Web Workers

Introduction

- The HTML 5 Web Workers provides background processing capabilities. We can use the Web Workers API for running separated threads concurrently with the main scripts in our web page.
- The Web Workers API is especially useful in the prevention of user messages such as the 'unresponsive script' message.

Limitations

- The code executed in a separated thread using the Web Workers API cannot access neither the web page nor its document object model.

Worker

- In order to get a specific JavaScript code executed concurrently in a separated thread we should instantiate the `Worker` type passing over the name of the file that includes the JavaScript code we want to execute in a separated thread.

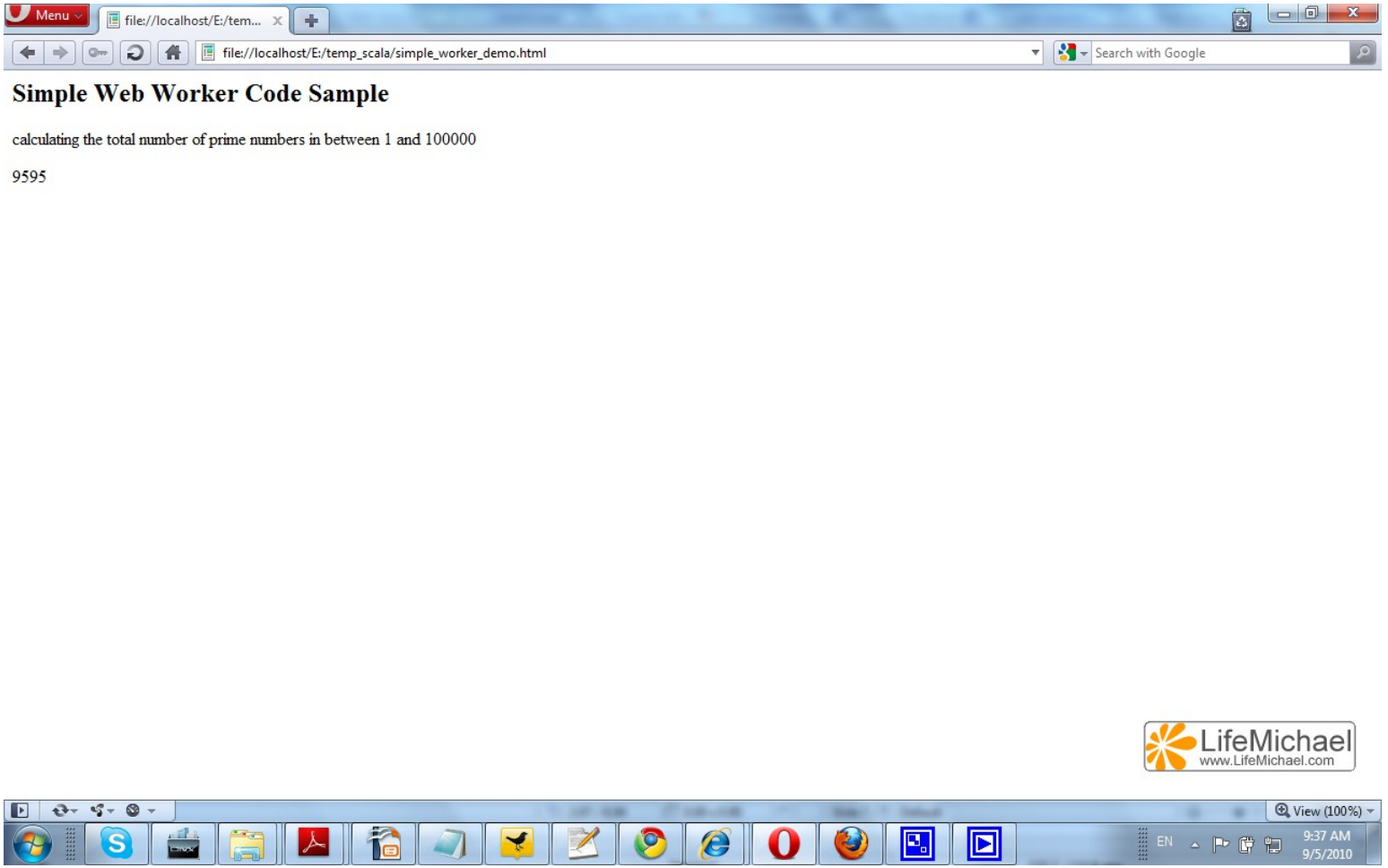
Sample

```
<h2>Simple Web Worker Code Sample</h2>
<p>calculating the total number of prime numbers in between 1 and 100000</p>
<div id="total">1</div>
<script>
    const worker = new Worker('background.js');
    worker.onmessage = updateResult;
    function updateResult(event)
    {
        document.getElementById('total').innerHTML = event.data;
    };
</script>
```

Sample

```
let total = 2;
outer: for(var n=1;n<=100000;n++)
{
    for (let i = 2; i <= Math.sqrt(n); i += 1)
    {
        if (n % i == 0) continue outer;
    }
    total++;
    postMessage(total);
}
```

Sample



The `addEventListener` Method

- We can use this function in order to set the code that should be executed when a message returns back from the other script.

The `addEventListener` Method

```
<!DOCTYPE html>
<html>
<head>
    <title>web worker demo</title>
</head>
<body>
    <h2>Simple Web Worker Code Sample</h2>
    <p>calculating the total of prime numbers in between 1 and 1000000</p>
    <div id="total">1</div>
    <script type="text/javascript">
        const worker = new Worker('background.js');
        worker.addEventListener('message', function(event)
            {
                document.getElementById('total').innerHTML = event.data;
            }
        );
    </script>
</body>
</html>
```

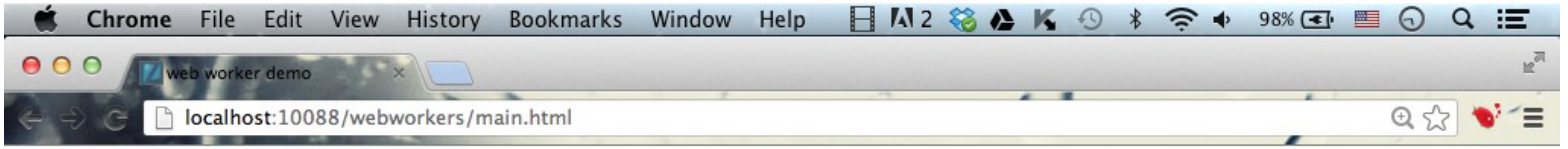
The `addEventListener` Method

```
let total = 2;
outer: for(let n=1;n<=1000000;n++)
{
    for (let i = 2; i <= Math.sqrt(n); i += 1)
    {
        if (n % i == 0) continue outer;
    }
    total++;
    postMessage(total);
}
```

The `addEventListener` Method

```
var total = 2;
outer: for(var n=1;n<=100000;n++)
{
    for (var i = 2; i <= Math.sqrt(n); i += 1)
    {
        if (n % i == 0) continue outer;
    }
    total++;
    postMessage(total);
}
```

The `addEventListener` Method



Simple Web Worker Code Sample

calculating the total number of prime numbers in between 1 and 1000000

41477

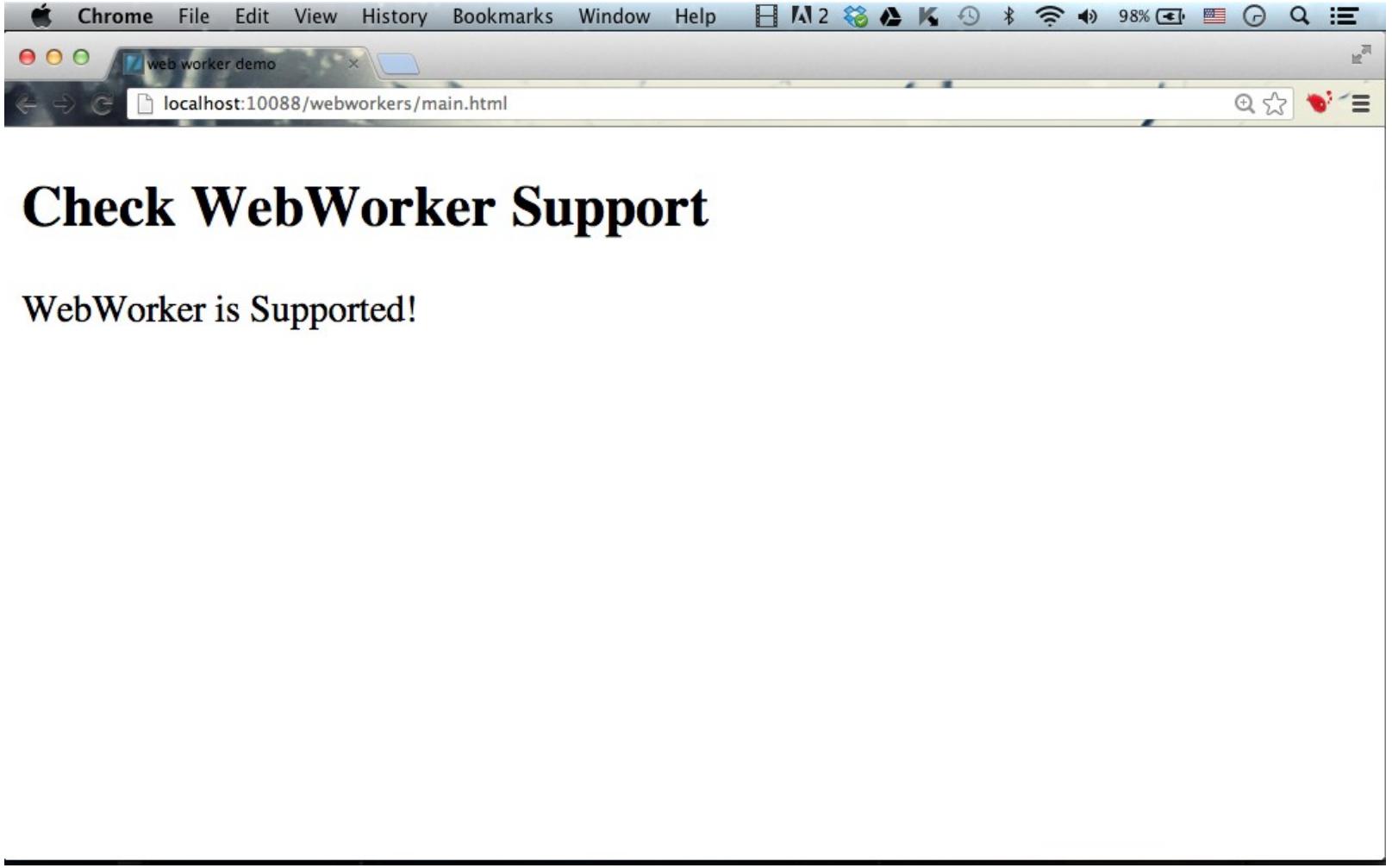
Checking WebWorker Support

- We can check whether the web browser supports the WebWorker API or not either by using the Modernizr.js library or by checking the Worker property in the window global object.

Checking WebWorker Support

```
<!DOCTYPE html>
<html>
<head>
    <title>web worker demo</title>
</head>
<body>
    <h2>Check WebWorker Support</h2>
    <script type="text/javascript">
        if(window.Worker!=undefined) {
            document.write("WebWorker is Supported!");
        }
    </script>
</body>
</html>
```

Checking WebWorker Support



The `terminate()` Function

- We can terminate a web worker by calling the `terminate()` function on it.

The `terminate()` Function

```
<!DOCTYPE html>
<html>
<body>
<p>
<button onclick="start()">Start</button>
<button onclick="stop()">Stop</button>
counter: <span id="output">0</span>
</p>
<script>
    const worker;
    function start() {
        let worker = new Worker("background.js");
        worker.onmessage = function (event) {
                document.getElementById("output").innerHTML=event.data;
            };
    }
    function stop() {
        worker.terminate();
    }
</script>
</body>
</html>
```

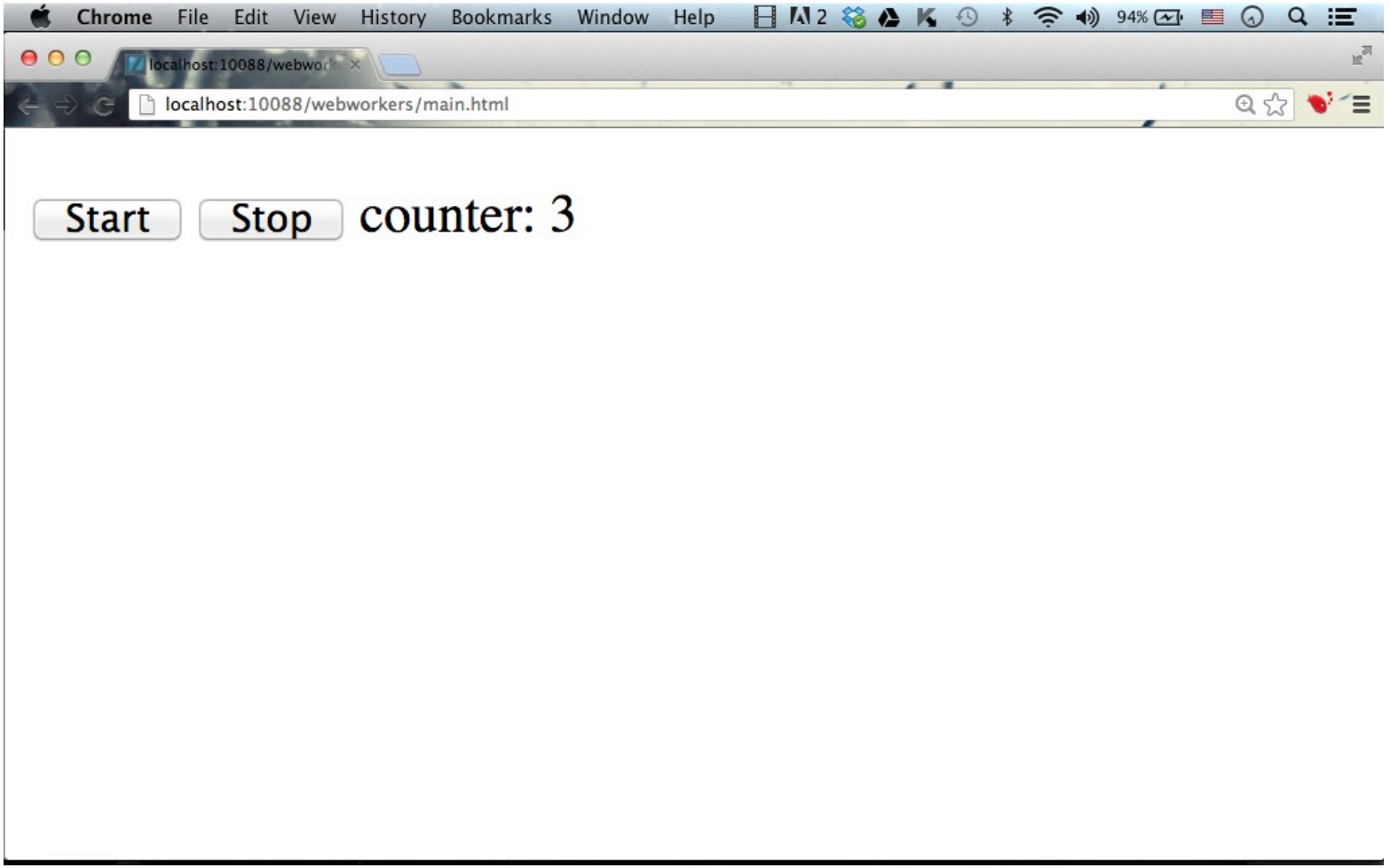
The `terminate()` Function

```
let num=0;

function tick()
{
    num=num+1;
    postMessage(num);
    setTimeout("tick()",1000);
}

tick();
```

The `terminate()` Function



Web Workers Debugging

- When using chrome we can easily debug our web workers .
  We just need to select 'Pause on Start' checkbox in the Workers section on the right.

Web Workers Debugging