HOA SEN UNIVERSITY

www.hoasen.edu.vn

HTML5 (cont)

Phạm Thị Kim Ngôn
ngon.phamthikim@hoasen.edu.vn

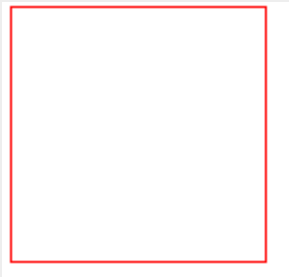
Content

- Canvas
- Media
- Geolocation
- Drag/drop
- Web Storage
- Server-Sent Events
- Web Workers

Canvas (1)

- HTML5 element <canvas> gives you an easy and powerful way to draw graphics using JavaScript.
- It can be used to draw graphs, make photo compositions or do simple animations.

```
<!DOCTYPE HTML>
<html>
    <head>
    <style>
        #mycanvas{border:1px solid red;}
     </style>
    </head>
    <body>
        <canvas id = "mycanvas" width = "100" height = "100"></canvas>
    </body>
</html>
```



Canvas (2)

```
var canvas = document.getElementById("mycanvas");
if (canvas.getContext) {
    var ctx = canvas.getContext('2d');
    // drawing code here
} else {
    // canvas-unsupported code here

}
```

Canvas (3)

1	Drawing Rectangles	9	Pattern and Shadow
2	Drawing Paths	10	Canvas States
3	Drawing Lines	11	Canvas Translation
4	Drawing Bezier	12	Canvas Rotation
5	Using Images	13	Canvas Scaling
6	Create Gradients	14	Canvas Transform
7	Styles and Colors	15	Canvas Composition
8	Text and Fonts	16	Canvas Animation

https://www.w3schools.com/tags/canvas_rect.asp

HTML Media (1)

- Audio

```
<audio controls>
  <source src="horse.ogg" type="audio/ogg">
  <source src="horse.mp3" type="audio/mpeg">
Your browser does not support the audio element.
</audio>
```

- Video

```
<video width="320" height="240" controls>
  <source src="movie.mp4" type="video/mp4">
  <source src="movie.ogg" type="video/ogg">
Your browser does not support the video tag.
</video>
```

HTML Media (2)

- HTML Helpers (Plug-ins)
  - Helper applications (plug-ins) are computer programs that extend the standard functionality of a web browser.
  - Plug-ins can be added to web pages with the **<object>** tag or the **<embed>** tag.
  - Plug-ins can be used for many purposes: display maps, scan for viruses, verify your bank id, etc.

HTML Media

- The **<object>** Element
  - This element is supported by all browsers
  - Defines an embedded object within an HTML document
  - It is used to embed plug-ins (like Java applets, PDF readers, Flash Players) in web pages.

```
<object width="400" height="50" data="bookmark.swf"></object>
```

```
<object width="100%" height="500px" data="snippet.html"></object>
```

```
<object data="audi.jpeg"></object>
```

HTML Media (3)

- The **<embed>** Element
  - is supported in all major browsers
  - Defines an embedded object within an HTML document
  - Web browsers have supported the <embed> element for a long time. However, it has not been a part of the HTML specification before HTML5.

```
<embed width="400" height="50" src="bookmark.swf">
```

```
<embed width="100%" height="500px" src="snippet.html">
```

```
<embed src="audi.jpeg">
```

HTML Media (4)

- HTML YouTube Videos

```
<iframe width="420" height="315"
src="https://www.youtube.com/embed/tgbNymZ7vqY">
</iframe>
```

- YouTube – Autoplay

```
<iframe width="420" height="315"
src="https://www.youtube.com/embed/tgbNymZ7vqY?autoplay=1">
</iframe>
```

- YouTube - Loop

```
<iframe width="420" height="315"
src="https://www.youtube.com/embed/tgbNymZ7vqY?playlist=tgbNy
mZ7vqY&loop=1">
</iframe>
```

HTML Media (5)

- YouTube Controls

```
<iframe width="420" height="315"
src="https://www.youtube.com/embed/tgbNymZ7vqY?controls=0">
</iframe>
```

  - Value 0: Player controls does not display.
  - Value 1 (default): Player controls display.

- YouTube - Using <object> or <embed>

```
<object width="420" height="315"
data="https://www.youtube.com/embed/tgbNymZ7vqY">
</object>
```

```
<embed width="420" height="315"
src="https://www.youtube.com/embed/tgbNymZ7vqY">
```

HTML APIs

HTML APIs

- Geolocation
- Drag/drop
- Server-Sent Events
- Web Storage

HTML5 Geolocation (1)

- The HTML Geolocation API is used to locate a user's position.
- The position is not available unless the user approves it.

```
<script>
var x = document.getElementById("demo");
function getLocation() {                                    (showPosition, showError)
    if (navigator.geolocation) {
        navigator.geolocation.getCurrentPosition(showPosition);
    } else {
        x.innerHTML = "Geolocation is not supported by this browser.";
    }
}
function showPosition(position) {
    x.innerHTML = "Latitude: " + position.coords.latitude +
    "<br>Longitude: " + position.coords.longitude;
}
</script>
```

HTML5 Geolocation (2)

```
function showError(error) {
    switch(error.code) {
        case error.PERMISSION_DENIED:
            x.innerHTML = "User denied the request for
Geolocation."
            break;
        case error.POSITION_UNAVAILABLE:
            x.innerHTML = "Location information is unavailable."
            break;
        case error.TIMEOUT:
            x.innerHTML = "The request to get user location
timed out."
            break;
        case error.UNKNOWN_ERROR:
            x.innerHTML = "An unknown error occurred."
            break;
    }
}
```

HTML Drag/Drop (1)

```
<body>

<div id="div1" ondrop="drop(event)" ondragover="allowDrop(event)">
</div>

<img id="drag1" src="img_logo.gif" draggable="true"
ondragstart="drag(event)" width="336" height="69">

</body>
```

- Demo

HTML Drag/Drop (2)

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

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

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

Web Storage

- HTML5 introduces two mechanisms, similar to HTTP session cookies, for storing structured data on the client side and to overcome following drawbacks.
  - Cookies are included with every HTTP request, thereby slowing down your web application by transmitting the same data.
  - Cookies are included with every HTTP request, thereby sending data unencrypted over the internet.
  - Cookies are limited to about 4 KB of data. Not enough to store required data.
- The two storages are **session storage** and **local storage** and they would be used to handle different situations.
- The latest versions of pretty much every browser supports HTML5 Storage including Internet Explorer.

Session Storage

- The *Session Storage* is designed for scenarios where the user is carrying out a single transaction, but could be carrying out multiple transactions in different windows at the same time.

```
<!DOCTYPE HTML>
<html>
    <body>
        <script type = "text/javascript">
            if( sessionStorage.hits ) {
                sessionStorage.hits = Number(sessionStorage.hits) +1;
            } else {
                sessionStorage.hits = 1;
            }
            document.write("Total Hits :" + sessionStorage.hits );
        </script>
        <p>Refresh the page to increase number of hits.</p>
        <p>Close the window and open it again and check the result.</p>
    </body>
</html>
```

Local Storage

- The *Local Storage* is designed for storage that spans multiple windows, and lasts beyond the current session. In particular, **Web applications may wish to store megabytes of user data**, such as entire user-authored documents or a user's mailbox, on the client side for performance reasons.

Example - Local Storage

```
<!DOCTYPE HTML>
<html>
<body>
        <script type = "text/javascript">
            if( localStorage.hits ) {
                localStorage.hits = Number(localStorage.hits) +1;
            } else {
                localStorage.hits = 1;
            }
            document.write("Total Hits :" + localStorage.hits );
        </script>
        <p>Refresh the page to increase number of hits.</p>
        <p>Close the window and open it again and check the result.</p>
</body>
</html>
```

Delete Web Storage

- Storing sensitive data on local machine could be dangerous and could leave a security hole.
- The *Session Storage Data* would be deleted by the browsers immediately after the session gets terminated.
- **Clear a local storage**
  - **localStorage.remove('key')**;

    'key' is the key of the value you want to remove.
  - **localStorage.clear()** (clear all settings)

Server-Sent Events (1)

- Server-Sent Events allow a web page to get updates from a server.

  Examples: Facebook/Twitter updates, stock price updates, news feeds, sport results, etc.

6.0	Not supported	6.0	5.0	11.5

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

Server-Sent Events (2)

- Check Server-Sent Events Support

```
if(typeof(EventSource) !== "undefined") {
    // Yes! Server-sent events support!
    // Some code.....
} else {
    // Sorry! No server-sent events support..
}
```

- The EventSource Object

Events	Description
onopen	When a connection to the server is opened
onmessage	When a message is received
onerror	When an error occurs

Server-Sent Events (3)

- Server-Side Code Example (Code in PHP)
  - Set the "Content-Type" header to "text/event-stream“
  - Output the data to send (**Always** start with "data: ")
  - Flush the output data back to the web page

```
<?php
header('Content-Type: text/event-stream');
header('Cache-Control: no-cache'); // recommended to prevent
//caching of event data.

$time = date('r');
echo "data: The server time is: {$time}\n\n";
flush();
?>
```

- Demo

Server-Sent Events (4)

- The response contain a "data:" line

```
data: My message\n\n
```

- The response contain multiple "data:" lines

```
data: first line\n
data: second line\n\n
```

- Send JSON Data

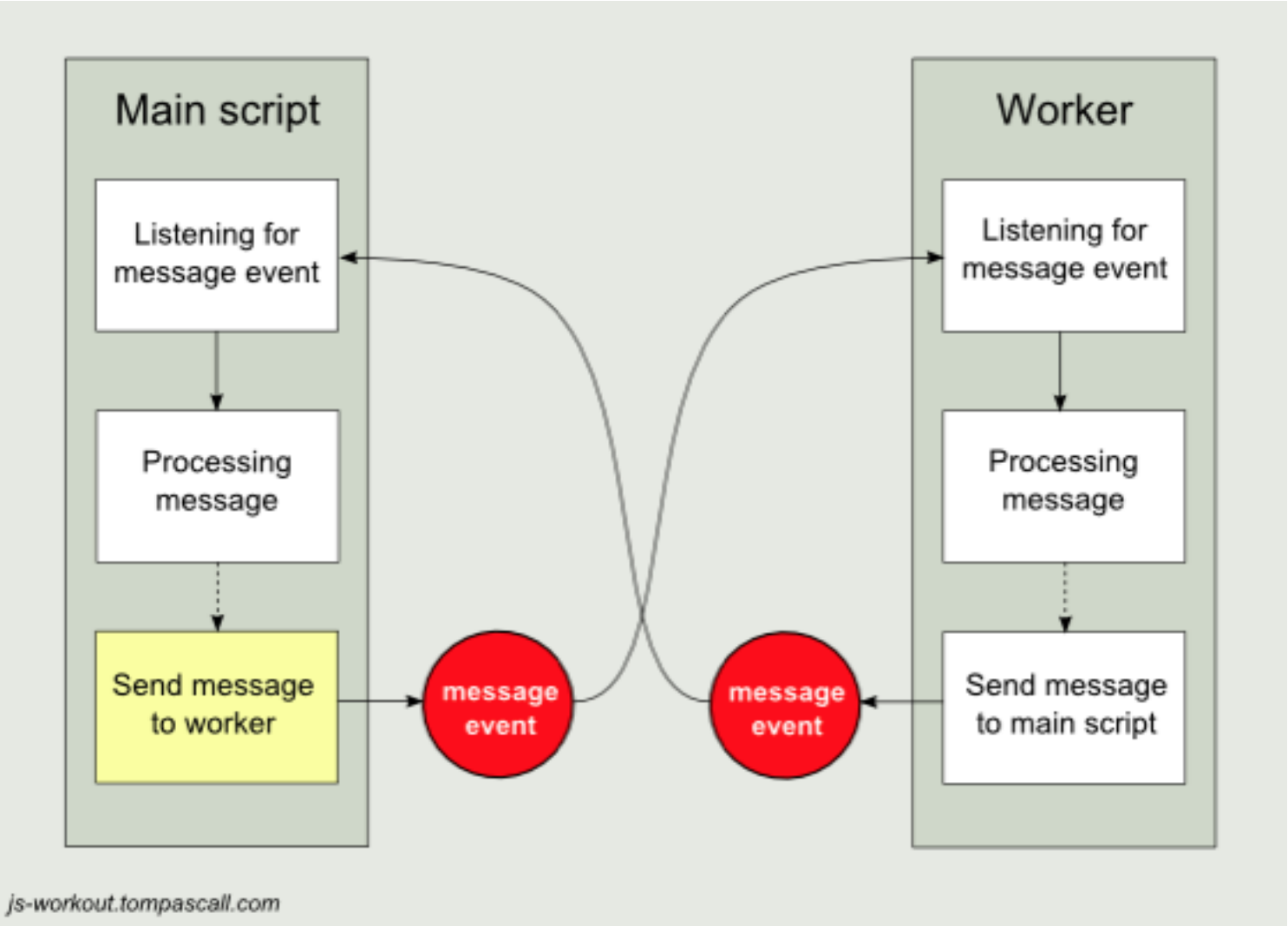
```
data: {\n
data: "msg": "hello world",\n
data: "id": 12345\n
data: }\n\n
```

Web Workers (1)

- A web worker is a JavaScript running in the background, independently of user-interface scripts without affecting the performance of the page.
- When web workers run in the background, they do not have direct access to the DOM but communicate with the document by message passing. This allows for multi-threaded execution of JavaScript programs.
- Check Web Worker Support

```
if (typeof(Worker) !== "undefined") {
    // Yes! Web worker support!
    // Some code.....
} else {
    // Sorry! No Web Worker support..
}
```

Model event for web workers



Example - Web Workers (2)

- demo_workers.js

```
var i = 0;
function timedCount() {
    i = i + 1;
    postMessage(i);
    setTimeout("timedCount()",500);
}
timedCount();
```

- Create a Web Worker Object

```
if (typeof(w) == "undefined") {
    w = new Worker("demo_workers.js");
}
```

Example - Web Workers (3)

- Add an "onmessage" event listener to the web worker.

```
w.onmessage = function(event){
    document.getElementById("result").innerHTML = event.data;
};
```

- Terminate a Web Worker

```
w.terminate();
```

- Reuse the Web Worker
  - If you set the worker variable to undefined, after it has been terminated, you can reuse the code:

```
w = undefined;
```

- Demo

References

- https://www.w3schools.com
- https://www.tutorialspoint.com