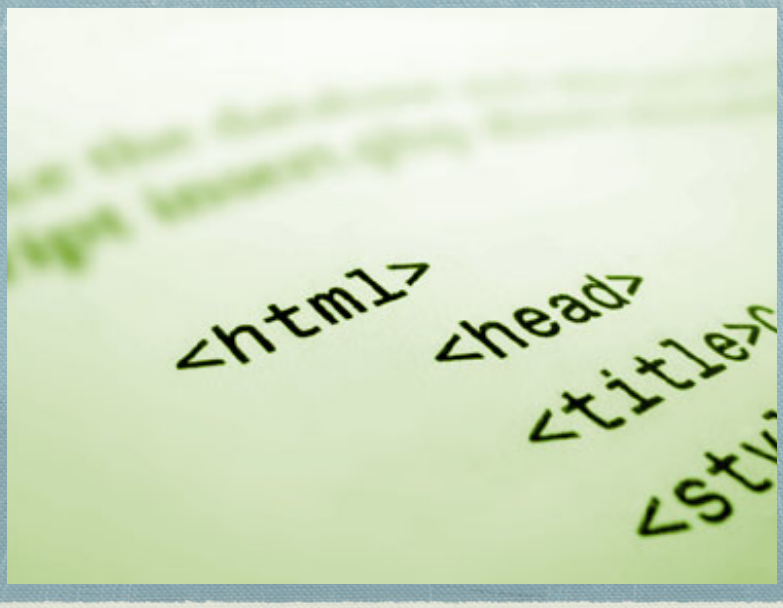



# HTML 5 - rich media

Corso di Web Design
Fabio Pittarello, Università Ca' Foscari Venezia - DAIS pitt@unive.it


## HTML 5

- E' possibile costruire molte cose utilizzando HTML, CSS e Javascript.
- Nonostante questo ci sono alcuni buchi nella palette delle possibilità degli standard web, ad esempio la pubblicazione di video e audio.
- HTML aiuta a coprire questi buchi, evitando l'uso di plugin proprietari.

# Audio

- Nel corso degli anni, il formato MP3 ha vinto la battaglia per l'ubiquità, ma fornire ai visitatori un modo facile di ascoltare un suono richiede ancora una tecnologia proprietaria. In questo caso il player Flash ha ancora un ruolo determinante.
- HTML5 permette di incorporare in modo facile un documento audio in un documento HTML
- Un esempio semplice:

```
<audio src="witchitalineman.mp3"> </audio>
```

# Audio: attributi booleani

- Generalmente è una **pessima** idea avviare automaticamente la fruizione di un segnale audio al caricamento di una pagina; tuttavia questo è possibile aggiungere l'attributo booleano autoplay:

  ```
  <audio src="witchitalineman.mp3" autoplay> </audio>
  ```

  e si può addirittura iterare la fruizione all'infinito con l'attributo booleano loop

  ```
  <audio src="witchitalineman.mp3" autoplay loop> </
  audio>
  ```

- Nota: attenzione a non confondere **attributi** e **valori booleani**; nel primo caso il criterio di verità è dato dal fatto che l'attributo venga incluso; "autoplay=false" da lo stesso risultato di scrivere autoplay

# Audio: dare il controllo agli utenti

- Dare agli utenti la possibilità di controllare il playback di un file audio è una buona idea che può essere implementata utilizzando l'attributo booleano controls:

  <audio src="witchitalineman.mp3" controls> </audio>


# Audio: dare il controllo agli utenti

- E' possibile in alternativa creare i propri controlli, utilizzando Javascript e la Audio API. Un esempio:

```
<audio id="player" src="witchitalineman.mp3">
</audio>
<div>
    <button onclick="document.getElementById
    ('player').play()">Play</button>
    <button onclick="document.getElementById
    ('player').pause()">Pause</button>
    <button onclick="document.getElementById
    ('player').volume  += 0.1">Volume Up</button>
    <button onclick="document.getElementById
    ('player').volume  -= 0.1">Volume Down</button>
</div>
```

# Precaricare l'audio

- La specifica HTML5 a un certo punto ha incluso l'attributo booleano **autobuffer** per richiedere al browser di precaricare l'audio in background, in modo da poterlo utilizzare subito al momento necessario
- Sfortunatamente Safari precaricava l'audio in ogni caso, che sia presente o no l'attributo booleano
- L'attributo autobuffer ora è stato sostituito dal più espressivo (e non booleano) attributo **preload**, che può assumere i seguenti tre valori: none, auto and metadata

```
<audio src="witchitalineman.mp3" controls
preload="none"> </audio>
```

# Tutto risolto con il tag audio?

- Sfortunatamente no, perchè il problema non sta nella specifica, ma nei formati audio:
  MP3 non è un formato aperto e non è possibile decodificare un file MP3 senza pagare royalties ai possessori di diritti sul formato; questo rende il formato inadatto all'uso open-source
- Per questo motivo Safari (posseduto da Apple) permette il playback di files MP3, mentre Firefox non lo consente.

- E' possibile tuttavia utilizzare formati audio alternativi, come il coded **Vorbis**. I contenuti codificati con questo formato vengono solitamente distribuiti come **.ogg** files
- Al momento attuale Firefox supporta Vorbis, ma Safari no …

# Tutto risolto con il tag audio?

| Browser | .aac | .mp3 | .ogg | .webm |
|---|---|---|---|---|
| Chrome 12+ | X | X | X | X |
| Firefox 3.6+ | | | X | X |
| IE9+ | X | X | | |
| Opera 10.6+ | | | X | X |
| Safari 5+ | X | X | | |

# Una possibile soluzione

- Utilizzare il tag source, che permette di specificare più formati audio. Grazie a questa soluzione ogni browser potrà effettuare il playback del contenuto utilizzando il formato appropriato:

```
<audio controls>
    <source src="witchitalineman.ogg" type="audio/ogg">
    <source src="witchitalineman.mp3" type="audio/mpeg">
</audio>
```

# Retrocompatibilità (graceful degradation)

- Che cosa fare per quei browser che ancora non supportano il tag audio? Ad esempio Internet Explorer?
- E' possibile aggiungere al codice precedente la descrizione dell'oggetto Flash che permetterà il playback del file
- E' possibile aggiunger un hyperlink come ulteriore risorsa nel caso l'utente non abbia installato un player Flash

```
<audio controls>
    <source src="witch.ogg" type="audio/ogg">
    <source src="witch.mp3" type="audio/mpeg">
    <object type="application/x-shockwave-flash"
    data="player.swf?soundFile=witch.mp3">
                <param name="movie"
          value="player.swf?soundFile=witch.mp3">
          <a href="witch.mp3">Download the song</a>
    </object>
</audio>
```

# Accessibilità del contenuto audio

- Fornire un modo alternativo di fruizione dello stesso tipo di contenuto non implica garantire l'accessibilità del contenuto stesso.
- Questa potrebbe essere garantita, nel caso della disponibilità di una trascrizione testuale del contenuto audio, da un ulteriore tag p, che potrebbe essere nidificato all'interno del tag audio oppure messo a disposizione di tutti:

```
<audio controls>
    <source src="witchitalineman.ogg" type="audio/ogg">
    <source src="witchitalineman.mp3" type="audio/mpeg">
</audio>
<p>I am a lineman for the county...</p>
```

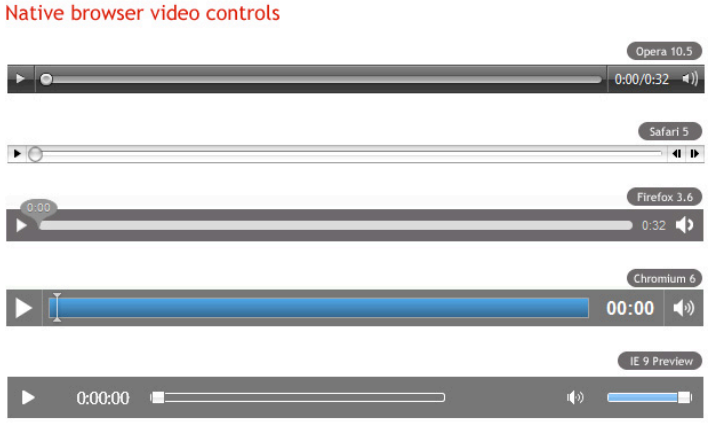
# Video

- Al momento il plugin Flash è una delle tecnologie dominanti per il video, ma HTML può cambiare questa situazione … e la sta già cambiando nell'ambito mobile
- HTML definisce il tag video e gli attributi opzionali autoplay, loop e preload che funzionano allo stesso modo di quanto previsto per il tag audio
- Anche in questo caso è possibile specificare la locazione del file video utilizzando l'attributo src o il tag source nidificato all'interno del tag video

# Video

- Il video naturalmente occuperà più spazio sullo schermo e tipicamente è necessario fornire le dimensioni del video; anche per il video è possibile permettere all'utente di controllare il playback con l'attributo controls

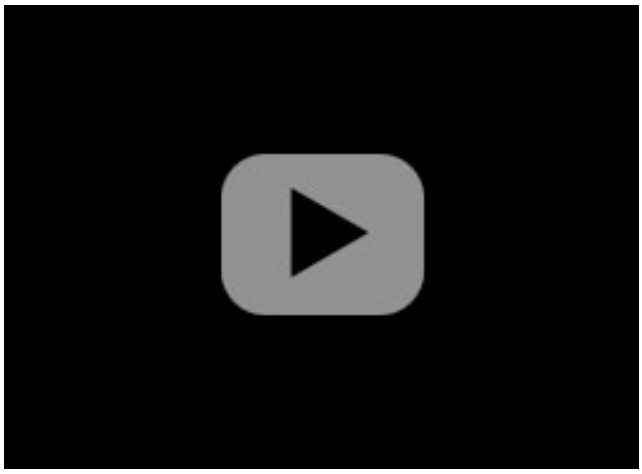
  <video src="movie.mp4" controls width="360" height="240"> </video>



# Video

- E’ possibile scegliere anche un’immagine rappresentativa da mostrare utilizzando l’attributo poster

  <video src="movie.mp4" controls width="360" » height="240" poster="placeholder.jpg"> </video>



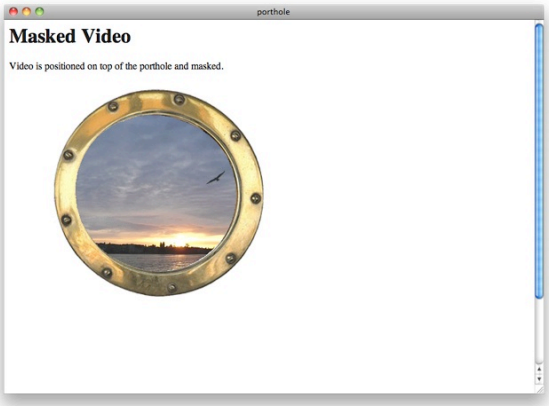
# Video (formati)

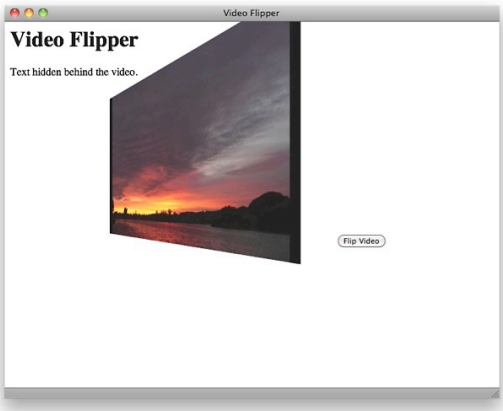
- Anche per il video esiste un problema di formati. Alcuni browser supportano il formato **MP4**, soggetto a royalties per la decodifica, mentre altri supportano **Theora Video**; è possibile fornire più opzioni attraverso l’attributo source.
  Anche in questo caso è possibile ottenere ulteriori forme di fallback attraverso il rif. a **Flash** o a download del file video.

```
<video controls width=”360” height=”240” poster==”place.jpg”>
    <source src=”movie.ogv" type=”video/ogg">
    <source src=”movie.mp4" type=”video/mp4">
    <object type="application/x-shockwave-flash”
    width=”360” height=”240”
    data="player.swf?file=movie.mp4">
                        <param name="movie”
          value="player.swf?file=movie.mp4">
          <a href=”movie.mp4">Download the movie</a>
    </object>
</audio>
```

# Scripting and styling

- Uno dei vantaggi di utilizzare codice HTML per specificare rich media come il video è che è possibile controllarli con le altre tecnologie del browser, Javascript e CSS.



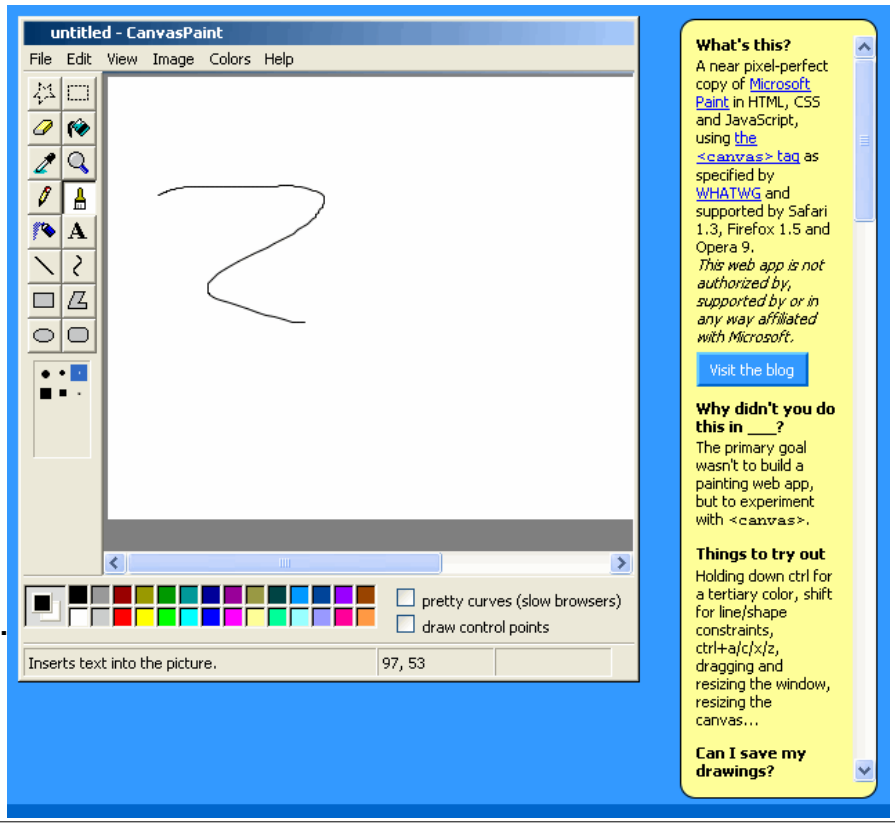

# Canvas

- Il tag canvas permette di definire un ambiente per la creazione di immagini dinamiche.

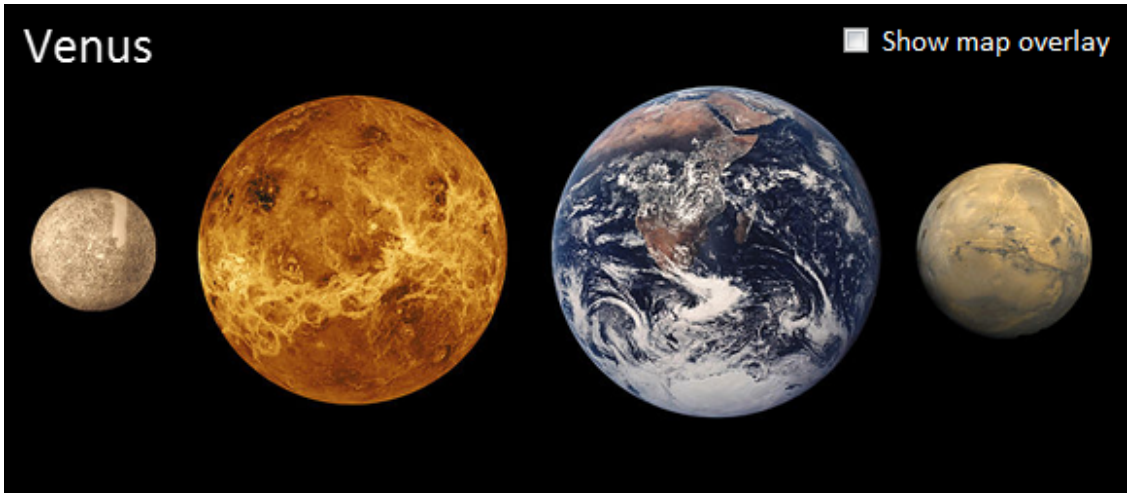
  <canvas id="my-first-canvas" width="360" height="240"> </canvas>

- Tutto quello che viene specificato all'interno del tag – attraverso l'uso di Javascript - verrà considerato solo dai browser che supportano canvas.

# Canvas




# Canvas




WebGL

# Canvas

- E’ possibile nidificare all’interno del tag canvas altri tag il cui contenuto verrà visualizzato solo dai browser che non supportano canvas

```
<canvas id="my-first-canvas" width="360" height="240">
    <p>No canvas support? Have an old-fashioned
    image instead:</p>
    <img src="puppy.jpg" alt="a cute puppy”>
</canvas>
```

# Accesso negato

- Tutti i documenti che stanno su web possono essere descritti attraverso il DOM (document object model)
- Questo permette di accedere e di manipolare gli elementi che costituiscono la pagina
- Sfortunatamente l’elemento canvas non ha un DOM; questo significa che il contenuto disegnato all’interno dell’elemento non può essere rappresentato come un albero di nodi e successivamente manipolato
- Le tecnologie assistive (come gli screen readers) si avvalgono del DOM per comprendere la struttura di un documento; l’attuale mancanza di questo accesso è un grosso problema per l’accessibilità di HTML a cui si sta tentando di trovare una soluzione

# Canvas intelligenti

- Il fatto che al momento il contenuto di canvas non sia accessibile non significa che non possa essere utilizzato.
- L'approccio corretto è quello di utilizzare canvas per riciclare contenuto esistente piuttosto che crearlo, attraverso tecniche quali l'*obnotrusive Javascript*. In altre parole l'idea è quella di arricchire l'esperienza del visitatore del sito che usa un browser adeguato senza privare gli altri visitatori della possibilità di fruire dei dati.