HTML - Web Messaging

Web Messaging is the way for documents to separates browsing context to share the data without Dom. It overrides the cross domain communication problem in different domains, protocols or ports.

For example, if we want to send the data from our page to ad container which is placed at iframe or voice-versa, in this scenario, Browser throws a security exception. With web messaging we can pass the data across as a message event.

Message Event

Message events fires Cross-document messaging, channel messaging, server-sent events and web sockets. It is described by Message Event interface.

Properties of Message Event

The following table contains a list of Message Event properties –

S.No.	Property & Description
1	**data** Contains string data
2	**origin** Contains Domain name and port
3	**lastEventId** Contains unique identifier for the current message event.
4	**source** Contains to A reference to the originating document's window
5	**ports** Contains the data which is sent by any message port

Sending a cross-document message

Before send cross document message, we need to create a new web browsing context either by creating new iframe or new window. We can send the data using with postMessage() and it has two arguments. They are as

- **message** – The message to send
- **targetOrigin** – Origin name

Examples

Sending message from iframe to button

```
var iframe = document.querySelector('iframe');
var button = document.querySelector('button');
var clickHandler = function(){
   iframe.contentWindow.postMessage('The message to
send.','https://www.tutorialspoint.com);
}
button.addEventListener('click',clickHandler,false);
```

Receiving a cross-document message in the receiving document

```
var messageEventHandler = function(event){
   // check that the origin is one we want.
   if(event.origin == 'https://www.tutorialspoint.com'){
      alert(event.data);
    }
}
window.addEventListener('message', messageEventHandler,false);
```

Channel messaging

Two-way communication between the browsing contexts is called channel messaging. It is useful for communication across multiple origins.

The MessageChannel and MessagePort Objects

While creating messageChannel, it internally creates two ports to sending the data and forwarded to another browsing context.

- **postMessage()** – Post the message throw channel
- **start()** – It sends the data
- **close()** – it close the ports

In this scenario, we are sending the data from one iframe to another iframe. Here we are invoking the data in function and passing the data to DOM.

```
var loadHandler = function(){
   var mc, portMessageHandler;
   mc = new MessageChannel();
   window.parent.postMessage('documentAHasLoaded','http://foo.example',[mc.port2]);
   portMessageHandler = function(portMsgEvent){
      alert( portMsgEvent.data );
    }
   mc.port1.addEventListener('message', portMessageHandler, false);
   mc.port1.start();
}
window.addEventListener('DOMContentLoaded', loadHandler, false);
```

Above code, it is taking the data from port 2, now it will pass the data to second iframe

```
var loadHandler = function(){
   var iframes, messageHandler;
   iframes = window.frames;
   messageHandler = function(messageEvent){
      if( messageEvent.ports.length > 0 ){
         // transfer the port to iframe[1]
         iframes[1].postMessage('portopen','http://foo.example',messageEvent.ports);
      }
   }
   window.addEventListener('message',messageHandler,false);
}
window.addEventListener('DOMContentLoaded',loadHandler,false);
```

Now second document handles the data by using the portMsgHandler function.

```
var loadHandler(){
   // Define our message handler function
   var messageHandler = function(messageEvent){

      // Our form submission handler
      var formHandler = function(){
         var msg = 'add <foo@example.com> to game circle.';
         messageEvent.ports[0].postMessage(msg);
      }
      document.forms[0].addEventListener('submit',formHandler,false);
   }
   window.addEventListener('message',messageHandler,false);
}
window.addEventListener('DOMContentLoaded',loadHandler,false);
```