

JavaScript Restrictor

Webextension that improves your privacy and security

Developer documentation

Libor Polčák



TARZAN project VI20172020062 document
Faculty of Information Technology, Brno University of Technology

Last change: June 12, 2020

How to write a new wrapper?

The primary focus of JSR is to provide security and privacy oriented wrappers of JavaScript APIs. As some of the code is very similar for each wrapper, we use a unified approach to describe wrappers. The approach is described below. This approach also helps to automatically modify `toString` conversions of the wrapped APIs, i.e. a correctly written wrapper creates a modified function but `wrapper.toString()` returns the original string. Finally, this approach also helps in generating code dealing with the [Firefox bug described in #25](#).

File structure

- `wrapping.js` provides main facilities for interacting with specific wrappers:
 - `add_wrappers()` has to be provided with all the wrappers. Hence, a typical module with wrappers of a web standard APIS. ends with a call of `add_wrappers(list_of_all_wrappers_defined_by_the_module)` .
 - `build_wrapping_code` contains all the registered wrappers. The keys are referenced by the levels wrapper list. Do not modify the `build_wrapping_code` variable directly, use `add_wrappers` instead.
 - The module also provides common functions used by the wrappers, e.g. `rounding_function` and `noise_function` .
- `wrappingS-XYZ` are files dealing with APIs introduced by specific web or ECMA standard. See the **naming conventions** for details on the XYZ part of the name. See the **wrapper specification** section for the properties of the wrapper objects.

Naming conventions

JSR adopted the naming conventions of the [Web API Manager](#). See the paper:

Peter Snyder, Cynthia Taylor, and Chris Kanich, “[Most Websites Don’t Need to Vibrate: A Cost–Benefit Approach to Improving Browser Security](#),” in Proceedings of the [2017 ACM Conference on Computer and Communications Security](#), 2017.

Wrapper specification

Once you are set with the file name for your wrappers, you can start coding. The basic structure of the file is:

```

(function() {
    // definition of common functions used by the wrappers bellow
    var wrappers = [
        {
            // wrapping object 1
        },
        {
            // wrapping object 2
        },
        ...
        {
            // wrapping object n
        },
    ]
    add_wrappers(wrappers);
})();

```

The content of the module should be in an IIFE so that it does not pollute the global namespace. The property values are strings that are generally interpreted either as identifiers or JS code.

Each wrapping object must have the following mandatory properties:

- `parent_object` and `parent_object_property` are used to define the name of the wrapping (`parent_object.parent_object_property`) that is referenced by level wrappers. Additionally, it is used if `wrapper_prototype` is defined to provide the object name to have the prototype changed. Finally, `object.freeze` is called on `parent_object.parent_object_property` .
- `wrapped_objects` is a list of objects, each having two properties:
 - `original_name` - the original name of the object to be wrapped. Do not mention `window` here!
 - `wrapped_name` - the variable name that can be used by `wrapping_function_body` , `helping_code` and other code fragments to reference the original object. Note that this name is not available outside the code generated by this wrapper.

Each wrapping object can have the following optional properties:

- `wrapping_function_args` is a string that is used as an argument list for the wrapping function. Typically this string reflects the parameters of the original method, it can be an empty string, a list of parameters, such as `"source, target, color"` or `"...args"` .

- `wrapping_function_body` specified the behaviour of the wrapped function. You can provide a completely different implementation but often, you want to refer to the original implementation (available in the variable `wrapped_name`) and modify the original implementation.
- `wrapping_code_function_name` can be used to call the wrapping function from the other code inside the wrapper. For example to create another wrapper similar to the original wrapper.
- `wrapper_prototype` - if defined, `parent_object.parent_object_property` prototype is set to the prototype identifier provided by `wrapper_prototype` .
- `original_function` if not provided, it defaults to `parent_object.parent_object_property` . This name is used for overloading the `toString` function. Instead of the wrapping code, `toString` returns the content of the original function.
- `helping_code` provides you an option to define code available for both `replacement` function and `post_wrapping_code`
- `replace_original_function` is used to control which function should be replaced
- `post_replacement_code` Allows to provide additional code with the access to the original function and to the wrapped function
- `post_wrapping_code` is a list of additional wrapping objects with a similar structure to the wrappers, see the section below.

Post wrapping code

Complex wrappers need to provide additional wrapping code. Currently JSR supports additional wrapping of:

- Definition of a function (used, for example, to reintroduce `Date.now()` function to the wrapped `Date` object)

```
[
  {
    code_type: "function_define",
    original_function: "originalDateConstructor.now",
    parent_object: "window.Date",
    parent_object_property: "now",
    wrapping_function_args: "",
    wrapping_function_body: "return func(originalDateConstructor.r
  },
]
```

- Export a function from the wrapping namespace (currently not used, the following code exposes the unwrapped, i.e. original, version of Date.now to page scripts)

```
[
  {
    code_type: "function_export",
    parent_object: "window.Date",
    parent_object_property: "now",
    export_function_name: "originalDateConstructor.now",
  },
]
```

- Redefine an object property (used to prevent leaking iframe properties to the unwrapped objects):

```
{
  code_type: "object_properties",
  parent_object: "HTMLIFrameElement.prototype",
  parent_object_property: "contentWindow",
  wrapped_objects: [
    {
      original_name: "HTMLIFrameElement.prototype.__lookupGe
      wrapped_name: "cw",
    },
  ],
  wrapped_properties: [
    {
      property_name: "get",
      property_value: `
        function() {
          var parent=cw.call(this);
          try {
            parent.HTMLCanvasElement;
          }
          catch(d) {
            return; // HTMLIFrameElement.c
          }
          wrapping(parent);
          return parent;
        }`,
    },
  ],
}
```

The generated wrapper structure

To get a better idea how the code is generated see the following pseudo code. Please do not refer to any name created by the code builders from your wrapper. Use your custom names. If it is not available, please, open an issue where you explain what you are trying to achieve. It is probable that we will introduce a new property that allows to provide your name to the code builders. The code lives in an anonymous namespaces, so variables introduced here do not directly leak to page scripts.

```
// Define wrapped_name(s) variables holding the original JS objects
helping_code // if present
function wrapping_code_function_name(param) {
    // Store original function: either original_function which defaults to
    function replacement(wrapping_function_args) {
        wrapping_function_body
    }
    if (replace_original_function)
        original_function = replacement
    else
        parent_object.parent_object_property = replacement
    // Modify toString
    post_replacement_code
}
wrapping_code_function_name(window if wrapping_code_function_call_window) // 1
// wrappings generated by post wrapping code
```

Compiling the wrappers

Of course the wrappers need to be compiled to JavaScript before inserting the code to page scripts. See `code_builders.js` and `ffbug1267027.js`.

Registering a new wrapper

`fix_manifest.sh` automatically adds all modules with file name of `wrapping*.js` to the `manifest.json` of the extension. There is no need for any additional action.

Register new wrapper to be used by the extension in a level or available in the GUI.

See `levels.js` and its list `wrapping_groups.groups` . Once you add your wrapper to an existing group or create a new group, the wrapper becomes available in the built-in levels containing the group and in the GUI for custom levels.

JavaScript Restrictor

Generated by Doxygen 1.8.17

1 File Index	1
1.1 File List	1
2 File Documentation	3
2.1 chrome/http_shield_chrome.js File Reference	3
2.1.1 Function Documentation	4
2.1.1.1 addEventListener()	4
2.1.1.2 beforeSendHeadersListener()	4
2.1.1.3 insertHostInStats()	5
2.1.1.4 messageListener()	5
2.1.1.5 notifyBlockedHost()	5
2.1.1.6 onErrorOccuredListener()	6
2.1.1.7 onHeadersReceivedRequestListener()	7
2.1.2 Variable Documentation	7
2.1.2.1 blockedHosts	7
2.1.2.2 chromeErrorString	7
2.1.2.3 errorsAllowed	8
2.1.2.4 hostStatistics	8
2.1.2.5 httpClientErrorsAllowed	8
2.1.2.6 httpErrorList	8
2.1.2.7 uniqueErrorHostsLimit	8
2.1.2.8 uniqueErrorHostsRatio	9
2.2 common/background.js File Reference	9
2.2.1 Function Documentation	9
2.2.1.1 addListener()	9
2.2.1.2 connected()	9
2.2.1.3 if()	10
2.2.1.4 tabActivate()	10
2.2.1.5 tabUpdate()	10
2.2.1.6 updateBadge()	11
2.2.2 Variable Documentation	11
2.2.2.1 current_level	11
2.2.2.2 queryInfo	11
2.2.2.3 tab_levels	11
2.3 common/browser.js File Reference	11
2.3.1 Function Documentation	12
2.3.1.1 if()	12
2.4 common/code_builders.js File Reference	12
2.4.1 Function Documentation	12
2.4.1.1 define_page_context_function()	12
2.4.1.2 enclose_wrapping()	13
2.4.1.3 enclose_wrapping2()	13

2.4.1.4 generate_assign_function_code()	14
2.4.1.5 generate_object_properties()	14
2.4.1.6 wrap_code()	14
2.4.2 Variable Documentation	14
2.4.2.1 build_code	14
2.5 common/document_start.js File Reference	15
2.5.1 Function Documentation	15
2.5.1.1 sendMessage()	15
2.6 common/ffbug1267027.js File Reference	15
2.6.1 Function Documentation	15
2.6.1.1 build_code_ffbug()	16
2.6.1.2 define_page_context_function_ffbug()	16
2.6.1.3 generate_assign_function_code_ffbug()	17
2.6.1.4 generate_object_properties_ffbug()	17
2.6.1.5 is_firefox_blocking_scripts()	17
2.6.2 Variable Documentation	18
2.6.2.1 injectScriptElement	18
2.7 common/helpers.js File Reference	18
2.7.1 Function Documentation	18
2.7.1.1 escape()	18
2.8 common/http_shield_common.js File Reference	19
2.8.1 Function Documentation	19
2.8.1.1 addListener()	19
2.8.1.2 checkWhitelist()	20
2.8.1.3 commonMessageListener()	20
2.8.1.4 CSVToArray()	21
2.8.1.5 expandIPV6()	21
2.8.1.6 get()	22
2.8.1.7 if()	22
2.8.1.8 isIPV4()	22
2.8.1.9 isIPV4Private()	23
2.8.1.10 isIPV6()	23
2.8.1.11 isIPV6Private()	23
2.8.1.12 notifyBlockedRequest()	24
2.8.1.13 parseCSV()	25
2.8.1.14 readFile() [1/2]	25
2.8.1.15 readFile() [2/2]	25
2.8.2 Variable Documentation	25
2.8.2.1 doNotBlockHosts	25
2.8.2.2 localIPV4DNSZones	26
2.8.2.3 localIPV6DNSZones	26
2.8.2.4 readFile	26

2.9 common/inject.js File Reference	26
2.9.1 Variable Documentation	26
2.9.1.1 injectScript	27
2.10 common/levels.js File Reference	27
2.10.1 Function Documentation	28
2.10.1.1 addListener()	28
2.10.1.2 changedLevels()	28
2.10.1.3 get()	28
2.10.1.4 getCurrentLevelJSON()	29
2.10.1.5 if()	29
2.10.1.6 init_levels()	29
2.10.1.7 saveDomainLevels()	30
2.10.1.8 setDefaultLevel()	30
2.10.1.9 updateLevels()	30
2.10.2 Variable Documentation	31
2.10.2.1 default_level	31
2.10.2.2 domains	31
2.10.2.3 L0	31
2.10.2.4 L1	31
2.10.2.5 L2	31
2.10.2.6 L3	31
2.10.2.7 level_0	32
2.10.2.8 level_1	32
2.10.2.9 level_2	32
2.10.2.10 level_3	32
2.10.2.11 levels	33
2.10.2.12 levels_initialised	33
2.10.2.13 levels_updated_callbacks	33
2.10.2.14 wrapped_codes	33
2.10.2.15 wrapping_groups	33
2.11 common/options.js File Reference	34
2.11.1 Function Documentation	34
2.11.1.1 add_to_whitelist()	34
2.11.1.2 addEventListener()	35
2.11.1.3 control_http_request_shield()	35
2.11.1.4 edit_level()	35
2.11.1.5 getElementById()	36
2.11.1.6 getElementsByClassName()	36
2.11.1.7 getSelectValues()	36
2.11.1.8 if()	37
2.11.1.9 insert_levels()	37
2.11.1.10 load_on_off_switch()	37

2.11.1.11 loadWhitelist()	37
2.11.1.12 prepare_level_config()	38
2.11.1.13 remove_from_whitelist()	38
2.11.1.14 remove_level()	39
2.11.1.15 restore_level()	39
2.11.1.16 sendMessage()	40
2.11.1.17 show_existing_level()	40
2.11.1.18 update_whitelist()	41
2.11.2 Variable Documentation	41
2.11.2.1 html_element_value_source	42
2.12 common/options_domains.js File Reference	42
2.12.1 Function Documentation	42
2.12.1.1 escape()	42
2.12.1.2 if()	42
2.13 common/popup.js File Reference	43
2.13.1 Function Documentation	43
2.13.1.1 addEventListener()	43
2.13.1.2 changeActiveLevel()	44
2.13.1.3 close()	44
2.13.1.4 control_whitelist()	44
2.13.1.5 getElementById()	44
2.13.1.6 getElementsByClassName()	45
2.13.1.7 if()	45
2.13.1.8 load_on_off_switch()	45
2.13.1.9 query()	45
2.13.1.10 showRefreshPageOption()	46
2.13.2 Variable Documentation	46
2.13.2.1 e	46
2.13.2.2 fadeIn	46
2.13.2.3 fadeOut	46
2.13.2.4 myAddon	46
2.13.2.5 queryInfo	47
2.13.2.6 url	47
2.14 common/update.js File Reference	47
2.14.1 Function Documentation	47
2.14.1.1 addListener()	47
2.14.1.2 if()	47
2.14.1.3 installUpdate()	48
2.15 common/url.js File Reference	48
2.15.1 Function Documentation	48
2.15.1.1 extractSubDomains()	48
2.16 common/wrapping.js File Reference	49

2.16.1 Function Documentation	49
2.16.1.1 add_wrappers()	49
2.16.1.2 noise_function()	50
2.16.2 Variable Documentation	50
2.16.2.1 build_wrapping_code	50
2.16.2.2 noise_function	50
2.16.2.3 rounding_function	50
2.17 common/wrappingS-AJAX.js File Reference	50
2.18 common/wrappingS-BATTERY-CR.js File Reference	50
2.18.1 Function Documentation	50
2.18.1.1 add_wrappers()	51
2.19 common/wrappingS-DM.js File Reference	51
2.20 common/wrappingS-ECMA-ARRAY.js File Reference	51
2.20.1 Function Documentation	52
2.20.1.1 add_wrappers()	52
2.20.1.2 constructDecorator()	52
2.20.1.3 for() [1/2]	52
2.20.1.4 for() [2/2]	53
2.20.1.5 getBigIntDecorator()	53
2.20.1.6 getByteDecorator()	53
2.20.1.7 getFloatDecorator()	54
2.20.1.8 if() [1/2]	54
2.20.1.9 if() [2/2]	54
2.20.1.10 offsetDecorator()	55
2.20.1.11 packF32()	55
2.20.1.12 packF64()	56
2.20.1.13 packIEEE754()	56
2.20.1.14 redefineDataViewFunctions()	57
2.20.1.15 redefineNewArrayConstructors()	57
2.20.1.16 redefineNewArrayFunctions()	58
2.20.1.17 setBigIntDecorator()	58
2.20.1.18 setByteDecorator()	58
2.20.1.19 setFloatDecorator()	59
2.20.1.20 unpackF32()	60
2.20.1.21 unpackF64()	60
2.20.1.22 unpackIEEE754()	61
2.20.2 Variable Documentation	61
2.20.2.1 _data	61
2.20.2.2 _target	61
2.20.2.3 DEFAULT_TYPED_ARRAY_WRAPPER	62
2.20.2.4 j	62
2.20.2.5 offsetF	62

2.20.2.6 proxy	62
2.20.2.7 proxyHandler	62
2.20.2.8 typedTypes	63
2.20.2.9 wrappers	63
2.21 common/wrappingS-ECMA-DATE.js File Reference	63
2.22 common/wrappingS-ECMA-SHARED.js File Reference	63
2.22.1 Function Documentation	63
2.22.1.1 add_wrappers()	63
2.22.2 Variable Documentation	63
2.22.2.1 _data	64
2.22.2.2 proxy	64
2.22.2.3 proxyHandler	64
2.22.2.4 wrappers	64
2.23 common/wrappingS-H-C.js File Reference	65
2.24 common/wrappingS-HRT.js File Reference	65
2.24.1 Function Documentation	65
2.24.1.1 add_wrappers()	65
2.25 common/wrappingS-HTML-LS.js File Reference	65
2.25.1 Function Documentation	66
2.25.1.1 addEventListener()	66
2.25.1.2 callErrorListener()	66
2.25.1.3 executeEach()	67
2.25.1.4 open()	67
2.25.1.5 postError()	67
2.25.1.6 postMessage()	68
2.25.1.7 removeEventListener()	68
2.25.1.8 runPostMessage()	68
2.25.1.9 send()	69
2.25.1.10 workerAddEventListener()	69
2.25.1.11 workerPostMessage()	70
2.25.2 Variable Documentation	70
2.25.2.1 _old	70
2.25.2.2 addEventListener	70
2.25.2.3 api	70
2.25.2.4 errorListeners	70
2.25.2.5 onreadystatechange	71
2.25.2.6 postMessage	71
2.25.2.7 postMessageListeners	71
2.25.2.8 removeEventListener	71
2.25.2.9 script	71
2.25.2.10 slowBody	72
2.25.2.11 terminate	72

2.25.2.12 terminated	72
2.25.2.13 workerErrorListeners	72
2.25.2.14 workerMessageListeners	72
2.25.2.15 workerSelf	72
2.25.2.16 wrappers	72
2.25.2.17 xhr	73
2.26 common/wrappingS-PT2.js File Reference	73
2.26.1 Function Documentation	73
2.26.1.1 for()	73
2.26.1.2 if()	73
2.26.2 Variable Documentation	73
2.26.2.1 common_function_body	73
2.26.2.2 func	74
2.26.2.3 ret	74
2.26.2.4 wrappers	74
2.27 firefox/http_shield_firefox.js File Reference	74
2.27.1 Function Documentation	74
2.27.1.1 addListener()	74
2.27.1.2 beforeSendHeadersListener()	75
2.27.1.3 messageListener()	75
2.28 firefox/level_cache.js File Reference	76
2.28.1 Function Documentation	76
2.28.1.1 addListener()	76
2.28.1.2 contentScriptLevelSetter()	76
2.28.2 Variable Documentation	76
2.28.2.1 domains_bug1267027	77
2.29 chrome/level_cache.js File Reference	77
2.29.1 Function Documentation	77
2.29.1.1 addListener()	77
2.29.1.2 contentScriptLevelSetter()	77
2.30 firefox/manifest.json File Reference	77
2.31 chrome/manifest.json File Reference	77

Chapter 1

File Index

1.1 File List

Here is a list of all files with brief descriptions:

chrome/http_shield_chrome.js	3
chrome/level_cache.js	77
chrome/manifest.json	77
common/background.js	9
common/browser.js	11
common/code_builders.js	12
common/document_start.js	15
common/ffbug1267027.js	15
common/helpers.js	18
common/http_shield_common.js	19
common/inject.js	26
common/levels.js	27
common/options.js	34
common/options_domains.js	42
common/popup.js	43
common/update.js	47
common/url.js	48
common/wrapping.js	49
common/wrappingS-AJAX.js	50
common/wrappingS-BATTERY-CR.js	50
common/wrappingS-DM.js	51
common/wrappingS-ECMA-ARRAY.js	51
common/wrappingS-ECMA-DATE.js	63
common/wrappingS-ECMA-SHARED.js	63
common/wrappingS-H-C.js	65
common/wrappingS-HRT.js	65
common/wrappingS-HTML-LS.js	65
common/wrappingS-PT2.js	73
firefox/http_shield_firefox.js	74
firefox/level_cache.js	76
firefox/manifest.json	77

Chapter 2

File Documentation

2.1 chrome/http_shield_chrome.js File Reference

Functions

- window `addEventListener` ("online", function() { `browser.webRequest.onErrorOccurred.addListener`(`onErrorOccuredListener`, {urls:["<all_urls>"]});})
If the browser regained connectivity - came online.
- function `onErrorOccuredListener` (responseDetails)
- function `onHeadersReceivedRequestListener` (headers)
- function `insertHostInStats` (targetDomain)
- function `beforeSendHeadersListener` (requestDetail)
- function `notifyBlockedHost` (host)
- function `messageListener` (message, sender, sendResponse)

Variables

- var `blockedHosts` = new Object()
Associative array of hosts, that are currently blocked based on their previous actions.
- var `hostStatistics` = new Object()
Information about hosts, for which cant be used DNS query.
- var `uniqueErrorHostsRatio` = 10.0
Percentage of hosts, that can register an HTTP error response.
- var `uniqueErrorHostsLimit` = 20
If there are more hosts than uniqueErrorHostsLimit which are targeted from the same origin, the origin host becomes blocked.
- var `errorsAllowed` = 10
Number of Request Timed Out errors allowed for one origin.
- var `httpClientErrorsAllowed` = 5
Number of HTTP client errors (eg. 404 not found, 403 forbidden etc.) per requestTimeInterval.
- var `httpErrorList`
Errors that are considered as possible attacker threat.
- var `chromeErrorString` = "net::ERR_CONNECTION_TIMED_OUT"

2.1.1 Function Documentation

2.1.1.1 addEventListener()

```
window addEventListener (
    "online" ,
    function() { browser.webRequest.onErrorOccurred.addListener(onErrorOccuredListener,
{urls:["<all_urls>"]});} )
```

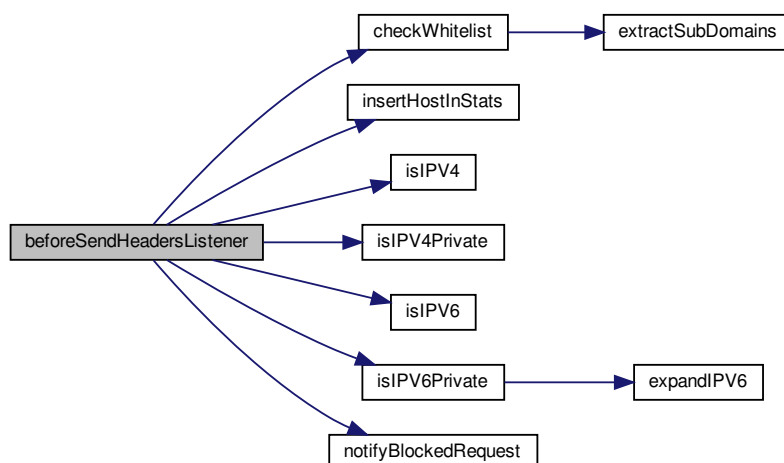
If the browser regained connectivity - came online.

If the browser lost connectivity - gone offline.

2.1.1.2 beforeSendHeadersListener()

```
function beforeSendHeadersListener (
    requestDetail )
```

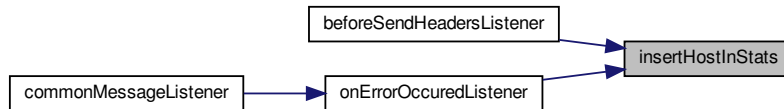
webRequest event listener, hooked to onBeforeSendHeaders event Catches all requests, analyzes them, does blocking, modifies counters, blocks if one of the limits was exceeded Here is the call graph for this function:



2.1.1.3 insertHostInStats()

```
function insertHostInStats (  
    targetDomain )
```

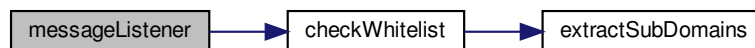
Function that creates object representing source host Receives target hostname in targetDomain argument Here is the caller graph for this function:



2.1.1.4 messageListener()

```
function messageListener (  
    message,  
    sender,  
    sendResponse )
```

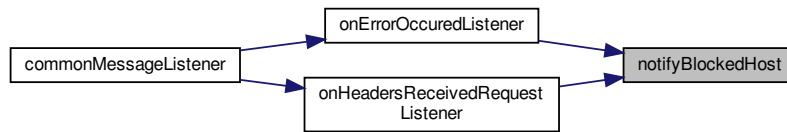
webRequest event listener, hooked to onMessage event obtains message string in message, message sender in sender and function for sending response in sendResponse Does appropriate action based on message text Here is the call graph for this function:



2.1.1.5 notifyBlockedHost()

```
function notifyBlockedHost (  
    host )
```

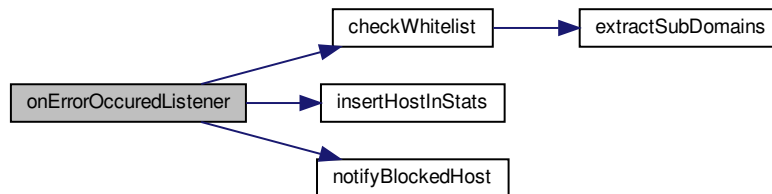
Creates and presents notification to the user works with webExtensions notification API Here is the caller graph for this function:



2.1.1.6 onErrorOccurredListener()

```
function onErrorOccurredListener (
    responseDetails )
```

webRequest event listener, hooked to onErrorOccurred event Catches all errors, checks them for Request Timed out errors Iterates error counter, blocks the host if limit was exceeded Takes object representing error in responseDetails variable Here is the call graph for this function:



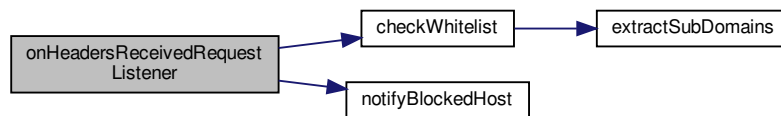
Here is the caller graph for this function:



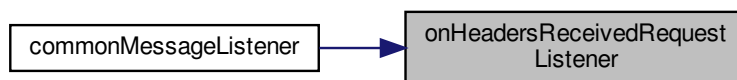
2.1.1.7 onHeadersReceivedRequestListener()

```
function onHeadersReceivedRequestListener (
    headers )
```

webRequest event listener, hooked to onErroccured event Catches all responses, analyzes those with record in hostStatistics Modifies counters, blocks if one of the limits was exceeded Here is the call graph for this function:



Here is the caller graph for this function:



2.1.2 Variable Documentation

2.1.2.1 blockedHosts

```
var blockedHosts = new Object()
```

Associative array of hosts, that are currently blocked based on their previous actions.

2.1.2.2 chromeErrorString

```
var chromeErrorString = "net::ERR_CONNECTION_TIMED_OUT"
```

String that defines Request Timed Out error in Chrome according to: <https://developer.chrome.com/extensions/webRequest#event-onErrorOccurred> It's not backwards compatible, but it's the best we have

2.1.2.3 errorsAllowed

```
var errorsAllowed = 10
```

Number of Request Timed Out errors allowed for one origin.

2.1.2.4 hostStatistics

```
var hostStatistics = new Object()
```

Information about hosts, for which cant be used DNS query.

2.1.2.5 httpClientErrorsAllowed

```
var httpClientErrorsAllowed = 5
```

Number of HTTP client errors (eg. 404 not found, 403 forbidden etc.) per requestTimeInterval.

2.1.2.6 httpErrorList

```
var httpErrorList
```

Initial value:

```
= {  
  400:true,  
  404:true,  
  405:true,  
  406:true,  
  408:true,  
  410:true,  
  413:true,  
  414:true,  
  415:true,  
  501:true,  
  503:true,  
  505:true  
}
```

Errors that are considered as possible attacker threat.

2.1.2.7 uniqueErrorHostsLimit

```
var uniqueErrorHostsLimit = 20
```

If there are more hosts than uniqueErrorHostsLimit which are targeted from the same origin, the origin host becomes blocked.

2.1.2.8 uniqueErrorHostsRatio

```
var uniqueErrorHostsRatio = 10.0
```

Percentage of hosts, that can register an HTTP error response.

2.2 common/background.js File Reference

Functions

- `if ((typeof browser)=== "undefined") {redefineNewArrayFunctions}`
- function [updateBadge](#) (level)
- function [tabUpdate](#) (tabid, changeInfo)
- function [tabActivate](#) (activeInfo)
- browser tabs onUpdated [addListener](#) ([tabUpdate](#))
- function [connected](#) (port)

Variables

- var [tab_levels](#) = {}
- var [current_level](#) = {level_id: "?"}
- var [queryInfo](#)

2.2.1 Function Documentation

2.2.1.1 addListener()

```
browser runtime onMessage addListener (
    tabUpdate )
```

2.2.1.2 connected()

```
function connected (
    port )
```

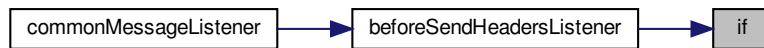
Create a port to popup window

The communication cannels are mostly used because `browser.runtime.getBackgroundPage()` does not work as expected. See also https://bugzilla.mozilla.org/show_bug.cgi?id=1329304. We always send back current level

2.2.1.3 if()

```
if (  
    (typeof browser) === "undefined" ) {redefineNewArrayFunctions}
```

Here is the caller graph for this function:



2.2.1.4 tabActivate()

```
function tabActivate (  
    activeInfo )
```

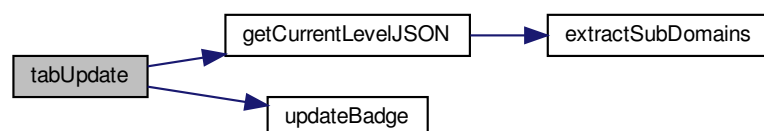
Here is the call graph for this function:



2.2.1.5 tabUpdate()

```
function tabUpdate (  
    tabid,  
    changeInfo )
```

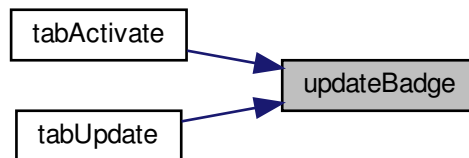
Here is the call graph for this function:



2.2.1.6 updateBadge()

```
function updateBadge (  
    level )
```

Here is the caller graph for this function:



2.2.2 Variable Documentation

2.2.2.1 current_level

```
var current_level = {level_id: "?"}
```

2.2.2.2 queryInfo

```
var queryInfo
```

Initial value:

```
= {  
  active: true,  
  currentWindow: true  
}
```

2.2.2.3 tab_levels

```
var tab_levels = {}
```

2.3 common/browser.js File Reference

Functions

- `if ((typeof browser)=== "undefined")`

2.3.1 Function Documentation

2.3.1.1 if()

```
if (
    (typeof browser) === "undefined" )
```

2.4 common/code_builders.js File Reference

Functions

- function [enclose_wrapping](#) (code,...args)
- function [enclose_wrapping2](#) (code, name, params, call_with_window)
- function [define_page_context_function](#) (wrapper)
- function [generate_assign_function_code](#) (code_spec_obj)
- function [generate_object_properties](#) (code_spec_obj)
- function [wrap_code](#) (wrappers)

Variables

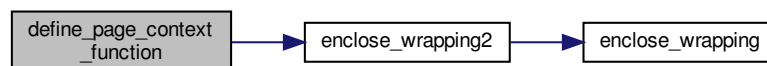
- var [build_code](#)

2.4.1 Function Documentation

2.4.1.1 define_page_context_function()

```
function define_page_context_function (
    wrapper )
```

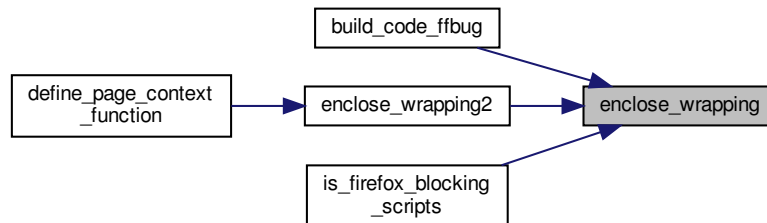
This function create code (as string) that creates code that can be used to inject (or overwrite) a function in the page context. Here is the call graph for this function:



2.4.1.2 `enclose_wrapping()`

```
function enclose_wrapping (
    code,
    args )
```

Create IIFE to wrap the code in closure Here is the caller graph for this function:



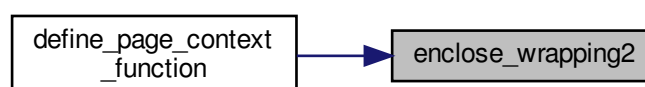
2.4.1.3 `enclose_wrapping2()`

```
function enclose_wrapping2 (
    code,
    name,
    params,
    call_with_window )
```

Create wrapping that might be IIFE or a function that is immediately called and also available for future. Here is the call graph for this function:



Here is the caller graph for this function:



2.4.1.4 generate_assign_function_code()

```
function generate_assign_function_code (
    code_spec_obj )
```

This function creates code that assigns an already defined function to given property.

2.4.1.5 generate_object_properties()

```
function generate_object_properties (
    code_spec_obj )
```

This function wraps object properties using Object.defineProperty.

2.4.1.6 wrap_code()

```
function wrap_code (
    wrappers )
```

Transform wrapping arrays into code.

Parameters

Array	of wrapping arrays.
-------	---------------------

Here is the caller graph for this function:



2.4.2 Variable Documentation

2.4.2.1 build_code

```
var build_code
```

This function builds the wrapping code.

2.5 common/document_start.js File Reference

Functions

- browser runtime [sendMessage](#) ({ message:"get wrapping for URL", url:window.location.href }, function handleResponse(reply) { [injectScript](#)(reply.code, [reply.wrappers](#), reply.ffbug1267027);})
Get current level configuration from the background script.

2.5.1 Function Documentation

2.5.1.1 sendMessage()

```
browser runtime sendMessage (
    { message:"get wrapping for URL", url:window.location.href } ,
    function handleResponse(reply) { injectScript(reply.code, reply.wrappers, reply.↵
ffbug1267027);} )
```

Get current level configuration from the background script.

2.6 common/ffbug1267027.js File Reference

Functions

- function [define_page_context_function_ffbug](#) (wrapper)
- function [generate_assign_function_code_ffbug](#) (code_spec_obj)
- function [generate_object_properties_ffbug](#) (code_spec_obj)
- function [build_code_ffbug](#) (wrapper,...args)
- function [is_firefox_blocking_scripts](#) ()

Variables

- var [injectScriptElement](#) = [injectScript](#)
The original script injecting function.

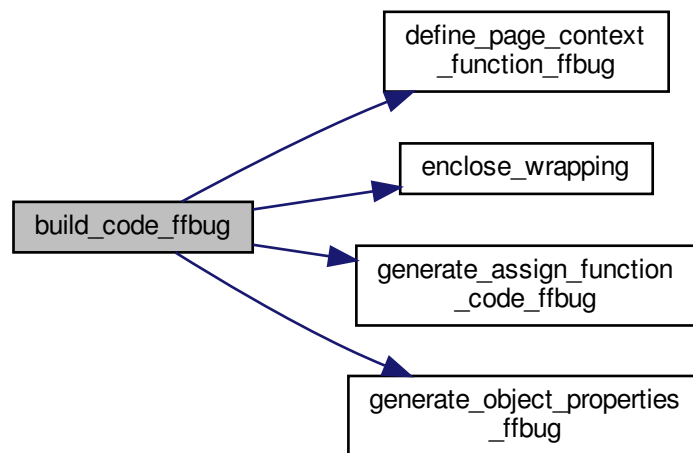
2.6.1 Function Documentation

2.6.1.1 build_code_ffbug()

```
function build_code_ffbug (  
    wrapper,  
    args )
```

Alternative definition of the build_code function.

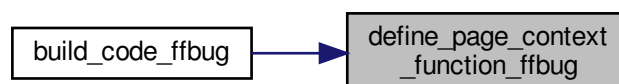
FIXME: this code needs improvements, see bug #25 Here is the call graph for this function:



2.6.1.2 define_page_context_function_ffbug()

```
function define_page_context_function_ffbug (  
    wrapper )
```

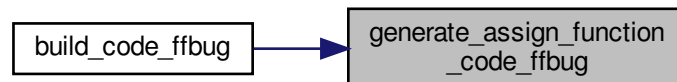
This function create code (as string) that creates code that can be used to inject (or overwrite) a function in the page context. Supporting code for dealing with bug https://bugzilla.mozilla.org/show_bug.cgi?id=1267027. Here is the caller graph for this function:



2.6.1.3 generate_assign_function_code_ffbug()

```
function generate_assign_function_code_ffbug (
    code_spec_obj )
```

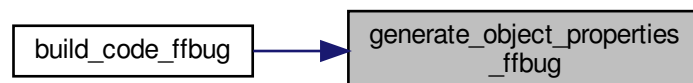
This function creates code that assigns an already defined function to given property. Supporting code for dealing with bug https://bugzilla.mozilla.org/show_bug.cgi?id=1267027. Here is the caller graph for this function:



2.6.1.4 generate_object_properties_ffbug()

```
function generate_object_properties_ffbug (
    code_spec_obj )
```

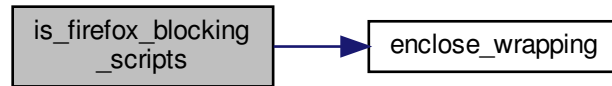
This function wraps object properties using `Object.defineProperty`. Supporting code for dealing with bug https://bugzilla.mozilla.org/show_bug.cgi?id=1267027. Here is the caller graph for this function:



2.6.1.5 is_firefox_blocking_scripts()

```
function is_firefox_blocking_scripts ( )
```

Determine if we are running in the context where FF blocks script inserting due to bug https://bugzilla.mozilla.org/show_bug.cgi?id=1267027 Here is the call graph for this function:



2.6.2 Variable Documentation

2.6.2.1 injectScriptElement

```
var injectScriptElement = injectScript
```

The original script injecting function.

2.7 common/helpers.js File Reference

Functions

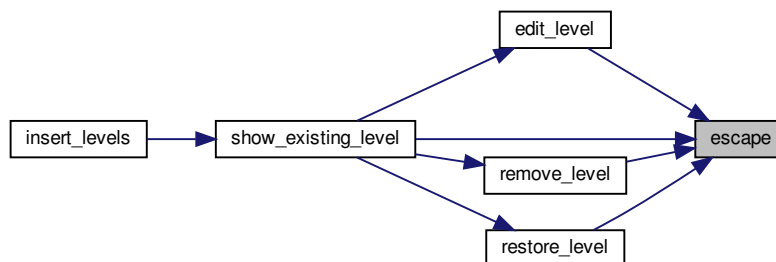
- function `escape` (str)

2.7.1 Function Documentation

2.7.1.1 escape()

```
function escape (
    str )
```

Here is the caller graph for this function:



2.8 common/http_shield_common.js File Reference

Functions

- `if ((typeof browser)===undefined)`
- browser storage sync `get` (`["requestShieldOn"]`, `function(result){ if(result.requestShieldOn===undefined||result.requestShieldOn) { browser.webRequest.onBeforeSendHeaders.addListener(beforeSendHeadersListener, {urls:["<all_urls>"],["blocking", "requestHeaders"]};if(typeof onHeadersReceivedRequestListener===function) { browser.webRequest.onHeadersReceived.addListener(onHeadersReceivedRequestListener, {urls:["<all_urls>"],["blocking"]};browser.webRequest.onErrorOccurred.addListener(onErrorOccuredListener, {urls:["<all_urls>"]});} } }`)
 - *Check the storage for requestShieldOn object.*
- browser runtime `onMessage` `addListener` (`commonMessageListener`)
 - *Hook up the listener for receiving messages.*
- `readFile` (`browser.runtime.getURL("ipv4.csv")`) `.then`(`_res`)
 - *Obtain file path in user's file system and read CSV file with IPv4 local zones.*
- `readFile` (`browser.runtime.getURL("ipv6.csv")`) `.then`(`_res`)
 - *Obtain file path in user's file system and read CSV file with IPv6 local zones.*
- function `isIPV4` (`url`)
- function `isIPV6` (`url`)
- function `isIPV4Private` (`ipAddr`)
- function `isIPV6Private` (`ipAddr`)
- function `parseCSV` (`csv`, `ipv4`)
- function `CSVToArray` (`strData`)
- function `expandIPV6` (`ip6addr`)
- function `checkWhitelist` (`hostname`)
- function `notifyBlockedRequest` (`origin`, `target`, `resource`)
- function `commonMessageListener` (`message`, `sender`, `sendResponse`)

Variables

- var `localIPV4DNSZones`
 - *Locally served IPV4 DNS zones loaded from IANA.*
- var `localIPV6DNSZones`
 - *Locally served IPV6 DNS zones loaded from IANA.*
- var `doNotBlockHosts` = `new Object()`
 - *Associative array of hosts, that are currently among trusted "do not blocked" hosts.*
- let `readFile`
 - *Function for reading locally stored csv file.*

2.8.1 Function Documentation

2.8.1.1 `addListener()`

```
browser runtime onMessage addListener (
    commonMessageListener )
```

Hook up the listener for receiving messages.

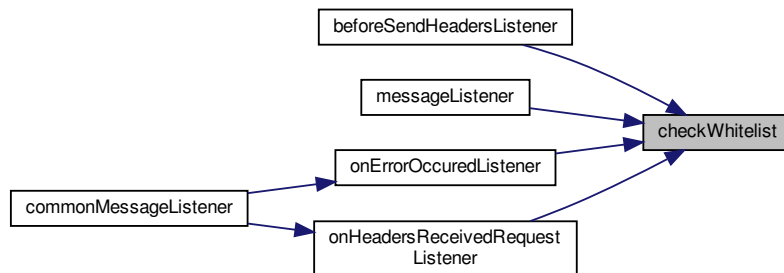
2.8.1.2 checkWhitelist()

```
function checkWhitelist (
    hostname )
```

Here is the call graph for this function:



Here is the caller graph for this function:

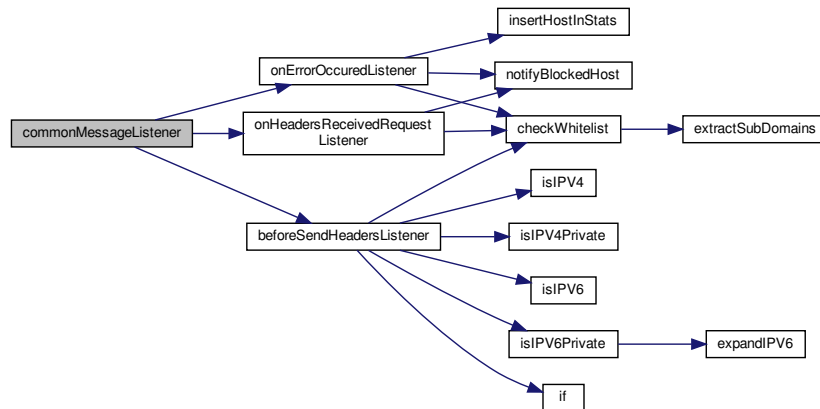


2.8.1.3 commonMessageListener()

```
function commonMessageListener (
    message,
    sender,
    sendResponse )
```

Event listener hooked up to `webExtensions` `onMessage` event Receives full message in `message`, sender of the message in `sender`, function for sending response in `sendResponse` Does appropriate action based on message

Here is the call graph for this function:



2.8.1.4 CSVToArray()

```
function CSVToArray (
    strData )
```

Auxillary function for parsing CSV files Converts CSV to array strData - loaded CSV file Returns array containing CSV rows Here is the caller graph for this function:



2.8.1.5 expandIPV6()

```
function expandIPV6 (
    ip6addr )
```

Function for expanding shorten ipv6 addresses Takes valid ipv6 address in ip6addr argument Returns expanded ipv6 address in string This function should be only called on valid IPv6 address Here is the caller graph for this function:



2.8.1.6 get()

```

browser storage sync get (
    function(result){ if(result.whitelistedHosts !==undefined) doNotBlockHosts=result.↔
    whitelistedHosts;} )
  
```

Check the storage for requestShieldOn object.

2.8.1.7 if()

```

if (
    (typeof browser) === "undefined" )
  
```

2.8.1.8 isIPv4()

```

function isIPv4 (
    url )
  
```

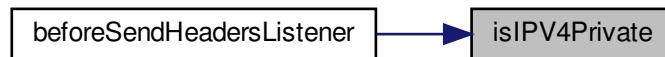
Checks validity of IPv4 addresses, returns TRUE if the url matches IPv4 regex FALSE otherwise Here is the caller graph for this function:



2.8.1.9 isIPv4Private()

```
function isIPv4Private (  
    ipAddr )
```

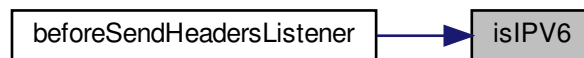
Checks whether the ipAddr is found in IPv4 localZones Returns TRUE if ipAddr exists in localZones fetched from IANA FALSE otherwise This function should only be called on valid IPv4 address Here is the caller graph for this function:



2.8.1.10 isIPv6()

```
function isIPv6 (  
    url )
```

Checks validity IPv6 address Returns TRUE, if URL is valid IPV6 address FALSE otherwise Here is the caller graph for this function:



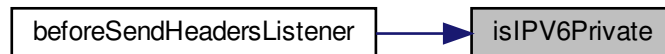
2.8.1.11 isIPv6Private()

```
function isIPv6Private (  
    ipAddr )
```


Checks whether the ipAddr is found in IPv6 localZones Returns TRUE if ipAddr exists in localZones fetched from IANA FALSE otherwise This function should only be called on valid IPv6 address Here is the call graph for this function:



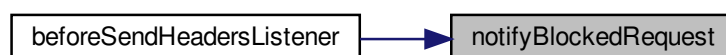
Here is the caller graph for this function:



2.8.1.12 notifyBlockedRequest()

```
function notifyBlockedRequest (
    origin,
    target,
    resource )
```

Creates and presents notification to the user works with webExtensions notification API Creates notification about blocked request Arguments: origin - origin of the request target - target of the request resource - type of the resource Here is the caller graph for this function:



2.8.1.13 parseCSV()

```
function parseCSV (
    csv,
    ipv4 )
```

Function for parsing CSV files obtained from IANA Strips .IN-ADDR and .IP6 from zones and comma delimiter, merges them into array by CSV rows Arguments: csv - CSV obtained from IANA ipv4 - bool, saying whether the csv is IPv4 CSV or IPv6 Returns: Array of parsed CSV values Here is the call graph for this function:



2.8.1.14 readFile() [1/2]

```
readFile (
    browser.runtime. getURL"ipv4.csv" )
```

Obtain file path in user's file system and read CSV file with IPv4 local zones.

2.8.1.15 readFile() [2/2]

```
readFile (
    browser.runtime. getURL"ipv6.csv" )
```

Obtain file path in user's file system and read CSV file with IPv6 local zones.

2.8.2 Variable Documentation

2.8.2.1 doNotBlockHosts

```
var doNotBlockHosts = new Object ()
```

Associative array of hosts, that are currently among trusted "do not blocked" hosts.

2.8.2.2 localIPV4DNSZones

```
var localIPV4DNSZones
```

Locally served IPV4 DNS zones loaded from IANA.

2.8.2.3 localIPV6DNSZones

```
var localIPV6DNSZones
```

Locally served IPV6 DNS zones loaded from IANA.

2.8.2.4 readFile

```
let readFile
```

Initial value:

```
= (_path) => {  
  return new Promise((resolve, reject) => {  
  
    fetch(_path, {mode:'same-origin'})  
      .then(function(_res) {  
  
        return _res.blob();  
      })  
      .then(function(_blob) {  
        var reader = new FileReader();  
  
        reader.addEventListener("loadend", function() {  
          resolve(this.result);  
        });  
  
        reader.readAsText(_blob);  
      })  
      .catch(error => {  
        reject(error);  
      });  
    });  
}
```

Function for reading locally stored csv file.

2.9 common/inject.js File Reference

Variables

- window [injectScript](#)

2.9.1 Variable Documentation

2.9.1.1 injectScript

window injectScript

Initial value:

```
= function (text) {
  var parent = document.documentElement;
  var script = document.createElement('script');
  script.text = text;
  script.async = false;
  parent.insertBefore(script, parent.firstChild);
  parent.removeChild(script);
}
```

Execute given script in the page's JavaScript context.

This function is a modified version of the similar function from Privacy Badger <https://www.eff.org/privacybadger> <https://github.com/EFForg/privacybadger/blob/master/src/js/utils.js> Copyright (C) 2014 Electronic Frontier Foundation

Derived from Adblock Plus Copyright (C) 2006-2013 Eyeo GmbH

Privacy Badger is free software: you can redistribute it and/or modify it under the terms of the GNU General Public License version 3 as published by the Free Software Foundation.

Parameters

<code>{String}</code>	text The content of the script to insert.
-----------------------	---

2.10 common/levels.js File Reference

Functions

- `if ((typeof browser)=== "undefined")`
- function `init_levels` ()
- function `updateLevels` (res)
- browser storage sync `get` (null, `updateLevels`)
- function `changedLevels` (changed, area)
- browser storage onChanged `addListener` (`changedLevels`)
- function `setDefaultLevel` (level)
- function `saveDomainLevels` ()
- function `getCurrentLevelJSON` (url)

Variables

- var `wrapping_groups`
- var `level_0`
- var `level_1`
- var `level_2`
- var `level_3`
- const `L0` = "0"
- const `L1` = "1"
- const `L2` = "2"

- const `L3` = "3"
- var `levels` = {}
- var `default_level` = {}
- var `domains` = {}
- var `wrapped_codes` = {}
- let `levels_initialised` = false
- let `levels_updated_callbacks` = []

2.10.1 Function Documentation

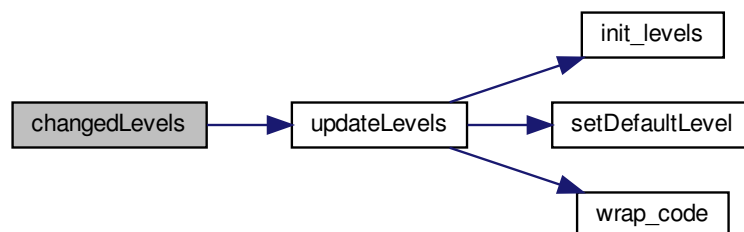
2.10.1.1 `addListener()`

```
browser storage onChanged addListener (  
    changedLevels )
```

2.10.1.2 `changedLevels()`

```
function changedLevels (  
    changed,  
    area )
```

Here is the call graph for this function:



2.10.1.3 `get()`

```
browser storage sync get (  
    null ,  
    updateLevels )
```

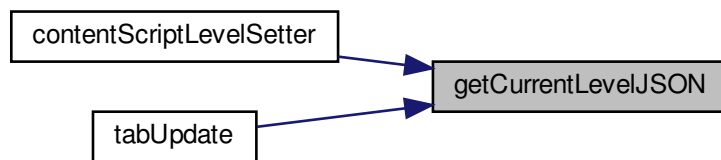
2.10.1.4 getCurrentLevelJSON()

```
function getCurrentLevelJSON (
    url )
```

Here is the call graph for this function:



Here is the caller graph for this function:



2.10.1.5 if()

```
if (
    (typeof browser) !== "undefined" )
```

2.10.1.6 init_levels()

```
init_levels ( )
```

Here is the caller graph for this function:



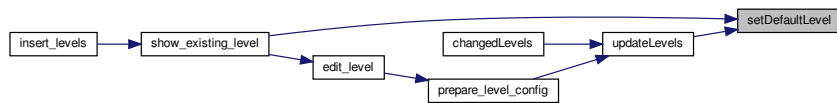
2.10.1.7 saveDomainLevels()

```
function saveDomainLevels ( )
```

2.10.1.8 setDefaultLevel()

```
function setDefaultLevel (
    level )
```

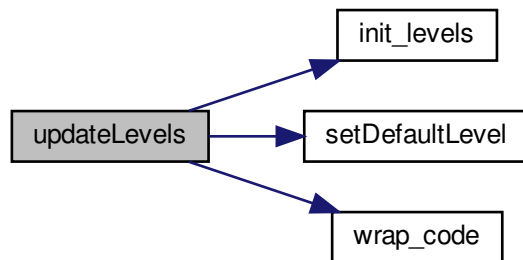
Here is the caller graph for this function:



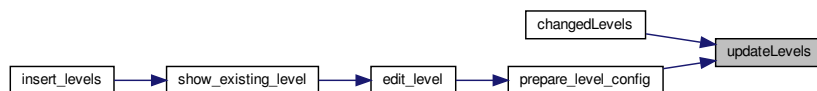
2.10.1.9 updateLevels()

```
function updateLevels (
    res )
```

Here is the call graph for this function:



Here is the caller graph for this function:



2.10.2 Variable Documentation

2.10.2.1 default_level

```
var default_level = {}
```

2.10.2.2 domains

```
var domains = {}
```

2.10.2.3 L0

```
const L0 = "0"
```

2.10.2.4 L1

```
const L1 = "1"
```

2.10.2.5 L2

```
const L2 = "2"
```

2.10.2.6 L3

```
const L3 = "3"
```


2.10.2.7 level_0

```
var level_0
```

Initial value:

```
= {  
  "level_id": "0",  
  "level_text": "Built-in 0",  
  "level_description": "No protection at all",  
}
```

2.10.2.8 level_1

```
var level_1
```

Initial value:

```
= {  
  "level_id": "1",  
  "level_text": "Built-in 1",  
  "level_description": "Minimal level of protection",  
  "time_precision": true,  
  "time_precision_precision": 2,  
  "time_precision_randomize": false,  
  "hardware": true,  
  "battery": true,  
}
```

2.10.2.9 level_2

```
var level_2
```

Initial value:

```
= {  
  "level_id": "2",  
  "level_text": "Built-in 2",  
  "level_description": "Recommended level of protection for most sites",  
  "time_precision": true,  
  "time_precision_precision": 1,  
  "time_precision_randomize": false,  
  "hardware": true,  
  "battery": true,  
  "htmlcanvaselement": true,  
}
```

2.10.2.10 level_3

```
var level_3
```

Initial value:

```
= {  
  "level_id": "3",  
  "level_text": "Built-in 3",  
  "level_description": "High level of protection",  
  "time_precision": true,  
  "time_precision_precision": 0,  
  "time_precision_randomize": true,  
  "hardware": true,  
  "battery": true,  
  "htmlcanvaselement": true,  
}
```

```
"xhr": true,  
"xhr_behaviour_block": false,  
"xhr_behaviour_ask": true,  
"arrays": true,  
"arrays_mapping": true,  
"shared_array": true,  
"shared_array_approach_block": true,  
"shared_array_approach_polyfill": false,  
"webworker": true,  
"webworker_approach_polyfill": true,  
"webworker_approach_slow": false,  
}
```

2.10.2.11 levels

```
var levels = {}
```

2.10.2.12 levels_initialised

```
let levels_initialised = false
```

2.10.2.13 levels_updated_callbacks

```
let levels_updated_callbacks = []
```

2.10.2.14 wrapped_codes

```
var wrapped_codes = {}
```

2.10.2.15 wrapping_groups

```
var wrapping_groups
```

Wrapping groups

Used to control the built-in levels and options GUI.

2.11 common/options.js File Reference

Functions

- `if ((typeof browser)=== "undefined")`
- function `prepare_level_config` (action_descr, params=wrapping_groups.empty_level)
- function `edit_level` (id)
- function `restore_level` (id, level_params)
- function `show_existing_level` (levelsEl, level)
- function `remove_level` (id)
- function `insert_levels` ()
- window `addEventListener` ("load", function() { `if(!levels_initialised)` { `levels_updated_callbacks.`←
`push(insert_levels);`} else { `insert_levels(); loadWhitelist();load_on_off_switch();`})
- document `getElementById` ("new_level").`addEventListener`("click" => `prepare_level_config`("Add new level"))
- document document document document `getElementsByClassName` ("slider")[0].`addEventListener`("click"
- function `add_to_whitelist` ()
- function `remove_from_whitelist` ()
- function `update_whitelist` (listbox)
- function `sendMessage` (message)
- function `getSelectValues` (select)
- function `loadWhitelist` ()
- function `load_on_off_switch` ()
- function `control_http_request_shield` ()

Variables

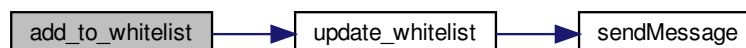
- const `html_element_value_source`
A map where to look for the values in HTML elements.

2.11.1 Function Documentation

2.11.1.1 `add_to_whitelist()`

```
function add_to_whitelist ( )
```

Here is the call graph for this function:



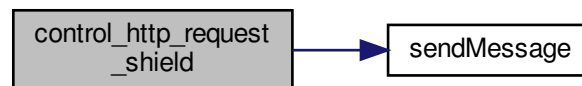
2.11.1.2 addEventListener()

```
window addEventListener (
    "load" ,
    function() { if(!levels_initialised) { levels_updated_callbacks.push(insert_levels);}
else { insert_levels();} loadWhitelist();load_on_off_switch();} )
```

2.11.1.3 control_http_request_shield()

```
function control_http_request_shield ( )
```

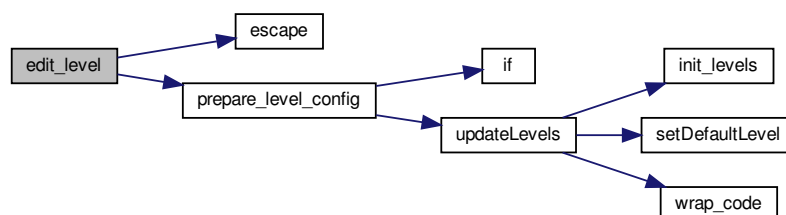
Here is the call graph for this function:



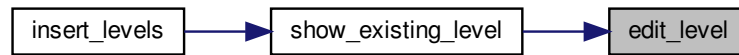
2.11.1.4 edit_level()

```
function edit_level (
    id )
```

Here is the call graph for this function:



Here is the caller graph for this function:



2.11.1.5 getElementById()

```
document document document getElementById (
    "new_level" ) => prepare_level_config("Add new level")
```

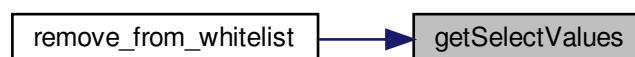
2.11.1.6 getElementsByClassName()

```
document document document document getElementsByClassName (
    "slider" )
```

2.11.1.7 getSelectValues()

```
function getSelectValues (
    select )
```

Here is the caller graph for this function:



2.11.1.8 if()

```
if (
    (typeof browser) === "undefined" )
```

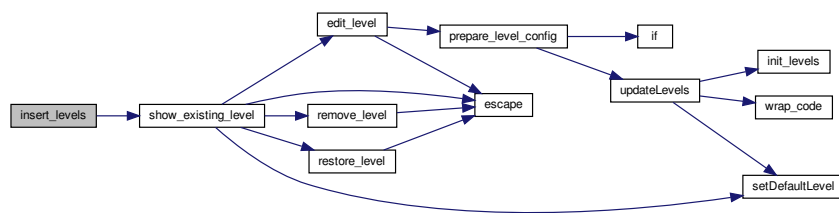
Here is the caller graph for this function:



2.11.1.9 insert_levels()

```
function insert_levels ( )
```

Here is the call graph for this function:



2.11.1.10 load_on_off_switch()

```
function load_on_off_switch ( )
```

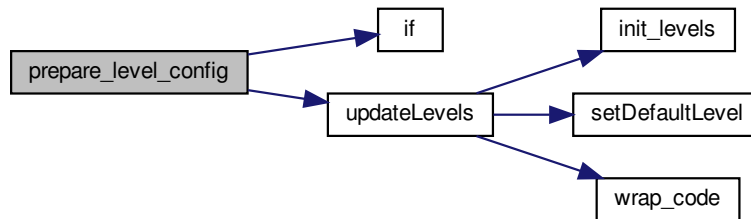
2.11.1.11 loadWhitelist()

```
function loadWhitelist ( )
```

2.11.1.12 prepare_level_config()

```
function prepare_level_config (
    action_descr,
    params = wrapping_groups.empty_level )
```

Here is the call graph for this function:



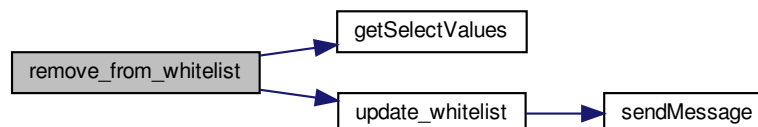
Here is the caller graph for this function:



2.11.1.13 remove_from_whitelist()

```
function remove_from_whitelist ( )
```

Here is the call graph for this function:



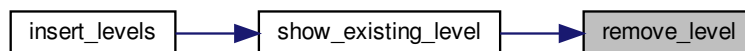
2.11.1.14 remove_level()

```
function remove_level (  
    id )
```

Here is the call graph for this function:



Here is the caller graph for this function:



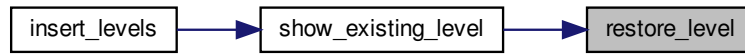
2.11.1.15 restore_level()

```
function restore_level (  
    id,  
    level_params )
```

Here is the call graph for this function:



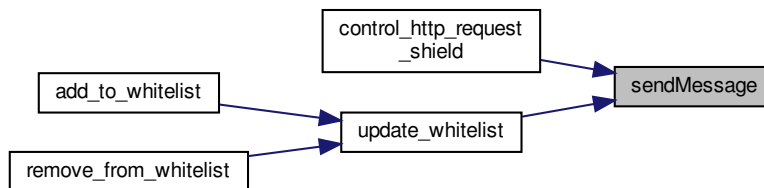
Here is the caller graph for this function:



2.11.1.16 sendMessage()

```
function sendMessage (
    message )
```

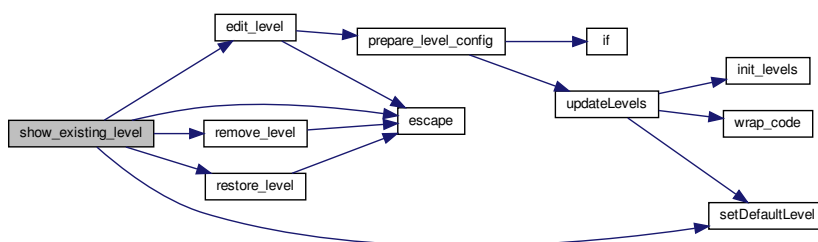
Here is the caller graph for this function:



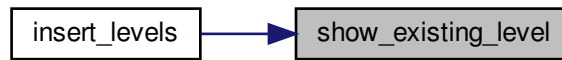
2.11.1.17 show_existing_level()

```
function show_existing_level (
    levelsEl,
    level )
```

Here is the call graph for this function:



Here is the caller graph for this function:



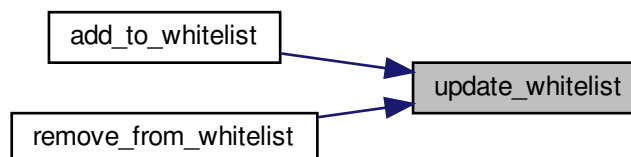
2.11.1.18 update_whitelist()

```
function update_whitelist (  
    listbox )
```

Here is the call graph for this function:



Here is the caller graph for this function:



2.11.2 Variable Documentation

2.11.2.1 `html_element_value_source`

```
const html_element_value_source
```

Initial value:

```
= {  
  "select": "value",  
  "input-checkbox": "checked",  
  "input-radio": "checked",  
}
```

A map where to look for the values in HTML elements.

2.12 `common/options_domains.js` File Reference

Functions

- `if` (`(typeof browser)=== "undefined"`)
- function `escape` (`str`)

2.12.1 Function Documentation

2.12.1.1 `escape()`

```
function escape (  
    str )
```

2.12.1.2 `if()`

```
if (  
    (typeof browser) == "undefined" )
```

2.13 common/popup.js File Reference

Functions

- `if ((typeof browser)=== "undefined")`
- function `showRefreshPageOption ()`
- function `changeActiveLevel (activeEl)`
- browser tabs `query (queryInfo, function(tabs) { let tab=tabs[0];url=new URL(tab.url);url.hostname=url.↵ hostname.replace(/^www\./,"");if(url.hostname=== ""||url.hostname===myAddon.hostname||url.hostname=="newtab") { document.getElementById("current_site_level_settings").style.opacity=fadeOut;return;} else { document.getElementById("current_site").innerHTML=url.hostname;} var port_to_background=browser.runtime.connect({name:"port_from↵ _popup"});var current_level={ level_id:"?" };port_to_background.onMessage.addListener(function(msg) { current_level=msg;var selectEl=document.getElementById("level-select");selectEl.innerHTML=`< span class="level level_control" id="default_level_select" title="Set the default level">Default`;document.getElementById("↵ _level_select").addEventListener("click", function() { delete domains[url.hostname];saveDomainLevels();changeActiveLevel(this↵ level_id in levels) { let level=levels[level_id];selectEl.appendChild(document.createRange().create↵ ContextualFragment(`< span class="level level_control" id="select-${level.level_id}" title="${level.level↵ text}">${escape(level.level_id)}`));document.getElementById(`select-${level.level_id}`).addEventListener("click", function() { domains[url.hostname]=[level_id:level.level_id,];saveDomainLevels();changeActiveLevel(this);current_level=level;}); if(current_level.is_default) { document.getElementById(`default_level_select`).classList.add("active");} else { var currentEl=document.getElementById(`select-${current_level.level_id}`);if(currentEl !==null) { currentEl.↵ classList.add("active");} } });`
- document `getElementById ('controls').addEventListener('click'`
- window `close ()`
- window `addEventListener ("load", function() { load_on_off_switch();})`
- document `getElementsByClassName ("slider")[0].addEventListener("click"`
- function `load_on_off_switch ()`
 - *Load switch state from storage for current site.*
- function `control_whitelist ()`
 - *Event handler for On/off switch.*

Variables

- const `fadeOut = "0.3"`
- const `fadeIn = "1.0"`
- var `myAddon = new URL(browser.runtime.getURL ('./'))`
- var `url`
- var `queryInfo`
- function `e`

2.13.1 Function Documentation

2.13.1.1 addEventListener()

```

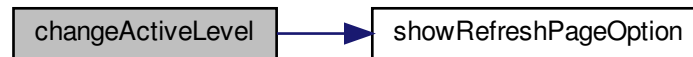
window addEventListener (
    "load" ,
    function() { load_on_off_switch();} )

```

2.13.1.2 `changeActiveLevel()`

```
function changeActiveLevel (
    activeEl )
```

Visually highlights the active level. Here is the call graph for this function:



2.13.1.3 `close()`

```
window close ( )
```

2.13.1.4 `control_whitelist()`

```
function control_whitelist ( )
```

Event handler for On/off switch.

Here is the call graph for this function:



2.13.1.5 `getElementById()`

```
document getElementById (
    'controls' )
```

2.13.1.6 `getElementsByClassName()`

```
document.getElementsByClassName (
    "slider" )
```

2.13.1.7 `if()`

```
if (
    (typeof browser) === "undefined" )
```

2.13.1.8 `load_on_off_switch()`

```
function load_on_off_switch ( )
```

Load switch state from storage for current site.

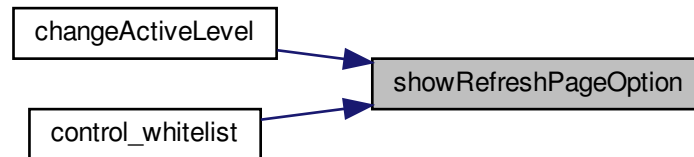
2.13.1.9 `query()`

```
browser.tabs.query (
    queryInfo ,
    function(tabs) { let tab=tabs[0];url=new URL(tab.url);url.hostname=url.hostname.↵
replace(/^www\.\/, '');if(url.hostname=="||url.hostname==myAddon.hostname||url.hostname=="newtab")
{ document.getElementById("current_site_level_settings").style.opacity=fadeOut;return;} else {
document.getElementById('current-site').innerHTML=url.hostname;} var port_to_background=browser.↵
runtime.connect({name:"port_from_popup"});var current_level={ level_id:"?" };port_to_background.onMessage.addL
{ current_level=msg;var selectEl=document.getElementById("level-select");selectEl.inner↵
HTML=`< span class="level level_control" id="default_level_select" title="Set the default
level">Default</span >`;document.getElementById(`default_level_select`).addEventListener("click",
function() { delete domains[url.hostname];saveDomainLevels();changeActiveLevel(this);current_level=default_lev
level_id in levels) { let level=levels[level_id];selectEl.appendChild(document.createRange().create↵
ContextualFragment(`< span class="level level_control" id="select-${level.level_id}" title="${level.↵
level_text}">${escape(level.level_id)}</span >`);document.getElementById(`select-${level.↵
level_id}`).addEventListener("click", function() { domains[url.hostname]={ level_id:level.↵
level_id, };saveDomainLevels();changeActiveLevel(this);current_level=level;});} if(current_↵
level.is_default) { document.getElementById(`default_level_select`).classList.add("active");}
else { var currentEl=document.getElementById(`select-${current_level.level_id}`);if(currentEl
!=null) { currentEl.classList.add("active");} } }); } )
```

2.13.1.10 showRefreshPageOption()

```
function showRefreshPageOption ( )
```

Enable the refresh page option. Here is the caller graph for this function:



2.13.2 Variable Documentation

2.13.2.1 e

```
function e
```

Initial value:

```
{  
  browser.runtime.openOptionsPage()  
}
```

2.13.2.2 fadeIn

```
const fadeIn = "1.0"
```

2.13.2.3 fadeOut

```
const fadeOut = "0.3"
```

2.13.2.4 myAddon

```
var myAddon = new URL(browser.runtime.getURL ('./'))
```

2.13.2.5 queryInfo

```
var queryInfo
```

Initial value:

```
= {  
  active: true,  
  currentWindow: true  
}
```

2.13.2.6 url

```
var url
```

2.14 common/update.js File Reference

Functions

- `if` (`(typeof browser)=== "undefined"`)
- function `installUpdate` ()
- browser runtime onInstalled `addListener` (`installUpdate`)

2.14.1 Function Documentation

2.14.1.1 `addListener()`

```
browser runtime onInstalled addListener (  
  installUpdate )
```

2.14.1.2 `if()`

```
if (  
  (typeof browser) == "undefined" )
```


2.14.1.3 installUpdate()

```
function installUpdate ( )
```

0.3 storage { **default**: 2, // Default protection level version: 2.1, // The version of this storage custom_levels: {}, // associative array of custom level (key, its id => object) {level_id: short string used for example on the badge level←_text: Short level description level_description: Full level description ... wrapping_params (key-value pairs), see wrapping_groups for the list of params and supported values } domains: {}, // associative array of levels associated with specific domains (key, the domain => object) {level_id: short string of the level in use } whitelistedHosts: {} // associative array of hosts that are removed from http protection control (hostname => boolean) requestShieldOn: {} // Boolean, if it's TRUE or undefined, the http request protection is turned on, if it's FALSE, the protection si turned off

2.15 common/url.js File Reference

Functions

- function [extractSubDomains](#) (thisDomain)

2.15.1 Function Documentation

2.15.1.1 extractSubDomains()

```
function extractSubDomains (
    thisDomain )
```

Get all sub domains for a domain.

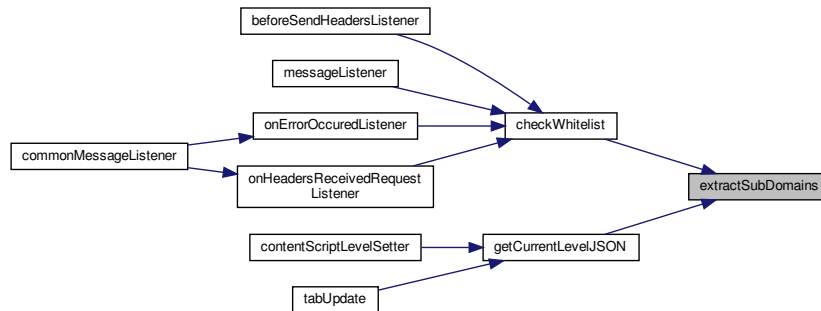
Parameters

<i>thisDomain</i>	Domain name string, e.g. www.fit.vutbr.cz
-------------------	---

Returns

List of strings representing all subdomains, TLD excluded. The list starts with the most generic domain and continues with the more and more specific domains. For example, `www.fit.vutbr.cz` -> `["vutbr.cz", "fit.vutbr.cz", "www.fit.vutbr.cz"]`

Here is the caller graph for this function:



2.16 common/wrapping.js File Reference

Functions

- function [add_wrappers](#) (`wrappers`)
- function [noise_function](#) (`numberToChange`, `precision`)

Variables

- var [build_wrapping_code](#) = {}
- var [rounding_function](#)
- var [noise_function](#) = `let lastValue = 0`

2.16.1 Function Documentation

2.16.1.1 add_wrappers()

```
function add_wrappers (
    wrappers )
```

Adds a list of wrapping objects to the `build_wrapping_code`.

This function is called from each wrapper in its file.

2.16.1.2 noise_function()

```
function noise_function (
    numberToChange,
    precision )
```

2.16.2 Variable Documentation

2.16.2.1 build_wrapping_code

```
var build_wrapping_code = {}
```

The object carrying all the wrappers

2.16.2.2 noise_function

```
var noise_function = `let lastValue = 0
```

Function to be used by wrapped code for adding randomized noise

2.16.2.3 rounding_function

```
var rounding_function
```

Initial value:

```
= `function rounding_function(numberToRound, precision) {
  return numberToRound - (numberToRound % Math.pow(10, Math.max(3 - precision)));
}`
```

Function to be used by wrapped code used for rounding

2.17 common/wrappingS-AJAX.js File Reference

2.18 common/wrappingS-BATTERY-CR.js File Reference

Functions

- function [add_wrappers](#) ([wrappers](#))

2.18.1 Function Documentation

2.18.1.1 add_wrappers()

```
function add_wrappers (
    wrappers )
```

Adds a list of wrapping objects to the build_wrapping_code.

This function is called from each wrapper in its file.

2.19 common/wrappingS-DM.js File Reference

2.20 common/wrappingS-ECMA-ARRAY.js File Reference

Functions

- function [packIEEE754](#) (v, ebits, fbits)

This function was adopted from <https://github.com/inexorabletash/polyfill/blob/master/typedarray.js> under MIT licence.
- function [unpackIEEE754](#) (bytes, ebits, fbits)

This function was adopted from <https://github.com/inexorabletash/polyfill/blob/master/typedarray.js> under MIT licence.
- function [unpackF64](#) (b)

Function was adopted from <https://github.com/inexorabletash/polyfill/blob/master/typedarray.js> under MIT licence.
- function [packF64](#) (v)

Function was adopted from <https://github.com/inexorabletash/polyfill/blob/master/typedarray.js> under MIT licence.
- function [unpackF32](#) (b)

Function was adopted from <https://github.com/inexorabletash/polyfill/blob/master/typedarray.js> under MIT licence.
- function [packF32](#) (v)

Function was adopted from <https://github.com/inexorabletash/polyfill/blob/master/typedarray.js> under MIT licence.
- function [constructDecorator](#) (wrapped)
- function [offsetDecorator](#) (wrapped, type, proxyRef, offsetF)
- function [redefineNewArrayFunctions](#) (target, offsetF)
- function [redefineNewArrayConstructors](#) (target)
- function [getBytesDecorator](#) (wrapped, offsetF, name, doMapping)
- function [setBytesDecorator](#) (wrapped, offsetF, name, doMapping)
- function [getFloatDecorator](#) (wrapped, name, doMapping)
- function [setFloatDecorator](#) (wrapped, name, doMapping)
- function [getBigIntDecorator](#) (wrapped, doMapping)
- function [setBigIntDecorator](#) (wrapped, doMapping)
- function [redefineDataViewFunctions](#) (target, offsetF, doMapping)
- function [if](#) (typeof target==='object' &&target !==null)
- [if](#) (doMapping)
- [for](#) (let i=0;i<_data['length'];i++)
- [for](#) (let p of typedTypes)
- [add_wrappers](#) (wrappers)

Variables

- var `proxyHandler`
Default proxy handler for Typed Arrays.
- `_data` = new originalF(...arguments)
- var `offsetF`
- let `_target` = target
- var `proxy` = new newProxy(`_data`, `proxyHandler`)
- let `j`
- var `wrappers`
- let `DEFAULT_TYPED_ARRAY_WRAPPER`
- var `typedTypes` = ['Uint8Array', 'Int8Array', 'Uint8ClampedArray', 'Int16Array', 'Uint16Array', 'Int32Array', 'Uint32Array', 'Float32Array', 'Float64Array']

2.20.1 Function Documentation

2.20.1.1 add_wrappers()

```
add_wrappers (
    wrappers )
```

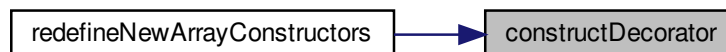
Adds a list of wrapping objects to the `build_wrapping_code`.

This function is called from each wrapper in its file.

2.20.1.2 constructDecorator()

```
function constructDecorator (
    wrapped )
```

Here is the caller graph for this function:



2.20.1.3 for() [1/2]

```
for ( )
```

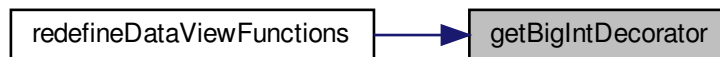
2.20.1.4 for() [2/2]

```
for (
    let p of typedTypes )
```

2.20.1.5 getBigIntDecorator()

```
function getBigIntDecorator (
    wrapped,
    doMapping )
```

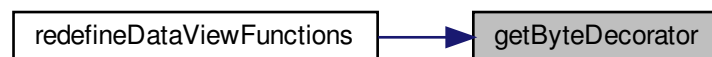
Here is the caller graph for this function:



2.20.1.6 getByteDecorator()

```
function getByteDecorator (
    wrapped,
    offsetF,
    name,
    doMapping )
```

Here is the caller graph for this function:



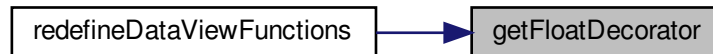
2.20.1.7 getFloatDecorator()

```
function getFloatDecorator (
    wrapped,
    name,
    doMapping )
```

Here is the call graph for this function:



Here is the caller graph for this function:



2.20.1.8 if() [1/2]

```
if (
    doMapping )
```

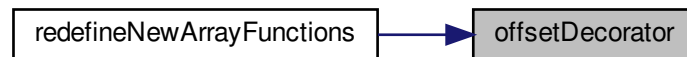
2.20.1.9 if() [2/2]

```
function if (
    typeof target == 'object' && target !== null )
```

2.20.1.10 offsetDecorator()

```
function offsetDecorator (
    wrapped,
    type,
    proxyRef,
    offsetF )
```

Here is the caller graph for this function:



2.20.1.11 packF32()

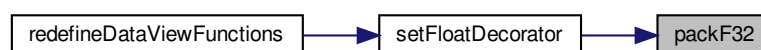
```
function packF32 (
    v )
```

Function was adopted from <https://github.com/inexorabletash/polyfill/blob/master/typedarray.js> under MIT licence.

Here is the call graph for this function:



Here is the caller graph for this function:



2.20.1.12 packF64()

```
function packF64 (
    v )
```

Function was adopted from <https://github.com/inexorabletash/polyfill/blob/master/typedarray.js> under MIT licence.

Here is the call graph for this function:



Here is the caller graph for this function:

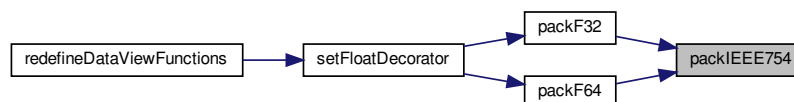


2.20.1.13 packIEEE754()

```
function packIEEE754 (
    v,
    ebits,
    fbits )
```

This function was adopted from <https://github.com/inexorabletash/polyfill/blob/master/typedarray.js> under MIT licence.

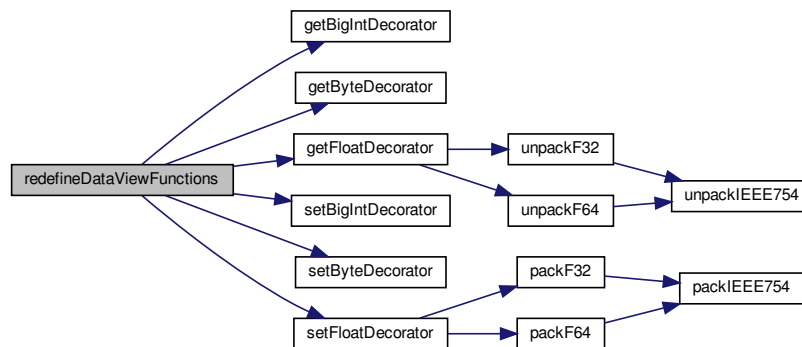
Here is the caller graph for this function:



2.20.1.14 `redefineDataViewFunctions()`

```
function redefineDataViewFunctions (
    target,
    offsetF,
    doMapping )
```

Here is the call graph for this function:



2.20.1.15 `redefineNewArrayConstructors()`

```
function redefineNewArrayConstructors (
    target )
```

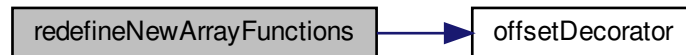
Here is the call graph for this function:



2.20.1.16 `redefineNewArrayFunctions()`

```
function redefineNewArrayFunctions (
    target,
    offsetF )
```

Here is the call graph for this function:



2.20.1.17 `setBigIntDecorator()`

```
function setBigIntDecorator (
    wrapped,
    doMapping )
```

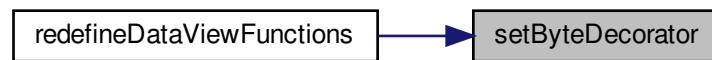
Here is the caller graph for this function:



2.20.1.18 `setByteDecorator()`

```
function setByteDecorator (
    wrapped,
    offsetF,
    name,
    doMapping )
```

Here is the caller graph for this function:



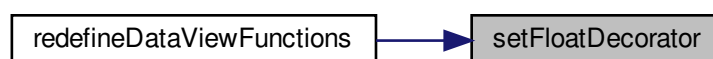
2.20.1.19 setFloatDecorator()

```
function setFloatDecorator (
    wrapped,
    name,
    doMapping )
```

Here is the call graph for this function:



Here is the caller graph for this function:



2.20.1.20 unpackF32()

```
function unpackF32 (  
    b )
```

Function was adopted from <https://github.com/inexorabletash/polyfill/blob/master/typedarray.
js> under MIT licence.

Here is the call graph for this function:



Here is the caller graph for this function:



2.20.1.21 unpackF64()

```
function unpackF64 (  
    b )
```

Function was adopted from <https://github.com/inexorabletash/polyfill/blob/master/typedarray.
js> under MIT licence.

Here is the call graph for this function:



Here is the caller graph for this function:

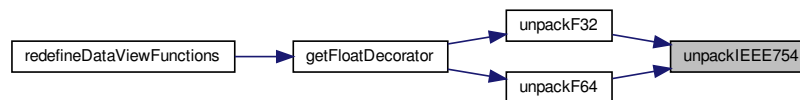


2.20.1.22 unpackIEEE754()

```
function unpackIEEE754 (  
    bytes,  
    ebits,  
    fbits )
```

This function was adopted from <https://github.com/inexorabletash/polyfill/blob/master/typedarray.js> under MIT licence.

Here is the caller graph for this function:



2.20.2 Variable Documentation

2.20.2.1 _data

```
_data = new originalF(...arguments)
```

2.20.2.2 _target

```
let _target = target
```

2.20.2.3 DEFAULT_TYPED_ARRAY_WRAPPER

```
let DEFAULT_TYPED_ARRAY_WRAPPER
```

2.20.2.4 j

```
let j
```

2.20.2.5 offsetF

```
var offsetF
```

Initial value:

```
= function(x) {
  return x;
}
```

2.20.2.6 proxy

```
return proxy = new newProxy(_data, ${proxyHandler})
```

2.20.2.7 proxyHandler

```
var proxyHandler
```

Initial value:

```
= `{
  get(target, key, receiver) {
    var random_idx = Math.floor(Math.random() * target['length']);

    var rand_val = target[random_idx];
    let proto_keys = ['buffer', 'byteLength', 'byteOffset', 'length'];
    if (proto_keys.indexOf(key) >= 0) {
      return target[key];
    }

    if (typeof key !== 'symbol' && Number(key) >= 0 && Number(key) < target.length) {
      key = offsetF(key)
    }
    let value = Reflect.get(...arguments);
    return typeof value == 'function' ? value.bind(target) : value;
  },
  set(target, key, value) {
    var random_idx = Math.floor(Math.random() * (target['length']));

    var rand_val = target[random_idx];
    rand_val = rand_val;
    if (typeof key !== 'symbol' && Number(key) >= 0 && Number(key) < target.length) {
      key = offsetF(key)
    }
    return Reflect.set(...arguments);
  }
}`
```

Default proxy handler for Typed Arrays.

2.20.2.8 typedTypes

```
var typedTypes = ['Uint8Array', 'Int8Array', 'Uint8ClampedArray', 'Int16Array', 'Uint16Array',  
'Int32Array', 'Uint32Array', 'Float32Array', 'Float64Array']
```

2.20.2.9 wrappers

```
var wrappers
```

2.21 common/wrappingS-ECMA-DATE.js File Reference

2.22 common/wrappingS-ECMA-SHARED.js File Reference

Functions

- [add_wrappers](#) ([wrappers](#))

Variables

- var [proxyHandler](#)
- function let [_data](#) = new originalF(target)
- var [proxy](#) = new Proxy([_data](#), \${[proxyHandler](#)})
- var [wrappers](#)

2.22.1 Function Documentation

2.22.1.1 add_wrappers()

```
add_wrappers (  
    wrappers )
```

Adds a list of wrapping objects to the build_wrapping_code.

This function is called from each wrapper in its file.

2.22.2 Variable Documentation

2.22.2.1 `_data`

```
function let _data = new originalF(target)
```

2.22.2.2 `proxy`

```
return proxy = new Proxy(_data, ${proxyHandler})
```

2.22.2.3 `proxyHandler`

```
var proxyHandler
```

Initial value:

```
= `{
  get(target, key, receiver) {
    let j;
    let slow = Math.floor(Math.random() * 10000)
    for (let i = 0; i < slow; i++) {
      j = i;
      i = j + 1;
    }
    let value = target[key];
    return typeof value == 'function' ? value.bind(target) : value;
  },
  set(target, key, value) {
    let j;
    let slow = Math.floor(Math.random() * 10000);
    for (let i = 0; i < slow; i++) {
      j = i;
      i = j + 1;
    }
    return Reflect.set(...arguments);
  }
}`
```

2.22.2.4 `wrappers`

```
var wrappers
```

Initial value:

```
= [
  {
    parent_object: "window",
    parent_object_property: "SharedArrayBuffer",
    original_function: "window.SharedArrayBuffer",
    wrapped_objects: [],
    helping_code: `
      if (window.SharedArrayBuffer === undefined) {
        return;
      }
      let forbid = args[0];
    `,
    wrapping_function_args: `target`,
    wrapping_function_body: wrappingFunctionBody,
    post_replacement_code: `
      if (forbid) {
        delete(window.SharedArrayBuffer);
      }
    `
  }
]
```

2.23 common/wrappingS-H-C.js File Reference

2.24 common/wrappingS-HRT.js File Reference

Functions

- function [add_wrappers](#) ([wrappers](#))

2.24.1 Function Documentation

2.24.1.1 add_wrappers()

```
function add_wrappers (  
    wrappers )
```

Adds a list of wrapping objects to the build_wrapping_code.

This function is called from each wrapper in its file.

2.25 common/wrappingS-HTML-LS.js File Reference

Functions

- function [executeEach](#) (arr, fun)
- function [callErrorListener](#) (err)
- function [addEventListener](#) (type, fun)
 - If the browser regained connectivity - came online.*
- function [removeEventListener](#) (type, fun)
- function [postError](#) (err)
- function [runPostMessage](#) (msg, transfer)
- function [postMessage](#) (msg, transfer)
- function [workerPostMessage](#) (msg)
- function [workerAddEventListener](#) (type, fun)
- [xhr open](#) ('GET', path)
- [xhr send](#) ()

Variables

- var `errorListeners` = []
- var `workerMessageListeners` = []
- var `workerErrorListeners` = []
- var `postMessageListeners` = []
- var `terminated` = false
- var `script`
- var `workerSelf`
- var `api` = this
- var `xhr` = new XMLHttpRequest()
- `xhr onreadystatechange`
- `api postMessage` = postMessage
- `api addEventListener` = addEventListener
If the browser lost connectivity - gone offline.
- `api removeEventListener` = removeEventListener
- `api terminate` = terminate
- var `slowBody`
- let `_old` = `_data.postMessage`
- var `wrappers`

2.25.1 Function Documentation

2.25.1.1 addEventListener()

```
function addEventListener (
    "online" ,
    function() { browser.webRequest.onErrorOccurred.addListener(onErrorOccuredListener,
{urls:["<all_urls>"]});} )
```

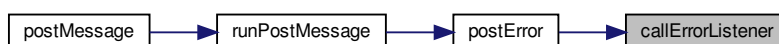
If the browser regained connectivity - came online.

If the browser lost connectivity - gone offline.

2.25.1.2 callErrorListener()

```
function callErrorListener (
    err )
```

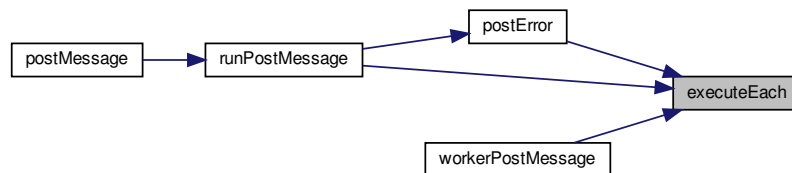
Here is the caller graph for this function:



2.25.1.3 executeEach()

```
function executeEach (  
    arr,  
    fun )
```

Here is the caller graph for this function:



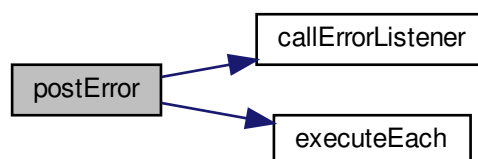
2.25.1.4 open()

```
xhr open (  
    'GET' ,  
    path )
```

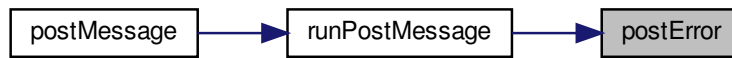
2.25.1.5 postError()

```
function postError (  
    err )
```

Here is the call graph for this function:



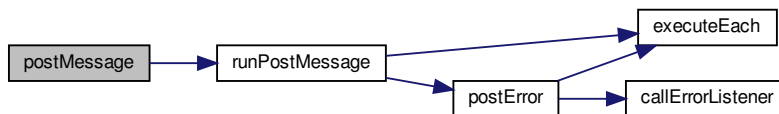
Here is the caller graph for this function:



2.25.1.6 `postMessage()`

```
function postMessage (
    msg,
    transfer )
```

Here is the call graph for this function:



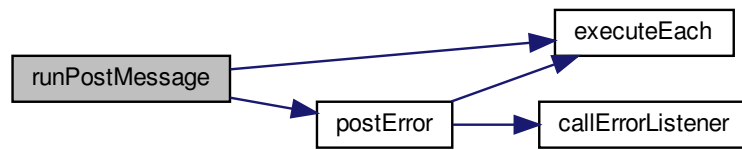
2.25.1.7 `removeEventListener()`

```
function removeEventListener (
    type,
    fun )
```

2.25.1.8 `runPostMessage()`

```
function runPostMessage (
    msg,
    transfer )
```

Here is the call graph for this function:



Here is the caller graph for this function:



2.25.1.9 send()

```
xhr send ( )
```

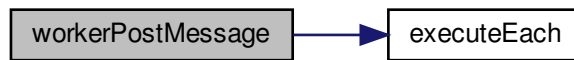
2.25.1.10 workerAddEventListener()

```
function workerAddEventListener (
    type,
    fun )
```

2.25.1.11 workerPostMessage()

```
function workerPostMessage (  
    msg )
```

Here is the call graph for this function:



2.25.2 Variable Documentation

2.25.2.1 _old

```
let _old = _data.postMessage
```

2.25.2.2 addEventListener

```
window addEventListener = addEventListener
```

If the browser lost connectivity - gone offline.

2.25.2.3 api

```
return api = this
```

2.25.2.4 errorListeners

```
var errorListeners = []
```

2.25.2.5 onreadystatechange

`xhr` onreadystatechange

Initial value:

```
= function () {
  if (xhr.readyState === 4) {
    if (xhr.status >= 200 && xhr.status < 400) {
      script = xhr.responseText;
      workerSelf = {
        postMessage: workerPostMessage,
        addEventListener: workerAddEventListener,
        close: terminate
      };
      doEval(workerSelf, script);
      var currentListeners = postMessageListeners;
      postMessageListeners = [];
      for (var i = 0; i < currentListeners.length; i++) {
        runPostMessage(currentListeners[i].msg, currentListeners[i].transfer);
      }
    } else {
      postError(new Error('cannot find script ' + path));
    }
  }
}
```

2.25.2.6 postMessage

`_data` postMessage = postMessage

2.25.2.7 postMessageListeners

`var` postMessageListeners = []

2.25.2.8 removeEventListener

`api` removeEventListener = removeEventListener

2.25.2.9 script

`var` script

2.25.2.10 slowBody

```
var slowBody
```

Initial value:

```
= `
  let _data = new originalF(path)
```

2.25.2.11 terminate

```
function terminate = terminate
```

2.25.2.12 terminated

```
var terminated = false
```

2.25.2.13 workerErrorListeners

```
var workerErrorListeners = []
```

2.25.2.14 workerMessageListeners

```
var workerMessageListeners = []
```

2.25.2.15 workerSelf

```
var workerSelf
```

2.25.2.16 wrappers

```
var wrappers
```

2.25.2.17 xhr

```
var xhr = new XMLHttpRequest()
```

2.26 common/wrappingS-PT2.js File Reference

Functions

- [if](#) (doNoise===true)
- [for](#) (measure of measures)

Variables

- var [common_function_body](#)
- [func](#) = [rounding_function](#)
- var [ret](#) = []
- var [wrappers](#)

2.26.1 Function Documentation

2.26.1.1 for()

```
for (
    measure of measures )
```

2.26.1.2 if()

```
if (
    doNoise == true )
```

2.26.2 Variable Documentation

2.26.2.1 common_function_body

```
var common_function_body
```

Initial value:

```
= `
    var measures = origFunc.call(this, ...args)
```

2.26.2.2 func

```
func = rounding_function
```

2.26.2.3 ret

```
return ret = []
```

2.26.2.4 wrappers

```
var wrappers
```

2.27 firefox/http_shield_firefox.js File Reference

Functions

- browser runtime onMessage [addListener](#) ([messageListener](#))
- function async [beforeSendHeadersListener](#) ([requestDetail](#))
- function [messageListener](#) ([message](#), [sender](#), [sendResponse](#))

2.27.1 Function Documentation

2.27.1.1 addListener()

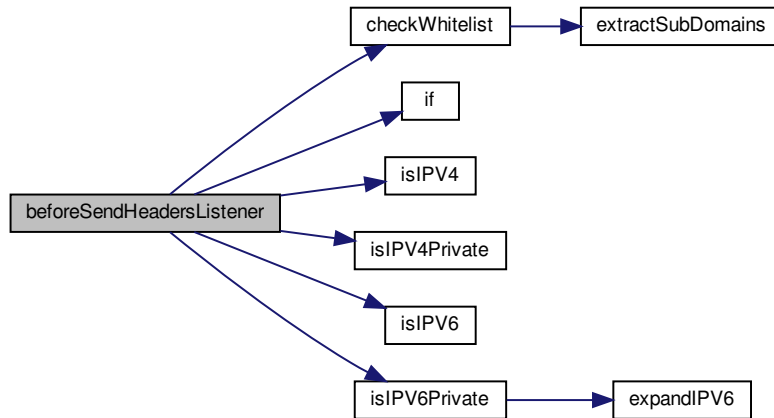
```
browser runtime onMessage addListener (  
    messageListener )
```

Implementation of HTTP webRequest shield, file: [http_shield_firefox.js](#) Contains Firefox specific functions Event handlers for webRequest API, notifications and messaging webRequest event listener Listens to onBeforeSendHeaders event, receives detail of HTTP request in requestDetail Catches the request, analyzes it's origin and target URLs and blocks it/permits it based on their IP addresses. Requests coming from public IP ranges targeting the private IPs are blocked by default. Others are permitted by default.

2.27.1.2 beforeSendHeadersListener()

```
function async beforeSendHeadersListener (
    requestDetail )
```

Here is the call graph for this function:



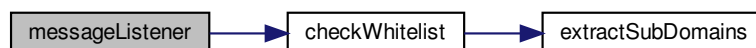
Here is the caller graph for this function:



2.27.1.3 messageListener()

```
function messageListener (
    message,
    sender,
    sendResponse )
```

Event listener hooked up to `webExtensions.onMessage` event. Receives full message in `message`, sender of the message in `sender`, function for sending response in `sendResponse`. Does appropriate action based on message. Here is the call graph for this function:



2.28 firefox/level_cache.js File Reference

Functions

- function `contentScriptLevelSetter` (message)
- browser runtime onMessage `addListener` (`contentScriptLevelSetter`)

Variables

- var `domains_bug1267027` = {}

2.28.1 Function Documentation

2.28.1.1 `addListener()`

```
browser runtime onMessage addListener (  
    contentScriptLevelSetter )
```

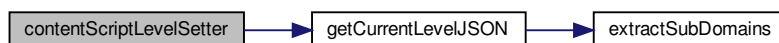
2.28.1.2 `contentScriptLevelSetter()`

```
function contentScriptLevelSetter (  
    message )
```

Messaging with content script.

@message The message from consent script.

Returns the promise with the message returned to the content script. Here is the call graph for this function:



2.28.2 Variable Documentation

2.28.2.1 domains_bug1267027

```
var domains_bug1267027 = {}
```

Keep list of domains that are known (not) to be affected by the Firefox bug 1267027.

2.29 chrome/level_cache.js File Reference

Functions

- function [contentScriptLevelSetter](#) (message, sender, sendResponse)
- browser runtime onMessage [addListener](#) ([contentScriptLevelSetter](#))

2.29.1 Function Documentation

2.29.1.1 addListener()

```
browser runtime onMessage addListener (  
    contentScriptLevelSetter )
```

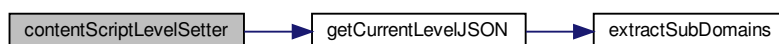
2.29.1.2 contentScriptLevelSetter()

```
function contentScriptLevelSetter (  
    message,  
    sender,  
    sendResponse )
```

Messaging with content script.

@message The message from consent script.

Sends back the wrapping code. Here is the call graph for this function:



2.30 firefox/manifest.json File Reference

2.31 chrome/manifest.json File Reference

Index

- [_data](#)
 - [wrappingS-ECMA-ARRAY.js, 61](#)
 - [wrappingS-ECMA-SHARED.js, 63](#)
 - [_old](#)
 - [wrappingS-HTML-LS.js, 70](#)
 - [_target](#)
 - [wrappingS-ECMA-ARRAY.js, 61](#)
- [add_to_whitelist](#)
 - [options.js, 34](#)
- [add_wrappers](#)
 - [wrapping.js, 49](#)
 - [wrappingS-BATTERY-CR.js, 50](#)
 - [wrappingS-ECMA-ARRAY.js, 52](#)
 - [wrappingS-ECMA-SHARED.js, 63](#)
 - [wrappingS-HRT.js, 65](#)
- [addEventListener](#)
 - [http_shield_chrome.js, 4](#)
 - [options.js, 34](#)
 - [popup.js, 43](#)
 - [wrappingS-HTML-LS.js, 66, 70](#)
- [addListener](#)
 - [background.js, 9](#)
 - [http_shield_common.js, 19](#)
 - [http_shield_firefox.js, 74](#)
 - [level_cache.js, 76, 77](#)
 - [levels.js, 28](#)
 - [update.js, 47](#)
- [api](#)
 - [wrappingS-HTML-LS.js, 70](#)
- [background.js](#)
 - [addListener, 9](#)
 - [connected, 9](#)
 - [current_level, 11](#)
 - [if, 9](#)
 - [queryInfo, 11](#)
 - [tab_levels, 11](#)
 - [tabActivate, 10](#)
 - [tabUpdate, 10](#)
 - [updateBadge, 10](#)
- [beforeSendHeadersListener](#)
 - [http_shield_chrome.js, 4](#)
 - [http_shield_firefox.js, 74](#)
- [blockedHosts](#)
 - [http_shield_chrome.js, 7](#)
- [browser.js](#)
 - [if, 12](#)
- [build_code](#)
 - [code_builders.js, 14](#)
- [build_code_ffbug](#)
 - [ffbug1267027.js, 15](#)
- [build_wrapping_code](#)
 - [wrapping.js, 50](#)
- [callErrorListener](#)
 - [wrappingS-HTML-LS.js, 66](#)
- [changeActiveLevel](#)
 - [popup.js, 43](#)
- [changedLevels](#)
 - [levels.js, 28](#)
- [checkWhitelist](#)
 - [http_shield_common.js, 19](#)
- [chrome/http_shield_chrome.js, 3](#)
- [chrome/level_cache.js, 77](#)
- [chrome/manifest.json, 77](#)
- [chromeErrorString](#)
 - [http_shield_chrome.js, 7](#)
- [close](#)
 - [popup.js, 44](#)
- [code_builders.js](#)
 - [build_code, 14](#)
 - [define_page_context_function, 12](#)
 - [enclose_wrapping, 12](#)
 - [enclose_wrapping2, 13](#)
 - [generate_assign_function_code, 14](#)
 - [generate_object_properties, 14](#)
 - [wrap_code, 14](#)
- [common/background.js, 9](#)
- [common/browser.js, 11](#)
- [common/code_builders.js, 12](#)
- [common/document_start.js, 15](#)
- [common/ffbug1267027.js, 15](#)
- [common/helpers.js, 18](#)
- [common/http_shield_common.js, 19](#)
- [common/inject.js, 26](#)
- [common/levels.js, 27](#)
- [common/options.js, 34](#)
- [common/options_domains.js, 42](#)
- [common/popup.js, 43](#)
- [common/update.js, 47](#)
- [common/url.js, 48](#)
- [common/wrapping.js, 49](#)
- [common/wrappingS-AJAX.js, 50](#)
- [common/wrappingS-BATTERY-CR.js, 50](#)
- [common/wrappingS-DM.js, 51](#)
- [common/wrappingS-ECMA-ARRAY.js, 51](#)
- [common/wrappingS-ECMA-DATE.js, 63](#)
- [common/wrappingS-ECMA-SHARED.js, 63](#)
- [common/wrappingS-H-C.js, 65](#)

- common/wrappingS-HRT.js, 65
- common/wrappingS-HTML-LS.js, 65
- common/wrappingS-PT2.js, 73
- common_function_body
 - wrappingS-PT2.js, 73
- commonMessageListener
 - http_shield_common.js, 20
- connected
 - background.js, 9
- constructDecorator
 - wrappingS-ECMA-ARRAY.js, 52
- contentScriptLevelSetter
 - level_cache.js, 76, 77
- control_http_request_shield
 - options.js, 35
- control_whitelist
 - popup.js, 44
- CSVToArray
 - http_shield_common.js, 21
- current_level
 - background.js, 11

- default_level
 - levels.js, 31
- DEFAULT_TYPED_ARRAY_WRAPPER
 - wrappingS-ECMA-ARRAY.js, 61
- define_page_context_function
 - code_builders.js, 12
- define_page_context_function_ffbug
 - ffbug1267027.js, 16
- document_start.js
 - sendMessage, 15
- domains
 - levels.js, 31
- domains_bug1267027
 - level_cache.js, 76
- doNotBlockHosts
 - http_shield_common.js, 25

- e
 - popup.js, 46
- edit_level
 - options.js, 35
- enclose_wrapping
 - code_builders.js, 12
- enclose_wrapping2
 - code_builders.js, 13
- errorListeners
 - wrappingS-HTML-LS.js, 70
- errorsAllowed
 - http_shield_chrome.js, 7
- escape
 - helpers.js, 18
 - options_domains.js, 42
- executeEach
 - wrappingS-HTML-LS.js, 66
- expandIPv6
 - http_shield_common.js, 21
- extractSubDomains
 - url.js, 48

- fadeOut
 - popup.js, 46
- ffbug1267027.js
 - build_code_ffbug, 15
 - define_page_context_function_ffbug, 16
 - generate_assign_function_code_ffbug, 16
 - generate_object_properties_ffbug, 17
 - injectScriptElement, 18
 - is_firefox_blocking_scripts, 17
- firefox/http_shield_firefox.js, 74
- firefox/level_cache.js, 76
- firefox/manifest.json, 77
- for
 - wrappingS-ECMA-ARRAY.js, 52
 - wrappingS-PT2.js, 73
- func
 - wrappingS-PT2.js, 73

- generate_assign_function_code
 - code_builders.js, 14
- generate_assign_function_code_ffbug
 - ffbug1267027.js, 16
- generate_object_properties
 - code_builders.js, 14
- generate_object_properties_ffbug
 - ffbug1267027.js, 17
- get
 - http_shield_common.js, 22
 - levels.js, 28
- getBigIntDecorator
 - wrappingS-ECMA-ARRAY.js, 53
- getByteDecorator
 - wrappingS-ECMA-ARRAY.js, 53
- getCurrentLevelJSON
 - levels.js, 28
- getElementById
 - options.js, 36
 - popup.js, 44
- getElementsByClassName
 - options.js, 36
 - popup.js, 44
- getFloatDecorator
 - wrappingS-ECMA-ARRAY.js, 53
- getSelectValues
 - options.js, 36

- helpers.js
 - escape, 18
- hostStatistics
 - http_shield_chrome.js, 8
- html_element_value_source
 - options.js, 41
- http_shield_chrome.js
 - addEventListener, 4
 - beforeSendHeadersListener, 4

- blockedHosts, [7](#)
- chromeErrorString, [7](#)
- errorsAllowed, [7](#)
- hostStatistics, [8](#)
- httpClientErrorsAllowed, [8](#)
- httpErrorList, [8](#)
- insertHostInStats, [4](#)
- messageListener, [5](#)
- notifyBlockedHost, [5](#)
- onErrorOccuredListener, [6](#)
- onHeadersReceivedRequestListener, [6](#)
- uniqueErrorHostsLimit, [8](#)
- uniqueErrorHostsRatio, [8](#)
- http_shield_common.js
 - addListener, [19](#)
 - checkWhitelist, [19](#)
 - commonMessageListener, [20](#)
 - CSVToArray, [21](#)
 - doNotBlockHosts, [25](#)
 - expandIPV6, [21](#)
 - get, [22](#)
 - if, [22](#)
 - isIPV4, [22](#)
 - isIPV4Private, [22](#)
 - isIPV6, [23](#)
 - isIPV6Private, [23](#)
 - localIPV4DNSZones, [25](#)
 - localIPV6DNSZones, [26](#)
 - notifyBlockedRequest, [24](#)
 - parseCSV, [24](#)
 - readFile, [25, 26](#)
- http_shield_firefox.js
 - addListener, [74](#)
 - beforeSendHeadersListener, [74](#)
 - messageListener, [75](#)
- httpClientErrorsAllowed
 - http_shield_chrome.js, [8](#)
- httpErrorList
 - http_shield_chrome.js, [8](#)
- if
 - background.js, [9](#)
 - browser.js, [12](#)
 - http_shield_common.js, [22](#)
 - levels.js, [29](#)
 - options.js, [36](#)
 - options_domains.js, [42](#)
 - popup.js, [45](#)
 - update.js, [47](#)
 - wrappingS-ECMA-ARRAY.js, [54](#)
 - wrappingS-PT2.js, [73](#)
- init_levels
 - levels.js, [29](#)
- inject.js
 - injectScript, [26](#)
- injectScript
 - inject.js, [26](#)
- injectScriptElement
 - ffbug1267027.js, [18](#)
- insert_levels
 - options.js, [37](#)
- insertHostInStats
 - http_shield_chrome.js, [4](#)
- installUpdate
 - update.js, [47](#)
- is_firefox_blocking_scripts
 - ffbug1267027.js, [17](#)
- isIPV4
 - http_shield_common.js, [22](#)
- isIPV4Private
 - http_shield_common.js, [22](#)
- isIPV6
 - http_shield_common.js, [23](#)
- isIPV6Private
 - http_shield_common.js, [23](#)
- j
 - wrappingS-ECMA-ARRAY.js, [62](#)
- L0
 - levels.js, [31](#)
- L1
 - levels.js, [31](#)
- L2
 - levels.js, [31](#)
- L3
 - levels.js, [31](#)
- level_0
 - levels.js, [31](#)
- level_1
 - levels.js, [32](#)
- level_2
 - levels.js, [32](#)
- level_3
 - levels.js, [32](#)
- level_cache.js
 - addListener, [76, 77](#)
 - contentScriptLevelSetter, [76, 77](#)
 - domains_bug1267027, [76](#)
- levels
 - levels.js, [33](#)
- levels.js
 - addListener, [28](#)
 - changedLevels, [28](#)
 - default_level, [31](#)
 - domains, [31](#)
 - get, [28](#)
 - getCurrentLevelJSON, [28](#)
 - if, [29](#)
 - init_levels, [29](#)
 - L0, [31](#)
 - L1, [31](#)
 - L2, [31](#)
 - L3, [31](#)
 - level_0, [31](#)
 - level_1, [32](#)
 - level_2, [32](#)
 - level_3, [32](#)

- levels, 33
- levels_initialised, 33
- levels_updated_callbacks, 33
- saveDomainLevels, 29
- setDefaultLevel, 30
- updateLevels, 30
- wrapped_codes, 33
- wrapping_groups, 33
- levels_initialised
 - levels.js, 33
- levels_updated_callbacks
 - levels.js, 33
- load_on_off_switch
 - options.js, 37
 - popup.js, 45
- loadWhitelist
 - options.js, 37
- localIPv4DNSZones
 - http_shield_common.js, 25
- localIPv6DNSZones
 - http_shield_common.js, 26
- messageListener
 - http_shield_chrome.js, 5
 - http_shield_firefox.js, 75
- myAddon
 - popup.js, 46
- noise_function
 - wrapping.js, 49, 50
- notifyBlockedHost
 - http_shield_chrome.js, 5
- notifyBlockedRequest
 - http_shield_common.js, 24
- offsetDecorator
 - wrappingS-ECMA-ARRAY.js, 54
- offsetF
 - wrappingS-ECMA-ARRAY.js, 62
- onErrorOccuredListener
 - http_shield_chrome.js, 6
- onHeadersReceivedRequestListener
 - http_shield_chrome.js, 6
- onreadystatechange
 - wrappingS-HTML-LS.js, 70
- open
 - wrappingS-HTML-LS.js, 67
- options.js
 - add_to_whitelist, 34
 - addEventListener, 34
 - control_http_request_shield, 35
 - edit_level, 35
 - getElementById, 36
 - getElementsByClassName, 36
 - getSelectValues, 36
 - html_element_value_source, 41
 - if, 36
 - insert_levels, 37
 - load_on_off_switch, 37
 - loadWhitelist, 37
 - prepare_level_config, 37
 - remove_from_whitelist, 38
 - remove_level, 38
 - restore_level, 39
 - sendMessage, 40
 - show_existing_level, 40
 - update_whitelist, 41
- options_domains.js
 - escape, 42
 - if, 42
- packF32
 - wrappingS-ECMA-ARRAY.js, 55
- packF64
 - wrappingS-ECMA-ARRAY.js, 55
- packIEEE754
 - wrappingS-ECMA-ARRAY.js, 56
- parseCSV
 - http_shield_common.js, 24
- popup.js
 - addEventListener, 43
 - changeActiveLevel, 43
 - close, 44
 - control_whitelist, 44
 - e, 46
 - fadeIn, 46
 - fadeOut, 46
 - getElementById, 44
 - getElementsByClassName, 44
 - if, 45
 - load_on_off_switch, 45
 - myAddon, 46
 - query, 45
 - queryInfo, 46
 - showRefreshPageOption, 45
 - url, 47
- postError
 - wrappingS-HTML-LS.js, 67
- postMessage
 - wrappingS-HTML-LS.js, 68, 71
- postMessageListeners
 - wrappingS-HTML-LS.js, 71
- prepare_level_config
 - options.js, 37
- proxy
 - wrappingS-ECMA-ARRAY.js, 62
 - wrappingS-ECMA-SHARED.js, 64
- proxyHandler
 - wrappingS-ECMA-ARRAY.js, 62
 - wrappingS-ECMA-SHARED.js, 64
- query
 - popup.js, 45
- queryInfo
 - background.js, 11
 - popup.js, 46
- readFile

- http_shield_common.js, 25, 26
- redefineDataViewFunctions
 - wrappingS-ECMA-ARRAY.js, 56
- redefineNewArrayConstructors
 - wrappingS-ECMA-ARRAY.js, 57
- redefineNewArrayFunctions
 - wrappingS-ECMA-ARRAY.js, 57
- remove_from_whitelist
 - options.js, 38
- remove_level
 - options.js, 38
- removeEventListener
 - wrappingS-HTML-LS.js, 68, 71
- restore_level
 - options.js, 39
- ret
 - wrappingS-PT2.js, 74
- rounding_function
 - wrapping.js, 50
- runPostMessage
 - wrappingS-HTML-LS.js, 68
- saveDomainLevels
 - levels.js, 29
- script
 - wrappingS-HTML-LS.js, 71
- send
 - wrappingS-HTML-LS.js, 69
- sendMessage
 - document_start.js, 15
 - options.js, 40
- setBigIntDecorator
 - wrappingS-ECMA-ARRAY.js, 58
- setByteDecorator
 - wrappingS-ECMA-ARRAY.js, 58
- setDefaultLevel
 - levels.js, 30
- setFloatDecorator
 - wrappingS-ECMA-ARRAY.js, 59
- show_existing_level
 - options.js, 40
- showRefreshPageOption
 - popup.js, 45
- slowBody
 - wrappingS-HTML-LS.js, 71
- tab_levels
 - background.js, 11
- tabActivate
 - background.js, 10
- tabUpdate
 - background.js, 10
- terminate
 - wrappingS-HTML-LS.js, 72
- terminated
 - wrappingS-HTML-LS.js, 72
- typedTypes
 - wrappingS-ECMA-ARRAY.js, 62
- uniqueErrorHostsLimit
 - http_shield_chrome.js, 8
- uniqueErrorHostsRatio
 - http_shield_chrome.js, 8
- unpackF32
 - wrappingS-ECMA-ARRAY.js, 59
- unpackF64
 - wrappingS-ECMA-ARRAY.js, 60
- unpackIEEE754
 - wrappingS-ECMA-ARRAY.js, 61
- update.js
 - addListener, 47
 - if, 47
 - installUpdate, 47
- update_whitelist
 - options.js, 41
- updateBadge
 - background.js, 10
- updateLevels
 - levels.js, 30
- url
 - popup.js, 47
- url.js
 - extractSubDomains, 48
- workerAddEventListener
 - wrappingS-HTML-LS.js, 69
- workerErrorListeners
 - wrappingS-HTML-LS.js, 72
- workerMessageListeners
 - wrappingS-HTML-LS.js, 72
- workerPostMessage
 - wrappingS-HTML-LS.js, 69
- workerSelf
 - wrappingS-HTML-LS.js, 72
- wrap_code
 - code_builders.js, 14
- wrapped_codes
 - levels.js, 33
- wrappers
 - wrappingS-ECMA-ARRAY.js, 63
 - wrappingS-ECMA-SHARED.js, 64
 - wrappingS-HTML-LS.js, 72
 - wrappingS-PT2.js, 74
- wrapping.js
 - add_wrappers, 49
 - build_wrapping_code, 50
 - noise_function, 49, 50
 - rounding_function, 50
- wrapping_groups
 - levels.js, 33
- wrappingS-BATTERY-CR.js
 - add_wrappers, 50
- wrappingS-ECMA-ARRAY.js
 - _data, 61
 - _target, 61
 - add_wrappers, 52
 - constructDecorator, 52
 - DEFAULT_TYPED_ARRAY_WRAPPER, 61

- for, [52](#)
- getBigIntDecorator, [53](#)
- getByteDecorator, [53](#)
- getFloatDecorator, [53](#)
- if, [54](#)
- j, [62](#)
- offsetDecorator, [54](#)
- offsetF, [62](#)
- packF32, [55](#)
- packF64, [55](#)
- packIEEE754, [56](#)
- proxy, [62](#)
- proxyHandler, [62](#)
- redefineDataViewFunctions, [56](#)
- redefineNewArrayConstructors, [57](#)
- redefineNewArrayFunctions, [57](#)
- setBigIntDecorator, [58](#)
- setByteDecorator, [58](#)
- setFloatDecorator, [59](#)
- typedTypes, [62](#)
- unpackF32, [59](#)
- unpackF64, [60](#)
- unpackIEEE754, [61](#)
- wrappers, [63](#)
- wrappingS-ECMA-SHARED.js
 - _data, [63](#)
 - add_wrappers, [63](#)
 - proxy, [64](#)
 - proxyHandler, [64](#)
 - wrappers, [64](#)
- wrappingS-HRT.js
 - add_wrappers, [65](#)
- wrappingS-HTML-LS.js
 - _old, [70](#)
 - addEventListener, [66, 70](#)
 - api, [70](#)
 - callErrorListener, [66](#)
 - errorListeners, [70](#)
 - executeEach, [66](#)
 - onreadystatechange, [70](#)
 - open, [67](#)
 - postError, [67](#)
 - postMessage, [68, 71](#)
 - postMessageListeners, [71](#)
 - removeEventListener, [68, 71](#)
 - runPostMessage, [68](#)
 - script, [71](#)
 - send, [69](#)
 - slowBody, [71](#)
 - terminate, [72](#)
 - terminated, [72](#)
 - workerAddEventListener, [69](#)
 - workerErrorListeners, [72](#)
 - workerMessageListeners, [72](#)
 - workerPostMessage, [69](#)
 - workerSelf, [72](#)
 - wrappers, [72](#)
 - xhr, [72](#)
- wrappingS-PT2.js
 - common_function_body, [73](#)
 - for, [73](#)
 - func, [73](#)
 - if, [73](#)
 - ret, [74](#)
 - wrappers, [74](#)
- xhr
 - wrappingS-HTML-LS.js, [72](#)