Mobile

Lecture 4 – HTML5 –Web Storage, geolocation, (SSE) and Web Workers

Semester I

Canvas

- Drawing graphics using JS
  - http://www.w3schools.com/tags/ref_canvas.asp

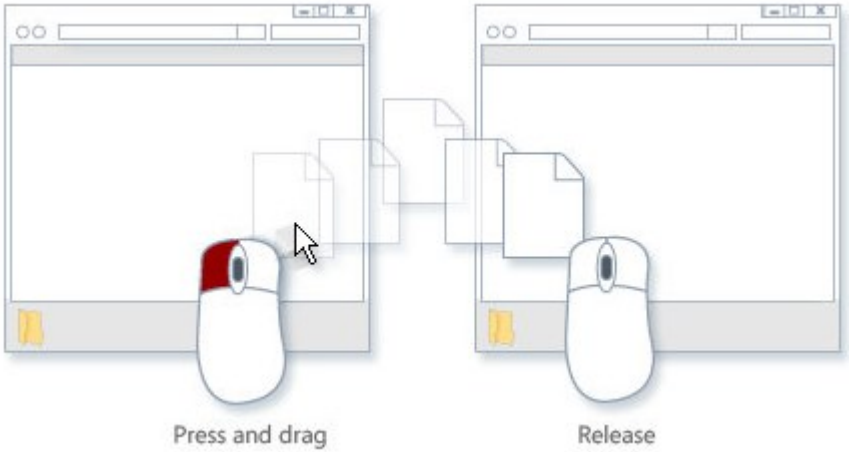
*Some of the stuff here is from http://www.w3schools.com

SVG

- SVG stands for Scalable Vector Graphics.
- SVG defines graphics in XML format.
  - http://www.w3schools.com/svg/svg_examples.asp

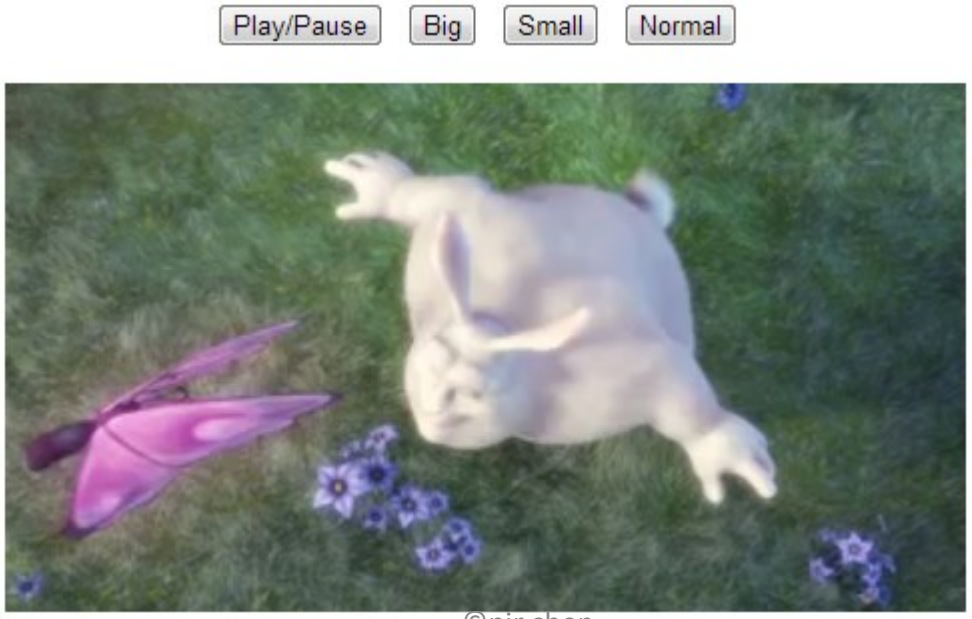
Drag and Drop

- The possibility to move objects around in the screen
  - http://www.w3schools.com/html/html5_draganddrop.asp



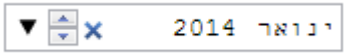
Video And Audio

- What can I write here?
  - http://www.w3schools.com/html/html5_video.asp
  - http://www.w3schools.com/html/html5_audio.asp



Input Types

- color
- date
- datetime
- datetime-local
- email
- month
- number
- range
- search
- tel
- time
- url
- week



Form Attributes

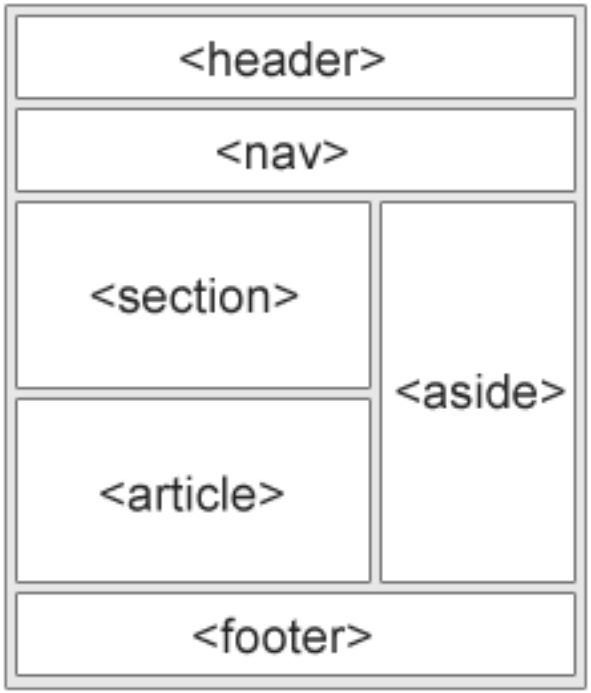
- New attributes for <form>:
  - autocomplete
  - novalidate

- New attributes for <input>:
  - autocomplete
  - autofocus
  - form
  - formaction
  - formenctype
  - formmethod
  - formnovalidate
  - formtarget
  - height and width
  - list
  - min and max
  - multiple
  - pattern (regexp)
  - placeholder
  - required
  - step

Semantic Elements

HTML5 offers new semantic elements to clearly define different parts of a web page:

- <header>
- <nav>
- <section>
- <article>
- <aside>
- <figcaption>
- <figure>
- <footer>



Web Storage

- Web storage is a key\value pair that is located on the clients browser.
  - Can contain 5 MB or more depend on the browser (2 KB for cookie)
  - Does not automatically included in the server request but only if needed
  - Faster and securer then cookie. The web page can access only the one stored by the same web site
  - Can contain only string. But we can use JSON to save objects
- Two kinds of storage:
  - `localStorage` – no expiration date
  - `sessionStorage` – expires when the window will be closed

localStorage

- First we need to check if the browser supports the web storage

=== or !== has no type casting , preferred!

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

- If the browser supports the storage then we can use the *localStorage* like this for saving and using the data

```
localStorage.LSname = document.getElementById("LSname").value;
document.getElementById("LSlblname").innerText = localStorage.LSname;
```

- The following is the same as above

```
localStorage.setItem("LSname", document.getElementById("LSname").value);
document.getElementById("LSlblname").innerText = localStorage.getItem("LSname");
```

sessionStorage

- First we need to check if the browser supports the web storage

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

- If the browser supports the storage then we can use the *sessionStorage* like this for saving and using the data

```
sessionStorage.Sesname = document.getElementById("SesName").value;
document.getElementById("SesLblname").innerText = sessionStorage.Sesname;
```

- The following is the same as above

```
sessionStorage.setItem("Sesname", document.getElementById("SesName").value);
document.getElementById("SesLblname").innerText =
sessionStorage.getItem("Sesname");
```

Try it yourself

webStorage.html

web storage - local storage

user name:

nir

user name: nir

web storage - session storage

user name:

chen

user name: chen

submit

webStorage2.html

webStorage2.html

Hello nir

back

Object storage

- If we need to store and object we can use the JSON (JavaScript Object Notation) functions. To store:

```
var myPerson = {
    name: document.getElementById("LSname").value,
    theDate: new Date()
};
localStorage.LSnameObject = JSON.stringify(myPerson);
```

- To read the stored info:

```
if (JSON.parse(localStorage.LSnameObject) != null) {
…
```

Try it yourself

webStorage Objects.html

web storage - local storage

user name:

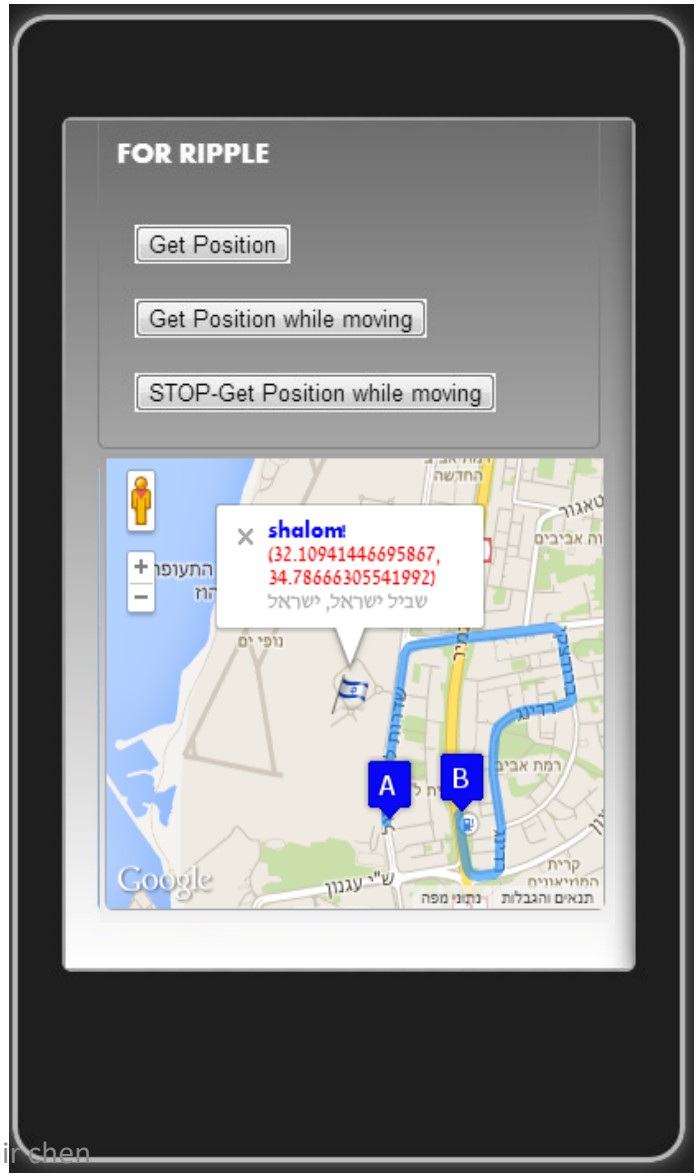
avi

user name: avi, 2013-09-01T12:54:59.385Z

submit

Using JSON and Object

GeoLocation



GeoLocation cont’

- The possibility to get the coordinates of the users position.
- Using Google Map API we can show the world map and add to it a lot of cool stuff like the direction between two places, street names and more… https://developers.google.com/maps/documentation/javascript/
- Lets start with the basics

```
<script src="http://maps.google.com/maps/api/js?sensor=false&language=he"></script>
<script
src="https://maps.googleapis.com/maps/api/js?key=YOUR_API_KEY&callback=initMap"
async defer></script>
```

getCurrentPosition

- First we need to check that the device has the geo location features and then we can call the *getCurrentPosition* function

```
if (navigator.geolocation) {
        navigator.geolocation.getCurrentPosition(showPosition, showError);
}
else { alert("Geolocation is not supported by this browser."); }
```

- If the device has geolocation then go to function *showPosition* else go to *showError*

```
function showPosition(position) {
    lat = position.coords.latitude;
    lon = position.coords.longitude;
    latlon = new google.maps.LatLng(lat, lon)
    var myOptions = {
        center: latlon,
        zoom: 14,
        mapTypeId: google.maps.MapTypeId.ROADMAP,
        mapTypeControl: false,
        navigationControlOptions: { style:
google.maps.NavigationControlStyle.SMALL }
    };
var map = new google.maps.Map(document.getElementById("mapholder"), myOptions);
```

Use in the Ripple Emulator

showError

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

```
<div id="mapholder"></div>
<!--can not be inside a section because it doesn't work then!!!  -->
```

watchPosition

- Instead of getting the position only once, we can get it automatically updated as we move using the *watchPosition* function

```
watchGeoMarkerProcess = navigator.geolocation.watchPosition(showPosition,
showError);
```

- In order to stop the automatic position updaeting we can use the *clearWatch* function

```
navigator.geolocation.clearWatch(watchGeoMarkerProcess);
```

Marker

- *Marker* is an icon on top of the map. We can place as many markers as we want

```
google.maps.event.addListener(map, 'click', function (e) {
…e.latLng…

var marker = new google.maps.Marker({
    position: position,
    map: map,
    title: "You are here!",
    icon: iconImage
});
…
map.panTo(position);
```

Will slide the position to center the map.(if within the same area already on screen)
You can use also
map.setCenter(latlon);
But this wont slide

```
if (marker != null) {
    marker.setMap(null);
}
…
map.setCenter(pos);
…
```

Remove a marker

Don't use the ripple emulator from here…

Infowindow

- On the marker we can open an *Infowindow* that can contain any html information.

```
var contentStr =
'<div id="markerDiv" style="color:blue;font-family:Aharoni;f…
    'shalom!' +
…
'</div>'

var infowindow = new google.maps.InfoWindow({
    content: contentStr
});

google.maps.event.addListener(marker, 'click', function () {
    infowindow.open(map, marker);
});
```

geocode service : address

- We can use the service that gives as the LatLng using the address as input.

```
var geocoder = new google.maps.Geocoder();
…

geocoder.geocode({ 'address': address }, function (results, status) {
    if (status == google.maps.GeocoderStatus.OK) {
        map.setCenter(results[0].geometry.location);
```

תל אביב

LatLng

geocode service : LatLng

- We can use the service that gives as the address using the LatLng as input.

```
var geocoder = new google.maps.Geocoder();
…

geocoder.geocode({ 'latLng': position }, function (results, status) {
    if (status == google.maps.GeocoderStatus.OK) {
        if (results[0]) {
            address = results[0].formatted_address;
```

LatLng

יהודה בורלא 24,
תל אביב יפו,
ישראל

DirectionsService

- The user can get the directions route between point A and point B on the map using the

```
var directionsDisplay;
var directionsService = new google.maps.DirectionsService();
…
directionsDisplay = new google.maps.DirectionsRenderer({
    suppressMarkers: true
});
…
map= …
directionsDisplay.setMap(map);
…

var request = {
    origin: routePointA,
    destination: routePointB,
    travelMode: google.maps.DirectionsTravelMode.DRIVING
};
directionsService.route(request, function (response, status) {
    if (status == google.maps.DirectionsStatus.OK) {
        directionsDisplay.setDirections(response);
    }
});
```

GeoLocation cont’

Ripple Emulator (Beta)

★★★★☆ (254) | Developer Tools | from ripple.tinyhippos.com | 92,649 users

ADDED TO CHROME

OVERVIEW | DETAILS | REVIEWS | RELATED

+1 5

A browser based html5 mobile application development and testing tool

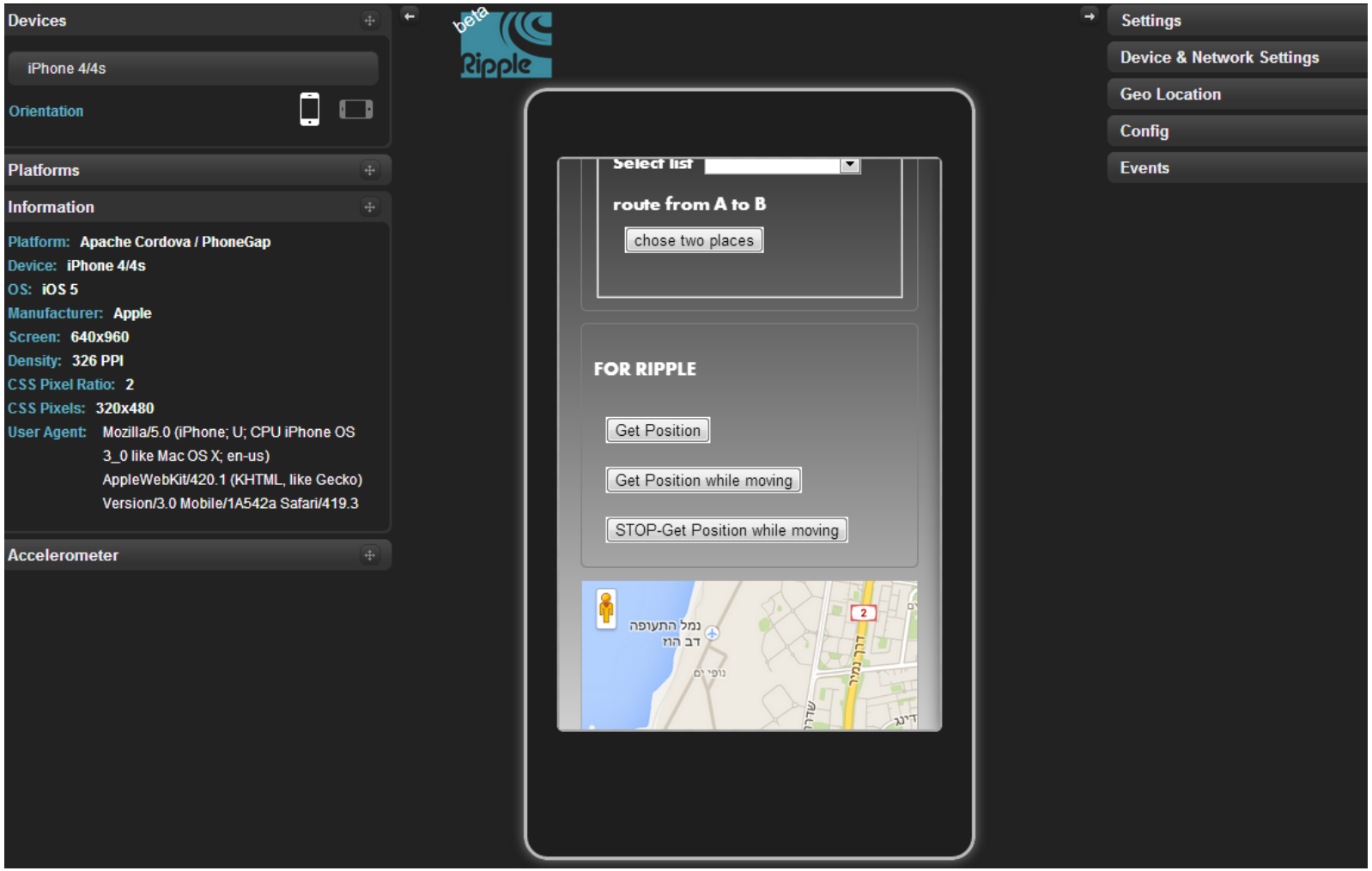
Welcome to Ripple, The Mobile Environment Emulator!

Ripple is a multi-platform mobile environment emulator that is custom-tailored to mobile HTML5 application development and testing. Ripple aims to reduce the challenges being faced by mobile developers caused by today's platform fragmentation in the marketplace.

Ripple is targeted towards WebWorks, PhoneGap, and mobile web development and testing!

Ripple offers the ability to look under the hood of your mobile application, giving you full visibility into what it is doing. It also allows for the use of existing tools to perform JavaScript debugging, HTML DOM inspection, automated testing, as well as multiple device and screen resolution

GeoLocation cont’



back

SSE-Server-Sent Event Notifications

- The SSE is a push notification mechanism that allow the browser to get updates from the server in an automatic way.
  - Automatic updates every ~ 5 seconds
  - The server can push updates in any frequency, also shorter then 5 seconds
  - String format updates
  - Object via JSON
  - Needs to be run through the VS in order to go through the local server

EventSource – Client side

- First we check the ability to use SSE

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

- Now we can initialize the *EventSource* and call define the callback function

```
source = new EventSource("TimeEventsRaiser.aspx");
source.onmessage = function (e) {
    var obj = JSON.parse(e.data);
…
source.close();
```

The link to the server file, in our case .aspx

To close the connection

Where the data will be found

Response- Server side

- In the .aspx file we have only the page directive

```
<%@ Page Language="C#" AutoEventWireup="true" CodeFile="TimeEventsRaiser.aspx.cs"
Inherits="Html5_SEvents_EventsRaiser" %>
```

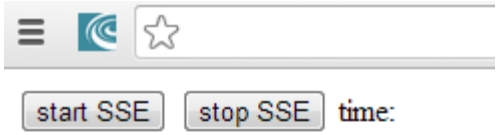
- In the .aspx.cs file we code our response using the *page_load* function

```
if (!IsPostBack)
{
    Response.ContentType = "text/event-stream";
    Response.Buffer = false;
    Response.Expires = -1;

    Response.Write("data: {\"time\": \"" + DateTime.Now.ToLongTime…
    Response.Write("\n\n");
    Response.Flush();
    Response.End();
}
```

Response only the following string (JSON) and not the whole page

The JSON string to send back

Try it yourself!



start SSE stop SSE time:

0, 14:42:05
1, 14:42:06
2, 14:42:07
3, 14:42:08
4, 14:42:09
0, 14:42:13
1, 14:42:14
2, 14:42:15
3, 14:42:16
4, 14:42:17
0, 14:42:21
1, 14:42:22
2, 14:42:23
3, 14:42:24
0, 14:42:28
1, 14:42:29
2, 14:42:30
0, 14:42:33
1, 14:42:34

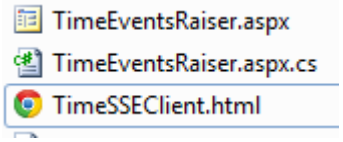
Start – it will get the time every second for five times. Then after ~ 4 seconds the client will ask the server again for info and again 5X 1 second

stop

start

stop

start



Custom event

- We can send the event with a specific name – client side

```
        source.addEventListener("ping", EventInvoked, false);
        …
}
function EventInvoked(e) {
    var obj = JSON.parse(e.data);
```

The event name

- Server side

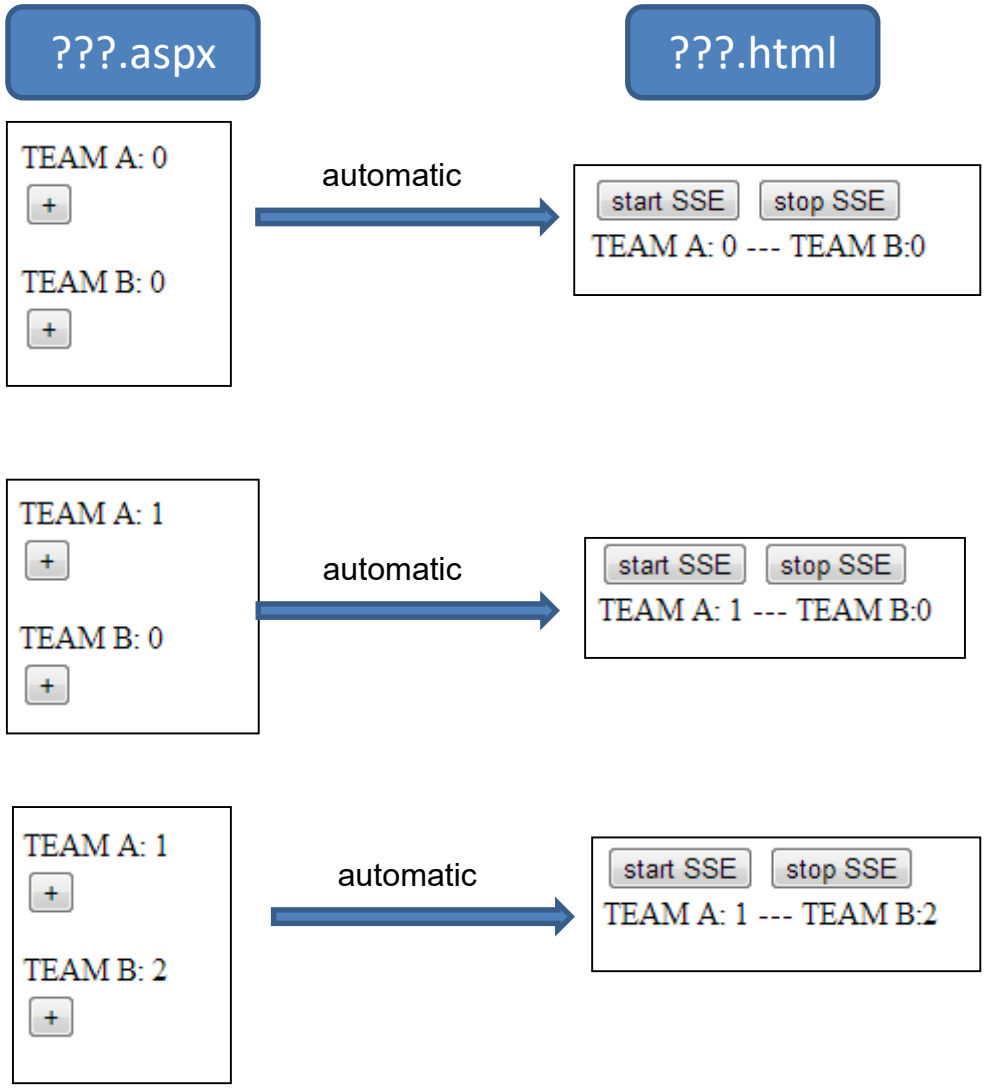
```
if (!IsPostBack)
{
    Response.ContentType = "text/event-stream";
    Response.Buffer = false;
    Response.Expires = -1;

    Response.Write("event: ping\n");  //for custom event
    Response.Write("data: {\"nameStr\": \"" + S…
    Response.Write("\n\n");
    Response.Flush();
    Response.End();
}
```

The event name

Try it yourself!

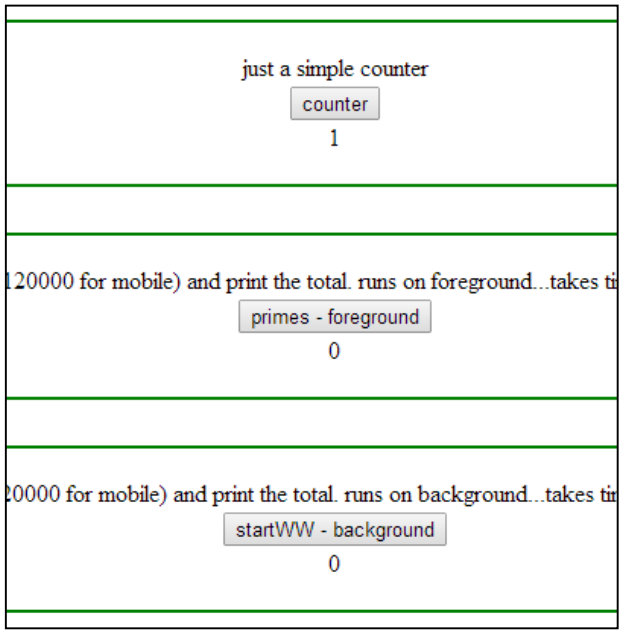
- We want to update the client page automatically with information from the server page that can be filled with the current score in a football game between team A and team B.
  - Use three files 1 html and 2 aspx
  - Use session if you need
  - Need to run two of the files. The client file to see the scores and the server file to update them

Score EventSource



Web Workers

- The Web Worker is a way of creating thread in JS. You can create a web worker that will work in parallel to other parts of your code and so the program will not “freeze” doing some time consuming tasks.
- The Web Worker resides in a separate JS file
- We can send some parameters to the WW and can get a response back



Web Workers

just a simple counter

counter

1

count the primes from 2-8000000 (put 120000 for mobile) and print the total. runs on foreground...takes time and "freezes" the program for ~5 seconds

primes - foreground

0

count the primes from 2-8000000 (put 120000 for mobile) and print the total. runs on background...takes time and DOESN'T "freezes" the program at all!

startWW - background

0

Web Worker

- To call the WW we need the following code. Pay attention that here we pass some parameter to the WW.

```
var w;
function WWprimesCount(num) {
    if (typeof (Worker) !== "undefined") {
        if (typeof (w) == "undefined") {
            w = new Worker("WebWorker.js");
        }
        w.postMessage({ "args": num });
        w.onmessage = function (event) {
            document.getElementById("WWprimesStr").innerHTML = event.data;
        };
    }
    else {
        document.getElementById("WWprimesStr").innerHTML = "Sorry, you br…
    }
}
```

The WW file

Send info to the WW

Get info back from the WW

Web Worker

- The WW that get's a parameter with info and returns the data back

```
self.addEventListener("message", function (e) {
    var args = e.data.args;
    primesCount(args);
}, false);

function primesCount(num) {
    var count = 0;
    for (var i = 2; i <= num; i++) {
        if (isPrime1(i)) {
            count++;
        }
    }
    postMessage(count);
}
```

Get the info from the main process

Send the data back