

# Security Idiots

[Home](#)[Categories](#)[Video Gallery](#)[The Idiots Team](#)[Contact Us](#)

## Tutorials Browser

### Web Pentest

#### Information-Gathering

Part-0-Purpose-of-Information-Gathering.html

Part-1-information-Gathering-with-websites.html

Part-2-information-Gathering-with-Google.html

Part-3-information-Gathering-with-nmap.html

Part-4-DNS-information-Gathering-with-DIG.html

Part-5-information-Gathering-with-Fierce.html

Part-6-information-Gathering-with-FOCA.html

Part-7-information-Gathering-with-MetagoFil.html

#### Cloudflare-Bypass

Part-1-Understanding-Cloudflare-Security.html

Part-2-Cloudflare-Security-Bypass.html

Part-3-Cloudflare-Security-Bypass.html

Part-4-Cloudflare-Security-Bypass.html

### LFI

guide-to-lfi.html

### SQL-Injection

Part-1-Basic-of-SQL-for-SQLi.html

Part-2-Basic-of-SQL-for-SQLi.html

Part-3-Basic-of-SQL-for-SQLi.html

Basic-Union-Based-SQL-Injection.html

basic-injection-single-line-or-death.html

## XSS Series by Securityidiots

### Table of content:

1. [Part one : Basics of Javascript for XSS](#)
2. Basics of XSS (Stealing Cookies)
3. Different Contexts for XSS execution
4. Different types of XSS Filters and their bypasses
5. Different Ways to Bypass XSS Filters by (leak/access)ing Objects
6. CSP Bypasses:
  - [\*] What is CSP
  - [\*] Whitelisting script-src for CDNs causing XSS CSP Bypass
  - [\*] Various other ways to CSP bypass
7. Browser Based Vulnerabilities causing (Old IE) XSS
  - [\*] Converting XSS from (Old IE) to newer version of IE
8. DOM Based XSS
  - [\*] various Sources/Sinks
9. Misc XSS Techniques(shorter payloads)
10. Stealing CSRF Token and Performing CSRF Actions using XSS
11. Exploiting Self XSSs via Login/Logout CSRF Chain.
12. AngularJs Template Injections to XSS
13. Manual VS Automated Scanning and Tools/methods for XSS testing

### BASICS of JAVASCRIPT Part 1 for XSS

In This Tutorial, We are going to discuss the basics of Javascript(js) for this upcoming series and exploit XSS, way better than just firing alerts.

### INTRODUCTION

Javascript is a client-side Programming/Scripting Language for the Web used within HT behavior/functionality of a web page ie. javascript could be used to Change the Style(CSS) & Model(DOM) of a Web Page and may or may not depend on the user interaction/input.

### What is DOM(Document Object Model)?

Simply put, When a Web Page is loaded a DOM Tree(Data Structure) is Created Which contains Nodes including Elements/Tag Names, attributes of the tags, id of the tag, name of the tag within another HTML element ie. DOM contains the structure of whole Web page.

A Simple HTML5 Page would be like This:

```
<!doctype html>
<html>
<body>
Some Text
<p name="para" id="1">Hi</p>
```

XPATH-Error-Based-Injection-  
Extractvalue.html

XPATH-Error-Based-Injection-  
UpdateXML.html

Error-Based-Injection-  
Subquery-Injection.html

sql-evil-twin-injection.html

Blind-SQL-Injection.html

bypass-login-using-sql-  
injection.html

dump-database-from-login-  
form-sql.html

url-spoofed-phishing-with-  
sql.html

ddos-website-with-sqli-  
siddos.html

delete-query-injection.html

update-query-injection.html

xss-injection-with-sqli-  
xssqli.html

time-based-blind-  
injection.html

insert-query-injection.html

group-by-and-order-by-sql-  
injection.html

Union-based-Oracle-  
Injection.html

Dump-in-One-Shot-part-1.html

Dump-in-One-Shot-part-2.html

DIOS-the-SQL-Injectors-  
Weapon-Upgraded.html

database-type-testing-sql-  
injection.html

routed\_sql\_injection.html

multi-query-injection.html

mssql-insert-query-  
injection.html

oracle-sql-injection-dios-  
query.html

mssql-out-of-band-  
exploitation.html

addslashes-bypass-sql-  
injection.html

MSSQL

mssql-dios.html

MSSQL-Union-Based-  
Injection.html

MSSQL-Error-Based-  
Injection.html

Tricks

Grab-IP-Address-Using-  
Image.html

WAF-Bypass

waf-bypass-guide-part-1.html

bypass-sucuri-webSite-  
firewall.html

XPATH-Injection

```
</body>
</html>
```

DOM would look like this:

```
#document
|_DOCTYPE: html
|_HTML
|  |_HEAD
|  |_BODY
|    |_#text: Some Text
|    |_P name="para" id="1"
|      |_#text: Hi
|    |_#text:
```

#### Different ways to execute JS:

- Inside <script>...</script>
- Loading External Js file with 'src' attribute
- Using Event handlers
- Browser console using F12

#### Javascript Comments:

- Single Line Comment : //This is a Comment
- Another Single Line comment <!--This is commented
- Multi Line Comment : /\*Multi line comment\*/
- Lesser Known Single Line Comment : --> This is also a Comment

#### DataTypes & Variables declaration in JS

The Syntax is bit Similar to C ie. each line ends with semicolons (they are not required \ written in different lines)etc.. except that we don't need to declare data types of variables for

```
var a=10,b=null,c="String",d=false,e={A:1,B:"String2",C:[1,2]},f=[1,2,3,"String3"],g; //Explained Be
```

We can also declare the same in different lines

```
var a=10;
var b=-20;
```

a is an integer data type

b is an Empty Object

c is a String

d is a boolean (true/false) data type

e is a javascript Object

Basic Syntax in javascript to declare objects, since javascript objects are declared within b Empty Object. An object is an array of multiple properties, value of a property could be any c example objects below

```
e={propertyname:"value of property"}
eg: {propertyname1:"value1",propertyname2:"value2"}
```

Try and define the below variables using any of the method mentioned above and then ch can simply run this in Browser Console(Press F12 on Chrome/FF), paste the below code a name to check each variable's value one by one.

```
var a=10,b=null,c="String",d=false,e={A:1,B:"String2",C:[1,2]},f=[1,2,3,"String3"],g;
```

An Example HTML Document for the above variables would be :

```
<!doctype html>
<html>
<script>
var a=10,b=null,c="String",d=false,e={A:1,B:"String2",C:[1,2]},f=[1,2,3,"String3"],g;
```

Basics-of-XPATH-for-XPATH-  
Injection-part-1.html

Basics-of-XPATH-for-XPATH-  
Injection-part-2.html

Basics-XPATH-injection.html  
xpath-injection-part-1.html

XSS

XXE

XXE-Cheat-Sheet-by-  
SecurityIdiots.html

```
</script>
</html>
```

Let us see how to access Javascript Objects, first of all using the below statement in javascript declare an object "obj" which have two properties prop1 and prop2, using the below statement

```
var obj={prop1:300,prop2:"String"};
```

The values of These Properties Could be accessed in 2 Ways

#### 1. Dot Notation

```
obj.prop1; // would return the value of property prop1
obj.prop2; // would return the value of property prop2
```

#### 2. Square Brackets [] Notation

```
obj["prop1"]; // would return the value of property prop1
obj["prop2"]; // would return the value of property prop2
```

#### Window and Document Object in javascript:

Window is the main JavaScript global object, accessed by "window" object it contains all the opened window (height, width, name of the window, DOM etc) and also its opener. Document is a property of "window" object only but since "window" object is global it can be accessed as "document" or even as "window.document".

So basically, DOM (Document Object Model) loads all the objects inside "WINDOW" and then the object gets loaded inside the window object. Some of the Important Properties of "window" objects are:

1. **Window.location** or location;, which contains all the information about the current loaded page. Hostname, port, protocol, path etc.

```
window.location.host : "www.host.com"
window.location.hostname : "www.hostname.com"
window.location.href : "https://www.hostname.com/path/to/file.php"
window.location.origin : "https://www.hostname.com"
window.location.pathname : "/path/to/file.php"
window.location.port : ""
window.location.protocol : "https:"
```

Changing the properties of location Object would redirect the page. location.href="https://google.com"; would redirect to <https://google.com>

2. **Window.opener**; returns the reference to the parent window which opened the current window. window.open() method for example:

```
window.open( URL, name); //Opens a Window of name and URL , if url is empty string it returns the current window
mywin=window.open("", "MyWindow"); //opens an about:blank window with name "MyWindow"
mywin.document.write("This is Child Window");

mywin.opener.document.write("The Parent Window");//We Could Change content of the Parent Window
```

Author : Rahul Maini

Date : 2017-04-22