# 비동기 데이터 요청 방식 비교

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 구분 | **AJAX** | **fetch** | **async/await** | |  | | --- | |  |  |  | | --- | | **Axios** | |
| 기본 설명 | JavaScript에서 서버와 통신할 수 있게 해주는 오래된 방식 | 최신 표준 API로, Promise 기반 비동기 통신 방식 | fetch + Promise를 더 간결하게 사용하는 async 함수 형태 | Promise 기반의 HTTP 요청 라이브러리, 간결하고 직관적 |
| 기본 문법 | $.ajax({...}) 또는 XMLHttpRequest | fetch(url).then(...).catch(...) | const data = await fetch(url) | axios.get(url).then(...).catch(...) |
| 코드 예시 | $.ajax({  url: 'url',  method: 'GET',  success: function(res){  console.log(res);  }  error: function(err) {  console.error(err);  } }); | fetch('url')  .then(res => res.json())  .then(data => console.log(data))  .catch(err => console.error(err)); | async function getData() {  try {  const res = await fetch('url');  const data = await res.json();  console.log(data);  } catch(err) {  console.error(err);  } } | axios.get('url')  .then(res => console.log(res.data))  .catch(err => console.error(err)); |
| 장점 | jQuery 사용 시 간편, 오래된 브라우저 호환성 좋음 | 가독성 향상, 간단한 문법 | 동기 코드처럼 작성 가능해 가독성 최고 | 자동 JSON 변환, 응답 간결화,  HTTP 메서드 직관적 |
| 단점 | jQuery 의존성 또는 verbose한 XML 방식 | 예외 처리 등 다소 복잡할 수 있음 | async, await, try-catch 문법에 익숙해야 함 | 별도의 라이브러리 설치 필요,  브라우저 환경에서 사용 시 CDN 필요 |
| 사용 시기 | jQuery를 이미 쓰고 있다면 적합 | 현대 브라우저에서 간단한 API 호출 | 복잡한 비동기 로직을 간결하게 처리할 때 | 브라우저와 Node.js 모두에서 HTTP 요청을 간편하게 할 때 |