JAVASCRIPT INTERMEDIATE

Week 1

Day 1

Popups and window methods

* A popup is a separate window which has its own independent JavaScript environment. So opening a popup from a third-party, non-trusted site is safe.
* It's very easy to open a popup.
* A popup can navigate (change URL) and send messages to the opener window.

Popup Blocking

Window.open

The syntax to open a popup is: Window.open (url, name, params)

**URL**: an URL to load into the new window.

**Name:** A name of the new window. Each window has a window.name, and here we can specify which window to use for the popup. If there’s already a window with such name – the given URL opens in it, otherwise a new window is opened.

**Params**

The configuration string for the new window. It contains settings, delimited by comma. There must be no spaces in params, for instance: 200, height=100.

Settings for params:

Position:

Left/top(numeric) – coordinates of the window top-left corner on the screen. There is limitation: a new window cannot be positioned offscreen.

width/height (numeric) – width and height of a new window. There is a limit on minimal width/height, so it’s impossible to create an invisible window.

**Accessing Window from Popup**

A popup may access the “opener” window as well using window.opener reference. It is null for all windows except popups.

If you run the code below, it replaces the opener (current) window content with “Test”;

Let newWin = window.open (“about:blank”, “hello”, “width=200, “height=200”);

newWin.document.write(

  "<script>window.opener.document.body.innerHTML = 'Test'<\/script>"

);

**Closing a popup**

To close a window: win.close().

To check if a window is closed: win.closed.

**Scrolling and resizing.**

**Methods to move/resize a window**

**Win.moveBy(x,y)**

Move the window relative to current position x pixels to the right and y pixels down. Negative values are allowed (to move left/up).

**Win.moveTo(x,y)**

Move the window to coordinate (x,y) on the screen.

**Win.resizeBy(width, height)**

Resize the window by given width/height relative to the current size. Negative values are allowed.

**Win.resizeTo(width, height)**

Resize the window to the given size.

There’s also window.onresize event.

**Only Popups**

To prevent abuse, The browser usually blocks these methods. The only work reliable on popups

That we opened, that have no additional tabs.

**Wind.scrollBy(x,y)**

Scroll the window x pixels right and y down relative the current scroll. Negative values are allowed.

**win.scrollTo(x,y)**

Scroll the window to the given coordinates (x,y).

**elem.scrollIntoView(top = true)**

**Scroll the window to make elem show up at the top (the default) or at the bottom for elem.scrollIntoView(false).**

**FOCUS/BLOR ON A WINDOW**

**Window.onblur = ()=> window.focus();**

**Day 2**

**Same Origin**

**Two URLs are said to have the “same origin” if they have same protocol, domain, and port.**

**These URLs all share the same origin**

* http://site.com
* http://site.com/
* http://site.com/my/page.html

**IFRAME**

An <iframe> tag hosts a separate embedded window, with its own separate document and window objects.

We can access them using properties:

* iframe.contentWindow to get the window inside the <iframe>.
* Iframe.contentDocument to get the document inside the <iframe>

The picture below shows errors for any operations except:

* Getting the reference to the inner window iframe.contentWindow – that’s allowed.
* Writing to location.

The iframe.onload event (on the <iframe>tag) is essentially the same as iframe.contentWindow.onload (on the embedded window object). It triggers when the embedded window fully loads with all resources.

Windows on Subdomains: document.domain

Two URLs with different domains have different origins. However, if windows share the same second-level domain, such as john.site.com, peter.site.com, and site.com, the browser can ignore this difference, treating them as coming from the same origin.

Document.domain = site.com

IFRAME: Wrong document pitfall

When an iframe comes from the same origin, and we may access its document, there’s a pitfall.

It is not related to cross-domain things, but important to know.

The document of a not-yet-loaded iframe should not be used as it is the wrong document and event handlers will be ignored. The correct document is found when iframe.onload triggers, when the entire iframe is loaded.

COLLECTION: WINDOW>FRAMES

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 first frame in the document.
* By name: window.frames.iframeName – the window object for the frame withname = “iframeName”.

Day 3

**Clickjacking**

Clickjacking is an attack that tricks users into clicking an invisible or disguised webpage element, potentially leading to malware downloads, malicious web pages, sensitive information disclosure, money transfers, or online purchases.

It is typically performed by displaying an invisible page or HTML element inside an iframe on top of the user's visible page, which could be a malicious or legitimate page.

There are several variations of the clickjacking attack, such as:

**Likejacking** – a technique in which the Facebook “Like” button is manipulated, causing users to “like” a page they actually did not intend to like.

**Cursorjacking** – a UI redressing technique that changes the cursor for the position the user perceives to another position. Cursorjacking relies on vulnerabilities in Flash and the Firefox browser, which have now been fixed.

**Day 4**

**ArrayBuffer**

Introduction Creating an ArrayBuffer Adding Elements Deleting Elements -= Assignment Operator remove method clear method Introduction An ArrayBuffer in Scala is a collection that comes under the sequence class. Like Arrays, it is a mutable collection, hence, when elements in an ArrayBuffer are modified, the original ArrayBuffer is updated.

ArrayBuffers are very similar to arrays with the difference that you can add and remove elements from an ArrayBuffer while adding and removing elements is not possible in simple Arrays. Array methods and operations are also available for ArrayBuffers.

Creating an ArrayBuffer#

To be able to use an ArrayBuffer, we first need to import it using the package scala.collection.mutable.ArrayBuffer.

Afterward, you can create and populate an ArrayBuffer the same way you create an Array.