1. Html의 form과 action
   1. Form : 웹페이지에서 사용자로부터 데이터를 입력 받을 때 사용되는 요소  
      input, button, checkbox, radio button, text area 등이 존재하며 사용자가 form 안에 데이터를 입력하고 submit 버튼을 클릭하면 입력한 데이터가 서버로 전송되어 처리됨
   2. Action : form 데이터가 제출될 서버측 스크립트 파일의 URL을 지정. 서버 script는 이 URL로 form data를 받아서 처리
   3. Method : form data를 서버로 전송하는 방법
      1. GET : 데이터가 URL에 노출. 전송 가능한 데이터 크기에 제한이 있으며 보안에 취약. 주로 데이터 조회에 사용
      2. POST : 데이터가 URL에 노출되지 않아 데이터 크기에 제한이 없으며 보안에서는 GET 방식보다는 안전. 주로 데이터 제출에 사용
   4. URL과 URI의 차이 : URI은 리소스를 식별하는 일반적 개념, URL은 리소스 위치를 나타내는 구체적인 형태의 URI
   5. Parameter : 웹 요청에 포함된 데이터를 가리키는 매개변수. URL에 포함된 데이터 또는 form data의 이름-값(key-value)쌍으로 전달. 서버에서 매개변수를 읽어서 해당 요청에 대한 응답을 생성하거나 처리
2. Javascript
   1. DOM Manipulation : 문서 객체 모델 조작. 문서의 구조를 표현하는 트리 구조. Js를 이용해서 DOM을 조작하면 웹 페이지의 내용, 구조, 스타일을 동적으로 변경 가능. DOM조작을 통해 요소의 속성이나 내용을 읽거나 변경 가능
   2. EventListener : 특정 이벤트(onclick, hover…)가 발생했을 때 실행할 JS코드를 지정하는 방법.
      1. Onclick과 addEventListener의 차이 : ‘onclick’은 html에 직접 이벤트 핸들러를 지정하는 방식이지만 ‘addEventListener’는 Js 코드 내에서 이벤트를 동적으로 처리하므로 여러 이벤트 헨들러를 동일한 이벤트에 추가할 수 있어 더 많은 유연성을 제공하므로 ‘addEventListener’ method를 이용하여 이벤트 처리하는 방법 권장.
   3. Event Loop : JS의 비동기 처리를 담당하는 메커니즘. 실행 스택(call stack), 메시지 큐(message queue), 백그라운드(back ground)로 구성되며 비동기적인 작업들이 이 구조를 통해 처리됨. Ex) 콜백함수, 프로미스, async/await 은 이벤트 루프를 통해 적절한 시점에서 실행됨
   4. getElementById : 문서 내에서 주어진 ID를 가진 첫 번째 요소를 반환. ID는 문서 내에서 고유해야함
   5. getElementsByClassName : 지정된 클래스 이름을 가진 모든 요소를 반환. 반환된 값은 HTMLCollection 형태로 반환
   6. querySelector VS querySelectorAll
      1. querySelector : 주어진 css 선택자와 일치하는 첫 번째 요소를 반환
      2. querySelectorAll : 주어진 css 선택자와 일치하는 모든 요소를 NodeList 형태로 반환
   7. Node의 속성 : DOM의 기본요소. Node 인터페이스는 html 요소 뿐만 아니라 텍스트 노드, 주석, 문서 노드 등 모든 종류의 노드를 나타냄. 아래의 속성들을 이용하여 DOM 트리에서 노드 간의 관계를 탐색
      1. parentNode : 현재 노드의 부모 노드를 반환. DOM 트리에서 상위에 위치한 노드
      2. childNodes : 현재 노드의 모든 자식 노드를 반환. NodeList 형태로 반환되며 유사 배열(1부터 시작하는 숫자로 인덱싱된 요소)
      3. firstChild : 현재 노드의 첫번째 자식 노드를 반환. 자식 노드가 없으면 ‘null’을 반환
      4. lastChild : 현재 노드의 마지막 자식 노드를 반환. 자식 노드가 없으면 ‘null’을 반환
      5. nextSibling : 현재 노드의 다음 형제 노드를 반환. 다음 형제 노드가 없으면 null’을 반환
      6. previousSibling : 현재 노드의 이전 형제 노드를 반환. 이전 형제 노드가 없으면 null’을 반환
      7. nodeName : 현재 노드의 이름을 문자열로 반환. HTML 요소 노드의 경우 대문자로 노드 이름을 반환
      8. nodeType : 현재 노드의 유형을 나타내는 숫자를 반환
         * ‘1’ : 엘리먼트 노드(html 요소)
         * ‘3’ : 텍스트 노드(엘리먼트의 텍스트 내용)
         * ‘8’ : 주석 노드
         * ‘9’ : 문서 노드(전체 문서)
         * ‘11’ : 문서 조각 노드
3. Javascript 문법 [예시 코드 포함]
   1. Var과 let의 차이
      1. Var : 변수를 선언하는 키워드. 함수 스코프를 갖고 있어 함수 내에서 선언된 변수는 함수 내에서만 접근 가능
      2. Let : 블록 스코프를 갖고 있어 블록 내에서 선언된 변수는 해당 블록 내에서만 접근 가능

*function exampleFunction() {*

*if (true) {*

*var varVariable = "I am var"; // 함수 스코프*

*let letVariable = "I am let"; // 블록 스코프*

*}*

*console.log(varVariable); // 출력: "I am var"*

*console.log(letVariable); // 에러: letVariable is not defined*

*}*

* 1. Const : 상수를 선언하는 키워드. 한번 지정한 값은 변경 불가

*const pi = 3.14159;*

*pi = 3; // 에러: Assignment to constant variable*

* 1. ES5의 Array 관련 메소드 (최소 4개 이상)
     1. forEach() : 배열의 각 요소에 대해 콜백 함수를 실행
     2. map() : 배열의 각 요소에 대해 콜백 함수를 실행하고 그 결과를 새로운 배열로 반환
     3. filter() : 주어진 함수의 조건을 만족하는 요소로 이루어진 배열을 반환
     4. reduce() : 배열의 각 요소에 대해 콜백 함수를 실행하고 누적된 결과 값을 반환

*const numbers = [1, 2, 3, 4, 5];*

*numbers.forEach(function(num) {*

*console.log(num);*

*});*

*const squaredNumbers = numbers.map(function(num) {*

*return num \* num;*

*});*

*const evenNumbers = numbers.filter(function(num) {*

*return num % 2 === 0;*

*});*

*const sum = numbers.reduce(function(acc, num) {*

*return acc + num;*

*}, 0);*

* 1. Rest parameter : 함수 정의에서 여러 개의 인수를 배열로 받을 수 있게 함

*function sum(...numbers) {*

*return numbers.reduce((acc, num) => acc + num, 0);*

*}*

*console.log(sum(1, 2, 3, 4, 5)); // 출력: 15*

* 1. Arrow function : 함수를 간결하게 정의할 수 있음

*const add = (a, b) => a + b;*

*console.log(add(2, 3)); // 출력: 5*

*const square = num => num \* num;*

*console.log(square(4)); // 출력: 16*

* 1. 1급 객체 : js에서 함수. 함수를 변수에 할당하고 다른 함수에 인자로 전달, 함수를 다른 함수의 반환 값으로 사용 가능

*const greet = function(name) {*

*return `Hello, ${name}!`;*

*};*

*const result = greet("Alice");*

*console.log(result); // 출력: "Hello, Alice!"*

* 1. Object의 개념 : js에서 객체는 속성(프로퍼티)들의 집합. 프로퍼티는 key-value로 구성. 객체는 중활호 ’{}’ 를 사용하여 생성, 프로퍼티에 접근할 때에는 도트(‘.’) 또는 대괄호(‘[]’)를 사용

*const person = {*

*name: "Alice",*

*age: 30,*

*greet: function() {*

*console.log(`Hello, my name is ${this.name} and I am ${this.age} years old.`);*

*}*

*};*

*console.log(person.name); // 출력: "Alice"*

*console.log(person.age); // 출력: 30*

*person.greet(); // 출력: "Hello, my name is Alice and I am 30 years old."*

*🡺 ‘person’객체는 ‘name’, ‘age’라는 프로퍼티를 갖고 있고 ‘greet’라는 메서드를 포함하고 있음*