

Make Your Own Custom OSINT Bookmarklets

Bookmarklets are small snippets of Javascript code that can be placed in the “Location” section of a traditional bookmark. When utilized, rather than navigating to a favorite website, these bits of code will execute tasks within the browser. Bookmarklets can be used in lots of useful ways, to manipulate source code, pull hidden information from a page, or automate multiple queries all at once.

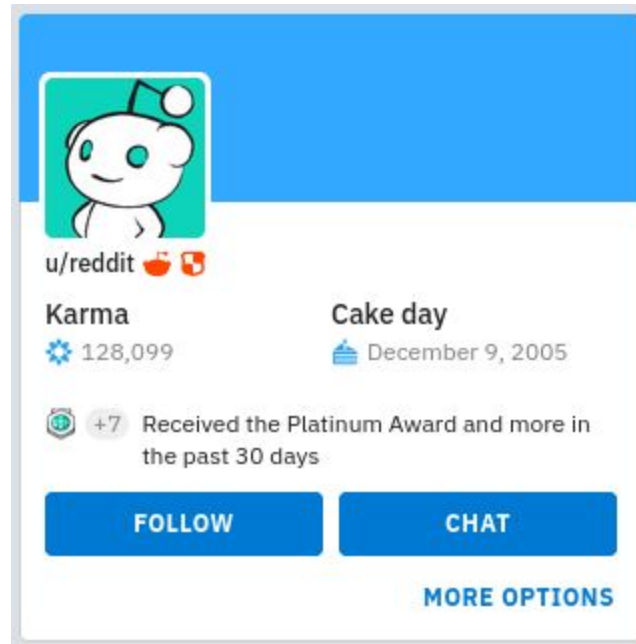
While there are numerous OSINT tools out there in the form of browser extensions or add-ons, there are many reasons I prefer to employ bookmarklets instead. Primarily, such bookmarklets are quick to code and debug as they usually only contain a few lines of code. Unlike more complicated extensions and add-ons, which may only be available for a specific browser, bookmarklets tend to be compatible across multiple browsers (with some exceptions). Additionally, while browser extensions tend to be blocked by production systems I’ve used I have successfully been able to run most bookmarklets on these same machines with little issue.

Part I: Extracting Information from Source Code

I often find myself creating bookmarklets to extract “hidden” information from a social media site or webpage. In most examples the sought after information is available to view within the source code but is often not visible on the display page and while finding this information is trivial for the average programmer or cyber investigator, it might present a challenge to someone new to this type of OSINT.

Additionally, manually inspecting this information on numerous accounts would quickly become tiresome when having to inspect the source and navigate to the code for every piece of target information. Instead, we can create a bookmarklet that will take all of the source code and parse it for us to quickly display only the specific targeted information we need. We can also then share this snippet of code with others so they too can quickly put it to good use in their own OSINT investigations.

Case Example: Reddit Full Profile Photo



We will be using the [u/reddit](#) user in order to walk through how to extract targeted information from a page's source code. When building this type of bookmarklet, the first thing we need to do is identify what information it is we want to extract. For example, on a Reddit user's profile, investigators are only shown a small, compressed, photo of the user's profile photo by default. While this might be enough for some photos, chances are you will want to check out the largest size photo possible in order to inspect some of the smaller details.

Once we've established what information we want to try and extract, in our case the full-size profile photo, we need to verify that it exists somewhere in the page source. This can be done by right-clicking on the webpage and clicking "View Source", "View Page Source" or other similar option depending on your browser of choice.

Searching through many lines of code to find what you're looking for is an art in and of itself. Try keywords that might be used to describe what you are looking for. I used "Profile", "photo", and eventually "avatar" to find the below lines of code which appears to show the URL to the full-size profile photo.

```
<meta property="og:image" content="https://www.redditstatic.com/avatars/avatar_default_05_0DD3BB.png"/>
"https://www.redditstatic.com/desktop2x/img/favicon/apple-icon-57x57.png"/><link rel="apple-touch-icon"
: //www.redditstatic.com/desktop2x/img/favicon/apple-icon-72x72.png"/><link rel="apple-touch-icon" sizes
ps://www.redditstatic.com/desktop2x/img/favicon/apple-icon-114x114.png"/><link rel="apple-touch-icon" s
tps://www.redditstatic.com/desktop2x/img/favicon/apple-icon-144x144.png"/><link rel="apple-touch-icon"
```

Once you find what information you want to extract, you will need to find the text that immediately appears before and after the target text and write it down. Using this information we will take the below template to craft our bookmarklet.

javascript:

```
var html = document.documentElement.innerHTML;
```

```
var subhtml = html.split('$leftlimittext')[1];
```

```
var output = subhtml.split('$rightlimittext')[0];
```

```
alert(output)
```

javascript:

This line should always be first in every bookmarklet. It will tell the browser to execute the Javascript code that follows and will not work if it is missing. You will not change this line.

```
var html = document.documentElement.innerHTML;
```

This line is responsible for obtaining the page source of the current window and then assigning that source code to a variable called “html”. You will not change this line.

```
var subhtml = html.split('$leftlimittext')[1];
```

This line is responsible for taking our source code (saved in the html variable) and parsing it for us using the text that appears immediately before our target code as a marker. After splitting the source code stored in the html variable it saves everything AFTER, but not including, our left limit text to a new variable called “subhtml”. The 1 in brackets is what is used to determine that we want everything after the left limit text.

```
var output = subhtml.split('$rightlimittext')[0];
```

This line is responsible for taking our remaining source code, which now only contains everything after our left limit text, and parsing it for us once again using our right limit text. This time it splits our remaining source code up to, but not including, our right limit text so that all that remains is our target text. It then saves the remaining target text into the variable “output” The 0 in brackets is what is used to determine that we want everything before our right limit text.

```
alert(output)
```

This line creates an alert pop-up and displays the target text which we have saved in the output variable.

Now that we understand what each line does, let's create our new bookmarklet using the template. Using our above code I have marked what our left limit text, targeted text, and right limit texts are.

Left Limit Text Targeted Text Right Limit Text

```
<meta property="og:image" content="https://www.redditstatic.com/avatars/avatar_default_05_0D038B.png"/>
"https://www.redditstatic.com/desktop2x/img/favicon/apple-icon-57x57.png"/><link rel="apple-touch-icon"
://www.redditstatic.com/desktop2x/img/favicon/apple-icon-72x72.png"/><link rel="apple-touch-icon" sizes
ps://www.redditstatic.com/desktop2x/img/favicon/apple-icon-114x114.png"/><link rel="apple-touch-icon" s
tps://www.redditstatic.com/desktop2x/img/favicon/apple-icon-144x144.png"/><link rel="apple-touch-icon"
```

Next, we will plug the left and right limit text information into our template to get the following code (changes in bold):

javascript:

```
var html = document.documentElement.innerHTML;
```

```
var subhtml = html.split('og:image" content="')[1];
```

```
var output = subhtml.split("")[0];
```

```
alert(output);
```

Now we take the above code and create a new bookmark in our browser.

Properties for "Reddit Full-size Profile Photo"

Name:
Reddit Full-size Profile Photo

Location:
javascript: var html = document.documentElement.innerHTM

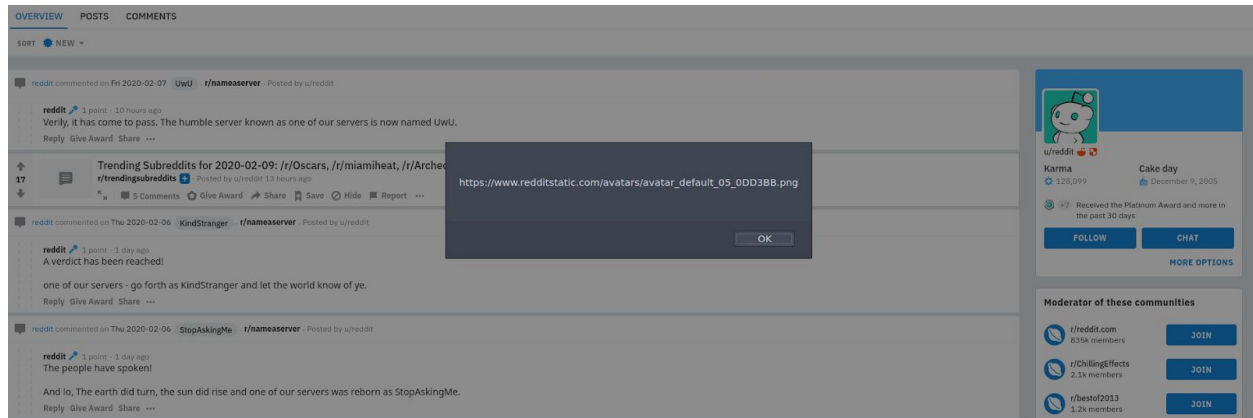
Tags:
Separate tags with commas

Keyword:

Cancel Save

Name will be what you decide to call the bookmarklet tool. This is the text that will appear in your bookmark bar. Location is usually where your URL to your favorite pages go, instead we are going to paste the entirety of our above code into this box. Click save and the new

bookmarklet tool is ready to go! Now, once we navigate back to the /u/reddit profile and click our newly created bookmark the following alert will appear.



Now if we were only extracting text then we could end this bookmarklet tool here. However with an image URL we will want to view the image itself rather than just the URL to it. (Although we could simply copy and paste the URL, we want to automate as many steps as possible.) So let's modify our bookmarklet to automatically open the image URL in the same tab rather than display the URL in an alert. We can do so by replacing the final line of code (**alert(output);**) with the following:

```
window.open(output,"_self");
```

This line takes the full-size profile photo URL saved in the output variable and opens it in the same browser tab you are currently on. Now when we activate our bookmarklet tool we no longer get the alert, instead we are shown the below full-size photo in the same tab.



Part I Conclusion

The above code example can be modified to pull numerous other types of information from source code, such as the direct video download link, Youtube video thumbnails, or uncropped photos or metadata that are not shown on the webpage by default. Now that you have some of the basics down, you should have little issue in building your very first bookmarklet tools for OSINT gathering. For ideas, be sure to take a look at some of the sample bookmarklet tools on my [Github](#), which also contains the [template](#) bookmarklet code we used in this article.

Keep an eye out for part II of this guide which will walk you through the creation of another common bookmarklet tool: running multiple queries. As always, should you have any questions please feel free to reach out to me on Twitter.

Part II: Running Multiple Search Queries

Another type of bookmarklet I frequently employ is one that uses a common variable and parses it out to multiple websites for analysis. Such an [example](#) would be running the same image URL through Yandex, Bing, and Google reverse image searches all using one bookmarklet tool. The biggest benefit of creating such a bookmarklet is reducing the time of checking multiple platforms as well as mitigating the chances of forgetting to exhaust all resources (by forgetting to run the information on one or more of the platforms). Running similar information across multiple tools is useful when each platform provides differing results as well as in instances where one or more platforms may go down.

Case Example: Multiple URL Expander Queries



Around two months ago a Twitter user created a series of [tweets](#) that showcased multiple suspicious Twitter accounts. One of the commonalities across these accounts was the frequent use of shortened bit.ly links. The goal for me was to identify what lied behind these links without risking compromise.

There is no shortage of URL expanders on the internet and many of them will give you varying bits of information that others might not include in their analysis. For this reason, you might want to run it across at least a few different platforms to ensure you have the fullest picture of what lies behind the URL. Using the second link in the screenshot (bit.ly/2T2ZdSj) I ran the shortened URL through three different URL expander platforms. Each platform provided some different information, and should one go down or fail to function we have other options as a backup. I received the following results below:

GetLinkInfo.com








GetLinkInfo.com

<http://bit.ly/2T2ZdSj>

Get Link Info

Enter any URL, for example: <http://tinyurl.com/2unsh>, <http://bit.ly/1dNVPaw>

Link Information

 Title	(none)
 Description	(none)
 URL	http://bit.ly/2T2ZdSj more info
 Effective URL	http://bit.ly/2T2ZdSj more info
 Redirections	(none)
 Errors	Website does not exist (Could not resolve host).
 Safe Browsing	Safe browsing data is temporarily unavailable

The first website did not manage to provide any information.

CheckShortUrl.com

Get long URL from hundreds of URL shortening services

Ensure your safety and prevent unsuitable content while surfing on the World Wide Web

<http://bit.ly/2T2ZdSj>

Expand




Long URL	http://abov.site/fbi-warns-nation-state-cyber-attacks-are-continuing/hmnk7LxdUHMTS_55zAYC4N9w4cJ3nSx8gPv1PP2hsUs
Delay	0.39 second(s)
Short URL	http://bit.ly/2T2ZdSj
Redirection	301
Search long URL on	Yahoo Google Bing Twitter
Check if safe on	Web Of Trust SiteAdvisor Google Sucuri Norton Browser Defender
Title	N/A
Description	N/A
Keywords	N/A
Author	N/A

The second website provides a snapshot of the website as well as options to check the URL on other platforms and provides the final website the URL redirects to.

ExpandUrl.net

RESULTS FOR HTTP://BIT.LY/2T2ZDSJ



Short URL: <http://bit.ly/2T2ZdSj>

Redirects: 1 ([show details](#))

Long URL: http://abov.site/fbi-warns-nation-state-cyber-attacks-are-continuing/hmnk7LxdUHMTS_55zAYC4N9w4cJ3nSx8gPv1PP2hsUs

Extra Information

EXTRA INFORMATION	
Meta Keywords:	No Keywords
Content-Type:	text/html; charset=utf-8
Google Safe Browsing:	 - This link appears to be safe! Advisory provided by Google.

The final website provides the destination URL as well as Google Safe Browsing information.

Now that we've chosen the three websites we want to run the information (in this case, a bit.ly URL) against, we need to gather all of our information in order to create our bookmarklet. First, we must determine what the URL structure is for querying the website. Thankfully, one of the websites give us this very information at the bottom of the webpage.

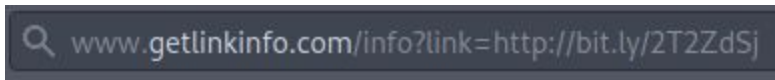
Share the link to this page [Tweet](#)

<http://checkshorturl.com/expand.php?u=http://bit.ly/2T2ZdSj>

The above photo shows us that the CheckShortUrl.com website appends the shortened URL to the end of the website like so: <http://checkshorturl.com/expand.php?u=#SHORTENEDURL>. Note down this URL for later.

Just as well, the website GetLinkInfo.com updates the URL in the address bar when we complete a search. Just as before we can see it appends the shortened link to the end of the

URL. We will also note this URL structure (www.getlinkinfo.com/info?link=#SHORTENEDURL) for later.

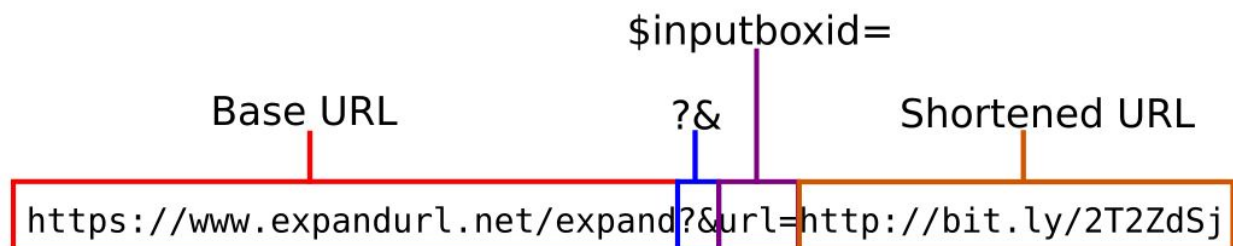


Unfortunately, the last site (ExpandUrl.net) does not give us a hint, nor does the URL in the address bar update to show us the URL structure of the query. However, this does not mean that we cannot use it. We just need to take a few extra steps.

Looking at the final website, we will right click the text box where we would normally enter the shortened URL and click Inspect (or Inspect Element depending on your browser). This will open up the page's source code and highlight the input box we selected. We are wanting to take note of the id (url) here.

```
<div class="col-md-8 col-md-offset-2">
  <div class="contact-form" style="text-align:center">
    <form class="default-form" action="https://www.expandurl.net/expand" method="post">
      <div class="form-field">
        <input id="url" type="text" name="url" value="test" data-kpzc-id="url">
      </div>
      <button id="submit" class="c-button m-color-2" type="submit">Expand URL</button>
    </form>
  </div>
```

Once we have the id, we will attempt to manually build the URL by modifying what we currently see in the URL bar. (<https://www.expandurl.net/expand>) In order to construct this, we will need to know the base URL, the id of the input box, as well as the shortened URL we want to search. We will need to construct it similar to our previous two URLs by using the following URL formula:



What the above URL does is automatically insert the shortened URL we want to expand into the input box (with the id of url) when the page loads. This will not automatically load the results page like our first two websites, however all that will be left to do is click on the "Expand URL" button once the page loads. In instances where a website does not let you fully automate the process this is the second best option as it automates everything but that last click. This is the

last of our three websites and we will note down the newly created URL structure for later (<https://www.expandurl.net/expand?url=#SHORTENEDURL>)

Now that we have all of our URL structures listed, we can use that information to plug into a template to create our new bookmarklet tool.

javascript:

```
var input = prompt("Prompt user for Input");
```

```
var variable1 = "$URLSTRUCTUREONE" + input;
```

```
var variable2 = "$URLSTRUCTURETWO" + input;
```

```
var variable3 = "$URLSTRUCTURETHREE" + input;
```

```
SiteOneFunction();
```

```
SiteTwoFunction();
```

```
SiteThreeFunction();
```

```
function SiteOneFunction() { setTimeout(function(){ window.open(variable1, "_blank"); }, 1000); }
```

```
function SiteTwoFunction() { setTimeout(function(){ window.open(variable2, "_blank"); }, 1000); }
```

```
function SiteThreeFunction() { setTimeout(function(){ window.open(variable3, "_blank"); }, 1000); }
```

I know that there are many more lines than the last bookmarklet tool that we created, however do not be intimidated! It is quite simple and the extra lines are due to it performing the same task three times rather than one.

javascript:

This line should always be first in every bookmarklet. It will tell the browser to execute the Javascript code that follows and will not work if it is missing. You will not change this line.

```
var input = prompt("Prompt user for Input");
```

This line will prompt the user for input via a popup. You will change the text within the quotations to whatever instructions you want to provide the user along with the prompt for input.

```
var variable1 = "$URLSTRUCTUREONE" + input;
```

This line will take the input received from the user and append it to the end of a URL structure for the first website and assign it to a variable called variable1. You will change **variable1** to whatever you want to name the variable, perhaps the website name. You will also change **\$URLSTRUCTUREONE** to the URL structure of the first website which appears before the user input (such as <http://www.getlinkinfo.com/info?link=>) You will repeat this line of code as many times as needed to cover all of the websites you will be querying. (In our case three total.)

SiteOneFunction();

This line will call on the corresponding function at the end of our code. You will change **SiteOneFunction** to whatever you want that function to be called. There will be as many lines like this one as you had in the above section (So three total.)

function SiteOneFunction() { setTimeout(function(){ window.open(variable1, "_blank"); }, 1000); }

This is where the magic happens. This line will open a tab using the full URL crafted from variable1. In this line you will change the **SiteOneFunction** to the function name you used in the previous line, make sure they match perfectly or they will not run. You will also change the **variable1** name to whatever you called the variable in the second line of code above. Finally, we wrapped it in a Timeout to prevent our browser from blocking the numerous popups that will otherwise occur rapidly. (Even so, you might also need to allow popups when you run the bookmarklet.) You will repeat this line three times like the others to use all three URL shorteners we selected.

Next, we will plug in our information into the above template to create the following code. I have marked the changes in the template in bold:

javascript:

var input = prompt("**Paste in the shortened URL to be expanded:** ");

var **getlinkinfo** = "http://www.getlinkinfo.com/info?link=" + input;

var **checkshorturl** = "https://checkshorturl.com/expand.php?u=" + input;

var **expandurl** = "https://www.expandurl.net/expand?&url=" + input;

GetLinkInfoFunction();

CheckShortUrlFunction();

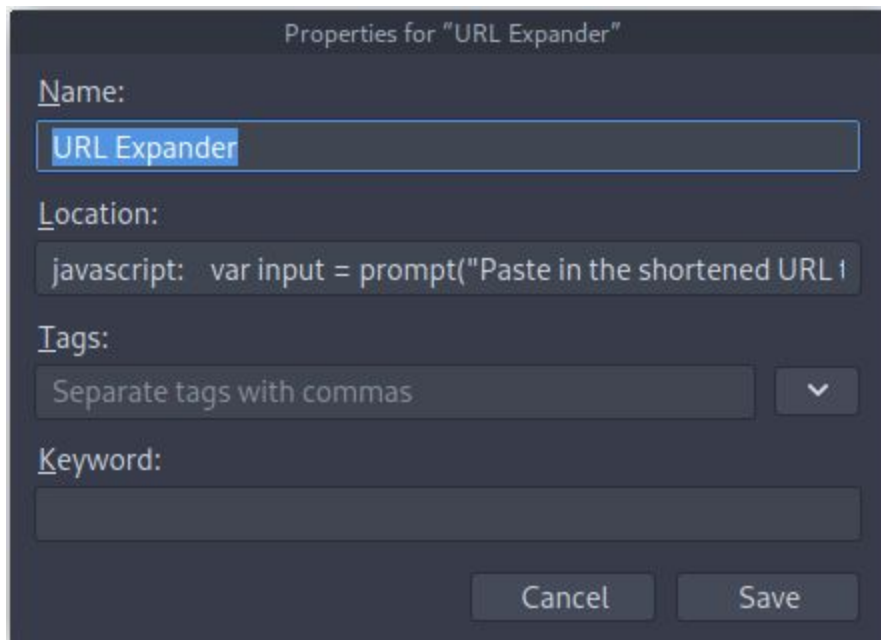
ExpandUrlFunction();

```
function GetLinkInfoFunction() { setTimeout(function(){ window.open(getlinkinfo, "_blank"); }, 1000); }
```

```
function CheckShortUrlFunction() { setTimeout(function(){ window.open(checkshorturl, "_blank"); }, 1000); }
```

```
function ExpandUrlFunction() { setTimeout(function(){ window.open(expandurl, "_blank"); }, 1000); }
```

Now we can take the above code and create a new bookmark in our browser just like we did in Part I. For these fields, Name will be what you call the bookmarklet tool and will appear in the bookmark bar of your browser. In the section titled Location we will paste all of our code created above. Click save and the new bookmarklet tool is ready to go!



Part II Conclusion

Between Parts I and II you should be well on your way to understanding how to craft basic OSINT bookmarklets to either find information hidden in the source code, submit multiple URL-crafted queries, or perhaps even a bookmarklet that does both! (Such as pulling the full-size profile photo URL from a website's source code and then running multiple reverse image searches using that URL.) If you are looking for inspiration you can find a number of bookmarklets on my [Github](#) page, including the one created during this article as well as some templates to create your own. Should you have any questions or run into some problems with a particular process found in this article don't hesitate to shoot me a message over on Twitter.