codingOn X posco

K-Digital Training 신재생에너지 활용 IoT 과정

데이터 수집과 웹 기초

데이터 수집 방식

- 웹에서 데이터를 수집하는 방식에는 크게 두가지 방식이 존재
- API(Application Programming Interface)
- 웹 크롤링(Web Crawling)

항목	API	웹 크롤링
데이터 제공	공식적으로 제공하는 데이터 인터페이스 사용	웹 페이지의 HTML을 직접 파싱
작동 방식	서버로 요청 → JSON 응답 처리	HTTP 요청 → HTML 분석 후 데이터 추출
장점	신뢰도 높고, 빠르고, 사용이 편리함	API 없이도 원하는 데이터 추출 가능
단점	사용량 제한, API 제공 범위 제한적	느리고, HTML 구조 변경 시 코드 수정 필요
사용	공식적으로 API가 제공될 때	API가 없거나 추가 데이터가 필요할 때

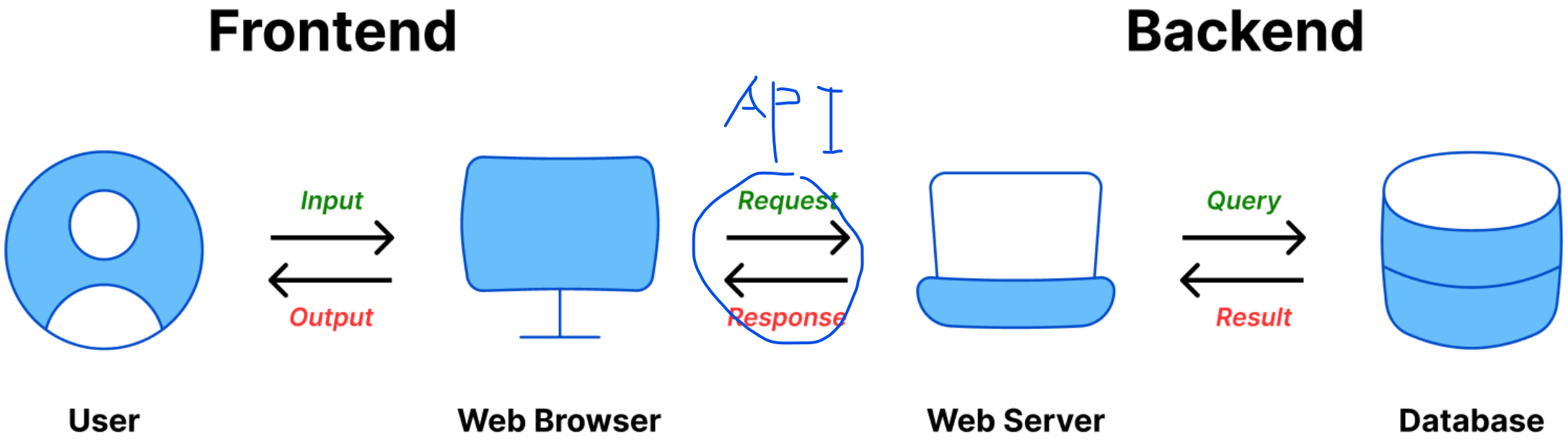
codingOn

API

API

- API(Application Programming Interface)란?
- 응용 프로그램이 서로 데이터를 주고받는 방법을 제공하는 인터페이스
- 특정 데이터 제공자가 정해진 규격으로 데이터를 전달(JSON or XML)
- 예)
  - 날씨 정보 제공 API (ex: OpenWeatherMap API)
  - 소셜 미디어 API (ex: Instagram API)
  - 구글 지도 API
  - 내 서버에서 내 클라이언트로 API 제공(프론트엔드-백엔드)

프론트엔드와 백엔드


Frontend
Backend
API
Input
Output
Request
Response
Query
Result
User
Web Browser
Web Server
Database

JSON이란?

- JavaScript Object Notation의 약자
- 데이터를 문자열의 형태로 나타내기 위해서 사용
  - 즉, 데이터를 표시하는 방법 중 하나
- 네트워크를 통해 다른 시스템들이 소통할 때 사용하는 경량의 DATA 교환 형식
- 자바스크립트에서 파생되었으나 현재는 다른 프로그래밍 언어에서도 지원하는 데이터 포맷

- 가독성이 뛰어나 컴퓨터와 사람 모두 해석하기 편함
- JavaScript의 Object를 기반으로 하는 텍스트 형식

JSON이란?

- Python 딕셔너리와 유사
- key 이름을 **큰 따옴표("key-name")로 감싸는 차이점**이 있음
- 문자열, 숫자(정수, 실수), 불리언(true, false), 중첩된 객체와 배열 저장도 가능함

```
{
  "name": "Sean",
  "age": 20,
  "skills": ["JavaScript", "JAVA", "Python"],
  "family": {
    "father": "Jake",
    "mother": "Sunny",
  },
  "height": 120.9
  "isLogin": true,
}
```

JSON 예시

요약하자면 Client와 Server가
JSON 이라는 특별한 형식의 데이터로 정보를 주고 받는다!

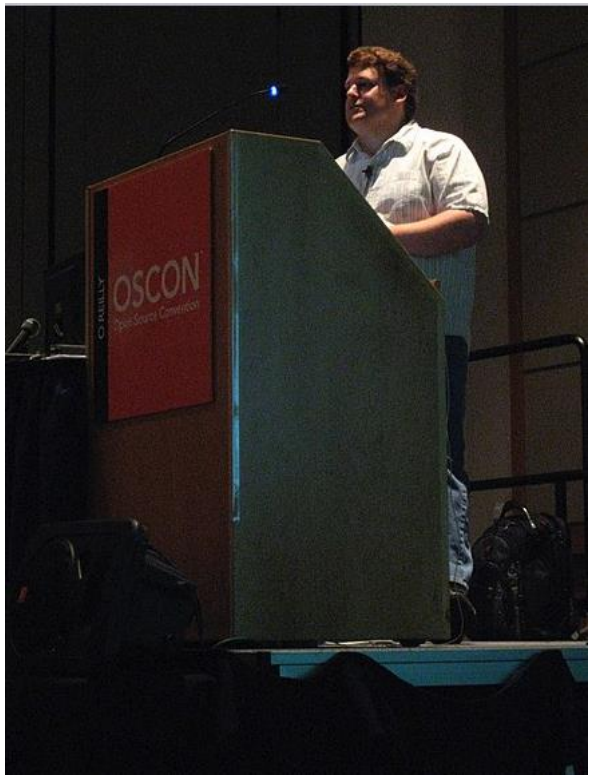
API HTTP통신

- HTTP란? Hypertext Transfer Protocol
- 인터넷 상에서 데이터를 주고 받기 위한 프로토콜 (약속)
- 클라이언트는 서버에게 자원을 요청(request)하고, 서버는 클라이언트에게 요청을 처리해서 응답(response)
- API는 주로 HTTP 통신을 통해 데이터를 주고받음
- API는 Rest API나 Restful API라고 불리움

Rest란?

REST는 Representational State Transfer의 약자로,
로이 필딩(Roy Fielding)이 2000년에
그의 박사학위 논문에서 소개된 웹 아키텍처 스타일

클라이언트와 서버 간의 경량 통신을 가능하게 하며,
웹 기반 소프트웨어 어플리케이션에서 널리 사용



Rest 특징

- 리소스(Resource): REST는 모든 것을 리소스로 간주
- 리소스란 웹 서비스에서 제공하는 모든 것을 의미합니다. 데이터, 기능, 서비스 등 어떠한 형태의 정보든지 리소스로 간주
- URI(Uniform Resource Identifier): 각 리소스는 고유한 식별자인 URI를 가지며, 이를 통해 리소스에 접근
- HTTP 메서드 활용: HTTP 메서드(GET, POST, PUT, DELETE 등)를 사용하여 리소스를 조작

Rest 특징

- 상태를 관리하지 않음(Stateless): REST 아키텍처는 상태를 관리하지 않으며, 각 요청은 독립적으로 처리되며, 서버는 요청에 필요한 모든 정보를 포함하고 있어야 함
- 캐시 처리 가능(Cacheable): REST 리소스는 캐싱이 가능하도록 설계되어, 클라이언트나 중개 서버가 캐시를 활용하여 효율적으로 리소스를 관리할 수 있음
- Client-Server: 클라이언트(사용자 인터페이스)와 서버(데이터 스토리지)가 분리되어 있어 각각 독립적으로 발전

GET(조회)

- 역할: 서버로부터 리소스를 조회
- URL로 리소스의 정보를 요청
- 서버는 요청한 리소스의 정보를 응답으로 전달
- 서버의 상태나 데이터를 변경시키지 않음
- 캐싱을 활용하여 자주 요청되는 데이터를 효율적으로 전달

POST(생성)

- 역할: 서버에 새로운 리소스를 생성
- URL로 요청 시 새로운 데이터를 서버로 전송
- 서버는 요청 데이터를 처리하여 새로운 리소스를 생성하거나 해당 데이터를 저장
- 서버의 상태나 데이터를 변경시키는 메서드로 주의해서 사용

PUT/PATCH(갱신)

- 역할: 서버의 자원을 갱신
- PUT: 리소스 전체를 업데이트
- PATCH: 리소스의 일부분을 업데이트
- 클라이언트는 수정하고자 하는 리소스의 새 데이터를 서버로 보냄
- 서버는 이 데이터를 사용하여 리소스를 업데이트하고, 성공 여부를 클라이언트에게 알림

DELETE(삭제)

- 역할: 리소스를 삭제
- 클라이언트가 특정 리소스의 삭제를 요청하면, 서버는 해당 리소스를 삭제하고 결과를 클라이언트에게 알림
- 서버의 데이터가 삭제되므로 주의해서 사용

HTTP 요청 메서드

codingOn

- API 요청(Request) HTTP 메서드 종류

메서드	설명	예(게시판)
GET	서버에서 데이터를 가져올 때 사용	게시물 목록 가져오기
POST	서버에 데이터를 전송하여 새로운 데이터를 생성할 때 사용	새 게시물 작성
PUT	서버의 기존 데이터를 전체 업데이트할 때 사용	게시물 전체 수정
PATCH	서버의 기존 데이터의 일부만 업데이트할 때 사용	게시물 일부 수정
DELETE	서버에서 데이터를 삭제할 때 사용	게시물 삭제

요청(Request)

- 요청의 구조
- URL: 요청 대상의 주소
- HTTP 메서드: 요청의 종류(GET, POST 등)
- Headers: 요청의 부가 정보(예: 인증 토큰, Content-Type)
- Body: 요청에 포함되는 데이터 (POST, PUT, PATCH에서 사용)

응답(Response)

- 응답의 구조
- HTTP 상태 코드: 요청 성공 또는 실패를 나타냄.
- Headers: 응답의 부가 정보 (예: Content-Type, 날짜)
- Body: 요청 결과 데이터 (JSON, XML 등)

requests 모듈

- HTTP 요청을 간단하고 효율적으로 보낼 수 있도록 도와주는 라이브러리
- 외부모듈로 pip install requests 로 설치하여 사용

```python
import requests

# 외부 API에서 데이터를 가져옴
url = "https://koreanjson.com/posts"
response = requests.get(url)

# 응답 처리
if response.status_code == 200:
    data = response.json()
    print("API 데이터:", data)
else:
    print("API 요청 실패:", response.status_code)
```

HTTP 응답 코드

200번대	성공	200(OK), 201(Created)
300번대	리다이렉션	301(Moved Permanently)
400번대	클라이언트 오류	400(Bad Request), 401(Unauthorized) 404(Not Found)
500번대	서버 오류	500(Internal Server Error)

웹 크롤링과 웹 스크래핑

웹 크롤링

- 웹 크롤링(Web Crawling)이란?
- 웹 페이지의 HTML 데이터를 직접 가져와 필요한 정보를 추출하는 기술
- 웹사이트에서 제공하는 데이터가 API로 제공되지 않을 때 사용
- 주로 HTML 문서나 URL 목록과 같은 원시 데이터를 수집
- 예) 구글 검색 엔진

웹 스크래핑

- 웹 스크래핑(Web Scraping)이란?
- 웹 페이지에서 특정 데이터를 추출하고 가공하는 작업.
- 특정한 텍스트, 이미지, 표 등 유용한 정보만 선별하여 수집
- 예) 상품 가격 목록, 뉴스 기사, 리뷰 등

- 웹 크롤링과 웹 스크래핑의 용어는 혼용해서 사용하는 경우가 대다수이며 그 경계도 모호하지만 개념과 목적이 다름
- 앞으로 우리가 하는 것은 웹 스크래핑임

codingOn

HTML

HTML이란?

HTML(Hypertext Markup Language)

- 마크업 언어: 태그를 이용해서 구조화하는 언어

```
<html>
    <head>

    </head>

    <body>

    </body>
</html>
```

HTML 구조 설명

```
1  <!DOCTYPE html>
2  <html>
3    <head>
4      <meta charset="utf-8">
5      <title>Hello World</title>
6    </head>
7    <body>
8      <h1>Hello World</h1>
9      <p>안녕하세요! HTML5</p>
10   </body>
11 </html>
```

- HTML5 문서는 반드시 <!DOCTYPE html> 으로 시작하여 문서 형식을 HTML5로 지정
- 실제적인 HTML 문서는 2번째 행부터 시작, <html>과 </html> 사이에 작성
- <head>와 </head> 사이에는 HTML 문서 정보를 정의하는 코드가 포함된다.(제목, 저장 방식, 브라우저의 크기 등등)
- 웹 브라우저에 출력되는 모든 요소는 <body> 와 </body> 사이에 위치한다

<head> 태그

codingOn

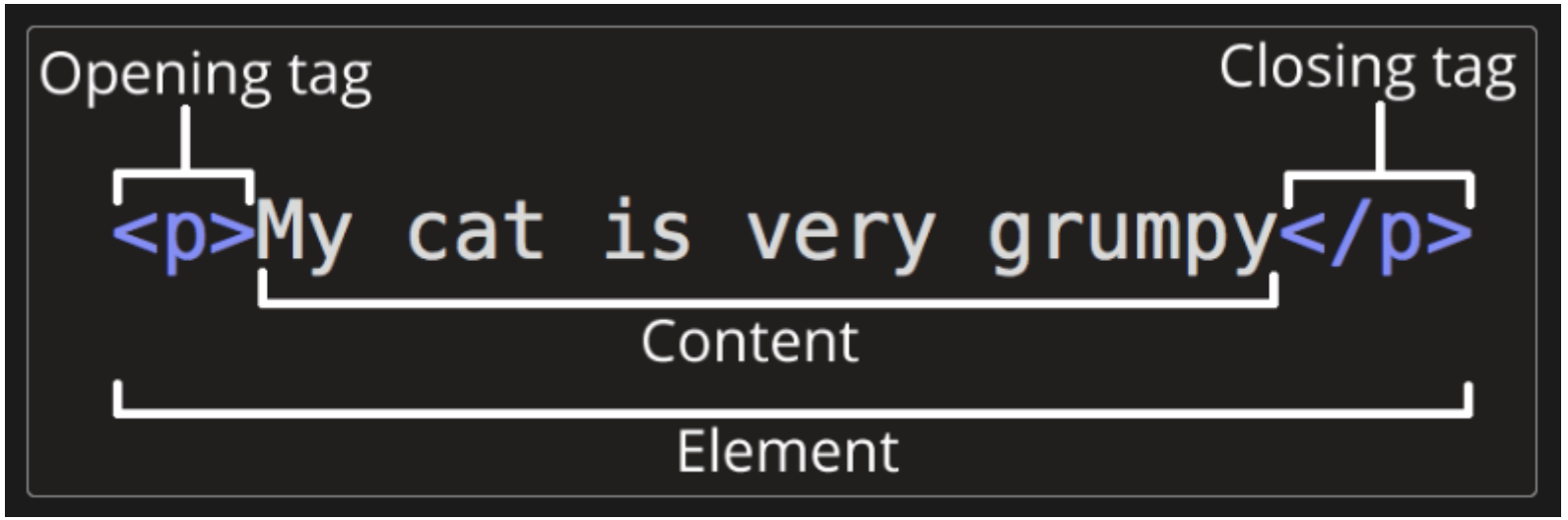
- HTML 문서의 메타 데이터와 문서 정보를 정의하는 데 사용되는 태그
  - 메타 데이터: 데이터에 대한 정보를 제공하는 데이터
- HTML 문서의 title, style, link, script, meta 에 대한 데이터로 화면에 표시되지 않음
  - title : HTML 문서의 제목
  - style : HTML 문서의 Style 정보 정의
  - link : 외부 리소스와의 연결 정보를 정의(CSS 파일 연계에 사용)
  - script : Javascript 를 정의
  - meta : 페이지 설명, 키워드, 저자, 화면 크기 등의 정보. 주로 브라우저 또는 검색 엔진에서 사용

<body> 태그

- HTML 문서의 실제 콘텐츠를 정의하는 데 사용되는 태그
- 다양한 태그를 사용하여 웹 페이지의 구조를 설계할 수 있다.

HTML 기본 문법, 요소(Element)

codingOn



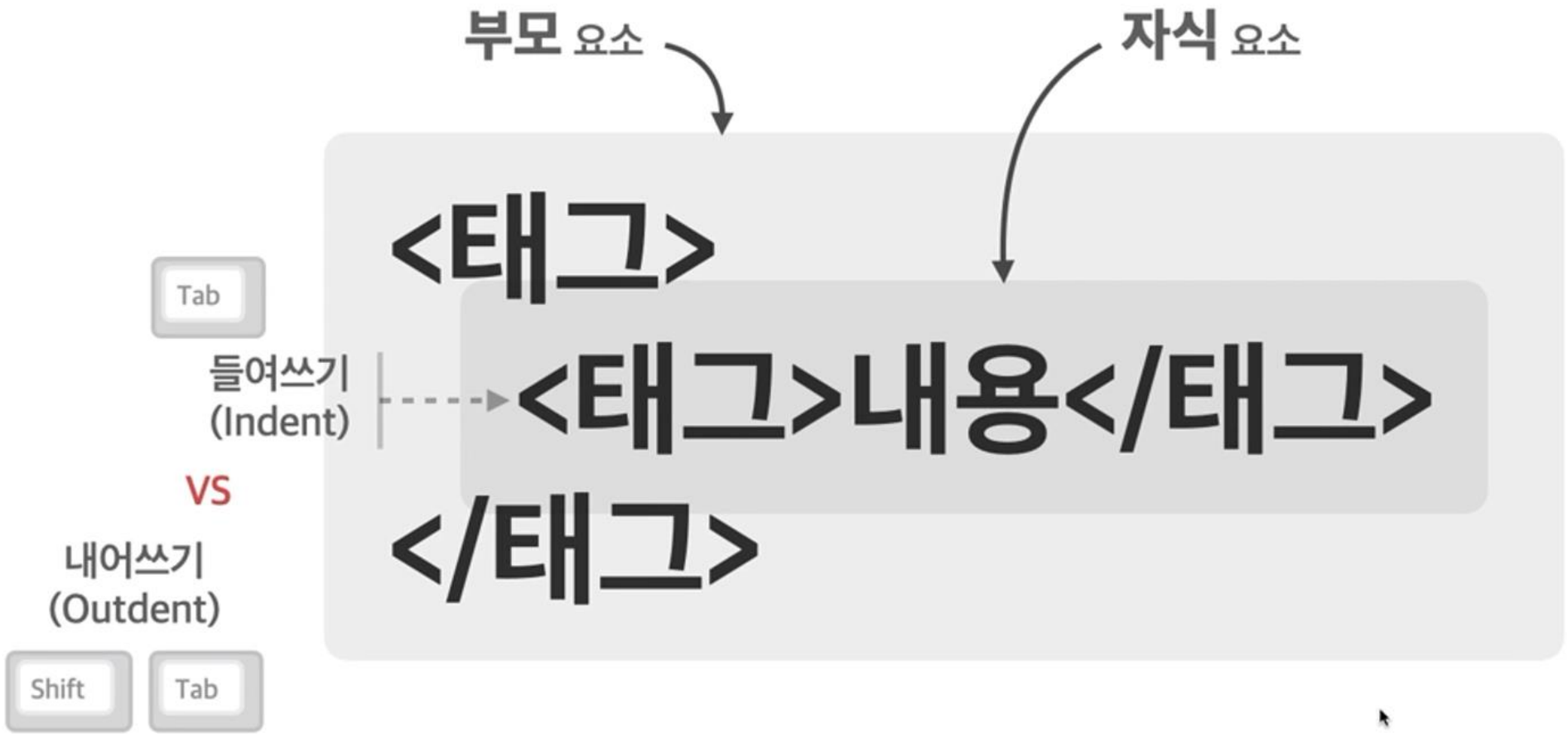
- HTML 요소(Element)는 시작 태그(Opening tag)와 종료 태그(Closing tag) 그리고 태그 사이에 위치한 내용(Content)로 구성

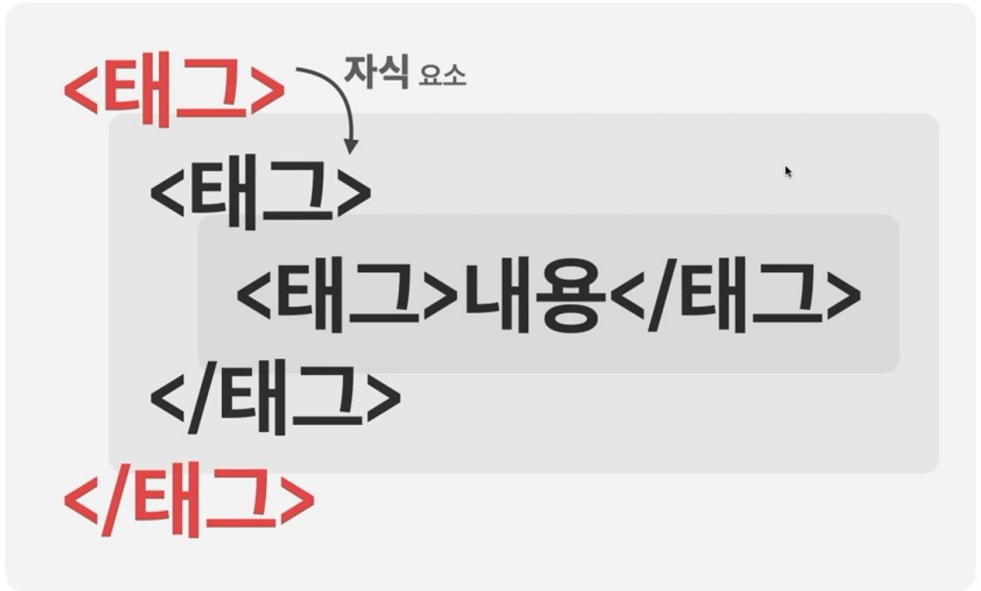
HTML 기본 문법, 중첩(Nested)

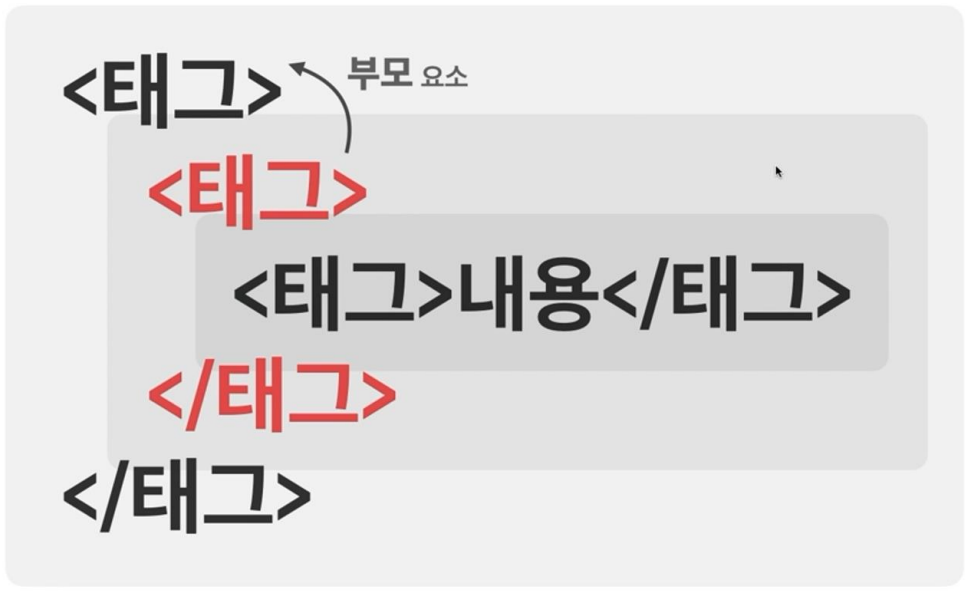
codingOn

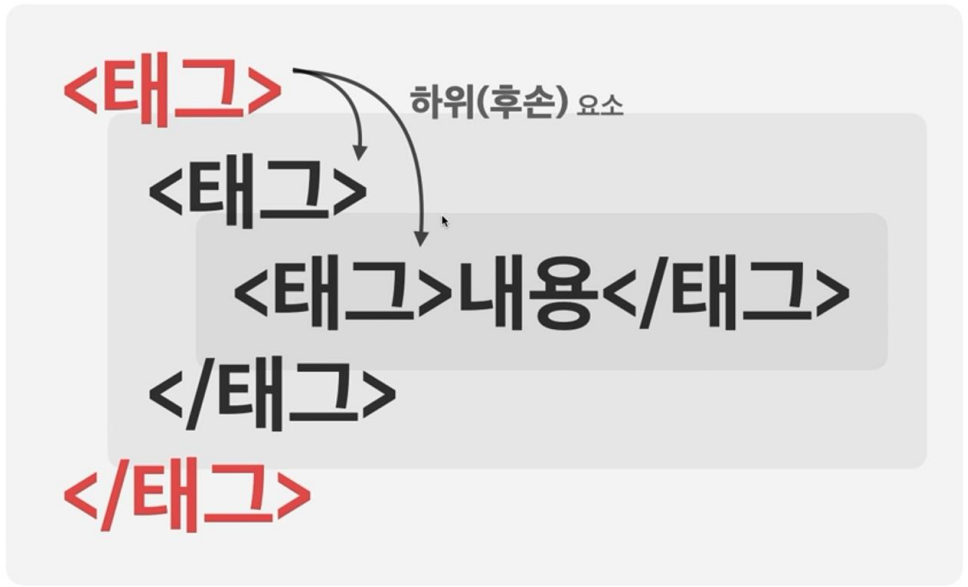
```
1  <!DOCTYPE html>
2  <html>
3    <head>
4      <meta charset="utf-8">
5      <title>Hello World</title>
6    </head>
7    <body>
8      <h1>Hello World</h1>
9      <p>안녕하세요! HTML5</p>
10   </body>
11 </html>
```

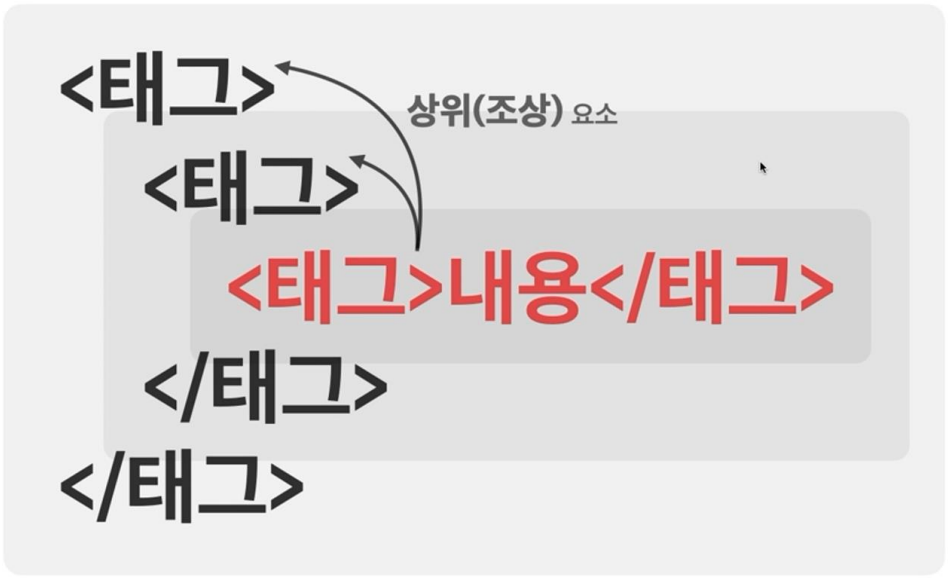
- <html></html>은 웹페이지를 구성하는 모든 요소들을 포함한다.
- 예제를 보면 <html>은 <head> 와 <body>를 포함하며 <head> 는 <meta> 와 <title> 요소를 <body> 요소는 <h1>, <p>를 포함
- 중첩 관계(부모자식 관계)로 웹페이지의 구조(structure)를 표현한다


부모 요소
자식 요소
<태그>
<태그>내용</태그>
</태그>
Tab
들여쓰기
(Indent)
VS
내어쓰기
(Outdent)
Shift
Tab


<태그>
자식 요소
<태그>
<태그>내용</태그>
</태그>
</태그>


<태그>
부모 요소
<태그>
<태그>내용</태그>
</태그>
</태그>


<태그>
하위(후손) 요소
<태그>
<태그>내용</태그>
</태그>
</태그>


<태그>
상위(조상) 요소
<태그>
<태그>내용</태그>
</태그>
</태그>

HTML 기본 문법, 빈 요소(Empty)

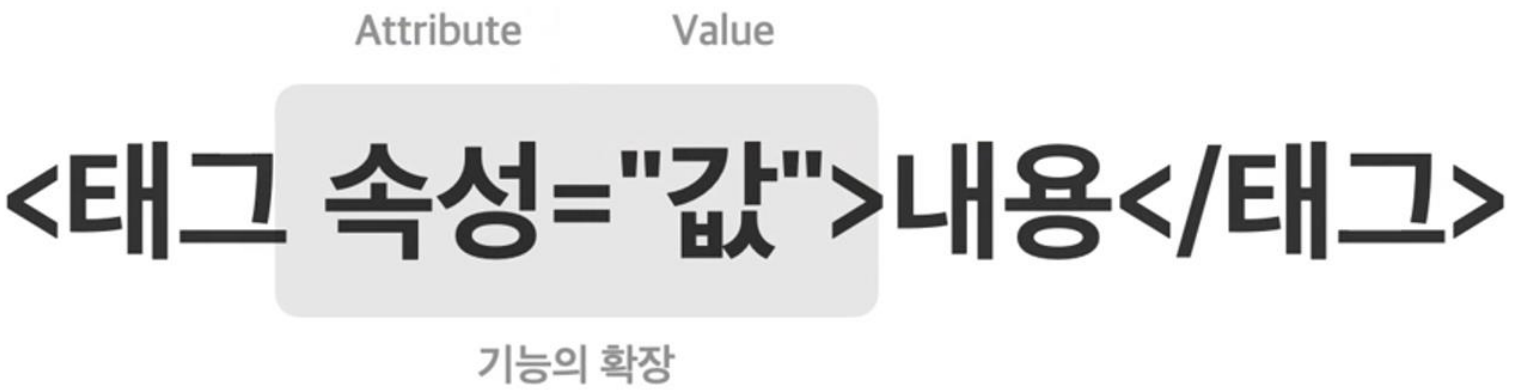
codingOn

```
1  <!DOCTYPE html>
2  <html>
3    <head>
4      <meta charset="utf-8">
5      <title>Hello World</title>
6    </head>
7    <body>
8      <h1>Hello World</h1>
9      <p>안녕하세요! HTML5</p>
10   </body>
11 </html>
```

- 내용이 없는 요소(가질 수 없거나, 필요 X)
- 빈 요소는 내용이 없으며, 속성만 소유
- 4번 라인의 <meta> 가 빈 요소!
  - 검색 키워드, 설명, 저자 등의 데이터를 설정
-
, <hr>, <img>, <input>, <link>, <meta> 등이 존재


편리함!
HTML 1/2/3/4/5
<태그>
VS
<태그 />
안전함!
XHTML / HTML5

HTML 기본 문법, 속성(Attribute)



- 속성(Attribute)이란 요소의 성질, 특징을 정의
- 속성은 요소에 추가적 정보(예를 들어 이미지 파일의 경로, 크기 등)를 제공한다.

제목 태그, <h1~6></h1~6>

codingOn

- 제목을 뜻하는 Heading 의 약자, h 사용!
- 자동 줄 바꿈! 왜? 제목이니까!
- 하나의 HTML 문서에는 하나의 h1 태그를 권장
- 웹 검색 엔진이 제일 먼저 검색하는 태그!

본문 태그,

- 본문을 뜻하는 paragraph 의 약자, 사용!
- 본문을 적기 위한 태그!

줄 바꿈 태그,

- 과연
은 무엇의 약자 일까요?
- 줄을 바꿔 준답니다!

목록, or

- : 순서 없는 목록 (unordered list)
- : 순서 있는 목록 (ordered list)

```
<!-- type 마커 모양 : disc, circle, square, none -->
<ul type="circle">
  <li>사과</li>
  <li>바나나</li>
  <li>파인애플</li>
</ul>
<!-- type : 1, A, a, I, i  -->
<!-- start : 시작 번호 -->
<!-- reversed : 역순으로 -->
<ol type="i" start="3" reversed>
  <li>1</li>
  <li>2</li>
  <li>3</li>
</ol>
```

수평 줄, <hr>

- 수평으로 된 줄을 그어 줍니다!

문자 꾸미기 태그들

codingOn

- : 두껍게!
- : 두껍게! + Semantic 한 의미를 지님
- : 이탤릭
- : Emphasized, 강조! 기울여서 표시됨
- <del></del> : 중간 줄!
- <u></u> : 밑 줄!

이미지를 넣어주는 <img>

codingOn

- 이미지를 넣을 때 사용!
- 속성 값 중 하나인, src 를 사용
- 이미지 로드가 안될 경우 alt 속성이 중요!
- 파일을 직접 가져오기, 인터넷 주소에서 가져오기 등등

HTML의 하이퍼링크, <a>

- Anchor 의 약자인 <a> 태그 사용
- 속성 값
  - href : Hypertext Reference 의 약자, 이동할 페이지의 링크
  - target : 링크 된 문서를 열었을 때 문서가 열릴 위치 표시
    - _blank : 새로운 탭에서
    - _self : 현재 탭에서 (기본 값)

입력 값 받기! <input>

codingOn

- type
  - button
  - text
  - password
  - checkbox
  - radio
  - date
  - color
  - range
  - file

버튼, type="button"

- 버튼을 생성
- 주로 특정 기능을 수행 시킬 때 사용

```
<input type="button" value="버튼" />
```

텍스트, type="text"

- 텍스트 입력 값을 받는 폼을 생성
- 우리가 입력하는 ID 입력 부분?
- 텍스트 값을 입력 받아 전달하는 기능

패스워드, type="password"

- 비밀번호 값을 받는 폼을 생성
- 입력 값을 자동으로 안보이게 처리
- 중요 or 비밀 텍스트 값을 전달하는 기능

체크 박스, type="checkbox"

codingOn

- 여러 선택지 중 여러 개를 선택 가능한 체크 박스 생성
- 속성
  - Name : 체크 박스의 이름, 같은 분류의 체크 박스는 같은 이름으로 설정
  - Value : 체크 박스가 실제로 전달하는 값을 지정
  - Checked : 화면 최초 로딩 시에 선택 된 상태로 로딩

라디오 버튼, type="radio"

codingOn

- 여러 선택지 중 하나만 선택 가능한 라디오 버튼 생성
- 속성
  - Name : 라디오 버튼의 이름, 같은 name 을 가지는 라디오 버튼은 하나만 선택이 가능
    → 하나를 선택하면 다른 선택 값이 취소 됨
  - Value : 라디오 버튼이 실제로 전달하는 값을 지정
  - Checked : 화면 최초 로딩 시에 선택 된 상태로 로딩

날짜 선택, type="date"

codingOn

- 특정 날짜(년/월/일)를 선택
- 속성
  - Name : 날짜 선택 폼 이름
- type="datetime-local"
  - 시간 까지 선택 가능!
  - 기존은 "datetime" 을 사용하였으나 시간은 시간대의 영향을 받기 때문에 정확한 데이터 값 전달이 불가능 하여 지금은 사용 X

선택 메뉴를 만드는, <select>

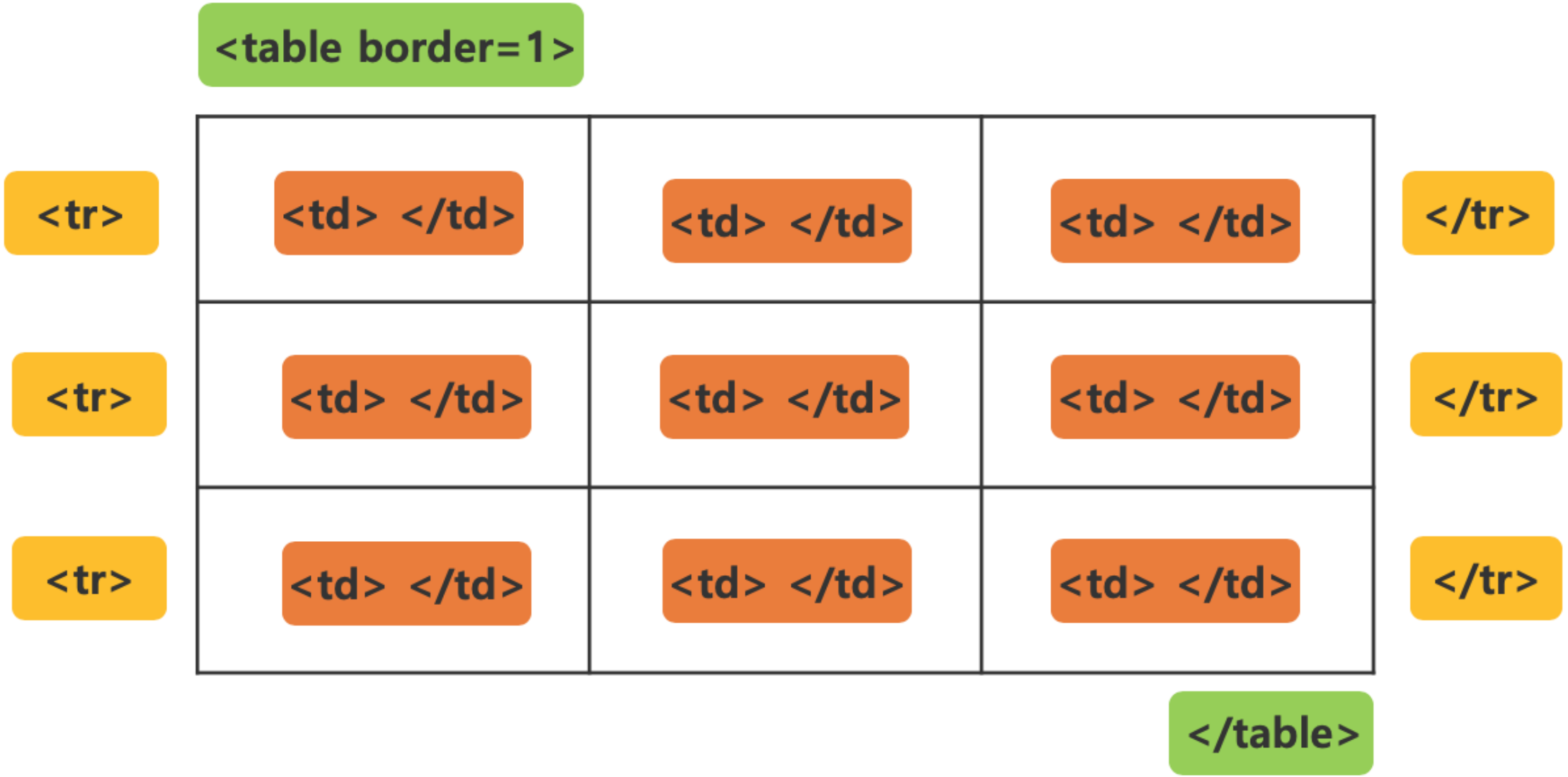
codingOn

- 선택 메뉴(드롭 다운)를 만드는 태그!
- <select> : select 폼 생성
  - Name : select 박스의 이름
- <option> : select 폼의 옵션 값 생성
  - Value : 실제적으로 전달 되는 값
  - Selected : 최초에 선택 된 값으로 설정
- <optgroup> : option 을 그룹화
  - Label : optgroup 이름 설정
- Disabled : 옵션은 보이지만 선택을 못하도록 설정

테이블을 만들자! <table>

codingOn

- 표를 만들 때 사용하는 태그
- 먼저 행을 쓰고 행의 자식 요소로 칸을 넣어주는 것이 기본!
  - <table> : 표를 감싸는 태그
  - <tr> : 표 내부의 행
  - <th> : 행 내부의 제목 칸
  - <td> : 행 내부의 일반 칸


<table border=1>
<tr>
<td> </td>
<td> </td>
<td> </td>
</tr>
<tr>
<td> </td>
<td> </td>
<td> </td>
</tr>
<tr>
<td> </td>
<td> </td>
<td> </td>
</tr>
</table>

<table> 속성

- <table>
  - **border** : 테두리 두께
  - **cellspacing** : 테두리 간격 사이의 너비
  - **cellpadding** : 셀 내부의 간격
  - **align** : 테이블 정렬 속성
  - **width 와 height** : 테이블의 너비와 높이
  - **bgcolor 와 bordercolor** : 테이블 배경색과, 테두리 색

<td> 속성

<td>

- **colspan** : 해당 칸이 점유하는 열의 수 지정
- **rowspan** : 해당 칸이 점유하는 행의 수 지정

CSS

CSS (Cascading Style Sheet)

codingOn

- 웹 페이지를 디자인하기위해 사용하는 언어

CSS 선택자

기본 | 복합 | 가상 클래스 | 가상 요소 | 속성

기본 선택자

- 기본 선택자니까 기본적인 선택자겠죠?
- 별도의 테크닉 없이, 순수하게 무엇인가를 호출 할 때 사용합니다!
- 종류
  - 전체 선택자
  - 태그 선택자
  - Class 선택자
  - ID 선택자

codingOn

*

기본 | 전체 선택자 (Universal Selector)

모든 요소를 선택.

```
<div>
  <ul>
    <li>사과</li>
    <li>딸기</li>
    <li>오렌지</li>
  </ul>
  <div>당근</div>
  <p>토마토</p>
  <span>오렌지</span>
</div>
```

선택

```
* {
  color: red;
}
```

codingOn

ABC

기본 | 태그 선택자 (Type Selector)

태그 이름이 ABC인 요소 선택.

```
<div>
  <ul>
    <li>사과</li>
    <li>딸기</li>
    <li>오렌지</li>
  </ul>
  <div>당근</div>
  <p>토마토</p>
  <span>오렌지</span>
</div>
```

선택

```
li {
  color: red;
}
```

codingOn

.ABC

기본 | 클래스 선택자 (Class Selector)

HTML class 속성의 값이 ABC인 요소 선택.

```
<div>
  <ul>
    <li>사과</li>
    <li>딸기</li>
    <li class="orange">오렌지</li>
  </ul>
  <div>당근</div>
  <p>토마토</p>
  <span class="orange">오렌지</span>
</div>
```

선택

```
.orange {
  color: red;
}
```

codingOn

#ABC

기본 | 아이디 선택자 (ID Selector)

HTML id 속성의 값이 ABC인 요소 선택.

```
<div>
  <ul>
    <li>사과</li>
    <li>딸기</li>
    <li id="orange" class="orange">오렌지</li>
  </ul>
  <div>당근</div>
  <p>토마토</p>
  <span class="orange">오렌지</span>
</div>
```

선택

```
#orange {
  color: red;
}
```

CSS 선택자

복합 선택자

- 특수한 요소를 호출하고 싶을 때, 기본 선택자만으로는 선택이 불가능한 경우에 사용
- 종류
  - 일치 선택자
  - 자식 선택자
  - 후손 선택자
  - 인접 형제 선택자
  - 일반 형제 선택자

ABCXYZ

복합 | 일치 선택자 (Basic Combinator)

선택자 ABC와 XYZ를 동시에 만족하는 요소 선택.

```
<div>
  <ul>
    <li>사과</li>
    <li>딸기</li>
    <li class="orange">오렌지</li>
  </ul>
  <div>당근</div>
  <p>토마토</p>
  <span class="orange">오렌지</span>
</div>
```

선택

```
span.orange {
  color: red;
}
```

codingOn

ABC > XYZ

복합 | 자식 선택자 (Child Combinator)

선택자 ABC의 자식 요소 XYZ 선택.

```
<div>
  <ul>
    <li>사과</li>
    <li>딸기</li>
    <li class="orange">오렌지</li>
  </ul>
  <div>당근</div>
  <p>토마토</p>
  <span class="orange">오렌지</span>
</div>
```

선택

```
ul > .orange {
  color: red;
}
```

ABC XYZ

복합 | 하위(후손) 선택자 (Descendant Combinator)

선택자 ABC의 **하위** 요소 XYZ 선택.
'띄어쓰기'가 선택자의 기호!

```
<div>
  <ul>
    <li>사과</li>
    <li>딸기</li>
    <li class="orange">오렌지</li>
  </ul>
  <div>당근</div>
  <p>토마토</p>
  <span class="orange">오렌지</span>
</div>
<span class="orange">오렌지</span>
```

선택

```
div .orange {
  color: red;
}
```

ABC + XYZ

복합 | 인접 형제 선택자 (Adjacent Sibling Combinator)

선택자 ABC의 다음 형제 요소 XYZ 하나를 선택.

```
.orange + li {
  color: red;
}
```

선택

```
<ul>
  <li>딸기</li>
  <li>수박</li>
  <li class="orange">오렌지</li>
  <li>망고</li>
  <li>사과</li>
</ul>
```

ABC ~ XYZ

복합 | 일반 형제 선택자 (General Sibling Combinator)

선택자 ABC의 다음 형제 요소 XYZ 모두를 선택.

```
.orange ~ li {
  color: red;
}
```

선택

```
<ul>
  <li>딸기</li>
  <li>수박</li>
  <li class="orange">오렌지</li>
  <li>망고</li>
  <li>사과</li>
</ul>
```

CSS 선택자

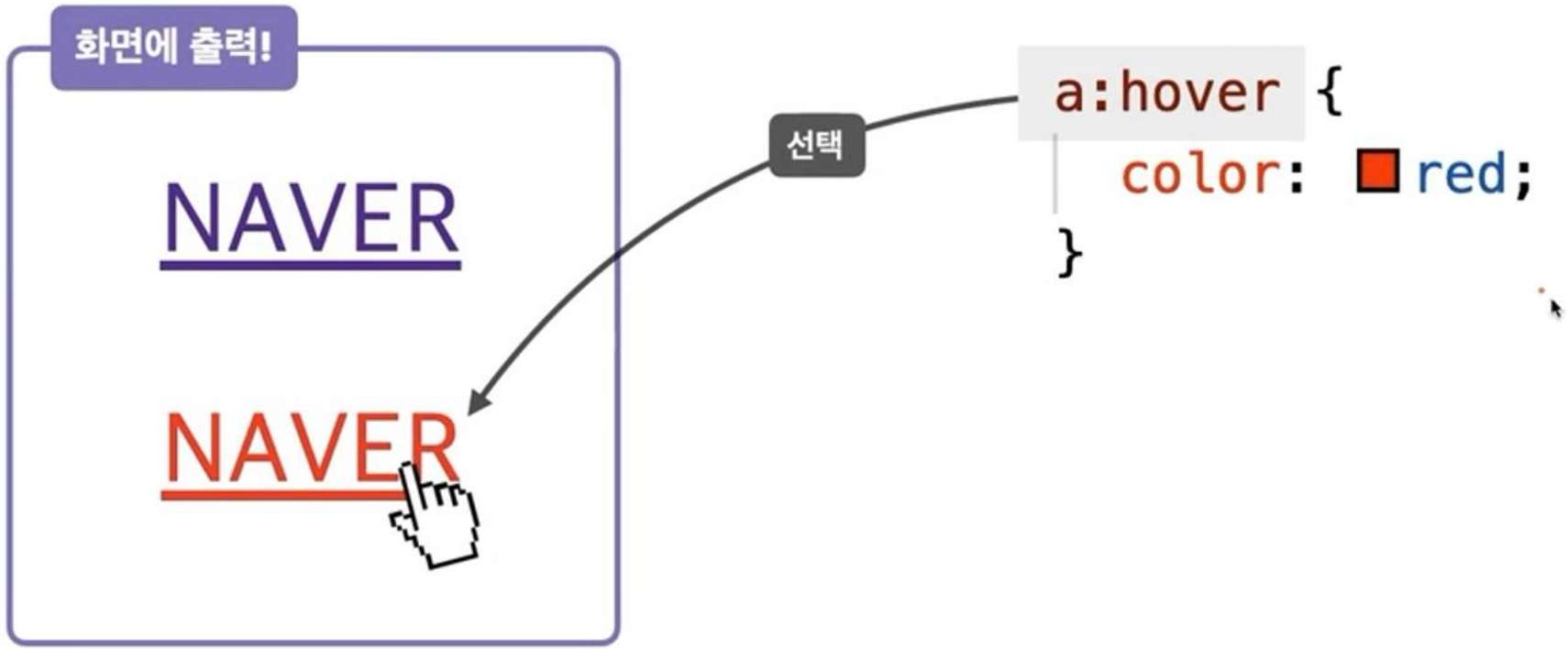
가상 클래스 선택자

- 사용자의 행동에 따라 변화하는 가상 상황에 따라서 요소 선택 시
- 각 요소의 상황에 따라 사용자가 원하는 요소를 선택 할 때 사용
- 특정 요소를 부정 할 때 사용
- 종류
  - 가상 클래스 선택자
  - 사용자의 행동에 따라 변화 : Hover, Active, Focus, …
  - 요소의 상황 : first-child, last-child, nth-child, …
  - 부정 선택 : not

ABC:hover

가상 클래스 선택자 (Pseudo-Classes) HOVER

선택자 ABC 요소에 마우스 커서가 올라가 있는 동안 선택.




```
<a href="https://www.naver.com">NAVER</a>
```

ABC:active

가상 클래스 선택자 (Pseudo-Classes) ACTIVE

선택자 ABC 요소에 마우스를 클릭하고 있는 동안 선택.

화면에 출력!

NAVER

NAVER

선택

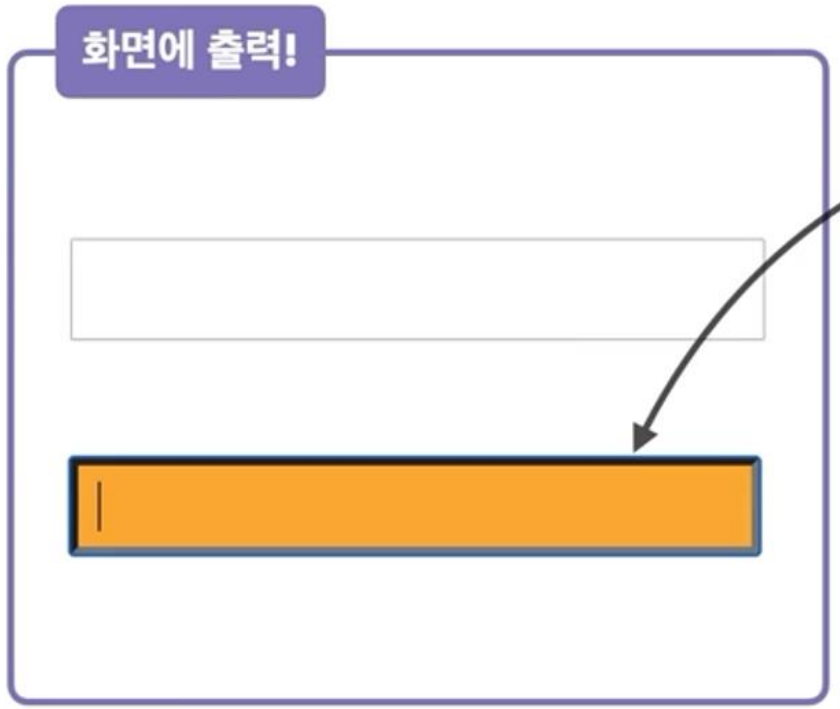
```
a:active {
    color: red;
}
```

```
<a href="https://www.naver.com">NAVER</a>
```

ABC:focus

가상 클래스 선택자 (Pseudo-Classes) FOCUS

선택자 ABC 요소가 포커스되면 선택.



선택

```
input:focus {
  background-color: orange;
}
```

```
<input type="text" />
```

ABC:first-child

가상 클래스 선택자 (Pseudo-Classes) FIRST CHILD

선택자 ABC가 형제 요소 중 첫째라면 선택.

```
<div class="fruits">
  <span>딸기</span>
  <span>수박</span>
  <div>오렌지</div>
  <p>망고</p>
  <h3>사과</h3>
</div>
```

선택

```
.fruits span:first-child {
  color: red;
}
```

?

```
.fruits div:first-child {
  color: red;
}
```

ABC:last-child

가상 클래스 선택자 (Pseudo-Classes) LAST CHILD

선택자 ABC가 형제 요소 중 막내라면 선택.

```
.fruits h3:last-child {
  color: red;
}
```

선택

```
<div class="fruits">
  <span>딸기</span>
  <span>수박</span>
  <div>오렌지</div>
  <p>망고</p>
  <h3>사과</h3>
</div>
```

ABC:nth-child(n)

가상 클래스 선택자 (Pseudo-Classes) NTH CHILD

선택자 ABC가 형제 요소 중 (n)째라면 선택.

```
.fruits *:nth-child(2) {
  color: red;
}
```

선택

```
<div class="fruits">
  <span>딸기</span>
  <span>수박</span>
  <div>오렌지</div>
  <p>망고</p>
  <h3>사과</h3>
</div>
```

ABC:nth-child(n)

가상 클래스 선택자 (Pseudo-Classes) | NTH CHILD

선택자 ABC가 형제 요소 중 (n)째라면 선택.

```
<div class="fruits">
  <span>딸기</span>
  <span>수박</span>
  <div>오렌지</div>
  <p>망고</p>
  <h3>사과</h3>
</div>
```

선택

```
.fruits *:nth-child(2n) {
  color: red;
}
```

n은 0부터 시작!
(Zero-Based Numbering)

```
.fruits *:nth-child(n+2) {
  color: red;
}
```

n은 0부터 시작!
(Zero-Based Numbering)

ABC:not(XYZ)

부정 선택자 (Negation) NOT

선택자 XYZ가 아닌 ABC 요소 선택.

```
.fruits *:not(span) {
  color: red;
}
```

선택

```
<div class="fruits">
  <span>딸기</span>
  <span>수박</span>
  <div>오렌지</div>
  <p>망고</p>
  <h3>사과</h3>
</div>
```

CSS 선택자

기본 복합 가상 클래스 가상 요소 속성

가상 요소 선택자

- 선택 된 요소의 앞, 뒤에 별도의 Content 를 삽입하는 선택자
- 반드시 content 라는 속성을 사용
- 빈 값(“”) 이라도 넣어 주어야 적용이 됨
- 종류
  - After : 요소의 뒤에 내용 삽입
  - Before : 요소의 앞에 내용 삽입

ABC::before

가상 요소 선택자 (Pseudo-Elements) BEFORE

선택자 ABC 요소의 내부 앞에 내용(Content)을 삽입.

인라인(글자) 요소

```
.box::before {
  content: "앞!";
}
```

화면에 출력!

앞! Content!

::before

```
<div class="box">
  
  Content!

</div>
```

ABC::after

가상 요소 선택자 (Pseudo-Elements) AFTER

선택자 ABC 요소의 내부 뒤에 내용(Content)을 삽입.

인라인(글자) 요소

```
.box::after {
  content: "뒤!";
}
```

화면에 출력!

Content! 뒤!

::after

```
<div class="box">

  Content!

</div>
```

가상 요소 선택자!

- 실제로 의미 없는 HTML 태그를 만들지 않고 요소 삽입이 가능하여 자주 사용!
- 예를 들어 쇼핑몰 페이지에 메뉴에 Hot, 추천 등을 넣기 위해 별도의 태그를 삽입하는 것이 아니라 가상 요소 선택자를 활용하여 처리하면 편리함!

CSS 선택자

기본 복합 가상 클래스 가상 요소 속성

속성 선택자

- 지정한 특정 속성을 가지고 있는 태그를 선택하는 선택자
- 종류
  - 특정 속성만 지정
  - 속성과 속성의 값을 지정

[ABC]

속성 선택자 (Attribute) ATTR

속성 ABC을 포함한 요소 선택

```
[disabled] {
  color: red;
}
```

선택

```
<input type="text" value="HEROPY">
<input type="password" value="1234">
<input type="text" value="ABCD" disabled>
```

[ABC="XYZ"]

속성 선택자 (Attribute) ATTR=VALUE

속성 ABC을 포함하고 값이 XYZ인 요소 선택.

```
[type="password"] {
  color: red;
}
```

선택

```
<input type="text" value="HEROPY">
<input type="password" value="1234">
<input type="text" value="ABCD" disabled>
```

복.습.철.저

수고하셨습니다