AJAX

AJAX =
Asynchronous JavaScript And **X**

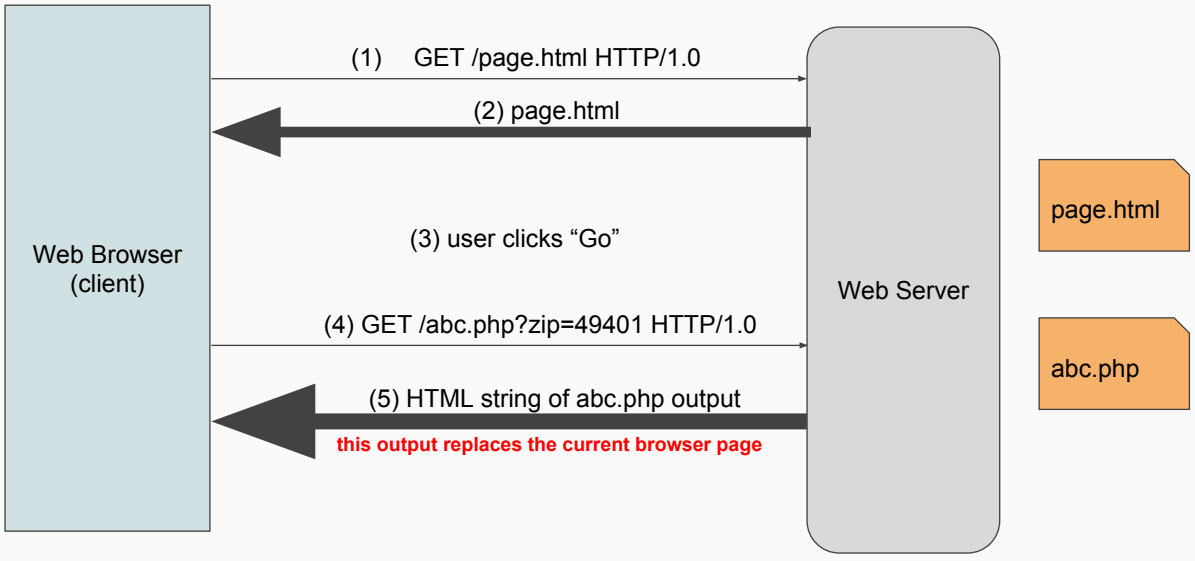
(Then) **X** ⇒ *XML*
(Now) **X** ⇒ *parseable data format*

Forms/Hyperlinks vs. AJAX

- `<form action="__">` or `<a href="___">` triggers a new HTTP request whose response **replaces** the entire current page on the browser
- Impossible to create *Single Page Applications* (SPAs)

- `<script>` initiates a new HTTP request whose response is handled **asynchronously** by the script itself
- Update to the current page is controlled entirely by the script (DOM manipulation)
- The browser stays on the same page, only part of the screen is updated

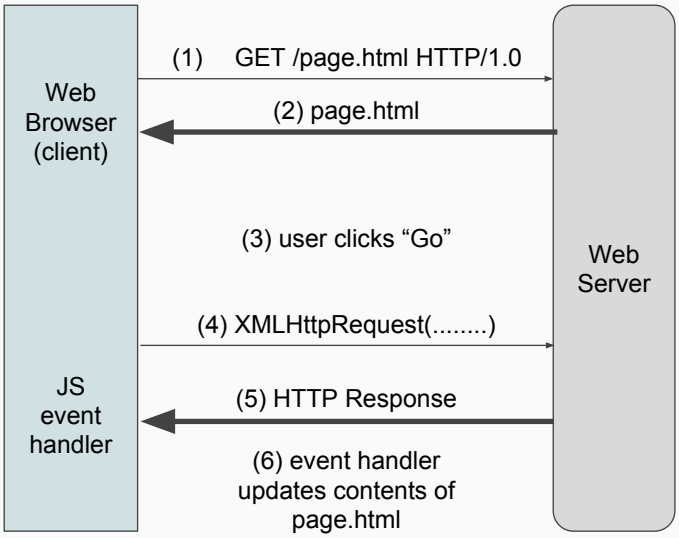
`<form action="abc.php" ...>`



XMLHttpRequest inside <script>

```
<script>
var xhr = new XMLHttpRequest();
/* setup event listeners */
xhr.onXXXXX = function() {
};
xhr.open ("GET",  // HTTP method
          url,    // target URL
          true); // asynchronous

xhr.send();       // async send
</script>
```



XMLHttpRequest

- A JavaScript class for sending HTTP request
- XMLHttpRequest2 (newer class) supported by
  - Chrome, FireFox, Opera, Safari
- XDomainRequest supported by
  - IE, Edge
- ActiveXObject (before IE7)

XMLHttpRequest Event Listeners (older interface)

```
<script>
var xhr = new XMLHttpRequest();
xhr.onreadystatechange = function() {
  if (xhr.readyState == XMLHttpRequest.DONE && xhr.status === 200) {
    // xhr.responseText is the body of the response
    // returned by the remote server
    console.log(xhr.responseText);
  }
};
xhr.open ("GET",      // HTTP method
          url,       // target URL
          true);     // asynchronous

xhr.send();          // async send, the response will be handled by onreadystatechange
</script>
```

XMLHttpRequest readyState

```
if (xhr.readyState == XMLHttpRequest._________)
```

State	Value	Description
UNSENT	0	XMLHttpRequest() object has been created
OPENED	1	open() has been called
HEADERS_RECEIVED	2	send() has been called and response status and headers are available
LOADING	3	response body is being downloaded
DONE	4	the operation is complete

XMLHttpRequest Event Listeners (newer interface)

```
<script>
var xhr = new XMLHttpRequest();
xhr.onload = function(e) {
  if (xhr.status === 200) {
    // xhr.responseText is the body of the response
    // returned by the remote server
    console.log(xhr.responseText);
  }
};
xhr.open ("GET",      // HTTP method
          url,       // target URL
          true);     // asynchronous

xhr.send();          // async send, the response will be handled by "onload"
</script>
```

Additional Reference on using XMLHttpRequest2

Event Handling Functions

Older XMLHttpRequest specifications define only **one** event handling function:

`onreadystatechange`

Newer XMLHttpRequest Event Handling Functions

Event Handler	Description
`onloadstart`	When the request starts
`onprogress`	When loading and sending data
`onabort`	When the request has been aborted
`onerror`	When the request has failed
`onload`	When the request has successfully completed
`ontimeout`	When the author specified timeout has passed before the request completed
`onloadend`	When the request has completed (either in success or failure)

Demo:
AJAX for Weather Underground

XMLHttpRequest Event Listeners (newer interface)

```
<script>
var xhr = new XMLHttpRequest();
xhr.onload = function(e) {
  if (xhr.status === 200) {
    var json = JSON.parse(xhr.responseText);
    console.log("Result", json);
  }
};
var baseurl = "http://api.wunderground.com/api";
var apiKey = "your-api-key-goes-here";
var url = `${baseurl}/${apiKey}/q/MI/Grand_Rapids.json`;  // use backquotes
xhr.open ("GET",      // HTTP method
          url,        // target URL
          true);      // asynchronous

xhr.send();           // async send, the response will be handled by "onload"
</script>
```

XMLHttpRequest Implementation by Browsers

Browser	XMLHttpRequest class
Chrome	`XMLHttpRequest` (Level 2)
Edge	`XDomainRequest`
FireFox	`XMLHttpRequest` (Level 2)
Internet Explorer	`XDomainRequest`
Opera	`XMLHttpRequest` (Level 2)
Safari	`XMLHttpRequest` (Level 2)

Handling Browser Variants

```
function createRequest (method, url) {
  var xhr = new XMLHttpRequest();
  if (“withCredentials” in xhr)          // are we using XMLHttpRequest level 2?
    xhr.open (method, url, true);
  else if (typeof XDomainRequest != ‘undefined’) { // IE or Edge
    xhr = new XDomainRequest();
    xhr.open(method, url);
  } else
    xhr = null;                          // unsupported by your browser!
  return xhr;
}
```

```
var xhr = createRequest (“GET”, “your-url-goes-here”);
if (xhr == null)
  throw new Error (“XMLHttpRequest is not supported”);
// more code here . . .
```

CORS (Cross-Origin Resource Sharing)

- **Traditionally,** browsers enforces **same-origin** policy
  - <script> in **www.example.org**/page.html can access data only from from www.example.org
- In 2014, a new W3C specification is published to enable *Cross-Origin Resource Sharing*
  - This feature must be **enabled by the server**!
  - Required header in server response: `Access-Control-Allow-Origin`