HTML SSE API - Server-Sent Events

Server-Sent Events (SSE API)

Server-sent events are a way of sending data from a server to a web page without requiring the page to refresh or make requests. These events are useful for creating real-time applications, such as chat, news feeds, or notifications. Using SSE, we can push DOM events continuously from our web server to the visitor's browser.

The event streaming approach opens a persistent connection to the server, sending data to the client when new information is available, eliminating the need for continuous polling. Server-sent events standardize how we stream data from the server to the client.

How to Use SSE in Web Application?

To use Server-sent events in a web application, we need to add an **<eventsource>** element to the document. The **src** attribute of the **<eventsource>** element should point to an URL that provides a persistent HTTP connection that sends a data stream containing the events. Furthermore, the URL points to a PHP, PERL, or any Python script that would take care of sending event data consistently.

Instance

Following is a sample HTML code of a web application that would expect server time:

```
<!DOCTYPE html>
<html>
<head>
   <script type="text/javascript">
      /* Define event handling logic here */
   </script>
</head>
<body>
   <div id="sse">
      <eventsource src="/cgi-bin/ticker.cgi" />
   </div>
   <div id="ticker">
      <TIME>
   </div>
</body>
</html>
```


Server-side Script for SSE

The following are the steps for sending server-sent events (SSE) from a server-side script:

1. Set the Content-Type Header

A server-side script should send **Content-Type** header specifying the type **text/event-stream** as follows.

```
print "Content-Type: text/event-stream\n\n";
```

2. Send an Event Name

After setting **Content-Type**, the server-side script would send an **Event:** tag followed by the event name. Following code snippet would send Server-Time as an event name terminated by a new line character.

```
print "Event: server-time\n";
```

3. Send Event Data

The final step is to send event data using **Data**: tag which would be followed by an integer of a string value terminated by a new line character as follows –

```
$time = localtime();
print "Data: $time\n";
```

4. Combine Steps into a Complete Script

Finally, following is a complete "ticker.cgi" written in Perl –

```
#!/usr/bin/perl
print "Content-Type: text/event-stream\n\n";
while(true){
   print "Event: server-time\n";
   $time = localtime();
   print "Data: $time\n";
   sleep(5);
}
```

Handle Server-Sent Events

You can also modify the web application to listen for and process server-sent events using an **eventsource** object. Let us modify our web application to handle server-sent events.

Example

The following example demonstrates handling server-sent events:

```
<!DOCTYPE html>
<html>
  <head>
    <script type="text/javascript">
      document.getElementsByTagName("eventsource")[0].addEventListener("server-time",
eventHandler, false);
         function eventHandler(event) {
            // Alert time sent by the server
            document.querySelector('#ticker').innerHTML = event.data;
         }
    </script>
  </head>
  <body>
    <div id="sse">
      <eventsource src="/cgi-bin/ticker.cgi" />
    </div>
    <div id="ticker" name="ticker"> [TIME] </div>
  </body>
</html>
```

Note: Before testing Server-Sent events, I would suggest that you make sure your web browser supports this concept.