Web Programming For webOS

LG전자 오재덕

Overview

Web 과 Native 의 차이에 대해서 알아보고, client / server의 개념과 Front-end/Back-end 개발에 대해서 알아본다

Web 개발 vs 네이티브 개발

Web 개발의 장단점

장점	단점
크로스 플랫폼 지원 (브라우저만 있으면)	낮은 성능
배포의 용이성	제한된 기능
비용 효율성	오프라인 기능의 제약
빠른 개발	

단점 보완을 위한 기술

- PWA (Progressive Web Apps)
  - 오프라인 지원 / 성능향상 / 설치 가능
- Wasm (WebAssembly)
  - 브라우저에서 고성능 실행을 가능하게 하는 이진형식의 코드
  - 성능향상 / 복잡한 계산 처리
- Service Workers
  - 브라우저의 백그라운드에서 실행되는 스크립트
  - 오프라인 지원 / 빠른 응답
- SPA (Single Page Application)
  - 하나의 페이지에서 동적으로 콘텐츠를 로드
  - 빠른 전환
- 등등

웹의 동작원리

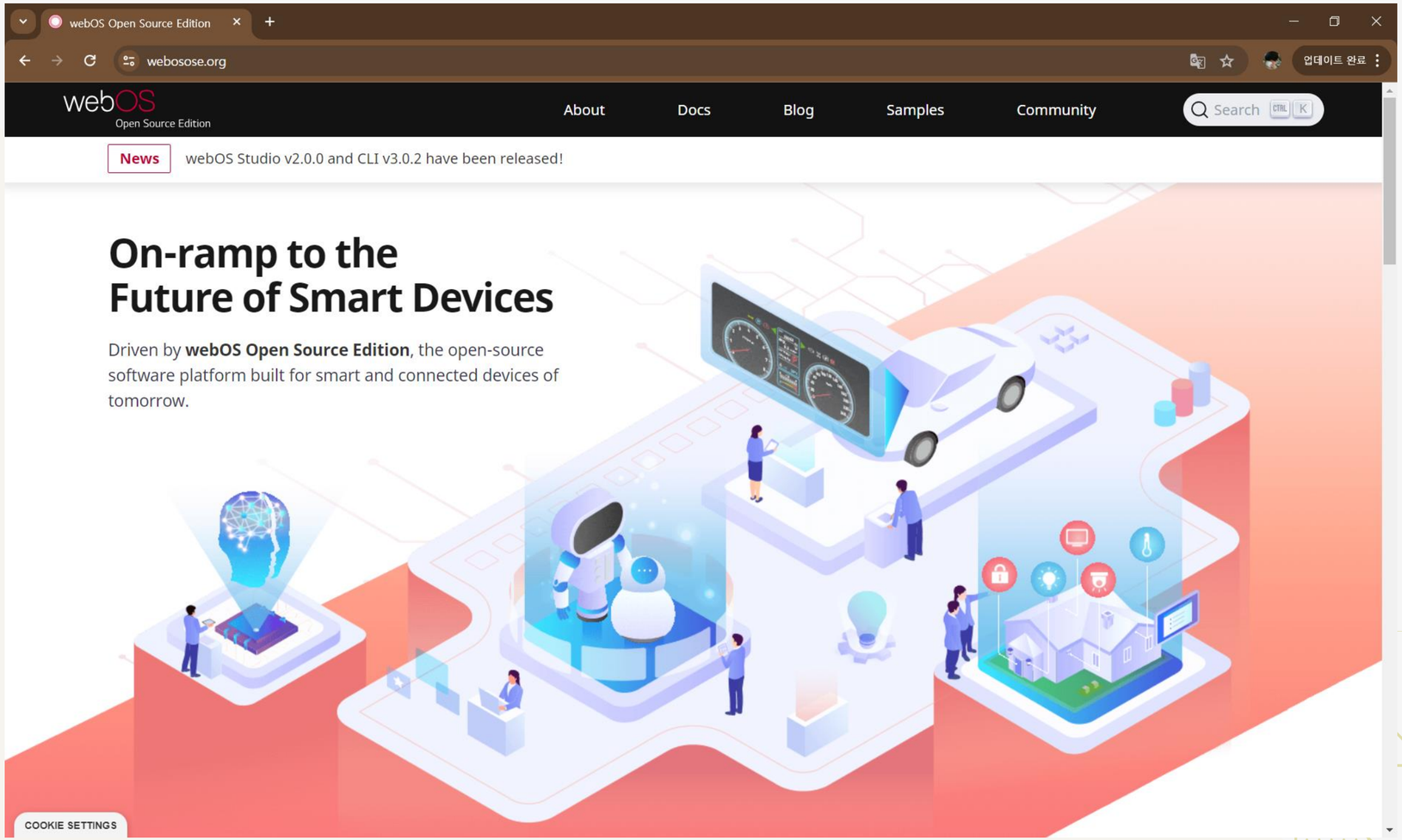
서버 (Server)

클라이언트로부터 받은 요청을 처리하여
데이터를 응답(HTTPResponse)

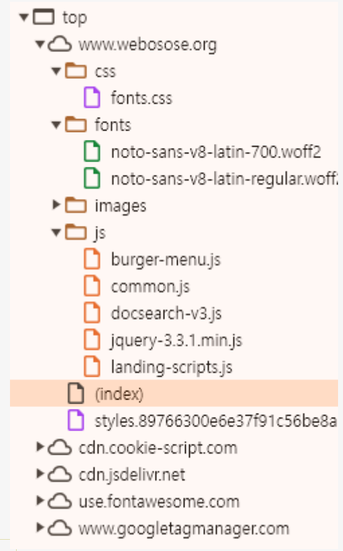
클라이언트 (Client)

사용자가 웹 브라우저를 통해 서버에게
필요한 정보를 요청(HTTPRequest)

웹의 동작원리



서버로부터 받은 리소스들로 구성한 화면



“website” vs “web application”

Website	Web Application
정적 콘텐츠 중심	동적 기능 중심
간단한 구조	복잡한 구조
상호작용이 적음	상호작용이 많음
HTML, CSS, JavaScript	HTML, CSS, JavaScript, 서버 프로그래밍, 데이터 베이스 등

이미지출처: https://redapplelearning.in/what-is-the-difference-between-web-application-and-website/

Front-End 개발

역할

- 사용자 인터페이스 (User Interface) 구현 : website나 web application의 시각적 요소를 구현
- 사용자 경험(User Experience) 개선 : 사용자가 쉽게 접근할 수 있도록 web page를 디자인
- 브라우저에서 실행

주요 기술

- HTML (HyperText Markup Language) : web page의 **구조**를 정의 (제목 / 단락 / 이미지 등)
- CSS (Cascading Style Sheet) : web page의 **스타일**을 정의 (색상 / 글꼴 / 레이아웃 등)
- JavaScript : web page의 **동적인 기능** 추가 (애니메이션, 사용자 상호작용 등)
- 프레임워크 및 라이브러리
  - Enact / React / Angular / Vue.js
- 도구 및 빌드 시스템
  - Webpack / Babel

Back-End 개발

역할

- 서버 관리 : web server를 설정하고 유지
- 데이터 베이스 관리 : 데이터를 저장, 관리, 검색
- 비즈니스 로직 처리 : web application의 핵심 기능을 처리 (사용자인증, 데이터 처리, API 통신 등)

주요 기술

- 서버 측 언어
  - Node.js / Python (Django, Flask) / Java (Spring)
- 데이터 베이스 (DB, Database)
  - SQL DB (MySQL, PostgreSQL, Oracle 등)
  - NoSQL DB (MongoDB, Redis 등)
- 서버 및 호스팅
  - Apache / Nginx / AWS, Azure 등

따라해보기

A VisualStudio Code 설치

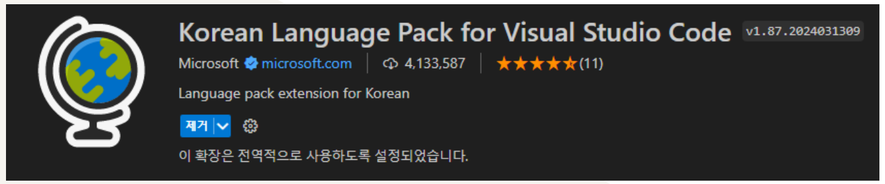
B Index.html 작성

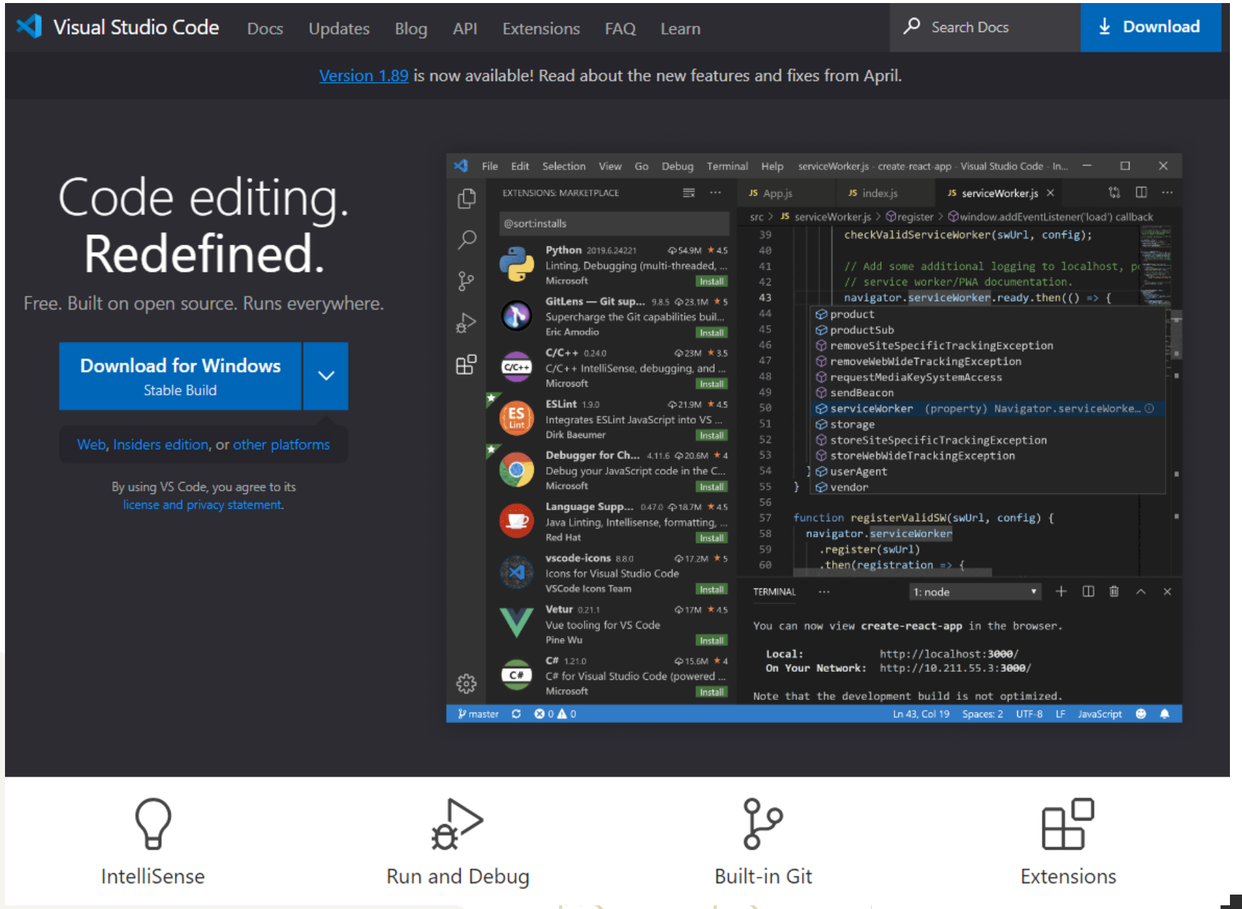
C LiveServer 실행

Visual Studio Code 설치

설치파일 내려받아 설치

- https://code.visualstudio.com/
- Host의 OS에 맞는 버전을 설치
- (필요 시) 한국어 언어팩 설치 - Extension

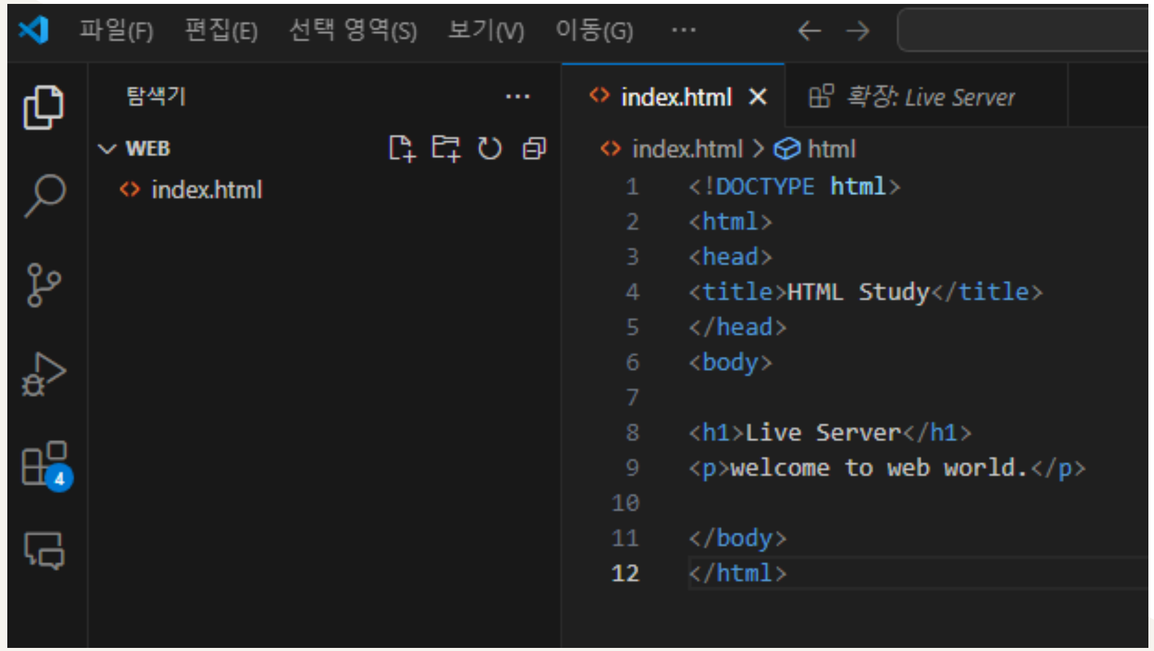




HTML 파일

간단한 코드 작성

- 새 파일을 열고 index.html 파일 생성

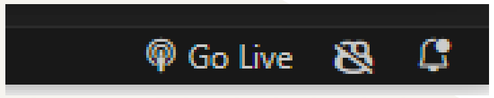



```
<!DOCTYPE html>
<html>
<head>
<title>HTML Study</title>
</head>
<body>

<h1>Live Server</h1>
<p>welcome to web world.</p>

</body>
</html>
```

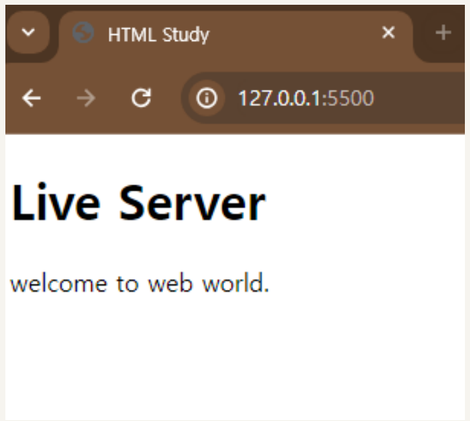
Live Server

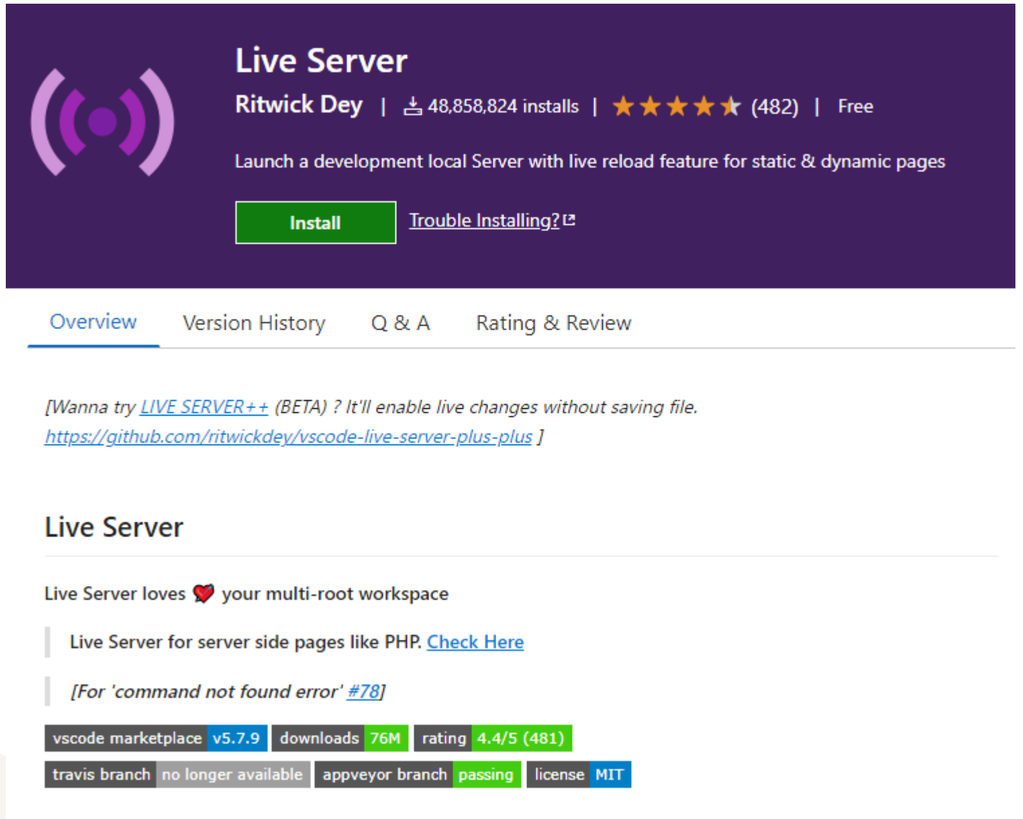
VS Code의 Extension 중 하나

- VS Code 마켓플레이스 (https://marketplace.visualstudio.com/items?itemName=ritwickdey.LiveServer)
- VS Code 에서 바로 서버를 실행 (5500번 포트)



- 기본 브라우저에서 실행





Overview - Summary

- 웹 개발 vs 네이티브 개발
- 클라이언트 – 서버 모델
- 웹사이트 vs 웹 애플리케이션
- 프론트엔드와 백엔드 개발
- IDE - Visual Studio Code
- Live Server Extension

HTML

HTML 기본 개념에 대해서 알아본다

HTML의 역할과 기본 구조

HTML이란

- HTML(Hyper Text Markup Language): 웹 페이지의 구조를 정의하는 마크업 언어

주요 역할

- 웹 페이지의 구조 정의 – 제목, 단락, 목록, 테이블, 이미지, 링크 등의 요소를 정의
- 콘텐츠 표시 – 텍스트, 이미지, 비디오 등의 다양한 콘텐츠를 웹 페이지에 포함
- 하이퍼 링크 생성 – 다른 웹 페이지나 리소스로의 링크를 생성하여 페이지 네비게이션을 가능하게 함
- 웹 폼 지원 – 사용자 입력을 위한 폼 요소 제공
- SEO 최적화 – 검색엔진 최적화(SEO)를 위해 중요한 정보를 구조화, 메타 데이터를 제공

HTML의 역할과 기본 구조

HTML 기본 구조

```
<!DOCTYPE html>
<html>
<head>
        <title>HTML Study</title>
</head>
<body>

    <h1>Live Server</h1>
    <p>welcome to web world.</p>
    <a href="https://webosose.org"> webOS
    OSE DevSite</a>
    <img src="image.jpg" alt="Description of
    image">

</body>
</html>
```

- <!DOCTYPE html> : HTML5 문서 선언
- <html> : HTML 문서의 루트 요소
- <head> : 메타 데이터를 포함하는 요소
  - 페이지 제목, 문자 인코딩, 외부 파일 링크
  - CSS / JavaScript 파일
- <title> : 웹 페이지의 제목, 브라우저의 탭에 표시되는 제목
- <body> : 사용자에게 표시되는 실제 콘텐츠
  - 텍스트, 이미지, 링크 등

HTML5란?

HTML5란

- HTML의 최신 표준으로 2014년에 W3C에서 공식 채택

주요 특징 및 새로운 기능

- 새로운 시맨틱 요소: <header>, <footer>, <article>, <section>, <nav> 등
- 폼 요소의 향상: <input>의 새로운 입력 타입 및 속성 지원
- 멀티미디어 요소: <audio>, <video>
- 그래픽 및 애니메이션: <canvas>, SVG(Scalable Vector Graphics) 지원
- 오프라인 및 저장: Web Storage (Local Storage / Session Storage), Application Cache
- 새로운 API: Geolocation API, Web Workers, WebSocket API

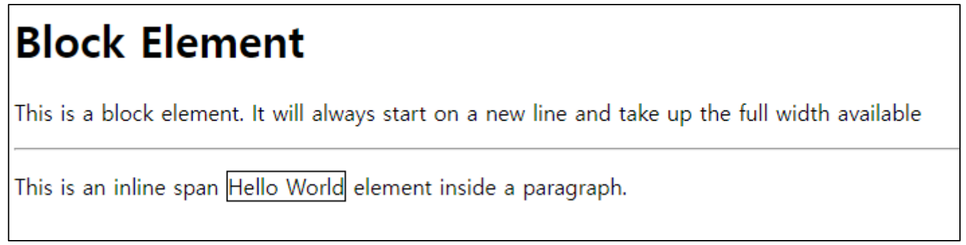
블록 요소와 인라인 요소

블록 요소

- 새로운 줄에서 시작, 가로 방향으로 가능한 모든 공간 차지
- 기본 스타일: display:block;
- 예시
  - ,<p>,<h1>~<h6>,<ul>,<ol>,<li>,<header>,<footer>, <section>, <article>

인라인 요소

- 같은 줄에 배치, 텍스트나 짧은 콘텐츠 조각에 사용, 콘텐츠 너비 만큼 공간 차지, 블록 요소 안에 포함 가능, 인라인 요소에는 블록 요소 포함 불가
- 기본 스타일: display:inline;
- 예시
  - ,<a>,<img>,,,<input>,<label>




```
<div>
   <h1> Block Element</h1>
   <p> This is a block element. It will always start on a new line and take up the full width available</p>
<div>
```

```
<p>This is an inline span <span style="border: 1px solid black">Hello World</span> element inside a paragraph.</p>
```

HTML 요소

HTML 요소 (HTML Element)

- HTML 요소는 시작 태그와 콘텐츠, 그리고 종료 태그로 구성
  - <tagname>Content goes here...</tagname>
- 종료 태그에 유의 (종료 태그를 잊어버리면 예상치 못한 결과 및 오류가 발생할 수 있음)
- Case Sensitive는 아니지만, lowercase 를 권장 (W3C)

```
<p>This is a <span style="color:red">second paragraph.
<p>This is a <span style="color:red">third</span> paragraph.
```

중첩된 HTML 요소 (Nested HTML Element)

- HTML 요소는 중첩 될 수 있음. 즉, 한 요소가 다른 요소에 포함 될 수 있음

```
<p>This is an inline span <span style="border: 1px solid black">Hello World</span> element inside a paragraph.</p>
```

빈 HTML 요소 (Empty HTML Element)

- 콘텐츠 없이 사용하는 HTML 요소
- 예)

HTML 속성

HTML 속성 (HTML Attribute)

- HTML 요소에 추가 정보를 제공하는데 사용
- 항상 시작 태그 내에 작성하며, 이름과 값으로 이루어짐
  - name=“value”
- <element attribute=“value”> Content </element>

주요 속성

- 전역 속성 (Global Attributes)
- 폼 관련 속성 (Form Attributes)
- 하이퍼링크 및 미디어 관련 속성 (Hyperlink and Media Attributes)
- 테이블 관련 속성 (Table Attributes)

HTML 속성 - 전역 속성

주요 속성 값

- id: 고유 식별자
- class: 클래스 이름
- style: 인라인 스타일
- title: 툴팁 정보
- data-(데이터 속성)*: 사용자 정의 데이터 저장, JavaSacript로 접근

This is a red heading

WHO
User Info

```
<div id=“container” class=“main_container”></div>
```

```
<h1 style=“color:red”>This is a red heading</h1>
```

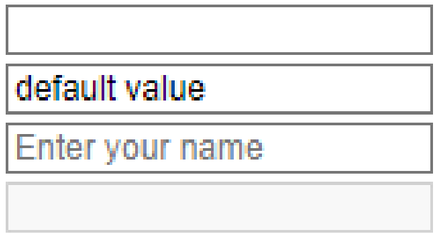
```
<abbr title="World Health Organization“>WHO</abbr>
```

```
<div data-user-id=“12345”>User Info</div>
```

HTML 속성 – 폼 관련 속성

주요 속성 값

- type: 입력 타입
  - text, password, submit, checkbox, radio
- name: 폼 요소 이름
- value: 초기 값
- placeholder: 입력 힌트
- required: 필수 입력 필드
- readonly: 읽기 전용 필드
- disabled: 비활성화 필드




```
<input type="text" name="username">
```

```
<input type="text" readonly value="default value">
```

```
<input type="text" placeholder="Enter your name" required>
```

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

HTML 속성 – 하이퍼링크 및 미디어, 테이블 속성

하이퍼링크 및 미디어 속성 값

- href: 링크 URL
- src: 소스파일 URL
- alt: 대체 텍스트
- width 및 height: 너비와 높이
- target: 링크 열기 위치

```
<a href=“https://www.webososе.org” target=“_blank”>
webOS OSE </a>
<img
src="https://www.webososе.org/images/webOS_OSE_lo
go.jpg" alt="webOS OSE Logo" width="200"
height="100">
```

테이블 관련 주요 속성 값

- colspan: 셀의 가로 병합 개수를 정의
- rowspan: 셀의 세로 병합 개수를 정의

```
<td colspan=“2”>Merged Cell</td>
<td rowspan=“2”>Merged Cell</td>
```

webOS OSE

Company	Contact
Alfreds Futterkiste	Maria Anders
Centro comercial Moctezuma	Francisco Chang
Ernst Handel	
N/A	

HTML 테이블

- 데이터를 행과 열로 정렬할 수 있음

테이블 요소

- table: 테이블
- tr: 행
- td: 데이터 칸
- th: 헤더 칸
- caption: 제목/ 설명
- col: 열
- colgroup: 열 그룹
- thead: 헤더 그룹
- tbody: 본문 그룹
- tfoot: 푸터 그룹

Company	Contact
Alfreds Futterkiste	Maria Anders
Centro comercial Moctezuma	Francisco Chang

```
<style>
table, th, td {
  border:1px solid black;
}
</style>
-------------------------------------------------------
<table style="width:100%">
  <tr>
    <th>Company</th>
    <th>Contact</th>
</tr>
  <tr>
    <td>Alfreds Futterkiste</td>
    <td>Maria Anders</td>
</tr>
  <tr>
    <td>Centro comercial Moctezuma</td>
    <td>Francisco Chang</td>
</tr>
</table>
```

HTML 주석

- 코드 내 설명 추가
- 특정 코드를 임시로 비 활성화

기본 형식

- "<!--” 로 시작하고 "-->“ 종료
  - <!-- Write your comments here -->

주의 사항

- 코멘트 중첩 금지: 중첩된 코멘트는 오류 발생
- 보안과 개인정보 보호: 민감한 정보 포함 금지
- 파일 크기 증가: 과도한 코멘트 사용 자제

This is a paragraph.

```
<!-- This is a comment -->
<p>This is <!-- chunk of --> a paragraph.</p>

<!-- Remember to add more information here -->

<!--
<p>This is another paragraph </p>
<img border="0" src="pic_trulli.jpg" alt="Trulli">
-->
```

```
<!-- This is a comment <!-- This is a nested comment --> -->
```

HTML 레이아웃

- 웹 페이지의 구조와 배치를 정의
- 콘텐츠를 논리적이고 시각적으로 그룹화

주요 레이아웃 요소

- : 블록 레벨 컨테이너
- : 인라인 컨테이너

시맨틱 레이아웃 요소

- <header>: 문서나 섹션의 헤더 (제목, 로고, 내비게이션 링크를 포함)
- <nav>: 내비게이션 링크를 그룹화
- <section>: 문서의 독립적인 섹션
- <article>: 독립적이고 자급자족적인 콘텐츠
- <aside>: 부가적인 콘텐츠
- <footer>: 문서나 섹션의 바닥글
- <main>, <details>, <summary>,<time> 등등

HTML - Summary

- HTML이란? 웹 페이지의 구조를 정의하는 마크업 언어
- 블록 요소 / 인라인 요소
- HTML 요소 (Element)
- HTML 속성 (Attribute)
  - 전역 속성
  - 폼 관련 속성
  - 하이퍼링크, 미디어, 테이블 속성
- HTML 테이블
- HTML 주석
- HTML 레이아웃
- 참고
  - w3schools.com : https://www.w3schools.com/html/default.asp
  - 생활코딩: https://opentutorials.org/course/3084
  - 모질라: https://developer.mozilla.org/en-US/docs/Learn/HTML

CSS 스타일링

CSS로 웹 페이지를 꾸미는 방법을 알아본다

CSS의 역할과 작동 원리

CSS(Cascading Style Sheet)란?

- 웹 페이지의 스타일과 레이아웃을 정의하는 스타일시트 언어
- HTML로 작성된 문서의 외관을 꾸미고 디자인

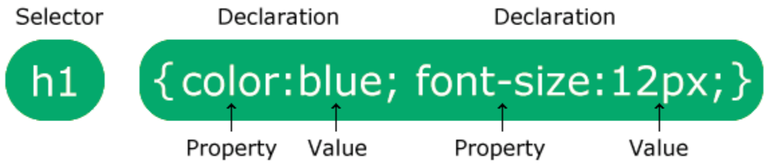
주요 역할

- 디자인 및 레이아웃 설정: 글꼴, 생상, 배경, 여백, 테두리 등 웹 페이지의 시각적 요소를 설정
- 반응형 웹 디자인: 다양한 화면 크기와 장치에 맞춰 레이아웃을 조정하여 사용자 경험을 향상
- 일관된 스타일 적용: 여러 페이지에 걸쳐 일관된 스타일을 유지 할 수 있음
- 프레젠테이션 분리: 콘텐츠(HTML)과 프레젠테이션(CSS)을 분리하여 코드의 유지보수성 및 가독성을 높임

CSS 기본 문법

기본 문법

- CSS는 선택자와 선언 블록으로 구성
  - 선택자는 스타일을 적용할 HTML 요소를 지정
  - 선언 블록은 한 쌍의 중괄호 {} 안에 CSS 속성과 값을 정의
  - 선언 블록은 하나 이상의 CSS 속성과 값을 가질 수 있음



용어 설명

- 선택자 (Selector) : 스타일을 적용할 HTML 요소
- 선언 (Declaration) : 세미콜론으로 구분된 하나 이상의 선언
- 속성 (Property) : CSS 속성 이름
- 값 (Value) : CSS 속성에 할당할 값

선택자, 속성, 값의 이해

선택자

- 요소(Element) 선택자: 특정 HTML 요소에 스타일을 적용 (예: h1, p, div)
- 클래스(Class) 선택자: 클래스 속성을 가진 요소에 스타일을 적용 (예: .classname)
- 아이디(Id) 선택자: 아이디 속성을 가진 요소에 스타일을 적용 (예: #idname)
- 속성(Property) 선택자: 특정 속성을 가진 요소에 스타일을 적용 (예: [type="text"])

```
h1 {
    color : blue;
    font-size: 24px;
}
```

```
h1, h2, p  {
    color : blue;
    font-size: 24px;
}
```

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

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

```
#idname {
    margin: 10px;
}
```

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

```
[type="text"] {
    text-align : center;
}
```

```
[class*="te"] {
  background: yellow;
}
```

선택자, 속성, 값의 이해

상속

- 상위 요소의 스타일을 하위 요소가 상속

HTML

```
...
<body>
   <p>Hello World!</p>
</body>
...
```

CSS

```
body {
   font-family: Arial;
}
P {
   color:gray;
}
```

계단식 (Cascading)

- 여러 스타일 규칙이 적용될 때 우선순위를 결정하는 계단식 구조를 가짐
  - 특이성: 아이디 선택자 > 클래스 선택자 > 요소 선택자 순
  - 출처: 브라우저 기본 스타일 > 사용자 스타일 > 작성자 스타일 순
  - 중첩: 마지막에 선언된 규칙이 우선 적용

CSS

```
#unique { color:blue; }
.class { color:red; }
div { color:green; }
```

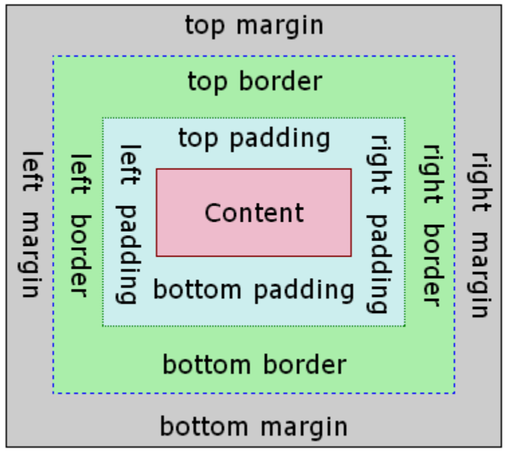
선택자, 속성, 값의 이해

박스 모델

- 콘텐츠, 패딩, 테두리, 마진으로 구성

```
div {
   width: 300px;
   padding: 20px;
   border: 5px solid black;
   margin: 10px;
}
```

- 콘텐츠 (Content): 텍스트나 이미지가 보여지는 영역
- 패딩 (Padding): 콘텐트 주위의 영역, 투명함
- 테두리 (Border): 패딩과 콘텐츠를 두른 테두리
- 마진 (Margin): 보더의 바깥쪽




```
div {
   width: 300px;
   padding: 20px;
   border: 5px solid black;
   margin: 10px;
}
```

< How to use >
padding : top padding, right padding, bottom padding, left padding;
padding: top padding, right and left padding, bottom padding;
Padding: top and bottom padding, right and left padding;

```
div {
  top-padding: 10px;
  right-padding: 20px;
  bottom-padding: 30px;
  left-padding: 20px;
}
```

==

```
div {
  padding: 10px 20px 30px;
}
```

CSS 적용

외부 CSS 파일 사용

- CSS가 외부에 정의되어 있고 HTML 페이지는 이를 참조
- 동일한 스타일을 반복 적용 가능
- <head>섹션에 <link> 요소에 외부 스타일을 정의

```
<head>
<link rel="stylesheet" href="mystyle.css">
</head>
```

내부 CSS 사용

- CSS가 HTML 코드 내에 정의
- 단일 페이지에서만 적용
- <head>섹션에 <style> 요소에 내부 스타일을 정의

```
<head>
<style>
body {
  background-color: linen;
}
</style>
</head>
```

인라인 CSS 사용

- CSS가 HTML 요소 내에 정의
- 스타일이 적용된 요소에만 적용
- HTML요소에 style 속성으로 인라인 스타일을 정의

```
<h1 style="color:blue;text-
align:center;">This is a heading</h1>
```

CSS 주석 달기

주석 (Comment)

- CSS 주석은 '/*' 로 시작하고 '*/' 로 끝남

```
/* This is a single-line comment */
p {
    color:gray;
}
```

```
p {
    color:gray; /* Set text color to red */
}
```

```
p {
    color: /*red*/gray;
}
```

```
/* This is
a multi-line
Comment */
p {
    color: gray;
}
```

CSS 프레임워크

다양한 프레임워크

- Bootstrap
  - https://getbootstrap.kr/
  - Twitter에서 개발한 가장 인기있는 CSS 프레임워크
- Tailwind CSS
  - https://tailwindcss.com/
  - 미리 정의된 클래스를 사용하여 빠르게 스타일링 할 수 있음
  - HTML 클래스 속성을 통해 스타일 적용
- Foundation
  - https://get.foundation/
  - ZURB에서 개발한 CSS 프레임워크
  - 사용하기 쉬운 그리드 시스템, 다양한 UI 컴포넌트 제공
- Bulma
  - https://bulma.io/
  - Flexbox를 기반으로 하고 있고, 직관적 클래스 네이밍과 다양한 UI 컴포넌트 제공

CSS 프레임워크를 사용하는 이유

- 개발 속도 향상
  - 재사용 가능한 구성 요소
  - 반응형 그리드 시스템 지원
- 일관된 디자인
  - 통일된 스타일 유지
  - 테마와 커스터마이징 지원
- 반응형 웹 디자인
  - 다양한 화면 크기와 디바이스 대응
  - 미디어 쿼리 내장
- 크로스 브라우저 호환성
  - 다양한 브라우저 환경에서 검증됨
- 유지보수 용이성
  - 표준화된 코드 구조
  - 많은 문서 및 활성화된 커뮤니티
- 접근성 향상
  - 웹 접근성 표준 준수

CSS - Summary

- CSS란? 스타일과 레이아웃을 정의하는 스타일시트 언어
- 기본 문법 - 선택자 / 속성 / 값
- 선택자 – 요소 / 클래스 / 아이디 / 속성
- 상속 및 계단식 적용
- 박스 모델의 이해
- HTML에 CSS 적용하기 – 외부 / 내부 / 인라인
- CSS 주석
- CSS 프레임워크
- 참고
  - w3schools.com : https://www.w3schools.com/css/default.asp
  - 생활코딩: https://opentutorials.org/course/3086
  - 모질라: https://developer.mozilla.org/en-US/docs/Learn/CSS

JavaScript

웹 브라우저 자바스크립트에 대해서 알아본다

JavaScript의 역할과 중요성

JavaScript 란?

- 웹 브라우저에서 실행되는 프로그래밍 언어
- 웹 페이지에 동적이고 상호작용적인 기능을 추가

주요 역할

- 동적 콘텐츠 생성: 콘텐츠를 동적으로 생성 및 조작
- 사용자 상호작용 처리: 사용자 이벤트 처리
- 비동기 통신: 서비와 비동기적으로 데이터 통신
- HTML 및 CSS 조작: DOM을 통해 HTML 요소와 CSS 스타일 조작

중요성

- 인터렉티브 웹 페이지: 사용자와 상호 작용할 수 있게 지원
- 웹 애플리케이션 개발: JS 기반의 웹 앱 프레임워크, 개발 생산성 및 유지보수성 향상
  - React, Angular, Vue.js, Enact 등
- 광범위한 생태계: 다양한 라이브러리 지원, Node.js를 이용한 서버 측 개발 가능
- 웹 표준: 대부분의 주요 브라우저에서 지원

JavaScript 적용

외부 JS 파일 사용

- JavaScript가 외부에 정의되어 있고 HTML 페이지는 이를 참조
- 성능 향상 및 유지보수에 용이
- <script> 요소의 src 속성에 외부 Script 명시
  - 전체 URL
  - 파일 경로
  - 경로 없이 사용 (같은 level에 위치)

```
<script src="http://www.abc.com/js/myScript.js"></script>
<script src="../js/myScript.js"></script>
<script src="myScript.js"></script>
```

내부 JS 코드 사용

- JavaScript가 HTML 코드 내에 정의
- <head>섹션과 <body> 섹션 내에 <script> 와 </script> 태그 사이에 명시

```
<head>
<script>
window.onload = function() {
        console.log("window.onload");
}
</script>
</head>
```

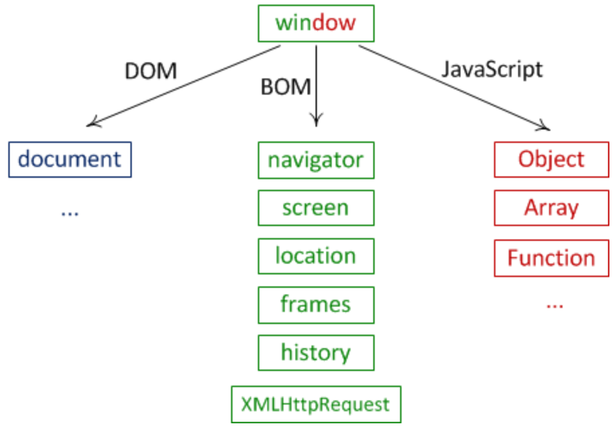
인라인 JS 코드 사용

- JavaScript가 HTML 요소 내에 정의
- <head>섹션과 <body> 섹션 내에 <script> 와 </script> 태그 사이에 명시

```
<input type="button" onclick="alert('Hello world')" value="Hello world" />
```

JavaScript – Object 모델

- 웹 브라우저의 구성요소들은 객체화 되어 있음
- JavaScript로 이 객체를 제어하여 웹 브라우저를 제어 할 수 있음
- 전역 객체인 window 객체를 중심으로 BOM(Browser Object Model)과 DOM(Document Object Model)을 포함한다



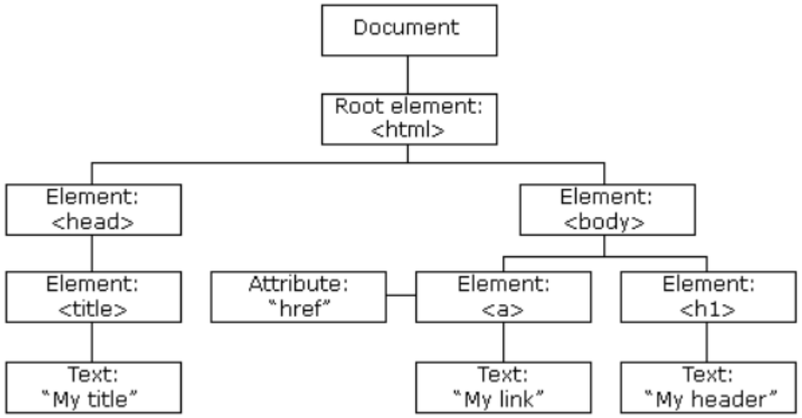
BOM과 DOM의 구성요소

- window: 브라우저 창 자체를 나타내며, 최상위 객체 (코드 내 생략 가능)
- navigator: 브라우저 정보를 제공하는 객체 (브라우저 이름, 버전, 운영체제 등)
- screen: 사용자의 화면 정보(해상도, 색 깊이 등)
- location: 현재 문서의 URL을 나타내는 객체
- frames: 현재 window 내에서 열려있는 모든 하위 프레임을 나타냄 (배열, 프레임 이름으로 접근)
- history: 브라우저의 세션 기록을 나타내는 객체, 사용자가 방문한 페이지들을 앞뒤로 이동
- document: DOM 트리의 루트 요소로 HTML 문서 전체를 나타내는 객체

JavaScript - DOM 조작

Document Object Model

- 웹페이지를 자바스크립트로 제어하기 위한 객체 모델
- Document 객체는 윈도우에 로드된 문서를 의미
- 트리 형태로 구성되고 이를 DOM Tree 라고 부름



DOM 조작

- 조작하고 싶은 HTML 요소를 찾기부터 시작
- HTML 요소의 변경
  - 콘텐츠 변경, 속성 변경, 스타일 변경
- HTML 요소의 추가 및 삭제
  - 새로운 요소 추가, 삭제, 변경
- 이벤트 핸들러 추가

JavaScript - DOM 조작

HTML 요소 찾기

- getElementById(): HTML 요소의 ID 값으로 찾기
- getElementsByTagName(): HTML 요소의 태그 이름으로 찾기 (NodeList로 반환)
- getElementsByClassName(): HTML 요소의 클래스 이름으로 찾기
- querySelectorAll(): CSS 선택자와 매치되는 HTML 요소 찾기

HTML 요소 변경

- innerHTML: HTML 요소의 콘텐츠를 변경
- attribute 이름: 새로운 값으로 변경. 예) src
- 동적 콘텐츠 생성
- HTML 아웃풋 스트림에 직접 쓰기

```
const element = document.getElementById("intro");
const element = document.getElementsByTagName("p");
```

```
<body>
<p id="demo"></p>
<script> document.write("Hello World"); </script>
<img id="myImage" src="smiley.gif">
<script>
function updateTime(){
        document.getElementById("demo").innerHTML =
"Date : " + Date();
}
let timerId = setInterval(updateTime, 1000);
document.getElementById("myImage").src = “myprofile.jpg";
</script>
</body>
```

JavaScript - DOM 조작

HTML 스타일 변경

- HTML 요소의 style 속성을 변경
- style.property 이름: property에 값을 할당

```
<html>
<body>
<p id="p2">Hello World!</p>
<script>
document.getElementById("p2").style.color = "blue";
</script>
</body>
</html>
```

JavaScript - DOM 조작

HTML 요소 생성 및 삭제

- 새로운 HTML 요소를 생성
- 부모 요소에 자식 요소를 추가 혹은 제거
- createElement(Tag name):
- appendChild(node 객체):
- removeChild(node 객체):
- replaceChild(para, child):

```
<!DOCTYPE html>
<html>
<body>

<h2>JavaScript HTML DOM</h2>
<p>Add a new HTML Element.</p>

<div id="div1">
<p id="p1">This is a paragraph.</p>
<p id="p2">This is another paragraph.</p>
</div>

<script>
const para = document.createElement("p");
const node = document.createTextNode("This is new.");
para.appendChild(node);
const element = document.getElementById("div1");
element.appendChild(para);
</script>

</body>
</html>
```

JavaScript - DOM 조작

```
<!DOCTYPE html>
<html>
<body>

<h2>JavaScript can Change HTML</h2>

<p id="p1">Hello World!</p>

<script>
document.getElementById("p1").innerHTML = "welcome to the webOS world!";
</script>

</body>
</html>
```

실행 결과는?

- _____

JavaScript – DOM 이벤트

- 웹 페이지의 요소와 사용자가 상호 작용 할 수 있게 해주는 기능

이벤트의 종류

- 마우스 이벤트
  - click: 요소를 클릭할 때 발생
  - dblclick: 요소를 더블클릭할 때 발생
  - mousedown: 마우스 버튼을 누를 때 발생
  - mouseup: 마우스 버튼을 뗄 때 발생
  - mouseover: 마우스가 요소 위로 이동할 때 발생
  - mouseout: 마우스가 요소 밖으로 벗어날 때 발생
- 키보드 이벤트:
  - keydown: 키보드 키를 누를 때 발생
  - keyup: 키보드 키를 뗄 때 발생
  - keypress: 키보드 키를 눌렀다 땔 때 발생
- 폼 이벤트:
  - submit: 폼을 제출할 때 발생
  - change: 입력 요소의 값이 변경될 때 발생
  - focus: 요소가 포커스를 받을 때 발생
  - blur: 요소가 포커스를 잃을 때 발생
- 문서/ 윈도우 이벤트:
  - load: 페이지나 이미지 등이 로드될 때 발생
  - resize: 브라우저 창 크기가 변경될 때 발생
  - scroll: 사용자가 페이지를 스크롤 할 때 발생
  - unload: 페이지를 떠날 때 발생

JavaScript – DOM 이벤트 리스너

- 특정 이벤트가 발생 할 때 실행할 함수를 정의하는 방법

addEventListner 메서드

- 가장 표준적이고 권장되는 방법
- 하나의 요소에 여러 개의 이벤트 리스너를 추가 할 수 있음

```
const button = document.getElementById('myButton');
button.addEventListner('click', function() {
  alert('Button clicked!');
});
```

이벤트 핸들러 프로퍼티

- HTML 요소의 이벤트 속성에 직접 함수를 할당
- 하나의 이벤트에 하나의 핸들러만 할당 할 수 있음

```
const button = document.getElementById('myButton');
button.onclick = function() {
  alert('Button clicked!');
};
```

HTML 속성

- HTML 요소 내에 직접 이벤트 핸들러를 설정

```
<button id="myButton" onclick = "alert('Button clicked!')">
Click me </button>
```

JavaScript – DOM 이벤트 객체

- 이벤트가 발생했을 때 브라우저가 이벤트 핸들러에게 전달하는 정보

이벤트의 정보

- type: 이벤트의 타입을 나타냄
- target: 이벤트가 발생한 요소
- clientX: 이벤트가 발생한 위치의 x 좌표
- clientY: 이벤트가 발생한 위치의 y 좌료
- key: 키보드의 눌러진 키

```
const button = document.getElementById('myButton');
button.addEventListner('click', function(event) {
  console.log(event);
  console.log(event.type);
});
```

JavaScript – DOM 이벤트 전파

- 이벤트가 발생했을 때 DOM 트리 내에서 이벤트의 전파 (캡쳐링, 버블링)

캡쳐링 (Capturing)

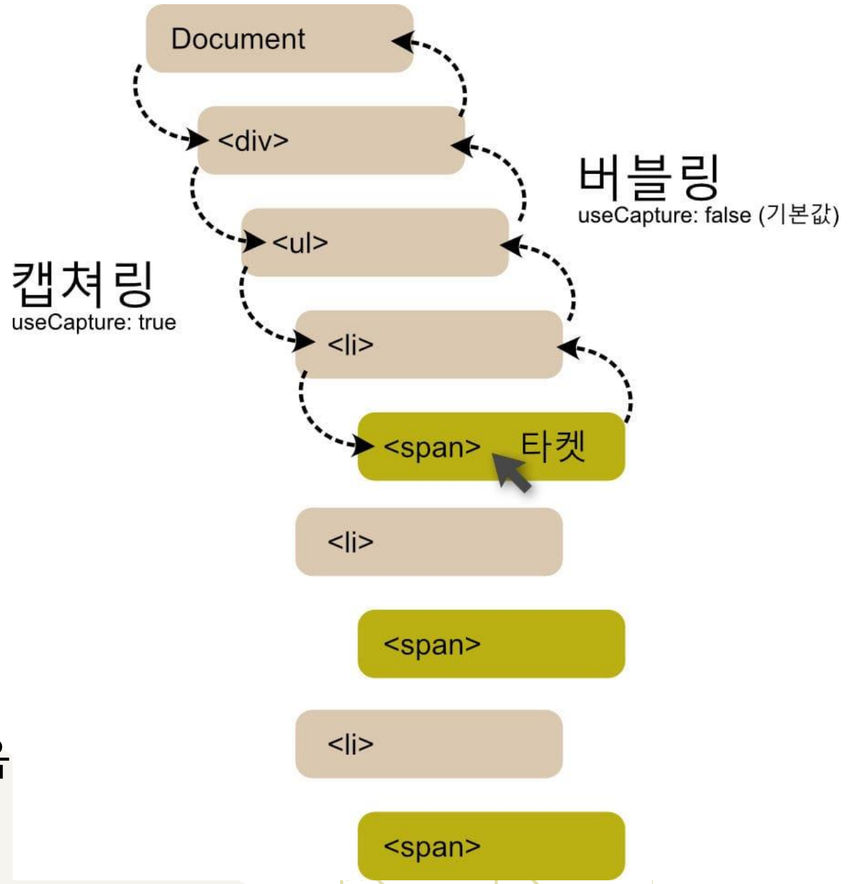
- 최상위 요소에서 목표 요소까지
- useCapture를 true로 설정

버블링 (Bubbling)

- 목표 요소에서 최상위 요소까지
- useCapture를 false로 설정 (기본값)

이벤트 전파 종료

- stopPropagation() 을 사용하여 이벤트 전파를 종료할 수 있음



https://www.w3schools.com/js/tryit.asp?filename=tryjs_addeventlistener_usecapture

출처: https://apost.dev/674/#google_vignette

JavaScript – DOM 이벤트 전파 (Code)

```
<body>
  <div>
    <ul>
      <li><span>아이템1</span></li>
      <li><span>아이템2</span></li>
      <li><span>아이템3</span></li>
    </ul>
  </div>
```

```
  <script>
    function eventPropagation(useCapture) {
      document.querySelector('body').addEventListener('click', function() {
        console.log("body");
      }, useCapture);
      document.querySelector('div').addEventListener('click', function() {
        console.log("div");
      }, useCapture);
      document.querySelector('ul').addEventListener('click', function() {
        console.log("ul");
      }, useCapture);
      document.querySelector('li').addEventListener('click', function() {
        console.log("li");
      }, useCapture);
    }
    //eventPropagation(true); // Capturing phase
    eventPropagation(false); // Bubbling phase
  </script>
</body>
```

JavaScript - BOM 조작

Browser Object Model

- 브라우저 창과 상호 작용하기 위한 객체 모델

window 객체

- window 객체는 BOM의 최 상위 객체
- 브라우저 창을 타나내며 모든 전역 객체와 함수는 window 객체의 프로퍼티로 접근 가능
- 생략할 수 있음

```
console.log (window.innerWidth);
console.log(window.innerHeight);
console.log (innerWidth);
console.log(innerHeight);
```

JavaScript - BOM 조작

navigator 객체

- 브라우저의 정보(브라우저 이름, 버전, 플랫폼 등)을 제공하는 객체
- 프로퍼티 보기 : https://developer.mozilla.org/en-US/docs/Web/API/Navigator
- 주요 프로퍼티
  - navigator.userAgent
  - navigator.platform
  - navigator.language
  - navigator.onLine

```
console.log (navigator.userAgent);
console.log(navigator.platform);
console.log (navigator.language);
console.log(navigator.onLine);
```

JavaScript - BOM 조작

location 객체

- 현재 문서의 URL을 나타내는 객체
- 프로퍼티 보기 : https://developer.mozilla.org/en-US/docs/Web/API/Location
- 주요 프로퍼티 & 메서드
  - location.href
  - location.protocol
  - location.host
  - location.port
  - location.pathname
  - location.search
  - location.assign(url)
  - location.reload()

```
console.log (location.href);
location.href = “https://www.webososе.org”;
location.assign(“https://www.webososе.org”);
```

JavaScript - BOM 조작

history 객체

- 사용자가 방문한 페이지의 세션 기록을 나타내는 객체
- 프로퍼티 보기 : https://developer.mozilla.org/en-US/docs/Web/API/History
- 주요 프로퍼티 & 메서드
  - history.back()
  - history.forward()
  - history.go(n)
  - history.length

```
console.log (history.length);
history.back();
history.forward();
history.go(-1);
```

JavaScript - BOM 조작

screen 객체

- 사용자의 화면 정보(해상도, 색 깊이 등)를 제공하는 객체
- 프로퍼티 보기 : https://developer.mozilla.org/en-US/docs/Web/API/Screen
- 주요 프로퍼티 & 메서드
  - screen.width
  - screen.height
  - screen.availWidth
  - screen.availHeight
  - screen.colorDepth

```
console.log (screen.width);
console.log (screen.height);
console.log (screen.colorDepth);
```

JavaScript - BOM 조작

대화상자

- **알림 (Alert)**
  - Alert 메서드를 사용하여 알림 상자를 표시

```
alert(“This is an alert box!”);
```

- **확인 (Confirm)**
  - confirm 메서드를 사용하여 확인 상자 표시
  - 사용자가 확인 또는 취소를 선택할 수 있음

```
let result = confirm("Are you sure?”);
console.log(result);
```

- **프롬프트 (Prompt)**
  - prompt 메서드를 사용하여 사용자 입력을 받을 수
  - 파라메터로 기본 입력 값을 설정할 수 있음

```
let userInput = prompt("Enter your name“, “Gildong Hong");
console.log(userInput);
```

JavaScript - BOM 조작

setTimeout 함수

- 일정 시간이 지난 후 한번 특정 코드를 실행
- setTimeout(function, delay, ...args);
  - function: 지연 시간 후 실행할 함수
  - delay: 밀리초 단위의 지연 시간 (예: 1000 -> 1초)
  - ..args: 선택적으로 전달할 인수
- 타이머 ID를 반환

setInterval 함수

- 일정 간격으로 특정 코드를 반복 실행
- setInterval(function, delay, ...args);
  - function: 반복 시간 간격마다 실행할 함수
  - delay: 밀리초 단위의 지연 시간 (예: 1000 -> 1초)
  - ..args: 선택적으로 전달할 인수
- 타이머 ID를 반환

```
function sayHello() {
  console.log(“hello, world!”);
}

let timerId = setTimeout(sayHello, 2000);
clearTimeout(timerId);

let intervalId = setInterval(sayHello, 1000);
clearInterval(intervalId);
```

JavaScript - Summary

- JavaScript란? 웹 브라우저에서 실행되는 프로그래밍 언어
- HTML에 JavaScript 적용 – 외부 / 내부 / 인라인
- JavaScript Object 모델
- DOM 과 BOM 조작
- DOM 조작 방법
  - HTML 요소 찾기 / HTML 요소 제어
- DOM 이벤트
  - 이벤트들 / 이벤트 리스너 / 이벤트 핸들링
  - 이벤트 전파 – 캡쳐링과 버블링
- BOM 조작
  - window, navigator, location, history, screen 객체
  - 대화상자, 타이머
- 참고
  - w3schools.com : https://www.w3schools.com/js/default.asp
  - 생활코딩: https://opentutorials.org/course/3085
  - 모질라: https://developer.mozilla.org/en-US/docs/Learn/JavaScript

프로젝트 실습

학습한 내용을 참고하여 간단한 프로젝트를 해보자

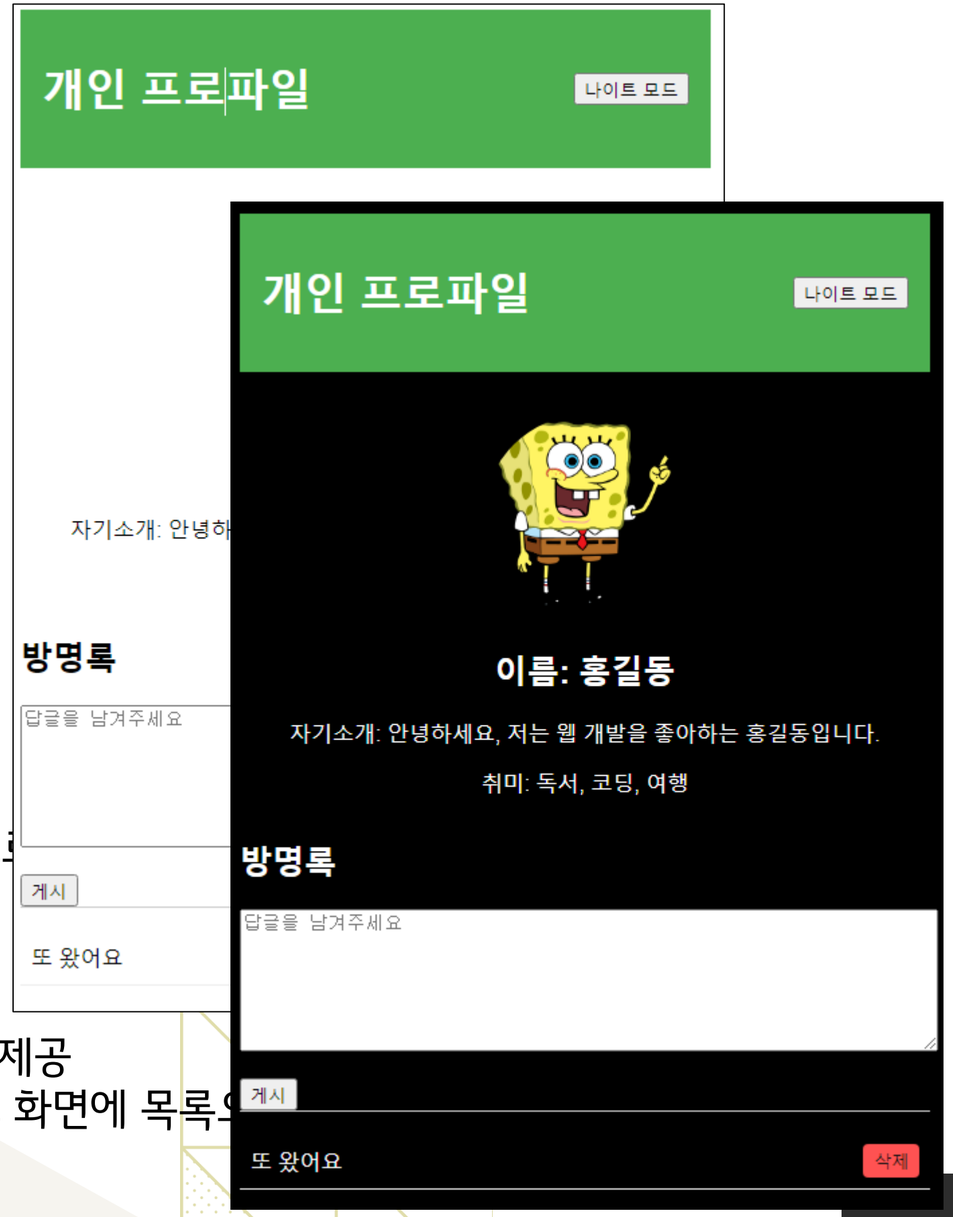
개인 프로필 페이지

목표

- HTML, CSS, JavaScript를 활용하여 개인 프로필 페이지를 만드는 것

주요 기능

- (Lv1) HTML 구성
  - 제목, 나이트모드, 프로필 사진, 이름, 자기소개, 취미, 방명록
- (Lv1) 나이트 모드
  - 나이트 모드 버튼을 클릭하면 배경은 검정색이고 글씨는 모두 흰색으
- (Lv2) 프로필 사진 배경색 변경
  - 마우스가 프로필 사진에 포커스가 되면 배경색을 랜덤하게 변경
- (Lv3) 방명록
  - 페이지 방문자들이 방명록을 남길 수 있도록 텍스트 박스, 게시 버튼 제공
  - 게시 버튼을 누르면 텍스트 박스의 내용을 로컬 스토리지에 저장하고 화면에 목록으



개인 프로필 페이지

TIPS

- 랜덤 배경 색 : '#000000'에서 '#ffffff' 사이의 무작위 색상 코드
  - `#${Math.floor(Math.random() * 16777215).toString(16)}`
- 부드러운 전환효과
  - transition: background-color 0.3s;
- 나이트 모드
  - 나이트모드의 CSS 변경사항 정리
  - JavaScript에서 나이트모드 버튼 클릭할때마다 스타일 토글
- 로컬 스토리지 사용
  - localStorage.getItem('guestbookEntries')
  - localStorage.setItem('guestbookEntries', JSON.stringify(guestbookEntries));

Questions

Email

nicky.oh@lge.com