XMLHttpRequest

Example:

```
var http = new XMLHttpRequest();
function sendRequest(action, responseHandler) {
        http.open("GET", action);
        http.onreadystatechange = responseHandler;
        http.send(null);
}
```

Pros:

- Can return any format
- Permits any type of request.
- Works on any modern browser
- Standardized by w3c
- Very maintainable

Cons:

- Does not allow external files
- Opens a TCP connection for each XMLhttpRequest

Iframes

Example:

```
var iframe = document.createElement(”iframe”);
iframe.setAttribute(”id”, ”theframe”);
iframe.setAttribute(”src”, ”some/path/to/file.html”);
document.appendChild(iframe);
```

Pros:

- Allows for external files
- ”Returns” html instead of javascript code that is shown directly in the browser.
- Works using any browser (Very cross compatible)

Cons:

- Only returns html, not xml or json.
- Does only permit GET-requests

- Risk of cross-site-request-forgery attacks as you are not in control of what is on the external server.

Dynamically generated script tags

Example:

```
function loadScript () {
        var head = document.getElementsByTagName(”Head”)[0];
        var script = document.createElement(”script”);
        script.src = ”some/path/to/a/script.js”;
                script.type = ”text/javascript”;
head.appendChild(script);
}
```

This would find the <head> element and dynamicly add a <script src=”some/path/to/a/script.js” type=”text/javascript”> element

Pros:

- It is very crossbrowser compatible, as it works even in very old browsers that does not permit XMLHttpRequest
- It allows you to load scripts from external servers
- When activeX is turned off it will still work in IE etc (As opposed to XMLhttp)

Cons:

- Does only permit GET-requests
- Can only return executable javascript code (ie.: no xml, json or plaintext like XMLHttpRequest
- Risk of cross-site-request-forgery attacks as you are not in control of what is on the external server.