Text Sync Tool - JavaScript

See file: **txt_sync.js**

The basis of the Text Sync Tool system is the browser's HTML <audio>/<video> media-player.

The JavaScript watches that media player and updates the style and the position of elements within the Transcript and Gloss display areas.

This connection, between the media-player and the JavaScript, is made in the HTML within the <audio>/<video> tag via the attributes: *onclick*, *onmousemove*, and *ontimeupdate* .

The connection is also made (redundantly) in the JavaScript at the initial setup that occurs on completion of the page load event.

```
media = document.getElementById("sync_player");

media.setAttribute("ontimeupdate", "sync(this.currentTime)");
media.setAttribute("onmousemove", "sync(this.currentTime)");
media.setAttribute("onclick", "sync(this.currentTime)");
```

Subsequently, during the script initialization, each <li> of the transcript has its *data-start* and *data-stop* attribute values read into arrays (these arrays have global scope).

Also, each of the color-coded speaker initials in the transcript display area is made clickable. When the initials are clicked, the media-player starts (and plays to the end of) the transcript segment that is contained within the same <li>:

```
ts_tag_array = document.getElementsByClassName("txt_ln");

number_of_lines = ts_tag_array.length;

for (i = 0; i < number_of_lines; i++)
{
        ts_start_time_array[i] = ts_tag_array[i].getAttribute("data-start");
        ts_stop_time_array[i] = ts_tag_array[i].getAttribute("data-stop");
        ts_tag_array[i].childNodes[0].setAttribute("onclick", "set_time_play_and_pause(" +
        ts_start_time_array[i] + ", " + ts_stop_time_array[i] + ")");
}
```

Finally, a check is made to see if the variable **initial_time** has been set to a value greater than **0**.

```
if (initial_time > 0)
{
        try { set_time_play_and_pause(initial_time, initial_time_end); }
        catch(error) { media.addEventListener("canplay", function() {
        set_time_play_and_pause(initial_time, initial_time_end); },true); }
}
```

If **initial_time** is greater than **0**, the synchronized transcript and media-player will jump to that position and play until **initial_time_end** is reached (see JavaScript at the end of the <body> of the HTML; also see HTML and PHP documentation).

After the initialization, the JavaScript provides two functions: **set_time_play_and_pause**, and **sync**.

set_time_play_and_pause updates the media-player's position in the audio/video file. It takes two parameters: *start_time*, and *end_time*.

When **set_time_play_and_pause** is called, no matter if the media-player is running or stopped, the media-player time will be set to *start_time* and play until *end_time* is reached.

```
function set_time_play_and_pause(start_time, end_time)
{
        media.pause();
        clearTimeout(sub_time);
        play_time = Math.ceil((end_time - start_time) * 1000);
        media.currentTime = start_time;
        media.play();
        sub_time = setTimeout(function() { media.pause(); }, play_time);
}
```

Note: **sub_time** and **media** have global scope.

sync is called by the media-player to update the transcript display area. As the media-player plays it calls the **sync** function periodically and passes it the value of the current playtime.

sync first checks the dimensions of the of the scrolling transcript display area. The dimensions may change on a page resize event, so this check is done within **sync** rather than during initialization.

```
var txt_lns_rect = document.getElementById("txt_lns").getBoundingClientRect();
var mid_point = txt_lns_rect.top + ((txt_lns_rect.bottom - txt_lns_rect.top) / 2);
```

Also, there is a "magic number" hardcoded at this stage: **470**

```
var max_scroll = (document.getElementById("txt_lns").scrollHeight - 470);
```

470 is the height in pixels of the transcript display area minus 2 pixels for the top and bottom border (specified in the CSS file, **txt_sync.css**).

When the ETST was being developed, determining this value programmatically with **txt_lns_rect.height** was problematic due to standards compliance differences between browsers.

As it is, a change in the size of the transcript display area will require editing this hardcoded value.

sync then iterates over the arrays of transcript <li> objects, looking for the <li> with *data-start* and *data-stop* attribute values that match the current playtime:

```
if ((current_time >= parseFloat(ts_start_time_array[i])) && (current_time <=
parseFloat(ts_stop_time_array[i])))
```

A matching <li> is assigned an *id* attribute and value for the CSS to give it a background color:

```
ts_tag_array[i].setAttribute("id", "txt-current_" + i);
```

The <li> is positioned in the center of the scrolling display area:

```
while((document.getElementById("txt_lns").scrollTop < max_scroll) &&
((ts_tag_array[i].getBoundingClientRect().top +
((ts_tag_array[i].getBoundingClientRect().bottom -
ts_tag_array[i].getBoundingClientRect().top) / 2)) > mid_point) )
{
        document.getElementById("txt_lns").scrollTop++;
}
```

Lastly, a style attribute for the Gloss display is conditionally assigned:

```
ref_id = ts_tag_array[i].childNodes[2].getAttribute("id");

if(document.getElementById("r" + ref_id))
{
        document.getElementById("r" + ref_id).style.display = "block";
}
```

If the <li> *data-start* and *data-stop* attribute values do not match the current playtime, then its CSS/style attributes are cleared/reset:

```
ts_tag_array[i].removeAttribute("id");

document.getElementById("r" + ref_id).style.display = "none";
```