Communication between content script and web_accessible_resources iframe

Asked 2 years, 8 months ago Modified 1 year, 2 months ago Viewed 2k times

1

I have a content script that injects an iframe into a webpage.

content.js

```
var iframe = document.createElement('iframe');
    iframe.id = "frame";
    iframe.style.cssText = "position:fixed;top: 15px;right: 15px;width: 
250px;height: 245px;overflow: hidden;background-color:#FFFFFF;border-radius: 
5px;";
    iframe.src = chrome.runtime.getURL('frame.html');
    document.body.appendChild(iframe);
```

The iframe displays some text values, has a submit and a close button.

part of frame.html

```
<div class="header">
   <span class="close">Name</span>
   <span class="close-btn" id="close-btn">&times;</span>
</div>
<div class="details-container">
    <span class="label">First Name : </span>
    <span id="fname" type="text" ></span>
</div>
<div class="details-container">
    <span class="label">Last Name : </span>
    <span id="lname" type="text" /></span>
</div>
<div class="btn-details-container">
    <button class="copy" id="copy-name">Copy</button>
</div>
```

frame.html has frame.js linked to it.

I want to do 2 things here.

1. Close/Remove/Hide the iframe when user clicks on close button on the iframe(#close-btn)
2. The values of first name and last name in span to be dynamically set (extracted from DOM of current webpage)

Problems:

1)I don't know how to propogate click event on frame.html to content script to close iframe(Unable to establish communication between frame.js and content.js)

2)Not able to set span.textContent for #fname and #lname because frame.js is not able to read webpage DOM.

javascript html iframe google-chrome-extension

Share Improve this question Follow

edited Jan 9, 2023 at 13:27
wOxxOm
69.8k ● 13 ● 144 ● 148

asked Aug 6, 2021 at 21:27
zoyo
13 ● 3

1 Answer

Sorted by: Highest score (default)

3

Extension messaging (iframe controls the logic)

Use chrome.tabs.sendMessage to communicate with the owner tab of the iframe, which can be retrieved using chrome.tabs.getCurrent inside the iframe.

content.js:

```
var FRAME_URL = chrome.runtime.getURL('frame.html');
var iframe = document.createElement('iframe');
iframe.src = FRAME_URL;
document.body.appendChild(iframe);

chrome.runtime.onMessage.addListener((msg, sender, sendResponse) => {
  switch (msg.cmd) {
    case 'close':
      iframe.remove();
      iframe = null;
      break;
    case 'getData':
      sendResponse([
        ['fname',
document.querySelector('.web.page.selector.for.fname').textContent],
        ['lname',
document.querySelector('.web.page.selector.for.lname').textContent],
      ]);
      break;
  }
});
```

iframe.js:

```
tellParent({cmd: 'getData'}, data => {
  for (const [id, val] of data) {
    document.getElementById(id).textContent = val;
  }
});

document.querySelector('.close-btn').onclick = () => {
  tellParent({cmd: 'close'});
};
```

```
function tellParent(msg, callback) {
  chrome.tabs.getCurrent(tab => {
    chrome.tabs.sendMessage(tab.id, msg, {frameId: 0}, callback);
  });
}
```

Extension messaging (two-way port)

Initiate the port using chrome.tabs.connect in the iframe, then use it in the content script.

content script:

```
let framePort;
chrome.runtime.onConnect.addListener(port => {
  if (port.name === 'frame') {
    // global framePort can be used by code that will run in the future
    framePort = port;
    port.postMessage({foo: 'bar'});
  }
});

// add iframe element and point it to chrome.runtime.getURL('iframe.html')
//...........
```

iframe script:

```
chrome.tabs.getCurrent(tab => {
  const port = chrome.tabs.connect(tab.id, {name: 'frame', frameId: 0});
  port.onMessage.addListener(msg => {
    if (msg.foo === 'bar') {
      console.log(msg);
    }
  });
});
```

Web messaging (two-way MessagePort)

It's super fast and supports binary data types like Blob or ArrayBuffer but requires certain care to avoid interception by the web page:

1. Create the iframe inside a closed ShadowDOM to avoid exposing `window[0]`
2. Don't set iframe's `src`, instead navigate its inner `location` using a random secret in the url parameters so that its URL won't be spoofed by the web page or other extensions which used chrome.dom.openOrClosedShadowRoot.
3. pass the safe MessagePort into the iframe via postMessage
4. use this safe MessagePort for two-way communication

// content.js

```
(async () => {
  const port = await makeExtensionFramePort('/iframe.html');
  port.onmessage = e => {
    console.log('from iframe:', e.data);
  };
  port.postMessage(123);
  port.postMessage({ foo: bar });
  port.postMessage(new Blob(['foo']));
})();

async function makeExtensionFramePort(path) {
  const secret = Math.random().toString(36);
  const url = new URL(chrome.runtime.getURL(path));
  url.searchParams.set('secret', secret);
  const el = document.createElement('div');
  const root = el.attachShadow({mode: 'closed'});
  const iframe = document.createElement('iframe');
  iframe.hidden = true;
  root.appendChild(iframe);
  (document.body || document.documentElement).appendChild(el);
  await new Promise((resolve, reject) => {
    iframe.onload = resolve;
    iframe.onerror = reject;
    iframe.contentWindow.location.href = url;
  });
  const mc = new MessageChannel();
  iframe.contentWindow.postMessage(1, '*', [mc.port2]);
  return mc.port1;
}
```

// iframe.html:

```
<script src="iframe.js"></script>
```

// iframe.js

```
let port;
window.onmessage = e => {
  if (e.data === new URLSearchParams(location.search).get('secret')) {
    window.onmessage = null;
    port = e.ports[0];
    port.onmessage = onContentMessage;
  }
};

function onContentMessage(e) {
  console.log('from content:', e.data);
  port.postMessage('ok');
}
```

- Modification: a direct two-way port between the content script and the extension's service worker by using `navigator.serviceWorker` messaging in the iframe:

  // iframe.js

  ```
  let port;
  window.onmessage = e => {
    if (e.data === new URLSearchParams(location.search).get('secret')) {
      window.onmessage = null;
      navigator.serviceWorker.ready.then(swr => {
        swr.active.postMessage('port', [e.ports[0]]);
      });
  ```

```
  }
};
```

// background.js

```
self.onmessage = e => {
  if (e.data === 'port') {
    e.ports[0].onmessage = onContentMessage;
  }
}
function onContentMessage(e) {
  // prints both in the background console and in the iframe's console
  console.log('from content:', e.data);
  port.postMessage('ok');
}
```

Share Improve this answer Follow edited Jan 28, 2023 at 9:29 answered Aug 7, 2021 at 6:17

wOxxOm
69.8k ● 13 ● 144 ● 148

Thank you for your answer. I am able to communicate between scripts now. But the if condition (sender.frameId && (sender.url || '').startsWith(FRAME_URL)) isn't working here. sender. frameId and sender.url says undefined when I log it. And the switch should say switch (msg.cmd) – zoyo Aug 7, 2021 at 9:37

Thanks, this condition is not really necessary so I've removed it now. – wOxxOm Aug 7, 2021 at 10:07

iframe.remove() didn't work for me, it returned null exception. I did it like--- var frame = document.getElementById("frame"); frame.parentNode.removeChild(frame); – zoyo Aug 7, 2021 at 11:11

Can I have a follow up question? I wanted to know if there is a subtle way to show extracted text contents in the iframe. Because the code I have now pop ups the iframe and then it take few seconds to insert text contents in it. – zoyo Aug 7, 2021 at 11:14

1) iframe.remove works in Chrome/Firefox/Edge for the past 5 years so that exception must be unrelated. 2) You can add the iframe only after you find the elements. – wOxxOm Aug 7, 2021 at 15:44