MODULE 4

Stockage hors-ligne, Canvas, Géolocalisation, Touch et Gestures

STOCKAGE HORS-LIGNE

- Stockage local de données, dans le navigateur
- Local Storage / Session Storage
  - Web Storage / DOM Storage
  - Évolution de la notion de cookies
  - localStorage: persistent / sessionStorage: non persistent
  - Interface Javascript: Storage()
  - Support: IE8+, Firefox 3.6+, Safari 4+, Chrome 4+, Opera 10.5+, iOS2+, Android 2+
- (Web SQL)
  - Déprécié
  - Bases de données relationnelles (SQLLite)
- IndexedDB
  - Non relationnel: modèle clé-valeur
  - Support: IE10+, Firefox 4+, Chrome 11+, Opera 15+, Android 4.4+

WEB STORAGE

- Alternative aux cookies (mais aucune transmission au serveur)
- Petite quantité de données (5-10 Mo / domaine)
- 2 types, une interface commune:
  - localStorage
  - sessionStorage

IMPLÉMENTATION

- `var db = window.localStorage / window.sessionStorage`
- `db.setItem(key, value);`
- `db.getItem(key);`
- `db.removeItem(key);`
- `db.clear();`

- Local Storage et objets
  - `JSON.stringify()`
  - `JSON.parse()`

TP

- Mise à jour d'un enregistrement à partir de saisies utilisateur.

SESSIONSTORAGE ET CONTEXTE

- Une session par fenêtre / onglet
- Gros atout par rapport aux cookies

INDEXEDDB

- clé / valeur
- Plus grande quantité de données que les Web Storages
- Recherche indexée
- État de la spécification "Candidate Recommendation" depuis Juillet 2013
  http://www.w3.org/TR/IndexedDB

- Support non consistent par les navigateurs
- Préfixes
- Progressive enhancement!

IMPLÉMENTATION

- `window.indexedDB`
- `var openRequest = indexedDB.open(name, version);`
  - Asynchrone
    - Callback
      - success
      - error
      - upgradeneeded
        (création ou modification des object stores possibles uniquement au cours de cet évènement)
      - blocked
    - e.target.result contient la base de données

```
openRequest.onsuccess = function(e)
{
     db = e.target.result;
}
```

OBJECT STORES

- `db.createObjectStore('osName');`
- `db.objectStoreNames`
  - `.contains('osName');`
  - `array-like (DOMStringList)`

TRANSACTIONS

- `var transaction = db.transaction('osName', 'mode');`

```
> var transaction = db.transaction('monObjectStore', 'readwrite');
```

- `var store = transaction.objectStore('osName');`

AJOUTER DES DONNÉES

- `var addRequest = store.add(object, key);`
  - ➡ Asynchrone:
    - onerror
    - onsuccess
- keyPath et autoIncrement
  - `db.createObjectStore('osName', { keyPath: 'prop'});`
    `// clé générée à partir de la propriété 'prop' de l'objet stocké`
  - `db.createObjectStore('osName', { autoIncrement: true });`

LIRE DES DONNÉES

- `var transaction = db.transaction('osName', 'readonly');`
- `var store = transaction.objectStore('osName');`
- `var readRequest = store.get(x)`
  - x est la clé de la paire clé/valeur à récupérer
  - Asynchrone

```
readRequest.onsuccess = function(e) {
     // l'objet lu se trouve dans e.target.result
}
```

LES CURSEURS

- `var cursor = objectStore.openCursor();`
- ```
  cursor.onsuccess = function(e) {
        var result = e.target.result;
        if ( result ) {
             var curKey = results.key;
             var curVal = results.value;
             results.continue();
        }
  }
  ```

LES INDEX

- Doivent être créés lors d'un évènement 'upgrade'
- `objectStore.createIndex('name', 'path', {unique: true/false});`

  ```
  > objectStore.createIndex('titre', 'titre', {unique: false});
  ```
- Utilisation
  - ```
    var index = store.index('titre');

    var indexReq = index.get('Django Unchained');
    ```
  - ```
    indexReq.onsuccess = function(e) {
         var res = e.target.result;
         if ( res ) {
              console.log( res );
         }
    }
    ```
  - ouvrir un curseur sur un index

    ```
    var index = store.index('indexName');
    var cursor = index.openCursor('value');

    cursor.onsuccess = function(e) {
       if ( res ) {
           var curKey = res.key;
           var curVal = res.value;
           res.continue();
       }
    }
    ```

MISE À JOUR / SUPPRESSION

- Mise à jour d'un enregistrement
  - `var updateRequest = store.put(object);`
  - Asynchrone: callbacks identiques à add()
- Suppression d'un enregistrement
  - ```
    var t = db.transaction(["people"], 'readwrite');

    var request = t.objectStore('people').delete(idToDelete);
    ```
  - Asynchrone: callbacks "habituels"

(TP)

- Ajout / édition / suppression d'enregistrements
- IndexedDB et/ou localStorage

ÉVÈNEMENT 'STORAGE'

- storage / onstorage
  - Modification extérieure (par un autre onglet ou une autre fenêtre partageant le même Storage)
  - ```
    if ( window.addEventListener ) {
       window.addEventListener( 'storage',
       handle_storage_event, false );
    } else {
       window.attachEvent( 'onstorage',
       handle_storage_event );
    }
    ```
- Propriétés de l'évènement
  - url
  - key
  - oldValue
  - newValue

MODULE 4 >

GRAPHISME ET ANIMATION: CANVAS HTML5
    ```

INTRODUCTION (1)

- <canvas></canvas>
- Support: IE 9+, Firefox 3+, Chrome 3+, Safari 3+, Opera 10+
- Solution de repli (fallback content)

```
<canvas id="canvas" width="640"
height="480">
   Votre navigateur ne supporte pas cette
   fonctionnalité
</canvas>
```

- Exemples
  - *https://developer.mozilla.org/en-US/demos/detail/zen-photon-garden/launch*
  - *http://codepen.io/suffick/pen/KrAwx*
  - *http://codepen.io/soulwire/pen/Ffvlo*
  - *http://www.freeriderhd.com/t/1016-layers*
  - *http://mudcu.be/sketchpad/*

INTRODUCTION (2)

- Système de coordonnées
- Par défaut, canvas invisible (fond noir transparent)
- Attributs
  - width (en px, 300 par défaut)
  - height (en px, 150 par défaut)
- Fonctions
  - toDataURL(type): convertir le contenu en image (image/png, image/jpg, ...)
  - getContext(ctxID)

DESSINER

➡ Obtenir une référence à l'élément <canvas> dans le DOM

➡ Obtenir le contexte de dessin

➡ Dessiner dans le context (si non null)

L'API

- Rectangles
- Lignes
- Arcs
- Tracés
- Couleurs / Styles
- Courbes de Bézier
- Courbes quadratiques
- Texte
- Compositing
- Patterns
- Gradients
- Ombres portées
- Clipping paths
- Transformations
- Images
- Vidéos
- Raw Pixels

FONDAMENTAUX DU DESSIN VIA L'API CANVAS

- Styles et couleurs
  - `fillStyle`
    - Couleur (CSS)
    - Gradient
    - Pattern
    - Défaut: noir
  - `strokeStyle`
    - idem
  - `lineWidth`
    - Défaut: 1

FORMES BASIQUES

- Une seule forme primitive: le rectangle
  - `strokeRect(x,y,w,h)`
  - `fillRect(x,y,x,h)`
  - `clearRect(x,y,w,h)`
- Les lignes
  - `moveTo(x,y) // ATTN! déplacement seulement, pas de tracé`
  - `lineTo(x,y) // tracé de la position courante à la position spécifiée`
  - `lineWidth`
  - `lineCap // Extrémité des lignes: butt (défaut), round, square`
  - `lineJoin // Liaison des lignes: round, bevel, miter (défaut)`
  - `miterLimit // défaut: 10`
  - `beginPath() // nouveau tracé: ensemble d'instructions`
  - `stroke() // récupère l'ensemble des commandes, et les trace`

ÉTAT DU CANVAS

- Chaque contexte maintient un état, accessible dans le code
- L'état peut être sauvegardé, restauré, ajouté à la pile d'états
- Garde trace des propriétés suivantes
  - valeurs des propriétés: lineWidth, strokeStyle, fillStyle, lineCap, etc...
  - Matrice de transformation
  - Région de clipping
- `context.save()`
- `context.restore()`

ARCS ET TRACÉS

- Tracés
  - Ensemble de points connectés par des lignes ou des courbes, ouvert ou fermé.
    Un contexte a TOUJOURS un et UN SEUL tracé courant.
  - Fermé: point final identique au point initial.
  - Création d'un tracé avec `beginPath()`;
  - Dessin d'un tracé via `stroke()`;
  - Remplissage d'un tracé via `fill()`;
  - `closePath()`

ARCS

- Portion d'un cercle (lui-même arc entier de 360 degrés)
- `arc(x, y, r, sA, eA, aC)`
  - Centre (x,y)
  - Rayon r
  - Angle de départ sA (starting Angle)
  - Angle final eA (ending Angle)
  - Sens horaire (aC = false) ou anti-horaire (aC = true)
- `arcTo(x1, y1, x2 ,y2, r)`
  - Centre: position courante du crayon
  - Deux points de contrôle (x1,y1) et (x2,y2)
  - Rayon r
- closePath()
- ATTN! les angles sont en radians
  - `rad = ( Math.PI / 180 ) * degres`

COURBES DE BÉZIER, COURBES QUADRATIQUES

- **Courbes de Bézier**
  - Un point de départ, un point d'arrivée, deux points de contrôle
  - `bezierCurveTo(cx1,cy1,cx2,cy2,endX, endY)`
    `// le point de départ est la position courante du crayon`
- **Courbes quadratiques**
  - Un point de départ, un point d'arrivée, un point de contrôle
  - `quadraticCurveTo(cx,cy,endX,endY)`
    `// le point de départ est la position courante du crayon`

TEXTE

- Similaire aux tracés (fillStyle, strokeStyle...)
- Non affecté par le modèle des boites CSS
- Opérations
  - `font`
  - `textAlign: 'start' (défaut), 'end', 'left', 'right', 'center'`
  - `textBaseline: 'top', 'hanging', 'middle', 'alphabetic' (défaut), 'ideographic', 'bottom'`
  - `fillText(txt,x,y, [maxW])`
    - Rendu de la chaîne 'txt'
    - À la position (x,y)
    - Sans dépasser la largeur maxW (optionnelle)
  - `strokeText(txt,x,y, [maxW])`
  - `measureText(txt) // renvoie la largeur de la chaine en fonction de la configuration de police`

IMAGES

- `drawImage(src, x, y)`
- `drawImage(src, x, y, width, height)`
- `drawImage(src, sX, sY, sW, sH, dX, dY, dW, dH)`
  - Dessine la portion de l'image (src) définie par le rectangle (sX, sY, sW, sH) dans le canvas de destination à la position (dX, dY), redimensionnée au format (dW, dH)

ÉVÈNEMENTS 'TOUCH', GESTURES

- Évènements Javascript
  - touchstart
    - `elem.addEventListener('touchstart', touchStart, false)`
      - `e.target`
      - `e.touches[]`
        - `e.touches[i].pageX / pageY / screenX / screenY / clientX / clientY / target`
  - touchemove
  - touchend
  - touchleave
  - touchcancel

LIBRAIRIES

- HammerJS
- **jGestures**
- QuoJS
- jQuery double tap
- ...

JGESTURES

- ***http://jgestures.codeplex.com/***
- `elem.bind('swipeone', function(el,ev) { // actions });`
- Évènement pinch
  - échelle: ev.scale
- Évènement rotate
  - angle: ev.rotation
- Prévention du zoom

```
<meta name="viewport" content="width= device-width, initial-scale=1, maximum-scale=1, user-scalable=0" />
```

MODULE 4 >

GÉOLOCALISATION ET GOOGLE MAPS API

GÉOLOCALISATION (1)

- Nature du service
  - Latitude + Longitude
  - Source variable
    - GPS
    - IP
    - Wifi, SSID
    - Triangulation appareil mobile
  - Précision, disponibilité, résolution... variables
  - Consommation
  - Vie privée
  - Serveur Web nécessaire
- Détection / Instance
  - `var geo = navigator.geolocation`

GÉOLOCALISATION (2)

- Position actuelle
  - geo.getCurrentPosition(onSuccess, onError, options)
  - onSuccess
    - `Geoposition`
      - `coords`
        - `accuracy`
        - `latitude`
        - `longitude`
        - `...`
      - `timestamp`
  - onError
    - `PositionError`
      - `message`
      - `code`
      - `PERMISSION_DENIED`
      - `POSITION_UNAVAILABLE`
      - `TIMEOUT`
  - options
    - `timeout: (ms)`
    - `enableHighAccuracy: true / false` (important sur mobile pour activer le GPS)
    - `maximumAge: (ms)`

GÉOLOCALISATION (3)

- Suivi de la position
- `geo.watchPosition(onSuccess, onError, options)`
- Exécuté à chaque changement de position

GOOGLE MAPS (1)

- *https://developers.google.com/maps/documentation/javascript/reference*
- Appel de l'API

```
<script type="text/javascript" src="http://maps.google.com/maps/api/js?
sensor=true"></script>
```

- Création d'une position

```
var cPos = new google.maps.LatLng(lat, lon);
```

GOOGLE MAPS (2)

- Création d'une carte
  - `new google.maps.Map(DOMElement, mapOptions)`
    - `mapOptions`
      - `zoom: nombre`
      - `center: latLng`
      - `mapTypeId`
        - `google.maps.MapTypeId.ROADMAP`
        - `google.maps.MapTypeId.SATELLITE`
        - `google.maps.MapTypeId.HYBRID`
        - `google.maps.MapTypeId.TERRAIN`
- Autres fonctions utiles
  - `map.panTo(latLng)`
  - `map.setTilt(angle)`
- Création d'un marqueur

```
var mapMarker = new google.maps.Marker({
     position: latLng,
     title: string
});
```

- Affichage d'un marqueur

```
mapMarker.setMap(map);
```