**Same Origin**

 Two URLs are said to have the “same origin” if they have the same protocol, domain, and port.  
  
These URLs all share the same origin:  
  
http://site.com  
http://site.com/  
http://site.com/my/page.html  
These ones do not:  
  
http://www.site.com (another domain: www.matters)  
http://site.org (another domain: .orgmatters)  
https://site.com (another protocol: https)  
http://site.com:8080 (another port: 8080)  
  
The “Same Origin” policy states that:  
If we have a reference to another window, e.g., a popup created by window.open or a window inside <iframe>, and that window comes from the same origin, then we have full access to that window.  
  
otherwise, if it comes from another origin, then we can’t access the content of that window: variables, document, anything.  
The only exception is location: we can change it, but we cannot read the location.

**Windows on subdomains: document.domain**

 Windows on Subdomains: document.domain  
States that two URLs with different domains have different origins, but if windows share the same second-level domain we can make the browser ignore that difference, so that they can be treated as coming from the “same origin” for the purposes of cross-window communication.

 Iframe: wrong document pitfall  
When an iframe comes from the same origin, and we may access its document, there’s a pitfall. It’s not related to cross-domain things, but important to know.  
  
The "Wrong Document" pitfall in the context of iframes (inline frames) refers to a situation where you try to manipulate or access elements within an iframe using JavaScript or other scripting languages, but you mistakenly attempt to do so from the wrong document or context. This can lead to unexpected behavior or errors in your web application.

**Iframe: wrong document pitfall**

 Iframe: wrong document pitfall  
When an iframe comes from the same origin, and we may access its document, there’s a pitfall. It’s not related to cross-domain things, but important to know.  
  
The "Wrong Document" pitfall in the context of iframes (inline frames) refers to a situation where you try to manipulate or access elements within an iframe using JavaScript or other scripting languages, but you mistakenly attempt to do so from the wrong document or context. This can lead to unexpected behavior or errors in your web application.

**window.frames**

 In JavaScript, the window.frames collection is an array-like object that provides access to all the iframe elements (inline frames) within the current window. Each iframe element within a window is represented as an object in the window.frames collection, and you can use this collection to interact with and manipulate iframes on a web page.  
  
An alternative way to get a window object for <iframe>– is to get it from the named collectionwindow.frames:  
  
By number: window.frames[0] – the window object for the first frame in the document.  
By name: window.frames.iframeName – the window object for the frame withname="iframeName".  
  
An iframe may have other iframes inside. The corresponding window objects form a hierarchy.  
  
Navigation links are:  
  
window.frames – the collection of “children” windows (for nested frames).  
window.parent – the reference to the “parent” (outer) window.  
window.top – the reference to the topmost parent window.

**The “sandbox” iframe attribute**

 The sandbox attribute allows for the exclusion of certain actions inside an <iframe> in order to prevent it from executing untrusted code. It “sandboxes” the iframe by treating it as coming from another origin and/or applying other limitations.  
  
There’s a “default set” of restrictions applied for <iframe sandbox src="...">. But it can be relaxed if we provide a space-separated list of restrictions that should not be applied as a value of the attribute, like this: <iframe sandbox="allow-forms allow-popups">.  
  
Here’s a list of limitations:  
  
allow-same-origin  
By default "sandbox" forces the “different origin” policy for the iframe. In other words, it makes the browser to treat the iframe as coming from another origin, even if its src points to the same site. With all implied restrictions for scripts. This option removes that feature.  
  
allow-top-navigation  
Allows the iframe to change parent.location.  
  
allow-forms  
Allows to submit forms from iframe.  
  
allow-scripts  
Allows to run scripts from the iframe.  
  
allow-popups  
Allows to window.open popups from the iframe.

**Cross Window Messaging**

 Cross Window Messaging  
The postMessage interface allows windows to talk to each other no matter which origin they are from, but only if they both agree and call corresponding JavaScript functions. That makes it safe for users.