FANSHAWE

INFO-6076

Web Security

JavaScript & Web Browsers

Agenda

- DOM & JavaScript
- JavaScript Injection
- Web Browsers
- Same Origin Policy
- Browser Exploitation Framework (BeEF)
- Lab 03 Overview

DOM & JavaScript

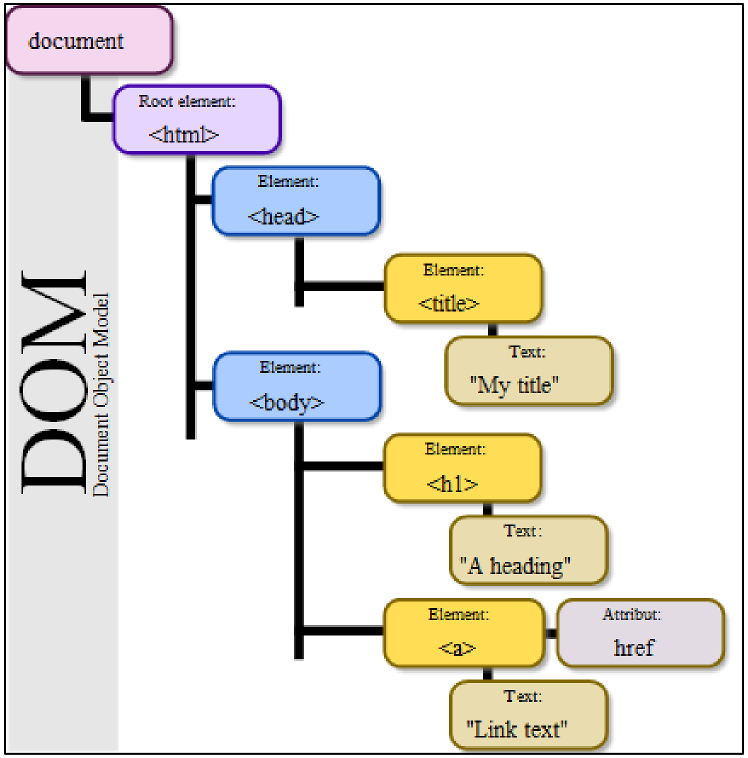
DOM & JavaScript

DOM: Document Object Model

Definition from w3.org:

The Document Object Model is a platform- and language-neutral interface that will allow programs and scripts to **dynamically access and update the content, structure and style of documents**. The document can be further processed and the results of that processing can be incorporated back into the presented page.

http://www.w3.org/DOM/

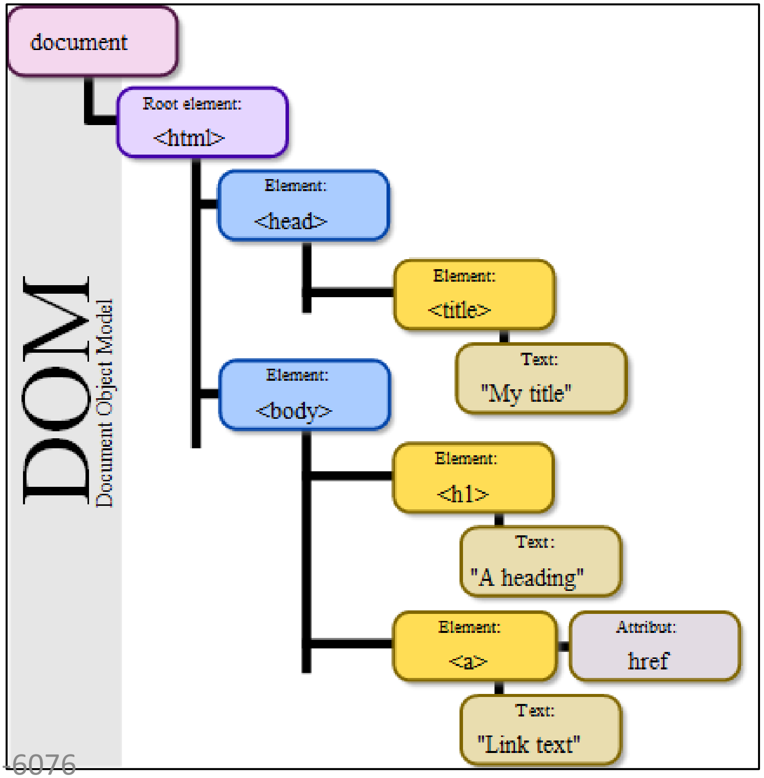


DOM: Document Object Model

DOM is a hierarchical model of a markup language (HTML,XML,XHTML).

DOM defines: **All Document nodes AND the methods to access them**

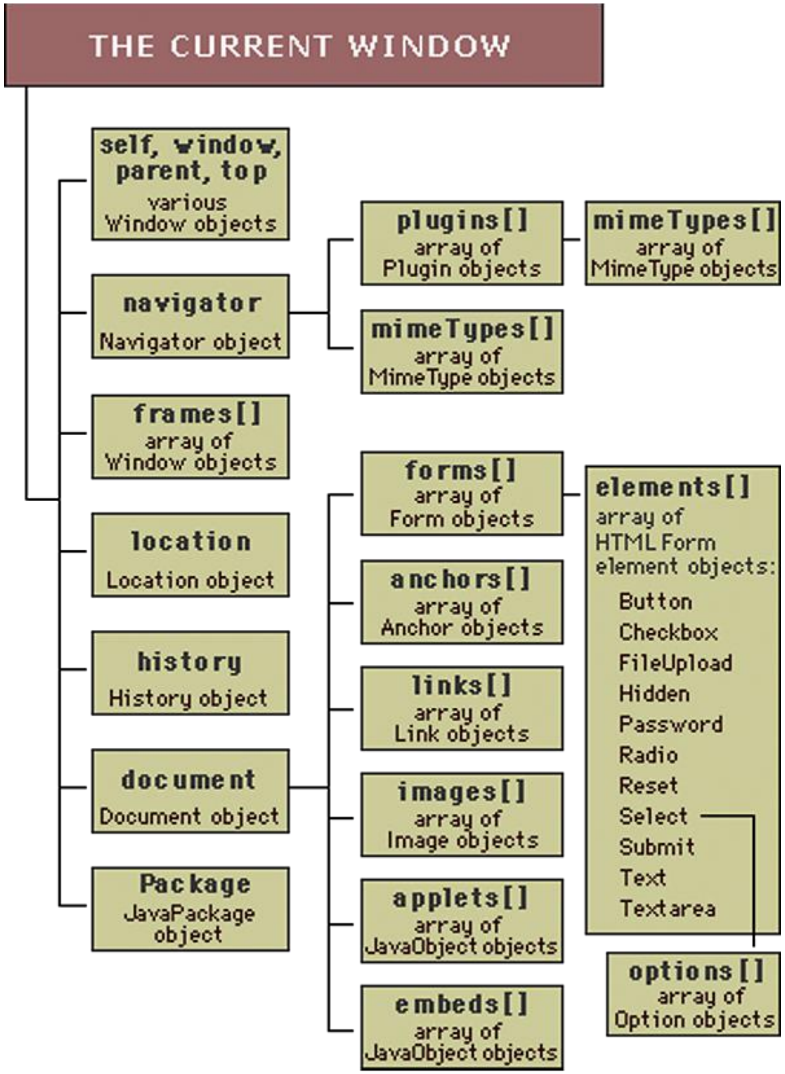
```
<html>
   <head>
            <title>My title</title>
   </head>
   <body>
            <h1>A heading</h1>
            <a href="file.html">My text</a>
   </body>
</html>
```



Element Node	contains an **HTML tag**
Attribute Node	describes **Element Node**
Text Node	contains **text**

Text Nodes are enclosed by **Element Nodes**

DOM: Document Object Model



HTML Page is a hierarchy:

The **Window** is the <u>parent</u> for a given <u>web page</u>.

Document is the <u>child</u> <u>with</u> the <u>objects</u> that are most commonly manipulated.

DOM provides an **API** (<u>A</u>pplication <u>P</u>rogramming <u>I</u>nterface) that allows programs to interact with **HTML** or **XML** documents.

What is object?

http://docs.oracle.com/javase/tutorial/java/concepts/object.html

Window Object Properties:

http://www.w3schools.com/jsref/obj_window.asp

DOM: 'window' Objects

document

The HTML document being displayed.

Using the global variable **document**, we can access **all the nodes** in the tree, as well as useful functions and other global information: **title, referrer, body, images, links, forms, etc.**

location

The URL of the document being displayed in this window.

If you set this property to a new URL, that URL will be loaded into this window.

history

Contains properties representing URLs the client has previously visited.

Property	Value
document.title	"A Simple Document"
document.fgColor	"#000000"
document.bgColor	"#ffffff"
location.href	"http://artmack.com"
history.length	"7"

More Examples:

location.reload() will refresh the window.

Window Objects:
http://www.w3schools.com/jsref/dom_obj_document.asp

Document Object Properties and Methods:
http://www.w3schools.com/jsref/dom_obj_document.asp

DOM: Why is this useful?

The **HTML Document Object Model** is a standard for structuring data on a web page.

The **HTML DOM** is made available to **scripts** running in the browser, not just the browser itself!

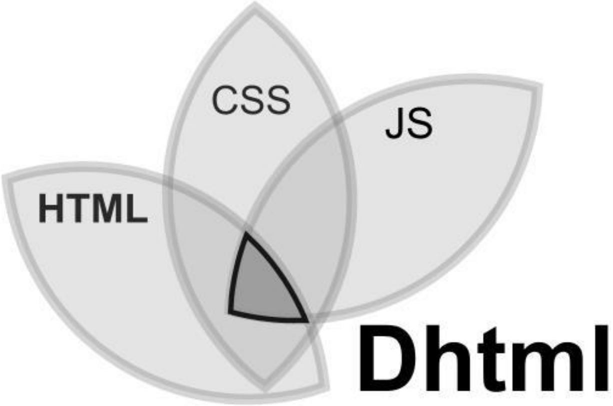
Scripts running in the browser can:

- Find things out about the state of the page (loading, closing, etc.)
- Change html nodes in response to events, including user requests
- Access sensitive information (history, cookies, etc.)

http://www.w3schools.com/htmldom/default.asp

DHTML: Components

DHTML (Dynamic HTML) is a collection of technologies brought together to create interactive websites.



DHTML includes:

- **DOM** (Document Object Model)
- **Scripting** language (JavaScript, Flash, etc.)
- **Presentation** language (CSS etc.)
- **Markup** languages (HTML, XML, etc.)

JavaScript: History

JavaScript created by Netscape (Mocha -> **LiveScript** (NN 2.0, 1995)->**JavaScript**)

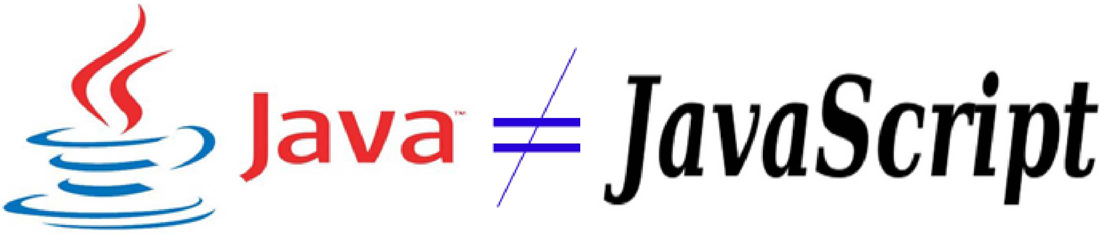
JScript created by Microsoft (IE 3.0, 1996)

Browser renderings are slightly different

Standardized by European Computer Manufacturers Association (**ECMA**)
http://www.ecma-international. org/publications/standards/Ecma-262.htm

ECMAScript language is primary used in the form of client-side **JavaScript**.

JavaScript is seldom used to write complete "programs”.
It's goal is to "enhance" (X)HTML.



JavaScript: Functionality

What can JavaScript (JS) do?

- It can detect the type of browser
- It can react to various events of the browser
- It can alter the structure of the html document (modify, delete, add tags on run-time)
- It can validate data before being sent to the server

Note: JavaScript cannot modify local (client) files!

Question: Do you see any security issues with JavaScript functionality?

JavaScript: Functionality

JavaScript comes in two primary forms:

- Client-Side Scripting
  - Takes places on the client's machine
  - Code is downloaded and executed by the browser, or run by a local application
  - JavaScript interpreters are embedded in a number of applications:
  - Adobe PDF, Apple's Dashboard Widgets, Microsoft's Gadgets, etc.
- Server-Side Scripting
  - Takes place on the server
  - Usually requires JavaScript engines (V8, Mozilla Rhino or SpiderMonkey, etc.)
  - node.js is an example of this
    - Used primary for CPU-intensive operations
    - Can support thousands of concurrent connections.
  - Typically browsers communicate with server-side scripts via AJAX calls

http://en.wikipedia.org/wiki/JavaScript

http://en.wikipedia.org/wiki/JavaScript_engine

JavaScript: Location inside (X)HTML

External
JS File**:**

```
<head>
   <script src="./js/myscript.js" type="text/javascript"></script>
 </head>
```

No <script> tags inside myscript.js:
alert('Hello World');

Embedded
JS Code:

```
<head>
    <script type="text/javascript">
            alert ('Hello World!');
   </script>
</head>
```

Inline
JS Code:

```
<head> …. </head>
<body>
…… <p>
     <script type="text/javascript">
           confirm ('Do you want to delete this record?');
     </script>
</p>
</body>
```

JavaScript: Location inside (X)HTML

HTML5 + CSS + Embedded JavaScript

```
<!doctype html>
<html lang="en">
        <head>
                <meta charset="UTF-8" />
                <title>My HTML5 Page</title>

                <style type="text/css">
                        h2 {  font: 40px Arial; color:#0000FF; }
                </style>

                <script type="text/javascript">
                        alert ('Hello World!');
                </script>
        </head>
        <body>
                <h2>Hello World!</h2>
                <p>This is my first paragraph</p>
        </body>
</html>
```

Can you guess what we'll get?

JavaScript: Location inside (X)HTML

HTML5 + CSS + Inline JavaScript

```
<!doctype html>
<html lang="en">
		<head>
				<meta charset="UTF-8" />
				<title>My HTML5 Page</title>

				<style type="text/css">
						h2 {  font: 40px Arial; color:#0000FF; }
				</style>

		</head>
		<body>
				<h2>Hello World!</h2>
				<p>This is my first paragraph</p>
				<script type="text/javascript">
						alert ('Hello World!');
				</script>
		</body>
</html>
```

Can you guess what we'll get?

JavaScript: Element manipulation

Example: simple manipulations with HTML elements

```
<!doctype html>
<html lang="en">
        <head>
                <meta charset="UTF-8" />
                <title>My HTML5 Page</title>

                <style type="text/css">
                        h2 {  font: 40px Arial; color:#0000FF; }
                </style>
        </head>
        <body>
                <h2 id="test">Hello World!</h2>
                <h2 id="test">I love INFO-6076!</h2>
                <p>This is my first paragraph</p>
                <script type="text/javascript">
                        document.getElementsByTagName('h2')[1].style.marginLeft = "200px";
                </script>
        </body>
</html>
```

JavaScript

Objects refers to windows, documents, images, tables, forms, buttons, links, etc.
Objects are an abstraction. They hold both data, and ways to manipulate the data.

Properties are object attributes.
Properties are defined by using the object's name + . + property name.
Example: background color is expressed by: `document.bgcolor`

Methods are actions applied to particular objects.
Methods are what objects can do.
Example: `document.write("Hello World")`

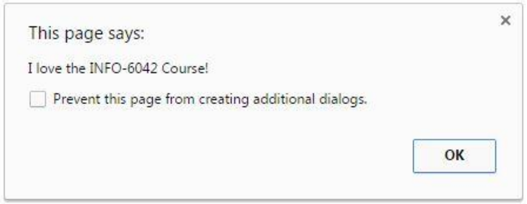
Events associate an **Object** with an **Action**. Typically user actions trigger events.
Example: `OnMouseover` event handler action can change an image.
`onSubmit` event handler sends a form.

Events are the major advantage of the JavaScript language that allow us to intercept not only an interactive event (mouse click, key pressed, element loosing focus), but also non-interactive events (page loaded, error, browser type, etc.)

JavaScript

JavaScript has 3 unique Pop-Up boxes/functions:

alert("….some text….");
displays text and the **Ok** button

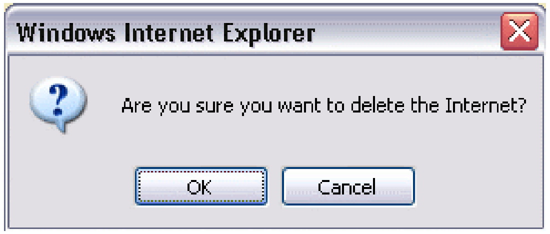


This page says:

I love the INFO-6042 Course!

☐ Prevent this page from creating additional dialogs.

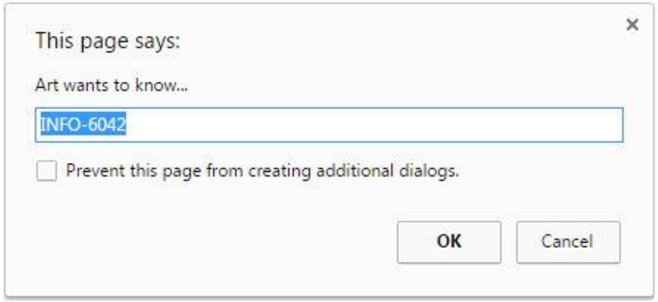
confirm("… some text…");
displays text and returns true if the **Ok** button is Clicked and false if the Cancel button is clicked



Windows Internet Explorer

Are you sure you want to delete the Internet?

prompt("text", "default value");
the user can enter an input value and then click **Ok** or **Cancel**



This page says:

Art wants to know...

INFO-6042

☐ Prevent this page from creating additional dialogs.

These are pre-defined simple functions

You can define your own function:

```
function DisplayMessage() {
                alert("Hello World!");
}
```

And call it as DisplayMessage();

- In general, function is a block of code that performs a single logical task
- Functions must be defined before they can be used

JavaScript Security

JavaScript Injection

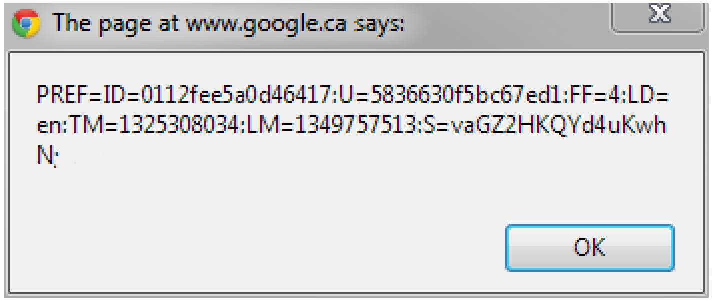
JavaScript Injection

JavaScript can be used not only for good purposes, but also for malicious purposes.

Using JavaScript an individual can manipulate (add, delete, change) HTML objects.
Examples: form input tags, cookie's that are currently set in the browser, etc.

In general, JavaScript Injection is a technique that allows an individual to alter the content of a web page. It can be done by inserting and/or executing JS code.

JavaScript can be executed not only from HTML page, but also from browser's URL or console using the **javascript:** command followed by any JavaScript command that can be executed.



```
PREF=ID=0112fee5a0d46417:U=5836630f5bc67ed1:FF=4:LD=
en:TM=1325308034:LM=1349757513:S=vaGZ2HKQYd4uKwh
N:
```

To see the cookie for the page you are on
you can use: javascript:alert(document.cookie);

JavaScript Injection

- javascript:alert();

  Can be used to retrieve information (or modify and retrieve):

  ```
  javascript:alert(document.cookie);
  javascript:alert(document.forms[0].to.value="something");
  ```

- javascript:void();

  Can be used to modify items without any visible notifications

  ```
  javascript:void(document.cookie="authorization=true");
  ```

In the example above a simple cookie allows/disallows access to a restricted page. We took advantage of this and granted ourselves the access by modifying the cookie.

JavaScript Injection

HTML Form Modifications

You can also use JavaScript to modify HTML forms (input names, values, etc.), including hidden forms, and disabled forms.

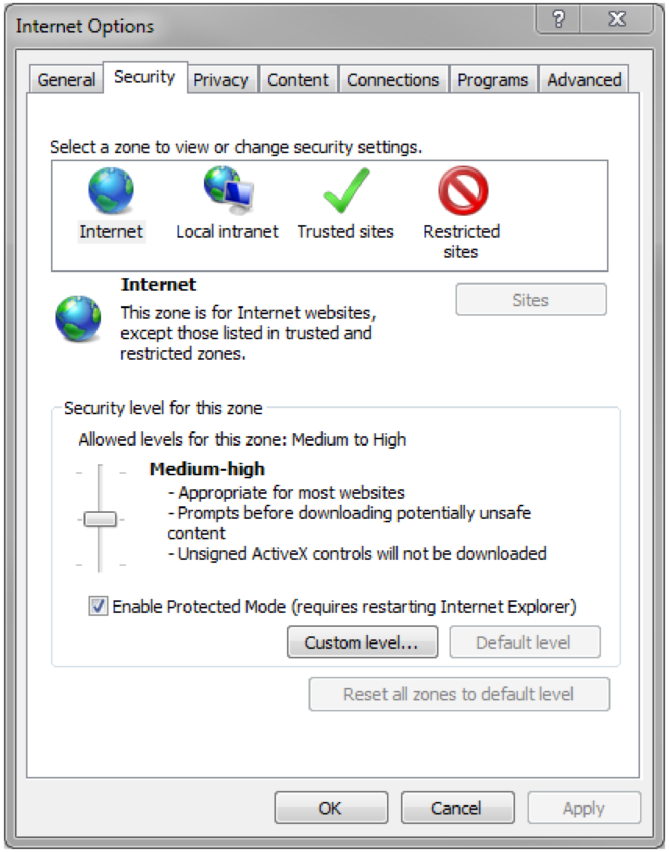
```
<form action="process_order.php" method="post">
        <input type="hidden" name="item" value="Cool T-shirt">
        <input type="hidden" name="price" value="99.99">
</form>
```

```
javascript:void(document.forms[0].price.value="0.01");
```

By looking at the page source code or using browser's Development Tools it's possible to find hidden information and modify it using javascript:void();

Controlling Scripts

Disabling Scripts in Internet Explorer (IE)



Internet Options

General | Security | Privacy | Content | Connections | Programs | Advanced

Select a zone to view or change security settings.

Internet | Local intranet | Trusted sites | Restricted sites

Internet

This zone is for Internet websites, except those listed in trusted and restricted zones.

Sites

Security level for this zone

Allowed levels for this zone: Medium to High

Medium-high

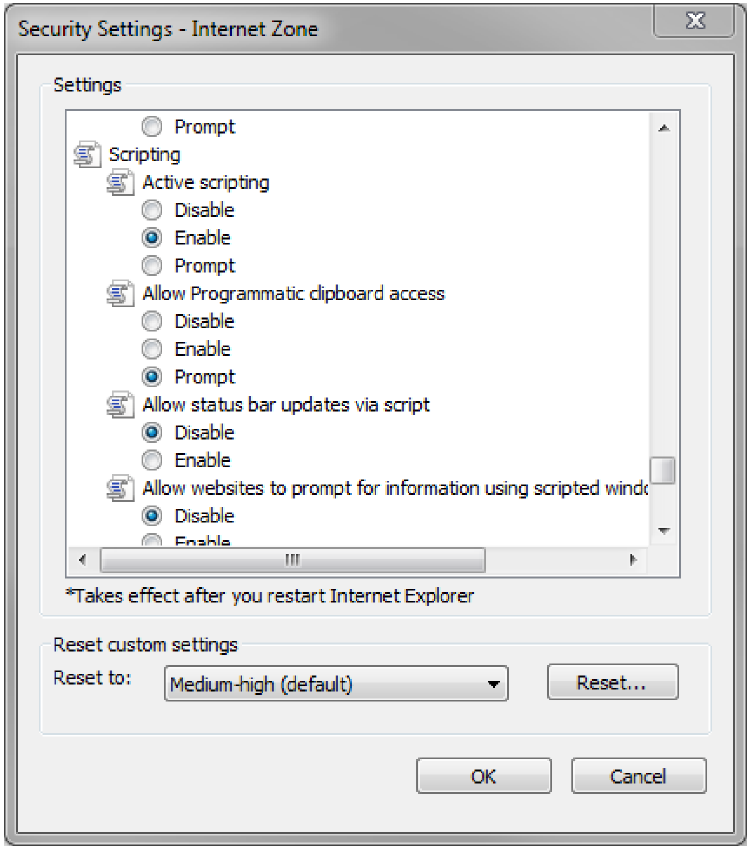
- Appropriate for most websites
- Prompts before downloading potentially unsafe content
- Unsigned ActiveX controls will not be downloaded

☑ Enable Protected Mode (requires restarting Internet Explorer)

Custom level... | Default level

Reset all zones to default level

OK | Cancel | Apply



Security Settings - Internet Zone

Settings

○ Prompt

Scripting

Active scripting
- ○ Disable
- ● Enable
- ○ Prompt

Allow Programmatic clipboard access
- ○ Disable
- ○ Enable
- ● Prompt

Allow status bar updates via script
- ● Disable
- ○ Enable

Allow websites to prompt for information using scripted windo
- ● Disable
- ○ Enable

*Takes effect after you restart Internet Explorer

Reset custom settings

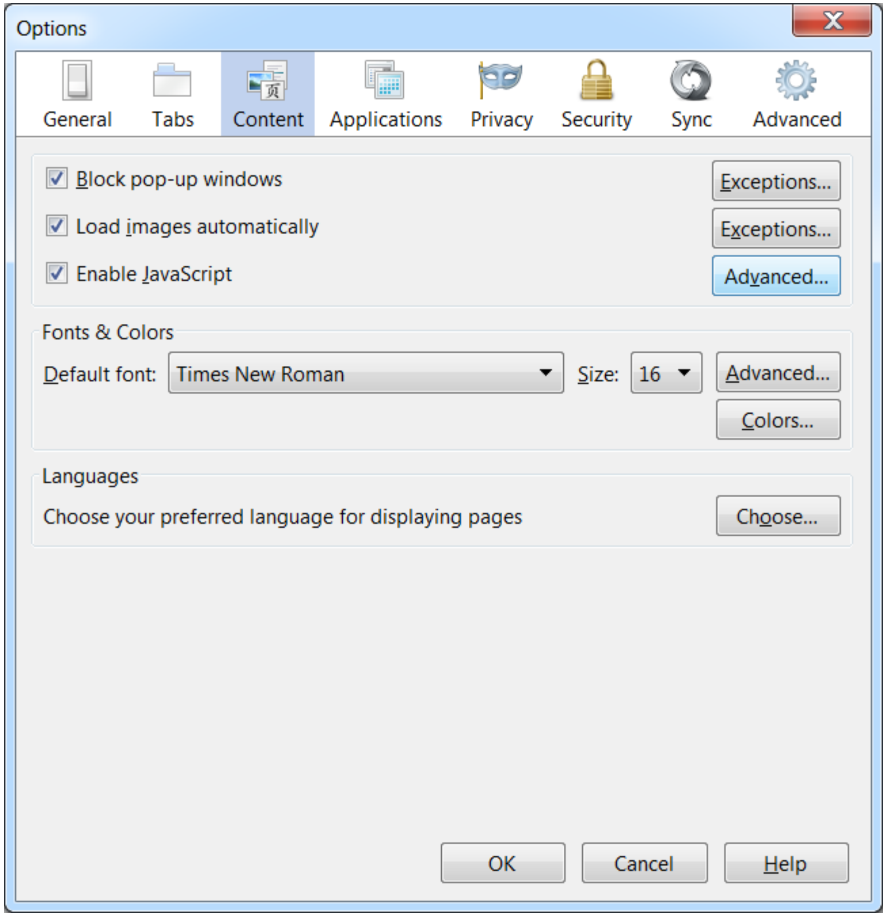
Reset to: Medium-high (default) | Reset...

OK | Cancel

Tools -> Internet Options -> Security -> Custom Level

Controlling Scripts

Disabling Scripts in FireFox (FF)



Options

☑ Block pop-up windows

☑ Load images automatically

☑ Enable JavaScript

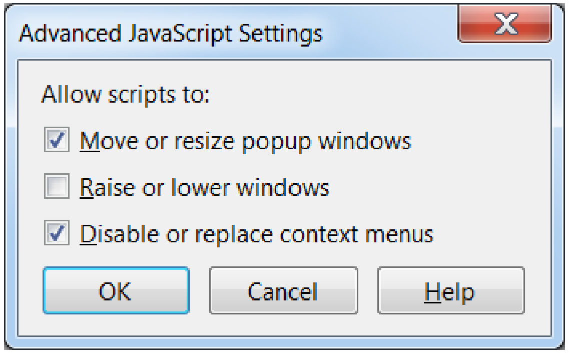
Fonts & Colors

Default font: Times New Roman

Size: 16

Languages

Choose your preferred language for displaying pages



Advanced JavaScript Settings

Allow scripts to:

☑ Move or resize popup windows

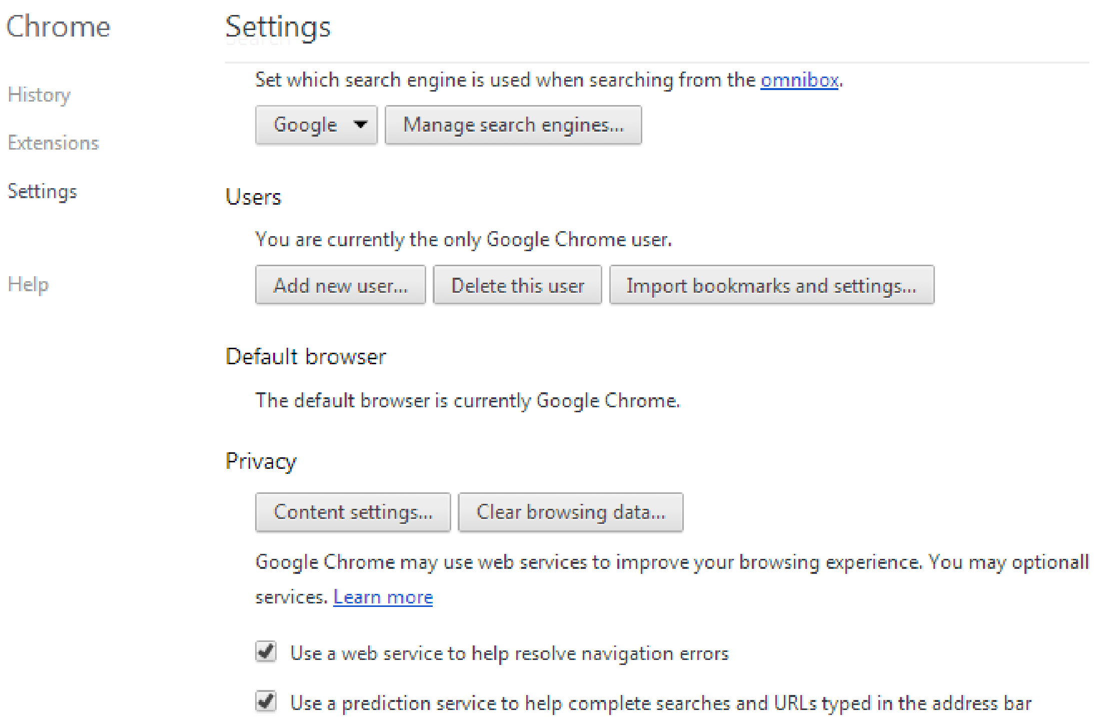
□ Raise or lower windows

☑ Disable or replace context menus

Firefox -> Options -> Content

Controlling Scripts

Disabling Scripts in Google Chrome (GC)



Settings

Set which search engine is used when searching from the omnibox.

Users

You are currently the only Google Chrome user.

Default browser

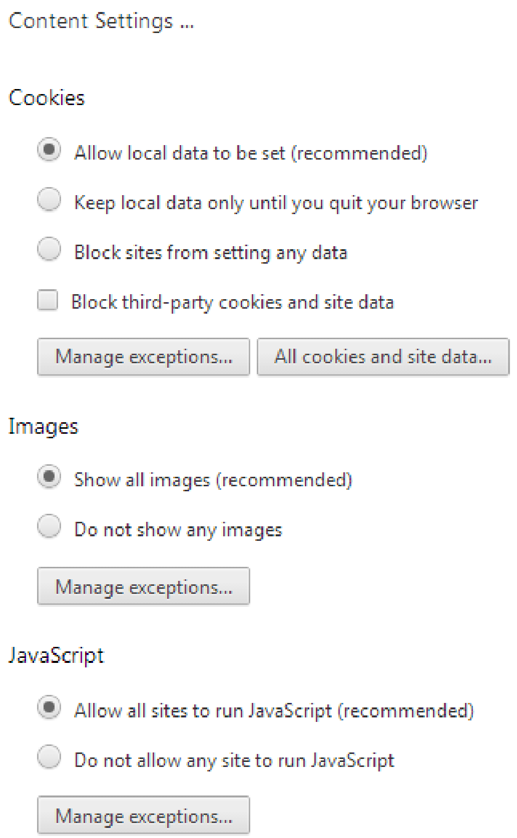
The default browser is currently Google Chrome.

Privacy

Google Chrome may use web services to improve your browsing experience. You may optionall services. Learn more

☑ Use a web service to help resolve navigation errors

☑ Use a prediction service to help complete searches and URLs typed in the address bar



Content Settings ...

Cookies

● Allow local data to be set (recommended)

○ Keep local data only until you quit your browser

○ Block sites from setting any data

□ Block third-party cookies and site data

Images

● Show all images (recommended)

○ Do not show any images

JavaScript

● Allow all sites to run JavaScript (recommended)

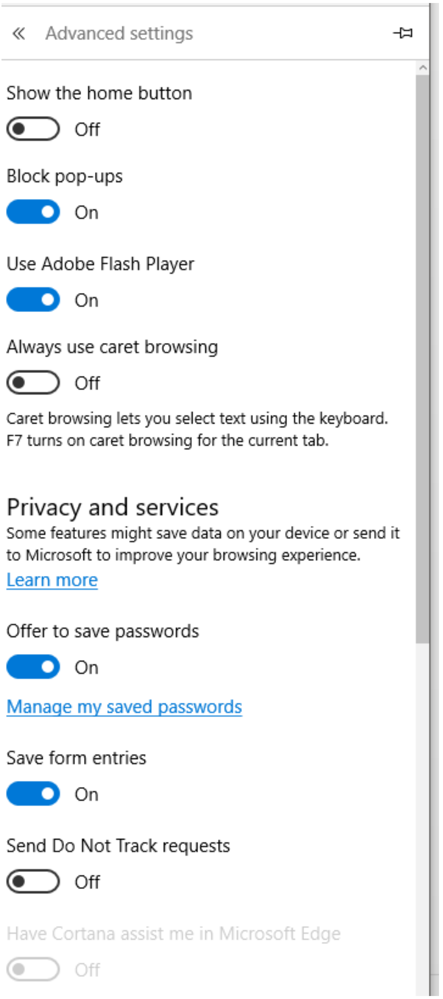
○ Do not allow any site to run JavaScript

Settings -> Show advanced settings -> Privacy -> Content settings…

Controlling Scripts

Disabling Scripts in Microsoft Edge

Settings -> View advanced settings -> ?



« Advanced settings

Show the home button

Off

Block pop-ups

On

Use Adobe Flash Player

On

Always use caret browsing

Off

Caret browsing lets you select text using the keyboard. F7 turns on caret browsing for the current tab.

Privacy and services

Some features might save data on your device or send it to Microsoft to improve your browsing experience.

Learn more

Offer to save passwords

On

Manage my saved passwords

Save form entries

On

Send Do Not Track requests

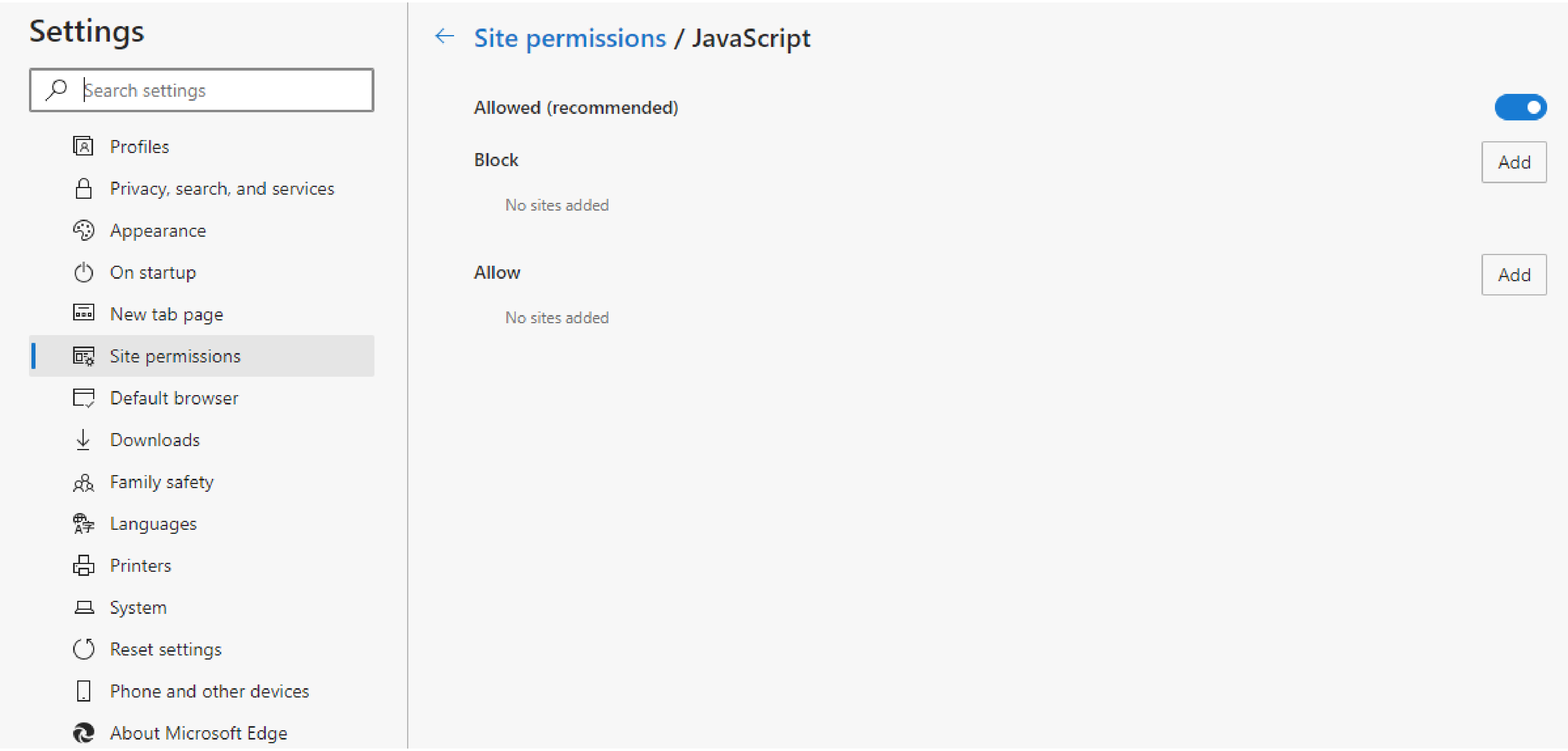
Off

Have Cortana assist me in Microsoft Edge

Off

Controlling Scripts

Disabling Scripts in Microsoft Edge



← Site permissions / JavaScript

Allowed (recommended) ●

Block

Add

No sites added

Allow

Add

No sites added

Controlling Scripts

Browsers allow you to set exceptions for trusted sites

There are also 3rd party tools that can be used to control script execution:

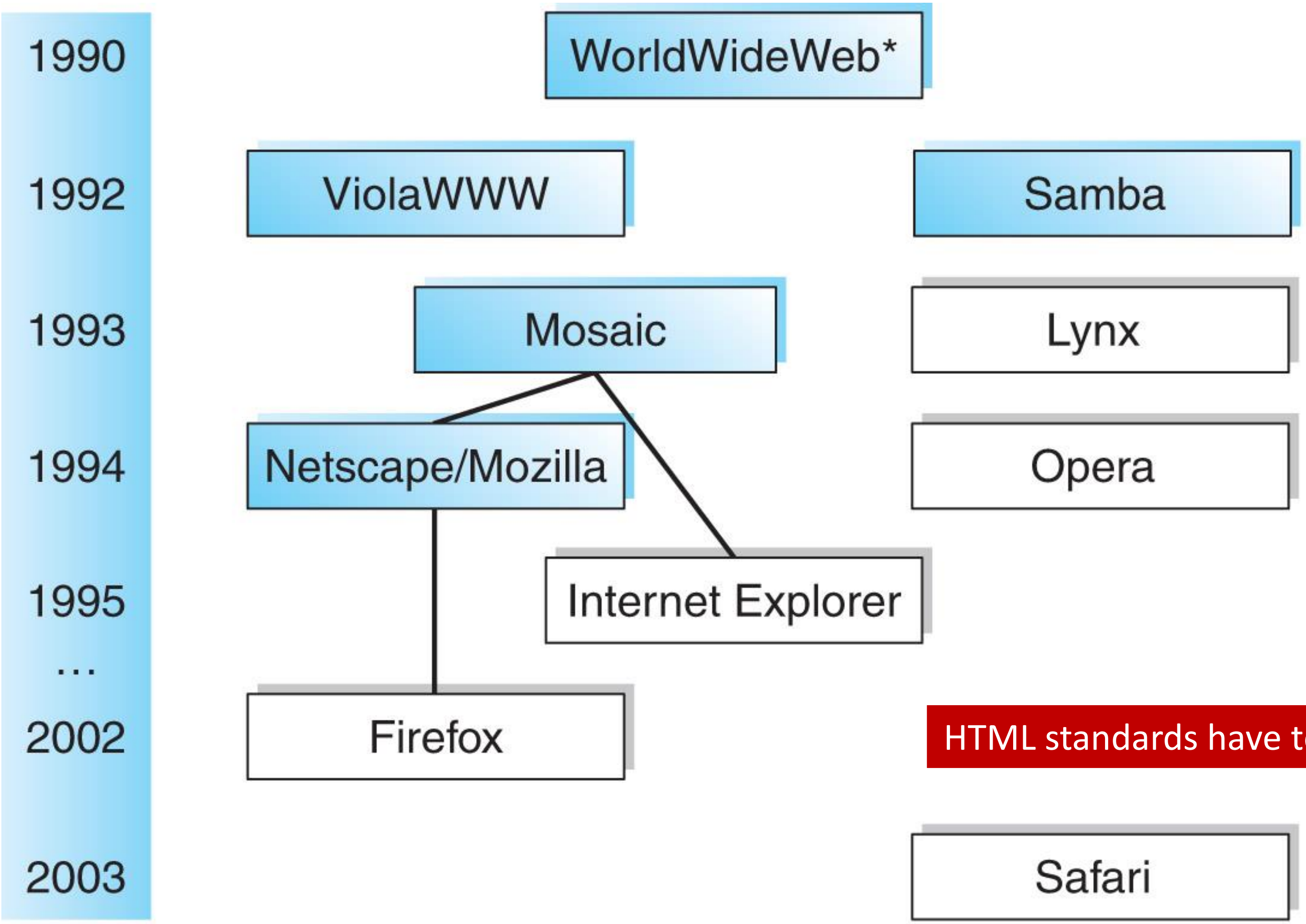
- NoScript is a Firefox extension that controls the execution of JavaScript
- NotScripts and ScriptSafe attempt to provide similar functionality for Chrome

These tools allow you to enable script execution on trusted sites, but disable it when visiting unknown or untrusted sites

Web Browsers

Web Browsers

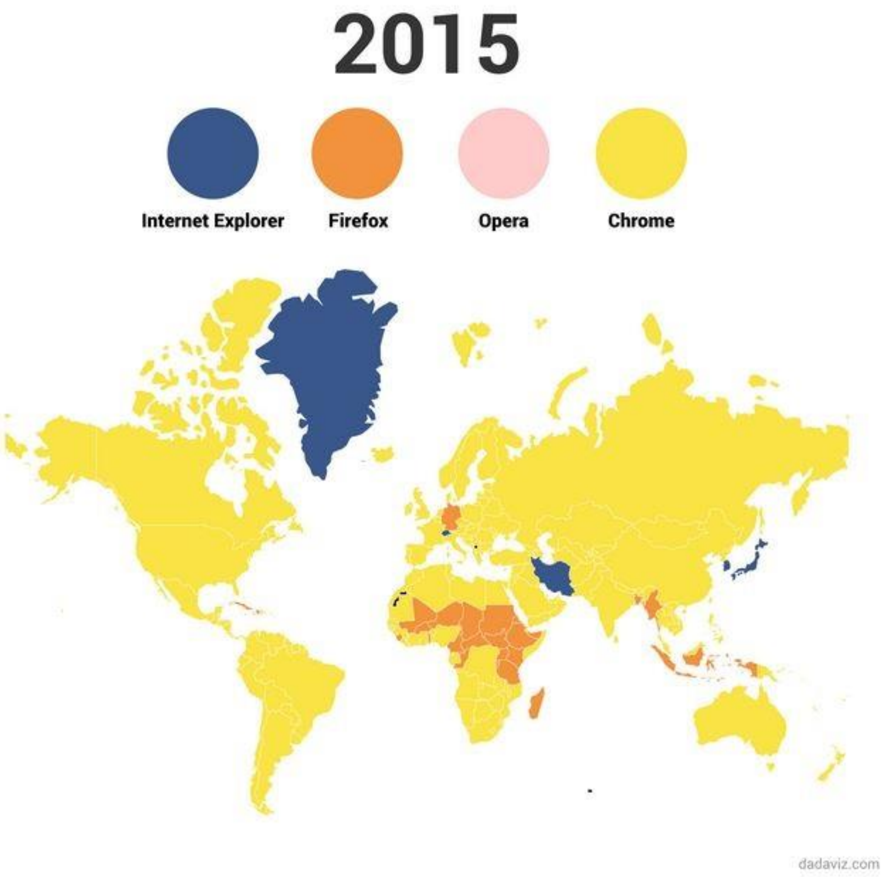
Web Browsers: Time-Line

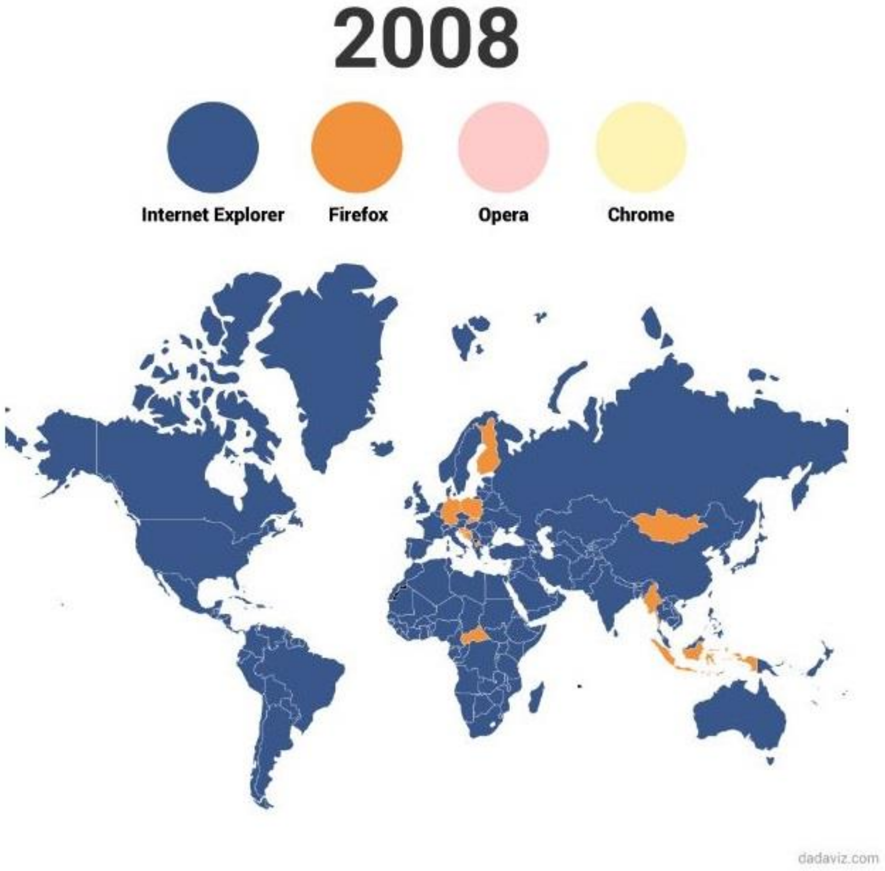


HTML standards have to be enforced !!!

*This was the name of the first web browser developed by Tim Bemers-Lee.

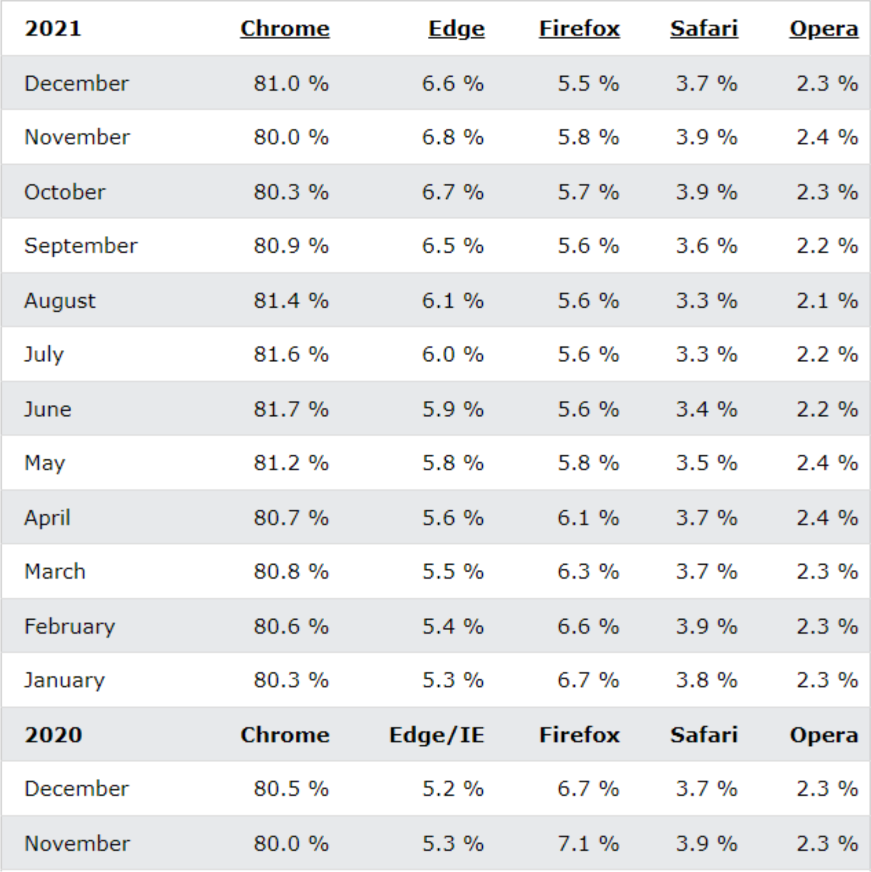
Web Browsers: World Use





http://fossbytes.com/this-map-tells-how-everybody-killed-internet-explorer-and-started-loving-chrome/

Web Browsers: Statistics (W3Schools)



2021	Chrome	Edge	Firefox	Safari	Opera
December	81.0 %	6.6 %	5.5 %	3.7 %	2.3 %
November	80.0 %	6.8 %	5.8 %	3.9 %	2.4 %
October	80.3 %	6.7 %	5.7 %	3.9 %	2.3 %
September	80.9 %	6.5 %	5.6 %	3.6 %	2.2 %
August	81.4 %	6.1 %	5.6 %	3.3 %	2.1 %
July	81.6 %	6.0 %	5.6 %	3.3 %	2.2 %
June	81.7 %	5.9 %	5.6 %	3.4 %	2.2 %
May	81.2 %	5.8 %	5.8 %	3.5 %	2.4 %
April	80.7 %	5.6 %	6.1 %	3.7 %	2.4 %
March	80.8 %	5.5 %	6.3 %	3.7 %	2.3 %
February	80.6 %	5.4 %	6.6 %	3.9 %	2.3 %
January	80.3 %	5.3 %	6.7 %	3.8 %	2.3 %
2020	**Chrome**	**Edge/IE**	**Firefox**	**Safari**	**Opera**
December	80.5 %	5.2 %	6.7 %	3.7 %	2.3 %
November	80.0 %	5.3 %	7.1 %	3.9 %	2.3 %

http://www.w3schools.com/browsers/browsers_stats.asp

Web Browsers

- Web Browsers handle data in different ways

- Ideally a web application looks and feels the same way across multiple browsers
  - Front end developers need to verify web designs in various browsers to ensure desired output
  - CSS reset scripts are commonly used

Web Browsers

Web Browsers use a browser engine

- Also called a Layout Engine or Rendering Engine

- Microsoft used MSHTML for Internet Explorer and EdgeHTML for the Edge browser
- This has since been replaced by the Blink Engine

- Google Chrome uses the Blink Engine
- Originally used the WebKit Engine

- Mozilla’s Firefox uses the Gecko Engine

- Safari browser uses Apple’s WebKit engine

Browser Attacks

- Attackers have since shifted their focus from attacking web servers to attacking clients/users
  - A server will typically be more secure than grandpa Donald’s outdated web browser
- A lot of Apps are now becoming more and more web based making the browser as the main interface tool
  - Makes it easier to sell SaS
  - Remote access to applications by users
  - Access to company data, etc.

Same Origin Policy

Same-Origin Policy

- Scripts running on a page should only be able to read from, or write to another page if both pages have the same origin
- The **Origin** has three components:
  - Application layer protocol
    - e.g. HTTP or HTTPS
  - TCP Port
  - Domain Name
- This standard is supported by all major browsers

Same-Origin Policy

Examples

- http://www.cnn.com
  - The base site where the scripting code is running
- http://cnn.com
  - Different site, gets redirected to www.cnn.com
  - Without redirect scripts might not work
- https://cnn.com
  - This one wouldn’t work because the port is different
- http://cnn.com:8080/
  - This one wouldn’t work because the port is different too

Same-Origin Policy

Examples from Mutillidae

- http://folusername/mutillidae
  - The base site where the scripting code is running
- http://www.folusername/mutillidae
  - Different site, doesn't work
- http://folusername:8080/
  - This one wouldn’t work because the port is different too

Same-Origin Policy

- It is important to keep in mind that this only applies client-side
- The server side code can access whatever it wants
- The whole idea is to keep personalized information within the relevant site / origin

Exceptions

- Developers can bypass the same-origin policy in certain controlled ways
  - This is allowed to provide more functionality for sites
- Whenever exceptions are made there is the possibility of abuse

Same-Origin Policy

The HTML **<script>** element can be used with the **src** attribute to load script code from another domain

- The user will be navigating to your page, but the code the browser is running is being loaded from somewhere else
- If there is malware at the other end it will look like it ran from your page

Google Hosted Library:

https://developers.google.com/speed/libraries/devguide

jQuery example:

```
<script src="//ajax.googleapis.com/ajax/libs/jquery/2.1.3/jquery.min.js"></script>
```

Browser Exploitation Framework (BeEF)

BeEF

- BeEF is a popular tool for exploiting web browsers and is natively found in Kali Linux

- It is browser based and can be launched by clicking on the 🐂 icon, or by running the executable file found in /usr/share/beef-xss

BeEF

- BeEF works by “Hooking” browsers

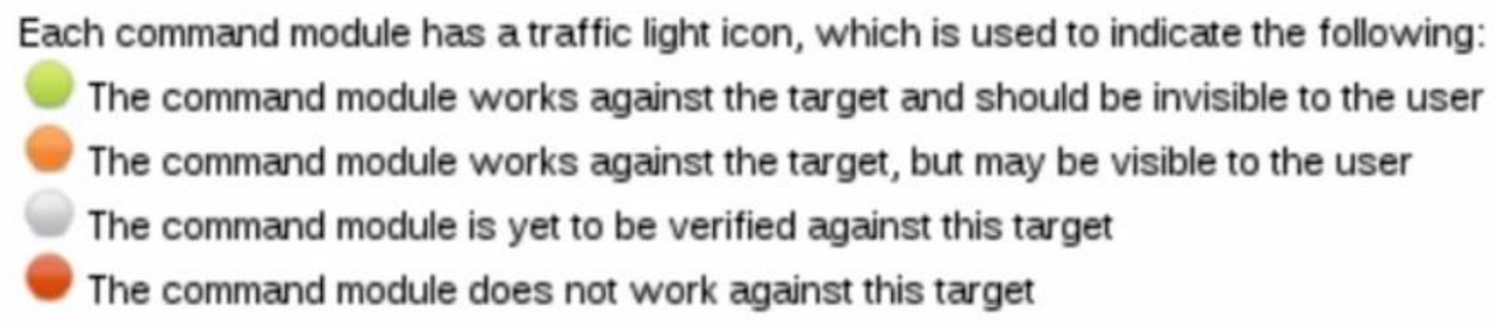
- This involves a client clicking on a link that contains a JavaScript file that will tie their browser back to the BeEF server
  - hook.js

- The Browser will remain “hooked” as long as the user has the window that hooked it open

BeEF

- Metasploit can be integrated into BeEF
  - Requires some configuration set up

- BeEF uses a “traffic light” system for payloads



Each command module has a traffic light icon, which is used to indicate the following:

🟢 The command module works against the target and should be invisible to the user

🟠 The command module works against the target, but may be visible to the user

⚪ The command module is yet to be verified against this target

🔴 The command module does not work against this target

BeEF

- Once a browser is hooked, the attacker can perform various actions against the compromised browser

  - BeEF lists these actions in “Command Modules”
  - Information gathering / Software detection
  - DOM manipulation
  - JavaScript Execution
  - Credential Excavation
  - Browser Redirection
  - Cookie Information
  - Etc.

Lab Details

LAB-03: Overview

Lab-03: XSS, CSRF, and BeEF

- Stored Cross Site Scripting **(XSS)** Page Redirection
- Cross Site Request Forgery **(CSRF)**
- **BeEF:** The Browser Exploitation Framework
- Create a script to hook a browser
- Inject JavaScript
- Explore different options for hooked browsers within BeEF