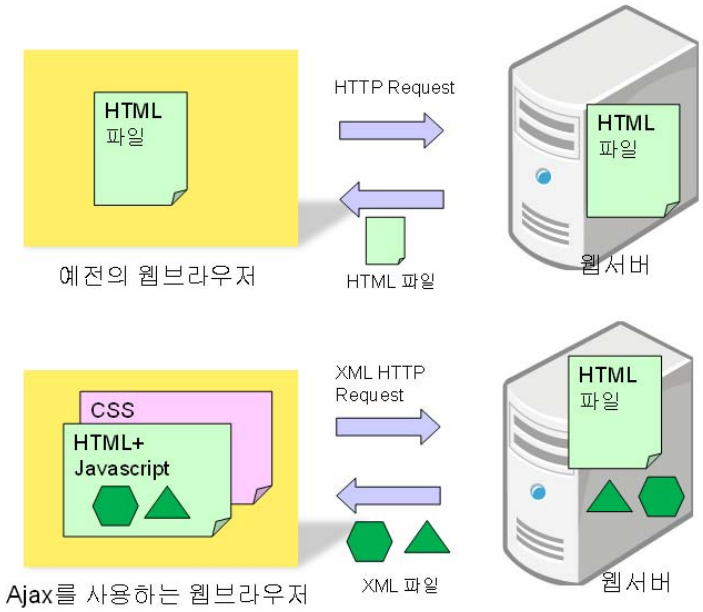
웹 프로그래밍

11. Ajax and JSON

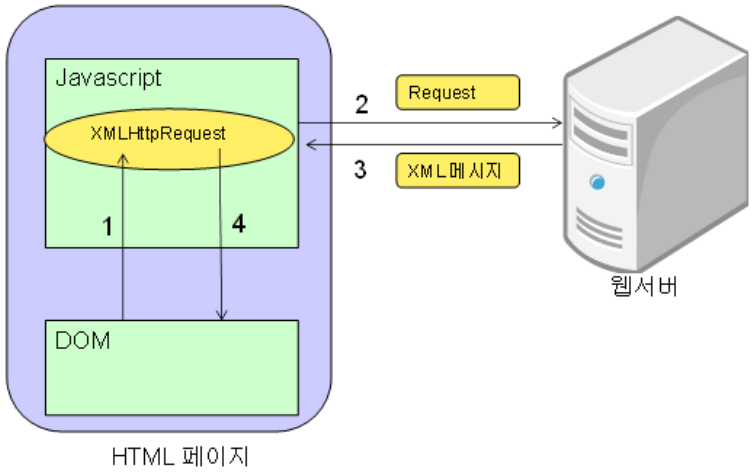
AJAX (Asynchronous JavaScript and XML)

- 웹 서버와 데이터를 교환하는 기술
  - 웹 서버 또는 DB에 저장되어 있는 "**데이터**"를 가져오거나,
  - 사용자가 브라우저를 통해 입력한 "**데이터**"를 웹 서버 또는 DB에 저장
- 비동기적 웹 페이지 갱신
  - 전체 페이지를 다시 적재하지 않고, **웹 페이지의 일부만 갱신**
    - .html 형식의 파일단위의 갱신이 아닌 XML 파일 조각 단위로 통신
  - ex) Google Map, Gmail, 유투브, 페이스북, Googld 검색어 제안
- 표준 데이터 전송 형식
  - XML 또는 JSON
- JavaScript
  - XML (또는 JSON) 형식의 데이터를 해석(parse)
  - HTML DOM 모델을 이용하여
  - 웹 페이지를 동적으로 갱신



AJAX의 동작 원리

- XMLHttpRequest
  - 웹 서버와의 http 통신을 위한 JS 객체
    - 웹 페이지 상태 정보를 XML 데이터로 구성하여 웹 서버에 송신
    - 웹 서버로부터 응답 XML 메시지 수신
    - XML 메시지를 파싱하여 DOM 객체 갱신



AJAX with JS

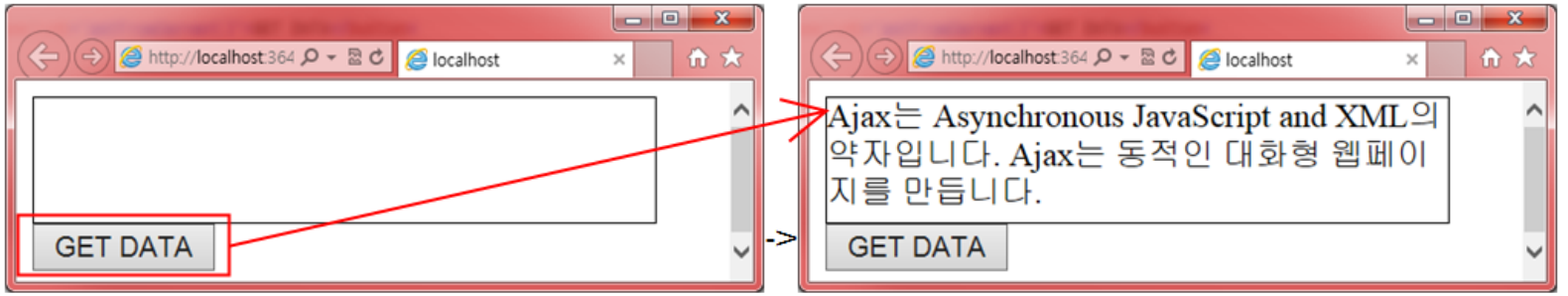
- 웹 서버에 다음의 testfile.txt 파일이 존재

testfile.txt

```
Ajax는 Asynchronous JavaScript and XML의 약자입니다.
Ajax는 동적인 대화형 웹페이지를 만듭니다.
```

- 버튼 클릭 시, testfile.txt에 있는 내용을 글상자에 출력

```
<!DOCTYPE html>
<html>
<body>
   <div id="target" style="width: 300px; height: 60px; border: solid; 1px black;">
   </div>
   <button type="button" onclick="getFromServer()">GET DATA</button>
</body>
</html>
```



AJAX with JS

```
<script>
    function getFromServer() {
      var req;
      if (window.XMLHttpRequest) {       // code for IE7+, Firefox, Chrome, Opera, Safari
        req = new XMLHttpRequest();
      }
      else {                             // code for IE6, IE5
        req = new ActiveXObject("Microsoft.XMLHTTP");
      }
      req.onreadystatechange = function () {          // 서버로부터 응답이 오면
        if (req.readyState == 4 && req.status == 200) {          // 응답 상태는 5가지
          document.getElementById("target").innerHTML = req.responseText;
        }
      }
      req.open("GET", "testfile1.txt", true);          //async = true
      req.send();
    }
  </script>
```

동기적 vs 비동기적

동기적: 서버에 요청을 했을 때, 해당 요청에 대한 서버의 응답을 받을 때 까지, 기다렸다가 다음 동작 실행

비동기적: 서버의 응답을 기다리는 것이 아니라, 다른 스트립트 및 이벤트를 독립적으로 실행. 단, 서버의 응답이 도착하면, 해당 응답 처리

AJAX with jQuery

```
$(document).ready(function () {
        $("button").click(function () {
            $.get("textfile1.txt", function (data, status){
                        $("#textbox").text(data);
            });
});
```

```
$(document).ready(function () {
        $("button").click(function () {
           $("#target").load("testfile1.txt");
        });
     });
```

```
$(document).ready(function () {
        $("button").click(function () {
           $("#target").load("sample.html  #myPara");
        });
     });
```

sample.html 문서에서 id=myPara인 요소의 콘텐츠만 로딩

jQuery methods for Ajax

종류	설명
load()	외부 콘텐츠를 가져올 때 사용
$.ajax()	데이터를 서버에 HTTP POST, GET 방식으로 전송할 수 있으며, HTML, XML, JSON, 텍스트 유형의 데이터를 요청할 수 있는 통합적인 메소드. $.post(), $.get(), $.getJSON() 메소드의 기능을 하나로 합쳐놓은 것
$.post()	데이터를 서버에 HTTP POST 방식으로 전송한 후, 서버 측의 응답을 받을 때 사용
$.get()	데이터를 서버에 HTTP GET 방식으로 전송한 후, 서버 측의 응답을 받을 때 사용
$.getJSON()	데이터를 서버에 HTTP GET 방식으로 전송한 후, 서버 측의 응답을 JSON 형식으로 받을 때 사용

$.ajax()

```
$.ajax({
        url: "호출 페이지 (script.php)",
        data: "전송 데이터",
        dataType: "요청 데이터 형식 ("html","xml","json","text" 등)",
        method: "http 전송 방식 ("POST","GET")"
        .....
})
  .done(function () { alert("success"); })          // 전송 성공 시 call-back 함수
  .fail(function () { alert("fail");. })            // 전송 실패 시 call-back 함수
  .always(function (){ alert("complete"); });       // 항상 수행할 call-back  함수
```

```
var req = $.ajax({
        url: "호출 페이지 (script.php)",
        data: "전송 데이터",
        dataType: "요청 데이터 형식 ("html","xml","json","text" 등)",
        method: "http 전송 방식 ("POST","GET")"
        .....
});

req.done(function (data) {      .....          } );
req.fail(function (jqXHR, status){      .....      });
```

eXtensible Markup Language(XML)

- **XML**: a "skeleton" for creating markup languages
- you already know it!
  - syntax is identical to HTML's:
  - *<element attribute="value">content</element>*
- languages written in XML specify:
  - names of **tags** — in HTML: h1, div, img, etc.
  - names of **attributes** — in HTML: id/class, src, href, etc.
  - **rules** about how they go together — inXHTML: inline vs. block-level elements
- used to present complex data in human-readable form
  - "self-describing data"

XML file

```
<?xml version="1.0" encoding="UTF-8"?>   <!-- XML prolog -->
<note>                                   <!-- root element -->
  <to>Tove</to>
  <from>Jani</from>                      <!-- element ("tag") -->
  <subject>Reminder</subject>            <!-- content of element -->
  <message language="english">           <!-- attribute and its value -->
    Don't forget me this weekend!
  </message>
</note>
```

- begins with an <?xml ... ?> header tag ("**prolog**")
- has a single **root element** (in this case, **note**)
- tag, attribute, and comment syntax is just like HTML

Uses of XML

- XML data comes from many sources on the web:
  - **web servers** store data as XML files
  - **databases** sometimes return query results as XML
  - **web services** use XML to communicate
- XML is the *de facto (real)* universal format for exchange of data
- XML languages are used for music, math, vector graphics
- popular use: RSS(Rich Site Summary) for news feeds & podcasts

Pros and cons of XML

- pro:
  - standard open format
  - can represent almost any general kind of data (record, list, tree)
  - easy to read (for humans and computers)
  - lots of tools exist for working with XML in many languages
- con:
  - bulky syntax/structure makes files large; can decrease performance (example)
  - JavaScript code to navigate the XML DOM is bulky and generally not fun

JSON (JavaScript Object Notation)

- Data format that represents data as a set of JavaScript objects
  - Douglas Crockford에 의하여 개발
    - RFC 4627에 기술
  - natively supported by all modern browsers (and libraries to support it in old ones)
  - not yet as popular as XML, but steadily rising due to its simplicity and ease of use
    - 종료 태그 없고 간결, 배열 사용 가능
  - 공식적인 타입은 application/json
  - 파일 이름 확장자는 .json

XML vs JSON

```xml
<?xml version="1.0" encoding="UTF-8"?>
<note private="true">
  <from>Alice Smith (alice@example.com)</from>
  <to>Robert Jones (roberto@example.com)</to>
  <to>Charles Dodd (cdodd@example.com)</to>
  <subject>Tomorrow's "Birthday Bash" event!</subject>
  <message language="english">
    Hey guys, don't forget to call me this weekend!
  </message>
</note>
```

```json
{
  "private": "true",
  "from": "Alice Smith (alice@example.com)",
  "to": [
    "Robert Jones (roberto@example.com)",
    "Charles Dodd (cdodd@example.com)"
  ],
  "subject": "Tomorrow's \"Birthday Bash\" event!",
  "message": {
    "language": "english",
    "text": "Hey guys, don't forget to call me this weekend!"
  }
}
```

Browser JSON methods

method	description
`JSON.parse(`*string*`)`	converts the given string of JSON data into an equivalent JavaScript object and returns it
`JSON.stringify(`*object*`)`	converts the given object into a string of JSON data (the opposite of `JSON.parse`)

JSON 예제

```
<body>
   <h4>학생  명단</h4>
   <p style="background-color: yellow">
                        이름: <span id="name"></span> 
                        나이: <span id="age"></span>
   </p>
   <script>
      var s = '[' +
            '{"name":"Hong","age":"21" },' +
            '{"name":"Kim","age":"22" },' +
            '{"name":"Park","age":"23" }]';
      var students = JSON.parse(s);

      students[1].name = "Lee";
      document.getElementById("name").innerHTML = students[1].name;
      document.getElementById("age").innerHTML = students[1].age;
   </script>
</body>
```

students = JS object들의 배열

data.json


```
[{
"name":"박소영",
"id":"20180001",
"department":"소프트웨어학과",
"class":["C 프로그래밍","웹 프로그래밍"],
"phone":"010-1234-5678"
},
{
"name":"김건우",
"id":"20180002",
"department":"소프트웨어학과",
"class":["수치해석","웹 프로그래밍"],
"phone":"010-1111-1111"
},
{
"name":"김건이",
"id":"20180003",
"department":"소프트웨어학과",
"class":["자료구조","웹 프로그래밍","윈도우 프로그래밍"],
"phone":"010-2222-2222"
},
{
"name":"홍길동",
"id":"20180004",
"department":"소프트웨어학과",
"class":["웹 프로그래밍","DB"],
"phone":"010-3333-3333"
}]
```