**WEB WORKERS – HTML5**

* HTML5 new API
* Allows us to execute some JS in background of page without interrupting the current running JS and page performance
* Mimic multithreading as JS runs in background without affecting user interface

**WHAT IS IT?**

* A worker is an object created using a constructor (e.g. [Worker()](https://developer.mozilla.org/en-US/docs/Web/API/Worker/Worker)) that runs a named JavaScript file
* this file contains the code that will run in the worker thread
* workers run in another global context that is different from the current [window](https://developer.mozilla.org/en-US/docs/Web/API/Window)
* worker object is accessible through self keyword

For e.g. :

If(window.Worker) **//** check browser supportas worker is globally available

{

var worker = new Worker(“worker.js”); **//** worker.js is file\_to\_be\_passed

}

**COMMUNICATION BETWEEN MAIN THREAD AND WORKERS :-**

* communication is done using message event and event hanlder (e.g. onmessage and postMessage)
* both side can share and respond to messages
* message which is passed can be either string or object

|  |  |
| --- | --- |
| Main thread (main.js) | Worker thread(worker.js) |
| var worker = new Worker(worker.js);  worker.postMessage(“main thread”);  worker.addEventListener(“message”,function(evt){  console.log(evt.data);  }, false); | self.addeventListener(“message”, function(evt){  console.log(evt.data);    self.postMessage(evt.data + “ worker thread”);  },false); |
| Output : main thread worker thread | Output : main thread |

**Types of Worker :**

* Dedicated Web Workers
* Shared Web Workers

|  |  |  |
| --- | --- | --- |
|  | Dedicated Web Worker | Shared Web Worker |
|  | * Standard Worker utilized by a single Script( Linked to creator) | * Allow any script of same domain/origin to obtain the reference of worker and communicate with it |
| Initialization | * var dWorker = new worker(“worker.js”); | * var sWorker = new SharedWorker(“shared\_worker.js”); |
| Listening & passing events | * dWorker.addEventListener(“message”, eventHandler, false); * dWorker.postMessage(“msg”); | * sWorker.port.addEventListener(“message”, eventHandler, false); * sWorker.port.postMessage(“msg”); |
| Closing a worker | * dWorker.terminate(); | * sWorker.terminate(); |