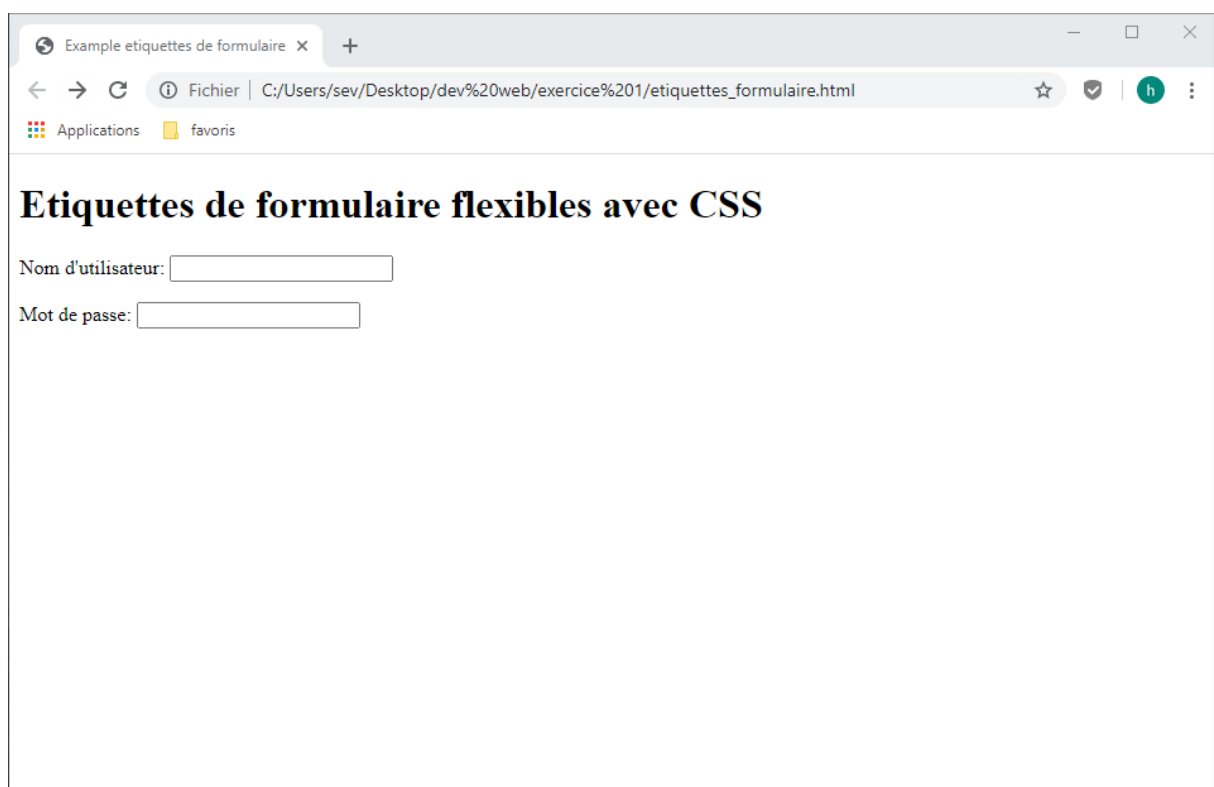


Evaluation Développement Web : pages web pour terminaux mobiles



Exercice 1 : Etiquettes formulaires

- Ouvrir le fichier etiquette_formulaire.html.

```
1 <!DOCTYPE html>
2
3 <html>
4   <head>
5     <title>Exemple etiquettes de formulaire</title>
6     <meta name="viewport" content="width=device-width" />
7   </head>
8   <body>
9     <h1>Etiquettes de formulaire flexibles avec CSS</h1>
10    <form action="" method="get">
11      <p><label for="username">Nom d'utilisateur:</label> <input type="text" /></p>
12      <p><label for="password">Mot de passe:</label> <input type="password" /></p>
13    </form>
14  </body>
15 </html>
16
```



- Créer un fichier CSS et liez-le à votre HTML.

 etiquettes_formulaire.html	12/06/2020 09:48	Chrome HTML Do...	1 Ko
 stylesheet.css	12/06/2020 09:46	Document de feui...	1 Ko

Et pour lier la feuille de style CSS à la page HTML on utilise la balise LINK dans le code HTML :

```
<!DOCTYPE html>

<html>
  <head>
    <title>Exemple etiquettes de formulaire</title>
    <meta name="viewport" content="width=device-width" />
    <link type="text/css" rel="stylesheet" href="stylesheet.css">
  </head>
```

Voir avant dernière ligne du code.

- Ajouter les lignes suivantes dans le fichier CSS et commentez ces règles :

```
1  h1
2  {
3      font-size: 18pt;    /* définit la taille de la police de h1*/
4  }
5  label
6  {
7      width: 100px;
8      /* définit la largeur à 100 pixel */
9      text-align: right;
10     /* aligne le texte à droite */
11     display: inline-block;
12     /* "lie" le label et le input de la même ligne pour une manipulation plus simple */
13     vertical-align: baseline;
14     /* aligne le bas du label avec le bas de l'input */
15 }
```

- 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.

```
@media screen and (max-device-width: 480px){  
  /* récupère la largeur de l'écran et si elle est inférieure à 480px alors on  
  applique le style suivant */  
  label  
  {  
    display: block;  
    /* permet de superposer label et input en block */  
  }  
}
```

- Afficher le fichier HTML dans un navigateur

Pour un grand écran :

Exemple etiquettes de formulaire x +

Fichier | C:/Users/sev/Desktop/dev%20web/exercice%201/etiquettes_formulaire.html

Applications favoris

Etiquettes de formulaire flexibles avec CSS

Nom d'utilisateur:

Mot de passe:

Pour un avec une largeur inférieure à 480px :

Exemple etiquettes de formulaire x +

Fichier | C:/Users/sev/Desktop/dev%20web/exercice%201/etiquett

Applications favoris

Etiquettes de formulaire flexibles avec CSS

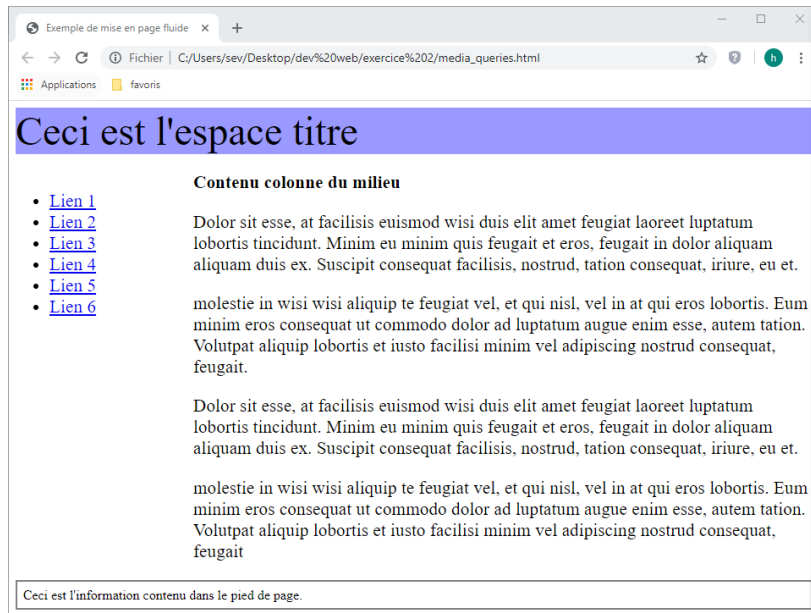
Nom d'utilisateur:

Mot de passe:

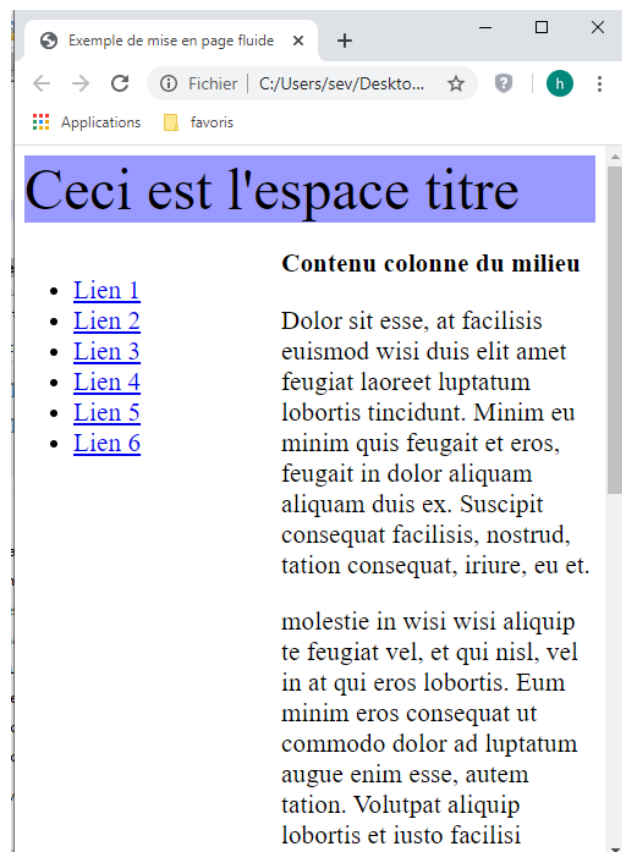
Remarque : on voit bien les changements grâce à la feuille de style CSS, notamment sur les petits écrans où les labels sont maintenant au-dessus des inputs.

Exercice 2 : Media Queries

- Affichez le fichier media_queries.html dans le navigateur. Redimensionnez la fenêtre et notez le comportement de la page.



On remarque que quel que soit la taille de la fenêtre la disposition ne s'adapte pas, seul le textes des paragraphes revient à la ligne quand il le faut.



- Nous souhaitons modifier l'affichage des éléments en fonction de la taille de la fenêtre.

On récupère le code media_queries.css :

```
media_queries.css  media_queries.html x
1  #titrePage {
2      font-size:36pt;
3      font-family:"Times New Roman", Times, serif;
4      background-color:#9999FF
5  }
6
7  #container {
8      font-size: 16pt;
9      position: relative;
10     width: 100%;
11 }
12
13 #colonneG {
14     width: 200px;
15     height: 100%;
16     float:left;
17 }
18
19 #contenuPage {
20     margin-left: 210px;
21 }
22
23 #footer {
24     border: 2px gray solid;
25     padding: 5pt;
26     margin-top: 5pt;
27 }
28
```

Et on ajoute le code existant dans la requête « @media screen and (min-width: 801px) {} » :

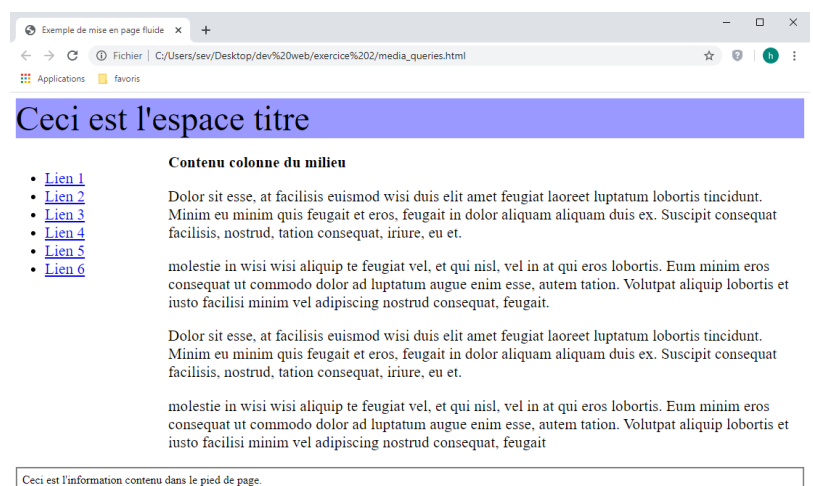
```
media_queries.css  media_queries.html x
1  @media screen and (min-width: 801px) {
2      #titrePage {
3          font-size:36pt;
4          font-family:"Times New Roman", Times, serif;
5          background-color:#9999FF
6      }
7
8      #container {
9          font-size: 16pt;
10         position: relative;
11         width: 100%;
12     }
13
14     #colonneG {
15         width: 200px;
16         height: 100%;
17         float:left;
18     }
19
20     #contenuPage {
21         margin-left: 210px;
22     }
23
24     #footer {
25         border: 2px gray solid;
26         padding: 5pt;
27         margin-top: 5pt;
28     }
29 }
```

Voir ligne 1

Enfin, toujours dans media_queries.css, on crée la requête pour les écrans inférieurs à 800px qui nous permet de obtenir le résultat demandé :



```
@media screen and (max-width: 800px)
{
  #titrePage
  {
    font-size:36pt;
    font-family:"Times New Roman", Times, serif;
    background-color:#999900;
    color:white;
    text-align: center;
  }
  #container
  {
    font-size: 16pt;
    position: relative;
    width: 100%;
  }
  #colonneG
  {
    width: 100%;
    height: 100%;
    float:up;
  }
  ul li{
    display: inline;
  }
  #footer {
    border: 2px gray solid;
    padding: 5pt;
    margin-top: 5pt;
  }
}
```

On obtient bien les deux rendus désirés :



Exercice 3 : Détection de fonctionnalités

- On récupère le fichier modernizr.js et le fichier detection_de_fonctionnalites.html :

 detection_de_fonctionnalites.html	12/06/2020 14:04	Chrome HTML Do...	2 Ko
 modernizr.js	12/06/2020 14:02	Fichier de JavaScript	29 Ko

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

```
1 <!DOCTYPE html>
2
3 <html>
4   <head>
5     <title>Utiliser la détection de fonctionnalités côté client</title>
6     <meta charset="UTF-8"/>
7     <meta name="viewport" content="width=device-width" />
8   </head>
9   <body>
10    <h1>Utiliser la détection de fonctionnalités côté client</h1>
11    <p>Cet exemple illustre comment utiliser Modernizr pour détecter la prise en charge de
12      fonctionnalités dans un navigateur donné.
13      L'ensemble du code est exécuté côté client, et utilise à la fois du JavaScript et
14      du CSS pour détecter la prise en charge
15      de fonctionnalités individuelles, au lieu d'employer du code côté serveur.</p>
16    <h3>Fonctionnalités JavaScript</h3>
17    <p>Geolocalisation: <span id="geoloc"></span></p>
18    <p>Evenements tactiles: <span id="touch"></span></p>
19    <h3>Fonctionnalités HTML5</h3>
20    <p>SVG: <span id="svg"></span></p>
21    <p>Canvas: <span id="canvas"></span></p>
22    <h3>Fonctionnalités CSS3</h3>
23    <p class="animtest">animations CSS prises en charge</p>
24    <p class="noanimtest">animations CSS non prises en charge</p>
25  </body>
26</html>
```

- Dans l'en-tête du fichier HTML on ajoute le chemin vers la librairie Modernizr ainsi que la balise `<html class= "no-js">` : voir ligne 8 et 9

```
4      <head>
5          <title>Utiliser la détection de fonctionnalités côté client</title>
6          <meta charset="UTF-8"/>
7          <meta name="viewport" content="width=device-width" />
8          <script type="text/javascript" src="modernizr.js"></script>
9          <html class= "no-js">
```

- On copie le code qui nous est donné dans un nouveau fichier JavaScript et on l'inclue lui aussi à notre fichier HTML (ligne 9 dans la 3^e figure):

 detection_de_fonctionnalites.html	12/06/2020 14:11	Chrome HTML Do...	2 Ko
 modernizr.js	12/06/2020 14:02	Fichier de JavaScript	29 Ko
 script.js	12/06/2020 14:15	Fichier de JavaScript	0 Ko

```
1 function testFonctionnalites()
2 {
3     document.querySelector("#geoloc").innerHTML = Modernizr.geolocation ? "pris en charge" : "non pris
4     document.querySelector("#touch").innerHTML = Modernizr.touch ? "pris en charge" : "non pris
5     document.querySelector("#svg").innerHTML = Modernizr.svg ? "pris en charge" : "non pris en c
6     document.querySelector("#canvas").innerHTML = Modernizr.canvas ? "pris en charge" : "non pri
7 }
8
9 window.onload = testFonctionnalites;
```

```
4      <head>
5          <title>Utiliser la détection de fonctionnalités côté client</title>
6          <meta charset="UTF-8"/>
7          <meta name="viewport" content="width=device-width" />
8          <script type="text/javascript" src="modernizr.js"></script>
9          <script type="text/javascript" src="script.js"></script>
10         <html class= "no-js">
```

Ce code JavaScript teste 4 fonctionnalités de notre document, notre page HTML, et adapte le texte en fonction pour nous en informer.

Les 4 fonctionnalités testées sont :

- « geolocation », détecte la prise en charge de l'API de géolocalisation pour que les utilisateurs fournissent leur emplacement aux applications Web.
- « touch », détecte si la prise en charge du tactile
- « svg », détecte la prise en charge de SVG dans `<embed>` ou des éléments `<object>`
- « canvas », détecte la prise en charge de l'élément `<canvas>` pour le dessin 2D

- On ajoute le code CSS suivant dans un fichier que nous lierons au fichier HTML (voir ligne 11 du fichier HTML) :

detection_de_fonctionnalites.html	12/06/2020 14:18	Chrome HTML Do...	2 Ko
modernizr.js	12/06/2020 14:02	Fichier de JavaScript	29 Ko
script.js	12/06/2020 14:20	Fichier de JavaScript	1 Ko
stylesheet.css	12/06/2020 14:35	Document de feui...	0 Ko

```
4      <head>
5          <title>Utiliser la détection de fonctionnalités côté client</title>
6          <meta charset="UTF-8"/>
7          <meta name="viewport" content="width=device-width" />
8          <script type="text/javascript" src="modernizr.js"></script>
9          <script type="text/javascript" src="script.js"></script>
10         <html class="no-js">
11         <link type="text/css" rel="stylesheet" href="stylesheet.css">
12     </head>
```

```
1  .animtest, .noanimtest
2  {
3      display: none;
4  }
5  .cssanimations .animtest {
6      display: block;
7  }
8  .no-cssanimations .noanimtest {
9      display: block;
10 }
```

Le code que nous venons d'ajouter dans notre fichier CSS sert à afficher si oui ou non les animations CSS sont prises en charge. Pour mieux comprendre il faut aussi regarder dans le fichier HTML ligne 25 et 26 :

```
25     <p class="animtest">animations CSS prises en charge</p>
26     <p class="noanimtest">animations CSS non prises en charge</p>
```

Dans un premier temps, le premier bloc de notre fichier CSS empêche l'affichage DES 2 textes.

Ensuite les deux autres blocs testent si les animations CSS sont prises en charges ou non et affiche ou n'affiche pas le texte qui leur est associé en fonction. Ainsi l'utilisateur n'aura d'afficher qu'un seul texte qui lui indiquera si oui ou non les animations CSS sont prises en charge. Tout le code est donc coté client et non coté serveur.

Exercice 4 : stockage local

- On ouvre le fichier stockage_local.html.

Utilisation du DOM localStorage

L'[API Web Storage du W3C](#) fournit 2 nouvelles manières pour stocker des informations côté client - les objets `sessionStorage` et `localStorage`. Cette démo illustre comment utiliser la fonctionnalité `localStorage` pour enregistrer des informations dans le navigateur sans avoir à utiliser les cookies.

Exemple de formulaire

Essayez de saisir des informations dans le formulaire, puis fermez et rouvrez la page.

Informations personnelles

Prénom:

Nom:

Code postal:

```
1 <!DOCTYPE html>
2 <html>
3   <head>
4     <title>API Web Storage du W3C</title>
5     <meta charset="UTF-8"/>
6     <meta name="Viewport" content="width=device-width" />
7   </head>
8   <body>
9     <h1>Utilisation du DOM localStorage</h1>
10    <div>
11      L'<a href="http://www.w3.org/TR/webstorage/">API Web Storage du W3C</a>
12      fournit 2 nouvelles manières pour stocker des informations côté client -
13      les objets <code>sessionStorage</code> et <code>localStorage</code>. Cette démo
14      illustre
15      comment utiliser la fonctionnalité localStorage pour enregistrer des informations
16      dans le navigateur
17      sans avoir à utiliser les cookies.</div>
18    <h2>Exemple de formulaire</h2>
19    <p>Essayez de saisir des informations dans le formulaire, puis fermez et rouvrez la
20    page.</p>
21    <div>
22      <form action="" method="post" id="infoform">
23        <fieldset name="LocalStorage" id="ls">
24          <legend>Informations personnelles</legend>
25          <p><label id="fnLabel" for="firstName">Prénom: </label>
26          <input id="firstName" name="firstName" /></p>
27          <p><label id="lnLabel" for="lastName">Nom: </label>
28          <input id="lastName" name="lastName" /></p>
29          <p><label id="pcLabel" for="postCode">Code postal: </label>
30          <input id="postCode" name="postCode" /></p>
31          <input type="button" value="Enregistrer" onclick="storeLocalContent(
32            document.querySelector('#firstName').value,
33            document.querySelector('#lastName').value,
34            document.querySelector('#postCode').value
35          )" />
36          <input type="button" value="Effacer" onclick="clearLocalContent()" />
37        </fieldset>
38      </form>
39    </div>
40  </body>
41 </html>
```

Le but de ce fichier HTML est de faire une démo en sauvegardant localement les informations entrées dans ce formulaire afin de les ressortir quand on ouvre à nouveau le fichier dans le navigateur. Pour ceci on utilise les fonctions « `storeLocalContent()` », « `querySelector()` » et « `clearLocalContent` » (lignes 28 à 33).

- On crée un fichier JavaScript et on fait le lien entre celui-ci et le fichier HTML, ligne 7 de la 3^e figure :

script.js	12/06/2020 15:19	Fichier de JavaScript	0 Ko
stockage_local.html	12/06/2020 15:05	Chrome HTML Do...	2 Ko

```
1 function initialize()
2 {
3     var bSupportsLocal = (('localStorage' in window) && window['localStorage'] !== null);
4 }
5 window.onload = initialize;
6
```

On crée la variable `bSupportLocal` et on lance la fonction « `initialize()` » au lancement de la fenêtre.

```
3 <head>
4 <title>API Web Storage du W3C</title>
5 <meta charset="UTF-8"/>
6 <meta name="Viewport" content="width=device-width" />
7 <script type="text/javascript" src="script.js"></script>
8 </head>
```

- On ajoute les lignes données dans la fonction « `initialize()` » de notre fichier JavaScript (lignes 5 à 9) :

```
1 function initialize()
2 {
3     var bSupportsLocal = (('localStorage' in window) && window['localStorage'] !== null);
4
5     if (!bSupportsLocal)
6     {
7         document.getElementById('infoform').innerHTML = "<p>Désolé, ce navigateur ne supporte pas
8         return;
9     }
10 }
11 window.onload = initialize;
```

Ces lignes permettent de tester si le navigateur supporte l'API Web Storage du W3C et si ce n'est pas le cas alors le texte de « `infoform` » change pour l'indiquer à l'utilisateur.

- Toujours dans la fonction « initialize() » on ajoute le code donné (lignes 11 à 16) :

```
10
11     if (window.localStorage.length != 0)
12     {
13         document.getElementById('firstName').value = window.localStorage.getItem('firstName');
14         document.getElementById('lastName').value = window.localStorage.getItem('lastName');
15         document.getElementById('postCode').value = window.localStorage.getItem('postCode');
16     }
```

- Ce code vérifie d'abord si des données sont stockées « if (window.localStorage.length != 0) ».
 - Si c'est le cas alors les inputs qui correspondent aux IDs « firstName », « lastName » et « postCode » prennent comme valeur, la valeur récupérée dans le stockage local avec les IDs du même nom.
- Encore une fois on ajoute du code dans le fichier JavaScript mais cette fois après la fonction « initialize() », lignes 19 à 24 :

```
1  function initialize()
2  {
3      var bSupportsLocal = (('localStorage' in window) && window['localStorage'] !== null);
4
5      if (!bSupportsLocal)
6      {
7          document.getElementById('infoform').innerHTML = "<p>Désolé, ce navigateur ne supporte pas";
8          return;
9      }
10
11     if (window.localStorage.length != 0)
12     {
13         document.getElementById('firstName').value = window.localStorage.getItem('firstName');
14         document.getElementById('lastName').value = window.localStorage.getItem('lastName');
15         document.getElementById('postCode').value = window.localStorage.getItem('postCode');
16     }
17 }
18
19 function storeLocalContent(fName, lName, pCode)
20 {
21     window.localStorage.setItem('firstName', fName);
22     window.localStorage.setItem('lastName', lName);
23     window.localStorage.setItem('postCode', pCode);
24 }
25
26 window.onload = initialize;
```

Les lignes que nous venons d'ajouter sont exécutées quand la fonction « storeLocalContent() » est appelée, c'est-à-dire quand on clique sur le bouton « enregistrer », voir image ci-dessous (lignes 29 à 33 du fichier HTML) :

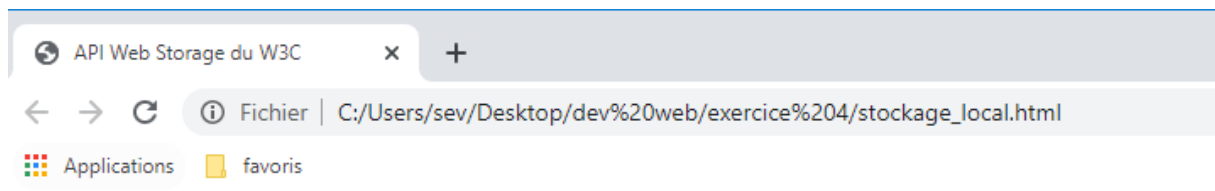
```
29     <input type="button" value="Enregistrer" onclick="storeLocalContent(
30         document.querySelector('#firstName').value,
31         document.querySelector('#lastName').value,
32         document.querySelector('#postCode').value
33     )">
```

Cette fonction (« storeLocalContent() ») prend en argument 3 valeurs, le nom, le prénom et le code postal, qu'elle récupère des inputs avec leurs IDs. Une fois la fonction appelée avec ces 3 valeurs, elle les stocke dans le stockage local grâce à 3 IDs différents.

- On écrit le corps de la fonction permettant de nettoyer le stockage local, "clearLocalContent()", qui elle n'a pas besoin de paramètre.

```
26  function clearLocalContent(){
27      window.localStorage.removeItem('firstName');
28      window.localStorage.removeItem('lastName');
29      window.localStorage.removeItem('postCode');
30  }
31
```

- Enfin on peut tester avec un navigateur après avoir sauvegardé.



Utilisation du DOM localStorage

L'[API Web Storage du W3C](#) fournit 2 nouvelles manières pour stocker des informations côté client - les objets : Cette démo illustre comment utiliser la fonctionnalité localStorage pour enregistrer des informations dans le navigateur.

Exemple de formulaire

Essayez de saisir des informations dans le formulaire, puis fermez et rouvrez la page.

Informations personnelles

Prénom: hugo

Nom: blain

Code postal: 21000

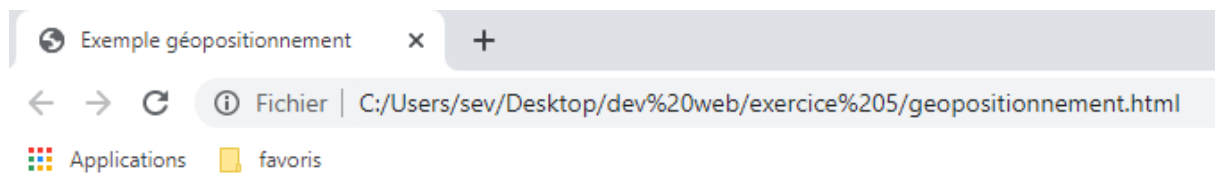
Enregistrer

Effacer

On remarque que si on essaye de naviguer vers une autre page, après avoir sauvegardé, les données sont à nouveau dans les cases sans avoir à les remettre.

Exercice 5 : Géolocalisation

- On ouvre le fichier geopositionnement.html



Exemple géopositionnement

Cet exemple illustre comment utiliser la fonction de géopositionnement du terminal mobile.

Données de position:

Longitude:

Latitude:

Précision:

Altitude:

Précision altitude:

Cap:

Vitesse:

Distance de l'ESIREM:


```
1 <!DOCTYPE html>
2
3 <html>
4   <head>
5     <title>Exemple géopositionnement</title>
6     <meta name="viewport" content="width=device-width" />
7     <meta charset="UTF-8"/>
8   </head>
9   <body>
10    <h1>Exemple géopositionnement</h1>
11    <p>Cet exemple illustre comment utiliser la fonction de géopositionnement du terminal
12    mobile.</p>
13    <h2>Données de position:</h2>
14    <p>Longitude: <span id="longitude"></span></p>
15    <p>Latitude: <span id="latitude"></span></p>
16    <p>Précision: <span id="precision"></span></p>
17    <p>Altitude: <span id="altitude"></span></p>
18    <p>Précision altitude: <span id="precisionAltitude"></span></p>
19    <p>Cap: <span id="cap"></span></p>
20    <p>Vitesse: <span id="vitesse"></span></p>
21    <p>Distance de l'ESIREM: <span id="distance"></span></p>
22  </body>
23 </html>
```

Ce fichier est sensé nous montrer les différentes informations que nous pouvons récupérer d'un terminal mobile une application telle que le calcul de la distance entre le terminal mobile et l'ESIREM.

- On ajoute dans la balise <head> du fichier HTML le lien vers la librairie Modernizr, ainsi que la balise <html class= "no-js"> (lignes 7 et 8).

```
4  <head>
5      <title>Exemple géopositionnement</title>
6      <meta name="viewport" content="width=device-width" />
7      <script src="modernizr.js"></script>
8      <html class= "no-js">
9      <meta charset="UTF-8"/>
10 </head>
```

- On crée un fichier JavaScript et on fait le lien entre lui et le fichier HTML ligne 8 dans la deuxième figure :

 geopositionnement.html	12/06/2020 16:58	Chrome HTML Do...	1 Ko
 script.js	12/06/2020 16:57	Fichier de JavaScript	0 Ko

```
4  <head>
5      <title>Exemple géopositionnement</title>
6      <meta name="viewport" content="width=device-width" />
7      <script src="modernizr.js"></script>
8      <script type="text/javascript" src="script.js"></script>
9      <html class= "no-js">
10 <meta charset="UTF-8"/>
11 </head>
```

- On met le code fournis dans le fichier JavaScript :

```
1  function getLocation()
2  {
3      /*code à ajouter par la suite */
4  }
5  window.onload = getLocation;
```

Cette fonction « getLocation() » sera appelée quand la fenêtre se lance.

- On ajoute les lignes données à notre fonction :

```
1 function getLocation()
2 {
3     if (Modernizr.geolocation)
4     {
5         navigator.geolocation.getCurrentPosition(geoSuccess, geoError);
6     }
7 }
8 window.onload = getLocation;
```

- D'abord on détecte si l'API de géolocalisation est prise en charge.
- Si c'est le cas alors on accède à la position actuelle de l'appareil avec la méthode « `getCurrentPosition()` ».

- On copie le code donné pour les fonctions « `geoError` » et « `geoSuccess` » :

```
9 function geoSuccess(positionInfo)
10 {
11     document.getElementById("longitude").innerHTML = positionInfo.coords.longitude;
12     document.getElementById("latitude").innerHTML = positionInfo.coords.latitude;
13     document.getElementById("precision").innerHTML = positionInfo.coords.accuracy;
14     document.getElementById("altitude").innerHTML = positionInfo.coords.altitude;
15     document.getElementById("precisionAltitude").innerHTML = positionInfo.coords.altitudeAccuracy;
16     document.getElementById("cap").innerHTML = positionInfo.coords.heading;
17     document.getElementById("vitesse").innerHTML = positionInfo.coords.speed;
18 }
19
20 function geoError(positionError)
21 {
22     if (errorInfo.code == 1)
23         alert("L'utilisateur ne souhaite pas partager sa position");
24     else if (errorInfo.code == 2)
25         alert("Impossible de déterminer une position");
26     else if (errorInfo.code == 3)
27         alert("Délai de recherche de position trop long");
28 }
29
```

La fonction « `geoSuccess` » a un argument, « `positionInfo` ». C'est de cet objet qu'elle va tirer toutes les informations nécessaires pour remplir les paragraphes, à l'aide de leurs IDs, comme Longitude, Latitude, Précision, Altitude, Précision altitude, Cap et Vitesse.

Par exemple pour récupérer la longitude : `positionInfo.coords.longitude`




La fonction « `geoError` », elle, va servir à informer l'utilisateur en cas d'erreur. Selon la valeur du code d'erreur ce sera une alerte différente qui sera envoyée. Pour prévoir les différents cas possibles on utilise un « `if/else if` ».

Remarque : l'argument récupéré et les variables dans les « if » n'ont pas le même nom. Je suppose que c'est une erreur et pour que la fonction fonctionne j'ai corrigé.

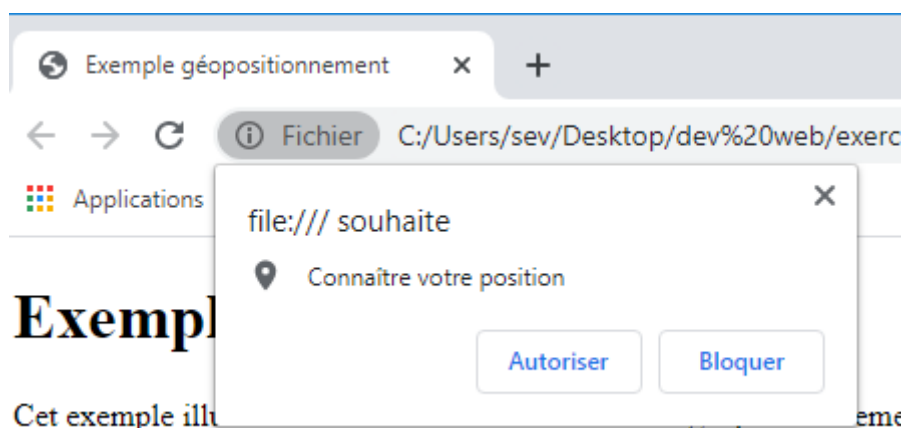
```
function geoError(positionError)
{
    if (positionError.code == 1)
        alert("L'utilisateur ne souhaite pas partager sa position");
    else if (positionError.code == 2)
        alert("Impossible de déterminer une position");
    else if (positionError.code == 3)
        alert("Délai de recherche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

➤ On sauvegarde et on lance la page dans notre navigateur :

Dans un premier temps on remarque que rien ne se passe, il manque le fichier « Modernizr.js ». Nous le rajoutons.

 geopositionnement.html	12/06/2020 17:20	Chrome HTML Do...	1 Ko
 modernizr.js	12/06/2020 11:27	Fichier de JavaScript	29 Ko
 script.js	12/06/2020 17:20	Fichier de JavaScript	2 Ko

Ensuite lors du second essaie on remarque que la page nous demande l'autorisation pour utiliser notre position.



Si on bloque alors la page reste comme au début alors que si on accepte on observe que certaines informations apparaissent, mais pas toutes, sûrement car j'utilise un ordinateur fixe et non un terminal mobile. Enfin la distance entre l'ESIREM et moi n'est calculée encore nulle part alors il est logique qu'elles n'apparaissent pas.

Exemple géopositionnement

Cet exemple illustre comment utiliser la fonction de géopositionnement du terminal mobile.

Données de position:

Longitude: 5.4853632

Latitude: 47.0810624

Précision: 3537

Altitude:

Précision altitude:

Cap:

Vitesse:

Distance de l'ESIREM:

- On utilise le code qui nous est fourni pour calculer et afficher la distance qui nous sépare de l'ESIREM à l'aide de coordonnées (longitude et latitude).

On ajoute les deux fonctions :

```
30 function calculDistance (startCoords, destCoords){
31     var startLatRads = degreesEnRadians(startCoords.latitude);
32     var startLongRads = degreesEnRadians(startCoords.longitude);
33     var destLatRads = degreesEnRadians(destCoords.latitude);
34     var destLongRads = degreesEnRadians(destCoords.longitude);
35     var Radius = 6371; // rayon de la Terre en km
36     var distance = Math.acos(Math.sin(startLatRads)*Math.sin(destLatRads)+Math.cos(startLatRads)*
37     return distance;
38 }
39
40 function degreesEnRadians(degrees){
41     radians =(degrees * Math.PI)/180;
42     return radians;
43 }
```

On modifie la fonction « geoSuccess() », lignes 19 et 22:

```
9 function geoSuccess(positionInfo)
10 {
11     document.getElementById("longitude").innerHTML = positionInfo.coords.longitude;
12     document.getElementById("latitude").innerHTML = positionInfo.coords.latitude;
13     document.getElementById("precision").innerHTML = positionInfo.coords.accuracy;
14     document.getElementById("altitude").innerHTML = positionInfo.coords.altitude;
15     document.getElementById("precisionAltitude").innerHTML = positionInfo.coords.altitudeAccuracy;
16     document.getElementById("cap").innerHTML = positionInfo.coords.heading;
17     document.getElementById("vitesse").innerHTML = positionInfo.coords.speed;
18
19     let coordonneesEsirem = {"longitude":5.0039326,"latitude":47.3121519};
20     var distance = calculDistance(positionInfo.coords, coordonneesEsirem);
21     document.getElementById("distance").innerHTML = distance + " km";
22 }
23 }
```

On observe bien maintenant le résultat sur notre page :

Exemple géopositionnement

Cet exemple illustre comment utiliser la fonction de géopositionnement du terminal mobile.

Données de position:

Longitude: 5.4853632

Latitude: 47.0810624

Précision: 3537

Altitude:

Précision altitude:

Cap:

Vitesse:

Distance de l'ESIREM: 44.53519490740232 km