JavaScript:

**자바스크립트는 무슨 언어인가요?**  
  
자바스크립트는 웹 브라우저에서 실행되는 스크립트 언어로, 웹 페이지를 동적으로 제어하고 상호 작용할 수 있게 해주는 언어입니다.

**변수 선언, 초기화, 할당의 차이점에 대해 설명해주세요.**  
변수 선언: 변수를 정의하고 이름을 부여하는 단계입니다. var, let, const 등의 키워드를 사용하여 선언합니다.

변수 초기화: 변수에 값을 처음으로 할당하는 단계입니다. 초기화하지 않으면 변수는 undefined가 됩니다.

변수 할당: 변수에 값을 다시 대입하는 것을 말합니다.

**데이터 타입에 대해 설명해주세요.**  
자바스크립트의 주요 데이터 타입은 다음과 같습니다:  
원시 타입(Primitive Types): 숫자(Number), 문자열(String), 불리언(Boolean), null, undefined, Symbol  
참조 타입(Reference Types): 객체(Object), 배열(Array), 함수(Function), 날짜(Date), 등

**생성자에 대해 설명해주세요.**  
생성자는 객체를 생성하기 위한 함수를 가리킵니다. new 키워드와 함께 생성자 함수를 호출하면 새로운 객체가 생성되고, 이 객체는 생성자 함수의 프로토타입을 상속받습니다.

**this에 대해 설명해주세요.**  
this는 현재 실행 중인 함수나 메서드가 속한 객체를 가리키는 특수한 키워드입니다. this를 사용하여 객체 내에서 자신을 참조하거나 객체의 속성과 메서드에 접근할 수 있습니다.

**call, apply, bind에 대해 설명해주세요.**  
call과 apply는 함수를 호출할 때 this의 값을 지정하고, 인자를 전달할 때 사용됩니다.

call: 함수를 호출하면서 this 값을 지정하고, 인자를 콤마로 구분하여 전달합니다.

apply: 함수를 호출하면서 this 값을 지정하고, 인자를 배열로 전달합니다.

bind: 함수의 this 값을 영구적으로 바인딩하고, 새로운 함수를 반환합니다.

**콜백 함수에 대해 설명해주세요.**  
콜백 함수는 다른 함수의 인자로 전달되는 함수를 말합니다. 이 함수는 나중에 실행될 때 비동기 작업을 처리하거나 코드의 재사용성을 높이는 데 사용됩니다.

**콜백 지옥을 해결하는 방법을 설명해주세요.**  
콜백 지옥은 콜백 함수가 중첩되어 가독성이 떨어지는 상황을 가리킵니다. 이를 해결하기 위한 방법으로는 다음과 같은 것들이 있습니다:

콜백 헬 피하기: 너무 많은 콜백 함수 중첩을 피하기 위해 비동기 코드를 Promise나 async/await로 변환합니다.

모듈화: 코드를 모듈로 분리하여 작은 함수와 콜백을 사용하여 가독성을 향상시킵니다.

프로미스 체이닝: Promise를 이용하여 비동기 코드를 체이닝하여 구조화하고 관리하기 쉽게 만듭니다.  
라이브러리 사용: 콜백 지옥을 해결하는 데 도움이 되는 라이브러리나 프레임워크를 사용합니다.

**Promise에 대해 설명해주세요.**  
  
Promise는 비동기 작업을 처리하기 위한 객체로, 작업이 완료되었을 때 성공 또는 실패 여부를 처리할 수 있도록 합니다. Promise는 세 가지 상태를 가집니다: 대기(Pending), 이행(Fulfilled), 거부(Rejected).

**Promise.all() 에 대해 설명해주세요.**  
  
promise.all()은 여러 개의 Promise를 동시에 처리하고, 모든 Promise가 이행될 때까지 기다린 후 하나의 Promise로 결과를 반환합니다. 이 때, 모든 Promise가 성공해야 전체 Promise가 성공으로 처리됩니다.

**Promise와 Callback를 비교 설명해주세요.**  
  
Promise와 Callback은 비동기 작업 처리에 사용되는 두 가지 접근 방식입니다.

Callback:  
콜백 함수를 이용하여 비동기 작업의 결과를 처리합니다.  
콜백 지옥 등 코드 가독성과 관리가 어려울 수 있습니다.

Promise:  
Promise 객체를 이용하여 비동기 작업을 보다 구조화하고 가독성 있게 처리할 수 있습니다.  
then() 및 catch()를 이용하여 성공과 실패 시 처리 로직을 분리하여 작성할 수 있습니다.

**Async, Await이 뭔지 그리고 사용 방법을 설명해주세요.**  
async와 await는 Promise를 더 쉽게 다루기 위한 자바스크립트의 키워드입니다.  
async 함수 내에서 await 키워드를 사용하여 Promise가 이행될 때까지 기다리고, 결과를 반환합니다.  
async 함수는 항상 Promise를 반환하며, await 키워드는 Promise가 아닌 값도 받을 수 있습니다.  
try...catch 블록을 사용하여 예외 처리가 가능하며, 비동기 코드를 동기적으로 작성할 수 있습니다.

**Promise와 Async, Await의 차이를 설명해주세요.**  
Promise와 Async, Await은 모두 비동기 코드를 처리하는 방법이지만, 다음 차이점이 있습니다:  
가독성: Async, Await이 더 가독성이 높으며, 코드의 구조화가 쉽습니다.  
예외 처리: Async, Await에서는 try...catch 블록을 사용하여 예외 처리가 용이합니다.  
Promise 체이닝 vs. 선형 구조: Promise는 .then()을 사용하여 체이닝 구조로 작성되므로 중첩을 피하기 위해 노력해야 합니다. 반면, Async, Await은 선형 구조로 코드를 작성할 수 있습니다.

**AJAX에 대해 설명해주세요.**  
  
AJAX(Asynchronous JavaScript and XML)는 비동기적으로 서버와 통신하는 기술로, 웹 페이지를 새로고침하지 않고 데이터를 동적으로 로드하거나 업데이트할 수 있게 합니다. AJAX는 XMLHttpRequest 객체나 Fetch API를 사용하여 구현됩니다.

**var, let, const 차이를 설명해주세요.**  
var: 함수 스코프를 가지며, 변수가 선언되기 전에 접근 가능합니다. 중복 선언이 허용되고 호이스팅됩니다.

let: 블록 스코프를 가지며, 변수가 선언되기 전에는 접근할 수 없습니다. 중복 선언은 허용되지 않습니다.

const: 블록 스코프를 가지며, 상수를 선언할 때 사용하며 변수에 할당된 값을 변경할 수 없습니다.

**TDZ에 대해 설명해주세요.**  
TDZ(Temporal Dead Zone)는 let 및 const 변수가 선언된 위치에서 초기화되기 전까지의 영역을 가리킵니다. TDZ 내에서 변수에 접근하면 ReferenceError가 발생합니다.

**함수 선언형과 함수 표현식의 차이에 대해 설명해주세요.**  
  
함수 선언형(Function Declaration): 함수를 선언하고 이름을 붙입니다. 전역 스코프나 함수 스코프에서 어디서든 접근할 수 있습니다.

함수 표현식(Function Expression): 함수를 변수에 할당하고, 이름이나 익명 함수로 생성할 수 있습니다. 변수 스코프 내에서만 접근할 수 있습니다.

**이벤트 버블링과 캡처링에 대해 설명해주세요.**  
  
이벤트 버블링(Event Bubbling): 이벤트가 발생한 요소에서 시작하여 상위 요소로 이벤트가 전파되는 현상을 가리킵니다. 즉, 가장 내부 요소부터 시작하여 가장 바깥 요소까지 이벤트가 전파됩니다.

이벤트 캡처링(Event Capturing): 이벤트 버블링과 반대로 이벤트가 발생한 요소에서 시작하여 하위 요소로 이벤트가 전파되는 현상을 가리킵니다. 캡처링 단계에서 이벤트 핸들러가 호출되며, 그 후 버블링 단계에서 호출됩니다.

**이벤트 위임에 대해서 설명해주세요.**  
이벤트 위임은 하위 요소에 대한 이벤트 핸들러를 상위 요소에 하나만 등록하여 관리하는 패턴입니다. 이를 통해 많은 하위 요소에 대한 이벤트 핸들러를 개별적으로 등록하지 않고도 이벤트를 처리할 수 있습니다. 주로 리스트나 테이블 등의 요소에서 사용됩니다.

**이벤트 위임의 동작 방식에 대해서 설명해주세요.**이벤트 위임은 다음과 같은 방식으로 동작합니다:  
상위 요소(부모 요소)에 이벤트 핸들러를 등록합니다.  
이벤트가 발생한 하위 요소에서 상위 요소로 이벤트가 버블링됩니다.  
상위 요소에서 이벤트 핸들러가 동작하며, event.target을 통해 어떤 하위 요소에서 이벤트가 발생했는지 확인할 수 있습니다.  
상위 요소에서 event.target을 기반으로 특정 동작을 수행합니다.

**호이스팅과 발생하는 이유에 대해 설명해주세요**.  
  
호이스팅(Hoisting)은 자바스크립트에서 변수 및 함수 선언이 컴파일 단계에서 메모리에 끌어올려지는 현상을 가리킵니다. 호이스팅은 다음과 같은 이유로 발생합니다:  
호이스팅 : 변수 및 함수 선언을 스코프의 맨 위로 끌어올려서 사용 가능하게 합니다.  
변수 선언 : 변수는 선언 시 undefined로 초기화됩니다.  
함수호이스팅: 함수는 전체가 끌어올려지므로 함수 선언 이전에 호출할 수 있습니다.

**스코프 (Scope)에 대해 설명해주세요.**  
스코프는 변수와 함수의 유효 범위를 나타내는 개념으로, 변수와 함수가 어디에서 접근 가능한지를 결정합니다. 자바스크립트에서는 전역 스코프(Global Scope)와 지역 스코프(Local Scope)가 있으며, 스코프 체인(Scope Chain)을 통해 변수를 검색합니다.

**스코프 체인에 대해 설명해주세요.**  
스코프 체인은 변수 검색 시 변수가 속한 스코프부터 시작하여 상위 스코프로 올라가며 변수를 찾는 과정을 나타냅니다. 스코프 체인은 변수의 유효 범위를 결정하고, 변수가 어떤 스코프에 있는지를 판단합니다.

**클로져(Closure)에 대해 설명해주세요.**  
클로져는 함수와 그 함수가 선언된 스코프의 조합입니다. 클로져는 함수가 자신이 선언된 스코프 외부에서 호출되더라도 해당 스코프의 변수에 접근할 수 있는 기능을 가집니다. 클로져는 주로 private 변수를 구현하거나 함수 팩토리(factory)로 사용됩니다.

**실행 컨텍스트에 대해 설명해주세요.**  
실행 컨텍스트(Execution Context)는 코드가 실행되는 환경을 나타내며, 변수, 함수 선언, this 등의 정보를 담고 있습니다. 실행 컨텍스트는 스택(Stack) 구조로 관리되며, 스택의 맨 위에 있는 실행 컨텍스트가 현재 실행 중인 컨텍스트입니다.

**렉시컬 환경(Lexical Environment)에 대해 설명해주세요.**  
렉시컬 환경은 변수와 스코프를 관리하는데 사용되는 자바스크립트 엔진의 내부 구조입니다. 렉시컬 환경은 현재 스코프의 변수, 상위 스코프의 참조, this 등의 정보를 포함합니다.

**자바스크립트에서 일어나는 데이터 형변환에 대해 설명해주세요.**자바스크립트에서 데이터 형변환은 자동 형변환(암시적 형변환)과 명시적 형변환(강제 형변환)으로 나뉩니다.  
암시적 형변환: 연산자 또는 함수 호출 시 자동으로 형변환이 일어나는 경우입니다. 예를 들어, 문자열과 숫자를 더하면 숫자가 문자열로 자동 변환됩니다.  
강제 형변환: 개발자가 명시적으로 데이터 타입을 변환하는 경우입니다. 예를 들어, parseInt() 함수를 사용하여 문자열을 정수로 변환할 수 있습니다.

**자바스크립트가 동적 언어인 이유는 무엇인가요?**  
자바스크립트는 변수의 데이터 타입을 런타임 시점에서 동적으로 결정합니다. 즉, 변수의 타입을 선언 시점이 아니라 실행 중에 결정하는 언어이기 때문에 동적 언어입니다.

**프로토타입에 대해 설명해주세요.**  
프로토타입(Prototype)은 객체지향 프로그래밍에서 객체 간 상속을 구현하는 메커니즘입니다. 모든 객체는 프로토타입 객체를 가지며, 프로토타입 객체의 프로퍼티와 메서드를 상속받아 사용할 수 있습니다.

**깊은 복사와 얕은 복사에 대해 설명해주세요.**  
깊은 복사(Deep Copy): 객체 또는 배열의 모든 내용을 복사하여 완전히 새로운 객체 또는 배열을 생성하는 것을 의미합니다. 새로운 객체와 기존 객체는 완전히 독립적입니다.

얕은 복사(Shallow Copy): 객체 또는 배열을 복사할 때, 내부 객체의 참조만 복사하여 부분적으로 새로운 객체를 생성하는 것을 의미합니다. 내부 객체는 여전히 동일한 메모리 주소를 참조할 수 있습니다.

불변성을 유지하려면 어떻게 해야하나요?  
  
불변성(Immutability)을 유지하려면 다음과 같은 방법을 사용할 수 있습니다:  
const를 사용하여 변수 재할당을 금지합니다.  
객체 또는 배열을 복사하여 새로운 객체 또는 배열을 생성하고 수정합니다.  
객체나 배열을 변경하는 대신 새로운 객체나 배열을 반환하는 메서드를 사용합니다(예: concat(), slice(), map(), filter()).  
불변성을 유지하는 라이브러리(예: Immutable.js)를 사용합니다.

**Blocking과 Non-Blocking에 대해 설명해주세요.**  
Blocking: 어떤 작업이 실행 중일 때 다른 작업이 대기해야 하는 상황을 가리킵니다. 예를 들어, 동기적인 함수 호출은 호출된 함수가 실행을 완료할 때까지 블록됩니다.

NonBlocking: 어떤 작업이 실행 중일 때 다른 작업이 대기하지 않고 계속 실행될 수 있는 상황을 가리킵니다. 예를 들어, 비동기적인 함수 호출은 호출된 함수가 실행 중이어도 다른 작업을 계속할 수 있습니다.

**동기와 비동기에 대해 설명해주세요.**  
동기(Synchronous): 작업이 순차적으로 실행되며, 한 작업이 끝날 때까지 다음 작업을 시작하지 않는 방식을 가리킵니다. 동기 작업은 블로킹(blocking)될 수 있습니다.

비동기(Asynchronous): 작업이 병렬 또는 별도의 스레드에서 실행되며, 작업이 완료되기를 기다리지 않고 다른 작업을 계속할 수 있는 방식을 가리킵니다. 비동기 작업은 논블로킹(non-blocking)됩니다.

**ES6에서 새로 생긴 기능을 아는대로 말씀해주세요.**  
ES6(ECMAScript 2015)에서 도입된 주요 기능 및 변경 사항은 다음과 같습니다:

let 및 const 변수 선언  
화살표 함수(Arrow Functions)  
클래스(Class) 정의  
모듈 시스템(ES6 Modules)  
템플릿 리터럴(Template Literals)  
비구조화 할당(Destructuring Assignment)  
확장된 객체 리터럴 문법  
Promise와 비동기/동기 처리 개선  
생성자 함수 및 프로토타입 개선  
Map과 Set 등 새로운 데이터 구조  
확장된 배열 메서드(예: map(), filter(), reduce())  
Rest 및 Spread 연산자  
제너레이터(Generator)  
Proxy와 Reflect 객체  
심볼(Symbol) 데이터 타입

**undefined, null, undeclared를 비교해주세요.**undefined: 값이 할당되지 않은 변수 또는 객체의 속성, 함수의 리턴값이 없을 때 기본적으로 갖는 값입니다.

null: 개발자가 의도적으로 값이 없음을 나타내는데 사용하는 값입니다.

undeclared: 선언되지 않은 변수로, 스코프 내에서 찾을 수 없는 변수를 가리킵니다. 참조하면 ReferenceError가 발생합니다.

**자바스크립트를 멀티 쓰레드처럼 사용하는 방법이 뭔가요? / 자바스크립트에서 비동기 로직이 어떻게 동작하는지 설명해주세요. (이벤트 루프)**  
자바스크립트는 단일 스레드 언어로, 한 번에 하나의 작업만 처리할 수 있습니다. 그러나 비동기 프로그래밍을 통해 멀티 쓰레드처럼 동작하도록 할 수 있습니다.

**태스크 큐와 마이크로 태스크 큐에는 어떤 함수가 들어가나요?**  
태스크 큐(Task Queue)와 마이크로 태스크 큐(Microtask Queue)는 비동기 작업의 스케줄링을 관리하는 자바스크립트 엔진의 일부입니다.  
태스크 큐에는 주로 비동기 작업 중에서도 콜백 함수나 이벤트 핸들러 같은 브라우저 이벤트와 관련된 작업이 들어갑니다.  
마이크로 태스크 큐에는 Promise와 관련된 작업이 들어갑니다. 즉, then() 또는 catch()와 같은 Promise의 처리 작업이 마이크로 태스크 큐에 들어가게 됩니다.

**requestAnimationFrame에 대해 설명해주세요.**  
requestAnimationFrame은 브라우저에서 제공하는 API로, 웹 애니메이션을 구현하기 위한 타이머 기능을 제공합니다. 이 메서드를 사용하면 브라우저의 리플로우(reflow) 및 리페인트(repaint) 사이클과 동기화하여 부드러운 애니메이션을 생성할 수 있습니다.

**비동기적으로 실행되는 것을 동기적으로 코딩하는 방법이 있나요?**비동기 코드를 동기적으로 처리하기 위해 async/await을 사용할 수 있습니다. await 키워드는 Promise가 이행될 때까지 기다린 후 다음 코드를 실행하므로, 비동기 코드를 동기적으로 작성할 수 있습니다.

**map과 forEach, reduce에 대해 설명해주세요.**  
  
map(): 배열의 각 요소에 대한 변환 함수를 적용하고, 새로운 배열을 반환합니다. 원본 배열은 변경되지 않습니다.

forEach(): 배열의 각 요소에 대한 작업을 수행합니다. 반환값이 없으며, 원본 배열을 변경하지 않습니다.

reduce(): 배열의 요소를 왼쪽에서 오른쪽으로 누적하면서 하나의 값으로 축소하는 함수를 적용합니다. 초기값과 함께 사용하며, 최종 결과값을 반환합니다.

**자바스크립트의 메모리 관리에 대해 아는 대로 설명해주세요.**  
자바스크립트는 가비지 컬렉션(Garbage Collection)을 통해 메모리를 관리합니다. 주요 개념은 다음과 같습니다:  
Reachability: 변수나 참조로 접근 가능한 객체는 메모리에서 제거되지 않습니다.  
메모리 누수(Memory Leak): 더 이상 필요하지 않는데 메모리에서 제거되지 않은 객체를 가리킵니다.  
V8 엔진과 같은 자바스크립트 엔진은 여러 가지 최적화 기법을 사용하여 가비지 컬렉션의 효율을 높이며, 메모리 누수를 방지합니다.

**클래스에 대해 설명해주세요.**  
  
클래스(Class)는 객체지향 프로그래밍(OOP)의 개념으로, 객체를 생성하기 위한 설계도 또는 템플릿입니다. 클래스는 속성(멤버 변수)와 메서드(멤버 함수)로 구성되며, 객체를 생성하기 위해 사용됩니다. ES6에서 도입된 클래스 문법을 사용하여 클래스를 정의할 수 있습니다.

**즉시 실행 함수 (IIFE)에 대해 설명해주세요.**  
  
IIFE(Immediately Invoked Function Expression)는 함수를 정의하고 즉시 실행하는 패턴입니다. IIFE는 주로 함수 스코프를 생성하여 변수의 유효 범위를 제한하거나 전역 스코프에서 변수 충돌을 방지하는 데 사용됩니다.

**엄격 모드에 대해 설명해주세요.**  
  
엄격 모드(Strict Mode)는 ES5부터 도입된 자바스크립트의 기능으로, 코드를 엄격하게 검사하는 모드입니다. 엄격 모드를 사용하면 일부 에러를 발생시키고, 더 엄격한 규칙을 적용합니다. 엄격 모드의 장점은 다음과 같습니다:  
예상치 못한 에러를 예방합니다.  
더 효율적인 코드 작성을 유도합니다.  
안전하지 않은 기능을 비활성화합니다.

**콜 스택 (Call Stack)과 힙 (Heap)에 대해 설명해주세요.**  
  
콜 스택(Call Stack): 함수 호출과 관련된 정보를 저장하는 스택 자료구조입니다. 함수를 호출할 때 스택에 프레임이 추가되고, 함수가 반환될 때 스택에서 제거됩니다. 이러한 작업은 스택의 맨 위에서부터 순차적으로 이루어집니다.  
힙(Heap): 동적으로 할당된 메모리가 저장되는 공간으로, 객체와 배열 같은 데이터 구조가 저장됩니다. 힙은 메모리 관리를 위해 가비지 컬렉션과 함께 사용됩니다.

**Rest 연산자와 Spread 연산자에 대해 설명해주세요.**  
  
Rest 연산자(...): 함수의 매개변수나 배열/객체에서 일부 요소를 추출하여 나머지를 하나의 변수로 모을 때 사용됩니다. Rest 연산자는 함수 정의나 배열/객체 리터럴에서 활용할 수 있습니다.  
Spread 연산자(...): 배열이나 객체를 개별 요소로 분해하거나 복사할 때 사용됩니다. Spread 연산자는 배열 리터럴, 객체 리터럴, 함수 호출 시 인수 전달에 활용됩니다.

**제너레이터에 대해 설명해주세요.**  
제너레이터(Generator)는 함수의 실행을 일시 중지하고 나중에 다시 시작할 수 있는 함수입니다. 제너레이터는 function\* 키워드를 사용하여 정의하며, yield 키워드를 통해 값을 반환하고 실행을 일시 중지할 수 있습니다. 제너레이터는 주로 비동기 코드의 제어 흐름을 관리하기 위해 사용됩니다.

**이터러블과 이터레이터 프로토콜에 대해 설명해주세요.**  
  
이터러블(Iterable): 이터레이터를 반환할 수 있는 객체를 가리키며, 배열과 같은 자료구조는 기본적으로 이터러블입니다.  
이터레이터 프로토콜(Iterator Protocol): 이터러블 객체는 반복 가능한 요소를 나타내는 Symbol.iterator 메서드를 가집니다. 이 메서드는 이터레이터 객체를 반환하며, 이터레이터는 next() 메서드를 가지고 있어 값을 순차적으로 접근할 수 있게 합니다.

**JavaScript에서의 화살표 함수(Arrow Function), 템플릿 리터럴(Template Literal), 클래스(Class), 및 비구조화 할당(Destructuring Assignment)에 대해 설명하세요.**  
화살표 함수(Arrow Function): =>를 사용하여 함수를 간결하게 정의할 수 있는 ES6의 기능입니다. 주로 익명 함수를 정의할 때 사용하며, this 바인딩이 외부 스코프와 연결되어 동작합니다.

템플릿 리터럴(Template Literal): 백틱()으로 묶은 문자열 리터럴로, 문자열 내에 변수나 표현식을 쉽게 삽입할 수 있습니다. ${}` 문법을 사용하여 값을 삽입할 수 있습니다.

클래스(Class): ES6에서 도입된 클래스 문법으로, 객체 지향 프로그래밍(OOP)의 클래스 개념을 자바스크립트에 추가합니다. 클래스를 사용하여 객체를 생성하고 메서드를 정의할 수 있습니다.

비구조화 할당(Destructuring Assignment): 배열 또는 객체에서 값을 추출하여 변수에 할당하는 문법으로, 코드를 간결하게 작성하는 데 도움을 줍니다.

**const, let, var의 차이점과 각각의 사용 사례에 대해 설명해보세요.**  
const: 변수를 선언하고 할당한 후 다시 재할당할 수 없는 상수를 선언할 때 사용됩니다. 주로 변하지 않는 값에 사용됩니다.

let: 블록 스코프를 가지며, 재할당이 가능한 변수를 선언할 때 사용됩니다. 블록 내에서만 유효한 변수를 선언할 때 유용합니다.

var: 함수 스코프를 가지며, 변수를 선언할 때 사용됩니다. ES6 이전의 자바스크립트에서 사용되며, 현재는 let과 const를 더 권장합니다. 호이스팅 현상이 발생할 수 있습니다.

**JavaScript를 사용하여 이벤트 핸들링을 어떻게 구현하는지 설명해보세요.**이벤트 핸들링은 웹 페이지에서 사용자 상호 작용을 감지하고 처리하는 과정을 가리킵니다. JavaScript를 사용하여 이벤트 핸들링을 구현하는 방법은 다음과 같습니다:  
HTML 요소에 이벤트 리스너를 추가합니다. 예를 들어, 클릭 이벤트를 감지하려면 addEventListener 메서드를 사용합니다.  
이벤트 리스너 함수를 정의하고, 이벤트가 발생했을 때 실행될 코드를 작성합니다.  
이벤트가 발생했을 때 브라우저는 이벤트 리스너 함수를 호출하며, 사용자의 상호 작용에 대응하는 작업을 수행합니다.

**이벤트 위임(Event Delegation)의 개념과 사용 이유에 대해 설명해주세요.**  
이벤트 위임은 하위 요소에 대한 각각의 이벤트 리스너를 등록하는 대신 상위 요소에서 하위 요소의 이벤트를 한 번에 관리하는 패턴입니다. 이벤트 위임의 사용 이유는 다음과 같습니다:  
성능 향상: 많은 수의 하위 요소에 대한 이벤트 리스너를 등록하는 것보다 상위 요소 하나에서 이벤트를 관리하는 것이 성능 면에서 우수합니다.  
동적 요소 처리: 동적으로 생성되는 하위 요소에 대해서도 쉽게 이벤트를 처리할 수 있습니다.  
메모리 관리: 많은 이벤트 리스너를 등록하면 메모리 누수가 발생할 수 있으나, 이벤트 위임을 사용하면 이를 방지할 수 있습니다.

**자바스크립트에서 비동기 코드를 처리하는 방법과 콜백(callback) 함수의 문제점에 대해 설명해주세요.**  
  
비동기 코드 처리 방법:

콜백 함수: 이전 작업이 완료되면 다음 작업을 실행하는 콜백 함수를 사용합니다.

Promise: 비동기 작업의 상태와 결과를 다루기 쉽게 처리하는 객체입니다.

async/await: Promise를 기반으로 한 비동기 코드를 동기적으로 작성하는 방법입니다.

콜백 함수의 문제점:

콜백 지옥(callback hell): 콜백 함수가 중첩되면 코드 가독성이 떨어지고 유지 보수가 어려워집니다.  
에러 처리 어려움: 콜백 함수로 에러 처리를 하려면 각 단계에서 에러를 처리해야 하므로 복잡성이 증가합니다.

**Promise와 async/await의 차이점을 설명하고, 언제 어떤 것을 사용해야 할까요?**  
Promise는 비동기 작업을 더 쉽게 다룰 수 있도록 도와주는 객체로, 비동기 작업의 상태와 결과를 처리하는데 사용됩니다.  
async/await는 Promise를 기반으로 한 비동기 코드를 동기적으로 작성할 수 있도록 하는 문법입니다.

차이점:

Promise는 .then() 및 .catch()를 사용하여 비동기 작업을 처리하며, 비동기 작업을 명시적으로 다루어야 합니다.  
async/await는 비동기 코드를 동기 코드처럼 작성할 수 있으며, async 키워드로 함수를 정의하고 await 키워드로 Promise의 결과를 기다릴 수 있습니다.

사용 시기:

복잡한 비동기 작업을 간결하게 다루려면 async/await을 사용하는 것이 좋습니다.  
콜백 지옥을 피하고 가독성을 향상시키려면 async/await을 사용하는 것이 좋습니다.  
기존 코드에서 Promise를 사용하고 있는 경우, async/await과 함께 사용할 수 있습니다.