ToDo

HTML
CSS
JavaScript - Jquery

Environnement de développement

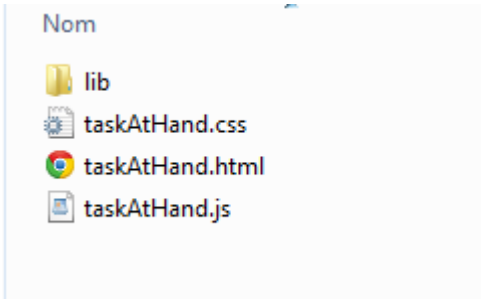
Nous allons créer notre premier Web application, une liste de tâche.
les listes de tâche (ToDo liste) sont devenues un bon exemple pour apprehender et comparer des bibliothèques de developpement.

Dans la suite, nous n'allons pas utiliser de bibliothèques.

Application HTML5

Nous pourrons compléter cette structure par l'ajout de dossiers pour inclure des images, son.

Nous allons commencer par créer la structure de notre application.



Introduction à jquery

Jquery

Télécharger une version de Jquery dans le dossier lib.

HTML

Ouvrir votre éditeur est taper le code suivant.

```
<!DOCTYPE html>
<html>
<head>
    <title>Task@Hand</title>
    <link href="taskAtHand.css" rel="StyleSheet" />
    <script src="lib/jquery-1.8.1.min.js"></script>
    <script src="taskAtHand.js"></script>
</head>
```

Javascript

Ecrire le code suivant dans le fichier taskAtHand.js.

Rassurez vous, *nous reviendrons sur la signification de chaque ligne de ce code.*

Mais, apprenez qu'en JS, les fonctions sont des objets privilégiés.

```
1  "use strict";//standart + strict
2
3  function TaskAtHandApp()//une fonction sert de classe !
4  {
5      var version = "v1.0"; //variable privée
6
7      function setStatus(message) //méthode privée
8      {
9          $("#app>footer").text(message);
10     }
11
12     this.start = function()//méthode public à utiliser avec this
13     {
14         $("#app>header").append(version);
15         setStatus("ready");
16     };
17 }//ressemble à une classe
18
19 $(function()
20 {
21     window.app = new TaskAtHandApp();//instantiation de la classe
22     window.app.start();
23 })
```

Pour l'instant rettenons que :
Nous définition une classe[1] TaskAtHandApp (ligne 3) et nous créons une instance (ligne 21).

Nous connaissons $(document).ready(handler)) en écriture compact (ligne 19).

Vous avez du temps devant vous, découvrez window en tapant dans la console **(F12)** : `console.dir(window)`

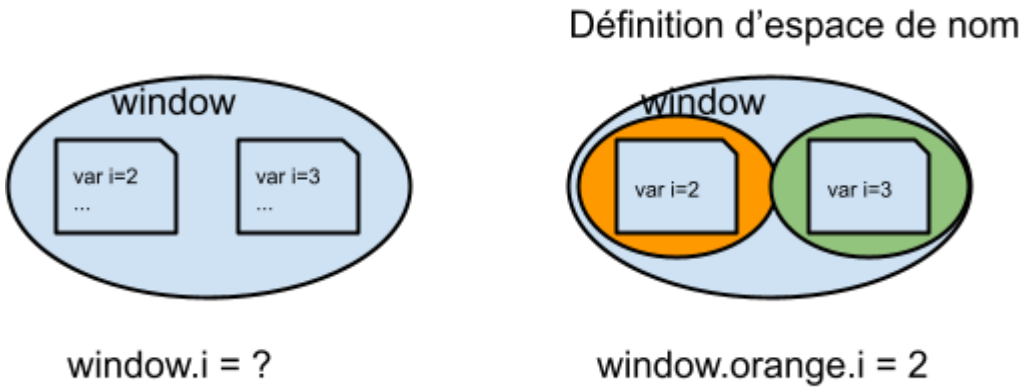
Mais que vient faire (ligne 21) window ?
Pour faire simple, lorsque nous travaillons en JS, nous le faisons dans un espace de noms : window.

Ainsi, lorsque nous déclarons une variable, elle se retrouve dans un window.

```
> var i = 2;
< undefined
> console.log(window.i);
  2
< undefined
>
```

[1] Nous pouvons utiliser une objet littéral pour la construction.

L'idée ici est de définir notre espace de nom en ecrivant window.app pour protèger notre code et ne pas surcharger l'espace window.
Nous pourrons travailler à plusieurs (ou plusieurs fichier js) sans se préoccuper des noms des fonctions.



Exemple d'isolation d'une variable.

```
Console was cleared
> window.ChezMoi = new Object();
Object {}
> window.ChezMoi.i = "topsecret";
"topsecret"
> console.log(i);
2
undefined
> console.log(ChezMoi.i);
topsecret
undefined
>
```

CSS

Ecrire le code css suivant

```
body
{
    font: 1em Verdana, Geneva, sans-serif;
    padding: 0;
    margin: 5px;
    color: Black;
    background-color: WhiteSmoke;
}
}
div
{
    padding: 0;
    margin: 0;
}
button
{
    cursor: pointer;
}
.hidden
{
    display: none;
}

```

Pour l'application nous écrivons

```
/*********************************
/* App */
/*********************************
 #app
{
    margin: 4px;
    background-color: #bbc;
}

#app>header
{
    padding: 0 0.5em;
    font-size: 1.5em;
    color: WhiteSmoke;
    background-color: #006;
}
#app>footer
{
    padding: 0.25em;
    color: WhiteSmoke;
    background-color: #006;
}

#main
{
    margin: 1em;
}
```

---

lecture

Introduction Objet
Design pattern