



HTML5

Novas API

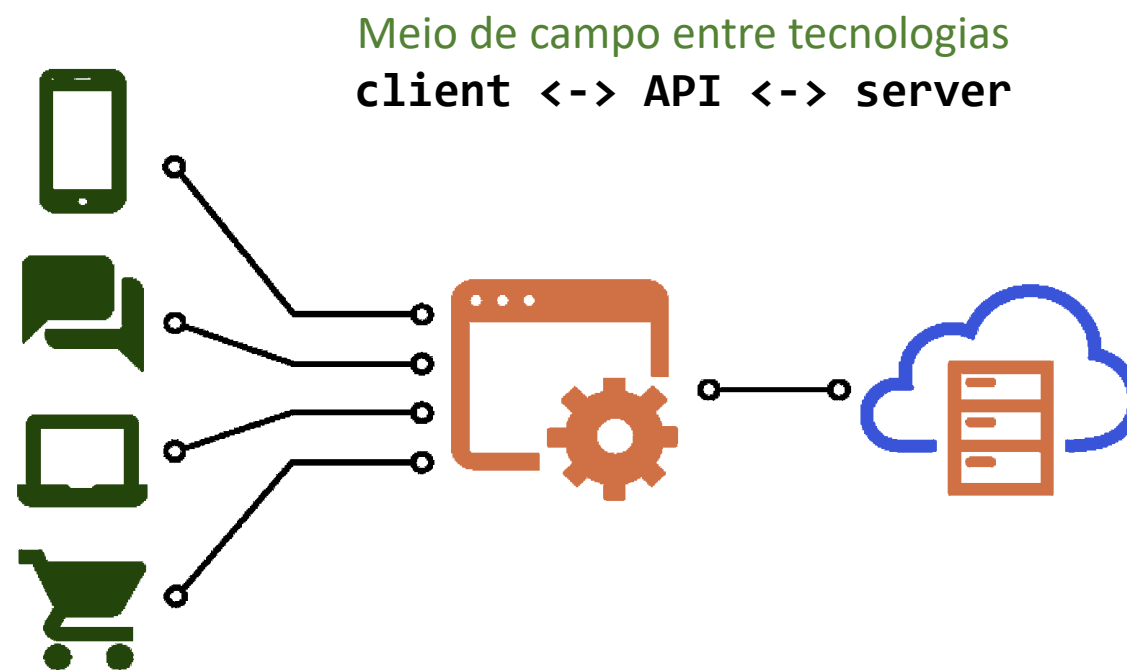


O que são API ?

Application Programming Interfaces ou API são construções de software que **facilitam o desenvolvimento de funcionalidades complexas.**

Abstraem o código mais complexo, oferecendo **sintaxes mais simples** em seu lugar.

Responsáveis por **estabelecer comunicação entre diferentes serviços.**



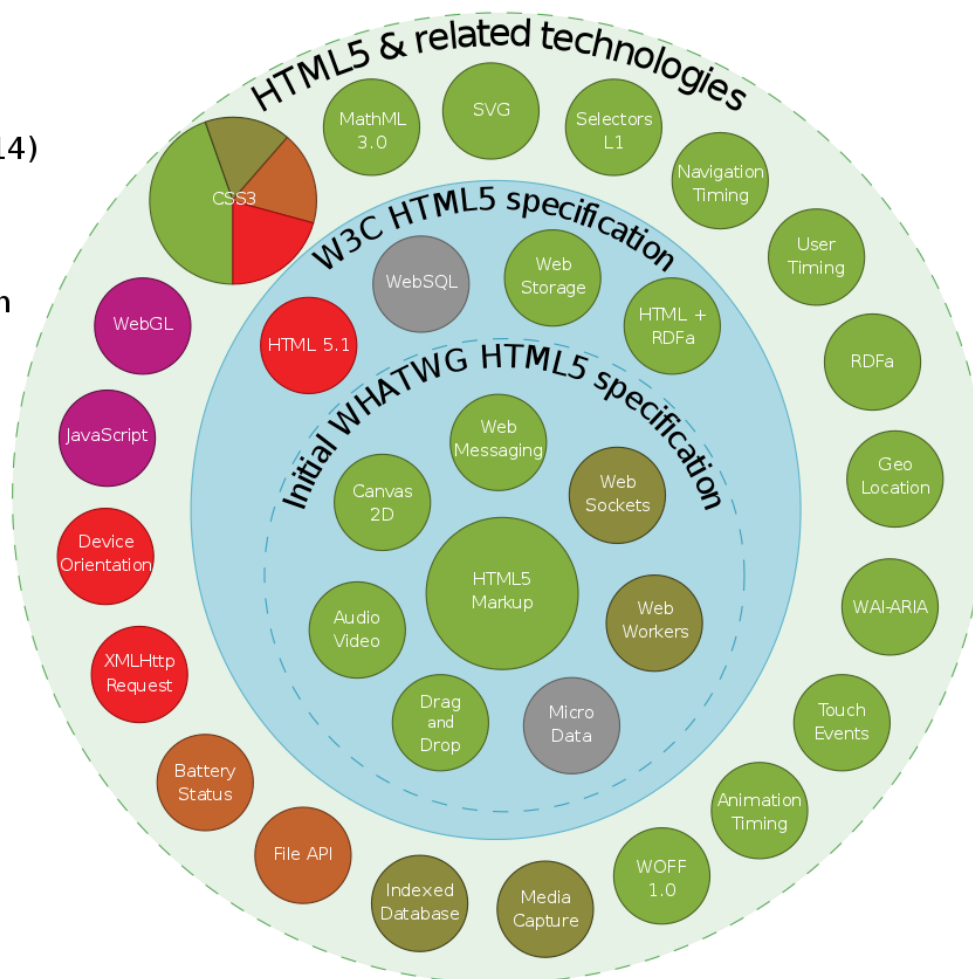


O HTML5 não se trata apenas de marcação, mas também de um **conjunto de novas funcionalidades encapsuladas em API** acessíveis por meio do JavaScript.

HTML5

Taxonomy & Status (October 2014)

- Recommendation/Proposed
- Candidate Recommendation
- Last Call
- Working Draft
- Non-W3C Specifications
- Deprecated or inactive



Porém, há diversas API em processo de padronização pelo W3C que **não fazem parte da especificação do HTML5.**

As API do HTML5 são mais específicas às funcionalidades que atuam **no escopo da página e da manipulação de elementos**. Elas se relacionam em grande parte com o **DOM**.

As demais usualmente trabalham **com features mais complexas**, tais como armazenamento de dados e manipulação de arquivos.



HTML Geolocation API



HTML Geolocation API








Usada para localizar a **posição geográfica** de um usuário.

Já que esse dado pode comprometer a privacidade do usuário, a **posição não está disponível a menos que o usuário aprove.**



A partir do Chrome 50, a **Geolocation API só funcionará em contextos seguros como HTTPS**. Se o site estiver hospedado em uma origem não segura (como HTTP), as solicitações para obter a localização dos usuários não funcionarão.

				
5.0 - 49.0 (http) 50.0 (https)	9.0	3.5	5.0	16.0



```
<body>
  <p>Clique no botão para pegar suas coordenadas.</p>

  <button onclick="getLocalizacao()">1, 2, 3, testando</button>

  <p id="teste"></p>

  <script>
    var x = document.getElementById("teste");

    function getLocalizacao() {
      if (navigator.geolocation) {
        navigator.geolocation.getCurrentPosition(mostraPosicao);
      } else {
        x.innerHTML = "Geolocalização não suportada pelo browser.";
      }
    }

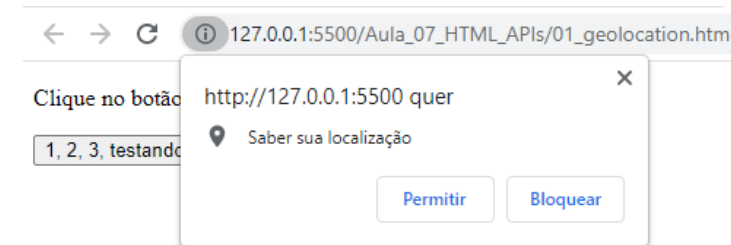
    function mostraPosicao(posicao) {
      x.innerHTML = "Latitude: " + posicao.coords.latitude +
        "<br>Longitude: " + posicao.coords.longitude;
    }
  </script>
</body>
```

0) Usando o DOM, busca-se o elemento cujo atributo ID é igual a **teste** e atribui-se ele à uma variável **x**.

1) Testa se o navegador suporta geolocalização. Para fazer isso utiliza-se a propriedade **geolocation** existente no objeto **navigator**

2) Se existir a propriedade, o script executa o método **getCurrentPosition()**. Caso contrário, uma mensagem é exibida para o usuário.

Se existe a propriedade, então o método **getCurrentPosition()** instrui o navegador a solicitar a permissão do usuário para acessar a localização dele.



Clique no botão para pegar suas coordenadas.

1, 2, 3, testando

Latitude: -31.7607818
Longitude: -52.3124246

3) Se o usuário permitir, o navegador obtém as coordenadas de localização, retornando um objeto com coordenadas geográficas para a função especificada no parâmetro (**mostraPosicao**).

Ao fazer isso, é acionada a função **mostraPosicao(posicao)**



```
<body>
  <p>Clique no botão para pegar suas coordenadas.</p>

  <button onclick="getLocalizacao()">1, 2, 3, testando</button>

  <p id="teste"></p>

  <script>
    var x = document.getElementById("teste");

    function getLocalizacao() {
      if (navigator.geolocation) {
        navigator.geolocation.getCurrentPosition(mostraPosicao);
      } else {
        x.innerHTML = "Geolocalização não suportada pelo browser.";
      }
    }

    function mostraPosicao(posicao) {
      x.innerHTML = "Latitude: " + posicao.coords.latitude + "  

        Longitude: " + posicao.coords.longitude;
    }
  </script>
</body>
```

4) O objeto **posicao** é passado para a função **mostraPosicao**, e ele contém informações sobre a posição geográfica atual do dispositivo.

5) O objeto **posicao** não é criado diretamente pelo código, mas fornecido pelo navegador em resposta à solicitação de geolocalização.

Ele contém os dados da localização. A função **mostraPosicao(posicao)** exibe então os dados existentes dentro desse objeto.

Clique no botão para pegar suas coordenadas.

1, 2, 3, testando

Latitude: -31.7607818
Longitude: -52.3124246



```
<script>
  var x = document.getElementById("teste");

  function getLocalizacao() {
    if (navigator.geolocation) {
      navigator.geolocation.getCurrentPosition(mostraPosicao, mostraErro);
    } else {
      x.innerHTML = "Geolocalização não suportada pelo browser.";
    }
  }

  function mostraPosicao(posicao) {
    x.innerHTML = "Latitude: " + posicao.coords.latitude +
      "<br>Longitude: " + posicao.coords.longitude;
  }
}
```

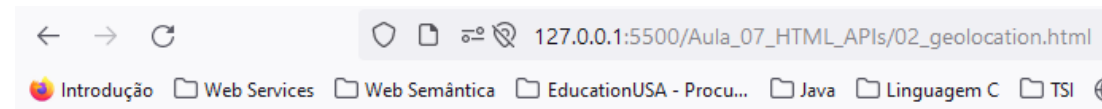
Um segundo parâmetro de `getCurrentPosition()` é usado no **tratamento de erros**.

Ao iniciar o processo de obtenção da localização do usuário, são passadas as funções `mostraPosicao` e `mostraErro` como tratadores de **sucesso** e **erro**, respectivamente.

Quando a localização é obtida com sucesso, a função `mostraPosicao` é chamada e quando ocorre um erro, a função `mostraErro` é chamada.

```
function mostraErro(erro) {
  switch (erro.code) {
    case erro.PERMISSION_DENIED:
      x.innerHTML = "Usuário negou a solicitação de localização."
      break;
    case erro.POSITION_UNAVAILABLE:
      x.innerHTML = "Informação de localização indisponível."
      break;
    case erro.TIMEOUT:
      x.innerHTML = "Tempo de requisição expirou."
      break;
    case erro.UNKNOWN_error:
      x.innerHTML = "Erro desconhecido."
      break;
  }
}
```

Função que **trata os erros** que podem ocorrer em tempo de execução.



Clique no botão para pegar suas coordenadas.

1, 2, 3, testando

Usuário negou a solicitação pela Geolocalização.

```
<html>
<body>
  <p>Clique para revelar suas coordenadas.</p>
  <button onclick="getLocalizacao()">Clique</button>
  <p id="teste"></p>
</body>
</html>
```




Retornos do método `getCurrentPosition()`

Em caso de sucesso, retorna um **objeto com coordenadas**.

As propriedades de **latitude**, **longitude** e **accuracy** são sempre retornadas.

Propriedade	Retorno
<code>coords.latitude</code>	A latitude como um número decimal (sempre retornado)
<code>coords.longitude</code>	A longitude como um número decimal (sempre retornado)
<code>coords.accuracy</code>	A precisão da posição (sempre retornada)
<code>coords.altitude</code>	A altitude em metros acima do nível médio do mar (retorna se disponível)
<code>coords.altitudeAccuracy</code>	A precisão da posição da altitude (retorna se disponível)
<code>coords.heading</code>	O rumo em graus no sentido horário a partir do norte (retorna se disponível)
<code>coords.speed</code>	A velocidade em metros por segundo (retorna se disponível)
<code>timestamp</code>	A data / hora da resposta (retorna se disponível)

Outras aplicações de geolocalização



- Apresentar **informações locais** atualizadas.
- Mostrar **pontos de interesse próximos** ao usuário.
- **Navegação** passo a passo (GPS).

Para exibir o resultado em um mapa **é necessário acessar um serviço de mapas**, como o Google Maps, por exemplo.



Outros métodos do objeto Geolocation

watchPosition()

Recupera a posição atual do usuário e **continua retornando este dado atualizado** a medida que o usuário se move (como o GPS em um carro).

clearWatch()

Interrompe o método `watchPosition()`





```
<html>
<body>
  <p>Clique no botão para pegar suas coordenadas.</p>

  <button onclick="getLocation()">1, 2, 3, testando</button>

  <p id="teste"></p>

  <script>
    var x = document.getElementById("teste");

    function getLocation() {
      if (navigator.geolocation) {
        navigator.geolocation.watchPosition(showPosition);
      } else {
        x.innerHTML = "Geolocalização não é suportada nesse browser.";
      }
    }

    function showPosition(position) {
      x.innerHTML = "Latitude: " + position.coords.latitude + "<br>Longitude: " + position.coords.longitude;
    }
  </script>

</body>
</html>
```

Método que recupera a posição e fica a atualizando continuamente.



HTML Drag and Drop API

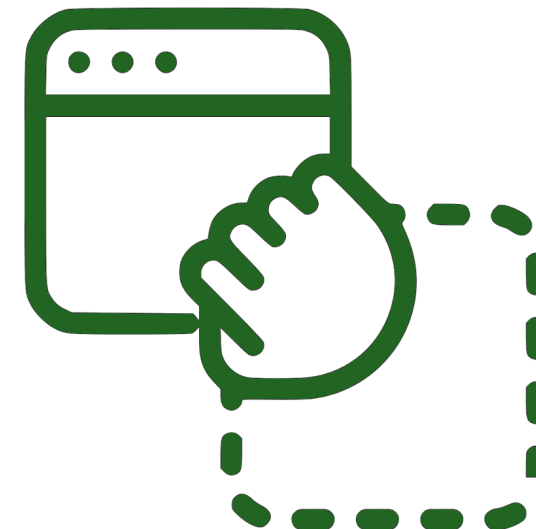
HTML Drag and Drop API







Em HTML, **qualquer elemento pode ser arrastado e solto**.

Recurso muito comum em aplicações.

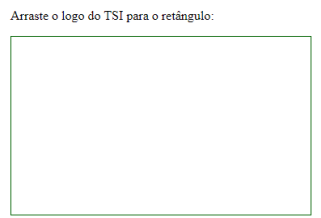
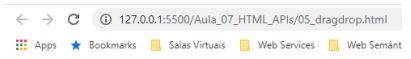
Permite ao usuário a possibilidade de **"pegar" um objeto e o "arrastá-lo"** para um local diferente.



API					
Drag and Drop	4.0	9.0	3.5	6.0	12.0



Objetivo: Arrastar a imagem para a **div**.



3) O método `dataTransfer.setData()` define o tipo de dados e o valor dos dados arrastados.

```
<script>
  function permiteSoltar(ev) {
    ev.preventDefault();
  }

  function arrastar(ev) {
    ev.dataTransfer.setData("text", ev.target.id);
  }

  function soltar(ev) {
    ev.preventDefault();
    var data = ev.dataTransfer.getData("text");
    ev.target.appendChild(document.getElementById(data));
  }
</script>
```

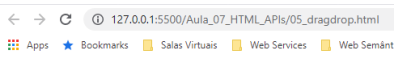
```
<html>
<head>
  <style>
    #div1 {
      width: 350px;
      height: 200px;
      padding: 10px;
      border: 1px solid darkgreen;
    }
  </style>
</head>
<body>
  <p>Arraste o logo do TSI para o retângulo:</p>
  <div id="div1" ondrop="soltar(event)" ondragover="permiteSoltar(event)">
    <br>
    
  </div>
</body>
</html>
```

(4) Evento que especifica onde os elementos podem ser soltos.

(1) Tornar elemento arrastável

(2) Definir o que acontece quando o elemento é arrastado

Objetivo: Arrastar a imagem para a div.



```
<html>
<head>
  <style>
    #div1 {
      width: 350px;
      height: 200px;
      padding: 10px;
      border: 1px solid darkgreen;
    }
  </style>
</head>
```

3) O método `dataTransfer.setData()` define o tipo de dados e o valor dos dados arrastados.

```
<body>
  (6) <p>Arraste o logo do TSI para o retângulo:</p>
  <div id="div1" ondrop="soltar(event)" ondragover="permiteSoltar(event)"></div>
  <br>
  
</body>
</html>
```

(1) Tornar elemento arrastável

(2) Definir o que acontece quando o elemento é arrastado

```
<script>
  (5) function permiteSoltar(ev) {
        ev.preventDefault();
      }

  (3) function arrastar(ev) {
        ev.dataTransfer.setData("text", ev.target.id);
      }

  (7) function soltar(ev) {
        ev.preventDefault();
        var data = ev.dataTransfer.getData("text");
        ev.target.appendChild(document.getElementById(data));
      }
</script>
```

(5) Por padrão, elementos não podem ser arrastados para outros elementos. Para permitir isso, previne-se o tratamento padrão do elemento, por meio do método `event.preventDefault()` para o evento `ondragover`.

(4) Evento que especifica onde os elementos podem ser soltos.





1) **Tornar um elemento arrastável:** para tanto, defina o atributo **draggable** como **true**;

2) **ondragstart()** : Em seguida, especifique o que deve acontecer quando o elemento é arrastado.

No exemplo, o atributo **ondragstart** chama uma função, **arrastar(event)**, que especifica quais dados devem ser arrastados.

3) **dataTransfer.setData()**: define o tipo de dados e o valor dos dados arrastados.

No exemplo, o tipo de dados é "text" e o valor é o **id** do elemento arrastável ("img1").

```
ev.dataTransfer.setData("text", ev.target.id);
```

4) **ondragover()**: Evento especifica onde os dados arrastados podem ser soltos.

5) **event.preventDefault()**: evita o tratamento padrão do elemento.

Por padrão, elementos não podem ser largados para outros elementos. Para permitir isso, devemos evitar o tratamento padrão do elemento. **Isso é feito chamando o método preventDefault**

6) **ondrop**: evento que deve ser configurado para se largar o objeto

Quando ocorre o **ondrop**, e os elementos arrastados são soltos, a função atrelada a este evento é chamada.

7) Função que adiciona o elemento a **div**



HTML Web Storage API



O que é HTML Web Storage?






API que visa permitir que os aplicativos web possam **armazenar dados localmente no navegador** do usuário.



Antes do HTML5, os dados tinham que ser armazenados em **cookies**, incluídos em **todas as solicitações do servidor**.



O Web Storage é **mais seguro** e permite que grandes **quantidades de dados sejam armazenadas localmente**, sem afetar o desempenho do site.

API					
Web Storage	4.0	8.0	3.5	4.0	11.5



Objetos HTML Web Storage

Esta API fornece dois objetos para armazenar dados no cliente:

window.localStorage - armazena dados sem data de expiração.

Os dados **não serão excluídos** quando o navegador for fechado e estarão disponíveis no próximo dia, semana ou ano.



window.sessionStorage - armazena dados para uma sessão.

Os dados **são perdidos** quando a guia do navegador é fechada.



```
if (typeof(Storage) !== "undefined") {  
    // Código para localStorage/sessionStorage.  
} else {  
    // Web Storage não suportado ☹  
}
```



Antes de usar a API, é preciso verificar se o navegador a suporta



Exemplo do objeto `localStorage`

// Armazenando

```
localStorage.setItem("sobrenome", "Wayne");
```

// Recuperando

```
document.getElementById("resultado").innerHTML = localStorage.getItem("sobrenome");
```

// Armazenando

```
localStorage.sobrenome = "Wayne";
```

// Recuperando

```
document.getElementById("resultado").innerHTML = localStorage.sobrenome;
```

Forma reduzida

// Removendo

```
localStorage.removeItem("sobrenome");
```



Os pares nome/valor são **armazenados como strings**. É preciso convertê-los para outro formato quando necessário.



```
<html>

<head>
  <script>

    function contaClique() {

      if (typeof (Storage) !== "undefined") {

        if (localStorage.contador) {
          localStorage.contador= Number(localStorage.contador) + 1;
        } else {
          localStorage.contador = 1;
        }
        document.getElementById("resultado").innerHTML = "Você clicou " + localStorage.contador + " vez(es).";

      } else {
        document.getElementById("resultado").innerHTML = "Opa! Seu navegador não suporta web storage...";
      }

    }

  </script>
</head>

<body>
  <p><button onclick="contaClique()" type="button">Clique aqui...</button></p>
  <div id="resultado"></div>
  <p>Clique no botão para incrementar o contador.</p>
  <p>Feche o browser e tente novamente. O contador continuará de onde parou....</p>
</body>
</html>
```

1) Antes de usar o Web Storage, é importante verificar o suporte a API.



Código para contar o número de vezes que um usuário clicou em um botão.



```
<html>

<head>
  <script>
    function contaClique() {

      if (typeof (Storage) !== "undefined") {
        if (sessionStorage.contador) {
          sessionStorage.contador = Number(sessionStorage.contador) + 1;
        } else {
          sessionStorage.contador = 1;
        }
        document.getElementById("resultado").innerHTML = "Você clicou " + sessionStorage.contador + " vez(es).";
      } else {
        document.getElementById("resultado").innerHTML = "Opa! Seu navegador não suporta web storage...";
      }
    }
  </script>
</head>

<body>
  <p><button onclick="contaClique()" type="button">Clique aqui...</button></p>
  <div id="resultado"></div>
  <p>Clique no botão para incrementar o contador.</p>
  <p>Feche a guia do navegador. Ao tentar novamente, o contador terá sido reinicializado.</p>
</body>
</html>
```

A string de valor é convertida em um número para poder aumentar o contador.



Métodos/Propriedades do objeto Storage

Propriedade/Método	Descrição
<code>key(n)</code>	Retorna o nome da enésima chave no armazenamento
<code>length</code>	Propriedade que retorna a quantidade de itens armazenados no objeto Storage
<code>getItem(chave)</code>	Retorna o valor do nome da chave especificada
<code>setItem(chave, value)</code>	Adiciona chave ao armazenamento ou atualiza o valor da chave (se já existir)
<code>removeItem(chave)</code>	Remove essa chave do Storage
<code>clear()</code>	Esvazie todas as chaves do Storage





HTML Web Workers API






O que é um Web Worker?

É um script JavaScript executado em segundo plano, independentemente de outros scripts, **sem afetar o desempenho da página**.

Permite que o usuário continue a fazer o que quiser: clicar, selecionar coisas, etc., enquanto o **web worker é executado em segundo plano**.

API					
Web Workers	4.0	10.0	3.5	4.0	11.5





webWorkers.js

```
var i = 0;

function temporizador() {
    i = i + 1;
    postMessage(i);
    setTimeout("temporizador()", 700);
}
```

temporizador();

(2) Criar o Web Worker

(1) Validar se o navegador suporta essa API

```
<!DOCTYPE html>
<html>
<body>
    <p>Contador: <output id="resultado"></output></p>
    <button onclick="startWorker()">Inicie o Web Worker</button>
    <button onclick="stopWorker()">Pare o Web Worker</button>

    <script>
        var ww;

        function startWorker() {
            if (typeof (Worker) !== "undefined") {
                if (typeof (ww) == "undefined") {
                    ww = new Worker("webWorkers.js");
                }
                ww.onmessage = function (event) {
                    document.getElementById("resultado").innerHTML = event.data;
                };
            } else {
                document.getElementById("resultado").innerHTML = "Opa! Teu navegador não suporta Web Workers...";
            }
        }

        function stopWorker() {
            ww.terminate();
            ww = undefined;
        }
    </script>
</body>
</html>
```



webWorkers.js

```
var i = 0;

function temporizador() {
  i = i + 1;
  postMessage(i);
  setTimeout("temporizador()", 700);
}

temporizador();
```

```
<!DOCTYPE html>
<html>
<body>
  <p>Contador: <output id="resultado"></output></p>
  <button onclick="startWorker()">Inicie o Web Worker</button>
  <button onclick="stopWorker()">Pare o Web Worker</button>

  <script>
    var ww;

    function startWebWorker() {
      if (typeof (Worker) !== "undefined") {
        if (typeof (ww) == "undefined") {
          ww = new Worker("webWorkers.js");
        }
        ww.onmessage = function (event) {
          document.getElementById("resultado").innerHTML = event.data;
        };
      } else {
        document.getElementById("resultado").innerHTML = "Opa! Teu navegador não suporta Web Workers...";
      }
    }

    function stopWebWorker() {
      ww.terminate();
      ww = undefined;
    }
  </script>
</body>
</html>
```

(3) Criar um objeto do tipo Web Worker

(4) Adicionar um event listener ao ww

5) Encerra o Web Worker e libera recursos do navegador



HTML SSE API

Server-Sent Events










Server-Sent-Events

Um SSE ocorre quando uma página **obtem atualizações automaticamente de um servidor**.

Essa *feature* já existia... A diferença é que a página web **tinha que perguntar se alguma atualização estava disponível**.

Exemplos: atualizações de redes sociais, *feeds* de notícias, resultados de esportes, atualizações de preços de ações, entre outros.

API					
SSE	6.0	79.0	6.0	5.0	11.5



Recebendo notificações SSE

O objeto **EventSource** é usado para receber notificações SSE.

```
var fonte = new EventSource("exemplo_sse.php");

fonte.onmessage = function(event) {

    document.getElementById("resultado").innerHTML += event.data + "<br>";

};
```



```
<!DOCTYPE html>
```

```
<html>
```

```
<body>
```

```
<h1>Recebendo atualizações do servidor</h1>
```

```
<div id="resultado"></div>
```

(1) Validar se o navegador suporta SSE.

```
<script>
```

```
if (typeof (EventSource) !== "undefined") {
```

(2) Receber notificações enviadas pelo servidor.

```
var source = new EventSource("exemplo_sse.php");  
source.onmessage = function (event) {  
    document.getElementById("resultado").innerHTML += event.data + "<br>";  
};
```

```
} else {  
    document.getElementById("resultado").innerHTML = "Opa! Teu navegador não suporta SSE...";  
}
```

```
</script>
```

```
</body>
```

```
</html>
```

```
<?php  
header('Content-Type: text/event-stream');  
header('Cache-Control: no-cache');  
  
$time = date('r');  
echo "Dados: A hora do servidor é: {$time}\n\n";  
flush();  
?>
```




Outros eventos do objeto EventSource

Evento	Descrição
onopen	Quando uma conexão com o servidor é aberta
onmessage	Quando uma mensagem é recebida
onerror	Quando ocorre um erro



HTML5

Novas API