웹 페이지 개발 기초

HTML5 기초

HTML: Hy[er Text Markup Language

HTML은 웹 페이지를 만들기 위해 사용하는 기본적인 마크업 언어

HyperText는 서로 다른 문서를 연결하는 기술

주로 링클 통해 다른 문서나 정보로 이동

Markup은 문서 구조화 하고 의미를 부여하기 위해 사용되는 태그나 기호

시멘틱 요소 추가

<!DOCTYPE html>

HTML5 문서임을 브라우저에게 알리는 지시문입니다. 이 선언을 통해 브라우저는 HTML5의 문서 규격을 따르게 되면, 최신 웹 표준을 준수한 렌더링을 수행

<html>

문서의 루트 요소입니다. 모든 HTML 요소는 이 태그 안에 위치

<head>

문서의 메타데이터를 포함하는 부분입니다. 메타 데이터는 문서의 제목, 문자 인코딩, 스타일 시트 링크, 스크립트 파일 드을 포함

<body>

실제 콘텐츠가 들어가는 부분. 이 태그 안에는 웹페이지에 표시될 모든 콘텐츠가 포함됩니다. 예를 들어, 텍스트, 이미지, 링크, 표 등 모든 요소가 이 안에 위치

시맨틱 구조

시맨틱 구존느 HTML 요소를 사용하여 문서의 구조를 명확하게 정의

시멘틱 요소는 단순히 스타일을 적용하기 위한 것이 아니라, 문서의 의미를 명확히 하고 검색 엔진이나 웹 크롤러가 문서를 이해하고 분석하는 데 도움을 줌

이를 통해 웹 접근성을 높이고 SEO(검색 엔진 최적화)를 개선할 수 있습니다.

**의미를 가지고 있지 않은 <div>의 사용을 지양하고, 최대한 의미가 있는 태그를 사용**

Header 섹션의 머리말을 정의 주로 로고 제목 검색 폼

Nav 네비게이션 링크를 정의 주로 사이트의 메뉴나 링크 목록

Main 문서의 주요 콘텐츠 정의 하나의 문서에는 하나의 <main> 요소만 사용

Section 문서의 섹션을 정의 주제별로 묶인 콘텐츠를 나타냄

Article 독립적인 콘텐츠 블록을 정의 뉴스 기사, 블로그 글, 포럼 게시물 등의 콘텐츠를 표현할 때 사용

Aside 사이드바 콘텐츠를 정의 주로 별도의 컨텐츠, 광고, 측면 정보 표시

Footer 문서나 섹션의 푸터를 정의 저작권 정보 , 연락처 등

자주 사용되는 태그

<div> 구획을 나누거나 그룹화할 때 사용하는 태그

<p> 문단을 정의하는 태그로, 일반적으로 텍스트를 포함

<a> 하이퍼링크를 생성하는 태그로, 다른 웹 페이지나 동일 페이지로 이동

<img> 이미지 삽입하는 태그로, 웹 페이지에 이미지를 표시

<ul> 순서 없는 목록을 정의하는 태그로, 항목들이 순서 없이 나열

<ol> 순서 있는 목록을 정의하는 태그로, 항목들이 순서대로 나열

<li> 목록의 항목을 정의하는 태그로, <ul>이나 <ol> 내부에서 사용

<table> 표를 정의하는 태그로, 데이터를 행과 열로 구조화 하여 표시

<form> 사용자 입력을 받는 양식을 정의하는 태그

<input> 사용자가 양식에 데이터를 입력할 수 있도록 하는 입력 필드를 생성하는 태그

<button> 클릭 가능한 버튼을 생성하는 태그로, 사용자의 상호작용을 유도할 때 사용됩니다.

Css3 기초

CSS Cascading Style Sheets 의 최신 버전으로, 웹페이지의 스타일과 레이아웃을 정의하는 언어

HTML이 웹 페이지의 구조를 담당하면, CSS는 이구조를 꾸며주는 역할을 합니다.

주로 색상, 폰트, 간격, 배치 등의 스타일을 지정하여 웹 페이지의 시각적인 표현을 다룸

주요 역할

스타일 지정

CSS는 HTML 요소에 스타일을 적용하여 웹 페이지의 시각적인 디자인 꾸밀 수 있음

Transition, animation 등의 속성을 사용하여 웹 요소에 다양한 애니메이션과 효과를 적용

레이아웃 조정

Css는 웹 페이지의 요소들을 배치하는 데 중요한 역할을 합니다

Flexbox난 Grid 같은 css 레이아웃 기술을 사용하여 요소들을 정렬하고 배치할 수 있습니다.

반응형 디자인

Css3는 미디어 쿼리와 함께 사용되어 다양한 디바이스 크기에 맞춰 웹 페이지를 반응형으로 만들 수 있습니다.

이를 통해 모바일 기기나 데스크톱 화면에 최적화 된 스타일 적용할 수 있습니다.

선택자 (Selector)

HTML 요소를 선택할 때 사용하는 패턴이나 규칙, 선택자는 CSS 규칙의 첫 부분에 위치

선택자 종류

전체 선택자 (\*) 모든 요소를 선택

요소 선택자: HTML 요소명을 직접 지정하여 스타일을 적용

Body 같은 태그들로 직접 사용

클래스 선택자 (.class) HTML 요소에 클래스를 추가하고 그 클래스를 선택하여 스타일 적용 클래스 이름으로 그룹 또는 여러 요소를 지정가능

아이디 선택자(#id): HTML 요소에 고유한 아이디를 추가하고 그 아이디를 선택하여 스타일 적용 #id

자식 선택자 (selector > selector) 특정 요소의 직계 자식 요소들을 선택하여 스타일을적용 .container > ul 직계 자식만 됨

자손 선택자 (selector selector) 특정 요소의 자손 요소들을 선택하여 스타일을 적용합니다.

.container li 직계 자식이 아닌 자손을 사용하고 싶을 때

박스 모델

HTML 요소는 content, padding, border, margin 으로 구성

Content 실제 내용이 표시되는 영역

Padding 콘텐츠와 테두리 사이의 여백

Border 콘텐츠 주위의 테두리입니다.

Margin 요소와 다른 요소 사이의 공간

플렉스 박스

요소를 효율적으로 정렬, 배치할 수 있는 레이아웃 모델

주로 한 방향(행 또는 열)으로 요소들을 정렬하는 데 사용

display: flex 를 사용하여 부모의 요소를 플렉스 컨테이너로 만들고, 자식 요소들을 유연하게 배치

그리드

2차원 레이아웃 시스템으로, 요소들을 행과 열의 격자 모양으로 배치할 수 있습니다.

Css display: grid를 사용하여 그리드 컨테이너를 만들고, 각 요소들을 그리드 셀에 배치

반응형 웹 디자인에 매우 유용하며, 복잡한 레이아웃을 쉽게 구현할 수 있습니다.

플로팅

Css에서 사용되는 레이아웃 기법으로, 특정 요소를 화면의 한쪽에 띄우는 방법

주로 이미지와 텍스트를 별렬로 배치할 때 사용되며, CSS에서 float:left 속성을 사용하여 요소를 띄움

알아두면 좋은 용어

반응형 디자인: 다양한 디바이스에서 웹 페이지가 잘 보이도록 하기 위해 미디어 쿼리 등을 사용

웹 접근성: 모든 사용자가 쉽게 접근하고 이용할 수 있도록 설계 적절한 태그 사용, 명확한 구조, 충분한 콘텐츠 대체 텍스트 등을 고려

크로스 브라우징: 다양한 웹 브라우저에서 동일하게 잘 작동하도록 스타일을 작성

자바스크립트 기초

자바스크립트는 웹페이지에 다양한 기능을 만들어주는 프로그래밍 언어로, HTML과 CSS와 함께 웹 프로그래밍의 핵심 요소 중 하나이다. 웹 페이지가 사용자와 상호작용하고 동작을 제어하는 데 사용

이벤트 처리: 사용자의 클릭, 마우스 오버, 입력 등의 이벤트를 감지하고 대응 하여 동작 수행

웹 페이지 조작: HTML 요소의 내용, 스타일, 구조를, 동적으로 변경하여 사용자 경험을 개선

데이터 처리: 서버와의 데이터 교환 , 데이터 유효성 검사 등을 통해 웹 애플리케이션 데이터 처리를 담당

자바스크립트의 기본 속성

동적 타입 언어: 변수의 타입을 선언하지 않고 사용할 수 있으며, 실행 시간에 타입이 결정됩니다.

인터프리터 언어: 자바스크립트는 인터프린터 언어로, 코드를 한 줄씩 해석하고 실행합니다. 이는 빠른 개발과 수정을 가능하게 하며, 빌드 과정 없이 바로 실행할 수 있습니다.

객체 기반: 자바스크립트는 객체 기반 언어로, 모든 것이 객체입니다. 함수도 객체이며, 함수에서 객체를 반환할 수 있습니다.

변수 선언

let name = "mac.bae";

const Pl = 3.14:

name ="mac.bae(주강사)": // 이 변수는 값을 변경할 수 있습니다.

PI= 3.141592; // 상수는 한 번 값을 할당하면 값을 변경할 수 없습니다.

함수 선언

function hello {

console.log("world!"):

}

const add = (a, b) => {//arrow function 함수 표현식

console.log(a + b);

}

기본 문법

객체는 프로그래밍에서 데이터와 기능을 하나의 단위로 묶어 표현하는 개념입니다. 객체는 속성(데이터)과 메서드(함수)를 포함할 수 있습니다.

객체 선언

const person = {

name: "mac.bae",

company: "goorm",

// 객체 안에 객체 혹은 함수 등 다양한 데이터를 넣을 수 있습니다.

hello: function 0) €

console.log("world");

},

}

console.log(person.name): // mac.bae 출력

console.log(person.helloO): // world 출력

DOM 조작 기초

DOM(Document Object Model)은 HTML 문서의 구조환된 표현을 제공하며, 이를 프로그래밍 언어가 이해하고 조작할 수 있게 해주는 인터페이스입니다. 즉, 웹 페이지의 요소들을 트리 구조로 표현하여 해당 요소를 수정하고 접근할 수 있습니다.

DOM을 통해 웹 페이지의 내용을 동적으로 수정하고 사용자의 상호작용을 구현할 수 있습니다. 이를 통해 웹 애플리케이션을 보다 동적이고 반응형을 만들수 있다.

DOM 요소 선택

Document: 자바스크립트에서 document 객체는 전체 HTML 문서를 나타내는 객체로서 여러가지 함수를 통해 DOM 과 상호작용할 수 있다.

let elementByld = document.getElementByld("myElementid");|

let elementsByClass = document.getElementsByClassName("myClassName");

let elementsByTag = document.getElementsByTagName("div");

let elementByQuery = document.querySelector(“p");

let elementsByQueryAll = document.querySelectorAll(".container >.item”);

DOM 요소 조작

let box = document.querySelector(“.box”)

let randomColor=”#”+Math.floor(Math.radnom()\*16777215).toString(16)

box.style.backgroundColor=randomColor:

이벤트 처리

addEventListner는 지정한 이벤트를 감지하고, 그 때 실행할 함수를 정의하는 역할

element.addEventListener(“Event Type”, callback)

EventType: click, dbclick등

Callback: 이벤트가 발생했을때 실행할 함수

비동기

프로그래밍에서 특정 작업이 다른 작업의 완료를 기다리지 않고 독립적으로 실행되는 방식을 말합니다.

이는 특히 네트워크 요청, 파일 읽기/ 쓰기 데이터베이스 쿼리 등의 i/o 작업에서 중요한 개념

비동기 동작의 특성

동시성: 비동기 작업은 여러 작업이 동시에 실행될 수 있습니다. 하나의 작업이 완료될 때까지 다른 작업이 차단도지 않고 진행될 수 있습니다.

비차단: 비동기적 작업은 요청이나 이벤트가 완료되기를 기다리지 않고 바로 다음 코드를 실행합니다. 이는 프로그램이 작업을 기다리는 동안 다른 작업을 수행할 수 있게 합니다.

콜백 함수: 비동기 작업에서는 보통 콜백 함수를 사용하여 작업이 완료되었을 때 실행할 동작을 정의 이는 코드의 순차적인 실행을 보장하면서도 비동기 작업을 처리할 수 있게 한다.

비동기 작업 예시

console.log(“시작”)  
setTimeout(function(){

console.log(“이 콜백은 3초 후에 실행”)

},3000)

console.log(“끝”)

출력  
시작

끝

이 콜백은 3초 후에 실행

콜백 함수의 문제점

콜백 함수는 비동기 작업이 완료되었을 때 실행할 함수를 전달하는 방법

하지만 콜백 함수 사용시 발생할 수 있는 문제점

콜백 지옥: 콜백 함수를 중첩하여 사용하다 보면 코드의 가독성이 떨어지고, 디버깅이 어려워질 수 있습니다.

에러 처리 어려움: 콜백 함수 내에서 발생하는 에러를 캐치하거나 처리하기가 어렵습니다.

Promise와 async.awiat

프로미스는 비동기 작업의 완료 또는 실패를 나타내는 객체

Promise의 세 가지 상태

대기(pending): 비동기 작업이 아직 완료되지 않은 상태

이행(fulfilled): 비동기 작업이 성공적으로 완료된 상태

거부(rejected): 비동기 작업이 실패한 상태

const myPromise = new Promise(resolve, reject) => €

// resolve는 Promise의 상태를 fulfiled로 변경하고, reject는 Promise의 상태를 rejected로 변경합니다.

// 비동기 작업 수행

setTimeout(C) =>{

let success = true; // 작업 성공 여부 설정

if (success){

resolve("작업이 성공했습니다!"); // 작업 성공 시, Promise 상태를 fulfiled 로 변경

else {

reject("작업이 실패했습니다."): // 작업 실패 시, Promise 상태를 rejected 로 변경

}, 10000):

});

myPromise;  
  
myPromise

.then((result)=> {

console.log(result): // 동작 성공 시 실행 (= Promise.resolve 가 실행되었을 때)

.catch((error) => {

console.error(error); // 작업 실패시 실행 (= Promise.reject 가 실행되었을 때))

async/await로 쉽게 비동기 다루기

async function fetchData(){

try{

let response=await fetch("https")

let data=await response.json()

console.log(data)

}

catch(error){

console.error("error")

}

}