Multimédia et HTML5

Structuration sémantique

- `<div>` : Division des contenus

Balises et attributs

- `<section>` : bloc de contenu
- `<article>`
- `<aside>` : encart
- `<header>` : entête de la page
- `<footer>` : pied de page
- `<nav>` : menu
- `<figure>` : associe contenu et légende

Formulaires

Accessibilité

- Types d'éléments de dialogue: `email`, `url`, `number`, `slider`, `color`, `date`, `tel`, ...
- Texte d'invitation (placeholder)
- Autofocus sur un champ particulier
- Validation à-priori

Exemple

```
<form action="..." method="post">
  <input type="number" min="0" max="12"
    step="4" value="8">
</form>
```

- la disponibilité ainsi que le rendu dépendra du navigateur

JavaScript & DOM

Différentes API en JavaScript

- Dessin 2D dans l'élément `<canvas>`
- Contrôle multimédia
- Stockage persistant (clé/valeur ou base de données SQL)
- Applications offline
- Peer2Peer (WebSockets)
- Édition avancée, drag & drop...
- Manipulation du DOM

Extensions multimédia

HTML

- Balises `<audio>` et `<video>`

Formats supportés

- Dépendant du navigateur et aussi de la plate-forme (présence de DRM)
- Pour l'audio: MP3 et AAC (IE, Safari), Ogg Vorbis (Firefox et Chrome), ...
- Pas de consensus pour un codec vidéo *standard* pour des raisons de licence
  - MPEG4 avec video H.264 et audio AAC: Safari/iOS, IE
  - Ogg Theora: Chrome, Firefox, Opera
  - WebM avec codec video VP8 et audio Vorbis: idem

Extensions multimédia

- Possibilité de fournir le media dans plusieurs formats
- Exemple:

```
<video>
    <source src="path/to/vid.mp4" type="video/mp4">
    <source src="path/to/vid.ogg" type="video/ogg">
    <source src="path/to/vid.webm" type="video/webm">

    <p>
      Ce navigateur ne supporte aucun des codecs
      de cette vidéo
    </p>
</video>
```

- Possibilité d'ajouter des sous-titres: `<track>`

Eléments multimédia

Elément <video>

- HTML:

```
<video id="vid" src="vid.mov"/>
```

- JavaScript:

```
var vid = document.getElementById("vid");
if (vid.paused) vid.play();
else vid.pause();
```

Elément <audio>

- HTML:

```
<audio src="sample.wav">
```

- JavaScript:

```
var a = new Audio("audio.wav");
a.addEventListener('ended', endMsg, false);
a.play();
```

Attributs multimédia

Attributs commun à `<audio>` et `<video>`

- `src`: adresse de la ressource multimédia (URL)
- `autoplay`: commence la lecture dès que possible
- `autobuffer`: chargement du média probable (booléen)
- `loop`: lecture en boucle (booléen)
- `controls`: affichage de contrôles par le navigateur si `false`

Attributs spécifiques à `<video>`

- `poster`: image représentative de la vidéo
- `width`/`height`: dimensions d'affichage souhaitées (unités CSS)

DOM multimédia

Propriétés DOM

- État
  - `a.paused`/`a.ended`
  - `a.play()`/`a.pause()`
  - `a.playbackRate` (R/W): vitesse de lecture (défaut: `1.0`)
- Position (en secondes)
  - `a.duration`
  - `a.currentTime` (R/W)
- Contrôles
  - `a.volume` (RW) entre 0.0 et 1.0
  - `a.muted` (RW) (booléen)
- Événements
  - `playing`, `ended`, `play`, `pause`, `waiting`, `...`

Canvas

Principes

- L'élément `<canvas>` définit un espace de dessin
- Attributs: height/weight (pixels CSS)
- Dessin bitmap (à la différence du format vectoriel XML SVG)
- Le tracé se fait en JavaScript par l'intermédiaire d'un contexte de dessin

Contexte graphique

- Permet le dessin en 2D
- Espace cartésien avec (0,0) en coin supérieur gauche
- Extension au 3D basée sur OpenGL (WebGL)

Canvas - exemple

HTML

```
<canvas width="500" height="300" id="cnv">
</canvas>
```

JavaScript

```
window.onload = function() {
  var canvas = document.getElementById("cnv1");
  var c2d = canvas.getContext('2d');
}
```

Canvas - configuration

Etat du contexte graphique c2d

- Matrice de transformation courante
- Clipping region (limite le tracé)
- Attributs courants de dessin (couleur de trait et de remplissage, police, transparence. . .)
- Conserve un état graphique: `c2d.save()`/`c2d.restore()`

Transformations

- facteur d'échelle: `c2d.scale(x,y)`
- rotation: `c2d.rotate(r)`
- translation: `c2d.translate(x,y)`
- matrice de transformation: `c2d.transform(m11,m12,m21,m22,dx,dy)`

Canvas - configuration et utilisation

Couleur et style

- Style de tracé : `c2d.strokeStyle` (R/W)
- Style de remplissage : `c2d.fillStyle` (R/W)
- Baleur: couleur, gradient, pattern, ombrages...
- Valeur RGB : `"#80FF00"` (Red: 128, Green: 255, Blue: 0)
- Nom standard : “black”, “cyan”, “orange”...
- Pour les lignes: c2d.lineWidth, c2d.lineCap...

Formes

- Rectangles: `c2d.clearRect(x,y,w,h)`, `c2d.fillRect(...)`, `c2d.strokeRect(...)`
- Formes plus complexes (arcs...): `path`

Texte

- police, alignement... `c2d.fillText(string, x, y)`

Canvas - exemple de tracé

```
var canvas = document.getElementById('cnv1');
var c2d = canvas.getContext('2d');
// rectangle vert plein
c2d.fillStyle = "green";
c2d.fillRect(10, 10, 20, 40);
// rectangle rouge vide
c2d.strokeStyle = "red";
c2d.lineWidth = 3;
c2d.strokeRect(100, 100, 30, 30);
// on efface tout
console.log("deleting...");
c2d.clearRect(0, 0, canvas.width, canvas.height);
```

Canvas - affichage d'une image

`c2d.drawImage(image, x, y)`

- Utilisation de la classe `Image` vue au cours précédent
- Affichage de l'image à la position (x,y)
- Possibilité d'extraire une partie de l'image source (ex: sprites)
- Attention, le chargement d'une image n'est pas immédiat

Exemple

```
var img = new Image();
img.src = "balle.png";
img.onload = function() {
  c2d.drawImage(img, 10, 20);
}
```

Canvas - exemple de préchargement d'image

HTML

```
<img id="im1" src="logo.jpg" style="visibility:hidden">
<canvas id="can" width="300" height="200"></canvas>
```

JavaScript

```
window.onload = function() {
  var canvas = document.getElementById('can');
  var c2d = canvas.getContext('2d');
  i = document.getElementById('im1');
  c2d.drawImage(i, 50, 50);
}
```

Canvas - manipulation du bitmap

Exemple

```
img = c2d.getImageData(x,y,width,height);
pix = img.data;
for (i = 0; i < pix.length; i+=4) {
  pix[i] = ... // red (0-255);
  pix[i+1] = ... // green;
  pix[i+2] = ... // blue;
  pix[i+3] = ... // alpha;
}
c2d.putImageData(img,x,y);
```

- Note: Certains navigateurs limitent l'accès aux fichiers locaux !

Faire une animation

Ce qui ne marche pas

- ```
  while (true) {
  // lecture du clavier
  // déplace les éléments
  // dessine l'image
  // pause 20ms
  }
  ```
- L'affichage n'est jamais mis à jour
- Bloquant

Ce qui marche

- Programmer un timer à `n` appels par seconde pour afficher une animation
- La fonction appelée re-dessine complètement la scène
- Si besoin, enregistrer les événements clavier à part
  ```