# Lezione 7: JavaScript e HTML

# JavaScript and HTML

- Web Storage
- Drag & Drop
- Data-attribute (SOLO LETTURA)
- HTML 5 audio & video (SOLA LETTURA)
- HTML5 Canvas (SOLO LETTURA)
- HTML 5 Geo location (SOLO LETTURA)

# Web storage

# Web storage

- HTML5 supporta memorizzazione dei dati ottenuti da internet nella propria macchina desktop o smartphone

- Due modalità
  - Application caching
    - Salva la logica applicativa e l'interfaccia utente
  - Offline storage
    - Cattura dati generati dall'utente e risorse di interesse per l'utente

- Ci concentreremo sull'offline storage
  - Da semplici coppie chiave/valore, a file, a interi database SQL

# Web storage

- Dati memorizzati sono connessi alla loro origine
- Di solito dati possono essere acceduti solo da un dominio
- Dati possono essere acceduti da tutte le applicazioni di quel dominio

# Web storage

- Web Storage API definisce uno standard per salvare dati localmente su un computer utente o su un device
- Prima dell'avvento dello standard Web Storage, gli sviluppatori web memorizzavano le informazioni in cookie, o tramite plugin
- Con Web Storage abbiamo un meccanismo standardizzato per memorizzare fino a 5-10MB di dati creati dal sito web o dall'applicazione web
- Web Storage è importante per lo sviluppo di Offline Web Application
  - C'è la necessità di memorizzare i dati utente mentre si lavora offline, e il Web Storage fornisce tale spazio

# Offline storage

- Due tipi di storage
  - Session storage
  - Local storage

# Offline storage

- Session storage permette di memorizzare dati specifici a una finestra/tab
  - Permette di isolare informazioni in ogni finestra
  - Se l'utente visita lo stesso sito in due finestre diverse, ogni finestra avrà il suo session storage individuale con dati separate e distinti
- Session storage non è persistente
  - Viene mantenuto solo per la durata della sessione utente in uno specific sito
  - Viene mantenuto per il tempo in cui la finestra/tab del browser è aperta e visualizza il sito

# Offline storage

- Local Storage
  - A differenza del session storage, local storage permette di salvare dati persistenti sul computer dell'utente, attraverso il browser
  - Quando un utente rivisita il sito successivamente, ogni dato salvato nel local storage può essere recuperato

# Offline storage

- Dati memorizzati come coppie chiave-valore (nel passato esisteva memorizzazione in DB)
  - key: name, value: suresh
  - key: trainer, value: html5
  - key: email, value: sureshjain17@gmail.com
- Simili ai cookie
- Dati non sono mai trasmessi al server
- HTML5 storage permette di accumulare una gran quantità di dati (5-10Mb)

# Local storage vs cookie

- Local storage possono essere scambiati a una prima occhiata per HTTP cookie
- Alcune differenze chiave
  - Cookie sono letti lato server, mentre local storage sono disponibili solo lato client
  - Se l'esecuzione del codice lato server dipende sul valore di alcune variabili, i cookie sono la soluzione ideale
    - Cookie inviati con ogni richesta al server portando overhead sostanziali in termini di banda usata
  - Local storage risiede sull'hard disk dell'utente, e viene letto a zero costo
  - Local storage fornisce molto spazio di memorizzazione
    - Cookie memorizza fino a 4KB di informazione
    - Local storage memorizza fino a 5-10MB di informazione

# Web storage

- Get e set di dati
- I metodi principali del Web Storage definiti in un oggetto chiamato Storage
  - interface Storage {

```
readonly attribute unsigned long length;
DOMString key(in unsigned long index);
getter any getItem(in DOMString key);
setter creator  void setItem(in DOMString key, in any value);
delete void removeItem(in DOMString key);
void clear();
```

};

# Metodi Local Storage

| | |
|---|---|
| setItem(key,value) | Salva un valore con una nuova chiave o ne aggiorna uno con una chiave esistente |
| getItem(key) | Accede a un valore con una determinata chiave |
| removeItem(key) | Cancella la entry con chiave key dal data store |
| clear() | Cancella tutte le entry dal data store per l'applicazione corrente |
| length | Proprietà read only che ritorna il numero di chiavi nel data store |

# Metodi getItem e setItem

- Memorizziamo una coppia chiave/valore nel local storage o session storage attraverso la funzione setItem e ne recuperiamo il valore attraverso la funzione getItem
- Se vogliamo memorizzare o recuperare dati dal session storage, usiamo setItem o getItem sull'oggetto globale sessionStorage
- Se vogliamo usare il local storage, chiamiamo setItem o getItem sull'oggetto globale localStorage
  - Ad esempio, localStorage.setItem("size", "6"); salva il valore 6 sotto la chiave size
  - Con var size = localStorage.getItem("size"); recuperiamo il valore 6
    - Scorciatoia var size = localStorage["size"];
  - Per convertire il valore in un intero
    - var size = parseInt(localStorage.getItem("size"));

# Esempio

```
<script>
  sessionStorage.nome="Pippo";
  sessionStorage.setItem("cognome", "Pluto");
alert(sessionStorage.getItem("cognome"))
</script>
```

- Memorizzo in session storage e poi riprendo un valore e lo stampo a video (sessionStorage.html)
- Visualizzabile all'interno di developer tookit di Chrome
  - Menù -> more tools –> developer tools

# Esempio

```
<script>
  sessionStorage.nome="Pippo";
  sessionStorage.setItem("cognome", "Pluto");
chiave=sessionStorage.key(0);
  alert(sessionStorage.getItem(chiave));
</script>
```

- Memorizzo in session storage e poi riprendo un valore e lo stampo a video (sessionStorage2.html)
- Visualizzabile all'interno di developer tookit di Chrome
  - Menù -> more tools –> developer tools

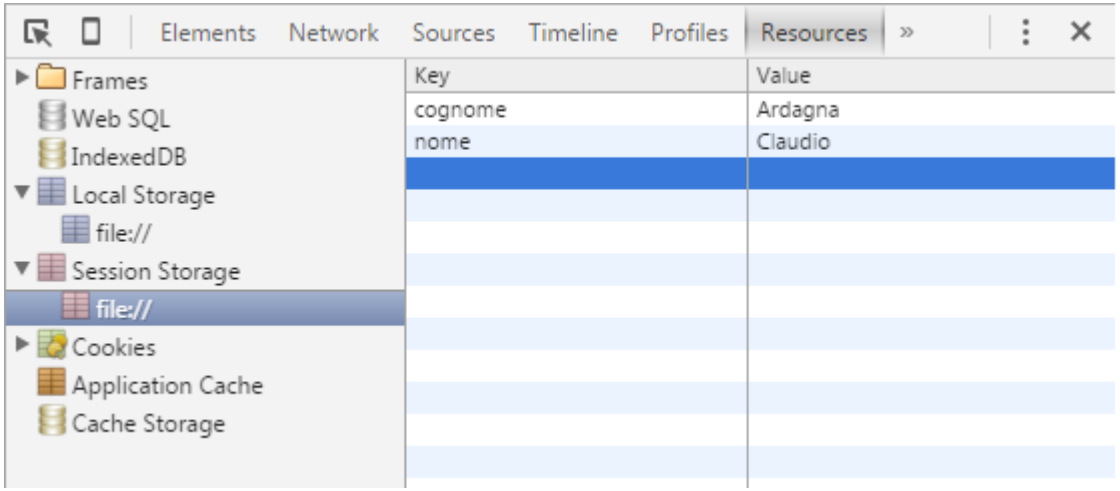
# Esempio

```
<script>
  localStorage.nome="Pippo";
  localStorage.setItem("cognome", "Pluto");
alert(localStorage.getItem("cognome"))
</script>
```

- Memorizzo in local storage e poi riprendo un valore e lo stampo a video (localStorage.html)
- Visualizzabile all'interno di developer tookit di Chrome
  - Menù -> more tools –> developer tools

# Esempio

- È possibile vedere i valori nel session storage e local storage con il web inspector
- È possibile vedere e anche cambiare i valori nel local storage con Safari o Chrome Web Inspector
- Setting -> more tools -> developer tools (Chrome)



# Prossimamente… JSON

```
<script type="text/JavaScript">
   var d = [{"count": 1, "id": 2, "message": "ciao"},{"count": 2,
"id": 3, "message": "hello"}];
   sessionStorage.setItem("dstring",JSON.stringify(d));
   var dstring = sessionStorage.getItem("dstring");
   var drestored = JSON.parse(dstring);
   alert(drestored[0].count+11);
</script>
```

- Memorizzo in session storage e poi riprendo un valore e lo stampo a video
- Visualizzabile all'interno di developer tookit di Chrome
  - Menù -> more tools –> developer tools

# Drag & Drop

# Drag&Drop

- Funzionalità comune
- Permette di spostare un elemento in un’altra posizione
- Ogni elemento può essere trascinato
- Attributo draggable="true" rende un elemento trascinabile

# Drag&Drop: esempio

- <img id="drag1" src="img_logo.gif" draggable="true" ondragstart="drag(event)" width="336" height="69">
  - Dopo aver definito un elemento trascinabile bisogna definire cosa fare quando un elemento è trascinato
  - Attributo ondragstart="nome_funzione_js()" specifica quale dato trascinare

- function drag(ev) {

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

  }
  - Tipo di dato da aggiungere all'elemento destinazione: testo
  - Elemento da trascinare: drag1

# Drag&Drop

- 
  - Definisce dove verrà posizionato l'elemento trascinato
  - Attributo ondragover

- function allowDrop(ev) {

  ev.preventDefault();

  }
  - Fa in modo di non seguire l'approccio di default che vieta il drag and drop

# Drag&Drop

- 
  - Definisce dove verrà posizionato l'elemento trascinato
  - Attributo ondragover

- function drop(ev) {

  ev.preventDefault();

  var data = ev.dataTransfer.getData("text");

  ev.target.appendChild(document.getElementById(data));

  }
  - Drop event scatena la funzione drop che permette di piazzare l'elemento

- drag-and-drop.html (prevent default) e drag-and-drop-link.html (default)

# Drag&Drop

- Call preventDefault() fa in modo che il browser non usi il comportamento di default nella gestione dei dati
- Call dataTransfer.getData() per ottenere il dato trascinato
  - Con questo metodo ritorneremo ogni dato che era stato impostato con lo stesso tipo nel metodo setData()
- Il dato trascinato è l'id dell'elemento trascinato ("drag1")
- Pone l'elemento trascinato nell'elemento in cui viene rilasciato

# Data-attribute (SOLO LETTURA)

# A cosa servono

- Necessità di associare metadati a elementi di una pagina HTML
  - Obiettivo rendere JavaScript più semplice e raggiungere porzioni di codice più velocemente
- Originariamente venivano usati gli attributi class o rel
- Grazie ai data-attribute di HTML5, è possibile aggiungere attributi personalizzati su tutti elementi con una sintassi standard “nome=valore”
  - Nome: deve essere almeno lungo un carattere, preceduto da data-, e non dovrebbe contenere lettere maiuscole
  - Valore: qualunque valore desideriamo

# A cosa servono

- Data-* attribute è usato per memorizzare dati privati e custom nella pagina o applicazione
- Data-* attribute forniscono l'abilità di aggiungere data attribute custom a tutti gli elementi HTML
- I dati memorizzati possono essere usati da JavaScript per creare una migliore user experience

# Definizione W3C

- Gli attributi data- sono destinati a memorizzare dati personalizzati privati alla pagina o all'applicazione, per i quali non ci sono attributi o elementi più appropriati. Ogni elemento HTML può avere qualsiasi numero di attributi personalizzati specificati, con qualsiasi valore

- Custom data attributes are intended to store custom data private to the page or application, for which there are no more appropriate attributes or elements. These attributes are not intended for use by software that is independent of the site that uses the attributes. Every HTML element may have any number of custom data attributes specified, with any value.

# Esempio

- Far apparire un tooltip quando un utente seleziona un periodo tra quelli indicati nell'esempio: il nuovo livello conterrà il costo dell'abbonamento, il periodo di riferimento e altre informazioni per quel periodo (http://www.w3schools.com/howto/howto_css_tooltip.asp)

```
<ul id="periodo_anno">
  <li data-costo="20" data-periodo="da Gennaio ad Aprile">Primo</li>
  <li data-costo="30" data-periodo="da Maggio a Agosto">Secondo</li>
  <li data-costo="50" data-periodo="da Settembre a Dicembre">Terzo</li>
</ul>
```

# Cosa memorizzare

- Altezza iniziale o l'opacità di un elemento che potrebbe essere richiesto nel calcolo di un'animazione JavaScript
- Parametri di un filmato Flash che viene caricato tramite JavaScript
- Dati sullo stato di salute, munizioni, o la vita di un elemento in un gioco basato su JavaScript
- Capitoli di un <video> o i sottotitoli, da sfruttare con Javascript (http://dev.opera.com/articles/view/accessible-html5-video-with-javascripted-captions/)

# Esempio di utilizzo: improprio

- Accesso tramite getAttribute e setAttribute (data-attribute-improprio.html)

```
<div id='mese-bolletta' data-mese='12'></div>
<script>
   // 'Leggo' il data- usando getAttribute
   var bolletta = document.getElementById('mese-bolletta');
   var mese = bolletta.getAttribute('data-mese'); // = '12'
   // 'Scrivo' il data- usando setAttribute
   bolletta.setAttribute('data-mese','7'); /
</script>
```

# Esempio di utilizzo: proprio

- Accesso tramite le proprietà dataset di un elemento (data-attribute-proprio.html)
  - Questa struttura dati, parte delle nuove API HTML5 JavaScript, restituisce un oggetto DOMStringMap contenente tutti gli elementi data-

```
<div id='mese-bolletta' data-costo='50' data-mese-anno='giugno'></div>
<script>
   // 'Leggo' il data- usando il dataset
   var abbonamento = document.getElementById('mese-bolletta');
   var costo = bolletta.dataset.costo; // = 50;

   // 'Scrivo' il data- usando il dataset
   var mese = bolletta.dataset.meseAnno; // ‘mese-anno' -> ‘meseAnno'
   bolletta.dataset.meseAnno = 'febbraio';  //
</script>
```

# Esempio di utilizzo: proprio

- Riferimento ai dati senza considerare il prefisso data-
  - Utilizzando il nome assegnato, invece del nome completo dell'attributo
  - Se all'interno del nome dell'attributo sono presenti dei trattini, questi saranno convertiti in CamelCase.

- Se uno specifico *data-qualcosa* diventa ridondante e non è più necessario, è possibile rimuoverlo dall'elemento DOM assegnandogli un valore null.
  - bolletta.dataset.costo = null;

# HTML 5 audio & video (SOLA LETTURA)

# Audio

- <audio>
- Con HTML nessun standard per eseguire file audio in una pagina web
- File audio eseguiti con plug-in (come flash)
  - Diversi browser, diversi plug-in
- HTML5 definisce un nuovo elemento che specifica un modo standard per inserire un file audio in una pagina web: elemento <audio>
- <audio src='file.ogg' controls></audio>

# Audio

- Possibile aggiungere alternative al tag HTML5
- Utile nel caso il browser non supporti il formato del file scelto
  - Ad es., Firefox 3.5 non supportava mp3
- Alternativa tra due file audio

```
<audio controls>
  <source src='audios/file.ogg' />
  <source src='audios/file.mp3' />
</audio>
```

# Audio

- Attributi
  - Autoplay: file inizia non appena pronto
  - Controls: visualizza i controlli audio
  - Loop: file viene eseguito in circolo all'infinito
  - Preload: assume valori auto, metadata e none
    - Specifica se e quando l'audio deve essere eseguito quando la pagina viene caricata
  - Src: specifica l'url del file
  - audio.html

# Video

- <video>
- Con HTML nessun standard per eseguire file video/movie in una pagina web
- File video eseguiti con plug-in
  - Diversi browser, diversi plug-in
- HTML5 definisce un nuovo elemento che specifica modalità standard per aggiungere un video/movie a una pagina web
  - Elemento <video>

# Video

- Attributi
  - Autoplay: file inizia non appena pronto
  - Controls: visualizza i controlli audio
  - Loop: file viene eseguito in circolo all'infinito
  - Preload: assume valori auto, metadata e none
    - Specifica se e quando l'audio deve essere eseguito quando la pagina viene caricata
  - Src: specifica l'url del file
  - Height, width: specificano altezza e larghezza in pixel
  - Muted: l'audio non viene eseguito
  - Poster: specifica l'url di un'immagine da mostrare fino a quando il video non è scaricato o l'utente non lo fa partire

# Video

- <video src='videos/BigBuck.ogg' **controls** ></video>
  - I controlli hanno layout diversi in browser diversi
  - video.html

# Video

- Aggiungere un fallback in caso non si riesca a visualizzare il file (link)
  - <video controls poster='poster320.jpg' width='320' height='180'>
    <source src='videos/BigBuck.ogg' type='video/ogg; codecs="theora,vorbis"'>
    <source src='videos/BigBuck.mp4' type='video/mp4; codecs="avc1.42E01E, mp4a.40.2"'>
    <h1><a href="videos/BigBuck.mp4">Download the video</a></h1>
    </video>
- h1 ignorato se va tutto bene

# Video

- Aggiungere un fallback in caso non si riesca a visualizzare il file (flash)
  - <video controls poster='poster320.jpg' width='320' height='180'>
    <source src='videos/BigBuck.ogg' type='video/ogg; codecs="theora, vorbis"'>
    <source src='videos/BigBuck.mp4' type='video/mp4; codecs="avc1.42E01E, mp4a.40.2"'>
    <object width="320" height="180" type="application/x-shockwaveflash" data="player/flowplayer-3.2.5.swf">
    <param name="movie" value="player/flowplayer-3.2.5.swf" />
    <param name="allowfullscreen" value="true" />
    <param name="flashvars" value='config={"clip": {"url": "videos/BigBuck.mp4", "autoPlay":false, "autoBuffering":true}}' />
    </object>
    </video>

# Video

- Aggiungere un fallback in caso non si riesca a visualizzare il file (flash+h1)
  - <video controls poster='poster320.jpg' width='320' height='180'>
    <source src='videos/BigBuck.ogg' type='video/ogg; codecs="theora, vorbis"'>
    <source src='videos/BigBuck.mp4' type='video/mp4; codecs="avc1.42E01E, mp4a.40.2"'>
    <object width="320" height="180" type="application/x-shockwaveflash" data="player/flowplayer-3.2.5.swf">
    <param name="movie" value="player/flowplayer-3.2.5.swf" />
    <param name="allowfullscreen" value="true" />
    <param name="flashvars" value='config={"clip": {"url": "videos/BigBuck.mp4", "autoPlay":false, "autoBuffering":true}}' />
    <h1><a href="videos/BigBuck.mp4">Download the video</a></h1>
    </object>
    </video>

# Video

- Esecuzione file .ogg
- Se fallisce, esecuzione file .mp4
- Se fallisce, esecuzione flash
- Se plug-in non installato, link per il download

# HTML5 Canvas (SOLO LETTURA)

# Canvas

- HTML5 Canvas permette di disegnare qualsiasi cosa nel nostro sito, tutto tramite JavaScript
  - Migliora le performance del nostro sito web evitando di scaricare immagini dalla rete
- Possiamo disegnare forme e linee, archi e testo, gradient e pattern
  - Canvas permette di manipolare i pixel di un'immagine e anche di video
- Canvas 2D Context spec è supportato dai maggiori browser
  - Safari 2.0+
  - Chrome 3.0+
  - Firefox 3.0+
  - Internet Explorer 9.0+
  - Opera 10.0+
  - iOS (Mobile Safari) 1.0+
  - Android 1.0+

# Canvas

- È un'API per il disegno in 2D parte della specifica HTML5
- Permette agli sviluppatori di disegnare dinamicamente su una pagina web
  - Usa dati acceduti in diversi modi
  - Input e interazioni con l'utente
- Aggiornamenti in real time
- Permette di definire la superficie del disegno, ma richiede **JavaScript** per definire le istruzioni di disegno, come le forme, colori e linee
  - <canvas id="lessonCanvas" width="300" height="300"style="margin:100px;"></canvas>

# Canvas: Esempio

- Creare un elemento canvas
  - <canvas id="myCanvas"></canvas>
- Disegnare un elemento canvas
  - Definire la funzione disegno e il suo contesto
    - <script>

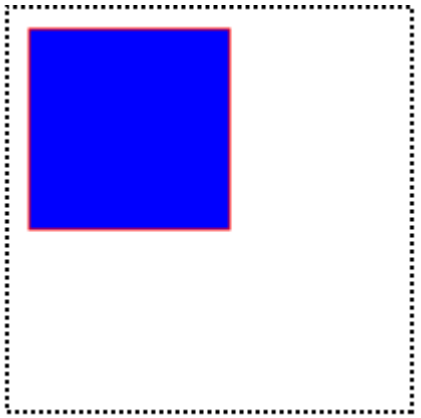
```
…
function disegno() {
        var canvas = document.getElementBy Id("myCanvas");
        var context = canvas.getContext("2d");
}
</script>
```

# Canvas: Esempio

- Un rettangolo con bordo rosso e riempimento blu (canvas-1.html)
  - function disegno() {

```
var canvas = document.getElementBy Id(“myCanvas”);
var context = canvas.getContext(“2d”);
context.strokeStyle = “red”;
context.fillStyle = “blue”;
context.fillRect(10,10,100,100);
context.strokeRect(10,10,100,100);
```

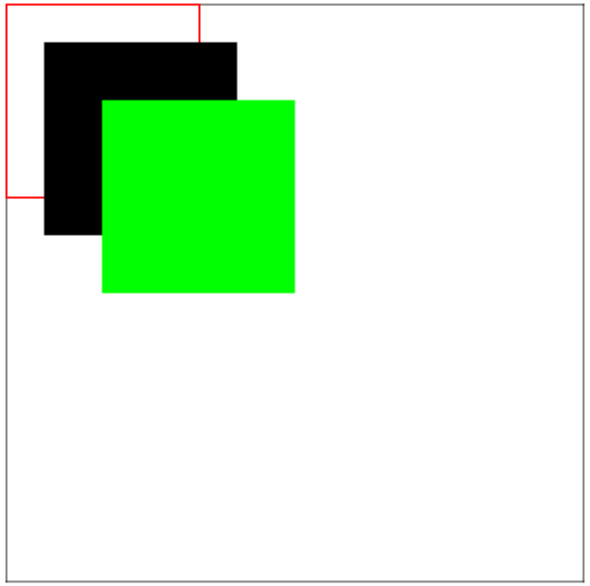
  }



# Canvas: Altro esempio

- Rettangoli multipli (canvas-2.html)

```
<body onload="setup()">
 <canvas id="lessonCanvas" width="300" height="300"style="margin: 100px;"></canvas>
 <script type="text/JavaScript">
  function setup() {
  var canvas = document.getElementById('lessonCanvas');
  if (canvas.getContext) {
    var ctx = canvas.getContext('2d');
    ctx.strokeRect(0, 0, 300, 300);
  }
  ctx.strokeStyle = 'rgb(255, 0, 0)';
  ctx.strokeRect(0.5, 0.5, 100, 100);
  ctx.fillRect(20, 20, 100, 100);
  ctx.fillStyle = 'rgb(0, 255, 0)';
  ctx.fillRect(50, 50, 100, 100);
  }
  </script>
</body>
```



# Canvas: Librerie

- Esistono anche librerie per il disegno
- Ad esempio Chart.js https://github.com/chartjs/Chart.js permette di disegnare dei grafici
- Package online https://cdnjs.com/libraries/Chart.js
- <script src="https://cdnjs.cloudflare.com/ajax/libs/Chart.js/2.4.0/Chart.js"></script>

# HTML 5 Geo location (SOLO LETTURA)

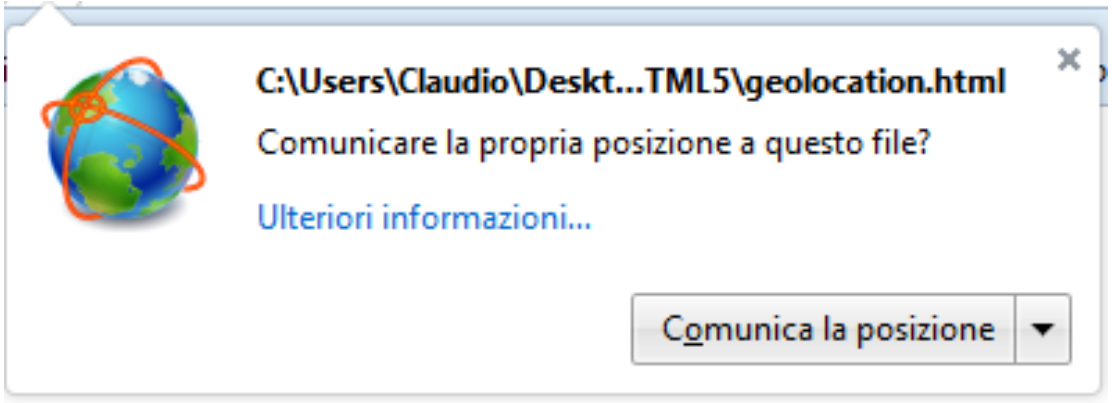
# Geolocation

- Geolocation API permette di identificare e visualizzare la posizione di un individuo
  - Geolocation permette ai visitatori di un sito web di condividere la loro locazione

- La posizione può essere identificata attraverso indirizzo IP, connessione a rete wireless, cell tower, GPS hardware

# Geolocation

- Problematiche di privacy
  - Non tutti vogliono condividere la loro locazione (informazione personale)
  - I visitatori devono dare il loro consenso per condividere la loro locazione
  - Niente sarà passato al sito/applicazione web senza il consenso dell'utente
  - La decisione è presa attraverso una notifica fatta dal browser



# Geolocation

- Geolocation non è mai stata parte di HTML5
  - È semplicemente una tecnologia legata al web sviluppata in concomitanza con HTML5

- Geolocation API permette a JavaScript di richiedere il permesso di accesso alla locazione geografica
  - Si basa sull'oggetto navigator.geolocation e funzione getCurrentPosition

# Geolocation

- Con geolocation viene determinata la posizione dell'utente
- È possibile ricevere una notifica sul cambiamento della posizione utente
  - Ad esempio, per fornire direzioni di guida real-time
- HTML5 estende JavaScript per utilizzare funzioni per trovare o tracciare un utente
- Geolocation: metodi
  - getCurrentPosition
  - watchPosition
  - clearPosition

# Geolocation: mostra posizione

- Esempio di una semplice applicazione che ritorna la posizione dell'utente come latitudine e longitudine

```
<script type="text/JavaScript">
output = document.getElementById("output");
function setup() {
   if(navigator.geolocation) {
      navigator.geolocation.getCurrentPosition(showLocation,
                                               locationError);
   }
   else locationError();
}

function showLocation(position){
   output.innerHTML = 'Your location is '+ position.coords.latitude + ' lat ' + position.coords.longitude + ' lon';
}

function locationError( error ){
   output.innerHTML = 'Location could not be determined';
}
</script>
```

- Se Geolocation è supportato esegue getCurrentPosition(), altrimenti mostra un messaggio di errore display a message to the user
- Se getCurrentPosition() ha successo ritorna un oggetto coordinate alla funzione specificata nel parametro (showLocation)
- showLocation() mostra latitudine e longitudine

# Geolocation: getCurrentPosition

- getCurrentPosition() ritorna un oggetto se ok
- Latitudine, longitudine e accuratezza sono sempre ritornate
- Le altre solo se disponibili

| Proprietà | Descrizione |
|---|---|
| coords.latitude | La latitudine come numero decimale |
| coords.longitude | La longitudine come numero decimale |
| coords.accuracy | Accuratezza della posizione |
| coords.altitude | Altitudine in metri sul livello del mare |
| coords.altitudeAccuracy | Accuratezza dell'altitudine |
| coords.heading | La direzione come gradi dal nord in senso orario |
| coords.speed | La velocità in m/s |
| coords.timestamp | Data e tempo della risposta |

# Geolocation: Gestione errore

```
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 infomation 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;
   }
}
```

# Geolocation: mappa

- Mostrare i risultati su googlemaps (geolocation-map.html)

```
function showLocation(position){
   var latlon=position.coords.latitude+","+position.coords.longitude;
   var img_url =
"http://maps.googleapis.com/maps/api/staticmap?center="+latlon"&zoom=14&size=400x300&sensor=false ";
   document.getElementById("mapholder").innerHTML="<img
                                                    src='"+img_url+"'/>";
}
```

# Geolocation: mappa

- Mostrare i risultati su googlemaps (geolocation-map.html)

```
function showLocation( position )
{
   var latlng = new google.maps.LatLng(position.coords.latitude,
                                                    position.coords.longitude );
   var mapOptions = {
                          zoom: 8,
                          center: latlng,
                          mapTypeId: google.maps.MapTypeId.ROADMAP
                        };
   var map = new google.maps.Map( map_canvas, mapOptions );
}
```

# Risorse Web

- http://www.w3cschool.com
- http://diveintohtml5.info/table-of-contents.html
- http://www.w3.org/TR/html5/

# Conclusioni

- HTML5 elementi e attributi
  - Elementi semantici
  - Form
  - Audio e video
  - Canvas
  - Storage
  - Geolocation
  - Drag&Drop