Javascript_4

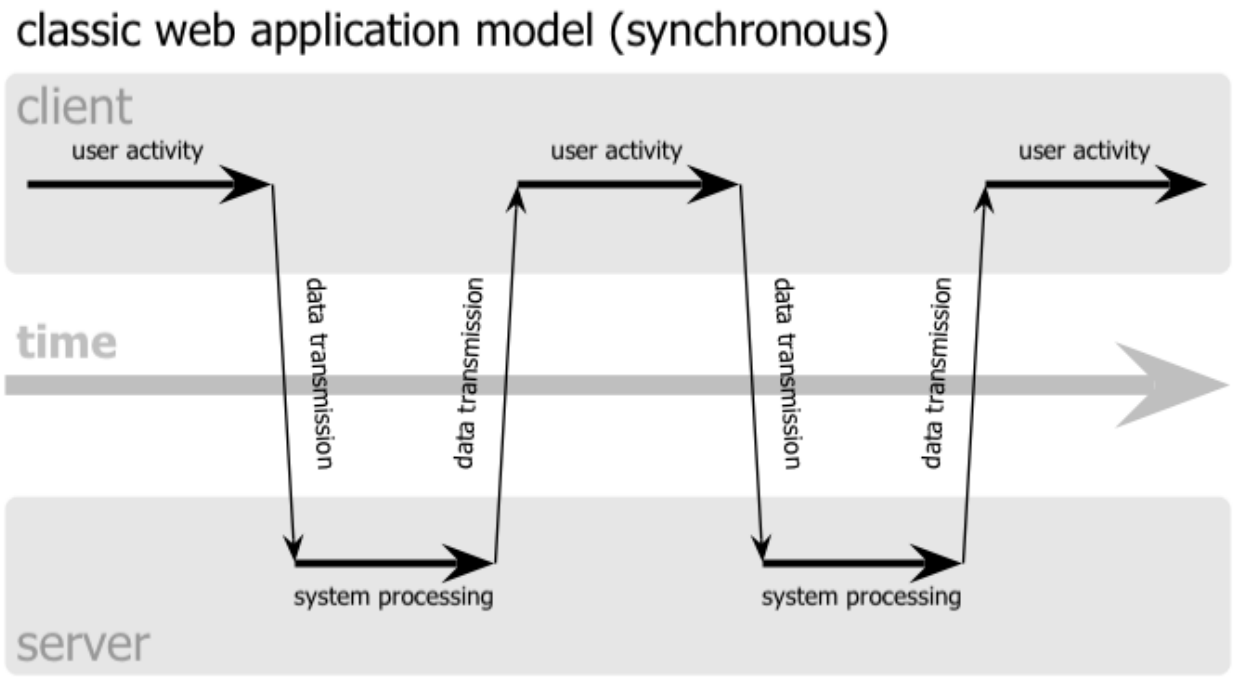
JSON (JavaScript Object Notation)

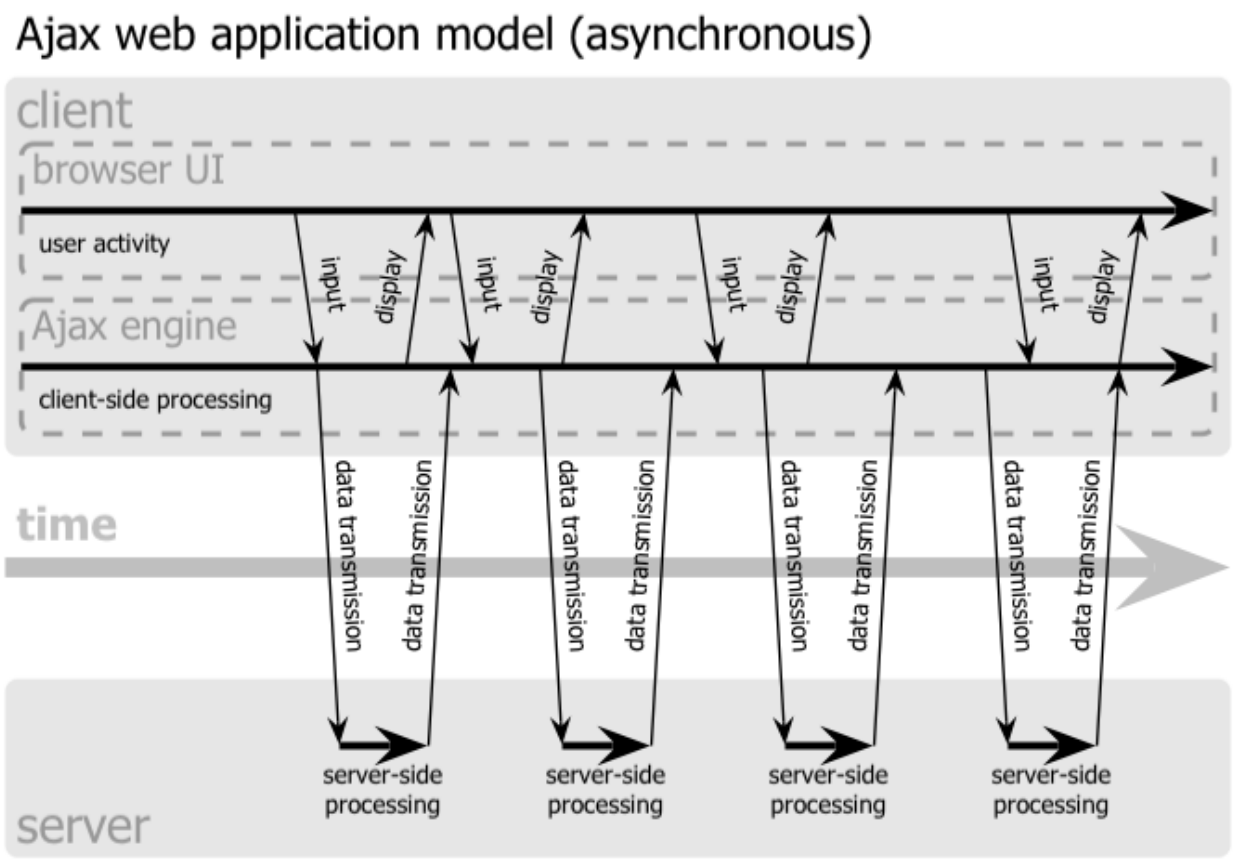
- JSON 이란 경량의 DATA-교환 형식이다. 이 형식은 사람이 읽고 쓰기에 용이하며, 기계가 분석하고 생성함에도 용이하다. JavaScript Programming Language, Standard ECMA-262 3rd Edition - December 1999의 일부에 토대를 두고 있다. JSON은 완벽하게 언어로 부터 독립적이지만 C-family 언어 - C, C++, C#, Java, JavaScript, Perl, Python 그외 다수의 프로그래머들에게 친숙한 관습을 사용하는 텍스트 형식이다. 이러한 속성들이 JSON을 이상적인 DATA-교환 언어로 만들고 있다.
- 현재 가장 널리 사용하고 있는 데이터 교환방식
- JSON 자세한 설명

```
var data = {
  "service" : [{"name":"tistory"}, {"name":"blog"}]
}

var dataString = JSON.stringify(data);

console.log(dataString);
console.log(JSON.parse(dataString));
```

AJAX (Ansynchronous Javscript And XML)

- AJAX란 사용자 요청을 즉시 처리하는 인터렉티브 형식의 웹 응용프로그램을 만들기 위한 컴퓨터 프로그래밍 방법이다.
- AJAX는 javascript, DHTML, XML, CSS, DOM 그리고 MS의 객체인 XMLHttpRequest 등 여러 가지 프로그래밍 도구들을 결합시킨다.
- AJAX는 새로운 페이지 전체가 새로 표시될 때까지 사용자들이 기다려야 했던 HTTP 요청과는 달리, 사용자가 어떤 동작을 수행하면 그 즉시 웹페이지의 내용이 수정되도록 해준다.
- AJAX는 XMLHttpRequest에 의존하고 있기 때문에, 초창기에는 MS의 IE에서 동작했지만 현재는 거의 모든 브라우져가 지원한다.
- AJAX flow





AJAX의 장단점

- 장점
  - 페이지 이동없이 화면 요소의 전환
  - 서버의 처리를 가다리지 않고 비동기 요청가능
  - 수신하는 데이터의 양을 줄일 수 있음

- 단점
  - 현재의 처리 상황에 대한 정보 필요 (비동기방식임으로 Sync문제)
  - 도메인이 다른경우 보안상 통신불가능

요청보내기

- XMLHttpRequest 객체 생성
- XMLHttpRequest의 open() 메서드로 특정 파일을 연다.
- 데이터를 수신하면 데이터로 무엇을 할 지 정한다.(Event/DOM handling)
- 객체에 요청을 전송하도록 send() 메서드를 사용한다.

```
리턴될 XML 데이터형식
<?xml version="1.0" ?>
<response>
  <success>true</success>
  <list>
    <name>홍길동</name>
    <part>Tistory</part>
  </list>
  <list>
    <name>전우치</name>
    <part>blog</part>
  </list>
  <msg>성공</msg>
</response>

// 리턴될 JSON 데이터형식
{"success": true, "response":[{"name":"홍길동","part":"Tistory"}, {"name":"전우치","part":"blog"}], "msg": "성공"}

<script type="text/javascript">
function XHR_get(url) {
 var xmlhttp;
   if (window.XMLHttpRequest)  {
     xmlhttp = new XMLHttpRequest();
   } else {
     xmlhttp = new ActiveXObject("Microsoft.XMLHTTP");
   }
   xmlhttp.onreadystatechange = function()  {
      if (xmlhttp.readyState == 4 && xmlhttp.status == 200) {
         // 성공시 처리할 구문
         // var result = xmlhttp.responseXML;
         var result = JSON.parse(xmlhttp.responseText);
         console.log(result.msg);
         //
      } else {
         // 실패시 처리할 구문
      }
   }
  xmlhttp.open('GET', 'ajax.php?type=blog', true);
  xmlhttp.setRequestHeader('Content-type', 'application/x-www-form-urlencoded');
  xmlhttp.send();
}

function XHR_post(url, param) {
  var xmlhttp;
   if (window.XMLHttpRequest)  {
     xmlhttp = new XMLHttpRequest();
   } else {
     xmlhttp = new ActiveXObject("Microsoft.XMLHTTP");
```

```
  }
  xmlhttp.onreadystatechange = function()  {
    if (xmlhttp.readyState == 4 && xmlhttp.status == 200) {
      // 성공시 처리할 구문
      // var result = xmlhttp.responseXML;
      var result = JSON.parse(xmlhttp.responseText);
      console.log(result.msg);
    } else {
      // 실패시 처리할 구문
    }
  }
  xmlhttp.open('POST', 'ajax.php', true);
  xmlhttp.setRequestHeader('Content-type', 'application/x-www-form-urlencoded');
  xmlhttp.send('type=blog');
}
```

```
</script>
```

- jQuery에서의 사용법

```
jQuery.ajax({
  type: "POST",
  url: "ajax.php",
  data: { type: "blog"},
  context: document.body,
  timeout: 5000,
  dataType: 'json',
  beforeSend: function() { console.log('전송시작') },
  error: function(result) {
    console.log("error message: " + result.msg);
  },
  complete: function() {console.log('전송완료') }
}).done(function(result) {
  console.log("success message: " + result.msg);
});


jQuery.ajax({
  type: "POST",// 전송타입을 결정, GET, POST, PUT, DELETE 설정가능
  url: "ajax.php",// 요청할 주소
  data: { type: "blog"},// 전송할 파라미터
  context: document.body,// 실행 함수안에서 this의 scope을 지정
  timeout: 5000,// 요청을 보내고 기다리는 시간 milliseconds
  dataType: 'json',// 받은 결과물이 해당 type이라 가정하고 parsing 해줌(json, xml, jsonp)
  beforeSend: function() { console.log('전송시작') },// 전송전에 실행하는 함수지정
  success: function(result) {// 요청한 결과를 받았을때 실행되는 함수지정
    console.log("success message: " + result.msg);
  },
  error: function(result) {// 요청이 실패한경우 실행되는 함수지정
    console.log("error message: " + result.msg);
  },
  complete: function() {console.log('전송완료') }// done, success, error 가 완료된후에 실행되는 함수지정
});
```

- http://api.jquery.com/jQuery.ajax/
- jsonp
  - json 타입의 결과물을 요청보낼대 callback 파라미터 보낸 함수이름으로 감싸서 보내주는 방식의 결과값을 타입
  - 해당 결과같이 리턴되는 동시에 해당 함수가 실행됨
  - 요청을 보내기전에 callback 함수가 등록된 script 태그를 생성시킨다.

```
요청한 URL
http://localhost/ajax.php?type=blog&callback=jsonpFunction

받은 데이터
jsonpFunction({"success": true, "response":[{"name":"홍길동","part":"Tistory"},
{"name":"전우치","part":"blog"}], "msg": "성공"});
```

performance

- 화면의 수정사항이 있을때는 한번 적용해서 화면의 렌더링을 최소화하자(특히 컨텐츠가 많을때)
    - 리스트를 추가할때는 루프를 돌면서 화면에 각각 추가하지 말고 documentFragment를 사용하자
    - elements에 스타일 줄때는 미리 정의된 CSS에 selector로 class를 바꾸나 추가해서 스타일을 변경하자
- innerHTML과 append 를 알맞게 사용하자
- for문에서 정방향보다는 역방향이 좀더 빠르다.

```
for (var i = 0, l = arr.length; i < l; i++} {
  var item = arr[i];
  ......
}

for (var i = arr.length; i--;) {
  var item = arr[i];
  ........
}
```