Chapter 75: .postMessage() and MessageEvent

Parameters	
message	
targetOrigin	
transfer	`optional`

Section 75.1: Getting Started

What is .postMessage(), when and why do we use it

`.postMessage()` method is a way to safely allow communication between cross-origin scripts.

Normally, two different pages, can only directly communicate with each other using JavaScript when they are under the same origin, even if one of them is embedded into another (e.g. `iframes`) or one is opened from inside the other (e.g. `window.open()`). With `.postMessage()`, you can work around this restriction while still staying safe.

You can only use `.postMessage()` when you have access to both pages' JavaScript code. Since the receiver needs to validate the sender and process the message accordingly, you can only use this method to communicate between two scripts you have access to.

We will build an example to send messages to a child window and have the messages be displayed on the child window. The parent/sender page will be assumed to be `http://sender.com` and child/receiver page will be assumed to be `http://receiver.com` for the example.

Sending messages

In order to send messages to another window, you need to have a reference to its `window` object. `window.open()` returns the reference object of the newly opened window. For other methods to obtain a reference to a window object, see the explanation under `otherWindow` parameter here.

```
var childWindow = window.open("http://receiver.com", "_blank");
```

Add a `textarea` and a `send button` that will be used to send messages to child window.

```
<textarea id="text"></textarea>
<button id="btn">Send Message</button>
```

Send the text of `textarea` using `.postMessage(message, targetOrigin)` when the `button` is clicked.

```
var btn = document.getElementById("btn"),
    text = document.getElementById("text");

btn.addEventListener("click", function () {
    sendMessage(text.value);
    text.value = "";
});

function sendMessage(message) {
    if (!message || !message.length) return;
    childWindow.postMessage(JSON.stringify({
        message: message,
        time: new Date()
    }), 'http://receiver.com');
}
```

In order send and receive JSON objects instead of a simple string, `JSON.stringify()` and `JSON.parse()` methods can be used. A `Transfarable Object` can be given as the third optional parameter of the `.postMessage(message, targetOrigin, transfer)` method, but browser support is still lacking even in modern browsers.

For this example, since our receiver is assumed to be `http://receiver.com` page, we enter its url as the `targetOrigin`. The value of this parameter should match the `origin` of the `childWindow` object for the message to be send. It is possible to use `*` as a `wildcard` but is **highly recommended** to avoid using the wildcard and always set this parameter to receiver's specific origin **for security reasons**.

Receiving, Validating and Processing Messages

The code under this part should be put in the receiver page, which is `http://receiver.com` for our example.

In order to receive messages, the `message event` of the `window` should be listened.

```
window.addEventListener("message", receiveMessage);
```

When a message is received there are a couple of **steps that should be followed to assure security as much as possible**.

- Validate the sender
- Validate the message
- Process the message

The sender should always be validated to make sure the message is received from a trusted sender. After that, the message itself should be validated to make sure nothing malicious is received. After these two validations, the message can be processed.

```
function receiveMessage(ev) {
    //Check event.origin to see if it is a trusted sender.
    //If you have a reference to the sender, validate event.source
    //We only want to receive messages from http://sender.com, our trusted sender page.
    if (ev.origin !== "http://sender.com" || ev.source !== window.opener)
        return;

    //Validate the message
    //We want to make sure it's a valid json object and it does not contain anything malicious
    var data;
    try {
        data = JSON.parse(ev.data);
        //data.message = cleanseText(data.message)
    } catch (ex) {
        return;
    }

    //Do whatever you want with the received message
    //We want to append the message into our #console div
    var p = document.createElement("p");
    p.innerText = (new Date(data.time)).toLocaleTimeString() + " | " + data.message;
    document.getElementById("console").appendChild(p);
}
```