Armand AUDEOUD

IT3A

Pages Web pour Terminaux Mobiles

Compte rendu

Table des matières

Table des illustrations

Exercice 1 – Etiquettes Formulaires

Après avoir créé notre fichier CSS, la première chose à faire est de le lier à notre fichier HTML :

```
<link rel="stylesheet" type="text/css" href="etiquettes_formulaire.css">
```

À la suite de cela, nous allons pouvoir noter des changements sur notre site web, notamment grâce à notre fichier CSS. Voici respectivement l'affichage sur une page de largeur supérieure à 480px puis inférieure à 480px :

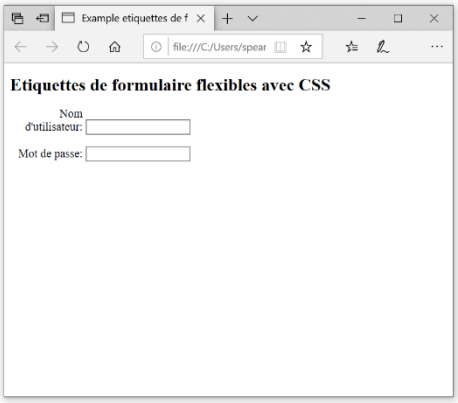

Figure 1 – formulaire > 480px

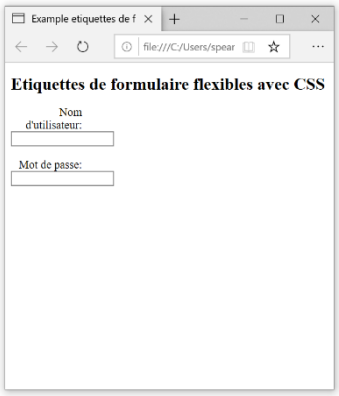

Figure 2 – formulaire < 480px

On peut modifier la mise en page en fonction de la largeur de la fenêtre grâce à ces lignes :

```
@media screen and ( min-width: 481px )
```

```
@media screen and ( max-width: 480px )
```

On a respectivement le cas pour une fenêtre de largeur supérieure à 480px, c'est-à-dire quand les étiquettes sont alignées par rapport aux champs, et le cas pour une fenêtre de largeur inférieure à 480px, c'est-à-dire quand les étiquettes sont affichées au-dessus des champs.

Exercice 2 – Media Queries

De la même manière que dans l'exercice précédent, on va pouvoir appliquer différentes mises en page en fonction de la largeur de la fenêtre :

```
@media screen and (min-width: 801px)
```

```
@media screen and (max-width: 800px)
```

Afin d'obtenir les changements souhaités, deux balises sont à modifier dans le fichier CSS :

```
#titrePage {
    font-size:36pt;
    font-family:"Times New Roman", Times, serif;
    color: #FFFFFF;
    background-color:#336600;
}
```

```
#colonneG {
    width: 100%;
    height: 100%;
    float: top;
}
```

```
#colonneG li {
    display: inline-block;
}
```

Ces modifications vont respectivement changer la couleur de police et de fond de la partie titre, placer la colonne de gauche en haut puis aligner les liens sur une ligne et non sur une colonne.

De plus, les modifications sur la balise contenuPage n'est plus utile, on peut donc les retirer.

Figure 4 – page < 800px

Figure 3 – page > 800px

Voici à droite notre page web lorsque la largeur de la fenêtre est supérieure à 800px (*Figure 3*) et à gauche lorsque la largeur est inférieure à 800px (*Figure 4*).

Exercice 3 – Détection de fonctionnalités

Le code fourni va permettre à notre script de déterminer si des fonctionnalités sont présentes sur l'appareil du client. En fonction du résultat, on va voir apparaître sur la page "pris en charge" ou "non pris en charge".

On va pouvoir tester si l'appareil dispose de la géolocalisation, des événements tactiles, du et du canvas en HTML5.

De même pour le fichier CSS, on va pouvoir vérifier si le client dispose bien de la prise en charge des animations en CSS. En fonction du résultat obtenu, la fonction va masquer une balise, soit <p class="animtest">, soit <p class="noanimtest">.

Figure 5 – détection des fonctionnalités côté client

On peut remarquer qu'ici, seuls les événements tactiles ne sont pas pris en charge (*Figure 5*).

Exercice 4 – Stockage local

Ce fichier sert à créer un formulaire qui va pouvoir enregistrer des informations. Pour cela, il faut affecter des fonctionnalités aux boutons grâce au JavaScript.

La ligne suivante va permettre de vérifier si la fonctionnalité est bien supportée par le navigateur utilisé par le client :

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

Par la suite, on va pouvoir avertir l'utilisateur, grâce à un if, si l'API est supportée par son navigateur ou pas. En effet, on place le résultat du test dans une variable qu'on va ensuite utiliser afin de d'effectuer le test. Si le test se révèle négatif, on affiche « Désolé, ce navigateur ne supporte pas l'API Web Storage du W3C. » à la place du formulaire.

```
if (window.localStorage.length != 0)
```

La condition ci-dessus nous permet de vérifier si les champs des formulaires ont bien été remplis. Si cela est le cas, alors on les enregistre grâce à window.localStorage.getItem :

```
document.getElementById('firstName').value = window.localStorage.getItem('firstName');
document.getElementById('lastName').value = window.localStorage.getItem('lastName');
document.getElementById('postCode').value = window.localStorage.getItem('postCode');
```

La fonction Initialize sera appelée à chaque rafraîchissement de la page.

C'est ici que nous allons pouvoir enregistrer les champs de notre formulaire. Les chaines de caractères seront placées dans leurs variables respectives :

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

Ces fonctions seront appelées à chaque validation du formulaire.

Enfin, voici notre fonction qui va permettre d'effacer les données locales :

```
function clearLocalContent()
{
    window.localStorage.clear();
    document.getElementById('firstName').value = window.localStorage.getItem('');
    document.getElementById('lastName').value = window.localStorage.getItem('');
    document.getElementById('postCode').value = window.localStorage.getItem('');
}
```

On efface les données locales et on efface le contenu des champs du formulaire.

Nous voici cette fois-ci sur Google Chrome, Microsoft Edge posant problème pour cet exercice. On peut constater que les données entrées sont bien enregistrées dans le navigateur. Ces dernières sont toujours là lorsqu'on actualise la page voire même quand on relance le navigateur :

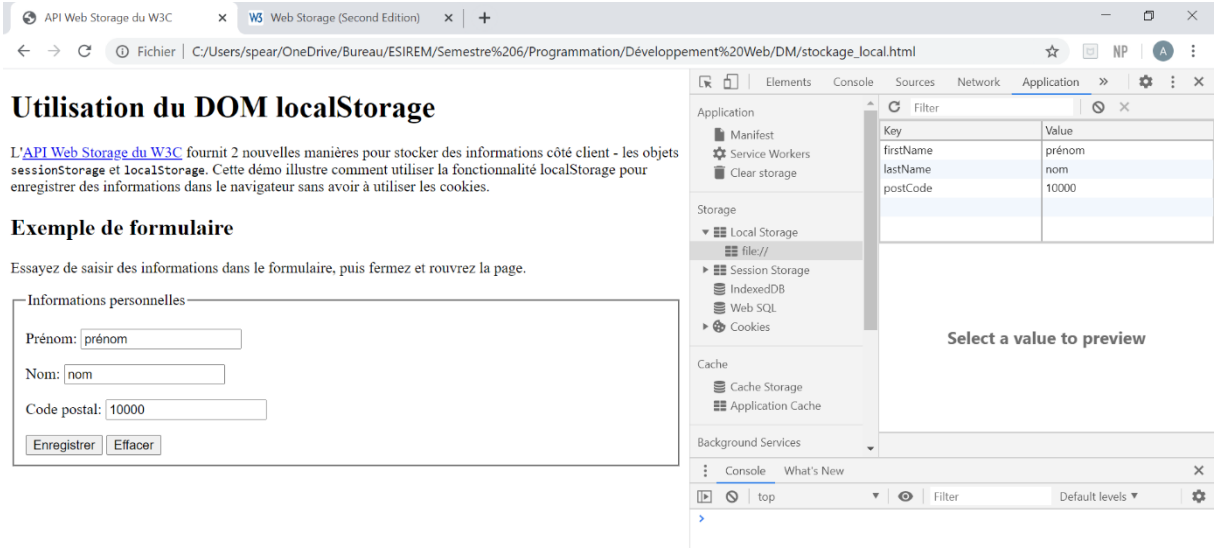

Figure 6 – utilisation du DOM localStorage

Ces données seront bien supprimées une fois qu'on aura sélectionné le bouton Effacer :

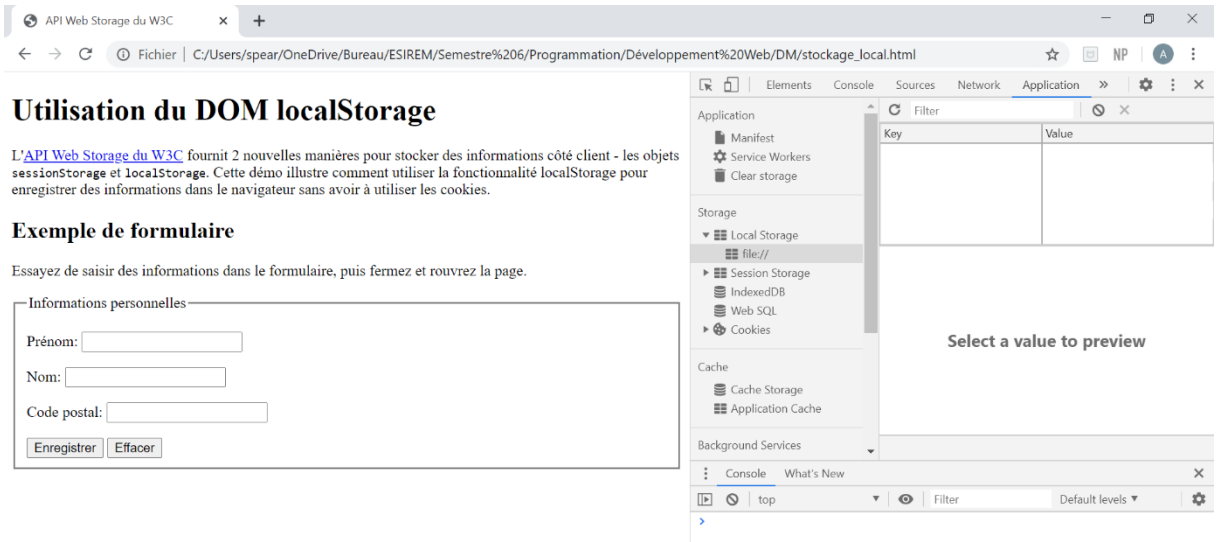

Figure 7 – effacer les données du DOM localStorage

Exercice 5 – Géolocalisation

On peut supposer que la page web va afficher les coordonnées GPS de l'appareil de l'utilisateur. Dans le fichier HTML, on peut y voir toutes les informations qui seront affichées.

La fonction getLocation sera appelée dès que la page web s'ouvrira. Dans cette fonction, on va tout d'abord afficher si les fonctions de la classe Modernizr peuvent bien accéder aux coordonnées GPS de l'appareil.

```
if (Modernizr.geolocation) {
    navigator.geolocation.getCurrentPosition(geoSuccess, geoError);
}
```

Par la suite, on va avoir deux méthodes : getError et geoSuccess. geoSuccess va nous permettre d'obtenir les informations souhaitées. Quant à getError, elle va nous permettre d'afficher l'erreur correspondante (soit un refus d'autorisation, soit une impossibilité à géolocaliser, soit un délais trop long).

Dès qu'on lance la page web, on reçoit cette notification :

Figure 8 – notification position

Si on accepte, on obtient les informations suivantes :

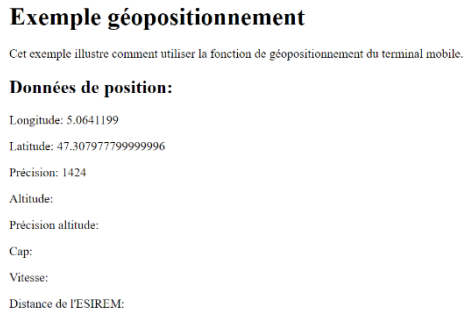

Figure 9 – géo positionnement

Néanmoins, si on refuse, rien ne se passe. Si on regarde la console dans le navigateur, on peut déduire que les numéros d'erreur n'ont pas été définis :

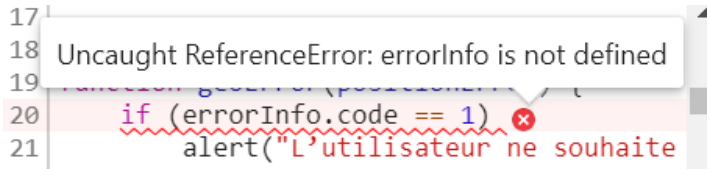

Figure 10 – refus de partager la localisation

Afin de pouvoir indiquer la distance entre la position de l'appareil de l'utilisateur et l'ESIREM, il faut ajouter ces quelques lignes à la fonction geoSuccess :

```
var esirem = {"latitude":47.3121519, "longitude":5.0039326};
var maison = {"latitude":positionInfo.coords.latitude, "longitude":positionInfo.coords.longitude};
var distanceEsirem = calculDistance(maison, esirem);

document.getElementById('distance').innerHTML = distanceEsirem;
```

Nous avons à présent notre distance par rapport à l'ESIREM :

Distance de l'ESIREM: 4.561409882217043

Figure 11 – distance ESIREM