PAGES WEB POUR TERMINAUX MOBILES

ETIQUETTES FORMULAIRES

Exercice 1

- Ouvrez le fichier **etiquettes_formulaire.html**.
- Créez un fichier CSS et liez-le à votre fichier HTML.
- Ajoutez les lignes suivantes dans le fichier CSS:

```
h1 {font-size: 18pt;}
label
{
      width: 100px;
      text-align: right;
      display: inline-block;
      vertical-align: baseline
}
```

  - Commentez ces règles
- Ecrivez le code CSS nécessaire afin que les étiquettes soient affichées au-dessus des champs du formulaire, et ce pour les terminaux ayant une largeur d'écran inférieure à 480px.
  - Expliquez votre code
- Affichez le fichier HTML dans un navigateur et remarquez les changements.

MEDIA QUERIES

Exercice 2

- Affichez le fichier **media_queries.html** dans le navigateur. Redimensionnez la fenêtre et notez le comportement de la page.
- Nous souhaitons modifier l'affichage des éléments en fonction de la taille de la fenêtre.
  - Pour ce faire, dans le fichier **media_queries.css**, inclure le code existant dans la requête suivante :

```
@media screen and (min-width: 801px) { /* code CSS existant */}
```

  - Toujours dans **media_queries.css**, écrire la requête média correspondant aux écrans de largeur inférieure à 800px. Y inclure le code CSS nécessaire pour produire les affichages suivants:

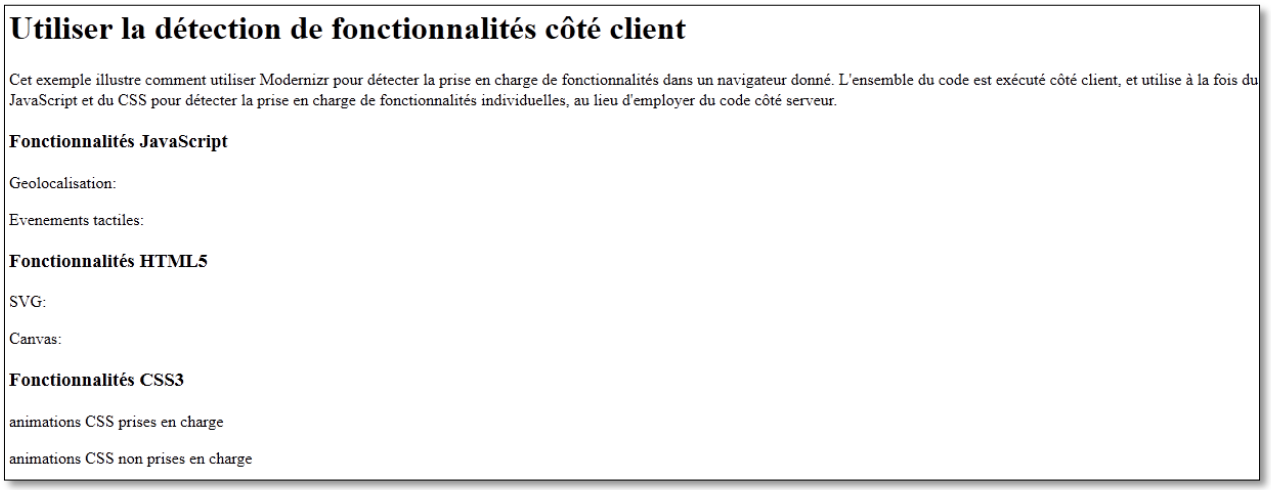
Ecrans de largeur > 800px

Ecrans de largeur < 800px

DÉTECTION DE FONCTIONNALITÉS

Exercice 3

- Récupérez le fichier modernizr.js (soit depuis le lien fourni ici, soit depuis le site Web de Modernizr : https://modernizr.com/)
- Ouvrez le fichier detection_de_fonctionnalites.html avec l'éditeur de votre choix et examinez-en le contenu.



Utiliser la détection de fonctionnalités côté client

Cet exemple illustre comment utiliser Modernizr pour détecter la prise en charge de fonctionnalités dans un navigateur donné. L'ensemble du code est exécuté côté client, et utilise à la fois du JavaScript et du CSS pour détecter la prise en charge de fonctionnalités individuelles, au lieu d'employer du code côté serveur.

Fonctionnalités JavaScript

Geolocalisation:

Evenements tactiles:

Fonctionnalités HTML5

SVG:

Canvas:

Fonctionnalités CSS3

animations CSS prises en charge

animations CSS non prises en charge

- Dans l'en-tête du fichier HTML, ajoutez le chemin vers la librairie Modernizr, ainsi que la balise **<html class= "no-js">**

Exercice 3 – suite

- Copiez le code suivant dans un nouveau fichier JavaScript que vous inclurez dans votre fichier HTML.

```
function testFonctionnalites() {
document.querySelector("#geoloc").innerHTML = Modernizr.geolocation ? "pris en charge" : "non pris en charge";
document.querySelector("#touch").innerHTML = Modernizr.touch ? "pris en charge" : "non pris en charge" ;
document.querySelector("#svg").innerHTML = Modernizr.svg ? "pris en charge" : "non pris en charge";
document.querySelector("#canvas").innerHTML = Modernizr.canvas ? "pris en charge" : "non pris en charge";
}

window.onload = testFonctionnalites;
```

  - Que fait le code ?
  - Quelles fonctionnalités sont testées ?

- Ajoutez le code suivant dans un fichier CSS que vous lierez au fichier HTML (balise <link>)

```
.animtest, .noanimtest { display: none;}
.cssanimations .animtest { display:block;}
.no-cssanimations .noanimtest { display:block;}
```

  - Comment fonctionne ce code ?
  - Que permet-il de faire ?

STOCKAGE LOCAL

Exercice 4

- Ouvrir le fichier **stockage_local.html** avec un éditeur de votre choix. Examinez le contenu du fichier.
  - Quel est son but ? Quelles fonctions JavaScript doivent être définies ?
- Créez un fichier JavaScript, puis faites le lien entre ce fichier et le fichier HTML (balise <link>)
  - Placez les lignes de code suivantes dans le fichier JavaScript

```
function initialize() { /*code à ajouter par la suite */ }
window.onload = initialize;
```

  - Copiez la ligne de code suivante à l'intérieur de la fonction **initialize()**.

```
var bSupportsLocal = (('localStorage' in window) && window['localStorage'] !== null);
```

    - Quel est son rôle ?
- Ajoutez les lignes de code suivantes toujours dans la fonction **initialize()**.

```
if (!bSupportsLocal) {
          document.getElementById('infoform').innerHTML = "<p>Désolé, ce navigateur ne supporte pas l'API Web Storage du W3C.</p>";
          return;
}
```

  - Qu'est-ce qu'elles permettent de faire ?

Exercice 4 – suite

- Copiez les lignes de code suivantes, toujours à la suite dans la fonction **initialize()**.

```
if (window.localStorage.length != 0) {
document.getElementById('firstName').value = window.localStorage.getItem('firstName');
document.getElementById('lastName').value = window.localStorage.getItem('lastName');
document.getElementById('postCode').value = window.localStorage.getItem('postCode');
}
```

  - Décrivez ce qu'elles permettent de faire

- Copiez les lignes de code suivantes, après la fonction **initialize()**.

```
function storeLocalContent(fName, lName, pCode) {
                window.localStorage.setItem('firstName', fName);
                window.localStorage.setItem('lastName', lName);
                window.localStorage.setItem('postCode', pCode);
}
```

  - Quand sont-elles appelées et quel est leur rôle ?

Exercice 4 – suite

- Ecrivez le corps de la fonction permettant de nettoyer le stockage local (e.g. **clearLocalContent**)
- Sauvegardez les fichiers HTML et JS, puis testez le résultat avec un navigateur.
  - Essayez de naviguer vers une autre page, après avoir sauvegardé vos données. Revenez ensuite vers la page. Que s'est-il passé ?

GÉOLOCALISATION

Exercice 5

- Ouvrez le fichier **geopositionnement.html**.
  - Que pouvez-vous dire sur ce que fait (ou est censé faire) ce fichier ?
- Dans la balise **<head>**, ajoutez le lien vers la librairie Modernizr, ainsi que la balise **<html class= “no-js“>.**
- Créez un fichier JavaScript, puis faites le lien entre ce fichier et le fichier HTML (balise <link>)
- Placez les lignes de code suivantes dans le fichier JavaScript

```
function getLocation() { /*code à ajouter par la suite */ }
window.onload = getLocation;
```

  - Quand sera appelée la fonction **getLocation()** ?
- Ajoutez les lignes suivantes dans la fonction **getLocation()**.
  - Quel est leur rôle ?

```
if (Modernizr.geolocation) {

                navigator.geolocation.getCurrentPosition(geoSuccess, geoError);

}
```

Exercice 5 – suite

- Copiez ensuite le code suivant pour les fonctions **geoError** et **geoSuccess**.
  - Quel est leur rôle ?

```
function geoSuccess(positionInfo) {
                document.getElementById("longitude").innerHTML = positionInfo.coords.longitude;
                document.getElementById("latitude").innerHTML = positionInfo.coords.latitude;
                document.getElementById("precision").innerHTML = positionInfo.coords.accuracy;
                document.getElementById("altitude").innerHTML = positionInfo.coords.altitude;
                document.getElementById("precisionAltitude").innerHTML = positionInfo.coords.altitudeAccuracy;
                document.getElementById("cap").innerHTML = positionInfo.coords.heading;
                document.getElementById("vitesse").innerHTML = positionInfo.coords.speed;
}

function geoError(positionError) {
                if (errorInfo.code == 1)
                    alert("L'utilisateur ne souhaite pas partager sa position");
                else if (errorInfo.code == 2)
                    alert("Impossible de déterminer une position");
                else if (errorInfo.code == 3)
                    alert("Délai de recherche de position trop long");
}
```

- Sauvegardez le fichier, puis testez les résultats avec un navigateur et/ou un terminal mobile.
  - Que remarquez-vous ? Commentez les résultats affichés.

Exercice 5 – suite

- Soit la fonction JavaScript suivante (**calculDistance**), permettant de calculer la distance entre deux positions identifiées à travers des couples {latitude, longitude}.
  - En utilisant cette fonction, modifiez le code de la fonction **geoSuccess**, afin de permettre :
    - a) le calcul de la distance entre la position obtenue précédemment et celle de l'ESIREM - {**47.3121519, 5.0039326**}
    - b) l'affichage de cette distance dans la **div id="XXX"**.

```
function calculDistance(startCoords, destCoords) {
    var startLatRads = degreesEnRadians(startCoords.latitude);
    var startLongRads = degreesEnRadians(startCoords.longitude);
    var destLatRads = degreesEnRadians(destCoords.latitude);
    var destLongRads = degreesEnRadians(destCoords.longitude);

    var Radius = 6371; // rayon de la Terre en km
    var distance = Math.acos(Math.sin(startLatRads) * Math.sin(destLatRads) +
                    Math.cos(startLatRads) * Math.cos(destLatRads) *
                    Math.cos(startLongRads - destLongRads)) * Radius;
    return distance;
}
function degreesEnRadians(degrees) {
    radians = (degrees * Math.PI)/180;
    return radians;
}
```