Same Origin Policy

The same-origin policy restricts how a document or script loaded from one origin can interact with a resource from another origin. It is a critical security mechanism for isolating potentially malicious documents.

Definition of an origin

Two pages have the same origin if the protocol, port (if one is specified), and host are the same for both pages. You'll see this referred to as the "scheme/host/port tuple" at times (where a "tuple" is a set of three components that together comprise a whole).

The following table gives examples of origin comparisons to the URL http://store.company.com/dir/page.html:

<table>
<thead>
<tr>
<th>URL</th>
<th>Outcome</th>
<th>Reason</th>
</tr>
</thead>
<tbody>
<tr>
<td><a href="http://store.company.com/dir2/other.html">http://store.company.com/dir2/other.html</a></td>
<td>Success</td>
<td></td>
</tr>
<tr>
<td><a href="http://store.company.com/dir/inner/another.html">http://store.company.com/dir/inner/another.html</a></td>
<td>Success</td>
<td></td>
</tr>
<tr>
<td><a href="https://store.company.com/secure.html">https://store.company.com/secure.html</a></td>
<td>Failure</td>
<td>Different protocol</td>
</tr>
<tr>
<td><a href="http://store.company.com:81/dir/etc.html">http://store.company.com:81/dir/etc.html</a></td>
<td>Failure</td>
<td>Different port</td>
</tr>
<tr>
<td><a href="http://news.company.com/dir/other.html">http://news.company.com/dir/other.html</a></td>
<td>Failure</td>
<td>Different host</td>
</tr>
</tbody>
</table>

See also [origin definition for file: URLs](#).

Inherited origins

Scripts executed from pages with an `about:blank` or `javascript:` URL inherit the origin of the document that opened that URL, since these types of URLs do not explicitly contain information about the server of origin. For example, `about:blank` is often used as a URL of a new, empty popup window into which the parent script writes content (e.g. via the `Window.open()` mechanism). If this popup window were to also contain code, that code would inherit the same origin as the script that created it. `data:` URLs get a new, empty, security context.

**Note:** Prior to Gecko 6.0, `data` URLs inherited the security context of the page currently in the browser window if the user enters a `data` URL into the location bar.

IE Exceptions

Internet Explorer has two major exceptions when it comes to same origin policy.
• Trust Zones: if both domains are in highly trusted zone e.g, corporate domains, then the same origin limitations are not applied
• Port: IE doesn't include port into Same Origin components, therefore http://company.com:81/index.html and http://company.com/index.html are considered from the same origin and no restrictions are applied.

These exceptions are non-standard and not supported in any other browser but would be helpful if developing an app for Windows RT (or) IE based web application.

Cross-origin network access

The same-origin policy controls interactions between two different origins, such as when you use XMLHttpRequest or an <img> element. These interactions are typically placed into three categories:

• Cross-origin writes are typically allowed. Examples are links, redirects and form submissions. Certain rarely used HTTP requests require preflight.
• Cross-origin embedding is typically allowed. Examples are listed below.
• Cross-origin reads are typically not allowed, but read access is often leaked by embedding. For example, you can read the width and height of an embedded image, the actions of an embedded script, or the availability of an embedded resource.

Here are some examples of resources which may be embedded cross-origin:

• JavaScript with <script src="..."></script>. Error messages for syntax errors are only available for same-origin scripts.
• CSS with <link rel="stylesheet" href="...">. Due to the relaxed syntax rules of CSS, cross-origin CSS requires a correct Content-Type header. Restrictions vary by browser: IE, Firefox, Chrome, Safari (scroll down to CVE-2010-0051) and Opera.
• Images with <img>. Supported image formats include PNG, JPEG, GIF, BMP, SVG, ...
• Media files with <video> and <audio>.
• Plug-ins with <object>, <embed> and <applet>.
• Fonts with @font-face. Some browsers allow cross-origin fonts, others require same-origin fonts.
• Anything with <frame> and <iframe>. A site can use the X-Frame-Options header to prevent this form of cross-origin interaction.

Cross-origin data storage access

Access to data stored in the browser such as localStorage and IndexedDB are separated by origin. Each origin gets its own separate storage, and JavaScript in one origin cannot read from or write to the storage belonging to another origin.

Cookies use a separate definition of origins. A page can set a cookie for its own domain or any parent domain, as long as the parent domain is not a public suffix. Firefox and Chrome use the
Public Suffix List to determine if a domain is a public suffix. Internet Explorer uses its own internal method to determine if a domain is a public suffix. The browser will make a cookie available to the given domain including any sub-domains, no matter which protocol (HTTP/HTTPS) or port is used. When you set a cookie, you can limit its availability using the Domain, Path, Secure and Http-Only flags. When you read a cookie, you cannot see from where it was set. Even if you use only secure https connections, any cookie you see may have been set using an insecure connection.

Reference