Tutorials Browser

Basics of Javascript for XSS part 2

Welcome to the Part2 of Basics of JavaScript for XSS.

In our last Tutorial, Part1 of Basics of JavaScript for XSS we learnt about:

- DOM
- JavaScript Basic Syntax
- Declaring variables
- Different datatypes (number,String,boolean,Object,Array)
- Different Notations to access Object's properties value

Now we will continue our discussion on some other Important Properties of "window" and "document" object.

window.location.hash or location.hash - This property returns the part of URI Fragment ie. Whatever is written in the URL after "#" (hash symbol) For example a URL https://www.securityidiots.com/#blablalabla in that case "location.hash;" would return "#blablalabla" and location.hash.slice(1); would return blablalabla ie. It will slice 1st character from the string, which is #.

This part after hash in the URL is for client side usage and hence it couldn't be accessed by Server Side Languages which is why is very helpful in bypassing Server Side WAF.

window.location.search or location.search - This property returns the query string or the "GET" parameter. For example:

```
https://www.securityidiots.com/?xyz=1&abc=zen
"location.search;" would return "?xyz=1&abc=zen"
```

document.domain - This property is used to return the hostname of where javascript is executing. Similar to location.hostname. This is mainly used to confirm that XSS is executing on the right domain.

document.cookie - This property is used to get all Cookies as a String, but if there is a "HttpOnly" flag set in the cookies, then cookies couldn't be accessed via JavaScript.

document.getElementsById('123') - This is a Method (a Function) which is used to get all elements (nodes) having 123 value of "id" attribute provided in the argument. We could then even modify/remove the selected nodes for Example:

```
<!doctype html>
<html>
<p id="someID">Select Me</p>
<script>
var selected=document.getElementById("someID");/*would store the <p id="someID">Select Me</p> node i
</script>
</html>
```

There are many other similar methods too to manipulate the DOM like:
document.getElementsByName('Name');
document.getElementsByTagName('TagName');
document.getElementsByClassName('ClassName');

However there could be Multiple Tags with the same value of Tag Names, name & class attributes, in that case, an array of nodes is returned and could be accessed via array indices like selected[0], selected[1]...etc.

document.innerHTML - This method is used to write HTML content within a selected node. For example:

```
<!doctype html>
<html>
<p id="someID">Select Me</p>
<script>
var selected=document.getElementById("someID");/*would store the <p id="someID">Select Me</p> node i
selected.innerHTML="<p>Text blabla</p>";/*you would see the content inside <p id="someID"> has now cha
</script>
</html>
```

Any User Input to This is Very Dangerous Even if its filtering > , < , " , ' only by using backslash(\) ,numbers(0-9),alphabets Malicious User could Execute Javascript.(Explained Later)

document.write/document.writeln("HTML Content") - This is used to rewrite(remove existing HTML) the Whole structure of web page with its supplied Argument(HTML content). For Example:

```
<!doctype html>
<html>
<script>
document.write("Contents of The Page");
</script>
</html>
```

Any User Input to this is Very Dangerous Even if it's filtering >, <, " , ' only by using backslash(\), numbers(0-9), alphabets Malicious User could Execute JavaScript which will be explained Later in out tutorial

document.createElement('Element Name') – createElement Method is used to create a new HTML element with

javascript. For example if we want to create an img element with src="http://securityidiots.com/post_images/backxss.png"

```
var imgtag = document.createElement("img");//creates img element
imgtag.src="http://securityidiots.com/post_images/backxss.png";//adds attribute src to img element
document.body.appendChild(imgtag);//appends the created element to the body tag of the DOM
```

Note: There may be more properties that we would cover as we move further but for now these properties should be enough to move ahead.

JAVASCRIPT FUNCTIONS

Like other languages, there are functions in JavaScript too which used to execute a set of statements/instructions to reduce repetition of our code.

Defining Javascript Functions:

A JavaScript function is defined with "function" keyword, followed by the function's name, followed by parenthesis () and inside it may include arguments/parameter names separated by commas, Functions in JavaScript are called, similar to other languages:
functionname([arg1,arg2]); // [] indicates that arguments/parameters are optional

Ways of Defining and Calling Functions:

```
//Defining
var x=function myFunction(a, b) {
   return a * b;
};

//Calling
z=x(30,40);// would return 1200 in variable z
```

we could also call a function directly without storing it in a variable

```
//Defining
function myFunction(a, b) {
    return a * b;
};

//Calling
z=myFunction(30,40);
```

We could also define functions without names called as "Closures" or anonymous functions like:

```
//Defining
var x=function(arg){
  return arg*100;
};

//Calling
x(10);//would return 1000
```

And also, we can do define and call, both at the same time:

```
var x=function(arg){
  return arg*100;
}(10);      // function would be called & would return 1000 in x variable
```

Concatenation in Javascript:
Since Javascript is a loosely typed language, We can Concatenate Strings with Strings as well as with Other Data Type like number (integer/floats)/boolean with "+" Operator

For example :

```
var x="aaaaaa"+"bbbb"; //console.log(x) would return aaaaaabbbb
var x="aaa"+1234;//console.log(x) would return aaa1234
```

IMPORTANT JS FUNCTIONS RELATED TO XSS

1. console.log: ("logs this to browser's developer console "); - This writes the output to the browser console which is used for debugging purposes ie. We could get/set/change/override the values of variables/functions etc. in the console itself. A very nice example would be:

```
<!doctype html>
<html>
<script>
console.log("Test!!");
</script>
</html>
```

Now open the Browser's Developer Console (by pressing F12/Ctrl+Shift+J depends on browser) and reload the page, you would see "Test!!" logged in the console.

Getting the values of variables in console:

```
<!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;
console.log("Value of a:");
console.log(a);
console.log("Value of b:");
console.log(b);
console.log("Value of c:");
console.log(c);
console.log("Value of d:");
console.log(d);
console.log("Value of e:");
console.log(e);
console.log("Value of f:");
```

```
console.log(f);
```

Open Browser's Developer Console and reload The above Page you would see the values of a,b,c,d,e,f,g variables in Console.

2. alert(1);prompt(1);confirm(1) -

alert"Argument Here"): (The very infamous :P alert() is used to popup an alert message box with the Argument String written in it, it doesn't return anything

prompt("Argument Here"): (is used to popup a prompt to take user input with a message supplied in Argument String, it returns the value entered by the user

```
var x=prompt("Enter Value: ");// after User enters value it will be stored in x for further operations
```

confirm("Message"): is used to popup a confirm box to confirm(OK/Cancel) Something, Pressing OK returns true and Cancel returns false.

```
var conf=confirm("Are You Sure To blablabla?");
```

From XSS point of view, these are the functions which we use to confirm/identify that XSS/JavaScript is Executing on the target

3. setTimeout() - The setTimeout() method calls a function after a specified number of milliseconds.

```
setTimeout(function(){ alert("Hello"); }, 3000); // will call anonymous function after 3 seconds
```

4. setInterval() - The setInterval() method calls a function after specified intervals endlessly

```
setInterval(alert(1),3000); // will call alert(1) after every 3 seconds
```

Therefore these setTimeout and setInterval functions could be used to execute JavaScript (XSS) when user input is in the first parameter.

6.var z=new Function('arg1','arg2','body') - creates a new Function object similar to var z=function(arg1,arg2){body;}. The User input to the last parameter is dangerous and could be used to XSS if that function is called anywhere in the page.

```
var z=new Function('arg1','arg2','alert(1)')
z(1,2);//would cause alert(1)
```

7.eval("JavaScript here") - used to evaluate string in its argument as JavaScript.

```
eg.eval('var x=1;alert(x)');
```

User input to this eval() is extremely dangerous.

That's enough for basics about Functions, for now, we would keep covering more about them if we require them in further topics. So in our next part of JavaScript Basics, we would cover **EventHandlers** and **AJAX (XMLHttpRequest)**.

Author : Rahul Maini
Date : 2017-04-29