**Using SharedWebWorker:**

var myWorker = new SharedWorker(*aURL*, *name*);

aURL : path of shared-worker.js file. It must obey the same-origin policy.

name:

* Optional argument
* specifies an existing SharedWorkerGlobalScope.name — if this is specified then that [SharedWorkerGlobalScope](https://developer.mozilla.org/en-US/docs/Web/API/SharedWorkerGlobalScope) will be used as the scope for this shared worker.

**Note:**

* The object returned by the SharedWorker () constructor holds a reference to the port on its port attribute. For e.g :

worker.port.postMessage('some message');

**Connecting to multiple pages**:

* The script uses the onconnect event listener to listen for multiple connections.

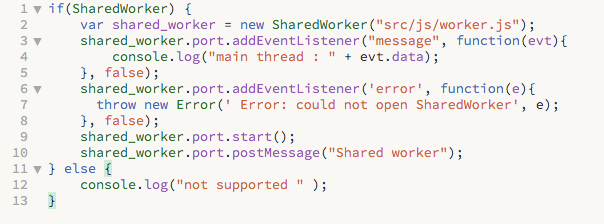
**Direct Channel**:

* When the worker receives a “msg” message from one viewer naming another viewer, it sets up a direct connection between the two, so that the two viewers can communicate directly without the worker having proxy all the messages.

**Pros:**

* With multicore CPUs , to obtain better performance is to split computationally expensive tasks amongst multiple workers( shared workers).

EXAMPLE - 01 -

MAIN.JS****

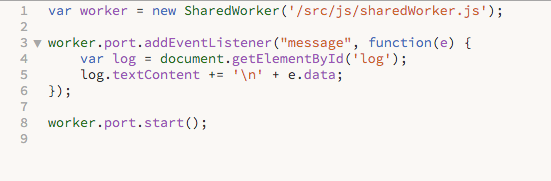
WORKER.JS****

EXAMPLE - 02 -

In this case, the second page is merely in an iframe on the first page, but the same principle would apply to an entirely separate page in a separate top-level browsing context.

INDEX.HTML

INNER.HTML

MAIN.JS

SHAREDWORKER.JS