

웹 개발자를 위한 기초

CONTENTS

목차

CHAPTER 1

웹 작동 원리

CHAPTER 2

개발환경 설정

CHAPTER 3

HTML

CHAPTER 4

주요 HTML 엘리먼트 실습

CHAPTER 5

CSS

CHAPTER 6

Flexbox로 레이아웃 구성하기

CHAPTER 7

폼 스타일링 실습

CHAPTER 8

반응형 웹 디자인

CHAPTER 9

정리 및 미리보기

CONTENTS

목차

CHAPTER 10 자바스크립트 소개

CHAPTER 11 변수와 자료형

CHAPTER 12 배열과 객체

CHAPTER 13 연산자 및 제어문

CHAPTER 14 함수와 파라미터

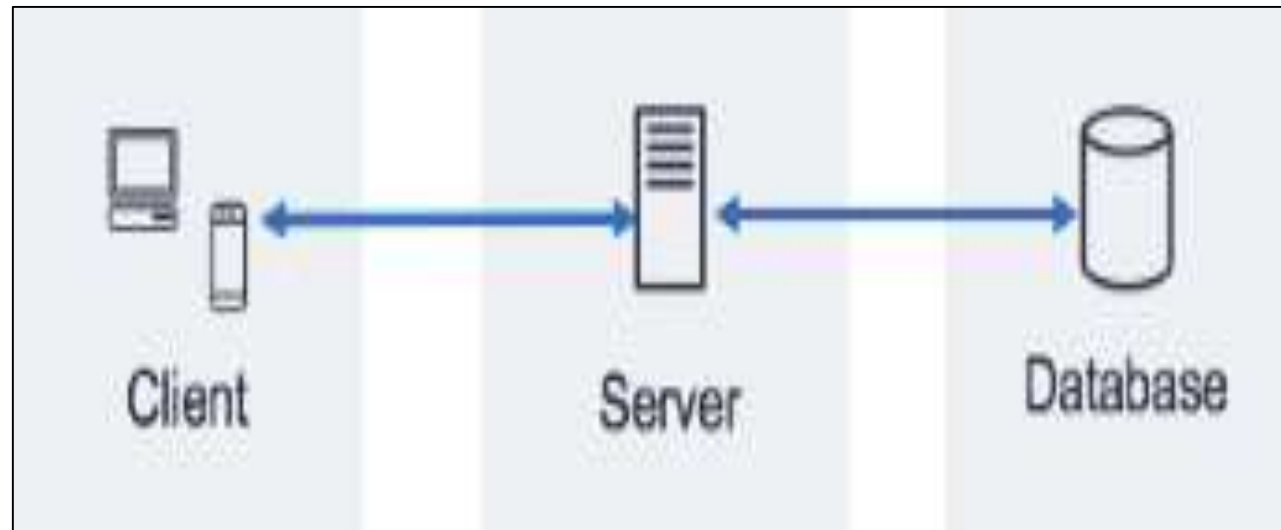
CHAPTER 15 이벤트 처리

CHAPTER 16 비동기처리

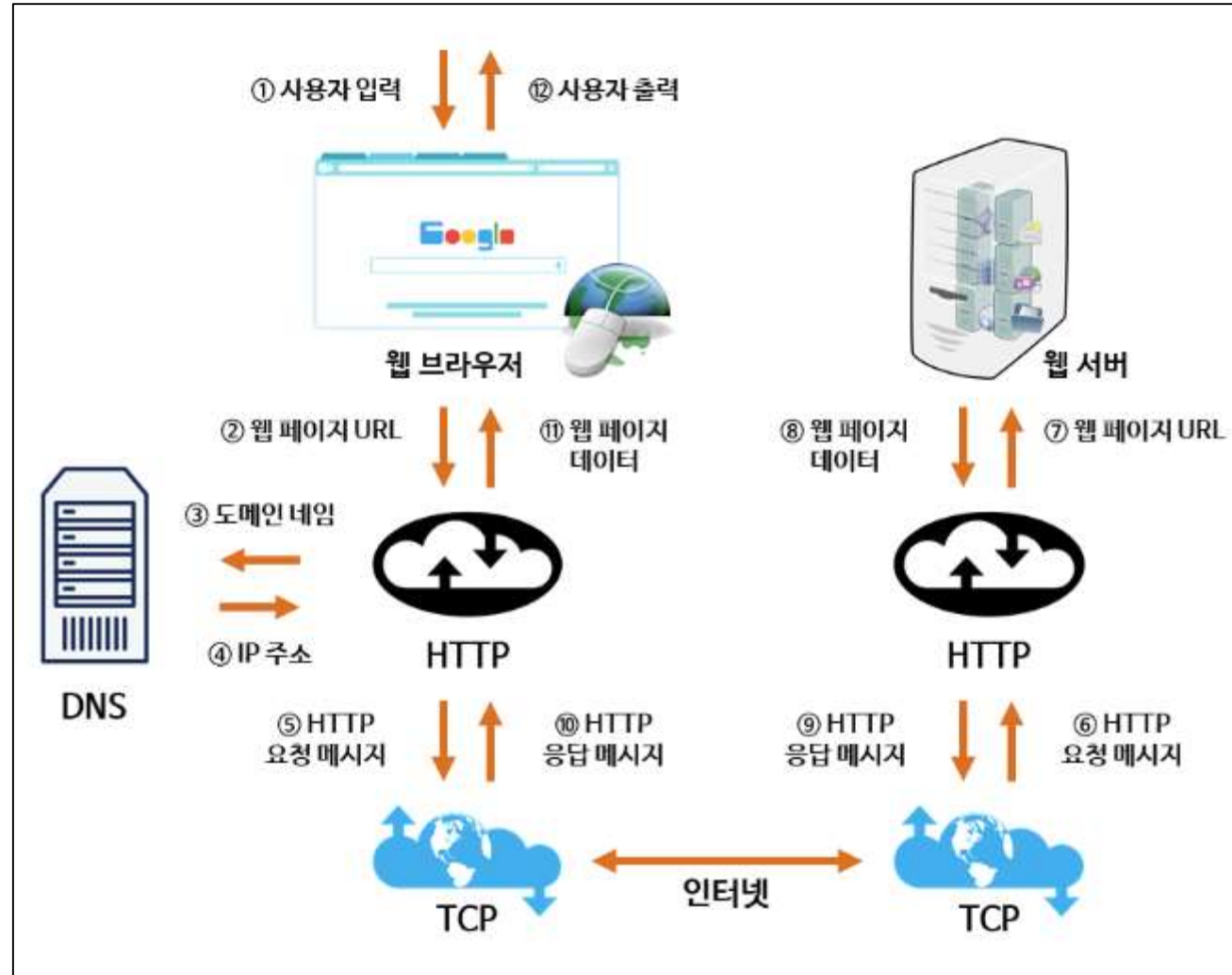
Chapter 01

웹 작동 원리

웹 개발자를 위한 기초



웹 작동 원리



Chapter 02

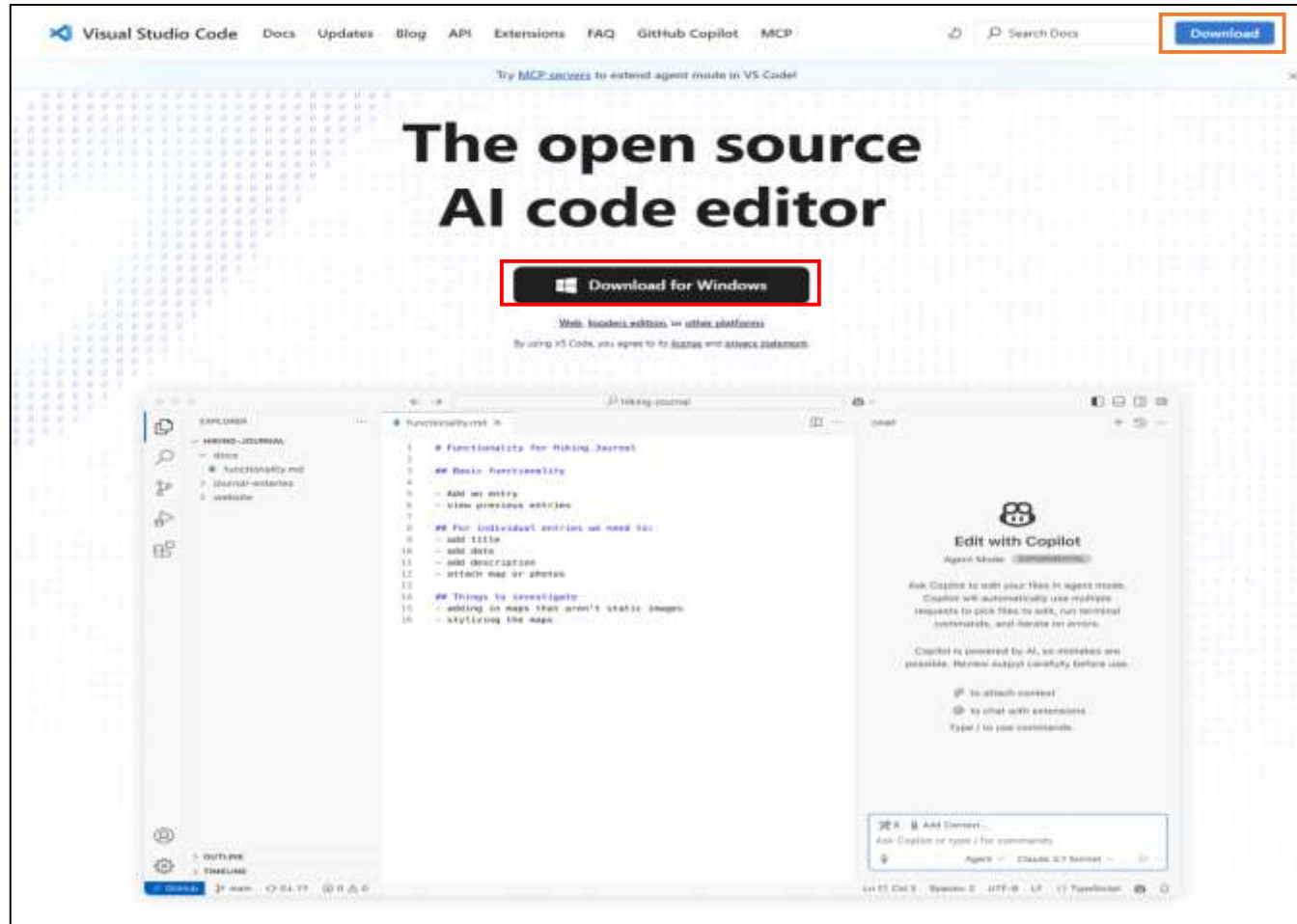
개발환경 설정

웹 개발자를 위한 기초

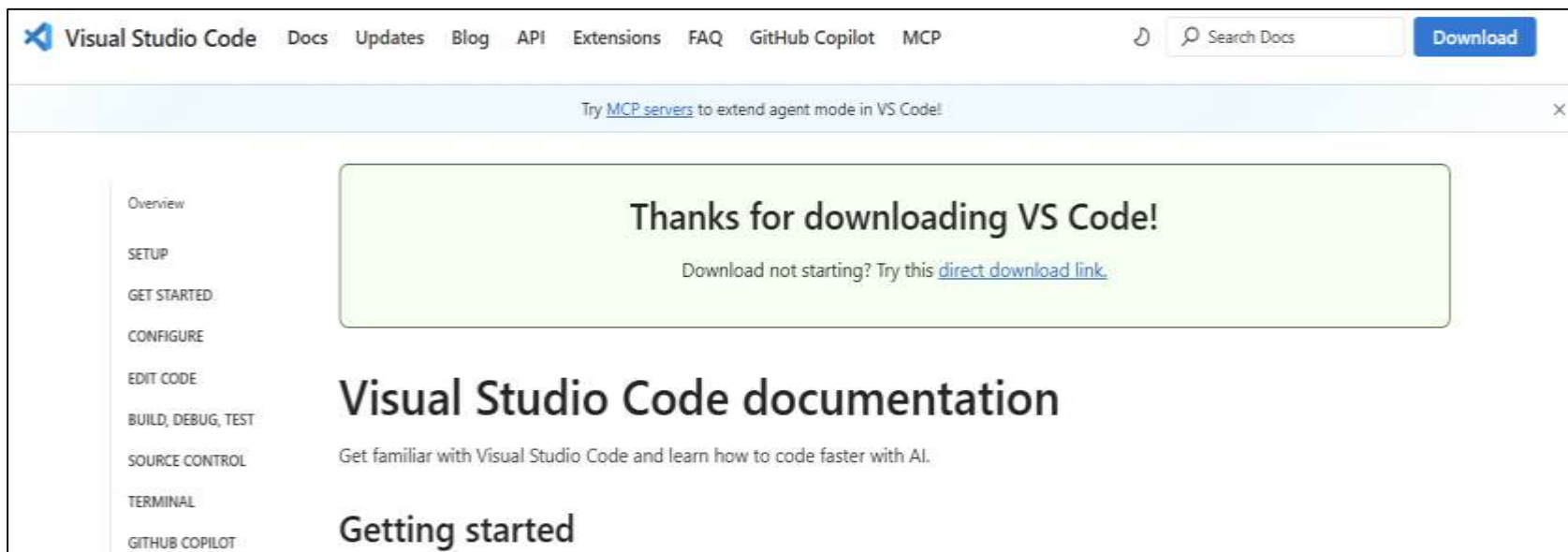
교안대로 설치하되, 화면이 다르더라도 **기본값**으로 설치 진행

개발환경 설정

- ✓ 검색엔진에 vscode 검색
- ✓ <https://code.visualstudio.com/> 방문



25.07 window기준




개발환경 설정

- ✓ 본인에게 맞는 운영체제 설치

Download Visual Studio Code


Free and built on open source. Integrated Git, debugging and extensions.



↓ Windows

Windows 10, 11

User Installer	x64	Arm64
System Installer	x64	Arm64
.zip	x64	Arm64
CLI	x64	Arm64




↓ .deb

Debian, Ubuntu

.deb	x64	Arm32	Arm64
.rpm	x64	Arm32	Arm64
.tar.gz	x64	Arm32	Arm64
Snap	Snap Store		
CLI	x64	Arm32	Arm64

↓ .rpm

Red Hat, Fedora, SUSE

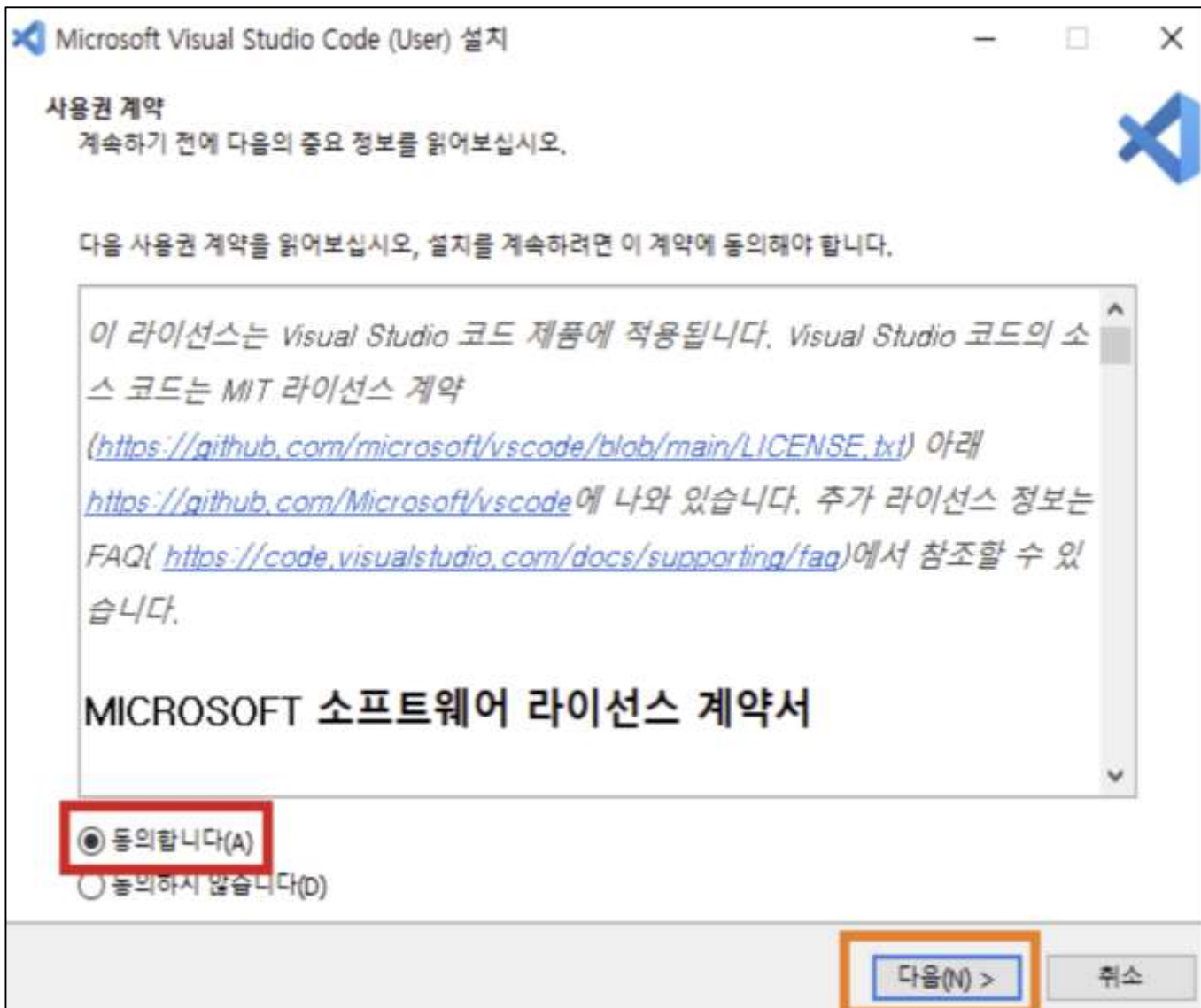


↓ Mac

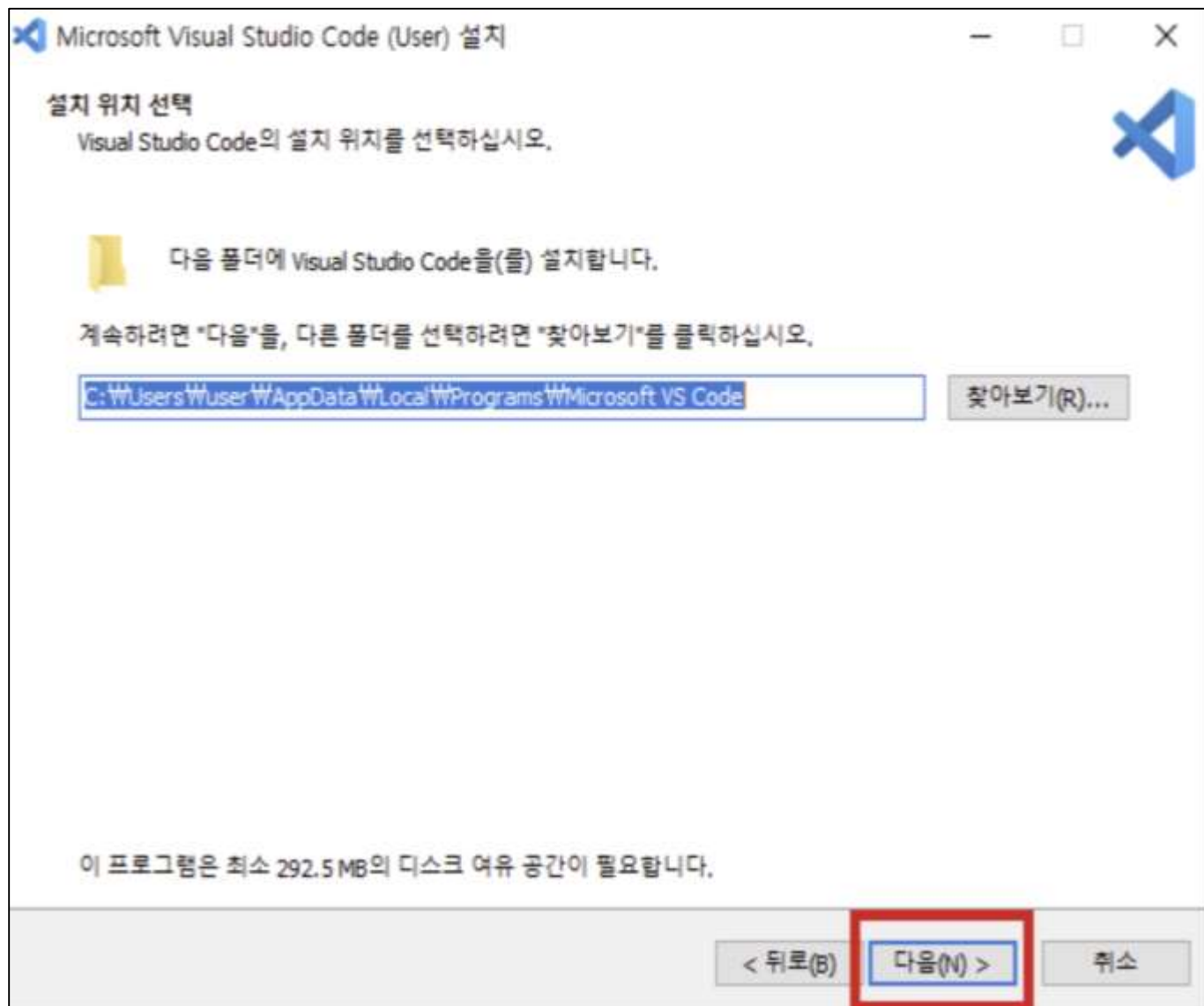
macOS 11.0+

.zip	Intel chip	Apple silicon	Universal
CLI	Intel chip	Apple silicon	

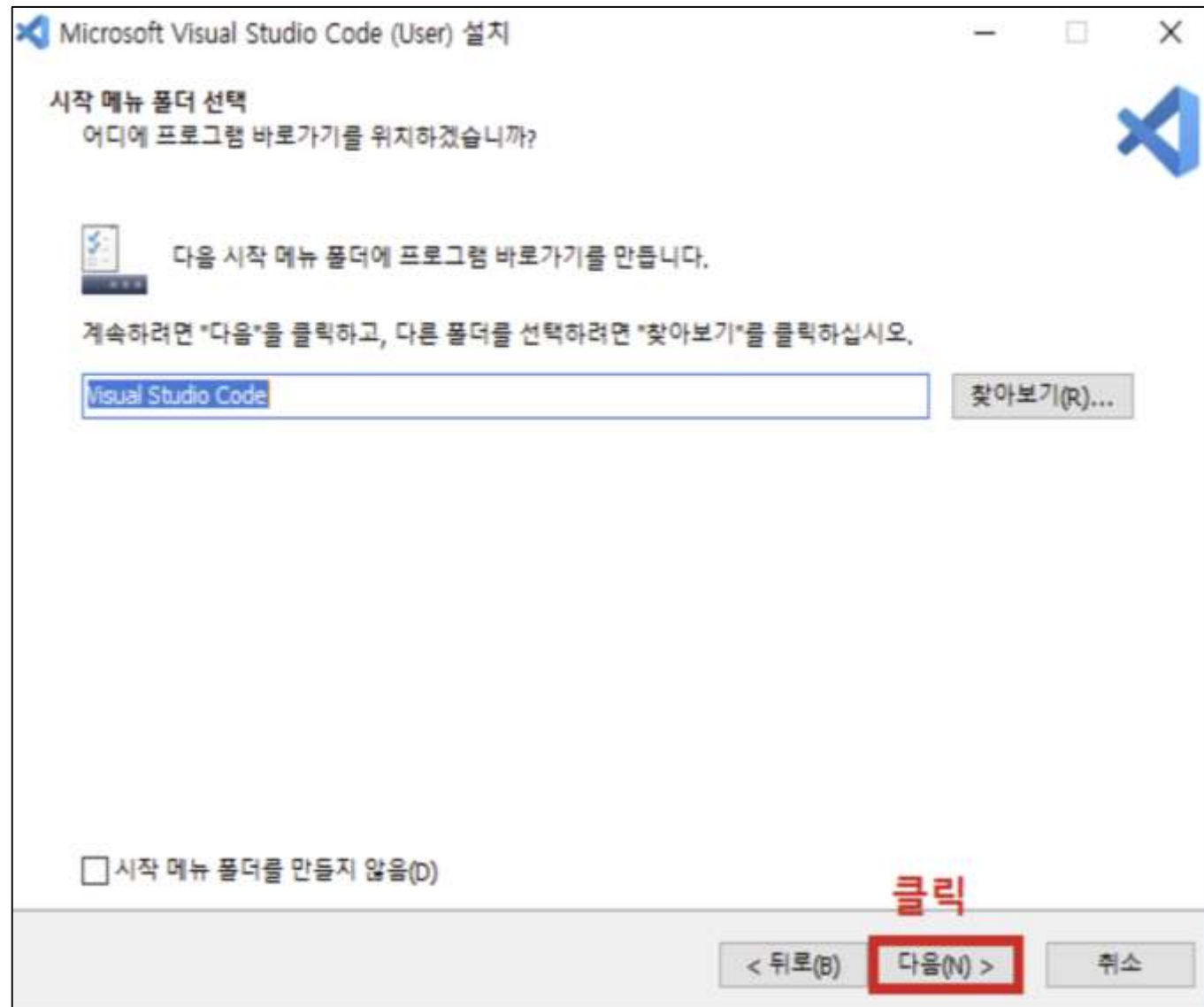
개발환경 설정



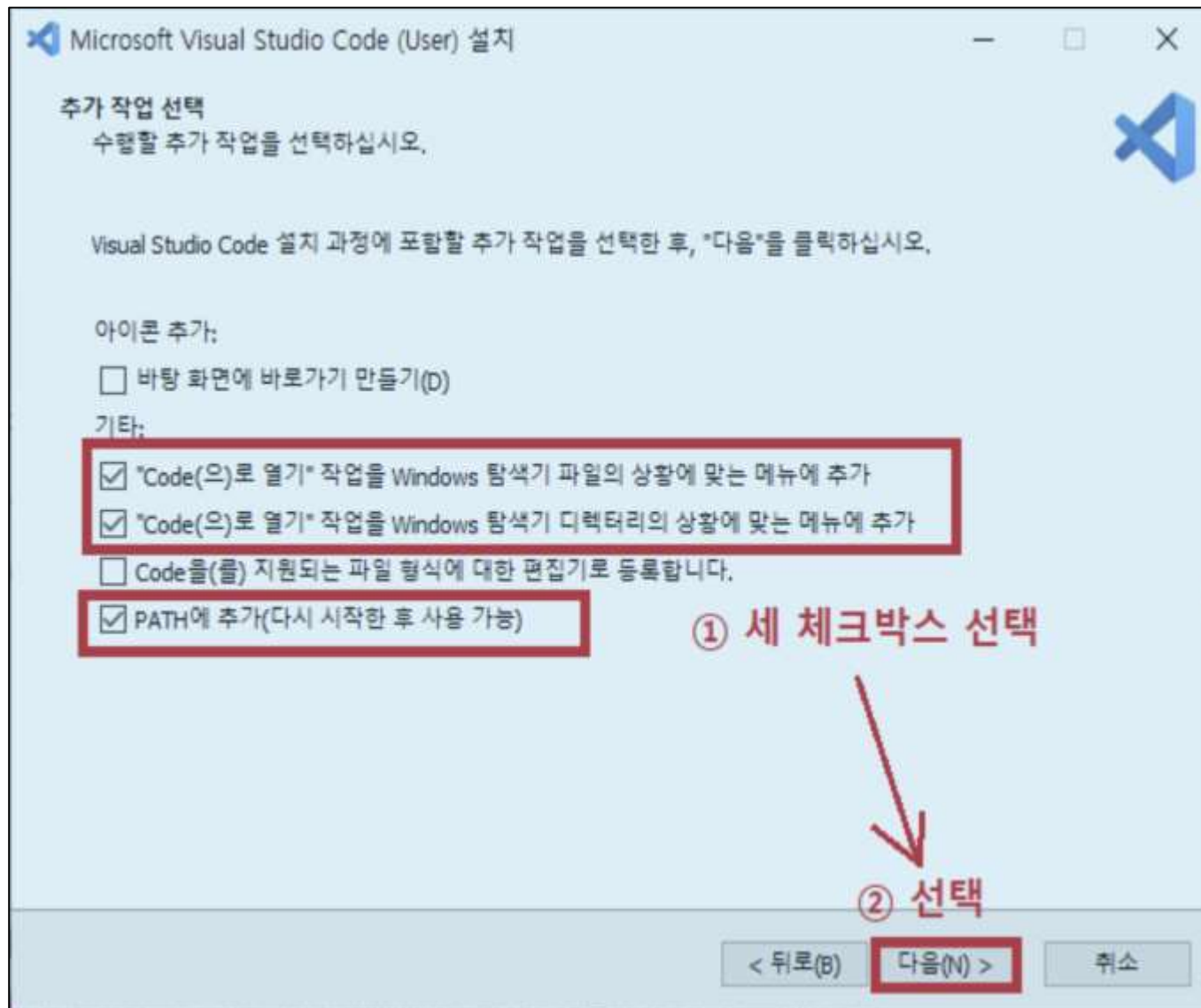
개발환경 설정



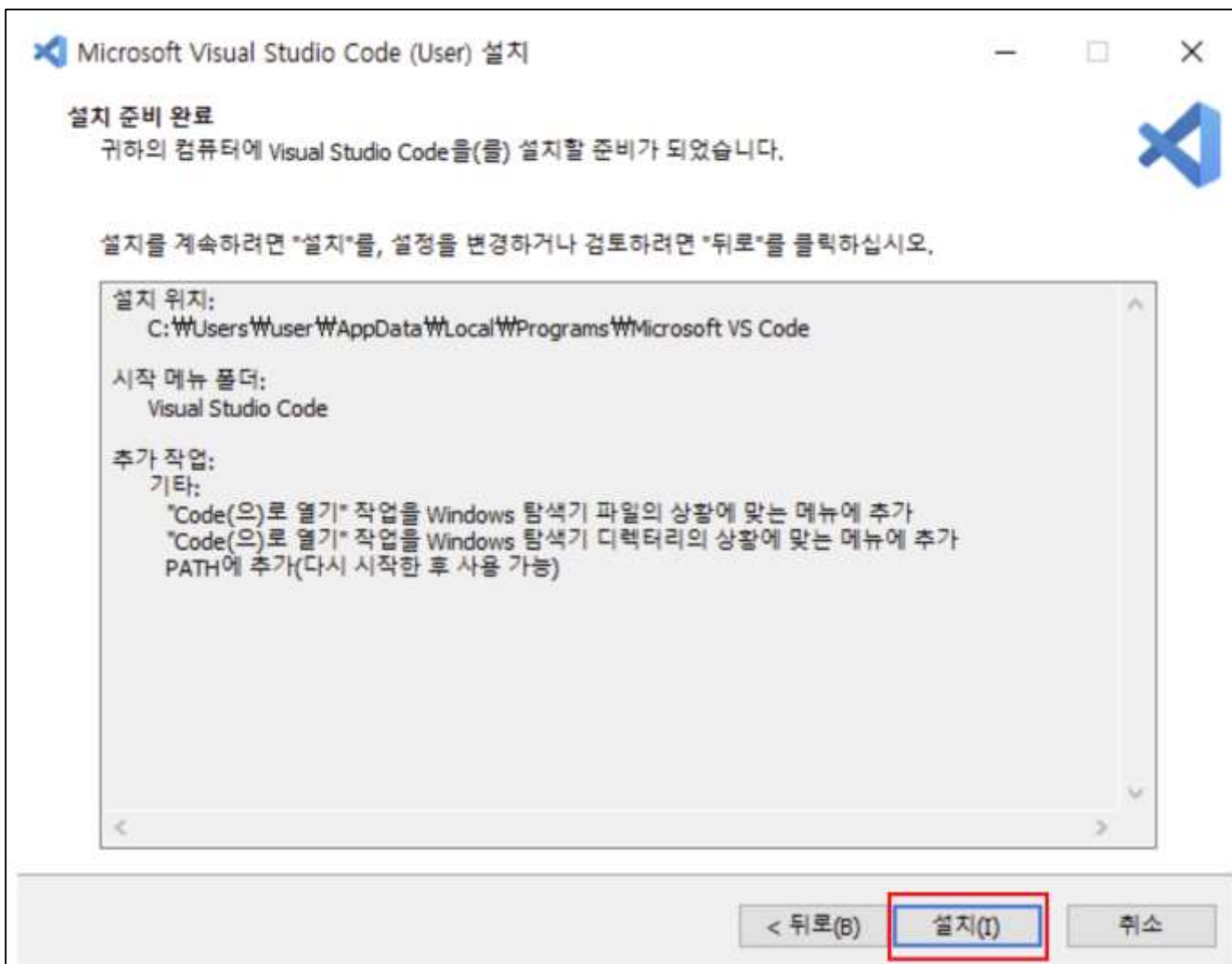
에디터 설치 & 익스텐션 설정



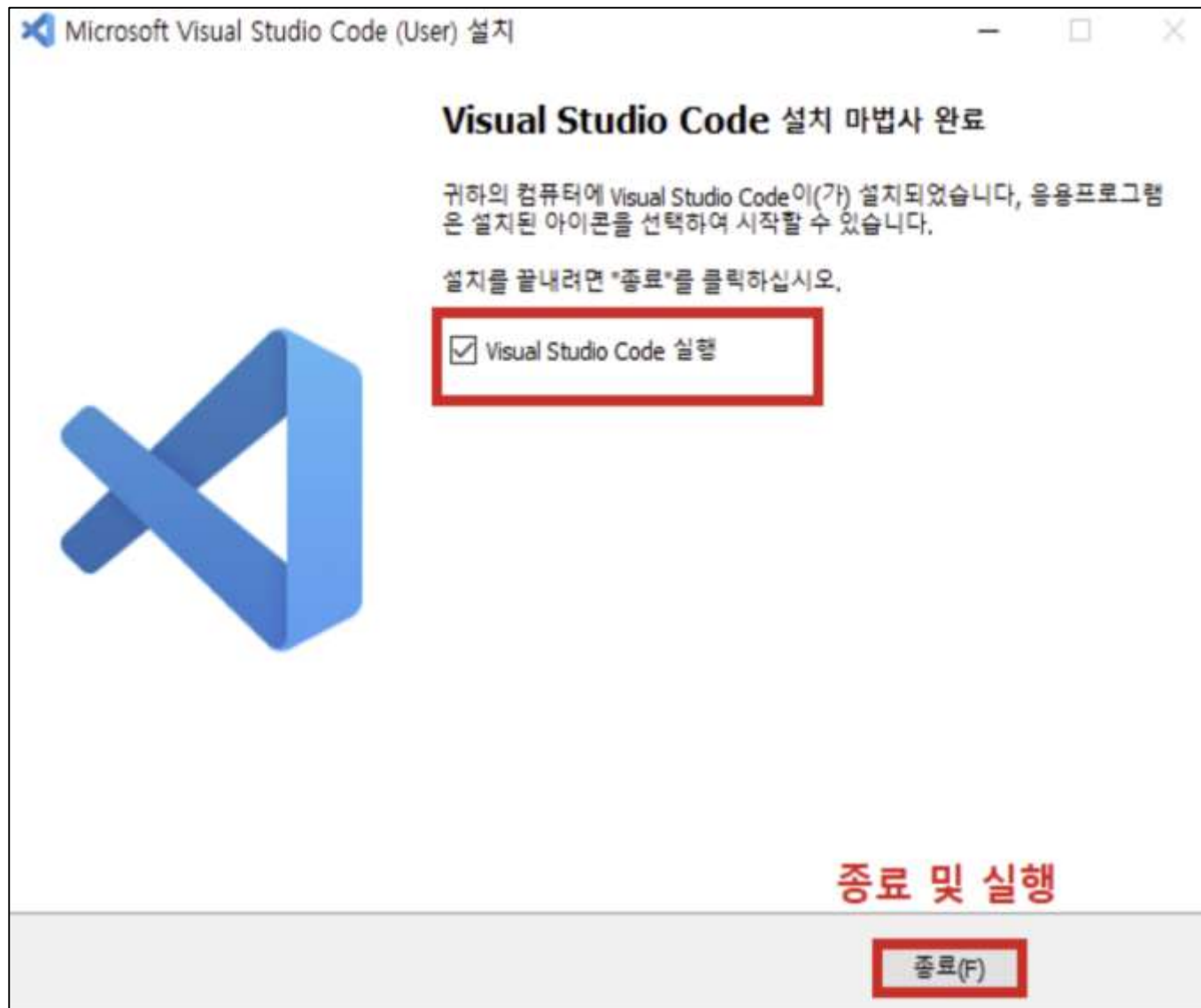
개발환경 설정



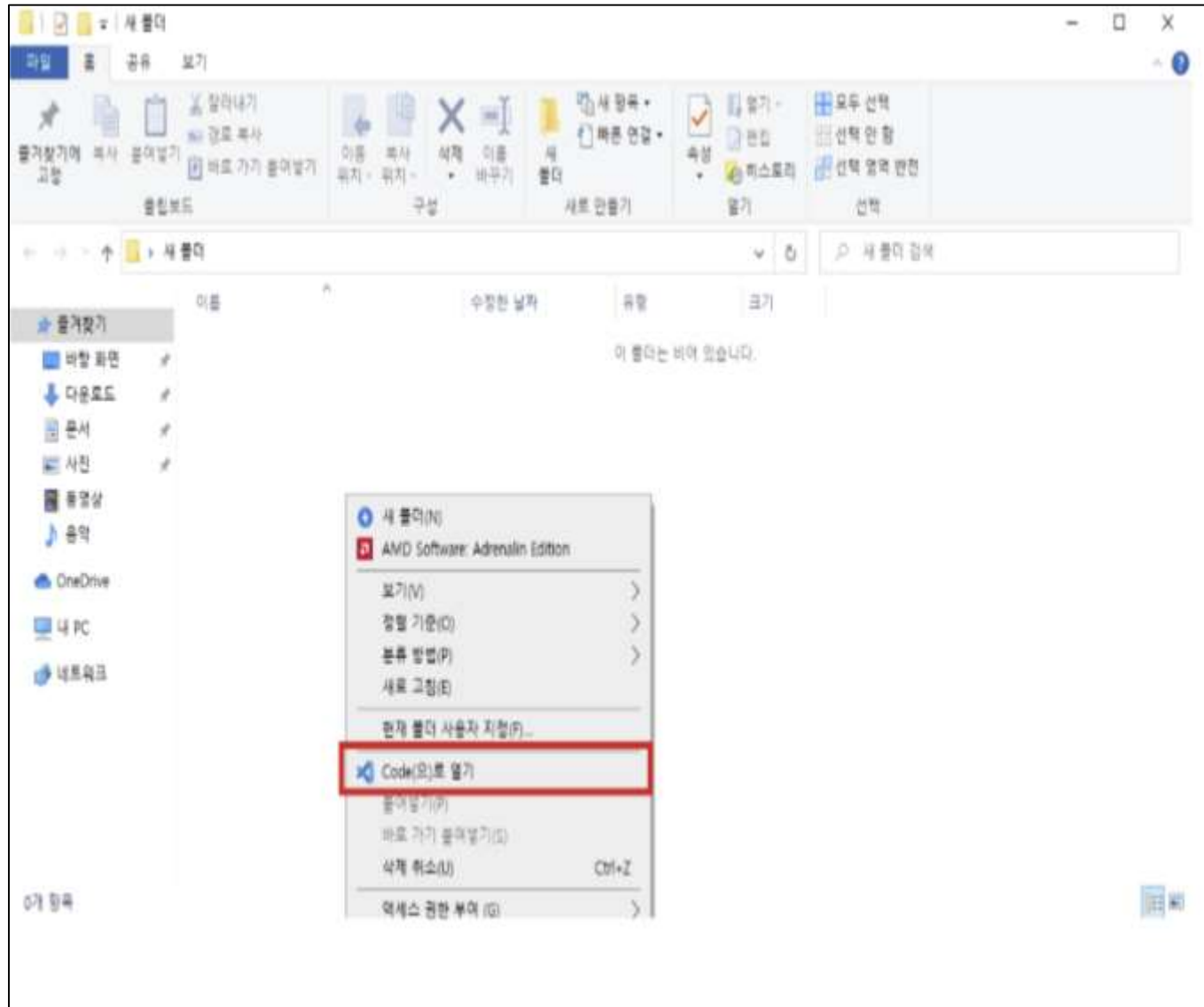
개발환경 설정



개발환경 설정



[실습] 설치 확인

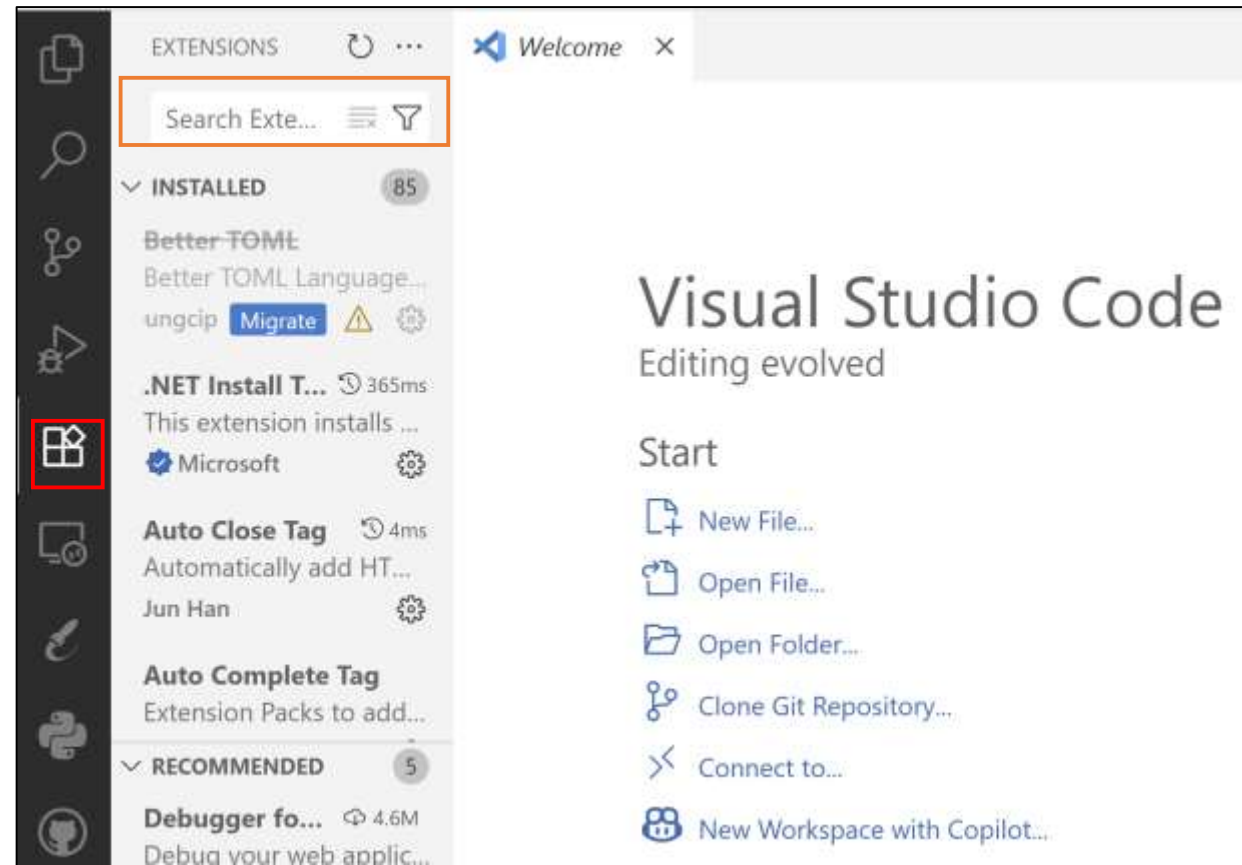


[참고] vscode 안전모드 설정

- ✓ 목적 : 악성코드 방지
- ✓ 엄격 모드
- ✓ 디버깅 사용불가
- ✓ 제약 조건 발생

확장프로그램 설치

- ✓ Auto Close Tag
- ✓ Live Server
- ✓ Auto Rename Tag
- ✓ Korean Language Pack for Visual Studio Code(선택)



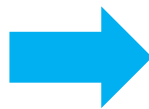
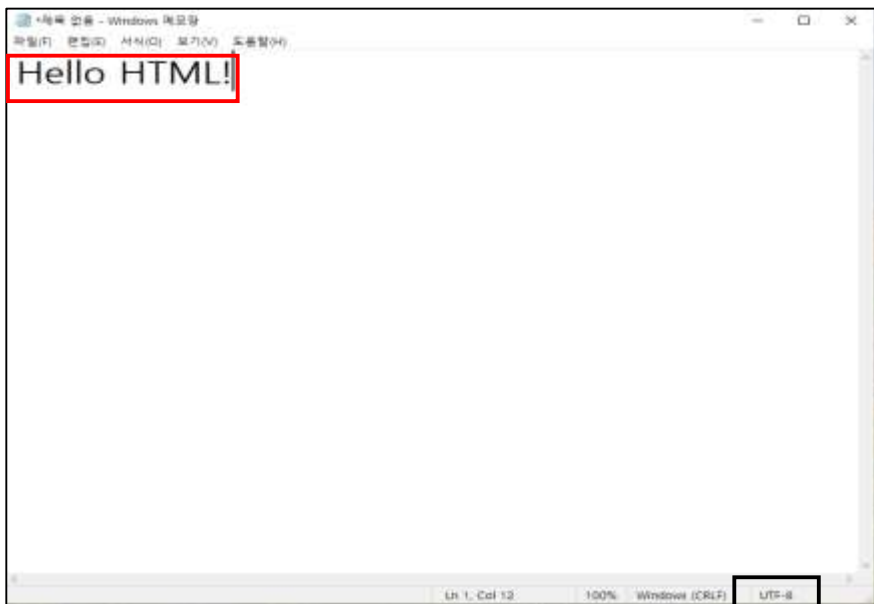
Chapter 03

HTML

웹 개발자를 위한 기초

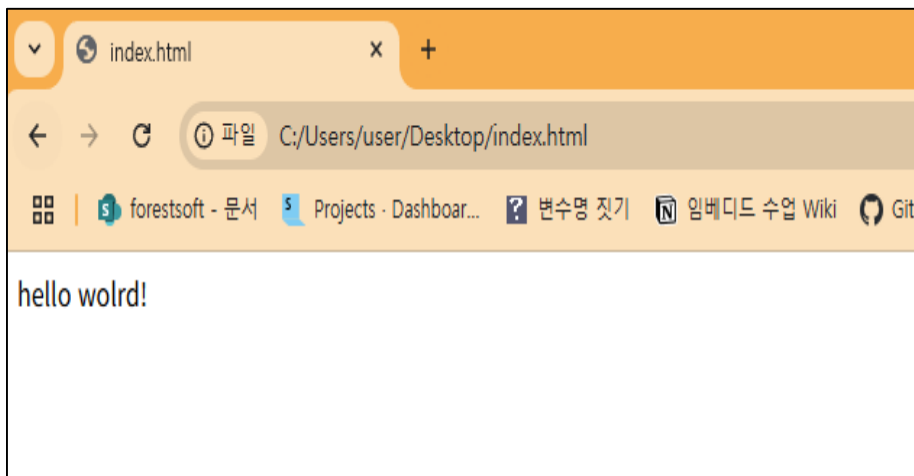
Hello HTML

✓ 메모장에 작성 후 저장하고 실행해보자.



세계 최초의 웹 페이지

✓ 첫번째 웹 페이지를 제작하였다.



http://info.cern.ch - home of the first website

From here you can:

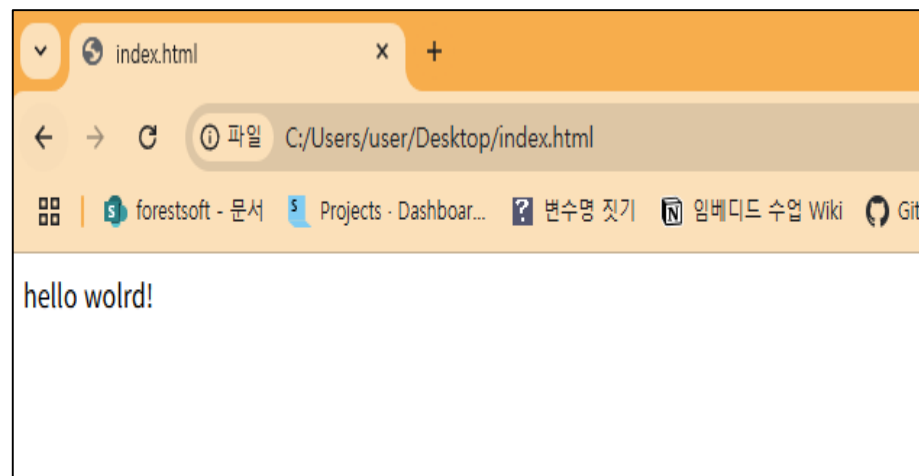
- [Browse the first website](#)
- [Browse the first website using the line-mode browser simulator](#)
- [Learn about the birth of the web](#)
- [Learn about CERN, the physics laboratory where the web was born](#)

코드 변경

✓ 코드를 변경하고, 실행해보자

 index.html

1 hello world!

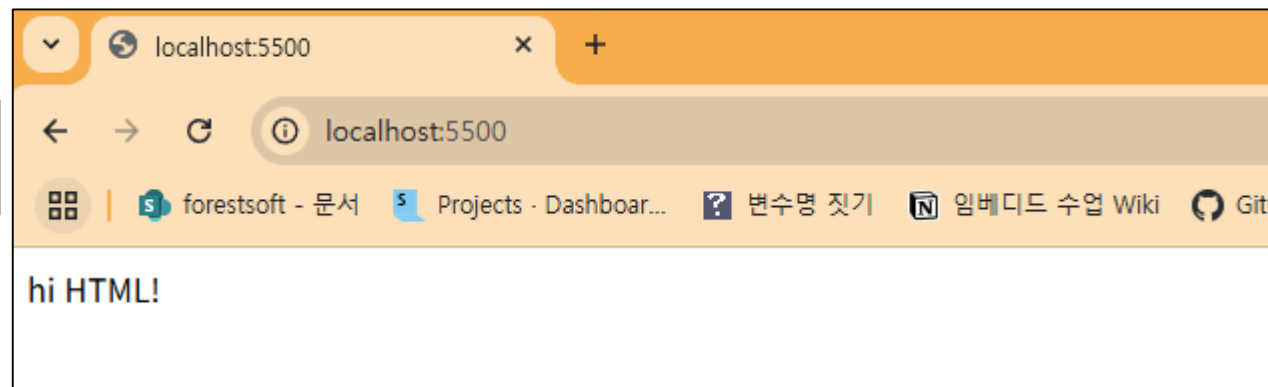


Live server 역할

- ✓ 저장하면, 자동 반영

Ln 1, Col 12 Spaces: 4 UTF-8 CRLF {} HTML  Go Live  tabnine basic  Prettier 

Ln 1, Col 13 Spaces: 4 UTF-8 CRLF {} HTML  Port: 5500  tabnine basic  Prettier 



[실습] 기본 코드 사용

✓ VSCODE에 ! 또는 html:5 입력



```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Document</title>
</head>
<body>
  |
</body>
</html>
```

[참고] HTML 기본 구조와 문서 선언

✓ <!DOCTYPE html>

<!DOCTYPE html>: HTML5 문서임을 선언

<html lang="ko"></html>: 문서의 루트, 언어 속성

<head>와 <body>: 문서 정보와 실제 콘텐츠 영역 구분

<meta>, <title>, <link>, <script>

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Document</title>
</head>
<body>
  |
</body>
</html>
```

✓ Hyper Text Markup Language

HTML이란?

[HTML](#) (Hypertext Markup Language, 하이퍼텍스트 마크업 언어)는 프로그래밍 언어는 아니고, 우리가 보는 웹페이지가 어떻게 구조화되어 있는지 브라우저로 하여금 알 수 있도록 하는 마크업 언어입니다. 이는 개발자로 하여금 복잡하게도 간단하게도 프로그래밍 할 수 있습니다. HTML은 [elements](#)로 구성되어 있으며, 이들은 적절한 방법으로 나타내고 실행하기 위해 각 콘텐츠의 여러 부분들을 감싸고 마크업 합니다. [tags](#) 는 웹 상의 다른 페이지로 이동하게 하는 하이퍼링크 내용들을 생성하거나, 단어를 강조하는 등의 역할을 합니다. 예를들어, 다음의 내용을 보시다.

HTML 소개

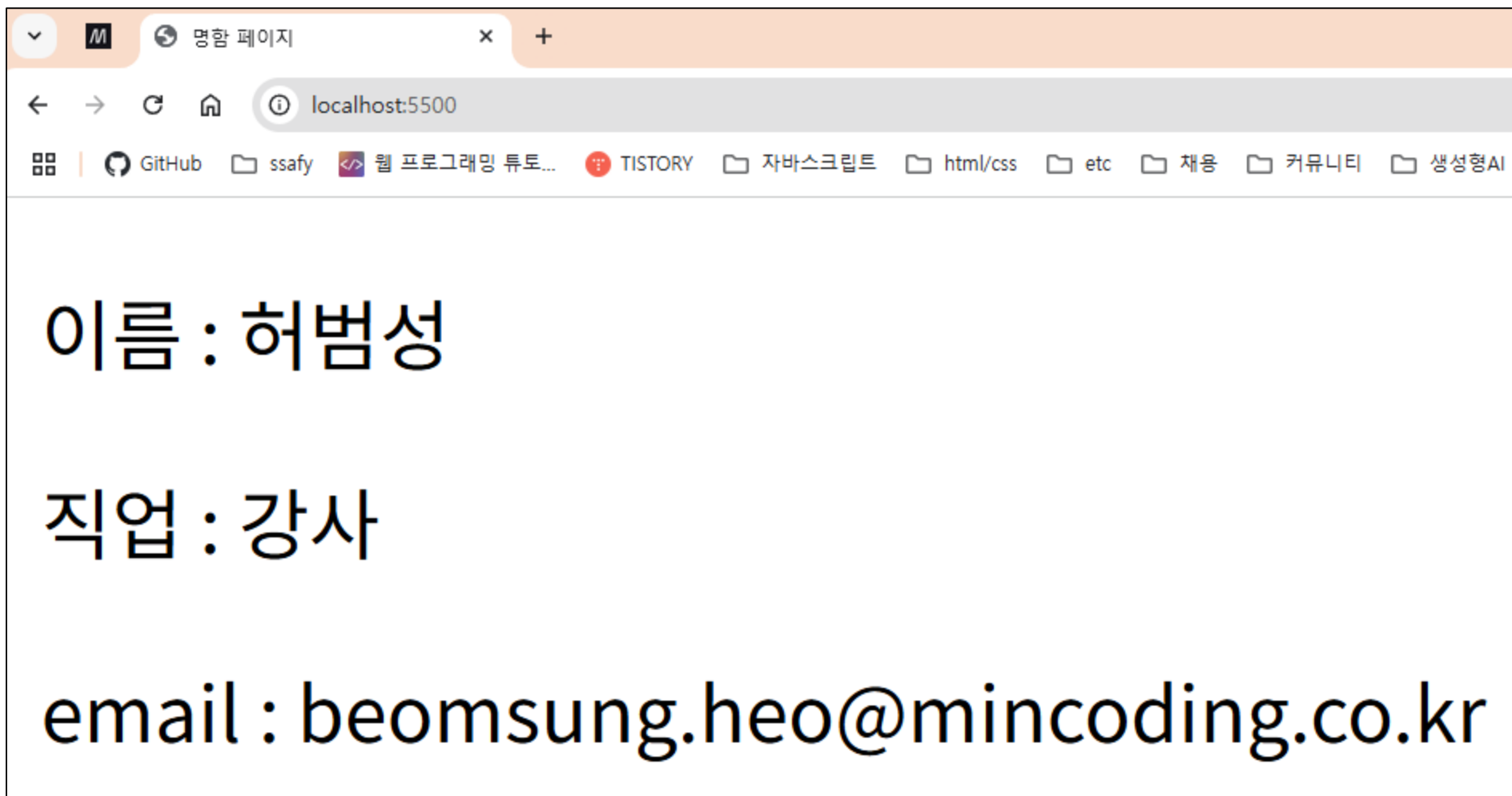
- ✓ 여는 태그
- ✓ 닫는 태그
- ✓ 내용
- ✓ 요소

Anatomy of an HTML element

The diagram illustrates the structure of an HTML element using the example `<p class="nice">Hello world!</p>`. Brackets and labels identify the following components:

- Opening tag:** A bracket above the `<p` part of the code.
- An attribute and its value:** A bracket below the `class="nice"` part of the code.
- Enclosed text content:** A bracket below the `Hello world!` text between the tags.
- Closing tag:** A bracket above the `</p>` part of the code.

[도전] 명함 만들기



Chapter 04

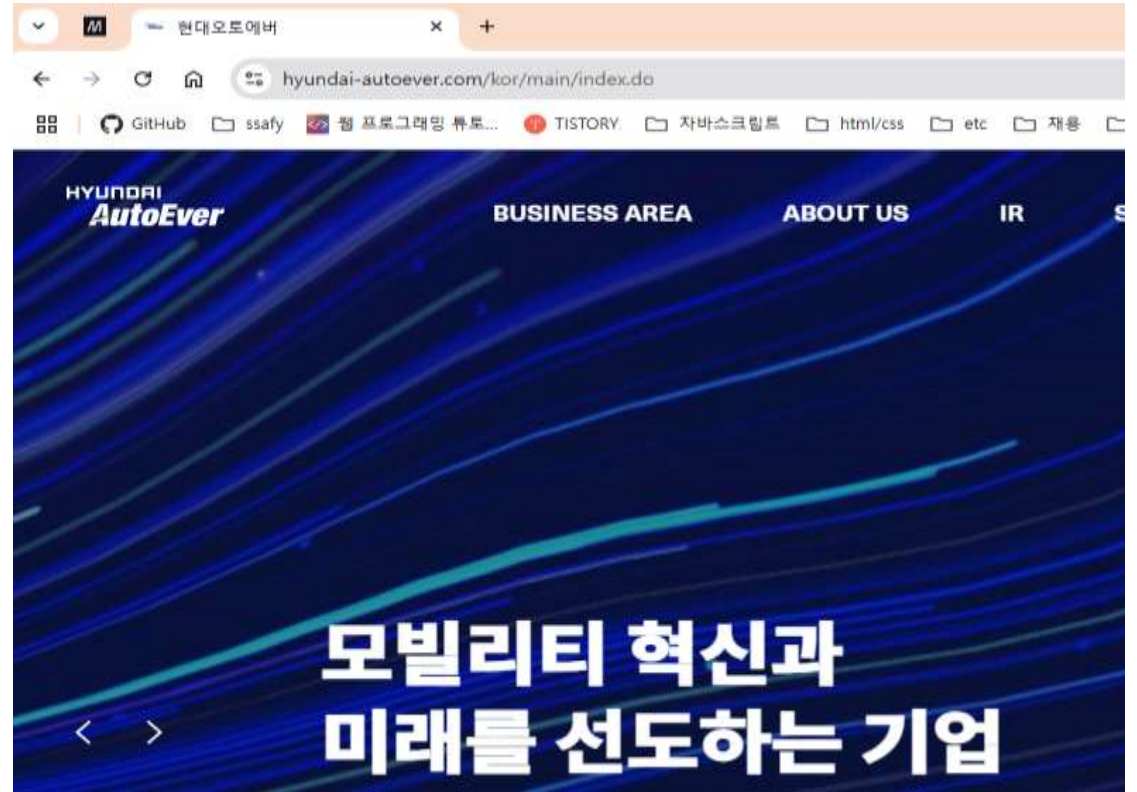
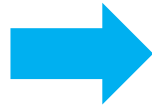
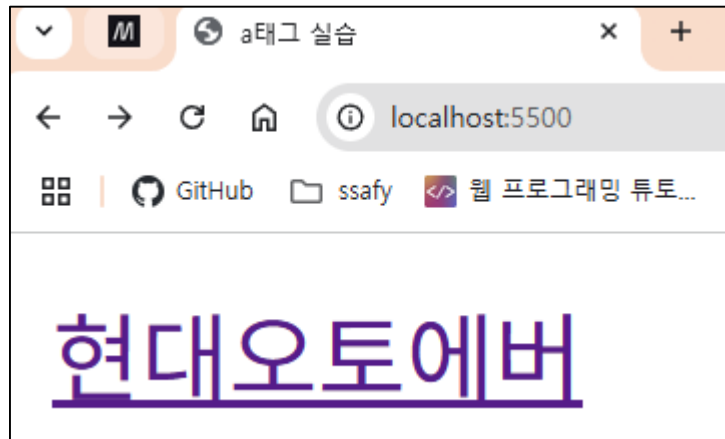
주요 element 실습

웹 개발자를 위한 기초

a태그 소개

- ✓ <a> 앵커태그
- ✓ href 속성 : 이동할 주소 작성(다른 페이지로 이동, 같은 페이지의 다른 위치 이동)
- ✓ Target : 링크 여는 창 / 탭 지정
- ✓ Title : 마우스 오버 시 툴팁 설명
- ✓ Download : 파일 다운로드 기능

[실습] a태그 사용



li 태그 소개

- ✓ 리스트 태그
- ✓ 목록의 항목을 나타냄
- ✓ 반드시 정렬(ol), 비정렬(ul), 메뉴(menu)안에 위치해야함

[실습] li태그 사용

스마트팩토리 주요 기능

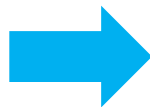
```
<ul>
  <li>실시간 생산 모니터링</li>
  <li>설비 자동화</li>
  <li>데이터 분석 및 예측</li>
  <li>품질 관리</li>
</ul>
```

스마트팩토리 점검 절차

```
<ol>
  <li>설비 전원 확인</li>
  <li>센서 상태 점검</li>
  <li>시운전 실시</li>
  <li>이상 유무 기록</li>
</ol>
```

스마트팩토리 대시보드 메뉴

```
<menu>
  <li>생산 현황</li>
  <li>설비 관리</li>
  <li>품질 분석</li>
  <li>알람 내역</li>
</menu>
```



스마트팩토리 주요 기능

- 실시간 생산 모니터링
- 설비 자동화
- 데이터 분석 및 예측
- 품질 관리

스마트팩토리 점검 절차

1. 설비 전원 확인
2. 센서 상태 점검
3. 시운전 실시
4. 이상 유무 기록

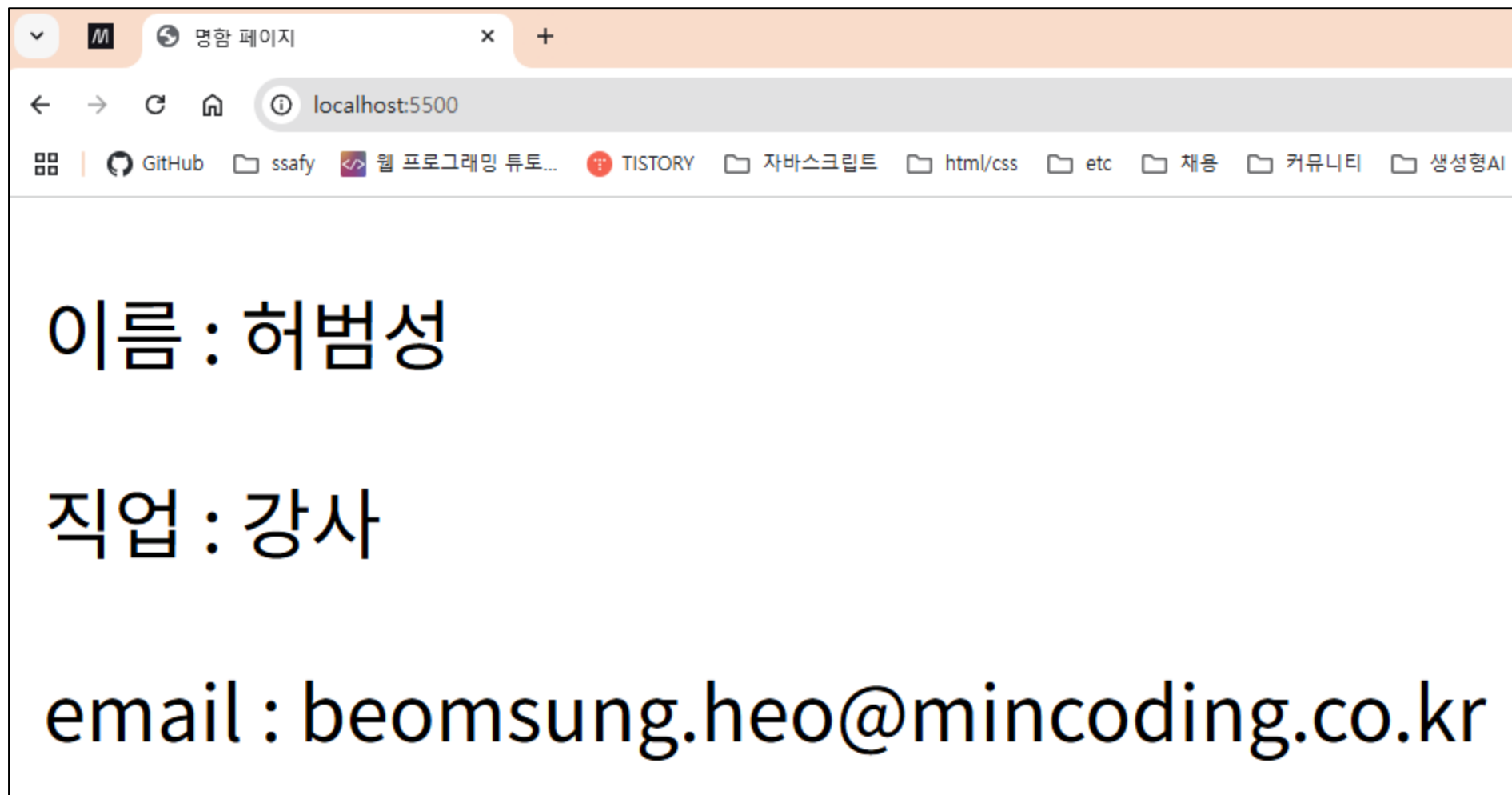
스마트팩토리 대시보드 메뉴

- 생산 현황
- 설비 관리
- 품질 분석
- 알람 내역

p태그 소개

- ✓ <p> 패러그래프 태그
- ✓ 하나의 문단을 나타냄
- ✓ 블록 레벨 요소

[리뷰] 명함 만들기



http://info.cern.ch - home of the first website

From here you can:

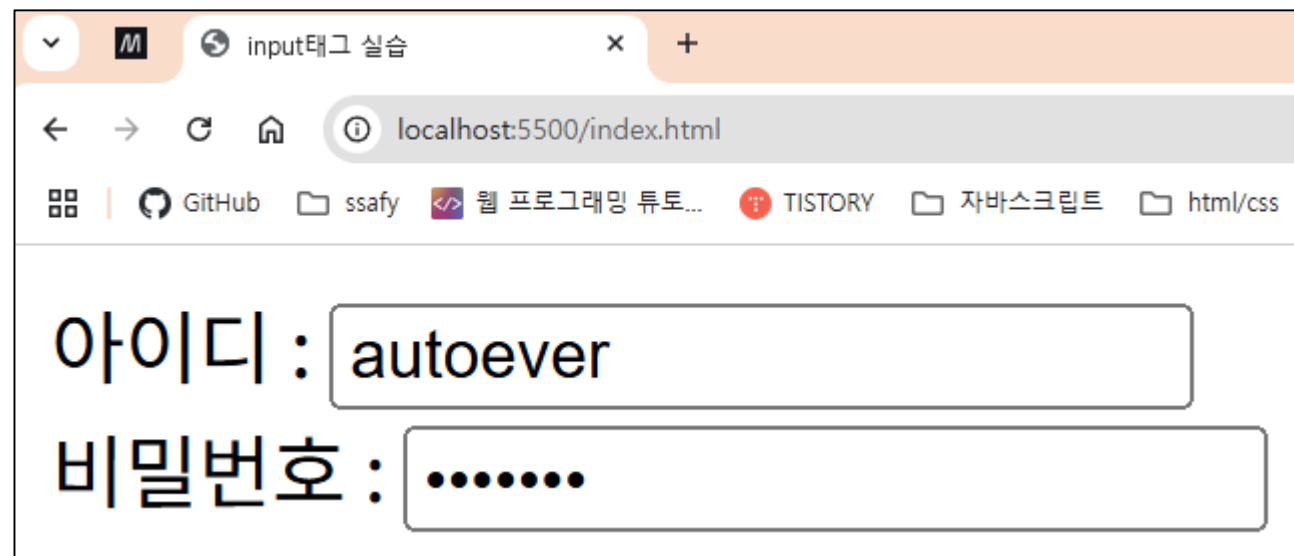
- [Browse the first website](#)
- [Browse the first website using the line-mode browser simulator](#)
- [Learn about the birth of the web](#)
- [Learn about CERN, the physics laboratory where the web was born](#)

input태그 소개

- ✓ <input> 인풋 태그
- ✓ 사용자로부터 데이터를 입력받음
- ✓ 다양한 유형(type)의 입력을 받을 수 있음
- ✓ Value 속성으로 기본값을 부여

[실습] input태그 사용

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>input태그 실습</title>
</head>
<body>
  아이디 : <input type="text" name="" id="">
  <br>
  비밀번호 : <input type="password" name="" id="">
</body>
</html>
```



[실습] input태그 사용

할 수 있는 프로그래밍 언어

c<input type="checkbox" name="" id="">


c++<input type="checkbox" name="" id="">

python<input type="checkbox" name="" id="">

java<input type="checkbox" name="" id="">

javascript<input type="checkbox" name="" id="">

<input type="submit" value="제출">



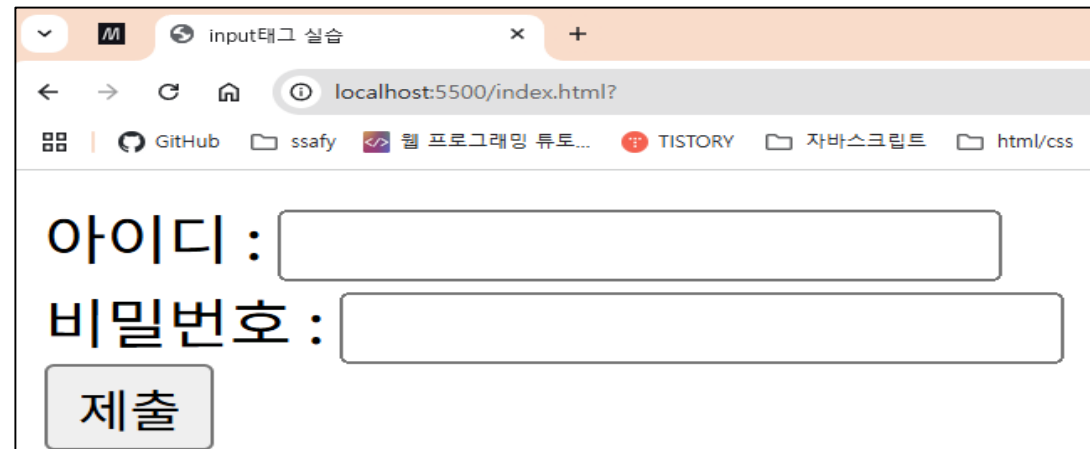
The screenshot shows a web browser window with the title "input태그 실습". The address bar shows "localhost:5500/index.html". The browser's bookmark bar includes "GitHub", "ssafy", "웹 프로그래밍 튜토...", "TISTORY", "자바스크립트", "html/css", "etc", and "채용". The main content area displays the rendered HTML form, which includes the text "할 수 있는 프로그래밍 언어" followed by five checkboxes for "c", "c++", "python", "java", and "javascript". Below these is a "제출" (Submit) button.

form태그 소개

- ✓ <form> form 태그
- ✓ 서버로 데이터를 전송하기 위한 문서 섹션

[실습] form태그 사용

```
<form action="">
  아이디 : <input type="text" name="" id="">
  <br>
  비밀번호 : <input type="password" name="" id="">
  <br>
  <input type="submit" value="제출">
</form>
```



A screenshot of a web browser window. The address bar shows 'localhost:5500/index.html?'. The browser's bookmark bar includes 'GitHub', 'ssafy', '웹 프로그래밍 튜토...', 'TISTORY', '자바스크립트', and 'html/css'. The page content displays the rendered HTML form: '아이디 :' followed by a text input field, '비밀번호 :' followed by a password input field, and a '제출' (Submit) button at the bottom.

[도전] 부품등록 양식 작성

- ✓ Form 태그 필수
- ✓ 비밀번호 암호화
- ✓ Radio 버튼은 한가지만 선택
- ✓ select는 2개 이상 작성하기

부품 등록 폼

부품명:

시리얼 번호:

관리자 비밀번호:

부품 종류(한 가지만 선택)
센서 ☒ 모터 ☐ 디스플레이 ☐

제조사:

생산일:

색상:

매뉴얼 파일 업로드: 선택된 파일 없음

특이사항 입력
비고:

[참고] 추가 학습 자료

- ✓ Semantic web(시멘틱 웹)
- ✓ 요소 비교 (inline vs block vs inline-block)

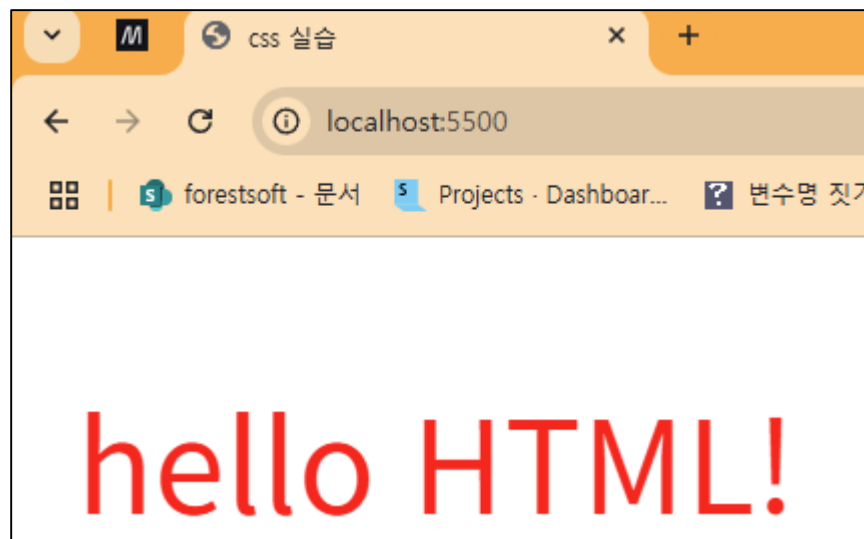
Chapter 05

CSS

웹 개발자를 위한 기초

css의 역할과 속성(property) 적용

```
<!DOCTYPE html>
<html lang="ko">
<head>
  <meta charset="UTF-8">
  <title>css 실습</title>
  <style>
    p {
      color: red;
    }
  </style>
</head>
<body>
  <p>
    hello HTML!
  </p>
</body>
</html>
```



[실습]css 적용

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Document</title>
  <style>
    p{
      color: ■ red;
    }
  </style>
  <link rel="stylesheet" href="style.css">
</head>
<body>
  <p>hello HTML!</p>
  <p>hello HTML!</p>
  <p style="color: ■ yellow;">hello HTML!</p>
</body>
</html>
```

```
p {  
  color: red;  
}
```

The diagram illustrates the components of a CSS declaration. A line points from the label "Selector" to the `p` in the selector. Brackets below the declaration identify the parts: "color" is the "Property", "red" is the "Property value", and the entire `color: red;` is the "Declaration".

선택자 종류

- ✓ Type selectors
- ✓ Class selectors
- ✓ ID selectors

Chapter 06

Flex box로 레이아웃 구성하기

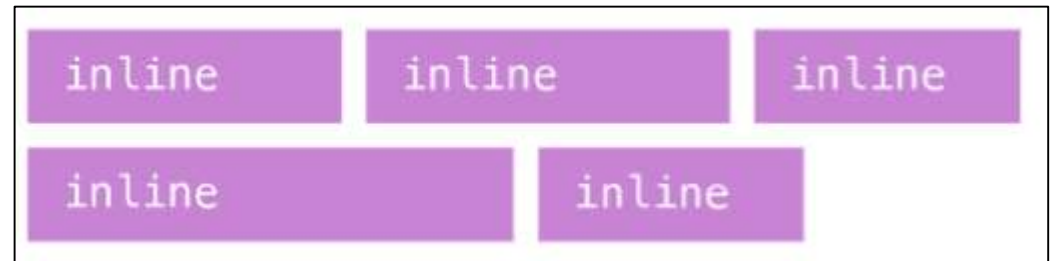
웹 개발자를 위한 기초

Layout - display property

- ✓ Inline
- ✓ Block
- ✓ Inline-block
- ✓ Flex
- ✓ Grid

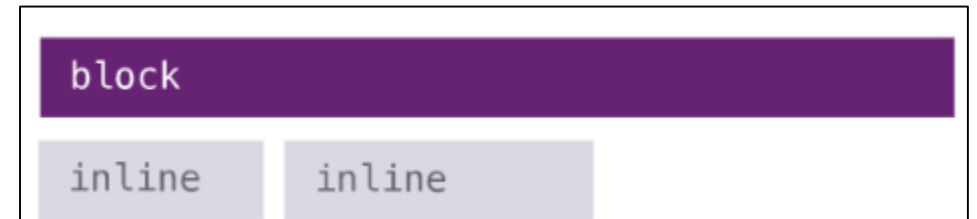
Display inline

- ✓ 줄 바꿈 없음
- ✓ 콘텐츠의 크기만큼 너비 차지(너비 조정불가)
- ✓ 종류 : a,span



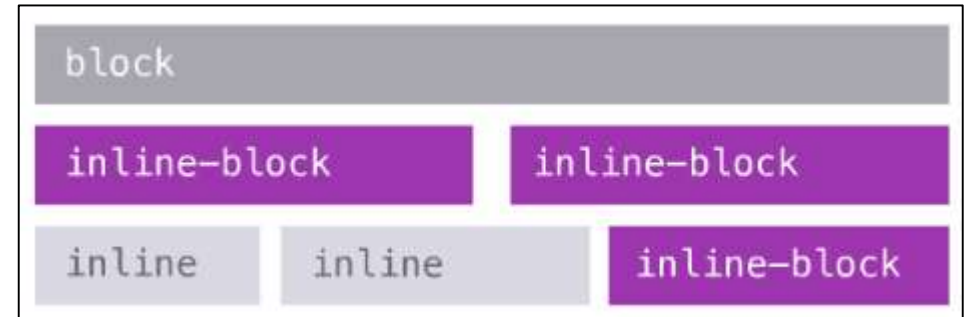
Display block

- ✓ 줄 바꿈 있음
- ✓ 콘텐츠 크기를 포함하여 한줄을 차지한다(너비 조정가능)
- ✓ 종류 : p,ul,ol,form,h1~h6,div



Display inline-block

- ✓ 줄 바꿈 없음
- ✓ 콘텐츠의 크기만큼 너비 차지(너비 조정가능)
- ✓ 종류 : input,button,img,select,textarea



Display flex

- ✓ 메인 축을 기반으로 정렬
- ✓ 한개의 Flex-container와 flex-item들로 구성
- ✓ 부모에 flex 속성을 선언하면 부모는 flex-container
- ✓ 자식 요소들은 자동으로 flex-item

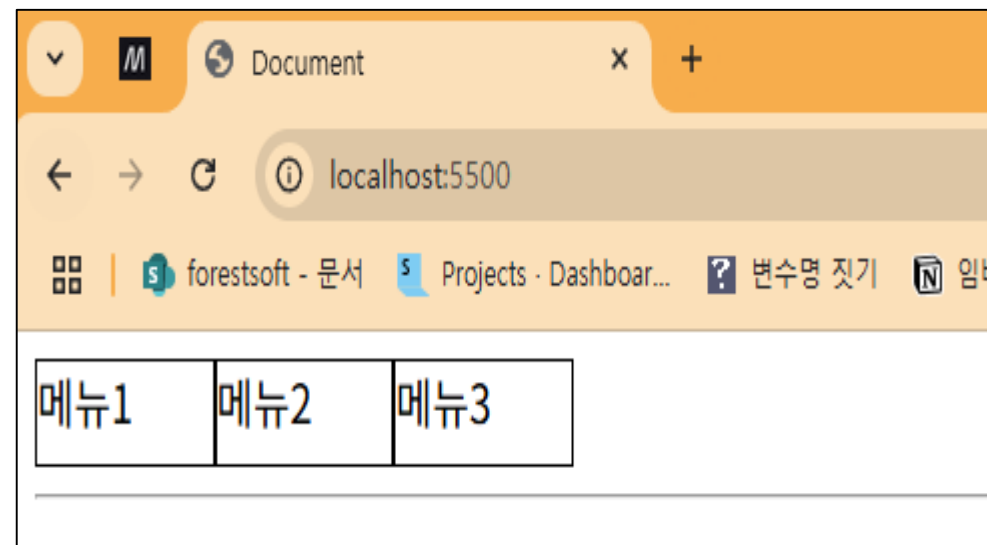
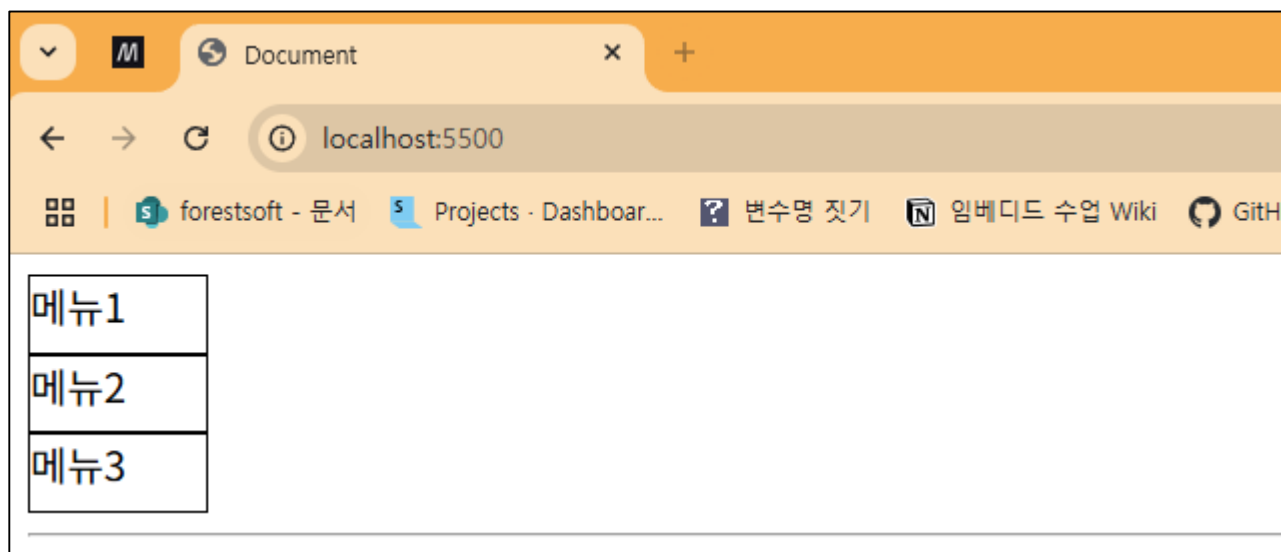
```
<div class="container">  
  <div class="item"></div>  
  <div class="item"></div>  
  <div class="item"></div>  
</div>
```

1
2
3

```
.container {  
  display: flex;  
}
```

123

[도전] flex 활용



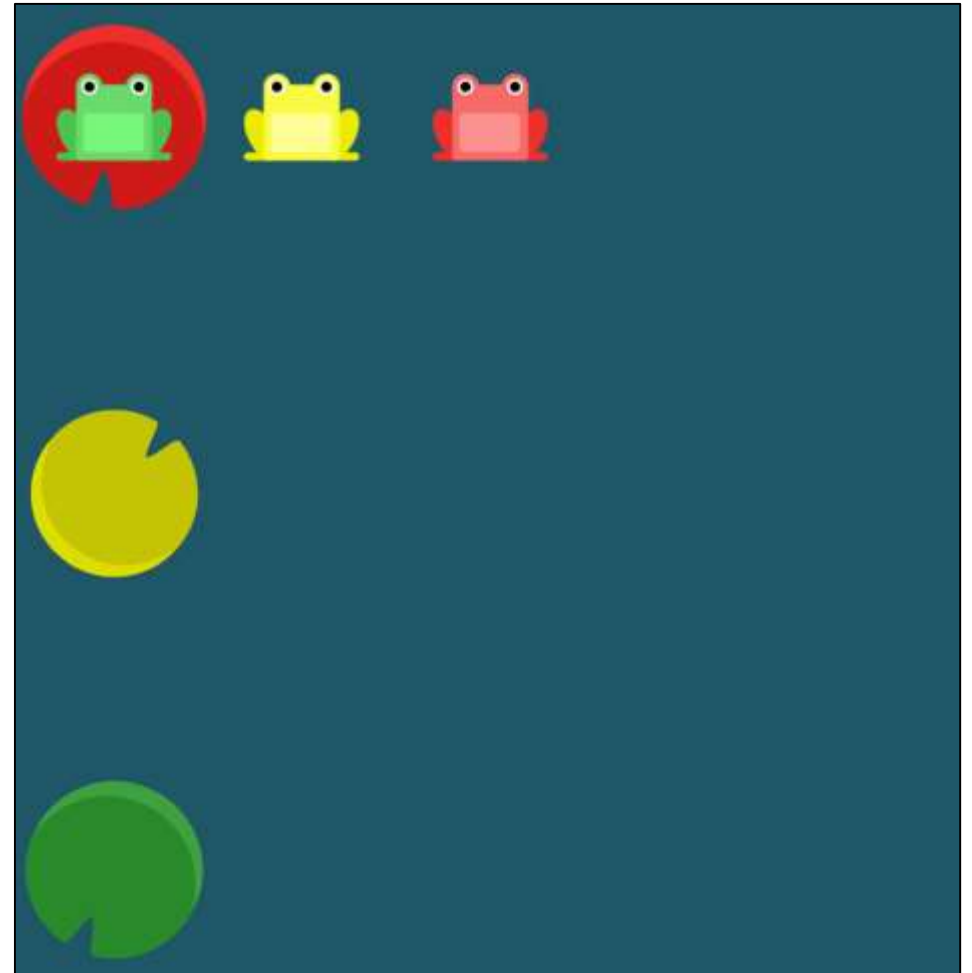
[도전] flex 활용



[참고] flex 개구리 게임

<https://flexboxfroggy.com/#ko>

- flex를 활용한 개구리 배치 게임 사이트



Chapter 07

폼 스타일링

웹 개발자를 위한 기초

폼 스타일링

- ✓ 사용자와 상호작용하는 사이트는 폼을 보유하고 있다.
- ✓ 폼에 효과를 적용하여 UX/UI를 향상시키자

부품 등록 폼

부품명

예: 센서

시리얼 번호

예: 1234

제조사

현대오토에버



비고

특이사항 입력

등록

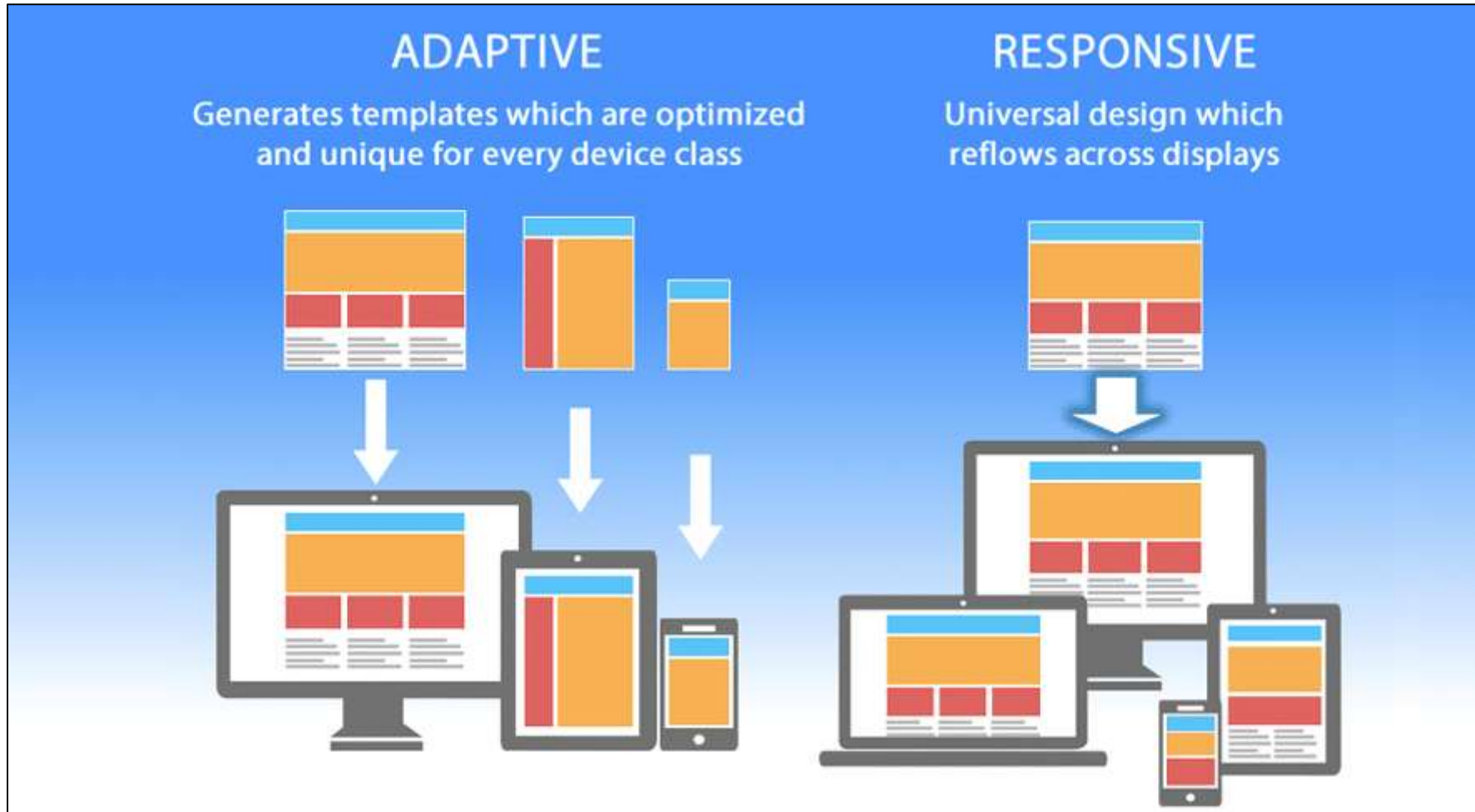
초기화

Chapter 08

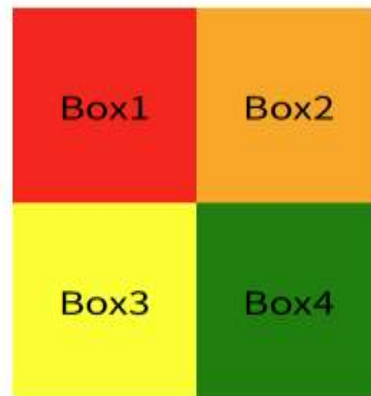
반응형 웹

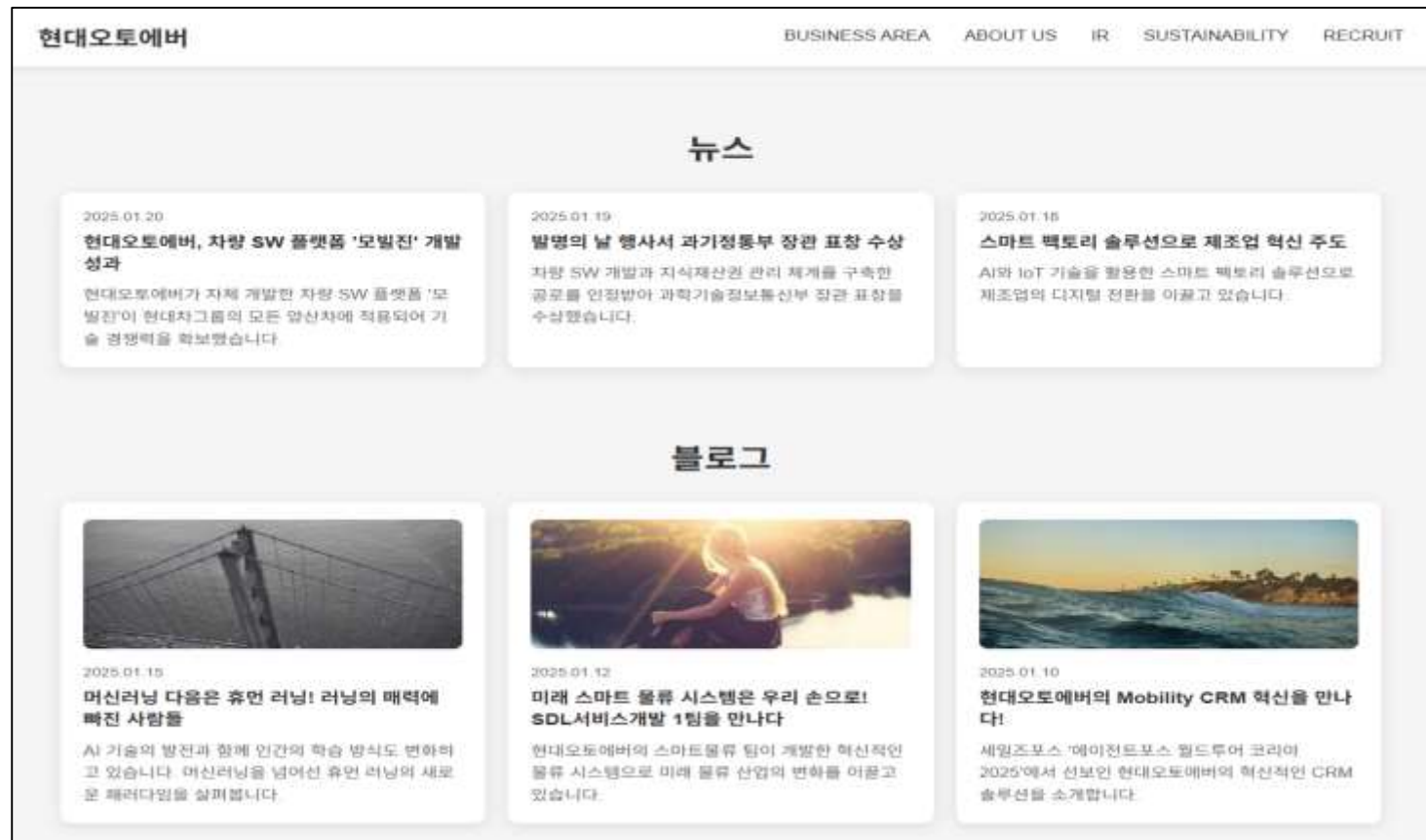
웹 개발자를 위한 기초

반응형 웹이란?









Chapter 09

정리 및 미리보기

웹 개발자를 위한 기초

정리

- ✓ HTML
- ✓ CSS
- ✓ 배운것들을 종합하여 웹 사이트 만들기

미리보기

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Document</title>
</head>
<body>
  <button onclick="document.querySelector('.msg').style.color='red';">색상 변경</button>
  <span class="msg">안녕하세요!</span>
</body>
</html>
```

미리보기

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Document</title>
</head>
<body>
  <input type="text" placeholder="이름을 입력하세요" oninput="updateName(this.value)">
  안녕하세요, <span class="name">게스트님</span>
  <script>
    function updateName(value) {
      document.querySelector(".name").innerText = value || "게스트님";
    }
  </script>
</body>
</html>
```

추가 수업자료

- ✓ position
- ✓ css framework
- ✓ min.css
- ✓ deploy

Chapter 10

자바스크립트 소개

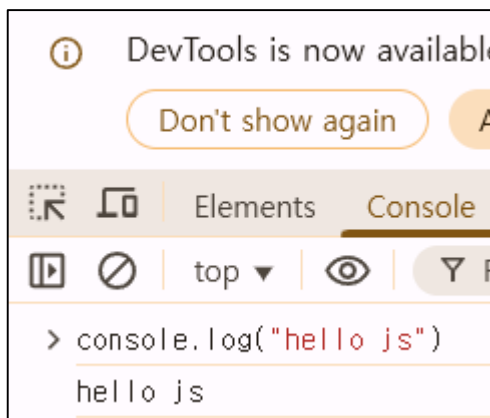
웹 개발자를 위한 기초

자바스크립트 사용이유?

- ✓ 웹 페이지에 동적인 작업을 할 수 있다
- ✓ 비동기 통신(ajax)
- ✓ 서버 구축가능(node.js)

Hello js

- ✓ 브라우저 콘솔 사용
- ✓ script 태그 사용



```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Document</title>
</head>
<body>
  <script>
    console.log('hello js!');
  </script>
</body>
</html>
```

Chapter 11

변수와 자료형

웹 개발자를 위한 기초

변수를 가장 먼저 배우는 이유

- ✓ 프로그램

- ✓ 순서도

- ✓ 순서도를 작성할때, 데이터가 중복 포함된경우(0,-1,1 제외)

- ✓ 숫자, 문자, 결과 출력 등 과정에 데이터가 관여

- ✓ 데이터를 저장(자동으로 진행)하고, 불러오는 방법이 필요

- ✓ 이름, 나이, 결과값 등을 어딘가에 저장됨, 불러오려면?

- ✓ 변수 = 이 저장 공간에 이름을 붙인 것

변수란?

- ✓ 데이터(한 값)를 저장하기 위한 메모리 공간에 붙인 이름
 - ✓ 숫자(정수)나 문자, 소수점 등을 저장할 수 있다.
- ✓ 변수(메모리)의 종류 예시



숫자(정수)가 들어가는
메모리 공간



글자가 들어가는
메모리 공간



소수점 숫자가 들어가는
메모리 공간

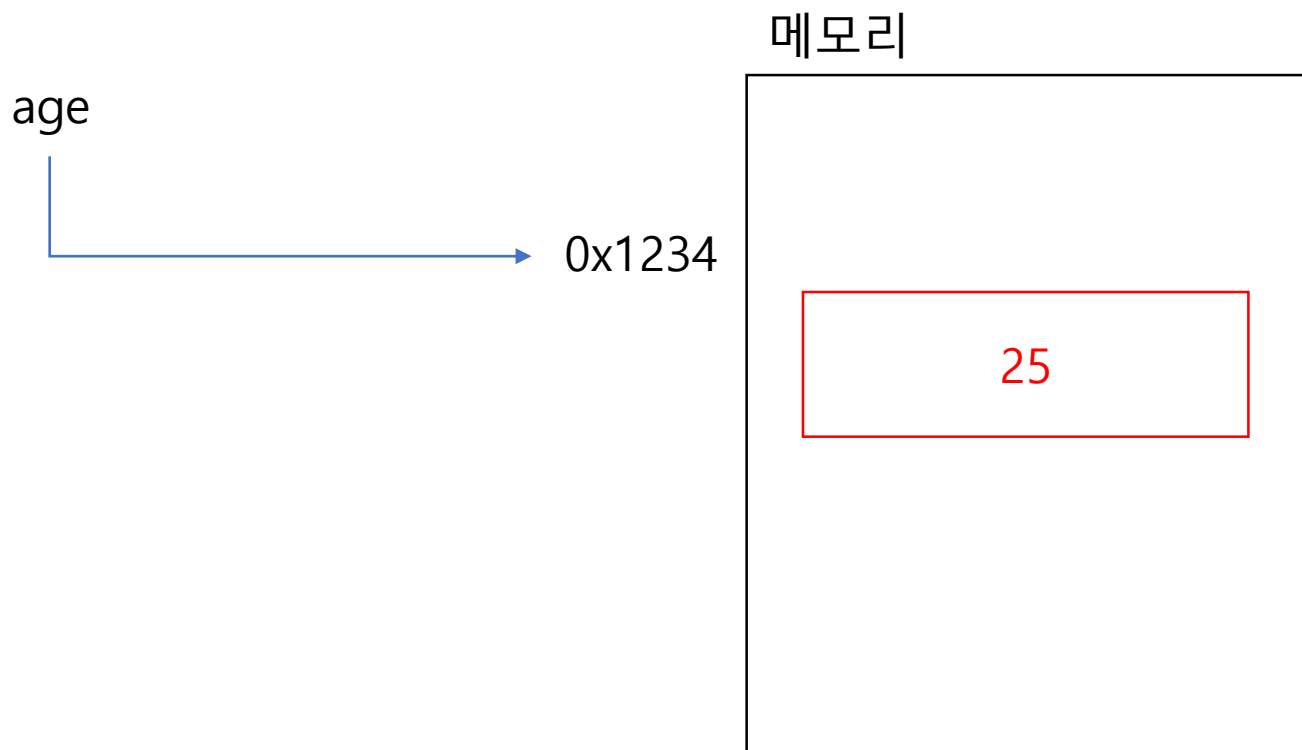
변수명 작명 규칙

- ✓ 영문자(Aa ~ Zz), 숫자(0~9), _ \$ 사용 가능
- ✓ 여러 단어 조합 시 카멜케이스(CamelCase) 사용 권장
 - ✓ 예시) username, totalScore, isLoggedIn
- ✓ 주의 사항
 - ✓ 숫자로 시작 불가 (예 : 1stNumber X)
 - ✓ 공백 사용 불가 (예 : user name X)
 - ✓ 예약어 사용 금지 (예 : const, class, function 등 X)
 - ✓ 대소문자 구분함 (예 : count != Count)

변수가 메모리에 저장되는 과정

- ✓ 변수는 값을 저장하거나 나중에 다시 사용하기 위해 필요
 - ✓ 입력값, 계산 결과, 출력 등 모든 흐름에 관여
- ✓ 메모리 구조 예시

Ex) `const age = 25`



변수 선언 방법

✓ 변수 선언 키워드 var, let, const

```
var age = 25;  
let name = "heobeomsung";  
const company = "mincoding";
```

변수 선언 키워드 비교

✓변수 선언 키워드 var,let,const

var

재선언,재할당 가능 / 함수레벨 스코프

let

재선언 불가 재할당 가능 / 블록레벨 스코프

const

재선언 불가 재할당 불가 / 블록 레벨 스코프

```
var a = 1;
var a = 2;

var a = 3;
a = 4;

let b = 5;
let b = 6;

let c = 7;
c = 8;

const d = 9;
const d = 10;

const e = 11;
e = 12; //런타임 에러
```

용어 정리

- ✓(재)선언
- ✓(재)할당
- ✓스코프

참고) 함수 레벨 스코프 vs 블록 레벨 스코프

```
> function hello(){  
  var a = 5;  
}  
hello();  
console.log(a);
```

✖ ▶ Uncaught ReferenceError: a is not defined
at <anonymous>:5:13

VM182:5

```
{  
  var a = 5;  
}  
console.log(a)
```

5

VM287:4

```
> function hello(){  
  const a = 5;  
}  
hello();  
console.log(a);
```

✖ ▶ Uncaught ReferenceError: a is not defined
at <anonymous>:5:13

```
> {  
  const a = 10;  
}  
console.log(a)
```

✖ ▶ Uncaught ReferenceError: a is not defined
at <anonymous>:4:13

VM472:4

자바스크립트 자료형

✓ 자바스크립트 2가지 데이터 타입

원시자료형 Number / String / Boolean / Null / Undefined

참조 자료형(객체) Array / Object / Function

(문제) 다음 코드의 결과 예상하기

✓ 키워드 : 호이스팅

```
console.log(a)  
var a = 1
```

[도전] 메모리 그리기

- ✓ `var a = 1` 을 선언하였을때, 메모리에 적재되는 과정 작성하기

Chapter 12

배열과 객체

웹 개발자를 위한 기초

- ✓ 회사 직원들을 관리하기 위해 모두 저장하려고 한다

```
const name1 = "김대리"  
const name2 = "이과장"  
const name3 = "박부장"  
const name4 = "금차장"
```

- ✓ 팀원이 10명 이상이라면...?
 - ✓ 변수를 계속 만들어주어야 한다.
 - ✓ 반복적인 데이터를 다룰 때 다른 저장 방법이 필요하다.

✓ 배열이란 ?

- ✓ 같은 자료형의 값들을 하나의 변수 이름으로 묶어서 저장
- ✓ 각 변수는 인덱스(index)로 구분한다.

✓ 인덱스

- ✓ 배열 안에 저장된 값 하나하나를 구분하기 위한 번호
- ✓ 배열의 인덱스는 0부터 시작

index	0	1	2
값	김대리	이과장	박부장

✓ 방법1. 선언 + 할당

```
const names = [];
```

✓ 방법2. 선언 + 초기값 대입

```
const names = ["김대리", "이과장", "박부장"];
```

index	0	1	2
값	김대리	이과장	박부장

배열

- ✓ 배열 관련 메서드 학습
- ✓ 배열 고차함수

[도전] 배열 문제

- ✓ [3,0,2,9,3,9,3]을 하드코딩 하고, 3이 몇개인지 카운팅
- ✓ ['A','U','T','O','E','V','E','R']를 하드코딩하고, 'A'가 존재하는지 검사하여 출력

객체

✓ 회사 직원들의 이름과 나이를 저장하고자 한다

```
const name1 = "김대리"  
const age1 = 31  
  
const name2 = "이과장"  
const age2 = 38  
  
const name3 = "박부장"  
const age3 = 45
```

```
const names = ["김대리", "이과장", "박부장"]  
const ages = [31, 38, 45]
```

객체 리터럴

```
const person1 = {  
  name : '김대리',  
  age : 31,  
}  
  
const person2 = {  
  name : '이과장',  
  age : 38,  
}  
  
const person3 = {  
  name : "박부장",  
  age : 45,  
}  
  
const persons = [person1, person2, person3]
```

객체 속성 접근/수정

```
const person = {  
  name : '김대리',  
  age : 31,  
}  
  
console.log(person.name)  
console.log(person.age);  
console.log(person['age']);  
person.age = 32  
console.log(person.age);  
person.job = '개발자'  
console.log(person)
```


객체 문법 : 구조 분해 할당(Destructuring)

```
const person = {  
  name : '김대리',  
  age : 31,  
}  
  
// const name = person.name;  
// const age = person.age;  
  
const {name, age} = person;
```

객체 문법 : 스프레드 연산자(Spread)

```
const person = {  
  name : '김대리',  
  age : 31,  
}  
  
// 객체 복사 및 확장  
const newPerson = { ...person, job: '개발자' };  
console.log(newPerson); // { name: '김대리', age: 31, job: '개발자' }
```

객체 문법 : Rest 파라미터

```
const person = {  
  name: '김대리',  
  age: 31,  
  job : '개발자',  
};  
  
const { name, ...rest } = person;  
  
console.log(name); // '김대리'  
console.log(rest); // { age: 31, job: '개발자' }
```

객체 문법 문제

- ✓ users 배열에서 첫 번째 사용자의 이름만 firstName에 할당하고,
- ✓ 나머지 사용자들의 이름만 뽑아서 names 배열에 할당하는 코드 작성

```
const users = [  
  { name: "철수", age: 21 },  
  { name: "영희", age: 22 },  
  { name: "민수", age: 23 },  
];
```

Chapter 13

연산자 및 제어문

웹 개발자를 위한 기초

연산자

- ✓ 산술 연산자(+, -, *, /, %, **, ++, -- ...)
- ✓ 대입 연산자(=, +=, -=, *=, /=, %=, **=)
- ✓ 비교 연산자(==, !=, ==, !=, >, <, >=, <=)
- ✓ 논리 연산자(&&, ||, !)
- ✓ 비트 연산자(&)
- ✓ 삼항 조건 연산자(조건식 ? 참일때 값 : 거짓일때 값)

논리 연산자

- ✓ 논리적 표현식을 비교하여, 하나의 참 또는 거짓의 결과를 반환
 - &&(AND 연산자)
 - 가장 먼저 평가되는 falsy 값을 반환, 없을 시 가장 마지막의 값 반환
 - ||(OR 연산자)
 - 가장 먼저 평가되는 truthy 값을 반환, 없을 시 가장 마지막의 값 반환
 - !(NOT)연산자
 - 피연산자의 논리값을 반전(단항 논리 연산자중 우선순위 가장 높음)

```
console.log(false && "hello" && 123);  
console.log("hi" && 0 && "world");  
console.log("a" && 1 && true && "끝");  
  
console.log(null || 0 || "" || "JS" || 42);  
console.log(false || undefined || 0);
```


[도전]결과 예상

```
console.log(0 && "A" && null && "B" && undefined && "C");  
console.log("X" && 1 && true && [] && {} && "마지막");  
  
console.log(null || 0 || "" || undefined || NaN || false);  
console.log(null || 0 || "" || "첫 truthy" || "두번째 truthy");
```

```
const isLogin = isValidId() && isValidPW()
```

조건문이란 ?

- ✓ 주어진 조건에 따라 코드 실행 흐름을 분기 시키는 문장
- ✓ "만약 ~~ 라면, ~~를 실행하라" 형태의 논리 구조
- ✓ 예시
 - ✓ 지각 할 것 같으면 택시를 탄다.
 - ✓ 회식 메뉴가 소고기라면 참석한다.

if 문 구조

✓ 조건문 형태

```
if (조건식) {  
    // 조건이 참일 때 실행될 코드  
} else {  
    // 위의 모든 조건이 거짓일 때 실행  
}
```

✓ 조건식은 true 또는 false 를 반환해야 함

if-else / else if

✓ 다양한 조건을 처리하는 방법

```
if (조건식1) {  
    // 조건식1 이 참일 때 실행될 코드  
} else if (조건식2) {  
    // 조건식1 이 거짓이고, 조건식2 가 참일 때 실행될 코드  
} else {  
    // 위의 모든 조건이 거짓일 때 실행  
}
```

다중 조건 처리 - 중첩 조건문

- ✓ if 문 안에 또 다른 if 문을 사용하는 구조
- ✓ 논리적 순서를 분리하거나, 1차 조건 만족 후 세부 조건 검토 시 사용

```
if (조건식1) {  
    // 조건식1이 참일 때 실행될 코드  
    if (조건식2) {  
        // 조건식1 과 조건식2가 모두 참일 때  
    }  
}
```

[도전]조건문

✓ 여행지를 선택하고, 나이와 회원 여부에 따라 추천 혜택을 받는다.

✓ 여행지 이름과 예약 수량, 회원 여부와 나이가 다음과 같이 저장되어 있다.

```
const destinations = [ "제주도", "강릉", "부산" ];  
const bookingCounts = [ "2", "0", "1" ];  
const isMember = true;  
const age = 25;
```

✓ 요구사항

✓ 각 여행지의 예약 현황을 출력한다. (예약 수량은 정수로 변환하여 활용)

✓ 1인당 여행 비용은 100,000원으로 고정되어 있다.

✓ 전체 여행 예약 수가 3건 이상이고, 회원이며, 나이가 20세 이상이면

✓ "여행 추천 혜택 대상" 출력

✓ 아니라면 "일반 고객" 출력

출력 결과

[예약 현황]

제주도: 2건

강릉: 0건

부산: 1건

총 금액: 300000원

여행 추천 혜택 대상

반복문

- ✓ 코드를 여러 번 실행하고 싶을 때 사용하는 중요한 문법
- ✓ 조건을 만족하는 동안 특정 코드를 반복 실행하는 구조

반복문 종류	기본 구조	사용 지점
for	횟수 기반	반복 횟수가 정해진 경우
while	조건 기반	반복 횟수를 모를 때

for 문

✓ 반복 횟수가 정해진 경우 활용

- ✓ 시작부터 조건이 만족할 때 까지 특정 횟수를 반복

✓ for문 기본 형태

```
for (시작점; 조건식; 매번 반복 후) {  
    // 조건이 만족하는 동안 반복될 코드  
}
```

✓ 기본 예제

- ✓ 횟수 반복 ("안녕하세요" 5번 출력)

```
for(let i = 0; i < 5; i++){  
    console.log("안녕하세요")  
}
```

while 문

- ✓ 조건이 참인 동안 코드를 계속 반복하는 문법
- ✓ 반복 횟수가 미리 정해지지 않은 상황에 유용
- ✓ 기본 구조

```
while (조건식) {  
    // 조건이 true 인 동안 실행될 코드  
}
```

while 문

```
let i = 1;
while (1 <= 5) {
    console.log(i, "번째 반복");
    i++;
}
```

[도전] 조건과 반복

✓ 문제

- ✓ 여러 개의 여행지가 있고, 각각은 계절 정보를 포함합니다.
- ✓ 사용자에게 계절을 입력 받아, 해당 계절에 맞는 여행지를 추천해주는 프로그램을 작성하세요.
- ✓ 여행지 목록과 계절 정보는 인덱스로 연결되어 있음
 - ✓ places[0] = 제주도, seasons[0] = 여름 -> 제주도는 여름에 추천

```
const places = ["제주도", "강릉", "부산", "설악산", "여수"];  
const seasons = ["여름", "겨울", "여름", "가을", "봄"];
```

- ✓ 일치하는 여행지가 없다면 "해당 계절의 추천 여행지가 없습니다." 출력

[도전] 조건과 반복

✓ 실행 예시

✓ 사용자 입력 대기

추천받고 싶은 계절을 입력하세요:

✓ 여름 입력 시

추천받고 싶은 계절을 입력하세요: 여름

[추천 여행지]

- 제주도
- 부산

✓ 없는 계절 입력 시

추천받고 싶은 계절을 입력하세요: 봄같은 가을

[추천 여행지]

해당 계절의 추천 여행지가 없습니다.

Chapter 14

함수와 파라미터

웹 개발자를 위한 기초

중복되는 로직의 문제점

✓ 같은 코드가 여러 번 중복되는 경우

✓ 예제

✓ 각 여행지의 가격이 정해져 있고, 할인 쿠폰을 적용할 수 있다.

✓ 할인 쿠폰 = (2만원 * 10% 추가) 할인

```
const jejuPrice = 200000;  
console.log("제주도 가격: " + (jejuPrice - 20000 * 1.1) + "원");  
  
const gangneungPrice = 150000;  
console.log("강릉 가격: " + (gangneungPrice - 20000 * 1.1) + "원");  
  
const busanPrice = 180000;  
console.log("부산 가격: " + (busanPrice - 20000 * 1.1) + "원");
```


중복되는 로직의 문제점

✓ 다음과 같은 상황에서 코드 유지보수를 어떻게 해야 할까 ?

✓ 할인 금액이 변경되는 경우

✓ 추가 할인 비율이 변경되는 경우

✓ 금액을 관리해야 할 도시가 더 많을 경우

```
const jejuPrice = 200000;  
console.log("제주도 가격: " + (jejuPrice - 20000 * 1.1) + "원");  
  
const gangneungPrice = 150000;  
console.log("강릉 가격: " + (gangneungPrice - 20000 * 1.1) + "원");  
  
const busanPrice = 180000;  
console.log("부산 가격: " + (busanPrice - 20000 * 1.1) + "원");
```

같은 로직이 반복되는 경우

리스크 ↑

시간 낭비 ↑

함수란

✓ 로직이 동일하고, 입력 값만 다를 때 코드를 따로 묶어주자

✓ 예제

```
function getDiscountedPrice(basePrice, discountAmount, discountRate) {  
    return basePrice - discountAmount * discountRate;  
}
```

함수

```
const discountAmount = 20000;  
const discountRate = 1.1;
```

```
const jejuPrice = getDiscountedPrice(200000, discountAmount, discountRate);  
const gangneungPrice = getDiscountedPrice(150000, discountAmount, discountRate);  
const busanPrice = getDiscountedPrice(180000, discountAmount, discountRate);
```

```
console.log("제주도 가격: " + jejuPrice + "원");  
console.log("강릉 가격: " + gangneungPrice + "원");  
console.log("부산 가격: " + busanPrice + "원");
```

함수 정의문

```
function 함수이름(전달받을값){//함수 선언식
    //실행할 코드
    return 1;
}

const 변수명 = function(전달받을값){//함수 표현식
    //실행할 코드
    return 2;
}

const 화살표함수 = () => { //화살표 함수
    //실행할 코드
    return 3;
}
```

함수 호출

```
function 함수이름(전달받을값){//함수 선언식
    //실행할 코드
    return 1;
}

const 변수명 = function(전달받을값){//함수 표현식
    //실행할 코드
    return 2;
}

const 화살표함수 = () => { //화살표 함수
    //실행할 코드
    return 3;
}

함수이름();
변수명();
화살표함수();
```

함수 정의 : 함수가 역할 작성

함수 호출 : 정의한 함수를 실행

함수 호출

```
function 함수이름(전달받을값){//함수 선언식
    //실행할 코드
    return 1;
}

const 변수명 = function(전달받을값){//함수 표현식
    //실행할 코드
    return 2;
}

const 화살표함수 = () => { //화살표 함수
    //실행할 코드
    return 3;
}

함수이름();
변수명();
화살표함수();
```

함수 정의 : 함수가 역할 작성

함수 호출 : 정의한 함수를 실행

파라미터 (Parameter)

- ✓ 외부에서 값을 전달 받아야 하는 경우
- ✓ 함수 내에서 이름을 함께 출력하고 싶은 경우
- ✓ 호출할 때마다 다른 값을 전달받아서 활용

파라미터

```
function sayHello(name) {  
    console.log("환영합니다! " + name + "님!");  
}
```

이름을 전달받아, 함수 내부에서 활용

파라미터 : 정의 시 받기로 한 값

```
sayHello("JS");    // "환영합니다! JS님!" 출력  
sayHello("VUE");   // "환영합니다! VUE님!" 출력
```

함수 호출 시 이름을 전달

인자 (Argument) : 호출 시 전달한 실제 값

전달인자 (Argument)

반환값 (Return Value)

- ✓ 함수를 실행한 반환값을 외부에서 사용할 수 있다
- ✓ 숫자 2개를 더한 값을 반환 (return 키워드)

```
function add(a, b) {  
  return a + b;  
}  
  
const result = add(3, 5);  
console.log(result);
```

[도전]3가지 방법으로 함수 정의

- ✓숫자 2개를 전달받아 더한 값을 출력
- ✓변수명, 함수명, 매개변수명 자유
- ✓함수 호출하여 결과 확인 및 에러 확인

[도전] 가격 계산기 만들기

✓ 가격 계산기 만들어보기

✓ 이름이 `getDiscountedPrice` 인 함수를 정의한다.

✓ 함수 구성

✓ 아래 3가지 데이터를 전달 받는다.

✓ `basePrice` : 기본 가격(int)

✓ `discountAmount`: 할인 가격(int)

✓ `discountRate`: 추가 할인 비율(double)

✓ 기본 가격 - (할인 가격 * 추가 할인 비율) 을 반환한다.

✓ 아래 조건에 맞게 3번 호출한다.

✓ 할인 가격: 20,000, 할인 비율: 1.1

✓ 기본 가격: 제주도(200000), 강릉(150000), 부산(180000)

• 출력 결과

제주도 가격: 178000원

강릉 가격: 128000원

부산 가격: 158000원

콜백함수

- ✓ 다른 함수의 인자로 전달되어, 그 함수 내부에서 호출되는 함수
- ✓ 나중에 호출될 함수
- ✓ 인자로 전달되어, 해당 함수 내부에서 실행되는 함수

```
function greet(name, callback) {  
  console.log(`안녕하세요, ${name}님!`);  
  callback(); // 전달받은 콜백 함수 실행  
}  
  
function sayBye() {  
  console.log("안녕히 가세요!");  
}  
  
greet("허범성", sayBye);
```

Chapter 15

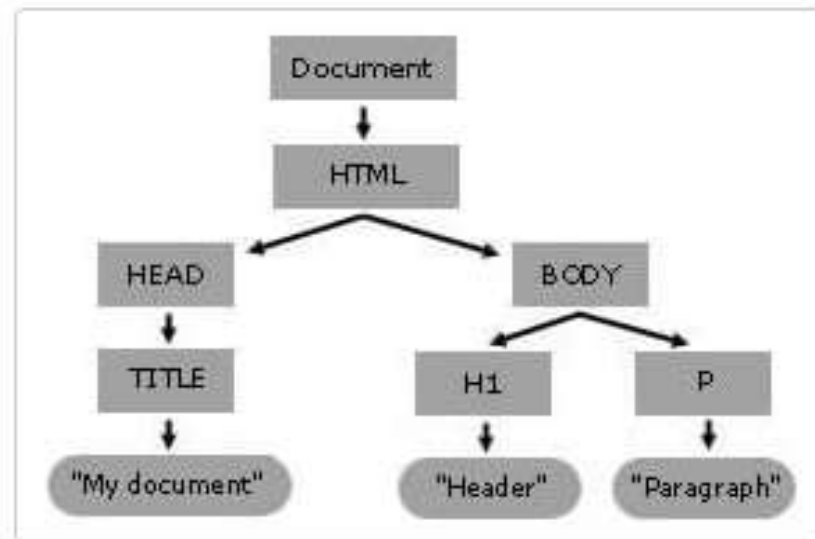
이벤트 처리

웹 개발자를 위한 기초

DOM

✓ HTML 문서의 구조적 표현 방법

```
<html lang="en">
  <head>
    <title>My Document</title>
  </head>
  <body>
    <h1>Header</h1>
    <p>Paragraph</p>
  </body>
</html>
```



[실습]QuerySelector

- ✓ H1태그의 콘텐츠를 hello wolrd로 변경

```
<html lang="en">
  <head>
    <title>My Document</title>
  </head>
  <body>
    <h1>Header</h1>
    <p>Paragraph</p>
  </body>
  <script>
    document.querySelector('h1').textContent = 'hello world!'
  </script>
</html>
```

Event Driven

✓ 브라우저에서 event(사건) 발생시, 지정된 함수 호출

1. 함수를 만든다

2. 이벤트 발생시, 함수가 자동 호출 되도록 "등록" 한다

3. 이벤트가 발생하면, 함수가 호출된다

용어 정리

✓ 콜백 함수

1. 다른 코드의 Argument로 함수 이름을 넘겨 주는 함수
2. 소스코드의 직접 호출이 아닌, 이벤트로 인해 호출되는 함수

이벤트 핸들러

1. 이벤트 발생시 호출되는 함수
2. "콜백 함수" 중 용도를 명확하게 나타낸 용어

이벤트 리스너

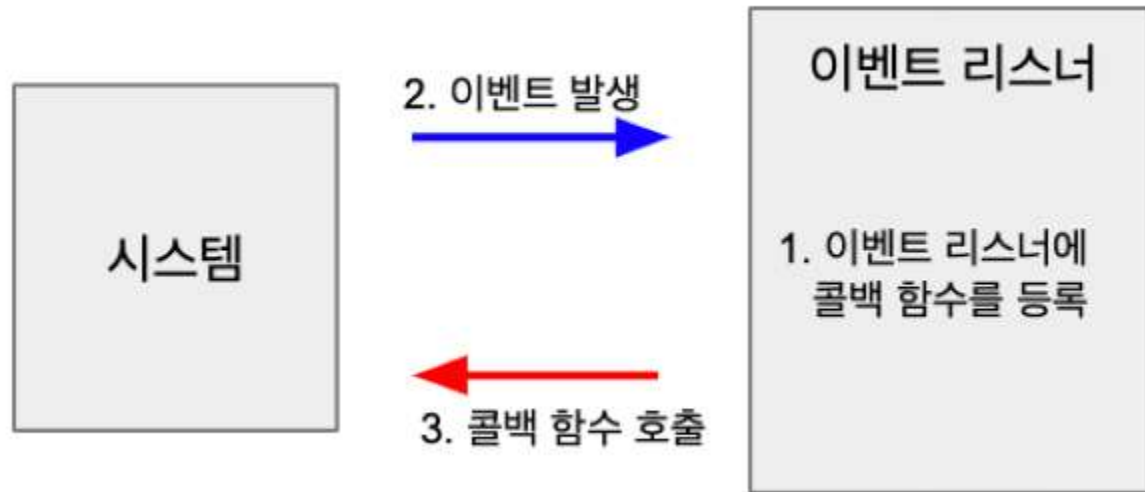
1. 이벤트가 발생하는지 감지하여, 예약된 이벤트 핸들러를 호출해주는 객체

이벤트 발생 동작 원리

✓ 이벤트 리스너에 함수를 등록한다

addEventListener 메서드를 사용하여 이벤트 핸들러(콜백함수)를 등록한다

addEventListener("이벤트이름", "핸들러이름");



[실습] 버튼 클릭시 함수 호출

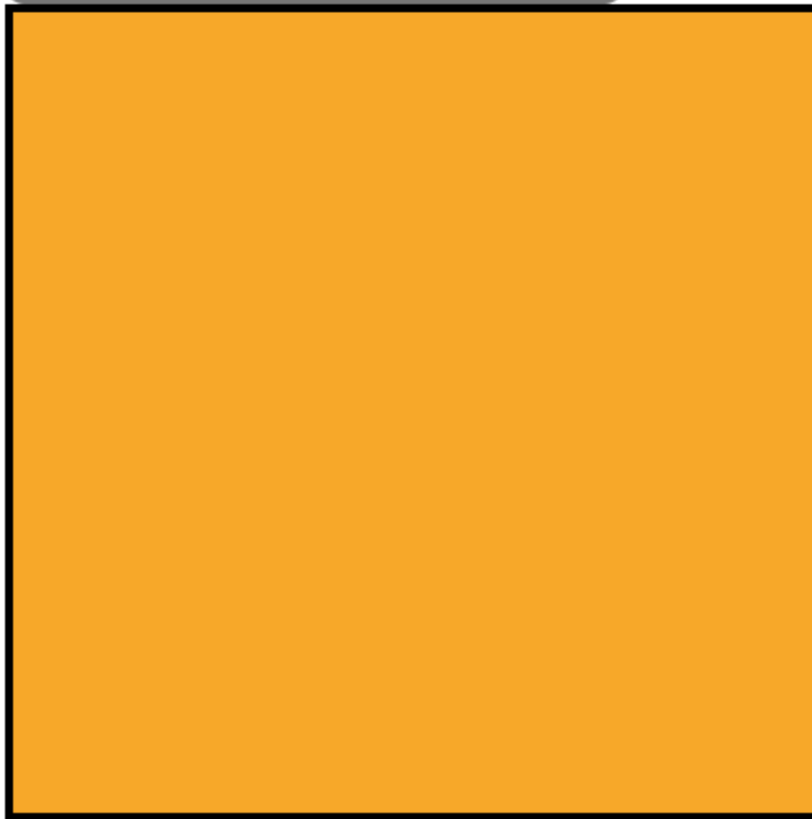
```
<html lang="en">
  <head>
    <title>My Document</title>
  </head>
  <body>
    <input type="button" value="버튼">
  </body>
  <script>
    document.querySelector('input').addEventListener("click", btnClick)

    function btnClick(){
      console.log("hello world!")
    }
  </script>
</html>
```


[실습] 버튼 클릭시 함수 호출 2

```
<html lang="en">
  <head>
    <title>My Document</title>
  </head>
  <body>
    <input type="button" value="버튼" onclick="btnClick()">
  </body>
  <script>
    function btnClick(){
      console.log("hello world!")
    }
  </script>
</html>
```

색상바꾸기



Change Event

✓ 값이 바뀔 때 발생하는 이벤트

```
<!DOCTYPE html>
<html lang="ko">
<head>
  <meta charset="UTF-8">
  <title>change event</title>
</head>
<body>
  <input></input>
  <div></div>

  <script>
    const input = document.querySelector('input')
    const div = document.querySelector('div')

    input.addEventListener('change', changeEvent)

    function changeEvent(event){
      div.textContent = event.target.value
    }
  </script>
</body>
</html>
```

autoever

autoever

Keydown Keyup Event

✓ 키를 눌렀을 때 반응하는 event

```
<!DOCTYPE html>
<html lang="ko">
<head>
  <meta charset="UTF-8">
  <title>change event</title>
</head>
<body>
  <input></input>
  <div></div>

  <script>
    const input = document.querySelector('input')
    const div = document.querySelector('div')

    input.addEventListener('keydown', changeEvent)

    function changeEvent(event){
      div.textContent = event.code
    }
  </script>
</body>
</html>
```

ShiftLeft

[도전]회원가입 검사하기

- ✓ Id 및 pw가 둘다 공백일때, 회원가입 방지하기

아이디와 비밀번호를 모두 입력하세요.

Chapter 16

비동기 처리

웹 개발자를 위한 기초

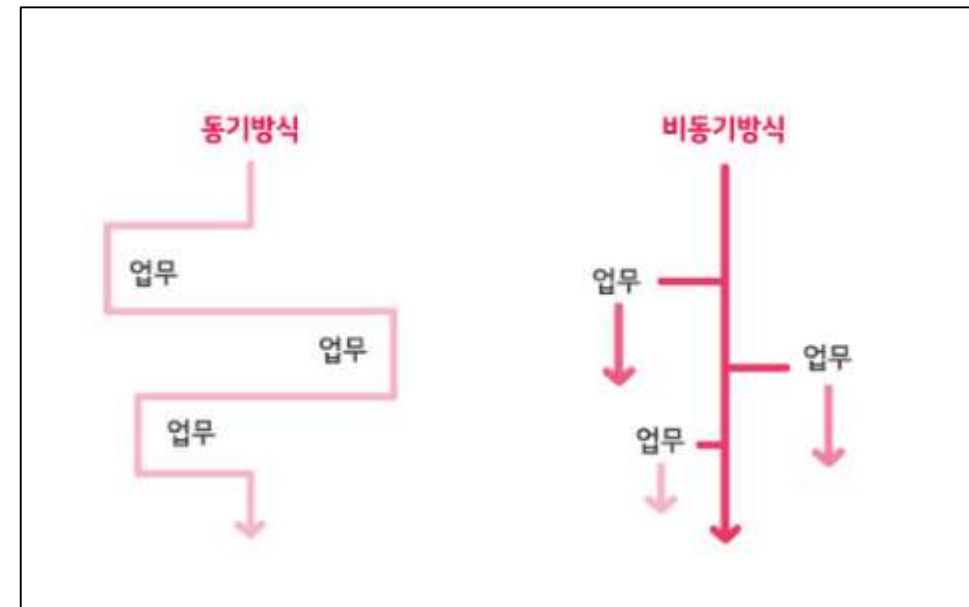
비동기

✓ 동기 방식

한 업무가 끝나야, 다음 업무를 시작한다

비동기 방식

다른 업무를 기다리지 않고, 진행한다



비동기

- ✓ 비동기란, 非(아닐비)동기 동기가 아니다
- ✓ 동기란, 同氣(같은동, 기약할기) -> 동시성의



사진=AFPBBNews

코드의 결과를 예측해보자

```
console.log(1)  
console.log(2)  
console.log(3)
```

코드의 결과를 예측해보자

```
console.log('1')
setTimeout(() => {
  console.log('2')
}, 1000);
console.log('3')
```

```
console.log('1')
setTimeout(() => {
  console.log('2')
}, 0);
console.log('3')
```

[도전]axios 활용

- ✓ axios 비동기 통신하여, 데이터 활용하기
- ✓ 서버 주소 : <https://jsonplaceholder.typicode.com/todos>
- ✓ 해당 서버에서 data값을 받아 배열에 할당한후

completed : false,
userId가 짝수,
title의 글자수가 10자 이상

위의 조건에 해당하는 데이터만 배열에 할당후 출력

추가 학습자료

- ✓ 필수)템플릿 리터럴
- ✓ 추가 학습 자료 : REST API

감사합니다.