input = input.map((line) => line.split(','))  

< ▶ (6) [Array(4), Array(4), Array(4), Array(4), Array(4), Array(4)]  

> JSON.stringify(input, null, 2)  

< '[\n  [\n    ["일자",\n      "달러",\n      "엔",\n      "유로"\n    ],\n    [\n      "02",\n      "1141.8",\n      "1097.46",\n      "1262.37"\n    ],\n    [\n      "03",\n      "1148.7",\n      "1111.36",\n      "1274.65"\n    ],\n    [\n      "04",\n      "1140.6",\n      "1107.81",\n      "1266.58"\n    ],\n    [\n      "07",\n      "1143.4",\n      "1899.58",\n      "1267.8"\n    ],\n    [\n      "08",\n      "1141.6",\n      "1091.97",\n      "1261.07"\n    ]\n]'  

>
```

The code demonstrates how to split a multi-line string into an array of arrays, where each inner array contains four elements corresponding to the columns in the original string.

The screenshot shows a terminal window with the following details:

- Title Bar:** javascript
- File:** study.js
- Code Content:**

```
1 // 자료를 생성합니다.
2 const data = [
3   {
4     name: '혼자 공부하는 파이썬',
5     price: 18000,
6     publisher: '한빛미디어',
7   },
8   {
9     name: 'HTML5 웹 프로그래밍 입문',
10    price: 26000,
11    publisher: '한빛아카데미',
12  },
13];
14
15 // 자료를 JSON으로 변환합니다.
16 console.log(JSON.stringify(data));
17 console.log(JSON.stringify(data, null, 2));
18
```

- Output:**

```
[{"name": "혼자 공부하는 파이썬", "price": 18000, "publisher": "한빛미디어"}, {"name": "HTML5 웹 프로그래밍 입문", "price": 26000, "publisher": "한빛아카데미"}]
```

```
[{"name": "혼자 공부하는 파이썬", "price": 18000, "publisher": "한빛미디어"}, {"name": "HTML5 웹 프로그래밍 입문", "price": 26000, "publisher": "한빛아카데미"}]
```

```
[Done] exited with code=0 in 0.096 seconds
```

- Bottom Status Bar:** 줄 18, 열 1 | 공백: 4 | UTF-8 | LF | JavaScript | 포트: 3000 | Prettier

The screenshot shows a dark-themed code editor interface with a sidebar on the left containing icons for file operations, search, and settings. The main area has a title bar with tabs for 'study.js' and 'javascript'. A search bar at the top right contains the text 'javascript'. The code editor displays the following JavaScript code:

```
study.js > ...
1 // 자료를 생성합니다.
2 const data = [
3   {
4     name: '혼자 공부하는 파이썬',
5     price: 18000,
6     publisher: '한빛미디어',
7   },
8   {
9     name: 'HTML5 웹 프로그래밍 입문',
10    price: 26000,
11    publisher: '한빛아카데미',
12  },
13];
14
15 // 자료를 JSON으로 변환합니다.
16 const json = JSON.stringify(data);
17 console.log(json);
18
19 // JSON 문자열을 다시 자바스크립트 객체로 변환합니다.
20 console.log(JSON.parse(json));
21
```

The output pane on the right shows the execution results:

```
[Running] node "/Users/seeker/javascript/study.js"
[{"name":"혼자 공부하는 파이썬","price":18000,"publisher":"한빛미디어"}, {"name":"HTML5 웹 프로그래밍 입문","price":26000,"publisher":"한빛아카데미"}]
[{"name": "혼자 공부하는 파이썬", "price": 18000, "publisher": "한빛미디어"}, {"name": "HTML5 웹 프로그래밍 입문", "price": 26000, "publisher": "한빛아카데미"}]

[Done] exited with code=0 in 0.105 seconds
```

The screenshot shows a terminal window with the following details:

- Title Bar:** javascript
- File:** study.js
- Code Content:**

```
1 const num = Math.random();
2
3 console.log('# 랜덤한 숫자');
4 console.log('0~1 사이의 랜덤한 숫자:', num);
5 console.log('');
6
7 console.log('# 랜덤한 숫자 범위 확대');
8 console.log('0~10 사이의 랜덤한 숫자:', num * 10);
9 console.log('0~50 사이의 랜덤한 숫자:', num * 50);
10 console.log('');
11
12 console.log('# 랜덤한 숫자 범위 이동');
13 console.log('-5~5 사이의 랜덤한 숫자:', num * 10 - 5);
14 console.log('-25~25 사이의 랜덤한 숫자:', num * 50 - 25);
15 console.log('');
16
17 console.log('# 랜덤한 정수 숫자');
18 console.log('-5~5 사이의 랜덤한 정수 숫자:', Math.floor(num * 10 - 5));
19 console.log('-25~25 사이의 랜덤한 정수 숫자:', Math.floor(num * 50 - 25));
```

- Output:**

```
[Running] node "/Users/seeker/javascript/study.js"
# 랜덤한 숫자
0~1 사이의 랜덤한 숫자: 0.3835903638976461

# 랜덤한 숫자 범위 확대
0~10 사이의 랜덤한 숫자: 3.835903638976461
0~50 사이의 랜덤한 숫자: 19.179518194882306

# 랜덤한 숫자 범위 이동
-5~5 사이의 랜덤한 숫자: -1.1640963610235389
-25~25 사이의 랜덤한 숫자: -5.820481805117694

# 랜덤한 정수 숫자
-5~5 사이의 랜덤한 정수 숫자: -2
-25~25 사이의 랜덤한 정수 숫자: -6

[Done] exited with code=0 in 0.102 seconds
```

- Terminal Status Bar:** 출 15, 열 17 공백: 4 UTF-8 LF () JavaScript 포트: 3000 Prettier

A close-up photograph of a person's hands holding a white smartphone. The person is wearing a white button-down shirt. The background is blurred.

객체/배열 고급

The screenshot shows a dark-themed code editor interface with a sidebar on the left containing icons for file operations, search, and settings. The main area has a title bar with tabs for 'study.js' and 'javascript'. The status bar at the bottom displays file statistics like '줄 19, 열 1' and encoding information.

```
study.js > ...
1 // 객체를 생성합니다.
2 const object = {
3     name: '혼자 공부하는 파이썬',
4     price: 18000,
5     publisher: '한빛미디어',
6 };
7
8 // 객체 내부에 속성이 있는지 확인합니다.
9 if (object.name !== undefined) {
10     console.log('name 속성이 있습니다.');
11 } else {
12     console.log('name 속성이 없습니다.');
13 }
14 if (object.author !== undefined) {
15     console.log('author 속성이 있습니다.');
16 } else {
17     console.log('author 속성이 없습니다.');
18 }
19
```

The output panel on the right shows the execution results:

- [Running] node "/Users/seeker/javascript/study.js"
- name 속성이 있습니다.
- author 속성이 없습니다.
- [Done] exited with code=0 in 0.095 seconds

```
study.js
1 // 객체를 생성합니다.
2 const object = {
3     name: '혼자 공부하는 파이썬',
4     price: 18000,
5     publisher: '한빛미디어',
6 };
7
8 // 객체의 기본 속성을 지정합니다.
9 object.name = object.name !== undefined ? object.name : '제목 미정';
10 object.author = object.author !== undefined ? object.author : '저자 미상';
11
12 // 객체를 출력합니다.
13 console.log(JSON.stringify(object, null, 2));
14
```

[Running] node "/Users/seeker/javascript/study.js"

```
{
  "name": "혼자 공부하는 파이썬",
  "price": 18000,
  "publisher": "한빛미디어",
  "author": "저자 미상"
}
```

[Done] exited with code=0 in 0.093 seconds

줄 14, 열 1 공백: 4 UTF-8 LF {} JavaScript 포트: 3000 Prettier

The screenshot shows a browser developer tools console window with the title bar "javascript". The left sidebar lists files: "study.html X" and "study.html". The main area is the "Console" tab, which contains the following JavaScript code and output:

```
> let [a, b] = [1, 2] //a=1, b=2가 할당
< undefined
> console.log(a, b)
1 2
VM26:1
< undefined
> [a, b] = [b, a] //a에 b가 할당되고, b에 a가 할당되므로 값이 서로 교환
< ▶ (2) [2, 1]
> console.log(a, b)
2 1
VM166:1
< undefined
> let arrayA = [1, 2, 3, 4, 5]
< undefined
> const [c, d, e] = arrayA
< undefined
> console.log(c, d, e)
1 2 3
VM241:1
< undefined
>
```

The console output shows the results of various array assignments and log statements. The first example demonstrates a shallow copy where both variables a and b are updated. The second example shows a swap between arrays. The third example uses a const declaration to show that changes to the array do not affect the const variable.

The screenshot shows a dark-themed code editor interface with a sidebar on the left containing icons for file operations, search, and settings. The main area has a title bar with tabs for 'study.js' and 'javascript'. The status bar at the bottom displays file information like '줄 17, 열 1' and encoding 'UTF-8'.

The code in 'study.js' is as follows:

```
study.js > ...
1 // 객체를 생성합니다.
2 const object = {
3     title: '혼자 공부하는 파이썬',
4     price: 18000,
5     publisher: '한빛미디어',
6 };
7
8 // 객체에서 변수를 추출합니다.
9 const { title, price } = object;
10 console.log('# 속성 이름 그대로 꺼내서 출력하기');
11 console.log(title, price);
12 console.log('');
13
14 const { a = title, b = price } = object;
15 console.log('# 다른 이름으로 속성 꺼내서 출력하기');
16 console.log(a, b);
17
```

The output window on the right shows the execution results:

```
[Running] node "/Users/seeker/javascript/study.js"
# 속성 이름 그대로 꺼내서 출력하기
혼자 공부하는 파이썬 18000

# 다른 이름으로 속성 꺼내서 출력하기
혼자 공부하는 파이썬 18000

[Done] exited with code=0 in 0.088 seconds
```

The screenshot shows a dark-themed code editor interface with a sidebar on the left containing icons for file operations, search, and settings. The main area has tabs for 'study.js' and 'javascript'. The code editor displays the following JavaScript code:

```
study.js > ...
1 // 사야 하는 물건 목록
2 const 물건_200301 = ['우유', '식빵'];
3 const 물건_200302 = 물건_200301;
4 물건_200302.push('고구마');
5 물건_200302.push('토마토');
6
7 // 출력
8 console.log(물건_200301);
9 console.log(물건_200302);
10
```

The right side of the interface shows the execution results in a terminal-like window:

```
[Running] node "/Users/seeker/javascript/study.js"
[ '우유', '식빵', '고구마', '토마토' ]
[ '우유', '식빵', '고구마', '토마토' ]

[Done] exited with code=0 in 0.42 seconds
```

At the bottom, there are status indicators and a toolbar with icons for file operations, search, and code analysis.

The screenshot shows a dark-themed code editor interface with a sidebar on the left containing icons for file operations, search, and settings. The main area has tabs for 'study.js' and 'javascript'. The code editor displays the following JavaScript code:

```
study.js > ...
1 // 사야 하는 물건 목록
2 const 물건_200301 = ['우유', '식빵'];
3 const 물건_200302 = [...물건_200301];
4 물건_200302.push('고구마');
5 물건_200302.push('토마토');
6
7 // 출력
8 console.log(물건_200301);
9 console.log(물건_200302);
10
```

The output panel on the right shows the execution results:

```
[Running] node "/Users/seeker/javascript/study.js"
[ '우유', '식빵' ]
[ '우유', '식빵', '고구마', '토마토' ]

[Done] exited with code=0 in 0.114 seconds
```

At the bottom, there are status indicators for file changes, a search bar, and other system information.

The screenshot shows a dark-themed code editor interface with a sidebar on the left containing icons for file operations, search, and settings. The main area has tabs for 'study.js' and 'javascript'. The code editor displays the following JavaScript code:

```
study.js > ...
1 // 사야 하는 물건 목록
2 const 물건_200301 = ['우유', '식빵'];
3 const 물건_200302 = ['고구마', ...물건_200301, '토마토'];
4
5 // 출력
6 console.log(물건_200301);
7 console.log(물건_200302);
8
```

The output panel on the right shows the results of running the script with Node.js:

```
[Running] node "/Users/seeker/javascript/study.js"
[ '우유', '식빵' ]
[ '고구마', '우유', '식빵', '토마토' ]

[Done] exited with code=0 in 0.101 seconds
```

At the bottom, status bar indicators show: 쿨 8, 열 1, 공백: 4, UTF-8, LF, JavaScript, 포트: 3000, Prettier.

The screenshot shows a dark-themed code editor interface with a sidebar on the left containing icons for file operations, search, and settings. The main area has tabs for 'study.js' and 'study.js > ...'. The code editor displays the following JavaScript code:

```
study.js
1 const 구름 = {
2   이름: '구름',
3   나이: 6,
4   종족: '강아지',
5 };
6 const 별 = 구름;
7 별.이름 = '별';
8 별.나이 = 1;
9
10 console.log(JSON.stringify(구름));
11 console.log(JSON.stringify(별));
12
```

The output panel on the right shows the results of running the script with node:

```
[Running] node "/Users/seeker/javascript/study.js"
{"이름":"별","나이":1,"종족":"강아지"}
{"이름":"별","나이":1,"종족":"강아지"}

[Done] exited with code=0 in 0.096 seconds
```

At the bottom, status bar elements include: ⌛ 0 △ 0 ⓘ 0, 🔍 줄 12, 열 1, 공백: 4, UTF-8 LF, {}, JavaScript ⓘ 포트: 3000, ✅ Prettier, and a refresh icon.

The screenshot shows a dark-themed code editor interface with a sidebar on the left containing icons for file operations, search, and settings. The main area has tabs for 'study.js' and 'study.js > ...'. The code in 'study.js' is as follows:

```
1 const 구름 = {
2     이름: '구름',
3     나이: 6,
4     종족: '강아지',
5 };
6 const 별 = { ...구름 };
7 별.이름 = '별';
8 별.나이 = 1;
9
10 console.log(JSON.stringify(구름));
11 console.log(JSON.stringify(별));
12
```

The right side of the interface shows the output of running the script with Node.js. It includes the command used, the JSON output for each object, and the completion message.

[Running] node "/Users/seeker/javascript/study.js"
{"이름": "구름", "나이": 6, "종족": "강아지"}
{"이름": "별", "나이": 1, "종족": "강아지"}
[Done] exited with code=0 in 0.096 seconds

At the bottom, there are status indicators for tabs, file type (JavaScript), port (3000), and Prettier.

The screenshot shows a dark-themed code editor interface with a sidebar on the left containing icons for file operations, search, and settings. The main area has a title bar with tabs for 'study.js' and 'javascript'. A search bar at the top right contains the text 'javascript'. The code editor displays the following JavaScript code:

```
study.js > ...
1 const 구름 = {
2     이름: '구름',
3     나이: 6,
4     종족: '강아지',
5 };
6 const 별 = {
7     ...구름,
8     이름: '별',
9     나이: 1,
10    예방접종: true,
11 };
12
13 console.log(JSON.stringify(구름));
14 console.log(JSON.stringify(별));
15
```

The output panel on the right shows the results of running the script with Node.js:

```
[Running] node "/Users/seeker/javascript/study.js"
{"이름":"구름","나이":6,"종족":"강아지"}
{"이름":"별","나이":1,"종족":"강아지","예방접종":true}

[Done] exited with code=0 in 0.103 seconds
```

At the bottom, there are status indicators for tabs, file count, encoding, and a Prettier button.

The screenshot shows a dark-themed code editor interface with a sidebar on the left containing icons for file operations, search, and settings. The main area has a title bar with a back/forward button, a search bar containing "javascript", and a tab labeled "study.js". The code editor displays the following JavaScript code:

```
study.js > ...
1 const 구름 = {
2   이름: '구름',
3   나이: 6,
4   종족: '강아지',
5 };
6 const 별 = {
7   이름: '별',
8   나이: 1,
9   예방접종: true,
10  ...구름,
11 };
12
13 console.log(JSON.stringify(구름));
14 console.log(JSON.stringify(별));
15
```

To the right of the code editor is a terminal window showing the output of running the script with Node.js:

```
[Running] node "/Users/seeker/javascript/study.js"
{"이름":"구름","나이":6,"종족":"강아지"}
{"이름":"구름","나이":6,"예방접종":true,"종족":"강아지"}

[Done] exited with code=0 in 0.097 seconds
```

The bottom of the screen shows the status bar with various icons and text indicating the current file is "study.js", the line and column are "줄 15, 열 1", the encoding is "UTF-8", the language is "JavaScript", the port is "포트: 3000", and Prettier is active.

목차

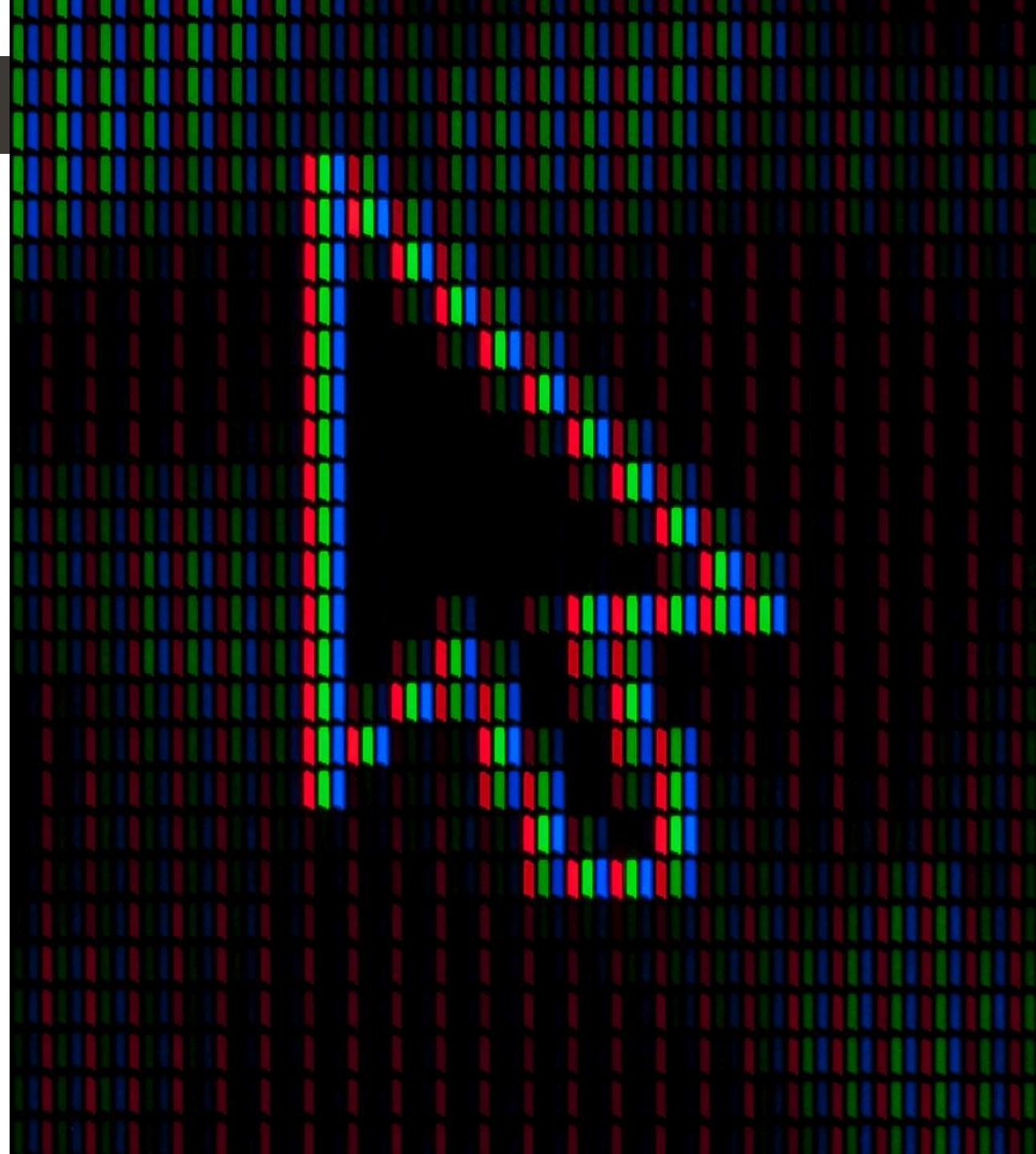
a table of contents

6 객체

7 문서 객체 모델

8 예외 처리

9 클래스



A close-up photograph of a person's hands holding a white smartphone. The person is wearing a white button-down shirt. A large, semi-transparent white rectangular box is overlaid on the center-left portion of the image. Inside this box, the Korean text "DOM조작" is displayed in a bold, black, sans-serif font.

DOM조작

요소



문서 객체

HTML 세계

자바스크립트 세계

The screenshot shows a browser window with developer tools open. On the left, the `study.html` file is displayed in the code editor. The code contains two `<script>` blocks in the head section. The first script creates an `h1` element with the text '1번째 h1 태그'. The second script creates another `h1` element with the text '2번째 h1 태그'. In the body section, there are three more `<script>` blocks. The first adds the text '1번째 script 태그' to the body. The second adds '2번째 script 태그'. The third adds '3번째 script 태그'. The right side of the screen shows the rendered HTML output. It displays the `h1` elements and the `document.body.innerHTML` strings as they appear in the browser.

```
study.html
1 <!DOCTYPE html>
2 <html>
3   <head>
4     <title>DOMContentLoaded</title>
5     <script>
6       // HTML 태그를 쉽게 만들 수 있는 콜백 함수를 선언합니다.
7       const h1 = (text) => `<h1>${text}</h1>`;
8     </script>
9     <script>
10       document.body.innerHTML += h1('1번째 script 태그');
11     </script>
12   </head>
13   <body>
14     <script>
15       document.body.innerHTML += h1('2번째 script 태그');
16     </script>
17     <h1>1번째 h1 태그</h1>
18     <script>
19       document.body.innerHTML += h1('3번째 script 태그');
20     </script>
21     <h1>2번째 h2 태그</h1>
22   </body>
23 </html>
24
```

DOMContentLoaded X

http://127.0.0.1:3000/study.html

2번째 script 태그

1번째 h1 태그

3번째 script 태그

2번째 h2 태그

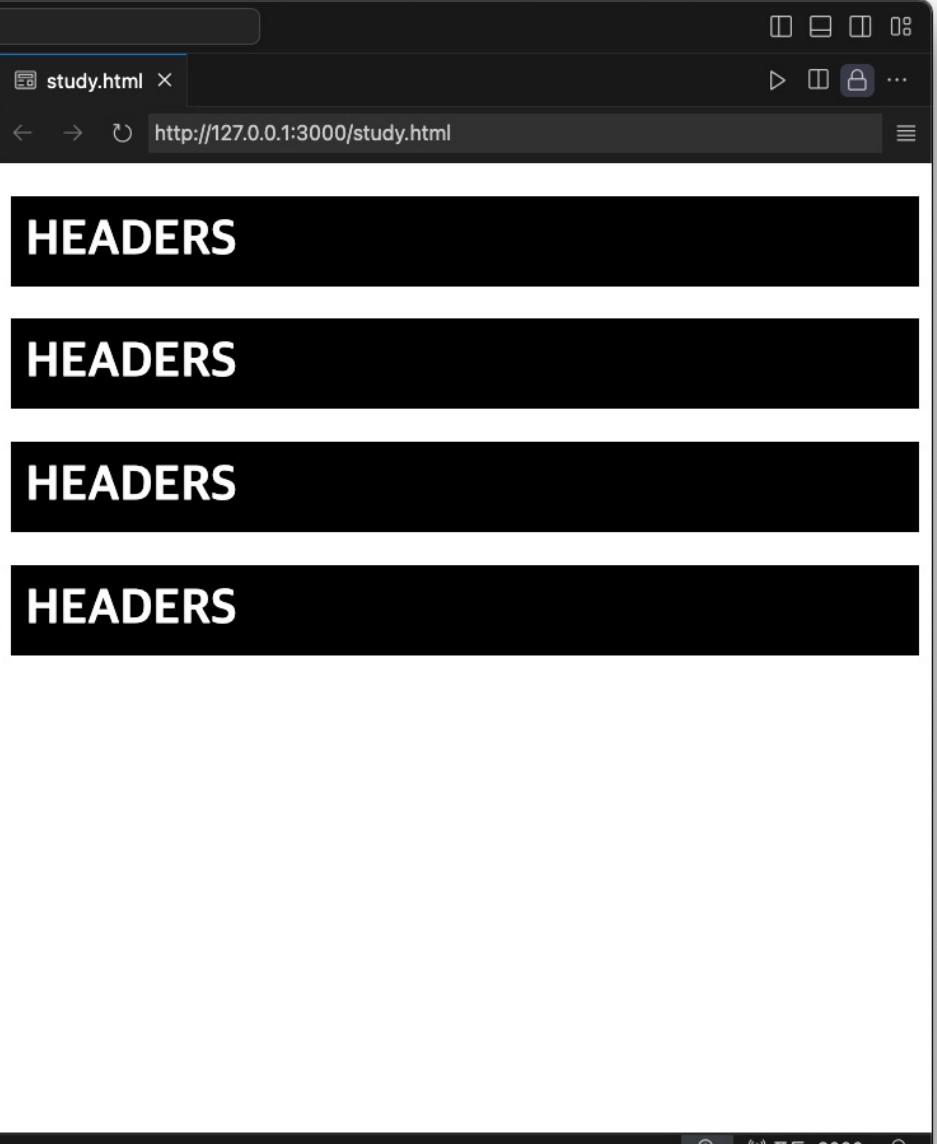
The screenshot shows a browser window with developer tools open. On the left, the code editor displays `study.html` with the following content:

```
1  <!DOCTYPE html>
2  <html>
3      <head>
4          <title>DOMContentLoaded</title>
5          <script>
6              // DOMContentLoaded 이벤트를 연결합니다.
7              document.addEventListener('DOMContentLoaded', () => {
8                  const h1 = (text) => `<h1>${text}</h1>`;
9                  document.body.innerHTML += h1('DOMContentLoaded 이벤트 발생');
10             });
11         </script>
12     </head>
13     <body></body>
14 </html>
15
```

The right pane shows the browser output for the URL `http://127.0.0.1:3000/study.html`. It contains the text **DOMContentLoaded 이벤트 발생**.

The screenshot shows a browser window with developer tools open. On the left, the DOM tree for a file named 'study.html' is displayed. The tree includes the root element <html>, followed by <head> and <body>. Inside the head element, there is a <title> element and a <script> element. The script contains JavaScript code that adds an event listener to the document. When the 'DOMContentLoaded' event fires, it selects the first h1 element and changes its text content to 'HEADERS', sets its color to white, and its background color to black, with a padding of 10px. The right side of the interface shows a preview of the web page at the URL <http://127.0.0.1:3000/study.html>. The preview displays a large, white, bold 'HEADERS' title centered on a black background.

```
1  <!DOCTYPE html>
2  <html>
3      <head>
4          <title></title>
5          <script>
6              document.addEventListener('DOMContentLoaded', () => {
7                  // 요소를 읽어들입니다.,
8                  const header = document.querySelector('h1');
9
10                 // 텍스트와 스타일을 변경합니다.
11                 header.textContent = 'HEADERS';
12                 header.style.color = 'white';
13                 header.style.backgroundColor = 'black';
14                 header.style.padding = '10px';
15             });
16         </script>
17     </head>
18     <body>
19         <h1></h1>
20     </body>
21 </html>
```



The screenshot shows a code editor on the left and a browser window on the right. The code editor displays an HTML file named 'study.html' with the following content:

```
study.html
<!DOCTYPE html>
<html>
  <head>
    <title></title>
    <script>
      document.addEventListener('DOMContentLoaded', () => {
        // 요소를 읽어들입니다.
        const headers = document.querySelectorAll('h1');

        // 텍스트와 스타일을 변경합니다.
        headers.forEach(header) => {
          header.textContent = 'HEADERS';
          header.style.color = 'white';
          header.style.backgroundColor = 'black';
          header.style.padding = '10px';
        });
      });
    </script>
  </head>
  <body>
    <h1></h1>
    <h1></h1>
    <h1></h1>
    <h1></h1>
  </body>
</html>
```

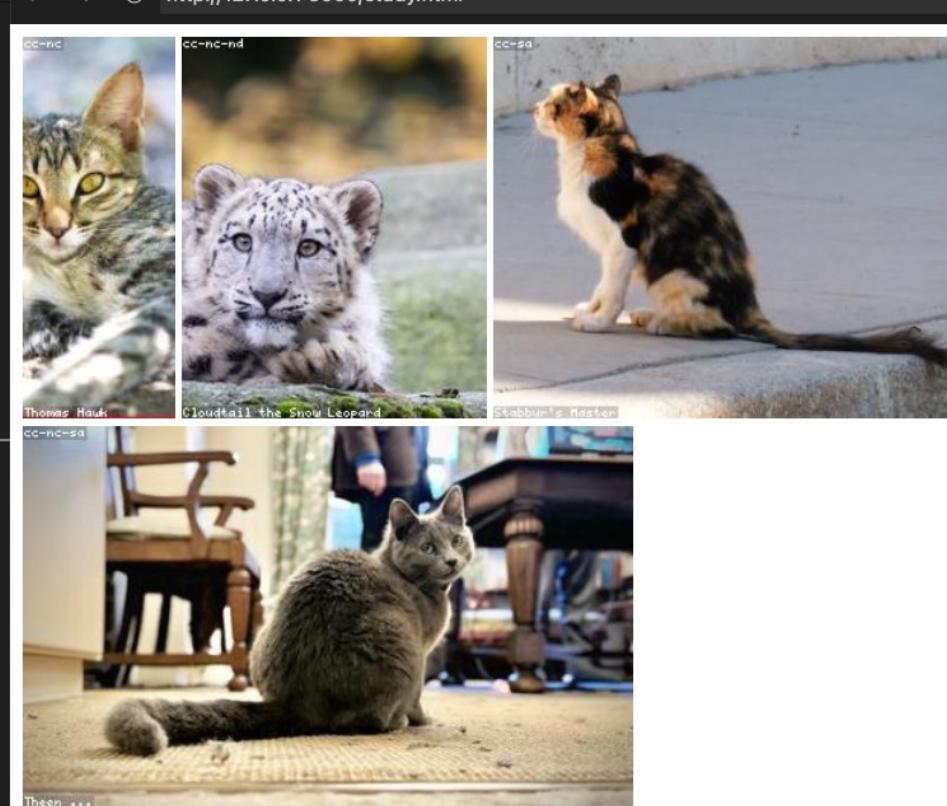
The browser window on the right shows the rendered output of the HTML. It consists of four identical 'HEADERS' headings stacked vertically. Each heading is displayed in white text on a black background with a padding of 10px.

The screenshot shows a browser developer tools window with two main panels. On the left is the DOM tree, titled 'study.html'. It displays the structure of an HTML document with script tags containing JavaScript code. The right panel shows a preview of the page at 'http://127.0.0.1:3000/study.html'. The page content includes an

element with the text 'textContent 속성' and another element with the text 'innerHTML 속성'.

```
<!DOCTYPE html>
<html>
  <head>
    <title></title>
    <script>
      document.addEventListener("DOMContentLoaded", () => {
        const a = document.querySelector("#a");
        const b = document.querySelector("#b");

        a.textContent = "<h1>textContent 속성</h1>";
        b.innerHTML = "<h1>innerHTML 속성</h1>";
      });
    </script>
  </head>
  <body>
    <div id="a"></div>
    <div id="b"></div>
  </body>
</html>
```



The screenshot shows a browser window with two tabs. The left tab is titled "study.html" and contains the following HTML code:

```
<!DOCTYPE html>
<html>
  <head>
    <title></title>
    <script>
      document.addEventListener('DOMContentLoaded', () => {
        const rects = document.querySelectorAll('.rect');

        rects.forEach((rect, index) => {
          const width = (index + 1) * 100;
          const src = `http://loremflickr.com/${width}/250`;
          rect.setAttribute('src', src);
        });
      });
    </script>
  </head>
  <body>
    <img class="rect" />
    <img class="rect" />
    <img class="rect" />
    <img class="rect" />
  </body>
</html>
```

The right tab is also titled "study.html" and displays the rendered HTML with four images. Each image has a small caption below it:

- cc-nc: Thomas Hawk -  Thomas Hawk
- cc-nc-nd: Cloudtail the Snow Leopard -  Cloudtail the Snow Leopard
- cc-sa: Clabbin's Master - 
- cc-wc-sa: Theen ... -  Theen ...

The browser interface includes a search bar at the top labeled "javascript" and a status bar at the bottom with icons and the text "포트: 3000".

The screenshot shows a browser developer tools window with the following components:

- Left Panel:** A sidebar with various icons for file operations, search, and settings.
- Top Bar:** Includes standard window controls (red, yellow, green buttons), a back/forward button, a search bar containing "javascript", and a status bar with icons.
- Left Content Area:** A code editor showing the HTML and JavaScript code for "study.html". The script uses DOM manipulation to create 25

elements with increasing heights and background colors.
- Right Content Area:** A preview panel showing the resulting web page at the URL <http://127.0.0.1:3000/study.html>. The page displays 25 stacked

elements, each with a height of 10px and a background color of `rgba(${val}, ${val}, ${val})`, where `val` increases from 0 to 240.
- Bottom Bar:** A toolbar with icons for zooming, refresh, and other developer tools functions.

The screenshot shows a browser developer tools window with two main panes. The left pane is a script editor titled 'study.html' containing the following code:

```
1  <!DOCTYPE html>
2  <html>
3      <head>
4          <title></title>
5          <script>
6              document.addEventListener('DOMContentLoaded', () => {
7                  // 문서 객체 생성하기
8                  const header = document.createElement('h1');
9
10                 // 생성한 태그 조작하기
11                 header.textContent = '문서 객체 동적으로 생성하기';
12                 header.setAttribute('data-custom', '사용자 정의 속성');
13                 header.style.color = 'white';
14                 header.style.backgroundColor = 'black';
15
16                 // h1 태그를 body 태그 아래에 추가하기
17                 document.body.appendChild(header);
18             });
19         </script>
20     </head>
21     <body></body>
22 </html>
```

The right pane is a preview of the browser window, showing the rendered HTML with a large black header containing the text '문서 객체 동적으로 생성하기'.

The screenshot displays a browser window with two panes. The left pane shows the source code of a file named 'study.html'. The code includes a script block that adds event listeners to the document. It defines two functions, `toFirst` and `toSecond`, which append an `<h1>` element to the first and second `<div>` elements respectively, with a 1-second delay between them. The right pane shows the rendered HTML output at the URL `http://127.0.0.1:3000/study.html`. The page contains two `<div>` elements with IDs `first` and `second`. The first `<div>` contains the text "첫 번째 div 태그 내부" and an `<h1>` element with the text "이동하는 h1 태그". The second `<div>` contains the text "두 번째 div 태그 내부". A horizontal line (`<hr />`) separates the two `<div>` elements.

```
<!DOCTYPE html>
<html>
  <head>
    <title></title>
    <script>
      document.addEventListener('DOMContentLoaded', () => {
        // 문서 객체 읽어들이고 생성하기
        const divA = document.querySelector('#first');
        const divB = document.querySelector('#second');
        const h1 = document.createElement('h1');
        h1.textContent = '이동하는 h1 태그';

        // 서로 번갈아가면서 실행하는 함수를 구현합니다.
        const toFirst = () => {
          divA.appendChild(h1);
          setTimeout(toSecond, 1000);
        };
        const toSecond = () => {
          divB.appendChild(h1);
          setTimeout(toFirst, 1000);
        };
        toFirst();
      });
    </script>
  </head>
  <body>
    <div id="first">
      <h1>첫 번째 div 태그 내부</h1>
    </div>
    <hr />
    <div id="second">
      <h1>두 번째 div 태그 내부</h1>
    </div>
  </body>
</html>
```

The screenshot shows a browser developer tools window with two main panes. The left pane displays the HTML code of a file named 'study.html'. The right pane shows a preview of the page at the URL <http://127.0.0.1:3000/study.html>.

```
<!DOCTYPE html>
<html>
  <head>
    <title></title>
    <script>
      document.addEventListener('DOMContentLoaded', () => {
        setTimeout(() => {
          const h1 = document.querySelector('h1');

          h1.parentNode.removeChild(h1);
          // document.body.removeChild(h1)
        }, 3000);
      });
    </script>
  </head>
  <body>
    <hr />
    <h1>제거 대상 문서 객체</h1>
    <hr />
  </body>
</html>
```

The preview tab on the right shows the resulting page content: "제거 대상 문서 객체". Below the preview, there is another smaller preview window showing a blank white page.

The screenshot shows a browser developer tools window with two main panes. The left pane is a code editor for an HTML file named 'study.html'. The code contains JavaScript and CSS. The right pane is a preview of the browser showing the output of the code.

```
<!DOCTYPE html>
<html>
  <head>
    <title></title>
    <script>
      document.addEventListener('DOMContentLoaded', () => {
        let counter = 0;
        const h1 = document.querySelector('h1');

        h1.addEventListener('click', (event) => {
          counter++;
          h1.textContent = `클릭 횟수:${counter}`;
        });
      });
    </script>
    <style>
      h1 {
        /* 클릭을 여러 번 했을 때
        글자가 선택되는 것을 막기 위한 스타일 */
        user-select: none;
      }
    </style>
  </head>
  <body>
    <h1>클릭 횟수: 0</h1>
  </body>
</html>
```

The preview pane shows the browser window with the title 'study.html' and the URL 'http://127.0.0.1:3000/study.html'. The content of the page is an

element with the text '클릭 횟수: 0'. After clicking the element 12 times, the text changes to '클릭 횟수:12'.

The screenshot shows a development environment with a code editor on the left and a browser window on the right.

Code Editor (study.html):

```
<!DOCTYPE html>
<html>
  <head>
    <title></title>
    <script>
      document.addEventListener('DOMContentLoaded', () => {
        let counter = 0;
        let isConnected = false;

        const h1 = document.querySelector('h1');
        const p = document.querySelector('p');
        const connectButton = document.querySelector('#connect');
        const disconnectButton = document.querySelector('#disconnect');

        const listener = (event) => {
          h1.textContent = `클릭 횟수: ${counter++}`;
        };

        connectButton.addEventListener('click', () => {
          if (isConnected === false) {
            h1.addEventListener('click', listener);
            p.textContent = '이벤트 연결 상태: 연결';
            isConnected = true;
          }
        });
        disconnectButton.addEventListener('click', () => {
          if (isConnected === true) {
            h1.removeEventListener('click', listener);
            p.textContent = '이벤트 연결 상태: 해제';
          }
        });
      });
    </script>
```

Browser Preview:

The browser window displays the output of the script. It shows an

element with the text "클릭 횟수: 0". Below it is a element with the text "이벤트 연결 상태: 해제". At the bottom of the page, there are two buttons: a "connect" button labeled "이벤트 연결" and a "disconnect" button labeled "이벤트 제거". On the right side of the browser window, there is a sidebar with the following content: ``` <style> h1 { /* 클릭을 여러 번 했을 때 글자가 선택되는 것을 막기 위한 스타일 */ user-select: none; } </style> </head> <body> <h1>클릭 횟수: 0</h1> <button id="connect">이벤트 연결</button> <button id="disconnect">이벤트 제거</button> <p>이벤트 연결 상태: 해제</p> </body> </html> ``` The browser status bar at the bottom indicates "포트: 3000".

A close-up photograph of a person's hands holding a white smartphone. The person is wearing a white button-down shirt. A white rectangular overlay box is positioned in the center of the image, containing the Korean text "이벤트 활용".

이벤트 활용

```
<study.html>
<script>
  document.body.addEventListener('keyup', () => {
    console.log(event)
  })
  document.body.onkeyup = (event) => {
    console.log(event)
  }
</script>
```

```
<study.html>
<script>
  const listener = (event) => {
    console.log(event)
  }
</script>
<body onkeyup="listener(event)">
</body>
```

The screenshot shows a browser developer tools interface with two panes. The left pane displays the HTML code for 'study.html' with line numbers. The right pane shows the rendered web page at <http://127.0.0.1:3000/study.html>.

```
1  <!DOCTYPE html>
2  <html>
3      <head>
4          <title></title>
5          <script>
6              document.addEventListener('DOMContentLoaded', () => {
7                  const textarea = document.querySelector('textarea');
8                  const h1 = document.querySelector('h1');
9
10                 textarea.addEventListener('keyup', (event) => {
11                     const length = textarea.value.length;
12                     h1.textContent = `글자 수: ${length}`;
13                 });
14             });
15         </script>
16     </head>
17     <body>
18         <h1></h1>
19         <textarea></textarea>
20     </body>
21 </html>
```

The rendered page contains an

element with the text "글자 수: 5" and a field containing the Korean text "안녕하세요".

The screenshot shows a browser developer tools window with a script editor on the left and a preview panel on the right.

Script Editor (left):

```
study.html
<!DOCTYPE html>
<html>
  <head>
    <title></title>
    <script>
      document.addEventListener('DOMContentLoaded', () => {
        const h1 = document.querySelector('h1');
        const print = (event) => {
          let output = '';
          output += `alt: ${event.altKey}<br>`;
          output += `ctrl: ${event.ctrlKey}<br>`;
          output += `shift: ${event.shiftKey}<br>`;
          output += `code: ${typeof event.code === 'undefined' ? event.code : event.keyCode}<br>`;
          h1.innerHTML = output;
        };

        document.addEventListener('keydown', print);
        document.addEventListener('keyup', print);
      });
    </script>
  </head>
  <body>
    <h1></h1>
  </body>
</html>
```

Preview Panel (right):

alt: true
ctrl: true
shift: true
code: KeyC

The preview panel displays the output of the script, which is the string "alt: true\nctrl: true\nshift: true\ncode: KeyC".

키보드 이벤트 > 키보드 키 코드 사용하기

The screenshot shows a browser developer tools window with two main panels. The left panel displays the DOM tree for a file named 'study.html'. The right panel shows a preview of the page at the URL <http://127.0.0.1:3000/study.html>. The preview contains a single black star character.

```
<!DOCTYPE html>
<html>
  <head>
    <title></title>
    <script>
      document.addEventListener('DOMContentLoaded', () => {
        // 별의 초기 설정
        const star = document.querySelector('h1');
        star.style.position = 'absolute';

        // 별의 이동을 출력하는 기능
        let [x, y] = [0, 0];
        const block = 20;
        const print = () => {
          star.style.left = `${x * block}px`;
          star.style.top = `${y * block}px`;
        };
        print();

        // 별을 이동하는 기능
        const [left, up, right, down] = [37, 38, 39, 40];
        document.body.addEventListener('keydown', (event) => {
          switch (event.keyCode) {
            case left:
              x -= 1;
              break;
            case up:
              y -= 1;
              break;
            case right:
              x += 1;
              break;
            case down:
              y += 1;
              break;
          }
          print();
        });
      });
    </script>
  </head>
  <body>
    <h1>★</h1>
  </body>
</html>
```

The screenshot shows a browser developer tools window with two main panes. The left pane is a code editor for an HTML file named 'study.html'. The code contains a script block that adds a keyup event listener to a text area. This listener reads the value's length and updates the text content of an h1 element. The right pane shows the resulting web page at <http://127.0.0.1:3000/study.html>. The page displays the text "글자 수: 5" above a text area containing "안녕하세요".

```
1  <!DOCTYPE html>
2  <html>
3      <head>
4          <title></title>
5          <script>
6              const listener = (event) => {
7                  const length = event.currentTarget.value.length;
8                  const h1 = document.querySelector('h1');
9                  h1.textContent = `글자 수: ${length}`;
10             }
11
12             document.addEventListener('DOMContentLoaded', () => {
13                 const textarea = document.querySelector('textarea');
14                 textarea.addEventListener('keyup', listener);
15             });
16         </script>
17     </head>
18     <body>
19         <h1></h1>
20         <textarea>안녕하세요</textarea>
21     </body>
22 </html>
23
```

The screenshot shows a development environment with a code editor and a browser window.

Code Editor:

```
2 <html>
3   <head>
4     <title></title>
5     <script>
6       document.addEventListener('DOMContentLoaded', () => {
7         const input = document.querySelector('input');
8         const button = document.querySelector('button');
9         const p = document.querySelector('p');
10
11        button.addEventListener('click', () => {
12          // 입력을 숫자로 변환합니다.
13          const inch = Number(input.value);
14          // 숫자가 아니라면 바로 리턴합니다.
15          if (isNaN(inch)) {
16            p.textContent = '숫자를 입력해주세요';
17            return;
18          }
19          // 변환해서 출력합니다.
20          const cm = inch * 2.54;
21          p.textContent = `${cm} cm`;
22        });
23      });
24    </script>
25  </head>
26  <body>
27    <input type="text" /> inch<br />
28    <button>계산</button>
29    <p></p>
30  </body>
31</html>
```

Browser Window:

The browser displays a simple form with an input field and a button. The input field contains "26" and has a placeholder "inch". The button is labeled "계산". Below the form, the output is shown as "66.04 cm".

A second browser window below it shows the same form, but the input field now contains "숫자" (number) and the output is "숫자를 입력해주세요" (Please enter a number).

The screenshot shows a browser developer tools interface with two tabs: 'study.html' and 'javascript'.

study.html Tab:

```
<!DOCTYPE html>
<html>
  <head>
    <title></title>
    <script>
      document.addEventListener('DOMContentLoaded', () => {
        const input = document.querySelector('input');
        const p = document.querySelector('p');
        const isEmail = (value) => {
          // 골뱅이를 갖고 있고 && 골뱅이 뒤에 점이 있다면
          return value.indexOf('@') > 1 && value.split('@')[1].indexOf('.') > 1;
        };

        input.addEventListener('keyup', (event) => {
          const value = event.currentTarget.value;
          if (isEmail(value)) {
            p.style.color = 'green';
            p.textContent = `이메일 형식입니다: ${value}`;
          } else {
            p.style.color = 'red';
            p.textContent = `이메일 형식이 아닙니다: ${value}`;
          }
        });
      });
    </script>
  </head>
  <body>
    <input type="text" />
    <p></p>
  </body>
</html>
```

javascript Tab:

```
import { strictEqual } from 'assert';

// 테스트 케이스 정의
const testCases = [
  { value: 'rint@hanbit.co.kr', expectedColor: 'green', expectedTextContent: '이메일 형식입니다: rint@hanbit.co.kr' },
  { value: 'rint@', expectedColor: 'red', expectedTextContent: '이메일 형식이 아닙니다: rint@' }
];

// 테스트 실행
testCases.forEach(({ value, expectedColor, expectedTextContent }) => {
  const p = document.createElement('p');
  const isEmail = (value) => {
    // 골뱅이를 갖고 있고 && 골뱅이 뒤에 점이 있다면
    return value.indexOf('@') > 1 && value.split('@')[1].indexOf('.') > 1;
  };
  const input = document.createElement('input');
  input.value = value;
  const result = isEmail(input.value);
  strictEqual(result, true, `Email validation failed for value: ${value}`);
  strictEqual(p.style.color, expectedColor, `Color does not match for value: ${value}`);
  strictEqual(p.textContent, expectedTextContent, `Text content does not match for value: ${value}`);
});
```

Two browser windows are shown on the right, both displaying the URL `http://127.0.0.1:3000/study.html`.

- Top Window:** Shows the input field with the value `rint@`. The paragraph below it displays the message `이메일 형식이 아닙니다: rint@` in red text.
- Bottom Window:** Shows the input field with the value `rint@hanbit.co.kr`. The paragraph below it displays the message `이메일 형식입니다: rint@hanbit.co.kr` in green text.

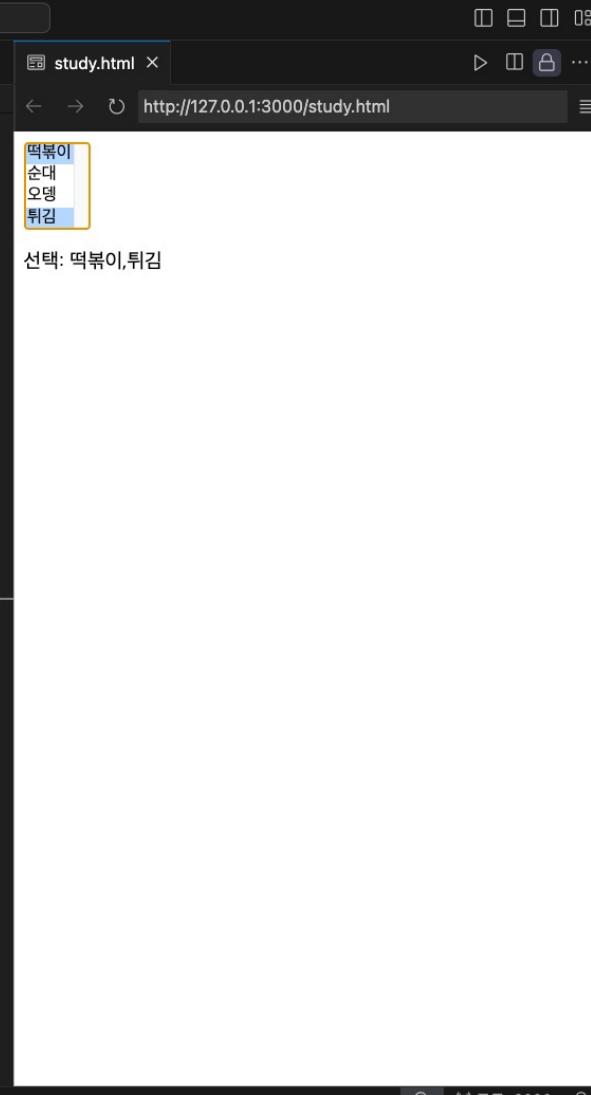
The screenshot shows a browser developer tools window with the following components:

- DOM Tree:** On the left, it displays the structure of the `study.html` page. It includes the DOCTYPE, HTML, head, and body sections. In the head section, there is a script block that adds an event listener to the document for the 'DOMContentLoaded' event. This listener selects a dropdown menu and a paragraph, then updates the paragraph's text content based on the selected option. The dropdown menu itself contains four options: '떡볶이', '순대', '오뎅', and '튀김'.

```
<!DOCTYPE html>
<html>
  <head>
    <title></title>
    <script>
      document.addEventListener('DOMContentLoaded', () => {
        const select = document.querySelector('select');
        const p = document.querySelector('p');

        select.addEventListener('change', (event) => {
          const options = event.currentTarget.options;
          const index = event.currentTarget.options.selectedIndex;

          p.textContent = `선택: ${options[index].textContent}`;
        });
      });
    </script>
  </head>
  <body>
    <select>
      <option>떡볶이</option>
      <option>순대</option>
      <option>오뎅</option>
      <option>튀김</option>
    </select>
    <p>선택: 떡볶이</p>
  </body>
</html>
```
- Script Editor:** The main panel contains the same script code as the DOM tree.
- Preview:** On the right, a preview pane shows the resulting web page at <http://127.0.0.1:3000/study.html>. The dropdown menu is visible, and the paragraph below it displays the text "선택: 떡볶이".



```
<!DOCTYPE html>
<html>
  <head>
    <title></title>
    <script>
      document.addEventListener('DOMContentLoaded', () => {
        const select = document.querySelector('select');
        const p = document.querySelector('p');

        select.addEventListener('change', (event) => {
          const options = event.currentTarget.options;
          const list = [];
          for (const option of options) {
            if (option.selected) {
              list.push(option.textContent);
            }
          }
          p.textContent = `선택: ${list.join(',')}`;
        });
      });
    </script>
  </head>
  <body>
    <select multiple>
      <option>떡볶이</option>
      <option>순대</option>
      <option>오뎅</option>
      <option>튀김</option>
    </select>
    <p></p>
  </body>
</html>
```

글자 입력 양식 이벤트 > 드롭다운 목록 활용

The screenshot shows a browser developer tools window with two panes. The left pane displays the HTML and JavaScript code for a file named 'study.html'. The right pane shows the browser's rendering of the page at the URL <http://127.0.0.1:3000/study.html>.

```
<!DOCTYPE html>
<html>
  <head>
    <title></title>
    <script>
      document.addEventListener('DOMContentLoaded', () => {
        let 현재값;
        let 변환상수 = 10;

        const select = document.querySelector('select');
        const input = document.querySelector('input');
        const span = document.querySelector('span');

        const calculate = () => {
          span.textContent = (현재값 * 변환상수).toFixed(2);
        };

        select.addEventListener('change', (event) => {
          const options = event.currentTarget.options;
          const index = event.currentTarget.options.selectedIndex;
          변환상수 = Number(options[index].value);
          calculate();
        });

        input.addEventListener('keyup', (event) => {
          현재값 = Number(event.currentTarget.value);
          calculate();
        });
      });
    </script>
  </head>
  <body>
    <input type="text" /> cm =
    <span></span>
    <select>
      <option value="10">mm</option>
      <option value="0.01">m</option>
      <option value="0.393701">inch</option>
    </select>
  </body>
</html>
```

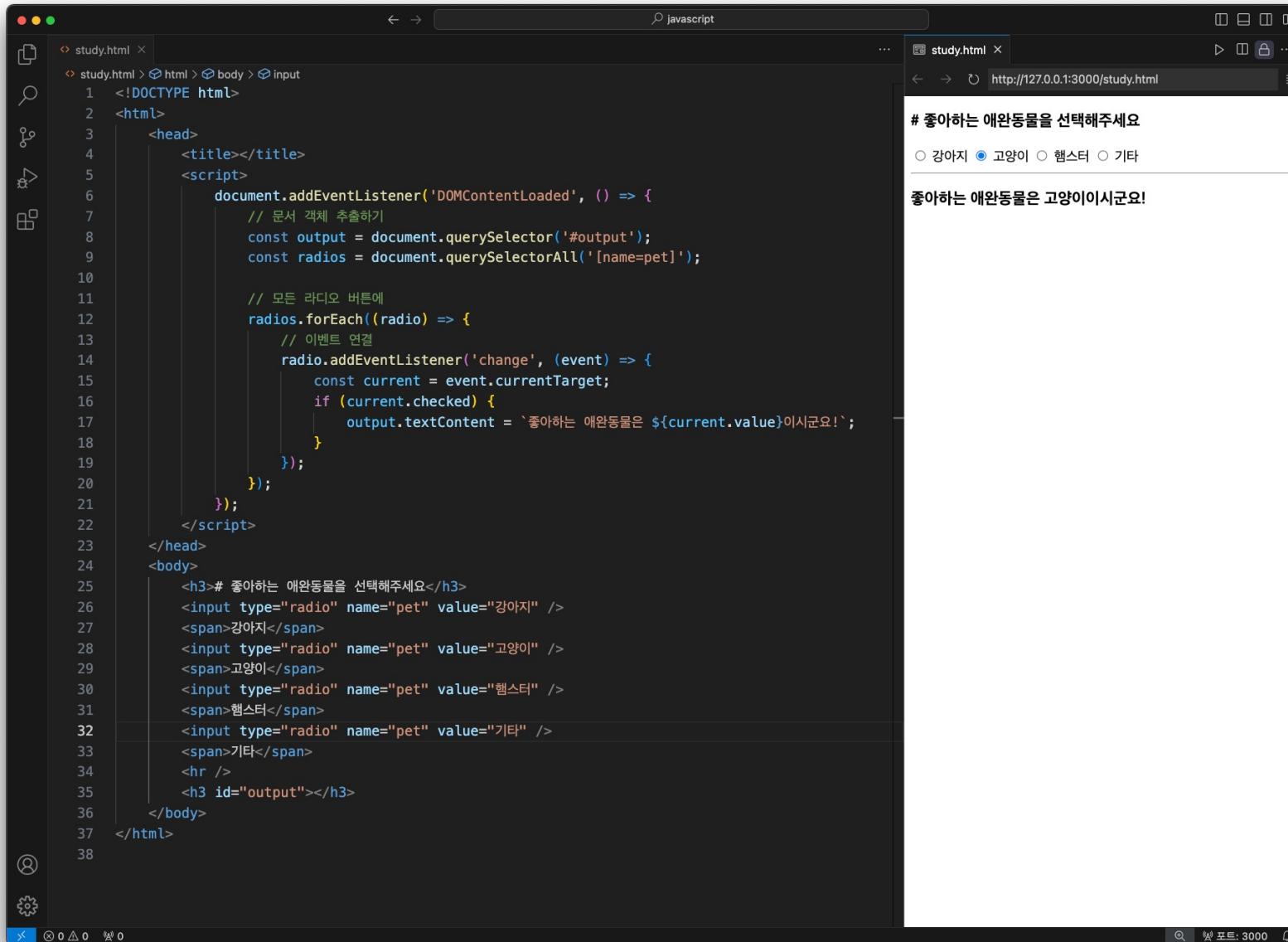
In the browser preview, a text input field contains '10' and a span element displays 'cm = 3.94'. A dropdown menu is open below the span, showing three options: 'mm', 'm', and 'inch'. The 'inch' option is highlighted.

The screenshot shows a browser window with developer tools open. The left panel displays the HTML code for 'study.html'. The right panel shows the browser's preview area with a checked checkbox labeled '타이머 활성화' and a large text '15초' indicating a 15-second timer.

```
<!DOCTYPE html>
<html>
  <head>
    <title></title>
    <script>
      document.addEventListener('DOMContentLoaded', () => {
        let [timer, timerId] = [0, 0];
        const h1 = document.querySelector('h1');
        const checkbox = document.querySelector('input');

        checkbox.addEventListener('change', (event) => {
          if (event.currentTarget.checked) {
            // 체크 상태
            timerId = setInterval(() => {
              timer += 1;
              h1.textContent = `${timer}초`;
            }, 1000);
          } else {
            // 체크 해제 상태
            clearInterval(timerId);
          }
        });
      });
    </script>
  </head>
  <body>
    <input type="checkbox" />
    <span>타이머 활성화</span>
    <h1></h1>
  </body>
</html>
```

글자 입력 양식 이벤트 > 라디오 버튼 활용



The image shows a screenshot of a browser window displaying a web page titled "study.html". The page content is as follows:

```
<!DOCTYPE html>
<html>
  <head>
    <title></title>
    <script>
      document.addEventListener('DOMContentLoaded', () => {
        // 문서 객체 추출하기
        const output = document.querySelector('#output');
        const radios = document.querySelectorAll('[name=pet]');

        // 모든 라디오 버튼에
        radios.forEach((radio) => {
          // 이벤트 연결
          radio.addEventListener('change', (event) => {
            const current = event.currentTarget;
            if (current.checked) {
              output.textContent = `좋아하는 애완동물은 ${current.value}이시군요!`;
            }
          });
        });
      });
    </script>
  </head>
  <body>
    <h3># 좋아하는 애완동물을 선택해주세요</h3>
    <input type="radio" name="pet" value="강아지" />
    <span>강아지</span>
    <input type="radio" name="pet" value="고양이" />
    <span>고양이</span>
    <input type="radio" name="pet" value="햄스터" />
    <span>햄스터</span>
    <input type="radio" name="pet" value="기타" />
    <span>기타</span>
    <hr />
    <h3 id="output"></h3>
  </body>
</html>
```

The browser's status bar at the bottom right indicates "포트: 3000".

The screenshot shows a browser window with two tabs. The left tab is titled 'study.html' and contains the following HTML and JavaScript code:

```
1  <!DOCTYPE html>
2  <html>
3      <head>
4          <title></title>
5          <script>
6              document.addEventListener('DOMContentLoaded', () => {
7                  const imgs = document.querySelectorAll('img');
8
9                  imgs.forEach((img) => {
10                     img.addEventListener('contextmenu', (event) => {
11                         event.preventDefault();
12                     });
13                 });
14             });
15         </script>
16     </head>
17     <body>
18         
19     </body>
20 </html>
21 
```

The right tab is also titled 'study.html' and shows the page at <http://127.0.0.1:3000/study.html>. It displays a single image of a cat sitting on a cobblestone path.

The screenshot shows a browser developer tools window with two main panes. The left pane is a code editor for an HTML file named 'study.html'. It contains the following code:

```
1 <!DOCTYPE html>
2 <html>
3   <head>
4     <title></title>
5     <script>
6       document.addEventListener('DOMContentLoaded', () => {
7         let status = false;
8
9         const checkbox = document.querySelector('input');
10        checkbox.addEventListener('change', (event) => {
11          status = event.currentTarget.checked;
12        });
13
14        const link = document.querySelector('a');
15        link.addEventListener('click', (event) => {
16          if (!status) {
17            event.preventDefault();
18          }
19        });
20      });
21    </script>
22  </head>
23  <body>
24    <input type="checkbox" />
25    <span>링크 활성화</span>
26    <br />
27    <a href="http://hanbit.co.kr">한빛미디어</a>
28  </body>
29</html>
```

The right pane is a preview of the browser window, showing the rendered HTML with a checked checkbox and a link to '한빛미디어'.

The screenshot shows a browser window with two panes. The left pane displays the source code of a file named 'study.html'. The right pane shows the rendered HTML page at the URL <http://127.0.0.1:3000/study.html>.

study.html Content:

```
1 <!DOCTYPE html>
2 <html>
3   <head>
4     <title></title>
5     <script>
6       document.addEventListener('DOMContentLoaded', () => {
7         const textarea = document.querySelector('textarea');
8         const h1 = document.querySelector('h1');
9         let timerId;
10
11         textarea.addEventListener('focus', (event) => {
12           timerId = setInterval(() => {
13             const length = textarea.value.length;
14             h1.textContent = `글자 수: ${length}`;
15           }, 50);
16         });
17         textarea.addEventListener('blur', (event) => {
18           clearInterval(timerId);
19         });
20       });
21     </script>
22   </head>
23   <body>
24     <h1></h1>
25     <textarea></textarea>
26   </body>
27 </html>
28 
```

Rendered Page:

The page displays the text "글자 수: 5" above a text area containing the placeholder "안녕하세요".

목차

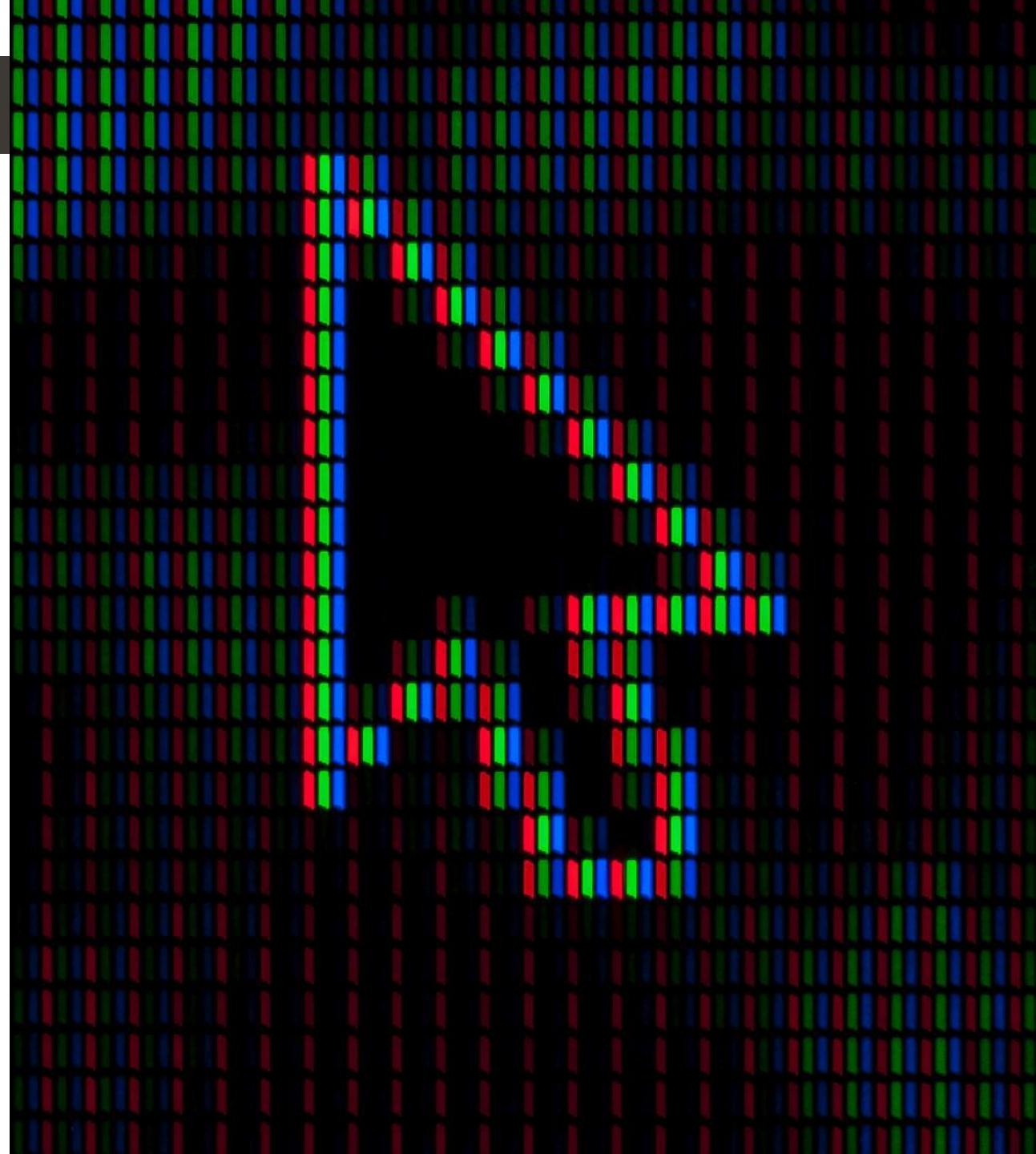
a table of contents

6 객체

7 문서 객체 모델

8 예외 처리

9 클래스



A close-up photograph of a person's hands holding a white smartphone. The person is wearing a white button-down shirt. The background is blurred.

구문 오류와 예외

The screenshot shows a terminal window with the following details:

- Title Bar:** javascript
- File:** study.js
- Code Content:**

```
1 //프로그램 시작 확인
2 console.log('# 프로그램이 시작되었습니다!');
3
4 //구문 오류가 발생하는 부분
5 console.log('괄호를 닫지 않는 실수를 했습니다.')
6
```

- Output:**

```
[Running] node "/Users/seeker/javascript/study.js"
/Users/seeker/javascript/study.js:5
console.log('괄호를 닫지 않는 실수를 했습니다.'
               ^^^^^^^^^^

SyntaxError: missing ) after argument list
  at wrapSafe (node:internal/modules/cjs/loader:1281:20)
  at Module._compile (node:internal/modules/cjs/loader:1321:27)
  at Module._extensions..js (node:internal/modules/cjs/loader:1416:10)
  at Module.load (node:internal/modules/cjs/loader:1208:32)
  at Module._load (node:internal/modules/cjs/loader:1024:12)
  at Function.executeUserEntryPoint [as runMain]
  (node:internal/modules/run_main:174:12)
  at node:internal/main/run_main_module:28:49

Node.js v20.13.1

[Done] exited with code=1 in 0.114 seconds
```

- Bottom Status Bar:** 줄 5, 열 34 공백: 4 UTF-8 LF {} JavaScript 포트: 3000 ✓ Prettier

The screenshot shows a terminal window with the following content:

```
study.js
1 //프로그램 시작 확인
2 console.log('# 프로그램이 시작되었습니다!');
3
4 //구문 오류가 발생하는 부분
5 console.rog('괄호를 닫지 않는 실수를 했습니다.');
6

[Running] node "/Users/seeker/javascript/study.js"
# 프로그램이 시작되었습니다!
/Users/seeker/javascript/study.js:5
console.rog('괄호를 닫지 않는 실수를 했습니다.');

TypeError: console.rog is not a function
    at Object.<anonymous> (/Users/seeker/javascript/study.js:5:9)
    at Module._compile (node:internal/modules/cjs/loader:1358:14)
    at Module._extensions..js (node:internal/modules/cjs/loader:1416:10)
    at Module.load (node:internal/modules/cjs/loader:1208:32)
    at Module._load (node:internal/modules/cjs/loader:1024:12)
    at Function.executeUserEntryPoint [as runMain]
    (node:internal/modules/run_main:174:12)
    at node:internal/main/run_main_module:28:49

Node.js v20.13.1

[Done] exited with code=1 in 0.103 seconds
```

The terminal window has a dark theme. The file path is shown as `/Users/seeker/javascript/study.js`. The error message is `TypeError: console.rog is not a function`, indicating a misspelling of `console.log`. The stack trace shows the error occurred at line 5 of the script. The Node.js version is `v20.13.1`. The process exits with a code of 1.

The screenshot shows a browser developer tools interface with the 'Console' tab selected. A red error message is displayed:

```
Uncaught TypeError: Cannot set properties of null (setting 'textContent')
at HTMLDocument.<anonymous> (study.html:10:28)
```

The code in the 'study.html' file is as follows:

```
<!DOCTYPE html>
<html>
  <head>
    <title></title>
  </head>
  <body></body>
  <script>
    document.addEventListener('DOMContentLoaded', () => {
      const h1 = document.querySelector('h1');
      h1.textContent = '안녕하세요';
    });
  </script>
</html>
```

The error occurs at line 10, character 28, where the script attempts to set the `textContent` property of a null object reference.

The screenshot shows a browser developer tools interface with the 'Console' tab selected. The code being run is:

```
1 <!DOCTYPE html>
2 <html>
3   <head>
4     <title></title>
5   </head>
6   <body></body>
7   <script>
8     document.addEventListener('DOMContentLoaded', () => {
9       const h1 = document.querySelector('h1');
10      if (h1) {
11        h1.textContent = '안녕하세요';
12      } else {
13        console.log('h1 태그를 추출할 수 없습니다.');
14      }
15    });
16   </script>
17 </html>
18 
```

The console output is:

```
h1 태그를 추출할 수 없습니다. vscode_livepreview_injected_script:141
```

A screenshot of a dark-themed code editor, likely VS Code, showing a file named `study.js`. The code contains a `try-catch-finally` block with comments explaining its purpose:

```
JS study.js > ...
1 try {
2     //예외가 발생할 가능성이 있는 코드
3 } catch (exception) {
4     //예외가 발생했을 때 실행할 코드
5 } finally {
6     //무조건 실행할 코드
7 }
8
```

The right side of the interface shows the output of running the script with `node`:

```
[Running] node "/Users/seeker/javascript/study.js"
[Done] exited with code=0 in 0.081 seconds
```

The status bar at the bottom provides information about the file: 줄 8, 열 1, 공백: 4, UTF-8, LF, JavaScript, 포트: 3000, Prettier.

The screenshot shows a dark-themed code editor interface. On the left is a sidebar with various icons: file, search, refresh, and settings. The main area has a title bar with a back/forward button, a search bar containing "javascript", and a tab labeled "study.js". The code editor displays the following JavaScript code:

```
study.js > ...
1  try {
2      willExcept.byeBye();
3      console.log('try 구문의 마지막 줄');
4  } catch (exception) {
5      console.log('catch 구문의 마지막 줄');
6 }
7
```

To the right of the code editor is a terminal window titled "Output" which shows the execution results:

```
[Running] node "/Users/seeker/javascript/study.js"
catch 구문의 마지막 줄

[Done] exited with code=0 in 0.089 seconds
```

At the bottom of the terminal window, there are status indicators: 줄 7, 열 1, 공백: 4, LF, JavaScript, 포트: 3000, Prettier, and a bell icon.

The screenshot shows a dark-themed code editor interface with a sidebar on the left containing icons for file operations, search, and settings. The main area has tabs for 'study.js' and 'study.js > ...'. The code in 'study.js' is as follows:

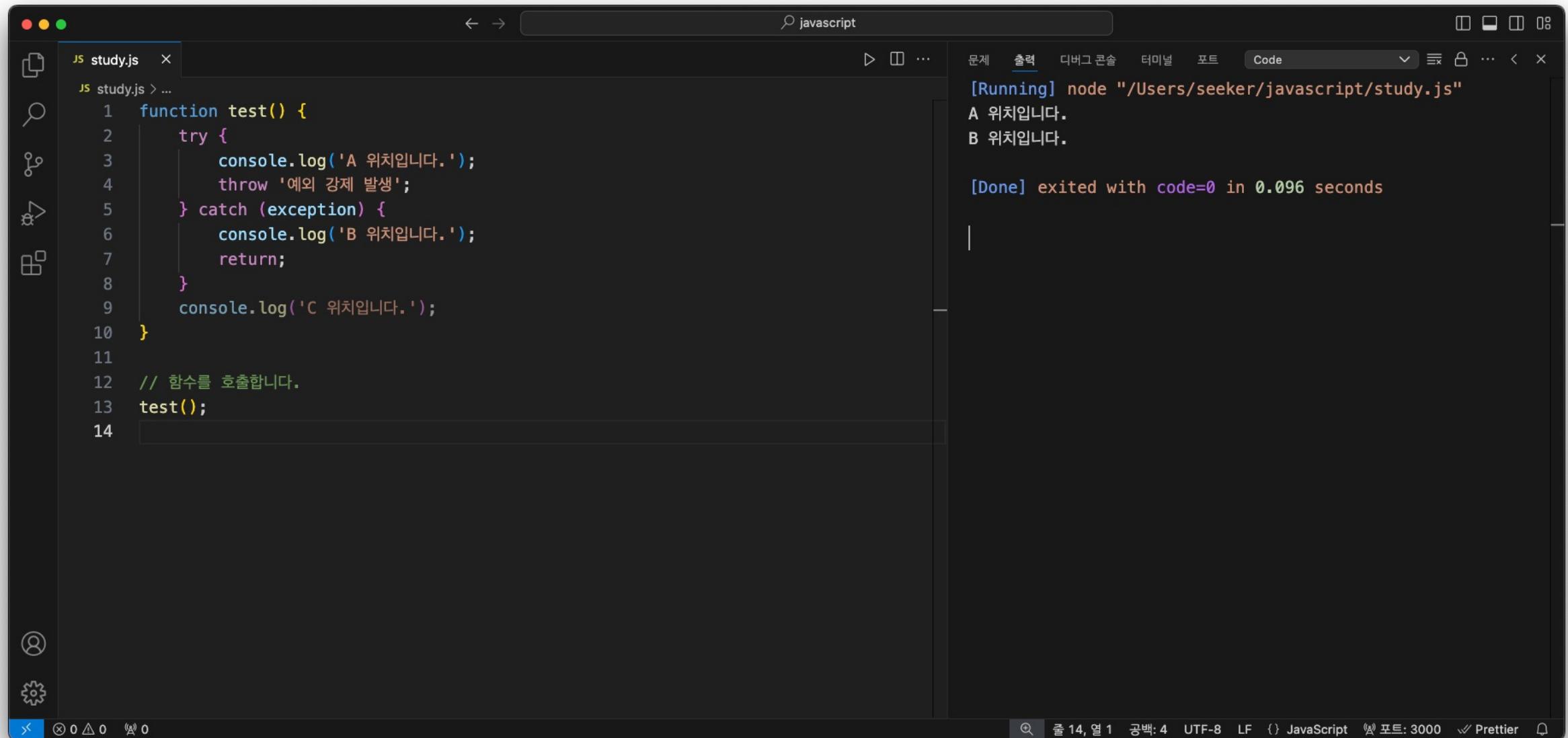
```
JS study.js > ...
1 try {
2     willExcept.byeBye();
3     console.log('try 구문의 마지막 줄');
4 } catch (exception) {
5     console.log('catch 구문의 마지막 줄');
6 } finally {
7     console.log('finally 구문의 마지막 줄');
8 }
9
```

The right side of the interface displays the execution results of the script:

[Running] node "/Users/seeker/javascript/study.js"
catch 구문의 마지막 줄
finally 구문의 마지막 줄

[Done] exited with code=0 in 0.086 seconds

At the bottom, there are status indicators for tabs, file count, encoding, port, and Prettier.



```
study.js > ...
1  function test() {
2    try {
3      console.log('A 위치입니다.');
4      throw '예외 강제 발생';
5    } catch (exception) {
6      console.log('B 위치입니다.');
7      return;
8    }
9    console.log('C 위치입니다.');
10 }
11
12 // 함수를 호출합니다.
13 test();
```

[Running] node "/Users/seeker/javascript/study.js"
A 위치입니다.
B 위치입니다.
[Done] exited with code=0 in 0.096 seconds

줄 14, 열 1 공백: 4 UTF-8 LF {} JavaScript 포트: 3000 Prettier

```
study.js > ...
1 function test() {
2     try {
3         console.log('A 위치입니다.');
4         throw '예외 강제 발생';
5     } catch (exception) {
6         console.log('B 위치입니다.');
7         return;
8     } finally {
9         console.log('C 위치입니다.');
10    }
11 }
12
13 // 함수를 호출합니다.
14 test();
15
```

[Running] node "/Users/seeker/javascript/study.js"

A 위치입니다.
B 위치입니다.
C 위치입니다.

[Done] exited with code=0 in 0.086 seconds

줄 15, 열 1 공백: 4 UTF-8 LF () JavaScript 포트: 3000 ✓ Prettier

A close-up photograph of a person's hands holding a white smartphone. The person is wearing a white button-down shirt. The background is blurred.

예외 처리 고급

The screenshot shows a terminal window with a dark background. On the left is a file browser sidebar with icons for files, search, and settings. The main area has tabs for '문서' (Document) and '출력' (Output). The 'Output' tab is active, displaying the following text:

```
[Running] node "/Users/seeker/javascript/study.js"
RangeError: Invalid array length
    at Object.<anonymous> (/Users/seeker/javascript/study.js:2:19)
    at Module._compile (node:internal/modules/cjs/loader:1358:14)
    at Module._extensions..js (node:internal/modules/cjs/loader:1416:10)
    at Module.load (node:internal/modules/cjs/loader:1208:32)
    at Module._load (node:internal/modules/cjs/loader:1024:12)
    at Function.executeUserEntryPoint [as runMain]
      (node:internal/modules/run_main:174:12)
    at node:internal/main/run_main_module:28:49

예외 이름: RangeError
예외 메시지: Invalid array length

[Done] exited with code=0 in 0.174 seconds
```

At the bottom of the window, there are status indicators for file operations (like saving and closing), encoding (UTF-8), line separator (LF), file type (JavaScript), port (3000), and Prettier.

```
study.js
1  function divide(a, b) {
2    if (b === 0) {
3      throw '0으로는 나눌 수 없습니다.';
4    }
5    return a / b;
6  }
7
8  console.log(divide(10, 2));
9  console.log(divide(10, 0));
10 
```

[Running] node "/Users/seeker/javascript/study.js"

5

/Users/seeker/javascript/study.js:3
| throw '0으로는 나눌 수 없습니다.';
|
0으로는 나눌 수 없습니다.
(Use `node --trace-uncaught ...` to show where the exception was thrown)

Node.js v20.13.1

[Done] exited with code=1 in 0.095 seconds

줄 10, 열 1 공백: 4 UTF-8 LF () JavaScript 포트: 3000 ✓ Prettier

```
study.js > ...
1  function test(object) {
2    console.log(object.a + object.b);
3  }
4
5  test({});
6
```

[Running] node "/Users/seeker/javascript/study.js"
NaN
[Done] exited with code=0 in 0.098 seconds

The screenshot shows a terminal window with the following content:

```
study.js
1 function test(object) {
2     if (object.a !== undefined && object.b !== undefined) {
3         console.log(object.a + object.b);
4     } else {
5         throw new Error('a 속성과 b 속성을 지정하지 않았습니다.');
6     }
7 }
8
9 test({});
10
```

[Running] node "/Users/seeker/javascript/study.js"
/Users/seeker/javascript/study.js:5
 | throw new Error('a 속성과 b 속성을 지정하지 않았습니다.');//
 |
Error: a 속성과 b 속성을 지정하지 않았습니다.
at test (/Users/seeker/javascript/study.js:5:15)
at Object.<anonymous> (/Users/seeker/javascript/study.js:9:1)
at Module._compile (node:internal/modules/cjs/loader:1358:14)
at Module._extensions..js (node:internal/modules/cjs/loader:1416:10)
at Module.load (node:internal/modules/cjs/loader:1208:32)
at Module._load (node:internal/modules/cjs/loader:1024:12)
at Function.executeUserEntryPoint [as runMain] (node:internal/modules/run_main:174:12)
at node:internal/main/run_main_module:28:49

Node.js v20.13.1

[Done] exited with code=1 in 0.09 seconds

목차

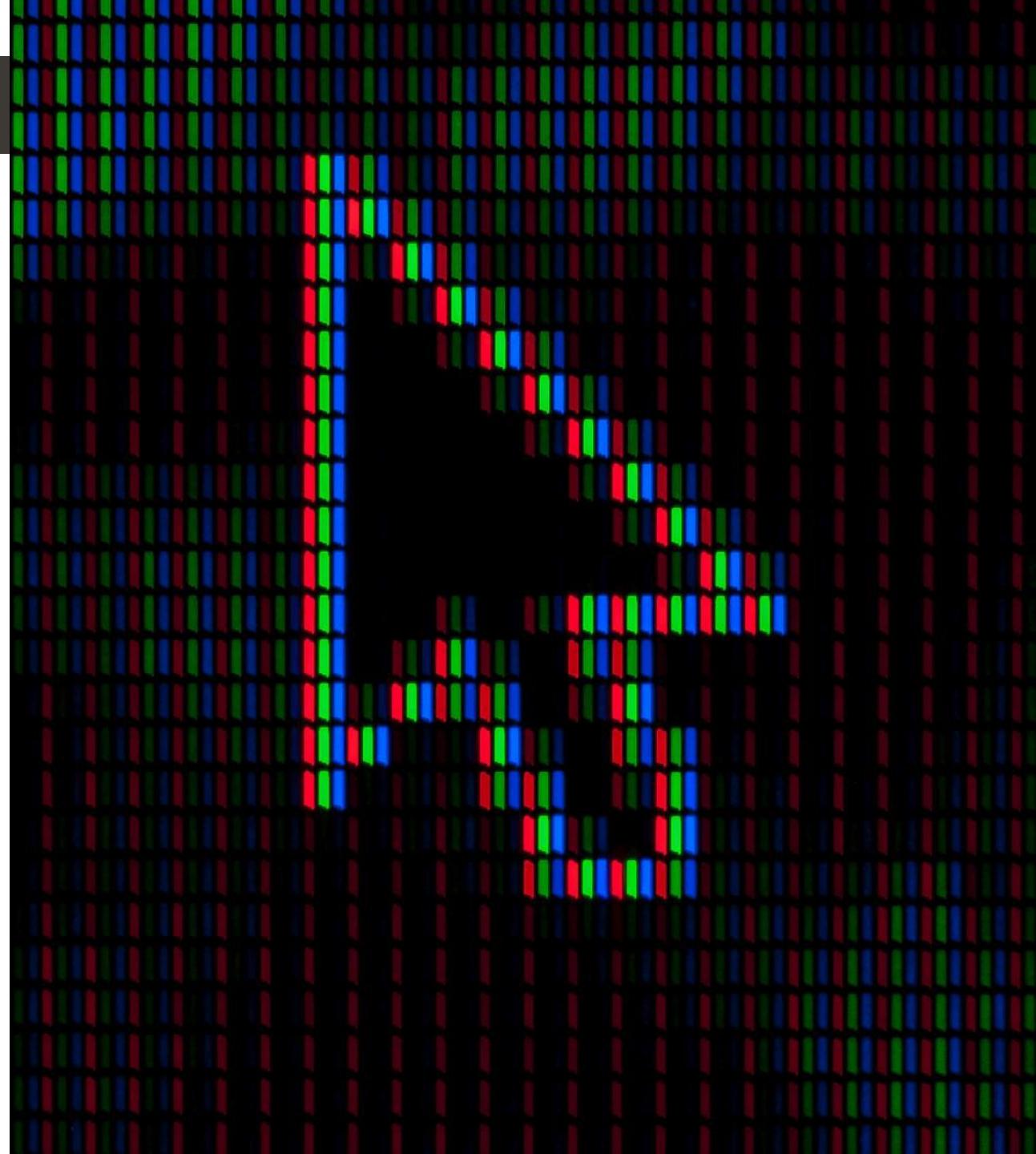
a table of contents

6 객체

7 문서 객체 모델

8 예외 처리

9 클래스



A close-up photograph of a person's hands holding a white smartphone. The person is wearing a white button-down shirt. The background is blurred.

기본기능

The screenshot shows a macOS terminal window with a dark theme. The title bar says "javascript". The main area displays the contents of a file named "study.js" and its execution output.

```
study.js
1 // 객체를 선언합니다.
2 const students = [];
3 students.push({ 이름: '구름', 국어: 87, 영어: 98, 수학: 88, 과학: 90 });
4 students.push({ 이름: '별이', 국어: 92, 영어: 98, 수학: 96, 과학: 88 });
5 students.push({ 이름: '겨울', 국어: 76, 영어: 96, 수학: 94, 과학: 86 });
6 students.push({ 이름: '바다', 국어: 98, 영어: 52, 수학: 98, 과학: 92 });
7
8 // 출력합니다.
9 console.log(JSON.stringify(students, null, 2));
10
```

[Running] node "/Users/seeker/javascript/study.js"

```
[{"이름": "구름", "국어": 87, "영어": 98, "수학": 88, "과학": 90}, {"이름": "별이", "국어": 92, "영어": 98, "수학": 96, "과학": 88}, {"이름": "겨울", "국어": 76, "영어": 96, "수학": 94, "과학": 86}, {"이름": "바다", "국어": 98, "영어": 52, "수학": 98, "과학": 92}]
```

```
study.js
1 // 객체를 선언합니다.
2 const students = [];
3 students.push({ 이름: '구름', 국어: 87, 영어: 98, 수학: 88, 과학: 90 });
4 students.push({ 이름: '별이', 국어: 92, 영어: 98, 수학: 96, 과학: 88 });
5 students.push({ 이름: '겨울', 국어: 76, 영어: 96, 수학: 94, 과학: 86 });
6 students.push({ 이름: '바다', 국어: 98, 영어: 52, 수학: 98, 과학: 92 });
7
8 // 출력합니다.
9 let output = '이름\t총점\t평균\n';
10 for (const s of students) {
11     const sum = s.국어 + s.영어 + s.수학 + s.과학;
12     const average = sum / 4;
13     output += `${s.이름}\t${sum}점\t${average}점\n`;
14 }
15 console.log(output);
16
```

[Running] node "/Users/seeker/javascript/study.js"

이름	총점	평균
구름	363점	90.75점
별이	374점	93.5점
겨울	352점	88점
바다	340점	85점

[Done] exited with code=0 in 0.092 seconds

줄 16, 열 1 공백: 4 UTF-8 LF () JavaScript 포트: 3000 ✓ Prettier

```
study.js > ...
1 // 객체를 선언합니다.
2 const students = [];
3 students.push({ 이름: '구름', 국어: 87, 영어: 98, 수학: 88, 과학: 90 });
4 students.push({ 이름: '별이', 국어: 92, 영어: 98, 수학: 96, 과학: 88 });
5 students.push({ 이름: '겨울', 국어: 76, 영어: 96, 수학: 94, 과학: 86 });
6 students.push({ 이름: '바다', 국어: 98, 영어: 52, 수학: 98, 과학: 92 });
7
8 // 객체를 처리하는 함수를 선언합니다.
9 function getSumOf(student) {
10     return student.국어 + student.영어 + student.수학 + student.과학;
11 }
12
13 function getAverageOf(student) {
14     return getSumOf(student) / 4;
15 }
16
17 // 출력합니다.
18 let output = '이름\t총점\t평균\n';
19 for (const s of students) {
20     output += `${s.이름}\t${getSumOf(s)}점\t${getAverageOf(s)}점\n`;
21 }
22
23 console.log(output);
24
```

[Running] node "/Users/seeker/javascript/study.js"

이름	총점	평균
구름	363점	90.75점
별이	374점	93.5점
겨울	352점	88점
바다	340점	85점

[Done] exited with code=0 in 0.09 seconds

줄 24, 열 1 공백: 4 UTF-8 LF () JavaScript 포트: 3000 ✓ Prettier

The screenshot shows a terminal window with the following details:

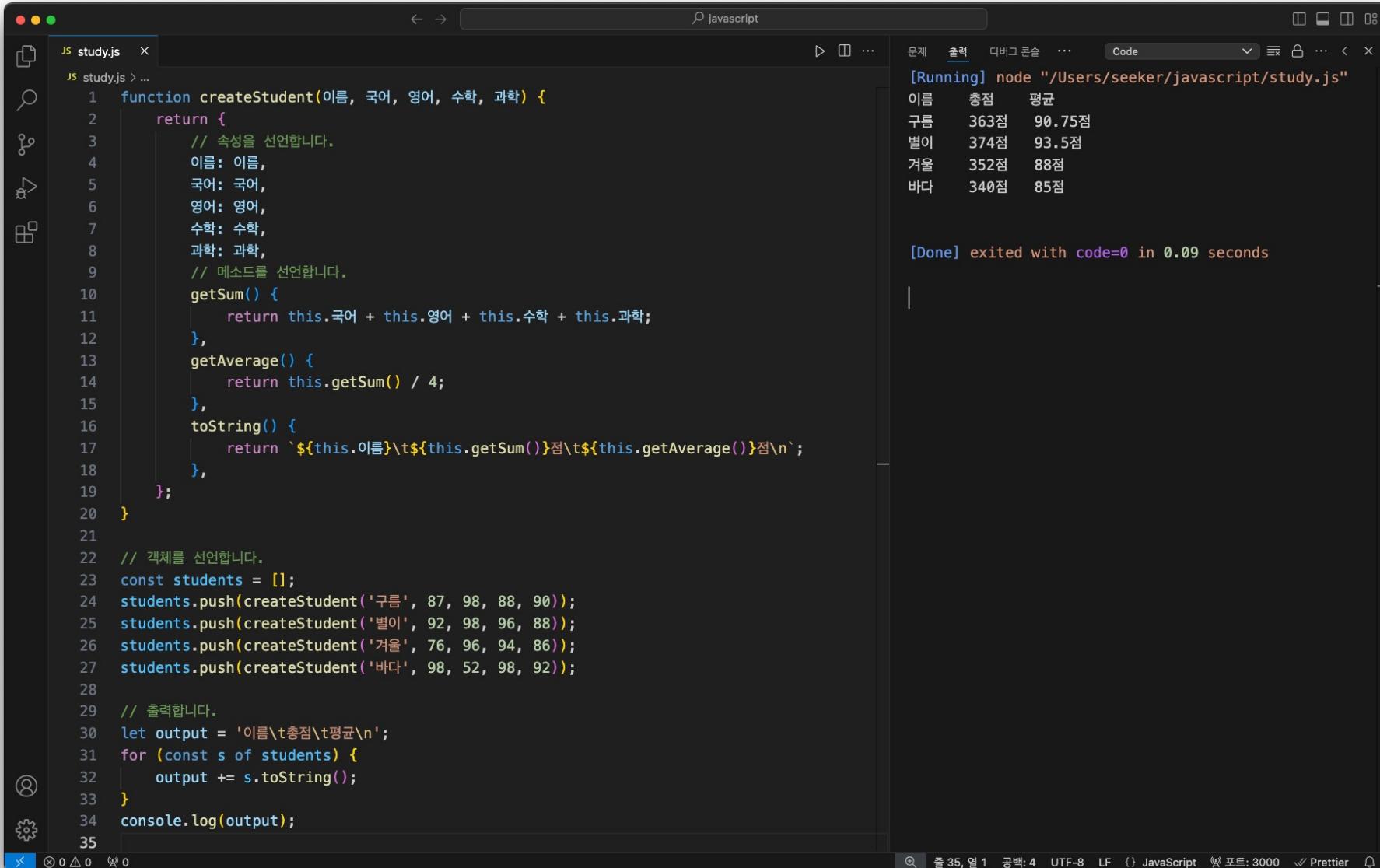
- Title Bar:** javascript
- File:** study.js
- Code Content:**

```
1 // 객체를 선언합니다.
2 const students = [];
3 students.push({ 이름: '구름', 국어: 87, 영어: 98, 수학: 88, 과학: 90 });
4 students.push({ 이름: '별이', 국어: 92, 영어: 98, 수학: 96, 과학: 88 });
5 students.push({ 이름: '겨울', 국어: 76, 영어: 96, 수학: 94, 과학: 86 });
6 students.push({ 이름: '바다', 국어: 98, 영어: 52, 수학: 98, 과학: 92 });
7
8 // students 배열 내부의 객체 모두에 메소드를 추가합니다.
9 for (const student of students) {
10     student.getSum = function () {
11         return this.국어 + this.영어 + this.수학 + this.과학;
12     };
13
14     student.getAverage = function () {
15         return this.getSum() / 4;
16     };
17 }
18
19 // 출력합니다.
20 let output = '이름\t총점\t평균\n';
21 for (const s of students) {
22     output += `${s.이름}\t${s.getSum()}점\t${s.getAverage()}점\n`;
23 }
24 console.log(output);
25
```
- Output:**

```
[Running] node "/Users/seeker/javascript/study.js"
이름    총점    평균
구름    363점   90.75점
별이    374점   93.5점
겨울    352점   88점
바다    340점   85점

[Done] exited with code=0 in 0.084 seconds
```
- Bottom Status Bar:** 줄 25, 열 1 공백: 4 UTF-8 LF () JavaScript 포트: 3000 Prettier

객체의 기능을 메소드로 추가하기



```
study.js
1 function createStudent(이름, 국어, 영어, 수학, 과학) {
2     return {
3         // 속성을 선언합니다.
4         이름: 이름,
5         국어: 국어,
6         영어: 영어,
7         수학: 수학,
8         과학: 과학,
9         // 메소드를 선언합니다.
10        getSum() {
11            return this.국어 + this.영어 + this.수학 + this.과학;
12        },
13        getAverage() {
14            return this.getSum() / 4;
15        },
16        toString() {
17            return `${this.이름}\t${this.getSum()}점\t${this.getAverage()}점\n`;
18        },
19    };
20 }
21 // 객체를 선언합니다.
22 const students = [];
23 students.push(createStudent('구름', 87, 98, 88, 90));
24 students.push(createStudent('별이', 92, 98, 96, 88));
25 students.push(createStudent('겨울', 76, 96, 94, 86));
26 students.push(createStudent('바다', 98, 52, 98, 92));
27
28 // 출력합니다.
29 let output = '이름\t총점\t평균\n';
30 for (const s of students) {
31     output += s.toString();
32 }
33 console.log(output);
34
35
```

[Running] node "/Users/seeker/javascript/study.js"

이름	총점	평균
구름	363점	90.75점
별이	374점	93.5점
겨울	352점	88점
바다	340점	85점

[Done] exited with code=0 in 0.09 seconds

The screenshot shows a dark-themed code editor interface with a sidebar on the left containing icons for file operations, search, and settings. The main area has a title bar with tabs for 'study.js' and 'javascript'. A search bar at the top right contains the text 'javascript'. The code editor displays the following JavaScript code:

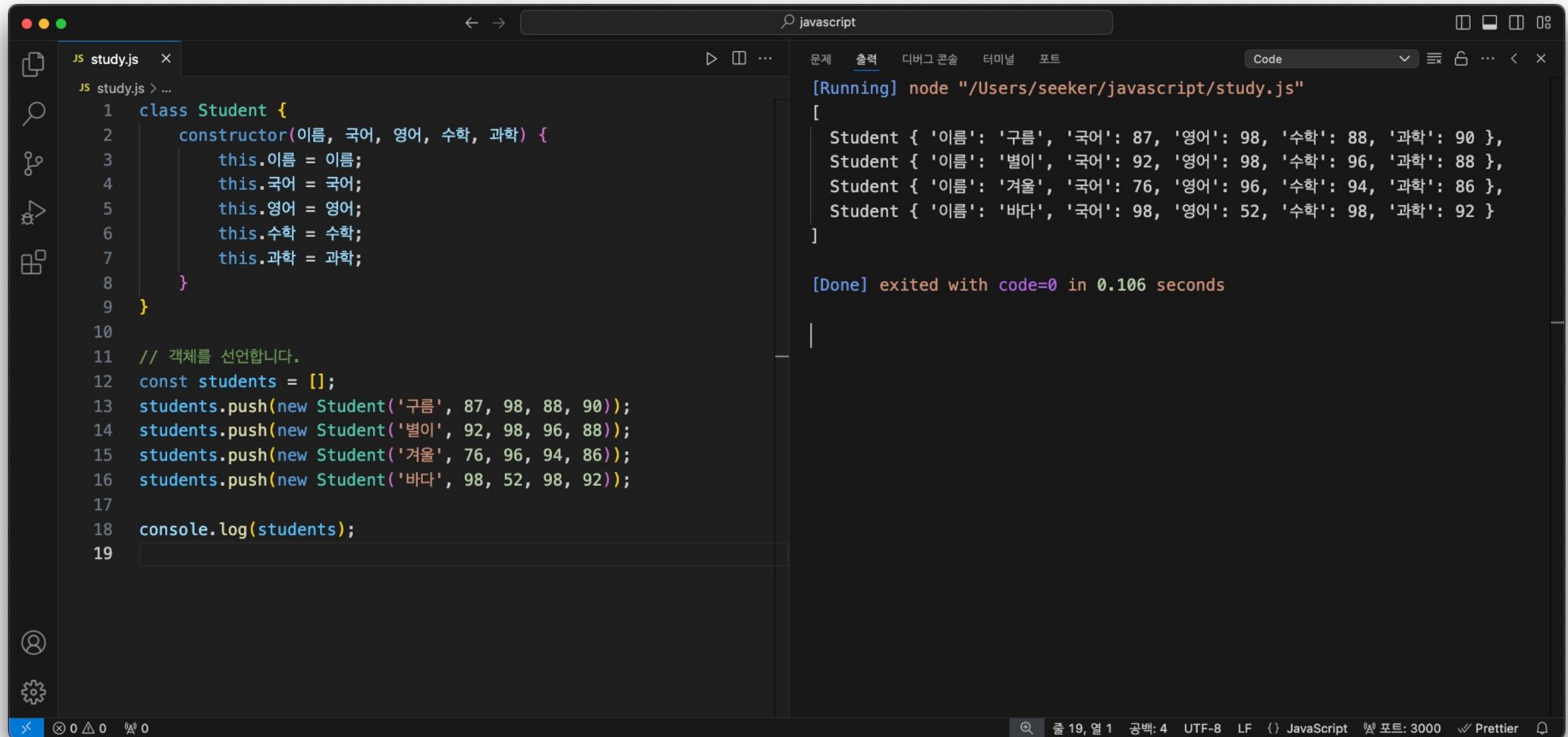
```
study.js > ...
1 // 클래스를 선언합니다.
2 class Student {
3
4 }
5
6 // 학생을 선언합니다.
7 const student = new Student();
8
9 // 학생 리스트를 선언합니다.
10 const students = [
11     new Student(),
12     new Student(),
13     new Student(),
14     new Student()];
15
16 console.log(students);
17
```

The output window on the right shows the results of running the script with Node.js:

```
[Running] node "/Users/seeker/javascript/study.js"
[ Student {}, Student {}, Student {}, Student {} ]

[Done] exited with code=0 in 0.105 seconds
```

At the bottom, there are status indicators for file changes, line numbers (줄 17), and character count (열 1).



```
study.js
1  class Student {
2      constructor(이름, 국어, 영어, 수학, 과학) {
3          this.이름 = 이름;
4          this.국어 = 국어;
5          this.영어 = 영어;
6          this.수학 = 수학;
7          this.과학 = 과학;
8      }
9
10
11 // 객체를 선언합니다.
12 const students = [];
13 students.push(new Student('구름', 87, 98, 88, 90));
14 students.push(new Student('별이', 92, 98, 96, 88));
15 students.push(new Student('겨울', 76, 96, 94, 86));
16 students.push(new Student('바다', 98, 52, 98, 92));
17
18 console.log(students);
19
```

[Running] node "/Users/seeker/javascript/study.js"

```
[  
  Student { '이름': '구름', '국어': 87, '영어': 98, '수학': 88, '과학': 90 },  
  Student { '이름': '별이', '국어': 92, '영어': 98, '수학': 96, '과학': 88 },  
  Student { '이름': '겨울', '국어': 76, '영어': 96, '수학': 94, '과학': 86 },  
  Student { '이름': '바다', '국어': 98, '영어': 52, '수학': 98, '과학': 92 }  
]
```

[Done] exited with code=0 in 0.106 seconds

```
study.js
1  class Student {
2      constructor(이름, 국어, 영어, 수학, 과학) {
3          this.이름 = 이름;
4          this.국어 = 국어;
5          this.영어 = 영어;
6          this.수학 = 수학;
7          this.과학 = 과학;
8      }
9
10     getSum() {
11         return this.국어 + this.영어 + this.수학 + this.과학;
12     }
13     getAverage() {
14         return this.getSum() / 4;
15     }
16     toString() {
17         return `${this.이름}\t${this.getSum()}점\t${this.getAverage()}점\n`;
18     }
19 }
20 // 객체를 선언합니다.
21 const students = [];
22 students.push(new Student('구름', 87, 98, 88, 90));
23 students.push(new Student('별이', 92, 98, 96, 88));
24 students.push(new Student('겨울', 76, 96, 94, 86));
25 students.push(new Student('바다', 98, 52, 98, 92));
26
27 // 출력합니다.
28 let output = '이름\t총점\t평균\n';
29 for (const s of students) {
30     output += s.toString();
31 }
32 console.log(output);
33 
```

[Running] node "/Users/seeker/javascript/study.js"

이름	총점	평균
구름	363점	90.75점
별이	374점	93.5점
겨울	352점	88점
바다	340점	85점

[Done] exited with code=0 in 0.426 seconds

줄 33, 열 1 공백: 4 UTF-8 LF JavaScript 포트: 3000 Prettier

A close-up photograph of a person's hands holding a white smartphone. The person is wearing a white button-down shirt. A large, semi-transparent white rectangular box is overlaid on the center-left portion of the image. Inside this box, the Korean text "고급 기능" (Advanced Features) is displayed in a bold, black, sans-serif font.

고급 기능

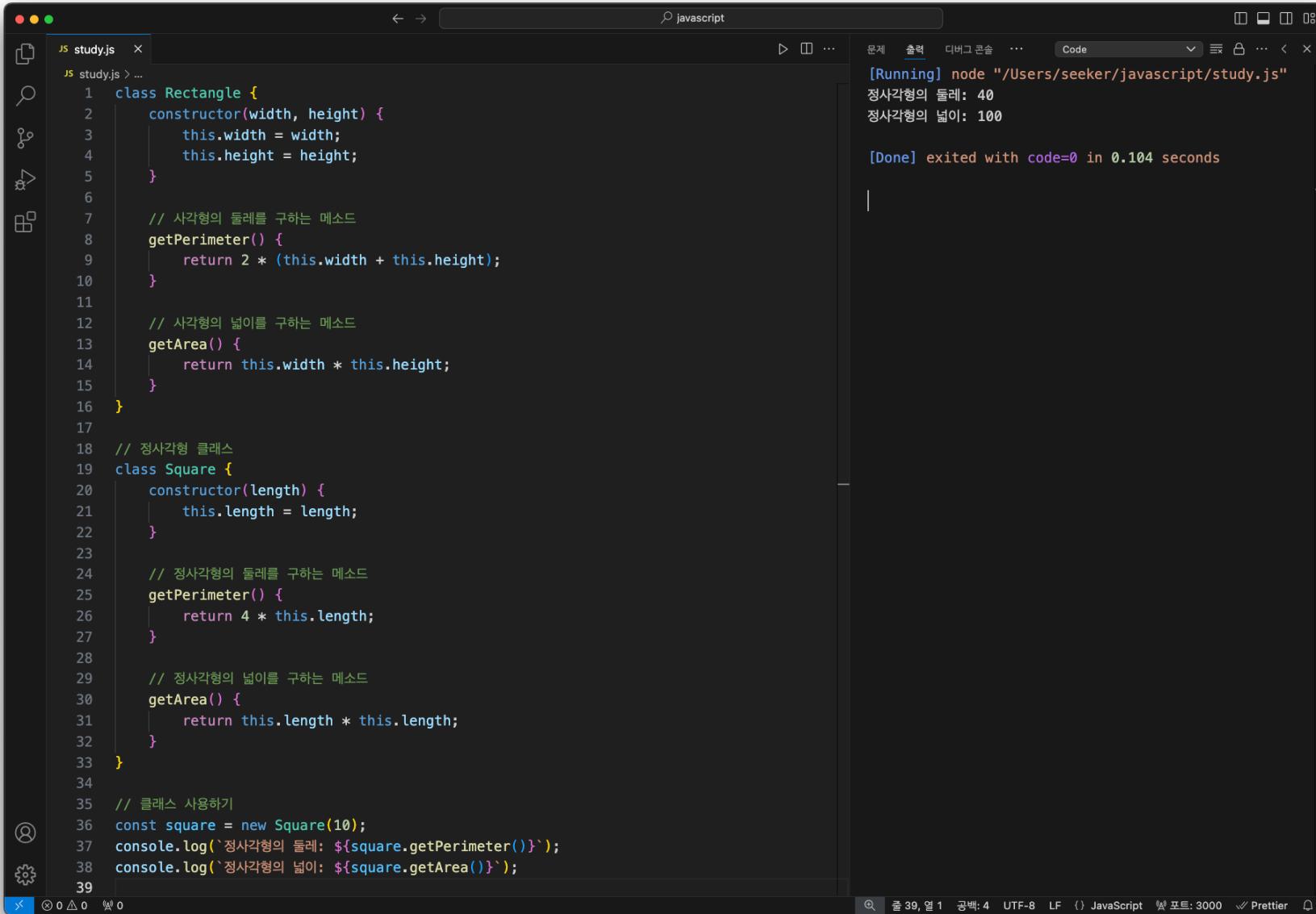
The screenshot shows a terminal window with the following content:

```
study.js > ...
1  class Rectangle {
2    constructor(width, height) {
3      this.width = width;
4      this.height = height;
5    }
6
7    // 사각형의 둘레를 구하는 메소드
8    getPerimeter() {
9      return 2 * (this.width + this.height);
10   }
11
12   // 사각형의 넓이를 구하는 메소드
13   getArea() {
14     return this.width * this.height;
15   }
16 }
17
18 const rectangle = new Rectangle(10, 20);
19 console.log(`사각형의 둘레: ${rectangle.getPerimeter()}`);
20 console.log(`사각형의 넓이: ${rectangle.getArea()}`);
21
```

[Running] node "/Users/seeker/javascript/study.js"
사각형의 둘레: 60
사각형의 넓이: 200
[Done] exited with code=0 in 0.432 seconds

At the bottom of the terminal window, there is a status bar with the following information:

줄 21, 열 1 공백: 4 UTF-8 LF {} JavaScript 포트: 3000 Prettier



```
study.js > ...
1  class Rectangle {
2      constructor(width, height) {
3          this.width = width;
4          this.height = height;
5      }
6
7      // 사각형의 둘레를 구하는 메소드
8      getPerimeter() {
9          return 2 * (this.width + this.height);
10     }
11
12     // 사각형의 넓이를 구하는 메소드
13     getArea() {
14         return this.width * this.height;
15     }
16 }
17
18 // 정사각형 클래스
19 class Square {
20     constructor(length) {
21         this.length = length;
22     }
23
24     // 정사각형의 둘레를 구하는 메소드
25     getPerimeter() {
26         return 4 * this.length;
27     }
28
29     // 정사각형의 넓이를 구하는 메소드
30     getArea() {
31         return this.length * this.length;
32     }
33 }
34
35 // 클래스 사용하기
36 const square = new Square(10);
37 console.log(`정사각형의 둘레: ${square.getPerimeter()}`);
38 console.log(`정사각형의 넓이: ${square.getArea()}`);
39
```

[Running] node "/Users/seeker/javascript/study.js"
정사각형의 둘레: 40
정사각형의 넓이: 100
[Done] exited with code=0 in 0.104 seconds

```
study.js
1 // 사각형 클래스
2 class Rectangle {
3     constructor(width, height) {
4         this.width = width;
5         this.height = height;
6     }
7
8     // 사각형의 둘레를 구하는 메소드
9     getPerimeter() {
10        return 2 * (this.width + this.height);
11    }
12
13    // 사각형의 넓이를 구하는 메소드
14    getArea() {
15        return this.width * this.height;
16    }
17 }
18
19 // 정사각형 클래스
20 class Square extends Rectangle {
21     constructor(length) {
22         super(length, length);
23     }
24 }
25
26 // 클래스 사용하기
27 const square = new Square(10, 20);
28 console.log(`정사각형의 둘레: ${square.getPerimeter()}`);
29 console.log(`정사각형의 넓이: ${square.getArea()}`);

[Running] node "/Users/seeker/javascript/study.js"
정사각형의 둘레: 40
정사각형의 넓이: 100

[Done] exited with code=0 in 0.225 seconds
```

The screenshot shows a dark-themed code editor interface with a sidebar on the left containing icons for file operations, search, and settings. The main area displays a file named `study.js` with the following content:

```
study.js > ...
1 // 정사각형 클래스
2 class Square {
3     constructor(length) {
4         this.length = length;
5     }
6     getPerimeter() {
7         return 4 * this.length;
8     }
9     getArea() {
10        return this.length * this.length;
11    }
12 }
13
14 // 클래스 사용하기
15 const square = new Square(-10);
16 console.log(`정사각형의 둘레: ${square.getPerimeter()}`);
17 console.log(`정사각형의 넓이: ${square.getArea()}`);
18 
```

The right side of the interface shows the output of running the script with Node.js:

```
[Running] node "/Users/seeker/javascript/study.js"
정사각형의 둘레: -40
정사각형의 넓이: 100

[Done] exited with code=0 in 0.413 seconds
```

At the bottom, there are status indicators and a toolbar with various icons.

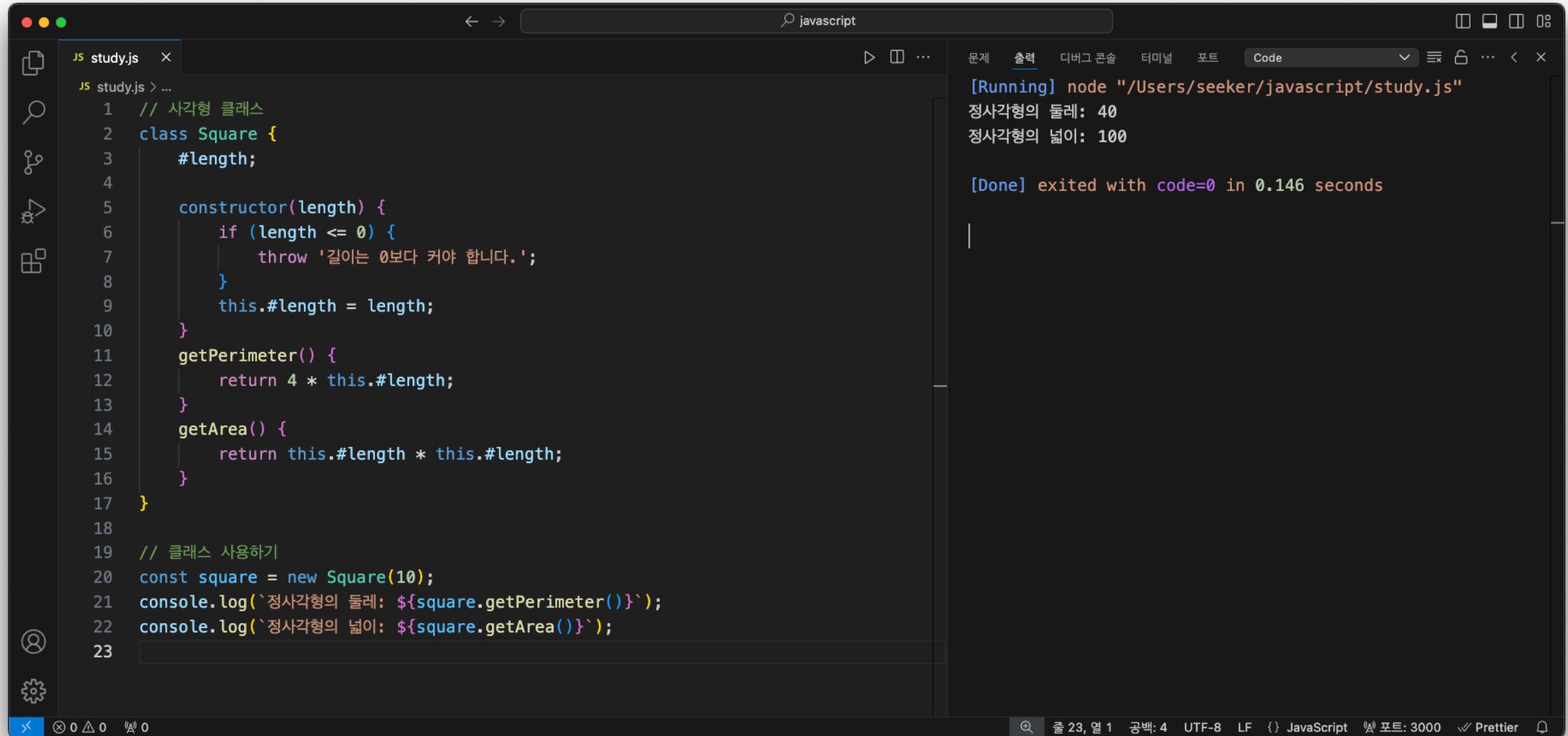
```
study.js
1 // 정사각형 클래스
2 class Square {
3     constructor(length) {
4         if (length <= 0) {
5             throw '길이는 0보다 커야 합니다.';
6         }
7         this.length = length;
8     }
9     getPerimeter() {
10    return 4 * this.length;
11 }
12 getArea() {
13    return this.length * this.length;
14 }
15 }
16
17 // 클래스 사용하기
18 const square = new Square(-10);
19 console.log(`정사각형의 둘레: ${square.getPerimeter()}`);
20 console.log(`정사각형의 넓이: ${square.getArea()}`);
21 
```

[Running] node "/Users/seeker/javascript/study.js"

/Users/seeker/javascript/study.js:5
throw '길이는 0보다 커야 합니다.';
^
길이는 0보다 커야 합니다.
(Use `node --trace-uncaught ...` to show where the exception was thrown)

Node.js v20.13.1

[Done] exited with code=1 in 0.385 seconds



```
study.js
1 // 사각형 클래스
2 class Square {
3     #length;
4
5     constructor(length) {
6         if (length <= 0) {
7             throw '길이는 0보다 커야 합니다.';
8         }
9         this.#length = length;
10    }
11    getPerimeter() {
12        return 4 * this.#length;
13    }
14    getArea() {
15        return this.#length * this.#length;
16    }
17 }
18
19 // 클래스 사용하기
20 const square = new Square(10);
21 console.log(`정사각형의 둘레: ${square.getPerimeter()}`);
22 console.log(`정사각형의 넓이: ${square.getArea()}`);
23
```

[Running] node "/Users/seeker/javascript/study.js"
정사각형의 둘레: 40
정사각형의 넓이: 100
[Done] exited with code=0 in 0.146 seconds

The screenshot shows a terminal window with the following details:

- Title Bar:** javascript
- Code Editor:** study.js (Line 1)
- Code Content:**

```
JS study.js 1 ×
JS study.js > ...
1 // 사각형 클래스
2 class Square {
3     #length;
4
5     constructor(length) {
6         if (length <= 0) {
7             throw '길이는 0보다 커야 합니다.';
8         }
9         this.#length = length;
10    }
11    getPerimeter() {
12        return 4 * this.#length;
13    }
14    getArea() {
15        return this.#length * this.#length;
16    }
17 }
18 // 클래스 사용하기
19 const square = new Square(10);
20 square.#length = -10;
21 console.log(`정사각형의 둘레: ${square.getPerimeter()}`);
22 console.log(`정사각형의 넓이: ${square.getArea()}`);
23
24
```
- Output:**

```
[Running] node "/Users/seeker/javascript/study.js"
/Users/seeker/javascript/study.js:21
square.#length = -10;
^

SyntaxError: Private field '#length' must be declared in
an enclosing class
    at wrapSafe (node:internal/modules/cjs/loader:1281:20)
    at Module._compile (node:internal/modules/cjs/
loader:1321:27)
    at Module._extensions..js (node:internal/modules/cjs/
loader:1416:10)
    at Module.load (node:internal/modules/cjs/
loader:1208:32)
    at Module._load (node:internal/modules/cjs/
loader:1024:12)
    at Function.executeUserEntryPoint [as runMain]
(node:internal/modules/run_main:174:12)
    at node:internal/main/run_main_module:28:49

Node.js v20.13.1

[Done] exited with code=1 in 0.109 seconds
```
- Bottom Status Bar:** 줄 24, 열 1 | 공백: 4 | UTF-8 | LF | JavaScript | 포트: 3000 | Prettier

getter와 setter

```
study.js > ...
1 // 정사각형 클래스
2 class Square {
3     #length;
4
5     constructor(length) {
6         this.setLength(length);
7     }
8
9     setLength(value) {
10        if (value <= 0) {
11            throw '길이는 0보다 커야 합니다.';
12        }
13        this.#length = value;
14    }
15
16    getLength() {
17        return this.#length;
18    }
19
20    getPerimeter() {
21        return 4 * this.#length;
22    }
23    getArea() {
24        return this.#length * this.#length;
25    }
26 }
27
28 // 클래스 사용하기
29 const square = new Square(10);
30 console.log(`한 변의 길이는 ${square.getLength()}입니다.`);
31
32 // 예외 발생시키기
33 square.setLength(-10);
34
```

[Running] node "/Users/seeker/javascript/study.js"
한 변의 길이는 10입니다.

/Users/seeker/javascript/study.js:11
throw '길이는 0보다 커야 합니다.';
^
길이는 0보다 커야 합니다.
(Use `node --trace-uncatched ...` to show where the exception was thrown)

Node.js v20.13.1

[Done] exited with code=1 in 0.418 seconds

getter와 setter

```
study.js > ...
1 // 정사각형 클래스
2 class Square {
3     #length;
4
5     constructor(length) {
6         this.length = length;
7     }
8
9     get length() {
10        return this.#length;
11    }
12
13    get perimeter() {
14        return this.#length * 4;
15    }
16
17    get area() {
18        return this.#length * this.#length;
19    }
20
21    set length(length) {
22        if (length <= 0) {
23            throw '길이는 0보다 커야 합니다.';
24        }
25        this.#length = length;
26    }
27
28 // 클래스 사용하기
29 const squareA = new Square(10);
30 console.log(`한 변의 길이: ${squareA.length}`);
31 console.log(`둘레: ${squareA.perimeter}`);
32 console.log(`넓이: ${squareA.area}`);
33
34 // 예외 발생시키기
35 const squareB = new Square(-10);
36
```

[Running] node "/Users/seeker/javascript/study.js"

한 변의 길이: 10
둘레: 40
넓이: 100

/Users/seeker/javascript/study.js:22
throw '길이는 0보다 커야 합니다.';
^
길이는 0보다 커야 합니다.
(Use `node --trace-uncatched ...` to show where the exception was thrown)

Node.js v20.13.1

[Done] exited with code=1 in 0.103 seconds

```
study.js
1  class Square {
2      #length;
3      static #conuter = 0;
4      static get counter() {
5          return Square.#conuter;
6      }
7      constructor(length) {
8          this.length = length;
9          Square.#conuter += 1;
10     }
11
12     static perimeterOf(length) {
13         return length * 4;
14     }
15     static areaOf(length) {
16         return length * length;
17     }
18
19     get length() {
20         return this.#length;
21     }
22     get perimeter() {
23         return this.#length * 4;
24     }
25     get area() {
26         return this.#length * this.#length;
27     }
28
29     set length(length) {
30         if (length < 10) {
31             throw '길이는 0보다 커야 합니다.';
32         }
33         this.#length = length;
34     }
35 }
36
37 // static 속성 사용하기
38 const squareA = new Square(10);
39 const squareB = new Square(20);
40 const squareC = new Square(30);
41 console.log(`지금까지 생성된 Square 인스턴스는 ${Square.counter}개입니다.`);
42 // static 메소드 사용하기
43 console.log(`한 변의 길이가 20인 정사각형의 둘레는 ${Square.perimeterOf(20)}입니다.`);
44 console.log(`한 변의 길이가 30인 정사각형의 둘레는 ${Square.areaOf(30)}입니다.`);

[Running] node "/Users/seeker/javascript/study.js"
지금까지 생성된 Square 인스턴스는 3개입니다.
한 변의 길이가 20인 정사각형의 둘레는 80입니다.
한 변의 길이가 30인 정사각형의 둘레는 900입니다.

[Done] exited with code=0 in 0.092 seconds
```

```
study.js > ...
1 // 클래스를 선언합니다.
2 class LifeCycle {
3     call() {
4         this.a();
5         this.b();
6         this.c();
7     }
8     a() {
9         console.log('a() 메소드를 호출합니다.');
10    }
11    b() {
12        console.log('b() 메소드를 호출합니다.');
13    }
14    c() {
15        console.log('c() 메소드를 호출합니다.');
16    }
17 }
18
19 // 인스턴스를 생성합니다.
20 new LifeCycle().call();
21
```

[Running] node "/Users/seeker/javascript/study.js"
a() 메소드를 호출합니다.
b() 메소드를 호출합니다.
c() 메소드를 호출합니다.

[Done] exited with code=0 in 0.094 seconds

```
study.js > ...
1 // 클래스를 선언합니다.
2 class LifeCycle {
3     call() {
4         this.a();
5         this.b();
6         this.c();
7     }
8
9     a() {
10        console.log('a() 메소드를 호출합니다.');
11    }
12    b() {
13        console.log('b() 메소드를 호출합니다.');
14    }
15    c() {
16        console.log('c() 메소드를 호출합니다.');
17    }
18 }
19
20 class Child extends LifeCycle {
21     a() {
22        console.log('자식의 a() 메소드입니다.');
23    }
24 }
25
26 // 인스턴스를 생성합니다.
27 new Child().call();
28
```

[Running] node "/Users/seeker/javascript/study.js"

자식의 a() 메소드입니다.

b() 메소드를 호출합니다.

c() 메소드를 호출합니다.

[Done] exited with code=0 in 0.09 seconds

```
study.js > ...
1 // 클래스를 선언합니다.
2 class LifeCycle {
3     call() {
4         this.a();
5         this.b();
6         this.c();
7     }
8
9     a() {
10        console.log('a() 메소드를 호출합니다.');
11    }
12    b() {
13        console.log('b() 메소드를 호출합니다.');
14    }
15    c() {
16        console.log('c() 메소드를 호출합니다.');
17    }
18 }
19
20 class Child extends LifeCycle {
21     a() {
22         super.a();
23         console.log('자식의 a() 메소드입니다.');
24     }
25 }
26
27 // 인스턴스를 생성합니다.
28 new Child().call();

[Running] node "/Users/seeker/javascript/study.js"
a() 메소드를 호출합니다.
자식의 a() 메소드입니다.
b() 메소드를 호출합니다.
c() 메소드를 호출합니다.

[Done] exited with code=0 in 0.155 seconds
```

```
study.js
1  class Pet {
2      constructor(name, age) {
3          this.name = name;
4          this.age = age;
5      }
6
7      toString() {
8          return `이름: ${this.name}\n나이: ${this.age}살`;
9      }
10 }
11
12 const pet = new Pet('구름', 6);
13 console.log(pet);
14 console.log(pet + '');

[Running] node "/Users/seeker/javascript/study.js"
Pet { name: '구름', age: 6 }
이름: 구름
나이: 6살

[Done] exited with code=0 in 0.105 seconds
```



감사합니다