**1. 자바스크립트의 기본형 변수와 참조형 변수란 각각 무엇인가. 또한 기본형 변수와 참조형변수의 예제 코드를 작성하시오.**

변수의 종류에는 기본형(원시형)과 참조형 2가지가 있다.

**1. 기본형 변수 - 원시 타입(primitive type)**

**1) Number 타입 : type of Number**

**2) String 타입 : type of String**

**3) Boolean 타입 : type of Boolean**

**4) undefined 타입 : type of undefined**

**5) null 타입 : type of null**

**2. 참조형 변수 - 참조 타입(reference type)**

**1) 함수형 : type of function**

**2) 객체형 : type of Object**

**3) 배열형 : type of Object**

**4) dom : type of Object**

기본형(원시형)은 변수에 할당될 때 메모리 상에 고정된 크기로 저장되고 해당 변수가 데이터 값을 보관한다.

반면 참조형 데이터는 크기가 정해져있지 않고 변수에 할당될 때 값이 직접 해당 변수에 저장되지 않는다.

var a = 100;

var b = a;

b = 200;

console.log(a, b);

a라는 메모리 공간이 만들어졌고 100이란 값이 할당되었다.

그 다음 b라는 공간을 만들고 거기에 a를 복사했다.

그 후 b라는 공간에 200이란 값을 재할당했다.

이 경우 console에 a, b를 출력하면 각각 100, 200이 출력된다.

즉, a를 b에 할당하는 순간 **a 와 b는 각각의 독립된 공간**이 된다.

반면 a에 참조 타입 데이터인 배열을 할당하고

b에 a를 복사한 다음 b의 0번째 요소에 5를 재할당하였다.

var a = [1, 2, 3];

var b = a; //[1,2,3]

b[0] = 5;

console.log(a, b);

a = [5,2,3] , b = [5,2,3]

b뿐만 아니라 a배열의 0번째 값도 5로 변한 것을 볼 수 있다.

즉, a 배열의 값이 b에 복사되는 것이 아니라 **a의 주소값이 b에 복사되는 것**이다.

즉, var b = a라고 선언하는 순간에 a와 b는 **같은 메모리 공간을 공유**하게 되는 것이다.

2. 비동기 서버통신의 모듈로는 axios, fetch가 있다. 각 모듈의 장단점은 무엇인가.

# Ajax(Asynchronous Javascript And Xml)란?

JavaScript를 사용한 비동기 통신이며, 클라이언트와 서버간의 XML 데이터를 주고받는 기술이다.

XMLHttpRequest객체를 이용해서 전체 페이지를 리로드하지 않고 필요한 데이터만 로드할 수 있다.

## 1. axios

* 구형 브라우저를 지원한다.
* 응답 시간 초과를 설정하는 방법이 있다.
* JSON 데이터 자동변환이 가능하다.
* node.js에서의 사용이 가능하다.
* request aborting(요청 취소)가 가능하다.
* catch에 걸렸을 때, .then을 실행하지 않고, console창에 해당 에러 로그를 보여준다.
* return값은 Promise 객체 형태이다.

## 2. fetch

* JavaScript의 내장 라이브러리이기 때문에 import 하지 않고 사용할 수 있다.
* 라이브러리의 업데이트에 따른 에러 방지가 가능하다.  
  (React Native의 경우 업데이트가 잦아서 라이브러리가 쫓아오지 못하는 경우가 많은데, fetch의 경우 이를 방지할 수 있다.)
* 네트워크 에러가 발생했을 때 기다려야한다. (response timeout API 제공X)
* 지원하지 않는 브라우저가 있다.
* return값은 Promise 객체 형태이다.

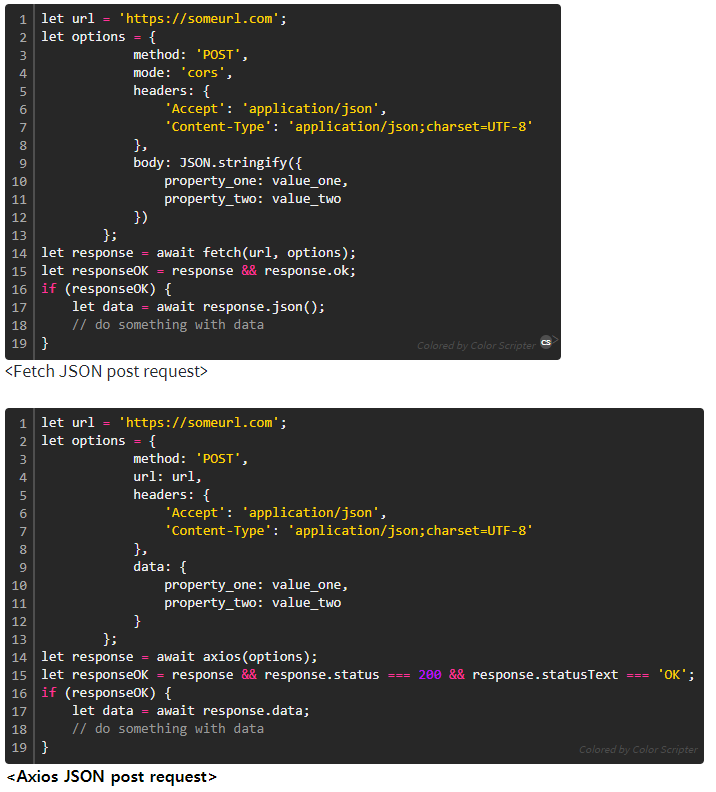
### 결론

### 1. 둘다 쓰는데 무리 없고, 잘 동작 하나, axios 는 주로 react, fetch 는 react native 에 사용한다.(React-native 의 빠른 업데이트 때문에 ..Quora 에 답변 다신 어떤분 개인 의견)

### 2. 전체적으로 이 둘은 비슷하나, axios 가 조금 더 장점이 많다. 동시에 fetch 의 단점도 꽤나 존재한다.

### 3. fetch 가 좀더 가볍고 axios 가 상대적으로 좀 더 무거운 느낌(제공하는 기능이 더 많다.)

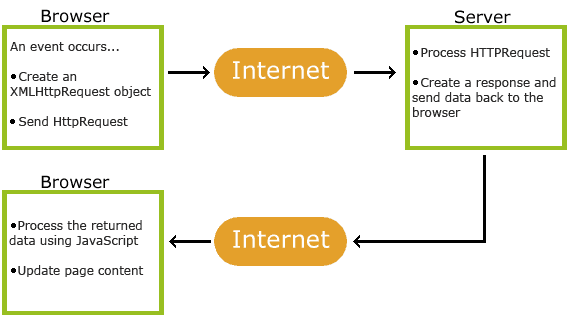
### 4. 요청 취소를 위해선 fetch 가 아닌 axios 등의 다른 라이브러리 사용이 필요하다.



* **코드** **자체는** **비슷비슷하다**. **용어의** **차이** **및** **객체로** **넘기는지** **혹은** string **화** **해서** **넘기는지** **등등** **중요** **부분만** **비교를** **해보자면**,
* **fetch** 의 **request** 함수는 **response** 객체가 **ok** 프로퍼티를포함하는것으로정상적인요청**/**응답을체크하고**, axios** 는 **status** 값이 **200** 인지와 **statusText** 를통해서확인한다**.**
* **fetch** 는 **response** 객체에 **.json()** 메소드를호출하여서 **json** 객체를얻고**, axios** 는 **response** 객체의 **data property** 에접근함으로써얻는다**.**

3. AJAX란 무엇인가. 예제 코드를 만드시오.

* Ajax는 JavaScript의 라이브러리중 하나이며 Asynchronous Javascript And Xml(비동기식 자바스크립트와 xml)의 약자
* 브라우저가 가지고있는 XMLHttpRequest 객체를 이용해서 전체 페이지를 새로 고치지 않고도 페이지의 일부만을 위한 데이터를 로드하는 기법
* 한마디로 정의하자면 JavaScript를 사용한 비동기 통신, 클라이언트와 서버간에 XML 데이터를 주고받는 기술이라고 할 수 있겠습니다.
* AJAX는 어떻게 작동하는가?



1. 웹페이지에서 이벤트 발생
2. JavaScript가 XMLHttpRequest 객체 생성
3. XMLHttpRequest 객체가 웹 서버에 데이터 요청
4. 서버에서 요청 처리
5. 요청에 관한 응답을 웹페이지로 돌려보냄
6. JavaScript가 응답 해석
7. JavaScript가 (이벤트에 대한) 적절한 동작 수행

* Ajax의 장점

1. 웹페이지의 속도향상
2. 서버의 처리가 완료 될때까지 기다리지 않고 처리 가능하다.
3. 서버에서 Data만 전송해면 되므로 전체적인 코딩의 양이 줄어든다.
4. 기존 웹에서는 불가능했던 다양한 UI를 가능하게 해준다. 사진 공유 사이트 Flickr의 경우 사진의 제목이나 태그를 페이지 리로드 없이 수정할 수 있다.

* Ajax 의 단점

1. 히스토리 관리가 안 된다. (보안에 좀 더 신경을 써야한다.)
2. 연속으로 데이터를 요청하면 서버 부하가 증가할 수 있다.
3. XMLHttpRequest를 통해 통신을 하는 경우 사용자에게 아무런 진행 정보가 주어지지 않는다. 그래서 아직 요청이 완료되지 않았는데 사용자가 페이지를 떠나거나 오작동할 가능성이 있다.

* GET 방식 예제

|  |
| --- |
| time.php: 현재시간 출력 |
| <?php  $d1 = new DateTime;  $d1->setTimezone(new DateTimezone("asia/seoul"));  echo $d1->format('H:i:s');  ?> |
| demo.html |
| <p>time : <span id="time"></span></p>  <input type="button" id="execute" value="execute" />  <script>  document.querySelector('input').addEventListener('click', function(event){     var xhr = new XMLHttpRequest(); // XMLHttpRequest 객체 생성     xhr.open('GET', './time.php');  // 접속 대상 지정, 첫번째 인자는 form 태그의 method에 대응하는 것으로 GET/POST 방식을 주로 사용한다. 두번째 인자는 접속하고자 하는 서버쪽 리소스의 주소로 form 태그의 action에 해당한다.     xhr.onreadystatechange = function(){ // 서버와 통신 끝났을 때 호출되는 이벤트         if(xhr.readyState === 4 && xhr.status === 200){  // 4: 통신 완료, 200: 통신 성공             document.querySelector('#time').innerHTML = xhr.responseText;  // responseText 프로퍼티는 서버에서 전송한 데이터를 담고 있다. 이것을 id가 time인 엘리먼트의 하위로 삽입한다. 이를 통해서 현재 서버에서 가져온 현재시간을 페이지 리로딩 없이 가져올 수 있다.         }     }     xhr.send();  });  </script> |

* POST 방식 예제

|  |
| --- |
| time2.php: 현재시간 출력 |
| <?php  $d1 = new DateTime;  $d1->setTimezone(new DateTimezone($\_POST['timezone']));  echo $d1->format($\_POST['format']);  ?> |
| demo2.html |
| **<p>time : <span id="time"></span></p>**  **<select id="timezone">**  **<option value="Asia/Seoul">asia/seoul</option>**  **<option value="America/New\_York">America/New\_York</option>**  **</select>**  **<select id="format">**  **<option value="Y-m-d H:i:s">Y-m-d H:i:s</option>**  **<option value="Y-m-d">Y-m-d</option>**  **</select>**  **<input type="button" id="execute" value="execute" />**  **<script>**  **document.querySelector('input').addEventListener('click', function(event){**  **var xhr = new XMLHttpRequest();**  **xhr.open('POST', './time2.php'); // 데이터 전송방식: POST**  **xhr.onreadystatechange = function(){**  **document.querySelector('#time').innerHTML = xhr.responseText;**  **}**  **xhr.setRequestHeader("Content-Type", "application/x-www-form-urlencoded"); // 서버로 전송할 데이터 타입의 형식(MIME)을 지정**  **var data = ''; // 서버로 전송할 데이터를 형식에 맞게 만든다. 이름=값&이름=값... 의 형식을 지켜야 한다.**  **data += 'timezone='+document.getElementById('timezone').value;**  **data += '&format='+document.getElementById('format').value;**  **xhr.send(data); // send 메소드 인자로 data 전달**  **});**  **</script>** |

* Jquery와의 시너지: 일반 Javascript만으로 Ajax를 하게되면 코딩량도 많아지고 브라우저별로 구현방법이 다른 단점이 있는데 jquery를 이용하면 더 적은 코딩량과 동일한 코딩방법으로 대부분의 브라우저에서 같은 동작을 할 수 있게 됩니다. jquery ajax를 사용하면, HTTP Get방식과 HTTP Post방식 모두를 사용하여 원격 서버로부터 데이터를 요청할 수 있습니다.

|  |
| --- |
| main.jsp |
| **<%@ page language="java" contentType="text/html; charset=UTF-8" pageEncoding="UTF-8"%>**  **<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">**  **<title>Ajax 기본적인 통신방법**  **<script type="text/javascript" language="javascript">**  **$(document).ready(function(){**    **$.ajax({**  **type : "GET", //전송방식을 지정한다 (POST,GET)**  **url : "InputData.jsp?type=2",**  **//호출 URL을 설정한다. GET방식일경우 뒤에 파라티터를 붙여서 사용해도된다.**  **dataType : "text",**  **//호출한 페이지의 형식이다. xml,json,html,text등의 여러 방식을 사용할 수 있다.**  **error : function(){**  **alert("통신실패!!!!");**  **},**  **success : function(Parse\_data){**  **$("#Parse\_Area").html(Parse\_data); //div에 받아온 값을 넣는다.**  **alert("통신 데이터 값 : " + Parse\_data);**  **}**  **});**  **});**  **</script>**  **</head>**  **<body>**  **<div id="Parse\_Area"gt;lt;/div>**  **</body>**  **</html>** |
| InputData.jsp |
| **<%@ page language="java" contentType="text/html; charset=EUC-KR"**  **pageEncoding="EUC-KR"%>**  **<!DOCTYPE>**  **<html>**  **<head>**  **<meta http-equiv="Content-Type" content="text/html; charset=EUC-KR">**  **<title>InputData</title>**  **</head>**  **<body>**  **데이터**  **</body>**  **</htm>** |



InputData.jsp에 있는 데이터들을 Main.jsp에서 받아옵니다. alert("통신데이터 값 : " + Parse\_data); 이 부분에서 InputData.jsp의 HTML태그들은 Alert창으로 받고 $("#Parse\_Area").html(Parse\_data); 이 부분에서 div안에 InputData의 Text를 받습니다.