● AJAX,
● JSONP,
● HTML5,
● WebSockets.
With AJAX, a JavaScript script can send data to, and retrieve data from, a server asynchronously (in the background) without interfering with the display and behavior of the existing page.

Despite the name, the use of XML is not required. In fact, JSON is much more popular.
```javascript
var xhr = new XMLHttpRequest();
xhr.open('get', 'http://example/method');

xhr.onreadystatechange = function() {
    // Ready state 4 means the request is done
    if (xhr.readyState === 4) {
        if (xhr.status === 200) {
            alert('Success: ' + xhr.responseText);
        } else {
            alert('Error: ' + xhr.status);
        }
    }
}

xhr.send(null);
```
xhr.responseText = "document";
xhr.responseXML.documentElement

xhr.responseText = "json";
eval ( xhr.responseText )
(if you trust the response source!).
AJAX requests can be made only to URLs of the same domain (host and port) as the page.

AJAX is hence useful for communication with the server of a web application, but not for doing calls to a third-party API.

For remote API calls, several workarounds are used:

● JSONP (by far the most popular),
● Using the application server as a proxy (costly),
● iframes / using the URL to communicate (tricky),
● Messages (the clean way, in HTML5).
● Level 1 (1999)
● Level 2 (2008)
  ○ progress events,
  ○ support for cross-site requests,
  ○ handling of byte streams

http://www.w3.org/TR/XMLHttpRequest/
http://www.w3.org/TR/XMLHttpRequest2/
An alternative to AJAX, for requesting data from a server in a different domain.

- The client script generates the request by adding a `script` tag to the page:

```
<script type="application/javascript" src="http://directory/?id=42">
```

- The server returns a JavaScript containing a JSON value, wrapped into a function call (the padding):

```
callback({"id": 42,
  "name": "Vincent Simonet"});
```
The name of the padding is usually passed as an argument in the request:

```javascript
<script type="application/javascript"
    src="...?id=42&jsonp=mycallback">
mycallback({"id": 42,
    "name": "Vincent Simonet"});
```

JavaScript frameworks provide helper functions for making this transparent. E.g. in jQuery:

```javascript
$.ajax({url : 'http://.../?id=42',
    dataType : 'jsonp',
    jsonp : 'jsonp',
    success : function(data){}
});
```
- Only GET (POST is doable, but tricky),
- No access to HTTP headers (in request and response), including cookies.
In HTTP and AJAX, all exchanges are initiated by client requests. In some applications, it is useful to have the server pushing information to the client. E.g.:

- Notifications in a news website,
- Messages in a chat system,
- etc.
The client can make periodic requests to the server,

The client can make a request to the server, which will answer with an "infinite" response. Known as Comet.

Several implementations:

- Streaming (using iframe or special XmlHttpRequest),
- Long polling (using XmlHttpRequest or script tags).
WebSocket is a protocol providing full-duplex communications channels over a single TCP connection.

Enables a stream of messages.

Its only relationship to HTTP are:

● its handshake is interpreted by HTTP servers as an Upgrade request,
● it is using port 80 as default port.
GET /mychat HTTP/1.1
Host: server.example.com
Upgrade: websocket
Connection: Upgrade
Sec-WebSocket-Key: x3JJHMbDL1EzLkh9GBhXDW==
Sec-WebSocket-Protocol: chat
Sec-WebSocket-Version: 13
Origin: http://example.com

HTTP/1.1 101 Switching Protocols
Upgrade: websocket
Connection: Upgrade
Sec-WebSocket-Accept: HSmrc0sM YUkAGmm5OPpG2HaGWk=
Sec-WebSocket-Protocol: chat
var connection = new WebSocket('ws://.../echo',
[ 'soap', 'xmpp' ]); 

connection.onopen = function () {
  connection.send('Ping');
};

connection.onerror = function (error) {
  console.log('WebSocket Error ' + error);
};

connection.onmessage = function (e) {
  console.log('Server: ' + e.data);
};
- Java: [Jetty](#)
- Node.js: [ws](#), [WebSocket-Node](#)
- Python: [pywebsocket](#)
What is HTML5?
The 5th version of the HTML language, subsuming HTML 4.01 and XHTML 1.1

New/extended markup, and a galaxy of APIs.
Specification status at the beginning of 2013
Source: [html5test.com](http://html5test.com)

<table>
<thead>
<tr>
<th>Browser</th>
<th>Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>Chrome</td>
<td>463/500</td>
</tr>
<tr>
<td>Opera</td>
<td>442/500</td>
</tr>
<tr>
<td>Firefox</td>
<td>414/500</td>
</tr>
<tr>
<td>Safari</td>
<td>378/500</td>
</tr>
<tr>
<td>Internet Explorer</td>
<td>320/500</td>
</tr>
</tbody>
</table>
Main HTML5 features

- Semantic tags,
- Canvas,
- Video,
- Geo-localisation,
- Local storage,
- Offline,
- Improved forms,
- Microdata,
- History manipulation.
• Semantic replacements of `<div>` or `<span>`:
  `<nav>` `<header>` `<footer>` `<section>` `<hgroup>` `<article>` `<aside>` `<time>` `<mark>`
• Replacements of `<object>`:
  `<audio>` `<video>`
• Removal of some style tags:
  `<font>` `<center>` `<strike>` `<tt>`

(non-exhaustive list)
<canvas id="c" width="500" height="375"></canvas>

var c_canvas = document.getElementById("c");
var context = c_canvas.getContext("2d");
for (var x = 0.5; x < 500; x += 10) {
    context.moveTo(x, 0);
    context.lineTo(x, 375);
}
navigator.geolocation.getCurrentPosition("handle_success", "handle_error", "options");

function handle_success(position) {
  var latitude = position.coords.latitude;
  var longitude = position.coords.longitude;
  // let's do something interesting!
}

function handle_error(error) {
  alert(error.code + ': ' + error.message);
}
### POSITIONOPTIONS OBJECT

<table>
<thead>
<tr>
<th>Property</th>
<th>Type</th>
<th>Default</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>enableHighAccuracy</td>
<td>Boolean</td>
<td>false</td>
<td>true might be slower</td>
</tr>
<tr>
<td>timeout</td>
<td>long</td>
<td>(no default)</td>
<td>in milliseconds</td>
</tr>
<tr>
<td>maximumAge</td>
<td>long</td>
<td>0</td>
<td>in milliseconds</td>
</tr>
</tbody>
</table>

### POSITION OBJECT

<table>
<thead>
<tr>
<th>Property</th>
<th>Type</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>coords.latitude</td>
<td>double</td>
<td>decimal degrees</td>
</tr>
<tr>
<td>coords.longitude</td>
<td>double</td>
<td>decimal degrees</td>
</tr>
<tr>
<td>coords.altitude</td>
<td>double or null</td>
<td>meters above the reference ellipsoid</td>
</tr>
<tr>
<td>coords.accuracy</td>
<td>double</td>
<td>meters</td>
</tr>
<tr>
<td>coords.altitudeAccuracy</td>
<td>double or null</td>
<td>meters</td>
</tr>
<tr>
<td>coords.speed</td>
<td>double or null</td>
<td>meters/second</td>
</tr>
<tr>
<td>timestamp</td>
<td>DOMTimeStamp</td>
<td>like a Date() object</td>
</tr>
</tbody>
</table>
Improved forms

- New input types: color, date, datetime, datetime-local, email, month, number, range, search, tel, time, url, week
  - `<input type="color" name="favcolor">`
  - `<input type="number" name="quantity" min="1" max="5">`

- New input attributes: autocomplete, autofocus, multiple, min, max, pattern, required, etc.

- New elements: `<datalist>`, `<keygen>`, `<output>`

Use them!
<article itemscope itemtype="http://data-vocabulary.org/Organization">
  <h1 itemprop="name">Google, Inc.</h1>
  <p itemprop="address" itemscope itemtype="http://data-vocabulary.org/Address">
    <span itemprop="street-address">1600 Amphitheatre Parkway</span><br>
    <span itemprop="locality">Mountain View</span>, <span itemprop="region">CA</span><br>
    <span itemprop="postal-code">94043</span><br>
    <span itemprop="country-name">USA</span>
  </p>
  <p itemprop="tel">650-253-0000</p>
</article>
window.history.pushState("object or string", "Title", "/new-url");

window.history.replaceState("object or string", "Title", "/new-url");

window.addEventListener("popstate", function(e) {
    swapPhoto(location.pathname);
});