Multimédia - Partie WEB

Syllabus de 6ème PHOTO

Edition : 2015

Professeur : J. Pochet

Table des matières

Introduction

Le domaine du développement WEB est très vaste et fait appel à de nombreuses connaissances et compétences ; il n'est donc pas possible d'être complet au sein d'un seul ouvrage.

L'objectif de ce syllabus est de constituer un support permettant à un élève débutant de constituer un site basique de photographe au travers de quatre manipulations évolutives.

Chaque manipulation décrit un exercice qui sera réalisé sur une période donnée et dont les notions théoriques nécessaires sont fournies juste après l'énoncé.
Au fur et à mesure de l'avancement des manipulations, les notions théoriques et les conseils donnés précédemment doivent être pris en compte.

Il est indispensable de disposer d'un environnement de travail à la maison (ordinateur et logiciels nécessaires installés) et de gérer ses backups régulièrement.
Il est également conseillé d'être relativement régulier dans son travail pendant l'année afin de se maintenir à l'aise avec les différentes balises HTML et propriétés CSS.

Les différents langages qui seront étudiés sont l'HTML5, le CSS3 et le Javascript.
Ces langages contribueront ensemble au site WEB et ses fonctionnalités :

- L'HTML est destiné à encapsuler le contenu du site au travers de balises
- Le CSS est destiné à mettre en forme la présentation du site (son style)
- Le Javascript va permettre de dynamiser le site à l'aide de programmation

Le fait d'utiliser l'HTML5, le CSS3 et le Javascript est un choix qui a été fait.
Cette façon de faire nécessite des navigateurs qui acceptent le Javascript et qui soient relativement récents. Par ailleurs, le fait d'utiliser un langage comme le Javascript permet de développer un site WEB sans avoir besoin d'une connexion Inernet.

Il est intéressant de savoir que les outils de développement WEB sont en constante évolution et que les différents navigateurs tentent de prendre en charge les nouvelles fonctionnalités mais n'y arrive pas toujours systématiquement.
Il est par exemple bon de garder à l'esprit que certaines propriétés récentes en CSS3 risquent de ne pas être gérées par certains navigateurs alors qu'on peut dire que toutes les propriétés de CSS2.1 (moins récentes) sont prises en charge actuellement par tous les navigateurs.

L'objectif du cours est donc de tabler sur l'HTML5 et le CSS2.1 (bien gérés par les navigateurs actuels) en ajoutant néanmoins certaines fonctionnalités CSS3 permettant des rendus visuels et des effets plus sophistiqués ; il est alors prudent de tester le bon fonctionnement de ces propriétés sur différents navigateurs (Firefox, Internet explorer, Safari, Chrome, …).

Certains exemples d'utilisation Javascript seront donnés dans le but d'une réutilisation mais ne feront pas l'objet d'une étude approfondie. Dans la même idée, une sensibilisation sera faite sur des outils comme Tweenmax, JQuery, Joomla, Wordpress,…

Manipulation 1 : Page d'introduction

Objectif général

- Installer un environnement de travail à la maison et le prendre en main
- Mettre en pratique des balises HTML de base en réalisant une page web
- Mettre en œuvre le lien entre l'HTML et le CSS et comprendre le mécanisme

Durée de la manipulation

- Consignes et théorie : 2 h
- Travail en classe : 4 h
- Travail à la maison : 4 h environ

Notions associées

Pour pouvoir réaliser la manipulation, il faut avoir installé un environnement de travail à la maison. Ensuite le vocabulaire de base, les notions d'HTML de base et le principe du CSS doivent être connus.

Objectifs spécifiques

A. INSTALLER UN ENVIRONNEMENT DE TRAVAIL COMPRENANT :

- Un éditeur de texte dédié au développement WEB (Notepad ++, Dreamweaver, …)
- Un ou plusieurs navigateurs récents

B. RÉDIGER UNE PAGE HTML COMPRENANT AU MINIMUM :

- Un titre (métadonnée)
- Un lien vers une page CSS
- Un paragraphe.
- Une Balise *div* comprenant une image fullscreen en background.

C RÉDIGER UNE PAGE CSS COMPRENANT AU MINIMUM LES MISES EN FORME SUIVANTES :

- Une image en fullscreen
- Une contrôle de la taille de typo pour chacun des éléments
- La ou les couleur(s) de typo

Théorie partie 1

Vocabulaire

Fichier	:	Dispose d'une extension, correspond à des données formatées
Dossier	:	Est simplement un conteneur ; n'a pas d'extension
Dossier racine	:	C'est le conteneur du projet web (contient le fichier index.html)
Arborescence d'un dossier	:	C'est la structure interne d'un dossier
index.html	:	C'est le nom de la page que le navigateur ouvre par défaut
Navigateur	:	Est l'utilitaire pour visualiser des pages web Ex : Chrome, Firefox, Safari, Opera, Internet explorer,…
Espace ftp	:	Est un espace de stockage distant. On tentera de la manipuler comme un dossier distant.
URL	:	C'est une chaîne de caractères permettant de localiser une page ou un fichier

Recommandations

Dossier réservé	:	Il est indispensable de consacrer un dossier dédié à chaque exercice web ou site web. Tout ce qui concerne l'exercice doit alors se trouver au sein du dossier afin d'assurer une portabilité. En effet, lorsqu'on déplace tout le dossier, les éléments liés à l'exercice sont toujours présents (et les chemins relatifs sont conservés).
Organisation d'un dossier	:	Il faut prendre soin d'organiser correctement les dossier afin d'assurer une clarté et de faciliter la manipulation des éléments.
Nom des dossiers/fichiers	:	Tout en minuscule. Eviter les espaces, les accents,... Afficher également les extensions.
Outils web	:	Il est possible de créer un site web avec simplement un éditeur de texte et un navigateur.

Environnement de travail

Pour construire une page Web, le plus simple est d'installer un éditeur de texte spécialisé comme *notepad++* ainsi que quelques navigateurs récents.
Si vous utilisez un MAC, il existe des alternatives gratuites à *notepad++* telles que *Text Wrangler*, *TextMate*, *Komodo Edit*, *Vim, Brackets,*… Le mieux est d'en tester plusieurs et choisir celui qui vous convient le mieux.

Pour chaque projet on pourra alors éditer les fichiers HTML, CSS, … et visualiser le rendu au travers du (des) navigateur(s).

HTML – Notions de base

C'est un langage informatique qui permet de structurer du contenu à l'aide de balises sur des pages destinées à être lues dans un navigateur web.
Le langage html est souvent utilisé en association avec le Javascript et le CSS.

A. STRUCTURE GENERALE D'UNE PAGE HTML

La structure de base :

```
<!doctype html>
<html>

<head>
<title>Untitled Document</title>
</head>

<body>
</body>

</html>
```

LE DOCUMENT HTML contient deux sections importantes : HEAD et BODY.
Les deux sections HEAD et BODY du document se trouvent au sein d'une balise <html>

La partie <html>… </html> constitue le containeur principal de tous les éléments de la page.
Elle sert à indiquer aux navigateurs où commencent les contenus html de la page. Elle se termine toujours par la balise de fermeture </html>.

La partie HEAD
Cette partie contient toutes les métadonnées[1] de la page. Elle s'ouvre avec <head> et se ferme avec </head>.
La balise <title> présente dans l'exemple permet de donner un titre à la page. Quand vous ouvrirez votre page dans votre navigateur, ce titre s'affichera dans l'onglet.

La partie BODY
La balise <body > spécifie le début de la partie visible du document. Elle se referme par la balise de fermeture </body>.
C'est le body qui contient tous les éléments de la page tels que : photo, vidéo, texte, hyperliens, tables, menu de navigation,....

B. BALISES HTML DE BASE

Les balises sont des marqueurs que l'on va ajouter avant et après une donnée (un texte , une photo, …) pour spécifier au navigateur de quelle manière l'afficher et de quel type de donnée il s'agit.

Exemple : <h1 > ceci est un titre </h1>
<h1 > ➔ marque le début d'un titre. </h1> ➔ marque la fin d'un titre.

Tout ce qu'il y a entre ces deux balises héritera de toutes les propriétés de la catégorie titre « h1 ».

[1] Une **métadonnée** est une donnée servant à définir ou décrire une autre donnée.
Le rôle premier de ces métadonnées est de fournir aux moteurs de recherche des informations sur la page.

La balise **fermante** est identique à la balise ouvrante, à la différence qu'elle contient un "/" pour indiquer que c'est une balise de fermeture.
< … > ➔ marque le début d'une balise. </ … > ➔ marque la fin d'une balise.

LES BALISES TITRE : <hX>

Il y a six niveaux de balise titre qui s'étendent de 1 à 6 et qui sont classés en fonction de leur importance. La balise représentant les niveaux de titre est **<hX>** où X représente le niveau.

- <h1> </h1> : signifie « titre très important »
- <h4> </h4> : titre encore moins important
- <h6> </h6> : titre secondaire

<h1> signifie « début du titre». </h1> signifie « fin du titre».

**LA BALISE PARAGRAPHE: **

La plupart du temps, lorsqu'on écrit du texte dans une page web, on le fait à l'intérieur de paragraphes.
Le langage HTML propose justement la balise pour délimiter les paragraphes.

 signifie « début du paragraphe ». signifie « fin du paragraphe ».

LA BALISE DIV: <DIV>

La balise est une sorte de conteneur vierge (de type bloc) capable de contenir toute une série d'éléments très différents (photos, audio, vidéo, texte,...).

Cette balise sera mise en forme et définie dans le CSS.

Exemple d'écriture : je suis du texte au sein d'un bloc

LA BALISE IMAGE: <IMG>

La balise <img> permet d'insérer une image directement au sein du fichier HTML.
Cependant le mécanisme de la balise <img> ne repose pas sur une paire de balises (ouvrante et fermante) comme les balises vues précédemment ; la balise <img> est une balise appelée balise orpheline (ou balise auto-fermante) car sa version complète ne nécessite qu'un seul marqueur (une seule balise).

Pour insérer une image à l'aide d'une balise <img> on écrira par exemple :

```
<img src="images/exemple.jpg" width ="400px"  />
```

On reconnaît qu'il s'agit d'une balise orpheline car elle se termine par « /> ».
Une balise orpheline prend donc une forme < … /> où « … » illustrent le nom de la balise ainsi que ses propriétés.
Pour rappel les balises ouvrantes et fermantes étaient représentées respectivement par <…> et </…>.
Pour qu'une balise <img> puisse faire référence à une image, il faut que le chemin vers le fichier image soit précisé. On précisera le chemin vers l'image entre guillemets comme valeur de la propriété « src » de la balise <img> (« src » signifie la source). Dans notre exemple l'image est « exemple.jpg », se trouve au sein

d'un dossier nommé « images » et est accessible à partir du fichier HTML à l'aide du chemin "images/exemple.jpg".

Par défaut l'image sera affichée avec le même nombre de pixels que le fichier image précisé. Ainsi, pour éviter de voir apparaître une image trop grande ou trop petite sur l'écran, il est de coutume de préciser une dimension pour cette image. Dans notre exemple la dimension a été fixée en précisant une largeur (width) de 400 pixels (400px).

Il est possible de fixer la largeur (width) mais également la hauteur (height) mais il est conseillé de ne préciser que la largeur ou que la hauteur pour éviter une déformation de l'image ; on conserve ainsi les proportions.

Lorsqu'on souhaite insérer une image au sein d'un site web, il faut avoir pris soin de la compresser ! Suivant l'application de l'image on peut imaginer une compression dont la largeur vaut :

- Quelques centaines de pixels pour une image de taille normale
- 800 à 1200 pixels pour une grande image
- 1200 à 2000 pixels pour une image en fond d'écran.

On pourrait éventuellement imaginer des images avec une largeur de plus de 2000 pixels pour des grands écrans et/ou des écrans avec une très haute définition mais il vaut mieux éviter si ce n'est pas nécessaire. En effet, une image bien compressée prendra moins de place en mémoire mais surtout prendra moins de temps à se charger au sein du navigateur ; le rendu sera alors plus fluide !

Même si une image peut être insérée à l'aide de la balise <img>, il est fréquent d'utiliser une balise comme conteneur et d'y placer une image de fond à l'aide d'une propriété CSS.

LES BALISES TEXTE EN GRAS … ET TEXTE EN ITALIQUE <I> … </I>

Le fait de placer du texte au sein d'une balise (entre et) va avoir pour effet de mettre ce texte en gras. De la même manière, la balise <i> va permettre de transformer le texte en italique.

On évite cependant aujourd'hui d'abuser de ce genre de balises car leur but est d'uniquement agir sur la présentation de données. On verra qu'il existe le CSS qui est justement destiné à gérer l'aspect présentation d'un site WEB.

Il existe également la balise qui permet de mettre en forme du texte en appliquant traditionnellement un effet gras sur celui-ci, comme le fait la balise .
L'usage de la balise donne cependant sémantiquement du poids au texte qu'elle contient au niveau des moteurs de recherche (pour le référencement).

**LES SAUTS DE LIGNE :
**

En HTML, si vous appuyez sur la touche « Enter », cela ne crée pas une nouvelle ligne comme vous en avez l'habitude.

Pour créer un saut de ligne, il faut rajouter la balise < br/>.
Comme pour la balise <img>, Il s'agit d'une balise dite « orpheline » car elle agit seule ; elle n'a pas besoin de fonctionner par pair (comme des balises classiques … , <h1> …</h1>, etc.).
En HTML5, la balise
 a été remplacée par
 pour la rendre plus concise.

On peut mettre plusieurs balises
 de suite pour faire plusieurs sauts de ligne, mais on considère que c'est une mauvaise pratique qui rend le code délicat à maintenir. En effet, il est d'usage de maintenir les actions liées à la présentation au sein du CSS prévu à cet effet.

Pour décaler un texte avec plus de précision, on utilisera le CSS, le langage qui met en forme le texte et les objets de la page html.

C. REGLE DE BASE POUR L'UTILISATION DE BALISES

La plupart des balises fonctionnent par deux : une balise pour commencer et une balise pour terminer.
On peut voir ça comme des parenthèses : il y en a une qui ouvre, une qui ferme et l'ensemble constitue un tout.

Nous appellerons l'ensemble BALISE_OUVRANTE + CONTENU + BALISE_FERMANTE_ASSOCIEE un ensemble complet.

Exemples :	Bonjour	↔	Ensemble complet
	du texte	↔	Ensemble incomplet
	exemple </h1>	↔	Ensemble incomplet
	au revoir	↔	Ensemble complet

Une règle très importante à respecter est qu'il faut éviter d'avoir un ensemble incomplet à l'intérieur d'une paire de balises.
Exemple d'erreur : Un texte en caractères gras et en italique

On voit qu'il y a une balise incomplète au sein de la paire de balise ….
De même il y a une balise incomplète au sein de la paire de balise ….
Ce type d'erreur est à éviter absolument, il peut conduire à une mauvaise interprétation du formatage des données par le navigateur.

Exemple correct : Un texte en caractères gras et en italique

Ici on voit qu'il n'y a pas d'erreur, la balise … contient un ensemble complet.
Le texte apparaîtra alors en gras et en italique.

D. EXEMPLE ILLUSTRATIF DE BASE

Ici à droite vous avez un exemple très simple de page HTML qui respecte la structure de base vue précédemment et qui illustre l'utilisation de quelques balises.

On remarque que les balises sont indentées (décalées vers la droite) pour faire ressortir la structure de la page.

```
<! DOCTYPE html>
  <html>
    <head>
    <meta charset="UTF-8">
    <title>Mon site Web</title>
    </head>

    <body>
    <h1>Un titre</h1>
    <p>du texte, du texte du texte, du texte</p>
    <h2>Sous-titre</h2>
    </body>
  </html>
```

CSS – Notions de base

Les lettres CSS correspondent aux initiales de *« Cascading Style Sheets » : **feuilles de style en cascade.***
Le CSS est un langage informatique qui met en forme les différents objets présents sur une page HTML.

Les feuilles de styles servent, comme leur nom l'indique, à contenir les styles que vont prendre les différents éléments de ma page html. C'est un langage informatique qui sert à décrire la présentation des documents HTML. On en est aujourd'hui à la 3ème version du langage : le CSS3[1].

Le CSS permet de déterminer, pour chaque élément d'une page html :

- la taille
- la couleur
- la profondeur
- les ombres
- la position
- …
- plus important, il permet maintenant de réaliser de petites animations.

C'est dans la page CSS qu'on va attribuer un style aux paragraphes, aux titres, … ainsi qu'aux balises créées précédemment.

Quelle est la différence entre CSS et HTML ?

HTML sert à structurer le contenu, CSS sert à mettre en forme un contenu structuré.

A. LIER LE CSS AU FICHIER HTML

Pour que votre page CSS puisse dialoguer avec votre page html, nous devons lier ces deux pages ensemble.

Après avoir créé votre page html et votre page css (dans un même dossier par facilité), il faut les lier avec la balise link dans la partie <head> du document html.

Écriture **:** <link rel="stylesheet" href="style.css" />

Balise link : Permet de lier d'autres documents à la page html.

Rappel : la balise <link> permet d'établir une connexion entre la page html et un élément extérieur. Dans ce cas-ci, le lien redirige vers une feuille de style ou « css ». La feuille CSS servira à mettre en page le document.

L'attribut rel : Décrit le type de relation entre le document à lier et la page html.

L'attribut href : Indique l'adresse de destination et le nom du fichier.

Exemple : href="css/style.css" :

Quel est le nom de ma page ? style.css
Où se trouve cette page ? Dans le dossier « css »

[1] Même si le CSS4 est déjà en construction, le CSS3 tend à voir ses différentes parties de mieux en mieux gérées par les navigateurs. Le CSS2.1 (plus ancien) est aujourd'hui entièrement géré par les navigateurs actuels.

B. FONCTIONNEMENT

La syntaxe de base du CSS (pour mettre en forme l'HTML) est sous la forme :

selector { proprety : value ;}

La syntaxe repose sur trois éléments :

Selector (sélecteur)	**:**	Correspond à la balise html à mettre en forme.
Property (propriété)	**:**	Correspond à la propriété CSS ; ex : taille de typo.
Value (valeur)	**:**	Correspond à la valeur de la propriété ; ex : 16px .

Exemple :

```
h3      {
        color : #00FF00 ;
        font-size : 16px ;
        font-family: Arial;
        }
```

Dans l'exemple ci-dessus le sélecteur est « h3 », ce qui signifie que toutes les propriétés CSS définies ici vont influencer la présentation des éléments HTML balisés par <h3>…</h3>. Lorsqu'il s'agit d'une balise existante en HTML, l'expression du sélecteur peut simplement se réduire au nom de la balise.

Les accolades « { » et « } » délimitent les propriétés CSS associées au sélecteur.
L'exemple ci-dessous montre trois propriétés CSS définies pour le sélecteur h3. Chaque propriété doit être terminée par un point-virgule.

Les propriétés CSS existantes sont définies par le langage.
L'exemple montre trois propriétés : « color », « font-size » et « font-family ».
Chaque propriété se verra en général assigné d'une valeur.
L'assignation est représentée à l'aide des deux points.

C. LES PROPRIETES CLASS ET ID

Le point précédent a illustré une manière de définir des propriétés CSS à une balise existante.
Il est cependant possible, au sein de l'HTML, de préciser un « id » et/ou une « class » à une balise HTML. Le cas le plus courant est le cas de la balise .

En lui attribuant un sélecteur contextuel, la balise ***div*** peut hériter de particularités qui lui sont spécifiques.
Il existe deux types de sélecteurs[1] contextuels :

- id
- class

Les sélecteurs contextuels class et id sont quasiment identiques.
La seule différence entre les deux est qu'une valeur de l'attribut « id » ne s'utilise qu'une seule fois alors qu'une valeur de l'attribut « class » peut s'utiliser plusieurs fois.

[1] Pour être complet, un sélecteur ne se limite pas uniquement à la valeur du champ « id » ou « class » mais peut être composé de plusieurs éléments ; la valeur du champ « id » ou « class » contribue alors au sélecteur composé.
Par simplicité d'utilisation, nous limiterons l'usage de sélecteurs composés et nous pourrons dans ce cas assimiler la valeur du champ « id » ou « class » au sélecteur en soi.

Ceci implique que si l'on veut appliquer un même style à différents objets, on utilisera l'attribut « class ». Lorsqu'on veut attribuer un style à un seul objet, on utilisera un « id ».

ÉCRITURE *<balise id="nom_de_ID"> .... </balise>* exemple :

où	**balise**	**:**	est le type de balise
	id ="nom_de_ID"	**:**	le nom que l'on choisit pour nommer la balise

D. TROIS TYPES DE SELECTEURS SIMPLES

Il existe des sélecteurs simples (un seul terme) et des sélecteurs plus compliqués composés de plusieurs termes. Nous allons ici nous contenter de décrire trois sélecteurs simples :

Les balises existantes	**:**	Les balises déjà existantes telles que <p>, <h3>, … peuvent être utilisées telles quelles comme sélecteur.
Les balises possédant un id	**:**	Ces balises sont ciblées sur la feuille CSS en notant leur nom d'identifiant **précédé d'un "# "**
Les balises avec la propriété class	**:**	Les balises possédant un nom de classe sont ciblées sur la feuille CSS en notant leur nom de classe **précédé d'un "."**

E. QUELQUES PROPRIETES CSS DE BASE

Il existe une centaine de propriétés importantes à connaître pour mettre correctement une page HTML en forme.

• Quelques propriétés relatives au texte

font-family : Définit, sous forme de liste classée par ordre de préférence, la famille de polices à utiliser pour afficher le contenu.

Syntaxe CSS : font-family: fontName, fontName, fontName ;

Valeur(s) possible(s) pour cette propriété : nom de famille de police.

Exemple : body {font-family: "Century Schoolbook", Times, serif;}

font-size : Définit la taille de caractères de l'élément.

Syntaxe CSS : font-size: Taille en px | Taille en %

Valeur(s) possible(s) pour cette propriété : infinie.

Exemples : body {font-size: 14pt ;}
span.larger {font-size: 150% ;}

font-style : Détermine si l'élément doit être affiché en caractères normaux (Roman), italiques ou obliques.

Syntaxe CSS : font-style: constante ;

Valeur(s) possible(s) pour cette propriété : L'une des trois constantes suivantes : normal | italic | oblique

Exemple : h2 {font-style: italic ;}

color : Définit la couleur du texte.

Syntaxe CSS : color: valeur HEX ou RGB,

Valeur(s) possible(s) pour cette propriété : couleur en hexadécimal ou RGB.

Exemple : h3 {color: #000080 ;}

• Deux propriétés relatives aux tailles

width : détermine la longueur (ou largeur) d'un élément de texte ou d'une image

Syntaxe CSS : width: largeur.

Valeur(s) possible(s) pour cette propriété : chiffre en pixels (px) ou pourcentage (%)

Exemple : #bloc_perso {width: 200px;}

height : détermine la hauteur d'un élément de texte ou d'une image

Syntaxe CSS : height: hauteur.

Valeur(s) possible(s) pour cette propriété : chiffre en pixels (px) ou pourcentage (%)

Exemple : #bloc_perso {height: 200px ;}

• Les propriétés relatives aux fonds

background-color : Définit la couleur d'arrière-plan de l'élément.

Syntaxe CSS : background-color: couleur en hexadécimal ou RGB.

Valeur(s) possible(s) pour cette propriété : Toute spécification de couleur valide ou transparente.

Exemple : p {background-color: #000;}

background : Propriété permettant de définir, à l'aide d'une seule instruction, cinq propriétés de style d'arrière-plan : La couleur, le lien vers une image, la répétition et les positions droites et top.

Syntaxe CSS : background: color url('img.png') no-repeat right% top%;

Valeur(s) possible(s) pour cette propriété : Toute combinaison possible des valeurs de propriété.

Exemple : body {background: #FFF url('img_tree.png') no-repeat 50% 50% ;}

background-size : Propriété pour spécifier la dimension des images d'arrière-plan.

Syntaxe CSS : background-size: constante.

Valeur(s) possible(s) pour cette propriété :

- cover, qui spécifie que l'image d'arrière-plan doit être mise à l'échelle pour être aussi petite que possible tout en assurant que les deux dimensions soient supérieures ou égales aux dimensions de la surface de positionnement.
- contain, qui spécifie que l'image d'arrière-plan doit être mise à l'échelle pour être aussi grande que possible tout en assurant que les deux dimensions sont plus petites ou égales aux dimensions de la surface de positionnement.

Exemple : body {background-size: cover;}

• Quelques explications sur les images de fonds

Lorsqu'on place une image de fond (d'un bloc ou de la page) il peut être pratique que l'image s'adapte aux dimensions de son conteneur.

On peut alors compléter les propriétés du conteneur et placer le « cover » comme ci-dessous :

```
selecteur_du_conteneur{ …
background-size : cover;
…
}
```

Cette façon de faire peut être utile pour un bloc quelconque dans la page car de cette manière si on décide de redimensionner le bloc, l'image de fond va s'adapter automatiquement. Ce qui donne plus de souplesse à la création de la page.

Un autre exemple d'application est lorsqu'on veut mettre une image de fond à toute notre page et qu'on veut qu'elle se redimensionne automatiquement suivant la taille de la page.
Dans ce cas, il est une bonne idée de ne pas mettre directement une image de fond sur le body mais de créer un conteneur qui a la même taille que la page et qui contiendra l'image de fond.
Ainsi on ne contraint pas le body à une propriété trop générale et on se réserve la possibilité d'utiliser le body pour des applications plus générales (comme le fait d'encapsuler plusieurs pages, …).

Pour une image de fond de la page, on peut alors créer au sein de l'html un conteneur (même vide) nommé par exemple .

On peut alors mettre par exemple :

```
#page_accueil{
position : absolute;
top : 0px;
left : 0px;
width : 100%; height : 100%;
background: #FFF url('img_tree.png') no-repeat 50% 50% ;
z-index : -1 ;
background-size : cover;
}
```

Etant en position « absolute », l'endroit de la balise dans l'html n'a d'importance que pour la lisibilité du code (en supposant un positionnement « absolu pur »). Il faut cependant savoir qu'il existe la propriété « position : relative;" » qui permet de nuancer la notion de positionnement « absolu pur » (voir plus tard).

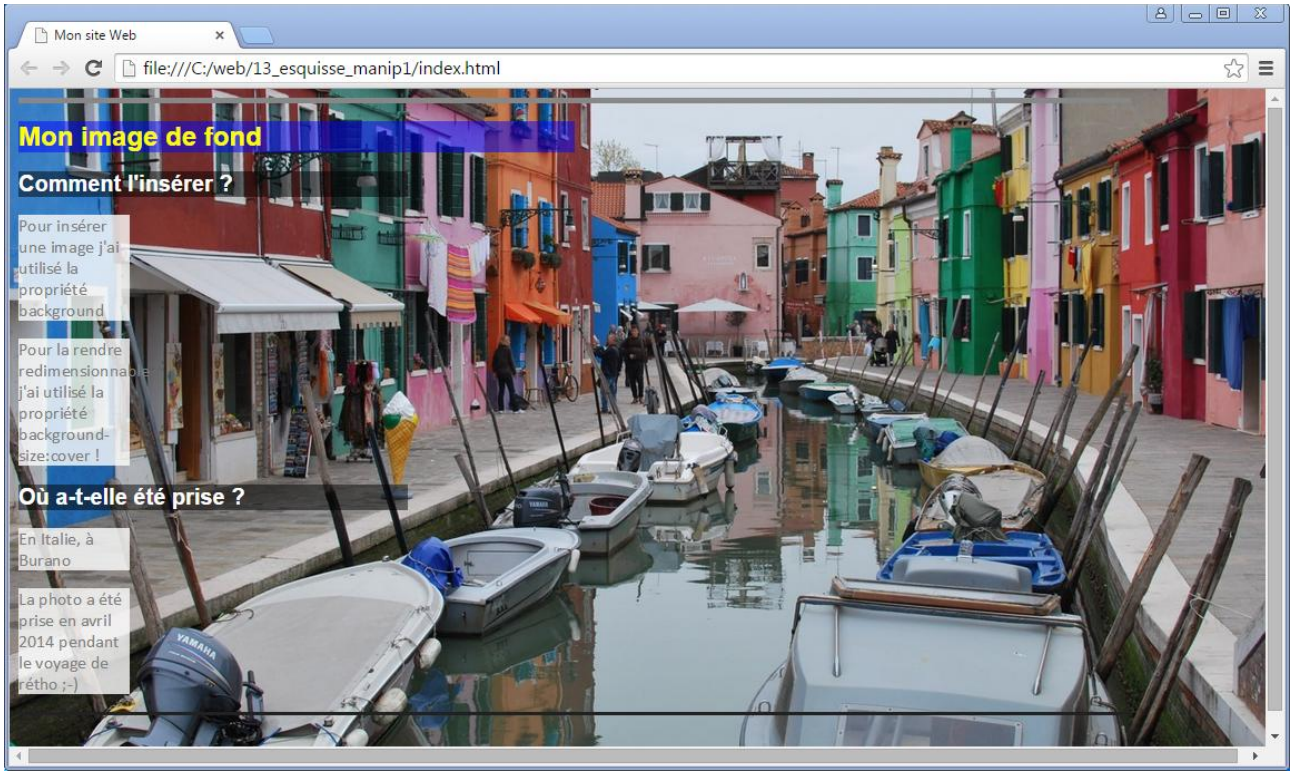
On peut voir que les propriétés « top » et « left » sont mises manuellement à « 0 px » même s'il s'agit du positionnement par défaut. Cette assignation est parfois nécessaire lorsqu'on utilise les transitions CSS.

Il peut être intéressant de compléter l'exemple ci-dessus en ajoutant au « body » et à la balise « html » des dimensions de 100% (« width : 100%; » et « height : 100%; »).

Il est courant que le fichier CSS contienne un ensemble de lignes d'initialisation pour éviter l'influence de certaines valeurs par défaut du navigateur.

Exemple de solution de la manipulation 1

L'exemple donné ici est volontairement trop simple. Vous trouverez l'aperçu de la page suivi de ses codes.



Code HTML :

index.html style.css

```
<!doctype html>
<html>
    <head>
        <meta charset="UTF-8">
        <title>Mon site Web</title>
        <link rel="stylesheet" href="style.css" />
    </head>

    <body>
    <div id="conteneur_fond"> </div>

    <div id="dessus"></div>
    <h1>Mon image de fond</h1>
    <h2>Comment l'insérer ?</h2>
        <p> Pour insérer une image j'ai utilisé la propriété background </p>
        <p> Pour la rendre redimensionnable j'ai utilisé la propriété background-size:cover !</p>
    <h2>Où a-t-elle été prise ?</h2>
        <p> En Italie, à Burano  </p>
        <p> La photo a été prise en avril 2014 pendant le voyage de rétho ;-)</p>

    <div id="dessous"></div>
    </body>
</html>
```

Code CSS :

```
html, body{
width:100%;
height:100%;
}

#dessus{
height:5px;
width:1000px;
background-color: #888;
}

#dessous{
height:3px;
width:1000px;
background-color: #222;
}

#conteneur_fond{
width:100%;
height:100%;
position:absolute;
top:0px;
left:0px;
z-index:-1;
background:#AAA url("h_coloree.jpg") no-repeat 50% 50%;
background-size:cover;
}
```

```
h1{
width:500px;
font-family:Arial;
font-size:24px;
background-color: rgba(0,0,255,0.5);
color:yellow;
}

h2{
width:350px;
font-family:Arial;
font-size:20px;
background-color: rgba(0,0,0,0.5);
color:white;
}

p{
font-family:Calibri;
width:100px;
font-size:16px;
background-color: rgba(255,255,255,0.8);
color:gray;
}
```

Manipulation 2 : Composition sur une seule page avec une vidéo

Objectif général

- Renforcer les notions vues précédemment
- Installer et prendre en main un logiciel client FTP
- Comprendre le positionnement naturel (flux normal) et le positionnement absolu
- Comprendre les propriétés « margin » et « padding »
- Insérer une vidéo personnelle en respectant les formats préconisés

Durée de la manipulation

- Consignes et théorie : 2 h
- Travail en classe : 8 h
- Travail à la maison : 6 h environ

Notions associées

Il faut avoir étendu son environnement de travail à la maison en installant un logiciel client FTP.
En plus des notions vues précédemment, il faut connaître le principe général du positionnement absolu (et ses paramètres « top », « bottom » ,…), le paramétrage des différentes profondeurs (avec z-index), l'utilisation des marges intérieurs (padding) et extérieures (margin) ainsi que la mise en œuvre de la balise HTML5 <video>.
Quelques propriétés CSS seront encore vues pour augmenter les possibilités de mise en forme.

Objectifs spécifiques

A. INSTALLER UN LOGICIEL CLIENT FTP :

Une fois votre logiciel client installé (au choix), vous pouvez mettre votre site en ligne dés que vous avez reçu vos accès FTP. Vous pourrez ainsi accéder à votre travail de n'importe quel ordinateur relié à Internet. D'autre part, une mise en ligne sur un serveur FTP vous assure automatiquement une sauvegarde.

B. RÉDIGER UNE PAGE HTML COMPRENANT AU MINIMUM :

- Les points HTML de la manipulation 1
- Deux balises DIV dans le flux normal avec un fond et du contenu
- Deux balises DIV positionné en absolu avec un fond et du contenu
- Une balise DIV contenant une vidéo personnelle et fonctionnelle

C RÉDIGER UNE PAGE CSS COMPRENANT AU MINIMUM LES MISES EN FORME SUIVANTES :

- Les points HTML de la manipulation 1
- Un paramétrage des dimensions des balises DIV
- La mise en œuvre de « margin » et de « padding » au sein de balises DIV
- Un positionnement absolu et esthétique de certaines balise DIV
- Une mise en forme du contenu des balises DIV

Théorie partie 2

La mise en ligne

Pour mettre un site en ligne il faut un hébergeur (un serveur FTP) qui pourra d'une part stocker tout votre site et d'autre part, dans certains cas, exécuter des lignes de codes pour rendre le site plus réactif.
Pour nous, afin de faciliter la réalisation du site, nous n'utiliserons le serveur que pour le stockage.
Bien entendu, pour mettre le site en ligne il faudra disposer d'une connexion Internet.

A. UTILISER SON SYSTEME D'EXPLOITATION

Certains systèmes d'exploitations permettent de manipuler directement un fichier distant (qui se trouve sur un serveur FTP). Par exemple sur Windows7 il est possible de taper le lien FTP complet au sein de la barre d'adresse d'un dossier. Une fois que l'on tape sur la touche « enter », le système nous demande le « login » et le mot de passe afin d'accéder au dossier distant.
Le contenu du dossier distant est alors disponible pour copier, enlever ou ajouter des éléments.

B. UTILISER UN GESTIONNAIRE FTP

Le plus courant est d'avoir recours à un logiciel qui va permettre d'établir la connexion entre notre machine (le client FTP) et notre hébergeur (le serveur FTP).

De nombreux logiciels sont disponibles sur Internet, vous trouverez par exemple :

- FileZilla
- Cyberduck
- Core FTP
- FTP expert
- …

Une fois votre logiciel installé, il est d'usage de créer votre site en spécifiant vos accès, votre lien FTP et votre dossier racine en local (sur votre ordinateur).
Il suffit alors de se connecter via votre logiciel pour faire les actions que vous souhaitez.

C. METTRE EN LIGNE VIA DREAMWEAVER

Comment mettre en ligne sur les ordinateurs de l'école ?

De manière à éviter l'installation d'un logiciel supplémentaire sur les ordinateurs de l'école, nous allons utiliser « Adobe Dreamweaver » pour prendre en charge la mise en ligne du site.
Les deux pages précédentes illustrent une manière de mettre en ligne via dreamweaver mais suivant la version de dreamweaver et/ou le type d'ordinateur que l'on a (windows/mac) la procédure diffère un peu.

Avec les ordinateurs de l'école la procédure est la suivante :

1°) On prépare le terrain

On enregistre tout le site au sein d'un dossier de l'ordinateur. Il doit y figurer un fichier nommé « index.html » qui correspond à la page par défaut.

Tous les liens internes au site doivent être des liens relatifs (à la racine du site).
Le dossier en question sera alors pour dreamweaver la racine du site, l'idée est d'établir une correspondance entre le dossier racine du site et l'espace FTP qui contiendra le site.

2°) On regarde ce qu'on a actuellement sur notre espace FTP

On peut commencer par voir ce qu'il y a actuellement sur le site, il suffit de taper l'URL au sein de la barre d'adresse de votre navigateur.
Par exemple : http://jhallyday.multimedia.inraci.be (exemple fictif)
Bien sûr il faut un accès internet pour cela.

3°) On gère le site dans dreamweaver

On ouvre dreamweaver. On va sur dans la barre de menu en haut et on sélectionne :
Site → nouveau site
On donne un nom de site (au choix). On va travailler au sein de l'onglet « avancé ».
A gauche de la fenêtre il y a plusieurs « rubriques », on va commencer avec « infos locales ».
On va spécifier à dreamweaver le chemin vers le dossier racine (créé en 1°).
On va cocher « Racine du site » pour les liens relatifs.
On introduit l'URL, par exemple http://jhallyday.multimedia.inraci.be (exemple fictif).
On choisit ensuite la « rubriques » « infos distantes » à gauche.
On spécifie un accès : « FTP »
On introduit l'hôte FTP, c'est l'adresse FTP sans le « ftp:// »
par exemple jhallyday.multimedia.inraci.be (exemple fictif).
Répertoire de l'hôte, on ne met rien.
On introduit le login et le mot de passe fournis par l'école. Le login est la 1ère lettre du prénom suivi du nom (tout attaché et minuscule). Exemple : jhallyday (exemple fictif).
Pour le mot de passe il fait respecter minuscules et majuscules.
On appuie alors sur « tester » pour vérifier si dreamweaver arrive à établir la connexion, si tout va bien il apparaît une fenêtre qui indique que la connexion a été établie avec succès.
S'il apparaît un problème, il faut vérifier les données introduites (hôte ftp, login et mot de passe). Si le problème subsiste vérifiez qu'Internet est disponible (sinon c'est peut-être le serveur FTP).
Une fois la connexion établie avec succès, on click sur OK.

Il apparaîtra alors une petite fenêtre (normalement à droite) qui permet de faire des actions en rapport avec le site.

4°) On met à jour le contenu de l'espace FTP

Dans la petite fenêtre qui est apparue à la fin du 3°) il y a la possibilité d'afficher le contenu local ou le contenu distant.
On choisit le contenu distant. Parfois il est indiqué qu'il faut appuyer sur un bouton pour pouvoir afficher le contenu distant.
On peut alors voir le contenu de ce qui se trouve actuellement sur le serveur FTP (le site précédent).

On appuie sur  pour synchroniser.

On choisit ensuite les options synchroniser tout le site et on coche pour supprimer les fichiers distants qui ne sont pas dans le dossier racine local.
On click sur « Aperçu … » puis on valide (avec OK) les actions de synchronisation.
Une fenêtre montre alors le nouveau contenu du serveur distant.

5°) On vérifie si le site est bien mis en ligne.

On retourne au sein de notre navigateur et on demande de rafraîchir la page avec l'URL déjà introduit. Les changements devraient avoir été faits.
Parfois il apparaît un petit problème lorsque certains fichiers sont les mêmes dans le dossier racine et au sein de l'espace FTP ; pour le fichier « index.html » par exemple, dreamweaver choisit de placer le plus récent au sein de l'espace FTP. Il apparaît alors un problème si le fichier « index.html » du dossier racine (local) est plus ancien que celui de l'espace FTP, en effet dans ce cas l'« index.html » local n'est pas transféré.
Dans un cas comme celui-ci, il faut alors prendre soin d'effacer les fichiers distants au préalable.

Notions de positionnement

Pour positionner des éléments d'une page Internet, il y a des outils HTML et des outils CSS.
En HTML par exemple, il est possible d'utiliser la balise … pour centrer un élément ou d'utiliser une succession de
 pour descendre des éléments.
Il est cependant recommandé d'éviter au maximum de mettre en forme une page à l'aide d'outils HTML mais d'utiliser plutôt le CSS prévu à cet effet.

QU'EST-CE QUE LE FLUX NORMAL ?

Par défaut, les éléments sont positionnés dans le même ordre que l'ordre spécifié dans la page HTML. On dit que le positionnement est régi par le flux normal (c'est-à-dire l'ordre normal d'apparition des éléments dans l'HTML).
Il est possible d'agir sur divers éléments de sorte à modifier son positionnement vis-à-vis du flux normal ; on peut utiliser par exemple la propriété « position » qui permet de préciser le type de positionnement souhaité pour l'élément. On pourra ensuite régler la position à l'aide des propriétés complémentaires « top », « right », « left » et « bottom ».

ELEMENTS DE TYPE « BLOC » OU DE TYPE « EN-LIGNE »

Les éléments de type « bloc » sont par défaut placés les uns au-dessus des autres lors du flux normal.
Un exemple d'élément de type « bloc » est la balise « h1 ».

En effet, si on écrit en HTML : <h1>bonjour </h1> <h1>tout le monde </h1>
Le « bonjour » apparaît sur une ligne et le « tout le monde » sur une autre car il s'agit de blocs et qu'un bloc occupe toute une ligne par défaut.

Un bloc « div » est prédéfini comme un bloc.
Par défaut la largeur du bloc est celle du conteneur. A moins d'avoir recours à une technique particulière, chaque bloc est suivi d'un passage à la ligne.

Les éléments de type «en-ligne » sont par défaut placés les uns à coté des autres lors du flux normal.
Un exemple d'élément de type « en-ligne», la balise « b ».

En effet, si on écrit en HTML : bonjour tout le monde
Le « **bonjour** » et le « **tout le monde** » apparaîtront sur une même ligne car il s'agit d'éléments de type « en-ligne ». Un passage automatique à la ligne suivante apparaîtra cependant lorsque les éléments dépassent la largeur du conteneur.

UNE UTILISATION SIMPLE D'UN POSITIONNEMENT

ABSOLU

Dans un premier temps, pour simplifier, nous dirons que le positionnement absolu permet de positionner un élément dans l'absolu par rapport à la page.

Il est cependant possible d'utiliser le positionnement absolu de manière plus nuancée.

Pour placer un élément dans l'absolu par rapport à la page, il faut écrire la propriété CSS :

position : absolute ;

Pour ensuite préciser à quel endroit on souhaite placer l'élément, on doit donner des informations complémentaires. On donne des valeurs aux propriétés suivantes :

Top	(c'est le haut)
Bottom	(c'est le bas)
Left	(c'est la gauche)
Right	(c'est la droite)

Leurs valeurs peuvent être définies en :

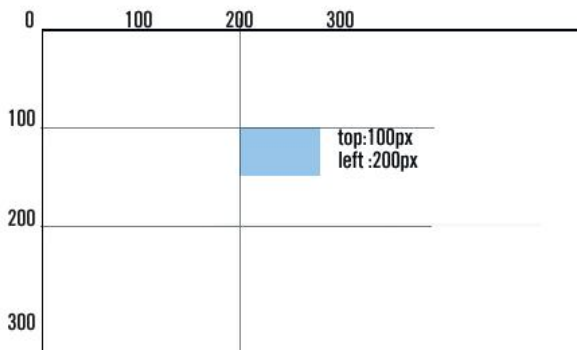
- pixels (vous précisez la valeur suivie de px),
- pourcentage (la même chose suivie de %).

La position absolue initiale se détermine par rapport au coin supérieur gauche de la fenêtre du navigateur. Les coordonnées de ce point sont : « top = 0 » et « left = 0 ».

Exemple :

```
#mon_bloc_bleu { …
    position  : absolute ;
    top : 100px ;
    left : 200px ;
    }
```

Illustration :



Dans le graphique ci-dessus j'ai positionné mon rectangle à 100px du top de la fenêtre et à 200px du côté gauche de la fenêtre.

Remarque : Pour avoir accès aux propriétés top, bottom, left et right, il faut absolument utiliser la propriété « position ».

Il est important de savoir que lorsqu'on travaille en positionnement absolu, on sort de ce qu'on peut appeler le flux normal (l'ordre donné par l'html) ; on travaille de la même manière qu'un calque.

Cette façon de faire permet de faciliter la composition de la page mais en contrepartie on sort de l'organisation naturelle donnée par l'html.

Il faut donc être extrêmement prudent et utiliser le positionnement absolu à bon escient. Il faut par exemple éviter de mettre trop d'éléments en absolu et d'avoir des risques de superpositions non souhaitées d'éléments,…

PROFONDEURS D'ELEMENTS

Dans la mesure où nous définissons un bloc en position absolue, il est possible de paramétrer la profondeur du bloc[1]. On utilisera alors la propriété « z-index » que l'on assignera d'un nombre entier.
Plus la valeur de z-index sera élevée et plus l'élément sera à l'avant (proche de l'observateur).

Il est par exemple utile d'initialiser z-index à une valeur faible (par exemple z-index = -1) pour les blocs destinés à contenir une image de fond.

On peut voir ci-dessous un exemple qui illustre l'utilisation de z-index avec des blocs positionnés en absolus :

HTML
Contenu du « body »

```
…

<div id="bloc_a"> </div>
<div id="bloc_b"> </div>
<div id="bloc_c"> </div>

…
```

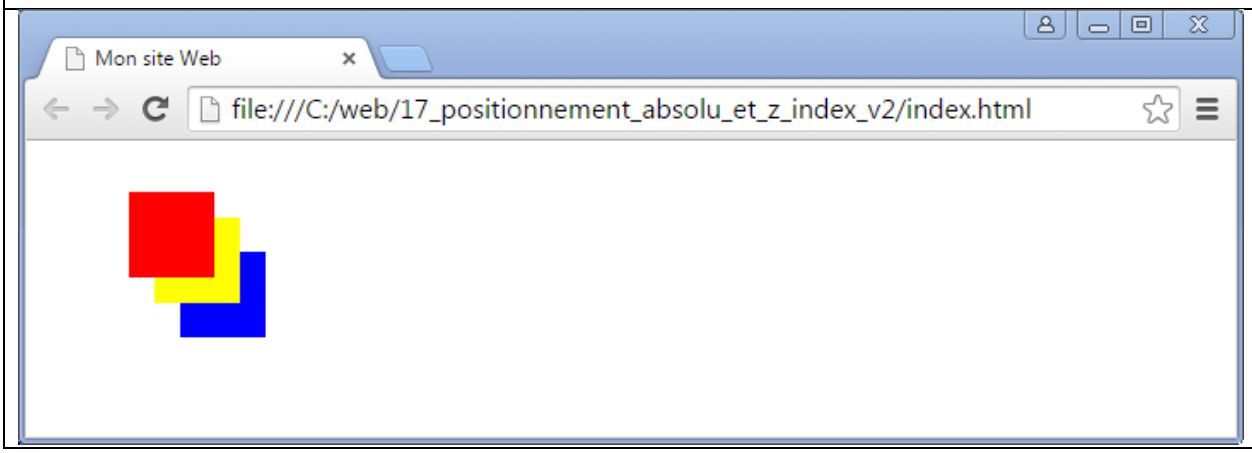
CSS
Propriétés des trois blocs

```
#bloc_a{
height:50px;
width:50px;
background-color: red;
position:absolute;
top: 30px;
left: 60px;
z-index:3;
}
```

```
#bloc_b{
height:50px;
width:50px;
background-color: yellow;
position:absolute;
top: 45px;
left: 75px;
z-index:2;
}
```

```
#bloc_c{
height:50px;
width:50px;
background-color: blue;
position:absolute;
top: 65px;
left: 90px;
z-index:1;
}
```

Aperçu au sein du navigateur



On peut remarque par exemple que le bloc rouge (dont la propriété « id » vaut « bloc_a ») est sur une couche plus proche de l'observateur que les deux autres blocs car sa valeur de « z-index » est plus élevée

L'utilisation des marges

Lorsqu'on crée une balise DIV (mais c'est vrai également pour d'autres éléments), on manipule une zone rectangulaire (un bloc). Cette zone peut ensuite contenir d'autres éléments comme du texte, des images,…
Cependant, il arrive que l'on souhaite que le contenu de notre zone rectangulaire ne soit pas collé aux bords de la zone. Dans ce cas on utilise des marges intérieures (padding).

D'autre part, lorsqu'il apparaît plusieurs blocs sur une page, on souhaite parfois maintenir de l'espace entre les différents blocs. Dans ce cas on utilise des marges extérieures (margin).

[1] C'est également vrai pour le positionnement relatif et le positionnement fixe.

MARGES EXTERIEURES : « MARGIN »

Lorsqu'on manipule des éléments de type blocs et que ceux-ci sont positionnés suivant le flux normal, il est possible d'assigner des marges extérieures à ceux-ci. Les marges peuvent alors être paramétrées séparément pour chaque bord d'un bloc à l'aide des propriétés CSS ci-dessous :

margin-top ↔ Détermine la marge supérieure (en unité de longueur)
Exemple : #bloc_normal { margin-top: 5px ; }

margin-right ↔ Détermine la marge droite (en unité de longueur)
Exemple : #bloc_normal { margin-right: 10px ; }

margin-bottom ↔ Détermine la marge inférieure (en unité de longueur)
Exemple : #bloc_normal { margin-bottom: 5px ; }

margin-left ↔ Détermine la marge gauche (en unité de longueur)
Exemple : #bloc_normal { margin-left: 10px ; }

Il est possible de paramétrer les marges extérieures des bords de façon plus concise en utilisant la propriété générale « margin ».

margin ↔ Regroupe les différentes propriétés de la marge

Si la propriété **margin** est suivie d'une seule valeur alors cette valeur est appliquée à toutes les marges extérieures (top, bottom, left, right).

Si la propriété **margin** est suivie de deux valeurs alors la première valeur est appliquée aux marges extérieures verticales (top/bottom) et la seconde aux marges extérieures horizontales (left/right).

Si la propriété **margin** est suivie de quatre valeurs elles correspondent dans l'ordre aux marges extérieures :
margin-top, **margin-right**, **margin-bottom**, **margin-left**.

Exemples :
```
#bloc_normal1 { margin: 5px 10px; }
#bloc_normal2 { margin: 8px; }
#bloc_normal3 { margin: 5px 8px 5px 2px; }
```

Il existe également la propriété très utile « margin :auto ; » qui ajustera automatiquement les marges gauches et droite d'un bloc (par rapport à son conteneur) lorsque la largeur du bloc est fixée.

FUSION DES MARGES

« S'il apparaît au sein du flux normal deux blocs successifs, ils seront disposés l'un au dessus de l'autre. Bien entendu si le bloc du dessus (le premier au sein de l'HTML) fait apparaître une marge inférieure (« margin-bottom ») il en résultera un décalage du bloc du dessous.
Imaginons maintenant que le bloc du dessous possède aussi une marge supérieure (« margin-top »). Qu'advient-il alors du décalage entre les deux blocs ? »

La réponse à cette question est liée à la notion de fusion des marges ; le décalage entre les deux blocs sera d'une valeur telle que la marge de chaque bloc soit au minimum respectée. Autrement dit, le décalage correspondra à la plus grande valeur entre le *« margin-bottom » du bloc supérieur et le « margin-top » du bloc inférieur.*

Un mécanisme de fusion des marges existe également lorsqu'un bloc contient un autre bloc.

On peut voir ci-dessous un exemple didactique qui illustre l'utilisation des marges extérieures avec des blocs positionnés suivant le flux normal :

HTML
Contenu de l'exemple

```
<!doctype html>
<html>
    <head>
        <meta charset="UTF-8">
        <title>Mon site Web</title>
        <link rel="stylesheet" href="style.css" />
    </head>

    <body>
    avant bloc1
    <div id="bloc1_de_body"> dans bloc1 </div>
    <div id="bloc2_de_body"> dans bloc2 </div>
    entre bloc2 et bloc3
    <div id="bloc3_de_body">
        texte debut bloc3
        <div id="bloc_a_de_bloc3"> dans bloc a </div>
        <div id="bloc_b_de_bloc3"> dans bloc b </div>
        texte fin bloc3
    </div>
    <div id="bloc4_de_body"> dans bloc4 </div>
    apres bloc4

    </body>
```

CSS
Propriétés des éléments

```
body,html{
height:100%;
width:100%;
background-color:brown;
padding:0px;
margin:0px;
}

#bloc1_de_body{
height:20px;
width:100px;
background-color: red;
margin-left:20px;
margin-bottom:20px;
}
#bloc2_de_body{
height:40px;
width:200px;
background-color: yellow;
margin-top:8px;
margin-bottom:8px;
}
#bloc3_de_body{
width:600px;
background-color: blue;
margin-top:20px;
}
#bloc_a_de_bloc3{
height:30px;
background-color:gray;
margin-left:30px;
}

#bloc_b_de_bloc3{
background-color:green;
width:350px;
margin:8px;
}

#bloc4_de_body{
height:50px;
width:150px;
background-color: red;
margin:25px 0px 10px 60px;
}
```

Aperçu au sein du navigateur

MARGES INTERIEURES : « PADDING »

Contrairement aux marges extérieures, les marges intérieures ne s'appliquent pas uniquement aux blocs positionnés dans le flux normal.
Comme l'idée est de créer de l'espace à l'intérieur du bloc (entre ses bords et son contenu), la propriété « padding » n'est pas en rapport avec la notion d'interaction de blocs. Pour cette raison la propriété « padding » convient pour des blocs positionnés suivant le flux normal (interaction naturelle) mais aussi pour les blocs positionnés en absolu.

Les marges intérieures peuvent être paramétrées séparément pour chaque bord :

padding-top ↔ Détermine la marge intérieure supérieure (unité de longueur)
Exemple : #bloc_normal { padding-top: 5px ; }

padding-right ↔ Détermine la marge intérieure droite (unité de longueur)
Exemple : #bloc_normal { padding-right: 10px ; }

padding-bottom ↔ Détermine la marge intérieure inférieure (unité de longueur)
Exemple : #bloc_normal { padding-bottom: 5px ; }

padding-left ↔ Détermine la marge intérieure gauche (unité de longueur)
Exemple : #bloc_normal { padding-left: 10px ; }

Tout comme pour le « margin », il est possible de paramétrer les marges intérieures des bords de façon plus concise en utilisant la propriété générale « padding ».

padding ↔ Regroupe les différentes propriétés de la marge intérieure

Si la propriété **padding** est suivie d'une seule valeur alors cette valeur est appliquée à toutes les marges intérieures (top, bottom, left, right).

Si la propriété **padding** est suivie de deux valeurs alors la première valeur est appliquée aux marges intérieures verticales (top/bottom) et la seconde aux marges intérieures horizontales (left/right).

Si la propriété **padding** est suivie de quatre valeurs elles correspondent dans l'ordre aux marges intérieures :
padding-top, **padding-right**, **padding-bottom**, **padding-left**.

Exemples :
```
#bloc_normal1 { padding: 5px 10px; }
#bloc_normal2 { padding: 8px; }
#bloc_normal3 { padding: 5px 8px 5px 2px; }
```

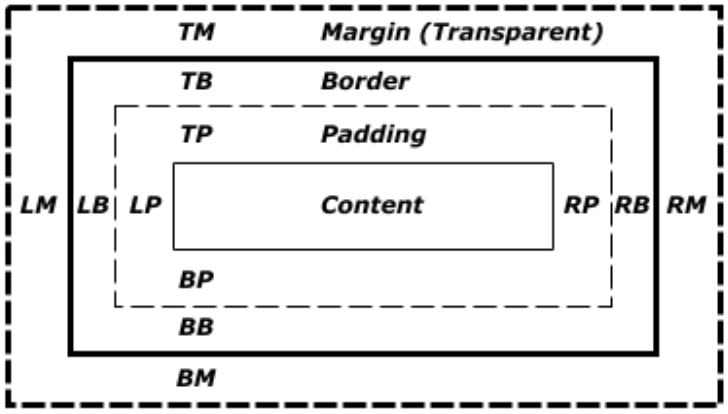
MODELE DE LA BOITE ET INFLUENCE DU « PADDING » SUR LES DIMENSIONS

Il est utile de savoir que l'ajout d'un « padding » à un bloc modifie généralement sa dimension.
Il est courant de fixer les dimensions d'un bloc grâce aux propriétés « width » et « height » et de décider plus tard des valeurs à donner aux marges intérieures (padding) ; petit inconvénient : les dimensions du bloc changent.

Pour comprendre ce phénomène, il faut se référer au modèle de la boîte qui illustre les éléments qui interviennent dans le calcul des dimensions d'une boîte (ou d'un bloc).

Les dimensions sont calculées sur le contenu mais aussi sur les marges intérieures est les bordures.

Modèle de la boîte (Box Model)



Modèle fourni par www.w3.org

Le modèle ci-dessus montre que pour chaque bord du contenu s'ajoutent :

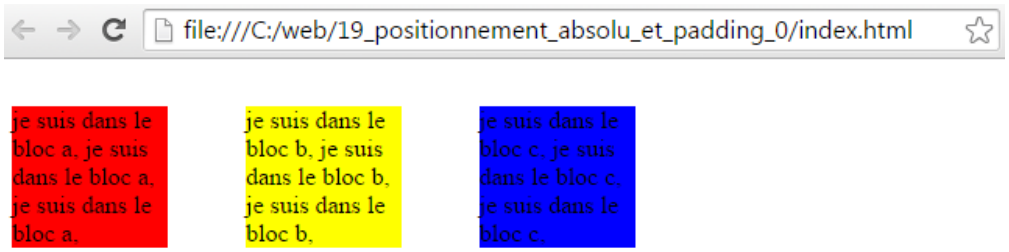
- Une marge intérieure (padding)
- Une bordure (border)
- Une marge extérieure

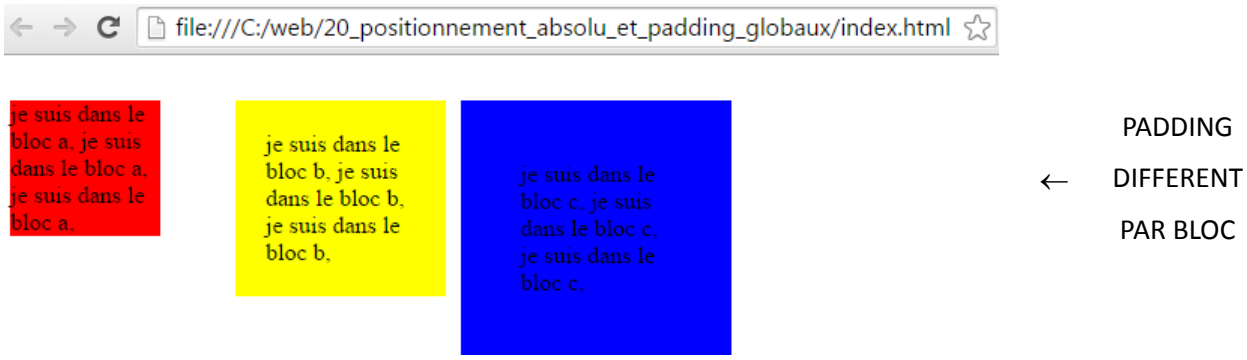
Il faut savoir que lorsqu'on fixe les propriétés « width » et « height » on fixe les dimensions du contenu.
Pour connaître les dimensions réelles du bloc, il faut encore ajouter les « padding » et les « border ».
On considère habituellement que les « margin » ne participent pas à la dimension réelle du bloc car se trouvent à l'extérieur du bloc.

Il existe une propriété CSS3 qui permet d'influencer les propriétés « width » et « height » de telle sorte qu'elles fixeront les dimensions réelles du bloc et non uniquement le contenu.
Cette propriété dont la syntaxe est « box-sizing :border-box ; » risque néanmoins de poser problème si nous avons affaire à un navigateur qui ne la reconnaît pas.

Illustration de l'influence du « padding » sur les dimensions des blocs :

Ci-dessous, tous les blocs ont été assignés des mêmes dimensions « width » et « height ».
Dans le premier aperçu le « padding » des blocs a été fixé à 0px alors que dans le deuxième, chaque bloc a été assigné d'une valeur différente de « padding ».





On doit alors garder à l'esprit que le « padding » influencera des dimensions réelles du bloc !

On peut par exemple décider d'ajuster les propriétés « height » et « width » pour que les dimensions totales du bloc correspondent à ce que l'on souhaite.

Une autre alternative est d'imaginer une structure de deux blocs imbriqués qui auront des rôles complémentaires. On peut par exemple imaginer un bloc nommé « outer » qui contiendra un autre bloc nommé « inner » ; « inner » contient quant à lui le contenu prévu.
L'idée est alors de fixer les dimensions uniquement pour le bloc « outer » et de fixer le « padding » uniquement pour le bloc « inner ».

L'insertion d'une vidéo

La lecture d'une vidéo par le navigateur nécessite l'utilisation d'un outil intégré au navigateur.
Il y a plusieurs façons d'insérer des vidéos mais la plus simple (qui est celle préconisée) est probablement d'utiliser la balise HTML5 <video> et de préciser plusieurs formats vidéos disponibles.

Il est alors parfois prudent d'intégrer, au sein du site, plusieurs fois la même vidéo mais sous des formats différents. La combinaison des formats OGG (extension .ogv ou .ogg) et MP4 devrait suffire.

Exemple complet :

```
<video controls id="ma_video">
        <source src="css3_validator.mp4" type="video/mp4" />
        <source src=" css3_validator.ogg" type="video/ogg" />
</video>
```

On peut remarquer dans l'exemple ci-dessus qu'en plus du fichier MP4, on place un fichier supplémentaire OGG. Ceci n'est pas obligatoire mais donne une sécurité supplémentaire pour les navigateurs moins récents.
On peut également remarquer qu'au sein de la balise vidéo nous avons un ID qui permettra de fixer certaines propriétés à notre vidéo (la taille au minimum).

Quelques propriétés CSS supplémentaires

text-transform :Contrôle la capitalisation du texte d'un élément.

Syntaxe CSS : text-transform: paramètre de capitalisation

Valeurs possibles pour cette propriété : none | capitalize | uppercase | lowercase | initial | inherit

Exemple : h2 { text-transform: uppercase ;}

text-align : Détermine l'alignement horizontal du texte à l'intérieur d'un élément.

Syntaxe CSS : text-align: alignement

Valeurs possibles pour cette propriété : center | justify | left | right.

Exemple : p {text-align: right;}

font-weight : Définit la graisse (épaisseur) de la police de caractères à appliquer à l'élément .

Syntaxe CSS : font-weight: constante

Valeurs possibles pour cette propriété : bold | normal | 100 | 200 | 300 | 400 | 500 | 600 | 700 | 800 | 900.

Exemple : p {font-weight: bold;}

letter-spacing : Définit l'espacement entre les caractères à l'intérieur d'un élément.

Syntaxe CSS : letter-spacing: longueur | normal

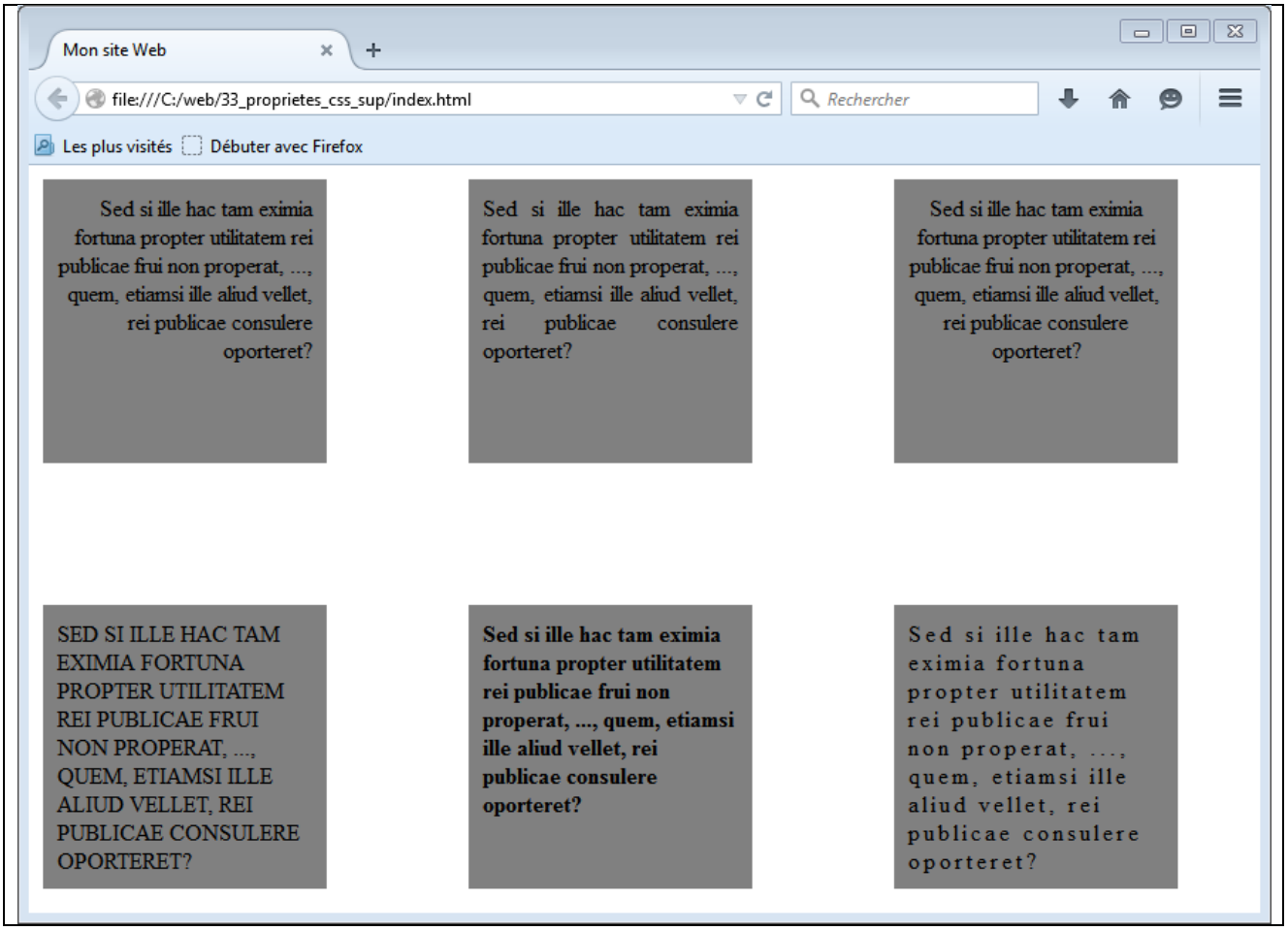
Valeurs possibles pour cette propriété : espace en px, em ou ex.

Exemple : a {letter-spacing: 1.1em;}

Illustration par un exemple :

On peut voir ci-dessous un exemple didactique qui illustre l'utilisation ces propriétés CSS supplémentaires. Vous trouverez l'aperçu au sein du navigateur suivi des codes HTML et CSS.

Aperçu au sein du navigateur


Mon site Web
file:///C:/web/33_proprietes_css_sup/index.html
Rechercher
Les plus visités
Débuter avec Firefox
Sed si ille hac tam eximia fortuna propter utilitatem rei publicae frui non properat, ..., quem, etiamsi ille aliud vellet, rei publicae consulere oporteret?
Sed si ille hac tam eximia fortuna propter utilitatem rei publicae frui non properat, ..., quem, etiamsi ille aliud vellet, rei publicae consulere oporteret?
Sed si ille hac tam eximia fortuna propter utilitatem rei publicae frui non properat, ..., quem, etiamsi ille aliud vellet, rei publicae consulere oporteret?
SED SI ILLE HAC TAM EXIMIA FORTUNA PROPTER UTILITATEM REI PUBLICAE FRUI NON PROPERAT, ..., QUEM, ETIAMSI ILLE ALIUD VELLET, REI PUBLICAE CONSULERE OPORTERET?
Sed si ille hac tam eximia fortuna propter utilitatem rei publicae frui non properat, ..., quem, etiamsi ille aliud vellet, rei publicae consulere oporteret?
Sed si ille hac tam eximia fortuna propter utilitatem rei publicae frui non properat, ..., quem, etiamsi ille aliud vellet, rei publicae consulere oporteret?

HTML
Contenu de l'exemple

```
<!doctype html>
<html>
    <head>
        <meta charset="UTF-8">
        <title>Mon site Web</title>
        <link rel="stylesheet" href="style.css" />
    </head>

    <body>
    <div id="bloc1" class="carre_200">
    Sed si ille hac tam eximia fortuna propter util
    </div>
    <div id="bloc2" class="carre_200">
    Sed si ille hac tam eximia fortuna propter util
    </div>
    <div id="bloc3" class="carre_200">
    Sed si ille hac tam eximia fortuna propter util
    </div>
    <div id="bloc4" class="carre_200">
    Sed si ille hac tam eximia fortuna propter util
    </div>
    <div id="bloc5" class="carre_200">
    Sed si ille hac tam eximia fortuna propter util
    </div>
    <div id="bloc6" class="carre_200">
    Sed si ille hac tam eximia fortuna propter util
    </div>
    </body>
</html>
```

Quelques commentaires

Le corps (body) de l'HTML contient simplement 6 blocs successifs.

Chaque bloc est créé à l'aide d'une balise DIV et contient du faux-texte.

Chaque bloc contient un ID qui lui est propre pour pouvoir mettre en évidence des propriétés particulières.

Tous les blocs utilisent une même classe ; ce qui permettra au sein du CSS d'y définir les propriétés communes à tous ces blocs.
La classe nommée « carre_200 » définira les propriétés évoquées par son nom (carré de 200 pixels).

CSS
Contenu de l'exemple

```
.carre_200{
padding:10px;
width:180px;
height:180px;
}

#bloc1, #bloc2, #bloc3, #bloc4, #bloc5, #bloc6{
background-color:gray;
position:absolute;
}
#bloc1{
top:10px;
left:10px;
text-align:right;
}

#bloc2{
top:10px;
left:310px;
text-align:justify;
}

#bloc3{
top:10px;
left:610px;
text-align:center;
}

#bloc4{
top:310px;
left:10px;
text-transform:uppercase;
}
#bloc5{
top:310px;
left:310px;
font-weight:600;
}

#bloc6{
top:310px;
left:610px;
letter-spacing:3px;
}
```

Quelques commentaires

Certaines propriétés communes sont définies au sein de la classe.
Comme le padding de 10px s'applique à tous les bords, le carré mesure 200px.

D'autres propriétés communes non liées à l'appelation « carre_200 » sont définies ici.

Chaque bloc est positionné dans l'absolu en respectant l'ordre des identifiants :

bloc1 bloc2 bloc3

bloc4 bloc5 bloc6

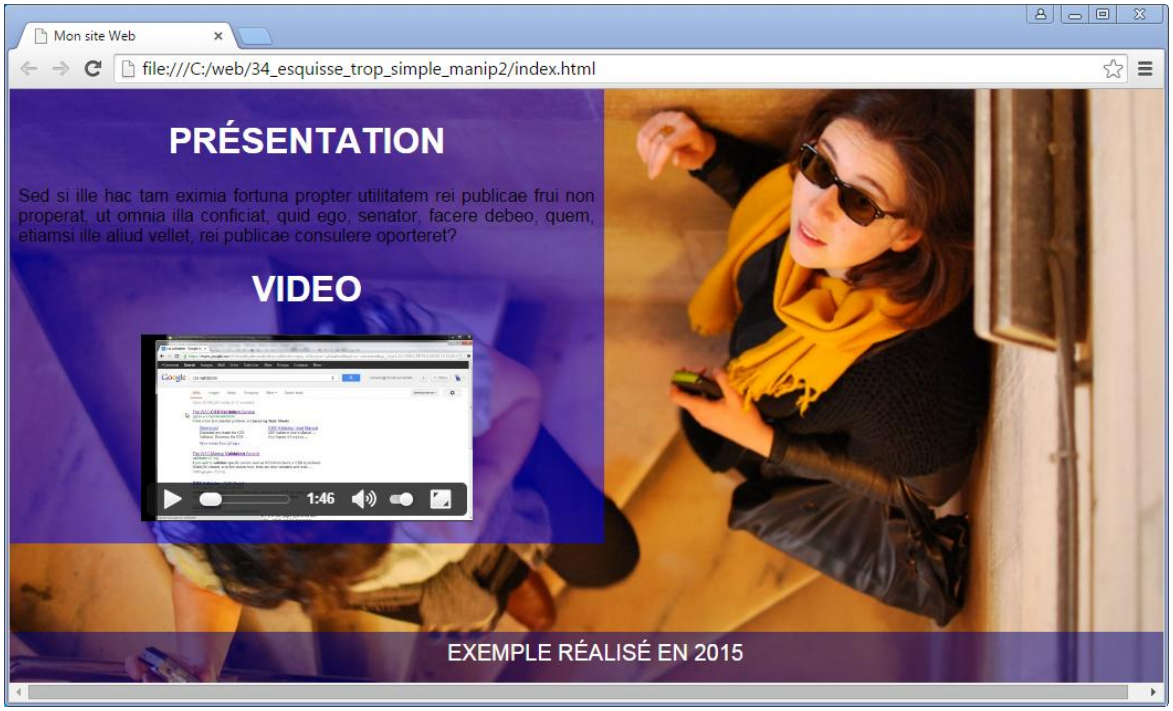
On voir des exemples de propriétés :

« text-align »
« text-transform »
« font-weight »
« letter-spacing »

Exemple de solution de la manipulation 2

L'exemple donné ici est volontairement trop simple. Vous trouverez l'aperçu de la page suivi de ses codes.

Aperçu :



Code HTML :

```
<!doctype html>
<html>
    <head>
        <meta charset="UTF-8">
        <title>Mon site Web</title>
        <link rel="stylesheet" href="style.css" ,
    </head>

    <body>
    <div id="conteneur_fond"> </div>

    <div id="contenu">
        <div id="bloc_presentation">
            <h1>Présentation</h1>
                <p>Sed si ille hac tam eximia fo
                rei publicae frui non properat, u
                quid ego, senator, facere debeo,
                rei publicae consulere oporteret
        </div>

        <div id="bloc_video">
            <h1>video</h1>
            <video controls id="ma_video">
                <source src="css3_validator.mp4"
                <source src="css3_validator.ogg"
            </video>
        </div>
    </div>

    <div id="pied"> Exemple réalisé en 2015</div
    </body>
</html>
```

Code CSS :

```css
html, body{
width:100%;
height:100%;
padding:0px;
margin:0px;
font-family:Arial;
}
#conteneur_fond{
width:100%;
height:100%;
position:absolute;
top:0px;
left:0px;
z-index:-1;
background:#AAA url("h_escaliers.jpg") no-repeat 50% 50%;
background-size:cover;
}
#ma_video{
display:block;
width:300px;
margin:auto;
}
#contenu{
width:50%;
background-color:rgba(0,0,255,0.5);
padding:8px 8px 20px 8px;
}
#bloc_presentation{
margin-bottom:20px;
}
h1{
color:white;
text-transform:uppercase;
text-align:center;
}
p{
text-align:justify;
}
#pied{
position:absolute;
bottom:0px;
width:100%;
height:30px;
background-color:rgba(0,0,128,0.5);
padding:8px;
text-align:center;
font-size:20px;
color:white;
text-transform:uppercase;
}
```

Manipulation 3 : Site complet

Objectif général

- Renforcer les notions vues précédemment
- Réaliser un site personnalisé avec minimum trois pages : Accueil, Galerie et Vidéo
- Soigner la fonctionnalité et l'esthétique du site
- Mettre en ligne le site finalisé
- Rédiger un rapport explicatif

Durée de la manipulation

- Consignes et théorie : 2 h
- Travail en classe : 8 h
- Travail à la maison : 10 h environ

Notions associées

En plus des notions vues précédemment, il faut connaître de manière plus approfondie les notions de positionnement (relatif, fixe et « absolu » par rapport à un conteneur) et pouvoir gérer une navigation.
Il faut pouvoir également inclure une galerie photo et rendre son site esthétique et convivial.
Pour ajouter de l'interaction, on utilisera des propriétés telles que « :hover », « transition », …
Pour améliorer l'esthétique, on verra quelques propriétés supplémentaires (bordures, ombres,…).

Objectifs spécifiques

A. GERER LA NAVIGATION ET CONSTRUIRE UNE GALERIE WEB :

Deux philosophies sont acceptées pour la navigation :

- Faire correspondre un fichier HTML par page
- Mettre plusieurs pages au sein d'un seul fichier HTML à l'aide de conteneurs

La construction d'une galerie Web reposera sur l'utilisation de Javascript.
Différents exemples sont fournis sur lesquels vous pouvez vous baser. Si vous le préférez, vous pouvez utiliser un autre outil de construction de galerie Web.
Dans tous les cas vous devez connaître les mécanismes principaux qui rendent la galerie fonctionnelle.

B. ASSURER UN SITE ESTHETIQUE ET FONCTIONNEL EN LIGNE :

Testez votre site sur différents navigateurs et arrangez-vous pour qu'il soit fonctionnel, fluide, bien structuré et esthétique. Prévoyez également que votre site convienne pour différentes tailles d'écrans.

C. REDIGER UN RAPPORT EXPLICATIF :

Vous devez rédiger un rapport consistant expliquant votre site pour trois parties différentes :

- Les éléments principaux de mise en page et de positionnement
- Les éléments esthétiques primordiaux
- L'aspect dynamique du site (navigation, galerie, transitions,…)

Si les explications sont consistantes et concises, chaque partie peut contenir environ une page.

Théorie partie 3

Notions plus avancées de positionnement

Lorsqu'on a abordé la notion de positionnement absolu, nous avons considéré l'exemple simple de positionnement absolu « pur ». On considérait alors que le bloc positionné était placé dans l'absolu par rapport à la fenêtre du navigateur.

En réalité lorsqu'on positionne un élément en absolu, il n'est pas forcément placé par rapport à la fenêtre du navigateur ; il se peut qu'il soit positionné par rapport à son conteneur direct ou à un conteneur plus lointain. Autrement dit, le positionnement absolu d'un bloc n'est pas forcément un positionnement absolu « pur » (par rapport à la fenêtre du navigateur) mais dans certains cas un positionnement absolu « relatif » à un conteneur.

Pour que le positionnement absolu d'un bloc soit « relatif » à un conteneur, il faut que ce conteneur ait été positionné respectivement à un des paramètres suivant : « absolute », « relative » ou « fixed ».
Le conteneur qui servira de référence au positionnement absolu du bloc est le conteneur le plus proche[1] dont le paramètre de positionnement est parmi « absolute », « relative » ou « fixed ».

On peut voir ci-dessous l'influence qu'apportera l'ajout de la ligne CSS « position : relative ; » à #bloc1.

Illustration d'un positionnement absolu « pur » et d'un positionnement absolu « relatif »

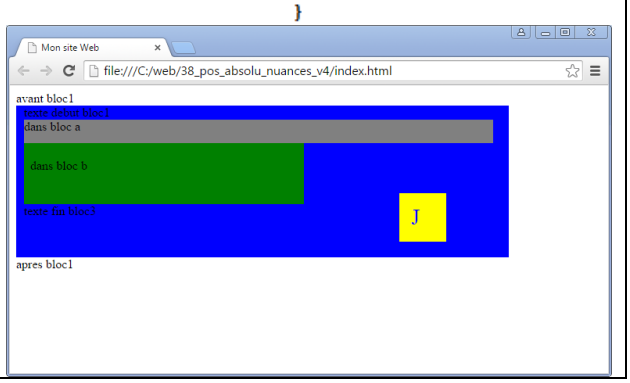
Le bloc jaune est positionné par rapport à la fenêtre

```
avant bloc1
<div id="bloc1">

    <div id="carre_jaune">J</div>

    texte debut bloc1
    <div id="bloc_a"> dans bloc a </div>
    <div id="bloc_b"> dans bloc b </div>
    texte fin bloc3
</div>

apres bloc1
```

```
#bloc1{
width:600px;
background-color: blue;
padding-left:10px;
padding-right:20px;
padding-bottom:50px;
}
```

Le bloc jaune est positionné par rapport au « bloc1 »

```
#bloc1{
position:relative;
width:600px;
background-color: blue;
padding-left:10px;
padding-right:20px;
padding-bottom:50px;
}
```



[1] Ici la notion de distance d'un conteneur fait référence à se profondeur d'imbrication. Un conteneur direct (aussi appelé « parent ») est considéré comme étant proche alors qu'un conteneur indirect (qui contient directement ou indirectement un parent de l'élément) sera considéré comme plus lointain.

Exemple illustrant une utilisation nuancée de positionnements absolus :

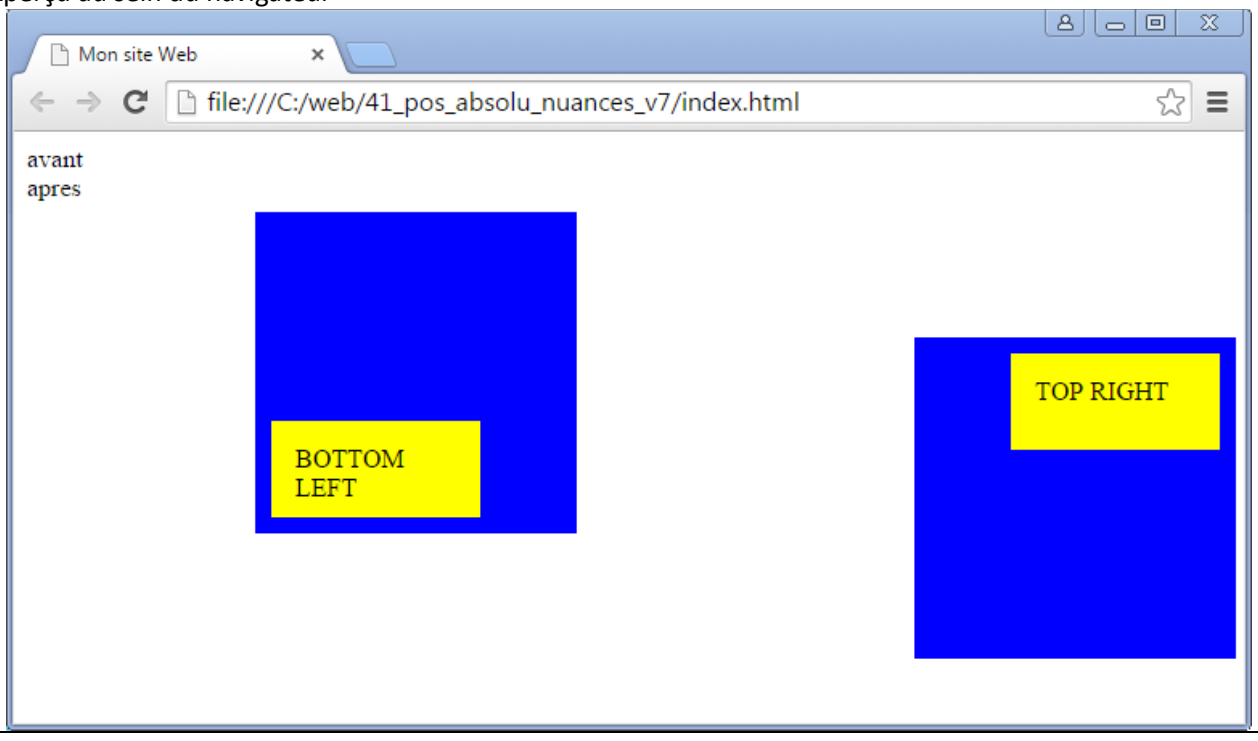
Code HTML de l'exemple

```
<!doctype html>
<html>
    <head>
        <meta charset="UTF-8">
        <title>Mon site Web</title>
        <link rel="stylesheet" href="style.css" />
    </head>

    <body>



    avant
    <div id="bloc1">
        <div id="top_right">TOP RIGHT</div>
    </div>
    <br>
    <div id="bloc2">
        <div id="bottom_left">BOTTOM LEFT</div>
    </div>

    apres

    </body>
</html>
```

Code CSS de l'exemple

```
#bloc1{
position:absolute;
width:200px;
height:200px;
bottom:40px;
right:5px;
background-color: blue;
}
#bloc2{
position:absolute;
width:200px;
height:200px;
top:50px;
left:150px;
background-color: blue;
}
#top_right{
position:absolute;
top:10px;
right:10px;
width:100px;
height:30px;
background-color: yellow;
padding:15px;
}
#bottom_left{
position:absolute;
bottom:10px;
left:10px;
width:100px;
height:30px;
background-color: yellow;
padding:15px;
}
```

Aperçu au sein du navigateur



Les deux blocs « bloc1 » et « bloc2 » sont positionnés dans l'absolu par rapport à la fenêtre du navigateur. Les blocs « top_right » et « bottom_left » sont positionnés de manière absolue mais relativement à leurs conteneurs respectifs « bloc2 » et « bloc1 ».

En plus du positionnement absolu (« pur » ou « relatif »), il existe le positionnement relatif et le positionnement fixe :

> Le positionnement relatif permet de paramétrer un décalage par rapport à la position par défaut lors du flux normal.
>
> Le positionnement fixe permet de maintenir une position fixe au sein de la fenêtre même lorsqu'on déplace la barre de défilement de la page.

La propriété « display »

Lorsqu'on manipule un élément d'une page Web, on le fait au travers d'une balise (« h1 », « div », « b », …). Suivant la balise que l'on utilise on manipulera un élément de type « bloc » (« h1 », « div »,…) ou un élément de type « en ligne » (« b », « i »,…).
En réalité, le fait d'avoir affaire à un élément de type « bloc » ou un élément de type « en ligne » est lié à la valeur par défaut de la propriété « display » de l'élément.

On retiendra trois valeurs différentes de la propriété « display » :

block	↔	On manipule un élément de type « bloc », un passage à la ligne apparaît.
inline	↔	On manipule un élément de type « en ligne » sans passage à la ligne.
inline-block	↔	On manipule un élément comme un « bloc » mais sans passage à la ligne.

Ainsi, si on par exemple créer une barre de navigation où chaque lien se trouve au sein d'un rectangle et où tous les rectangles sont sur la même ligne, on utilisera la propriété « display : inline-block ; ».
On permet de mettre plusieurs zones rectangulaires sur une même ligne sans sortir du flux normal (sans devoir utiliser un positionnement absolu).

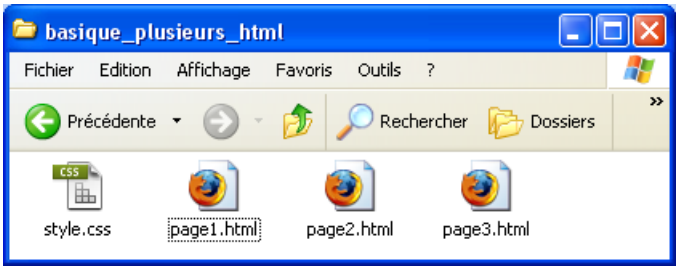
La gestion d'un menu

La manière d'implémenter le menu est libre, il y a en gros deux approches :
- Créer plusieurs pages html et changer de page grâce à des balises <a>…
- Créer une seule page html et gérer les transitions en CSS, Javascript,…

UN FICHIER HTML PAR PAGE

Nous allons commencer par décrire une gestion de menu où chaque page correspond à un fichier HTML qui lui est propre. Il s'agit de la gestion classique d'un menu.
Cette façon de faire est encore beaucoup utilisée aujourd'hui, elle a l'avantage d'être simple.

Imaginons l'arborescence suivante :



Considérons les pages basiques telles que donnée ci-dessous :

page1.html

```
<!doctype html>
<html>
<head>
<meta charset="UTF-8">
<title>page1</title>
<link rel="stylesheet" href="style.css"/>
</head>

<body>

<nav>
<a href="page1.html"> vers page1</a>
<a href="page2.html"> vers page2</a>
<a href="page3.html"> vers page3</a>
</nav>

<p> paragraphe représentant page1</p>

</body>
</html>
```

page2.html

```
<!doctype html>
<html>
<head>
<meta charset="UTF-8">
<title>page2</title>
<link rel="stylesheet" href="style.css"/>
</head>

<body>

<nav>
<a href="page1.html"> vers page1</a>
<a href="page2.html"> vers page2</a>
<a href="page3.html"> vers page3</a>
</nav>

<p> paragraphe représentant page2</p>

</body>
</html>
```

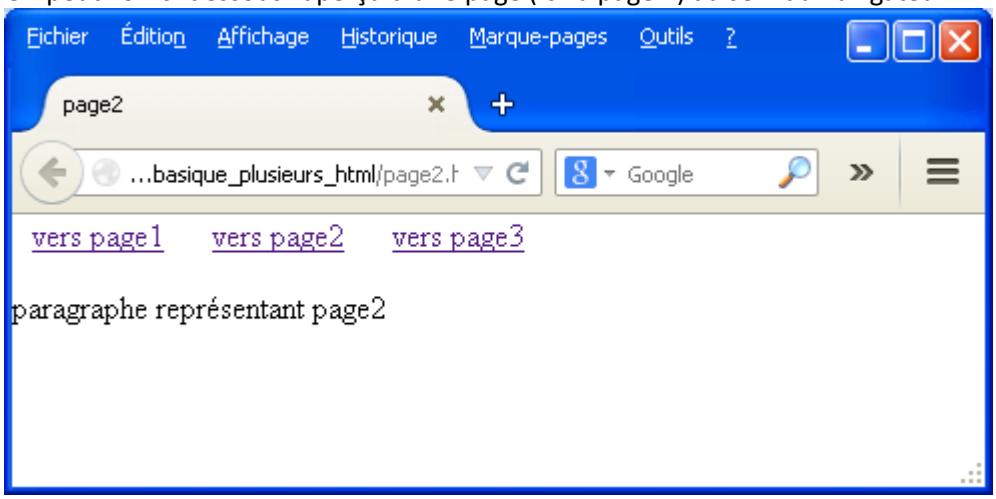
page3.html

```
<!doctype html>
<html>
<head>
<meta charset="UTF-8">
<title>page3</title>
<link rel="stylesheet" href="style.css"/>
</head>

<body>

<nav>
<a href="page1.html"> vers page1</a>
<a href="page2.html"> vers page2</a>
<a href="page3.html"> vers page3</a>
</nav>

<p> paragraphe représentant page3</p>

</body>
</html>
```

On peut remarquer que le contenu de la balise <nav> … </nav> est identique pour chaque page.
Comme tous les fichiers HTML se trouvent dans un même dossier, il suffit de donner le nom du fichier comme chemin.
La balise <a>… permet de créer un lien, le contenu de la balise servira alors de lien ; par défaut le fait de cliquer sur ce qui sert de lien chargera au sein de la fenêtre le fichier dont la référence est donnée par la propriété « href ».
Supposons que l'on soit au sein de la page2.html, le fait de cliquer sur « vers page3 » va avoir pour effet de charger le fichier précisé par la propriété « href » de la balise <a> englobante.
Le navigateur va alors chercher à charger le fichier « page3.html » du dossier courant.

L'exemple illustre des liens au sein d'une balise <nav> ; cette balise est une balise structurelle en HTML5, elle sert à faciliter la gestion de la partie navigation et améliore la lisibilité en indiquant la fonction du bloc.

On peut voir ci-dessous l'aperçu d'une page (ici la page 2) au sein du navigateur.



Le code CSS très basique de cet exemple est donné et expliqué ci-dessous :

Code de style .css

```
@charset "UTF-8";
/* CSS Document */

body {
    margin: 0;
}

nav a{
    margin-left:10px;
    margin-right:10px;
}

nav a:hover{
    background-color:#FF0;
}
```

Explications

Il est pratique de mettre toutes les marges extérieures (margin) au body à 0 afin d'éviter les marges par défaut.

Un sélecteur tel que « nav a » est un sélecteur composé permettant de sélectionner un lien <a> qui se trouve au sein d'une balise <nav>.
On précise ici une marge extérieure à gauche et à droite de 10 pixels pour que les liens ne soient pas collés.

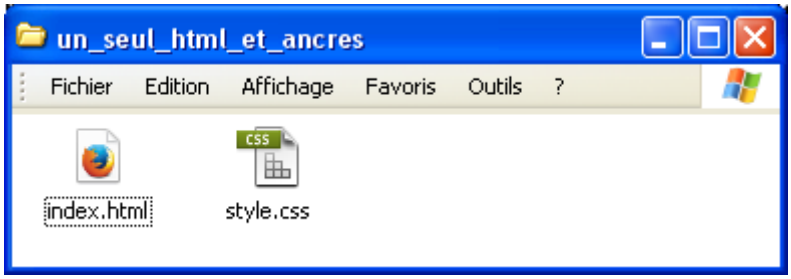
« :hover » correspond à l'état survol de souris.
Ici la couleur de fond d'un lien <a> d'une balise <nav> sera jaune lors du survol de ce lien.

Il faudra bien entendu compléter cet exemple pour améliorer l'esthétique du menu.

UN SEUL HTML ET UTILISATION D'ANCRES

Il devient courant d'utiliser un seul fichier HTML pour plusieurs pages.

On peut alors imaginer l'arborescence suivante :



Pour simuler plusieurs pages au sein d'un même fichier, on fait correspondre chaque page à une balise div (une zone rectangulaire). On s'arrangera alors au sein du CSS pour que les balises div qui correspondent aux pages aient une largeur et une hauteur de 100% du body.

Considérons le code html donné ci-dessous (accompagné d'une brève explication) :

Code de la page index.html

```
<!doctype html>
<html>
<head>
<meta charset="UTF-8">
<title>Exemple de menu</title>
<link rel="stylesheet" href="style.css"/>
</head>

<body>

<nav>
<a href="#page1">PAGE1</a>
<a href="#page2">PAGE2</a>
<a href="#page3">PAGE3</a>

</nav>

<div id="page1"></div>
<div id="page2"></div>
<div id="page3"></div>
</body>
</html>
```

Quelques explications

Comme pour l'exemple précédant la navigation contient des liens vers les différentes pages.
Cependant ici les liens font références à des éléments de la même page. On appelle cela des ancres.

Pour faire un lien vers un élément de la page courante, il suffit de noter « # » suivi de l' « id » de l'élément à atteindre.

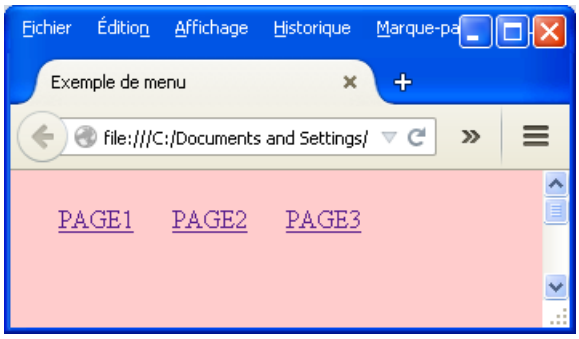
Les différentes pages correspondent à des balises div identifiables à l'aide d'un « id » unique.

On peut remarquer ici que les balises des pages sont vides ; ceci n'est bien entendu qu'un exemple, il faudra y ajouter les contenus pour un site réel.

Pour l'exemple, on pourra distinguer chaque page grâce à une couleur de fond qui lui est propre. La couleur de fond de chaque page est alors précisée au sein du CSS.

Comme pour l'exemple précédent, les différents liens sont compris au sein d'une balise <nav>.

Voici l'aperçu de la page1 au sein d'un navigateur :



Ici comme il n'y a qu'une seule page, une façon simple de gérer le positionnement du menu est de donner la propriété « position : fixed ; » à la balise <nav>.

Par prudence on peut remarquer dans le code CSS une valeur de 999 pour le z-index de la navigation. L'idée est de donner une valeur élevée du z-index (couche supérieure) pour la garder visible par défaut.

Code de style.css

```
body {
    margin: 0;
}

nav {
    position: fixed;
    top: 20px;
    left: 20px;
    z-index: 999; /* grande valeur pour la garder au dessus */
}

nav a {
    margin-left:10px;
    margin-right:10px;
}

nav a:hover {
    background-color: #FF0;
}
```

Suite du code de style.css

```
#page1 {
    position: absolute;
    width: 100%;height: 100%;
    background:#FCC;
    top: 0px;
}
#page2 {
    position: absolute;
    width: 100%;height: 100%;
    background:#E9E484;
    top: 100%;
}
#page3 {
    position: absolute;
    width: 100%;height: 100%;
    background:#3CC;
    top: 200%;
}
```

Le bloc navigation a une position fixe et est positionné à 20 pixels du bord supérieur et 20px du bord gauche.

Les trois balises correspondant aux pages sont positionnées de manière absolue, ont une largeur et hauteur de 100% du body et ont une couleur de fond qui leur est propre.
Chaque page est alors positionnée par rapport au bord supérieur :

- La page1 est collée au bord supérieur (« top »)
- La page2 se trouve juste après la page 1 donc à une distance de 100% du « top ».
- La page3 se trouve juste après la page 2 donc à une distance de 200% du « top ».

La fenêtre ne fait apparaître qu'une hauteur de 100%, l'idée est que les trois pages se suivent mais qu'une seule apparaît à l'écran.

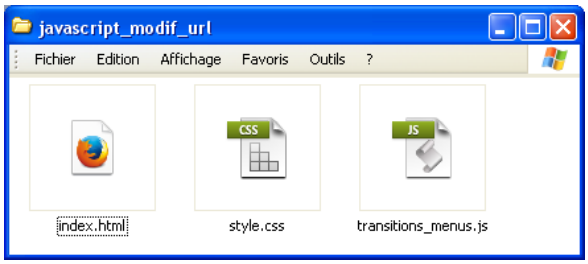
JAVASCRIPT POUR MODIFIER L'URL

Jusqu'à maintenant nous avons utilisé des liens (balises <a>) pour gérer la navigation du menu.
Le principe était le même que l'on ait plusieurs fichiers html ou un seul : le click sur un lien permettait de modifier la « location » du document à apparaître au travers du navigateur.

Avec plusieurs fichiers html la « location » du document correspondait au fichier html à ouvrir alors qu'avec un seul fichier html la « location » du document correspondait à un emplacement au sein du même fichier identifiable grâce à une ancre.

Nous allons maintenant modifier manuellement la « location » du document lors d'un click de souris en utilisant une fonction javascript.
Pour rendre les codes plus lisibles, nous suggérons de mettre le code javascript dans un fichier réservé : celui-ci se terminera par l'extension « .js ».

Voici le contenu du dossier :



Voici le code html de l'exemple (accompagné d'une brève explication) :

Code de la page index.html

```
<!doctype html>
<html>
<head>
<meta charset="UTF-8">
<title>Exemple de menu</title>
<link rel="stylesheet" href="style.css"/>
<script type="text/javascript" src="transitions_menus.js"></script>
</head>

<body>

<nav>
<h6 ONCLICK="click_bouton_page1()"> PAGE1 </h6>
<h6 ONCLICK="click_bouton_page2()"> PAGE2 </h6>
<h6 ONCLICK="click_bouton_page3()"> PAGE3 </h6>
</nav>

<div id="page1"></div>
<div id="page2"></div>
<div id="page3"></div>

</body>
</html>
```

Quelques explications

L'en-tête fait apparaître une balise <script> qui permet de préciser la source du fichier javascript.

On continue à utiliser une balise <nav> pour mettre en évidence la navigation.

Les balises qui serviront de liens sont ici <h6>, c'est un choix qui a été fait, on aurait pu faire autrement.
On peut préciser le nom de la fonction à exécuter lors d'un click grâce à la propriété « ONCLICK »

Les différentes pages sont toujours symbolisées par des

Remarque : parfois le script est mis juste avant la fin du body plutôt que dans l'en-tête.

La propriété « ONCLICK » permet de définir les actions à effectuer lors de l'événement «clic de souris » sur un élément.
Ainsi, si on clique sur le texte « PAGE2 » au sein de la balise <h6>, l'action qui sera effectuée sera l'action décrite au sein de la fonction « click_bouton_page2() » de notre fichier javascript.
Nous pouvons choisir le nom de la fonction comme nous voulons (en évitant cependant les espaces, caractères spéciaux,…) pour autant que ce soit le même nom au sein de notre fichier javascript.

Dans notre exemple le fichier javascript s'appelle « transitions_menus.js », voici son contenu :

Code du fichier transitions_menus.js »

```
// JavaScript Document

function click_bouton_page1()
    {
    document.location="#page1"
    }

function click_bouton_page2()
    {
    document.location="#page2"
    }

function click_bouton_page3()
    {
    document.location="#page3"
    }
```

Quelques explications

La fonction nommée « click_bouton_page1 » va modifier la « location » du document du navigateur en lui donnant l'ancre « #page1 » → la page 1 s'affiche

La fonction nommée « click_bouton_page2 » va modifier la « location » du document du navigateur en lui donnant l'ancre « #page2 » → la page 2 s'affiche

La fonction nommée « click_bouton_page3 » va modifier la « location » du document du navigateur en lui donnant l'ancre « #page3 » → la page 3 s'affiche

Le code est très semblable à l'exemple précédent sauf qu'ici nous avons « h6 » au lieu de « a ».
Avec « h6 » il a fallu ajouter « display : inline-block ; » pour maintenir l'affichage sur une ligne.

Code de style.css

```
body {
    margin: 0;
}

nav {
    position: fixed;
    top: 20px;
    left: 20px;
    z-index: 999; /* grande valeur pour la garder au dessus */
}

nav h6 {
    display:inline-block;
    margin-left:10px;
    margin-right:10px;
}

nav h6:hover {
    background-color: #FF0;
}
```

Suite du code de style.css

```
#page1 {
    position: absolute;
    width: 100%;height: 100%;
    background:#FCC;
    top: 0px;
}
#page2 {
    position: absolute;
    width: 100%;height: 100%;
    background:#E9E484;
    top: 100%;
}
#page3 {
    position: absolute;
    width: 100%;height: 100%;
    background:#3CC;
    top: 200%;
}
```

JAVASCRIPT POUR AGIR SUR DU CSS

Le code HTML ici est assez proche de l'exemple précédant, c'est principalement le Javascript qui change.

L'arborescence du dossier racine est identique à l'exemple précédant ainsi que le contenu du fichier index.html.

Le fichier javacript est cependant différent, voici son code :

Code du fichier transitions_menus.js »

```
// JavaScript Document

function click_bouton_page1()
    {
    document.getElementById("page1").style.top = "0%";
    document.getElementById("page2").style.top = "100%";
    document.getElementById("page3").style.top = "200%";
    }

function click_bouton_page2()
    {
    document.getElementById("page1").style.top = "-100%";
    document.getElementById("page2").style.top = "0%";
    document.getElementById("page3").style.top = "100%";
    }

function click_bouton_page3()
    {
    document.getElementById("page1").style.top = "-200%";
    document.getElementById("page2").style.top = "-100%";
    document.getElementById("page3").style.top = "0%";
    }
```

Quelques explications

Chaque fonction va mettre à jour la propriété CSS « top » des éléments d'« id»
« page1 », « page2 » et « page3 ».

Pour la page1 on met à 0 le « top » de la balise « page1 ». Le « top » des pages 2 et 3 sont respectivement mis à 100% et 200%.

Pour la page2 on met à 0 le « top » de la balise « page2 ». Le « top » des pages 1 et 3 sont respectivement mis à -100% et 100%.

Pour la page3 on met à 0 le « top » de la balise « page3 ». Le « top » des pages 1 et 2 sont respectivement mis à -200% et -100%.

En écrivant « document.getElementById (…)» on sélectionne l'élément dont l' « id» est précisé entre parenthèses. Une fois sélectionné on peut dire qu'on veut agir sur le style de cet élément.

Une ligne telle que « document.getElementById("page2").style.top = "-100%"; » signifie :

On sélectionne l'élément dont l'« id » est « page2 » pour modifier la propriété de style « top » et lui mettre une nouvelle valeur de « -100% ».

Il est très intéressant de pouvoir modifier des valeurs de propriétés CSS à l'aide du javascript.
Cette façon de faire va ajouter du dynamisme et va permettre de faire apparaître des effets et des transitions.
Pour donner un exemple de transition, nous avons légèrement modifié le code CSS de l'exemple précédant.

Voici le code CSS :

Code de style.css

```css
body {
    margin: 0;
}

nav {
    position: fixed;
    top: 20px;
    left: 20px;
    z-index: 999; /* grande valeur pour la garder au dessus */
}

nav h6 {
    display:inline-block;
    margin-left:10px;
    margin-right:10px;
}

nav h6:hover {
    background-color: #FF0;
}
```

Suite du code de style.css

```css
#page1, #page2, #page3 {
    position: absolute;
    width: 100%;height: 100%;
    background:#FCC;
    transition: 0.3s;
}

#page{
    background:#FCC;
    top: 0%;
}

#page2 {
    background:#E9E484;
    top: 100%;
}

#page3 {
    background:#3CC;
    top: 200%;
}
```

La partie de gauche n'a pas changé vis-à-vis de l'exemple précédant.
La partie de droite ajoute la propriété « transition : 0.3s ;» aux différentes pages.
Cette transition met en évidence le temps qui sera d'application pour transiter d'une valeur de propriété à une autre.

On peut également remarque que l'organisation des propriétés a été faite différemment. En effet, les propriétés communes aux différentes pages ont été rassemblées pour plus de lisibilité.
La virgule permet de préciser que plusieurs sélecteurs sont concernés par les propriétés.

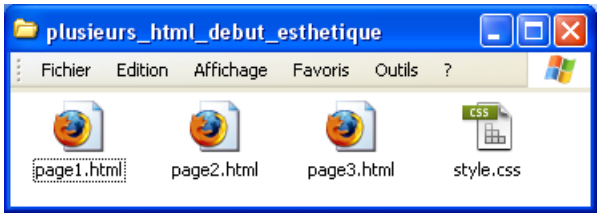
L'aspect esthétique d'un menu

Dans les pages précédentes nous avons discuté de la gestion du menu au niveau de sa fonctionnalité. Ici nous allons décrire quelques éléments liés à l'esthétique du menu.

Il faut pouvoir garder une indépendance entre la fonctionnalité et l'esthétique ; autrement dit, ce n'est pas parce qu'on modifie l'esthétique que cela doit influencer le fonctionnement et, inversement, le fait de modifier le fonctionnement ne doit pas interagir sur l'esthétique.

Nous allons repartir de notre exemple basique (un fichier HTML par page) et ajouter quelques propriétés esthétiques pour améliorer l'esthétique.

Le contenu du dossier racine est identique :



Les pages HTML sont presque identiques à l'exception du fait que nous avons ajouté la balise <h6> pour encadrer le texte qui sert de lien.

Code page1.html avec <h6> ajouté

```
 <!doctype html>
<html>
<head>
<meta charset="UTF-8">
<title>page1</title>
<link rel="stylesheet" href="style.css"/>
</head>

<body>

<nav>
<a href="page1.html"> <h6>vers page1</h6></a>
<a href="page2.html"> <h6>vers page2</h6></a>
<a href="page3.html"> <h6>vers page3</h6></a>
</nav>

<p> paragraphe représentant page1</p>

</body>
</html>
```

Code page1.html de l'exemple

```
<!doctype html>
<html>
<head>
<meta charset="UTF-8">
<title>page1</title>
<link rel="stylesheet" href="style.css"/>
</head>

<body>

<nav>
<a href="page1.html"> vers page1</a>
<a href="page2.html"> vers page2</a>
<a href="page3.html"> vers page3</a>
</nav>

<p> paragraphe représentant page1</p>

</body>
</html>
```

L'intérêt d'ajouter la balise <h6> est de manipuler un élément de type bloc ; par ailleurs ce type de balise a déjà quelques propriétés CSS par défaut.
Nous aurions très bien pu utiliser une balise à la place ou encore un autre élément.

Les modifications esthétiques vont maintenant se faire au sein de la feuille de style.

Code de la page style.css

```css
body {
    margin: 0;
}

nav a{
    text-decoration: none;
    color:#000; /* redéfini la couleur par défaut des liens */
}


nav h6{
    display: inline-block;

    background: #000;
    color:#DDD; /* redéfini la couleur */

    border-radius: 4px;

    margin: 15px;
    padding: 8px;


    font-size: 9px;
    font-family: Gotham, "Helvetica Neue", Helvetica, Arial, sans-serif;
    text-transform:uppercase;
    letter-spacing: 2px;

    transition: 1s;
}

nav h6:hover{
    background-color:#AAA;
    color:#222;
}
```

Quelques explications

Les marges extérieures sont mises à 0.

Les liens <a> dans une balise <nav> ne sont pas soulignés par défaut et ont une couleur noire.

Pour un <h6> dans un <nav> :

Affichage présenté sur une ligne

Couleurs de fond et de texte respectivement noir et gris clair.

Coins arrondis avec un rayon de 4px.

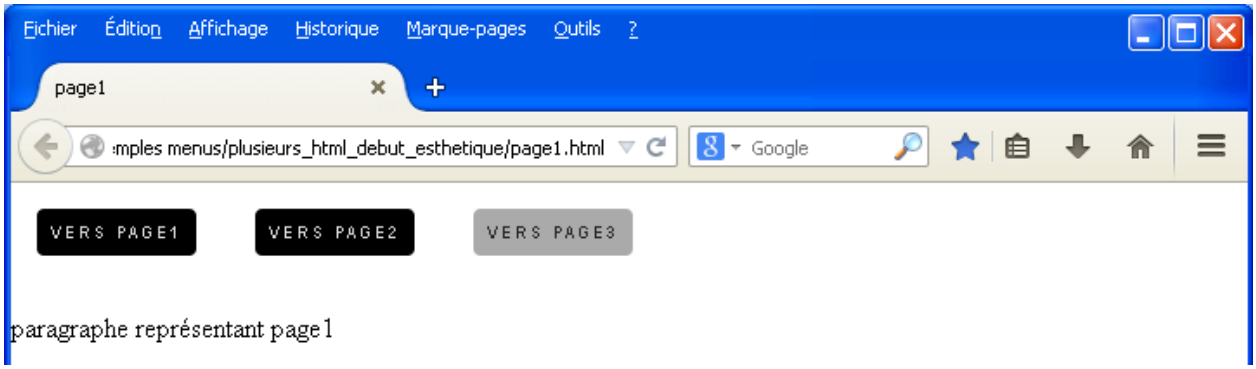
Marges extérieures et intérieures respectivement de 15px et 8px.

Définition d'éléments de texte : Taille, police, mise en majuscule et espacement entre les lettres.

1s de transition lors de variations CSS.

Lorsqu'on survol avec la souris, les couleurs de fond et de texte changent. Le délai de transition précédant est appliqué.

On peut voir ci-dessous l'aperçu de la page1.html avec le lien « vers page3 » survolé par la souris :



La galerie photo

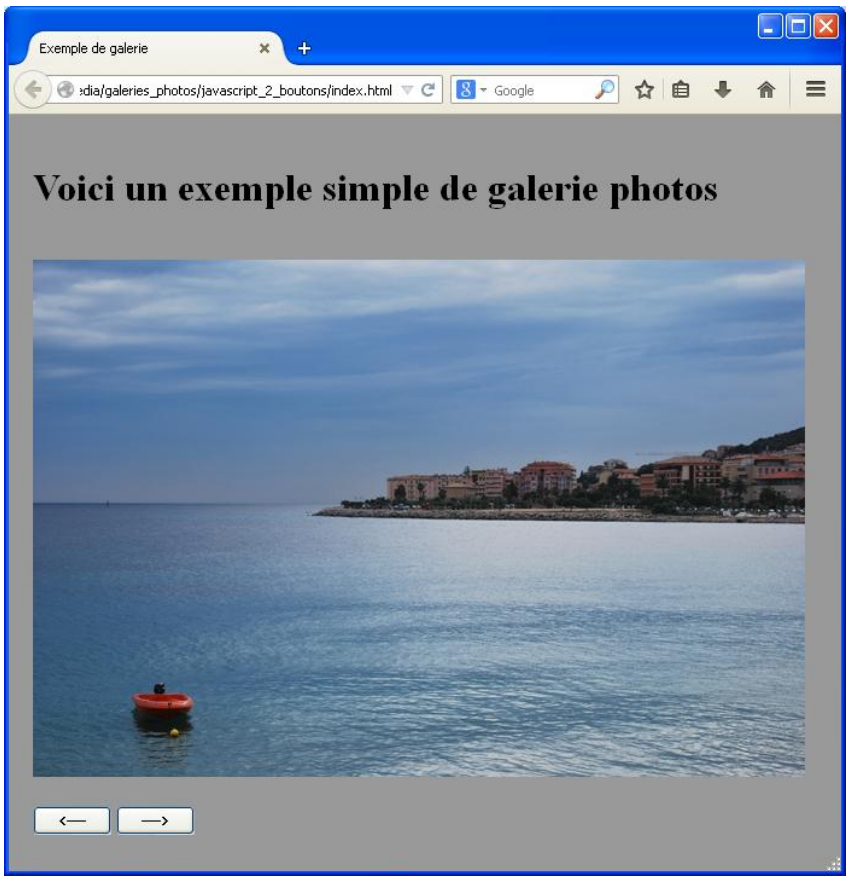
Il y a de nombreuses façons d'implémenter une galerie photo.
Nous allons nous contenter d'illustrer trois exemples qui mettent en oeuvre du javascript.

GALERIE PHOTO VIA LE CHAMP « SRC »

Nous allons montrer un exemple qui repose sur le principe de modifier le champ « src » d'une image à l'aide de deux boutons.

Au sein du code HTML, une première image est insérée. On a cependant pris soin, lors de l'insertion de l'image, de donner un nom à la « zone » qui contient cette image.
Ainsi, grâce au nom de la zone, on peut modifier le champ « src » de l'image.

L'aperçu de l'exemple est donné ci-dessous :



On peut trouver ci-dessous les codes HTML et CSS de cet exemple :

Le code de index.html

```
<!doctype html>
<html>
<head>
<meta charset="UTF-8">
<title>Exemple de galerie</title>
<link rel="stylesheet" href="style.css"/>
<script type="text/javascript" src="galerie.js"></script>
</head>

<body>

    <h1> Voici un exemple simple de galerie photos </h1>
    <br>

    <img name="photo_en_cours" src= "images/bateau.jpg"/>
    <br><br>

    <INPUT TYPE="BUTTON"  VALUE="    <----    " ONCLICK="gauche()">
    <INPUT TYPE="BUTTON" VALUE="    ---->    " ONCLICK="droite()">

</body>
</html>
```

Le code de style.css

```
@charset "UTF-8";
/* CSS Document */

body {
    margin: 0;
    padding: 20px;
    background-color:#999;
}
```

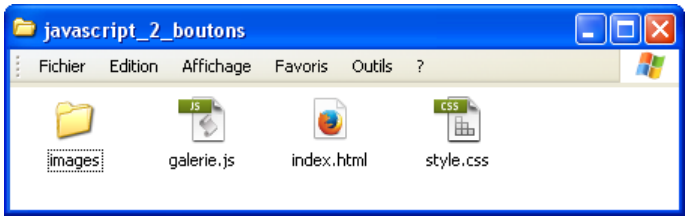
Le fichier HTML fait apparaître :

- Un lien vers la feuille de style et un lien vers un fichier javascript externe.
  Ces deux liens se trouvent au sein du <head>.

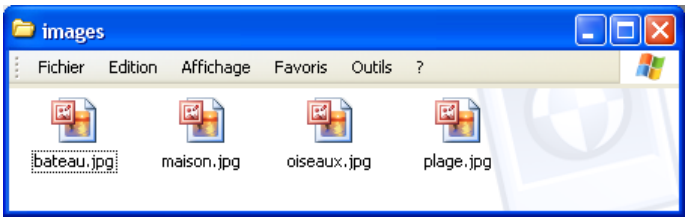
- Une image au sein du <body> dont la source est « images/bateau.jpg ».
  La source correspond au chemin relatif en partant du dossier courant (celui qui contient le fichier index.html) ; il précise qu'il faut entrer dans le dossier « images » avant d'accéder à l'image « bateau.jpg ».
  La balise montre également le champ « name » qui vaut « photo_en_cours » ; ce champ permet de préciser un nom à la zone de manière à pouvoir modifier la source de l'image qui s'y trouvera.

- Deux balises <input> de type « button » précisant le nom de la fonction à exécuter lorsqu'on clique dessus avec la souris.

Le contenu du dossier racine ici à droite:



Le dossier racine contient un dossier « images » dont le contenu est ici à droite:



Le contenu du fichier javascript est donné ci-dessous :

Code de la page galerie.js

```
// JavaScript Document

var nombre_images= 4    //on adapte ici en fonction du nombre d'images
var indice_image = 0    //on commence la numérotation à 0
var nom_photos =  ["bateau.jpg","maison.jpg","oiseaux.jpg","plage.jpg"];



function gauche()
    {
    indice_image=indice_image-1
    if(indice_image==-1)
        {indice_image=nombre_images-1}
    document.photo_en_cours.src="images/"+nom_photos[indice_image]
    }

function droite()
    {
    indice_image=indice_image+1
    if(indice_image==nombre_images)
        {indice_image=0}
    document.photo_en_cours.src="images/"+nom_photos[indice_image]
    }
```

Quelques explications

Trois variables sont utilisées :
- *nombre_images* (permet de paramétrer)
- *indice_image pour indiquer au* navigateur
l'image en cours à afficher
- *nom_photos* qui est un tableau qui contient
les noms d'images à afficher.

Les deux fonctions gauche() et droite() sont les fonctions qui sont appelées lorsqu'on appuie sur les boutons inséré dans l'HTML.

Ces deux fonctions ont un principe identique :
1° Mettre à jour la valeur de indice_image
2° modifier la source de l'image dont « name » vaut « photo_en_cours ».
La nouvelle source se réfère au tableau nom_photos et à la valeur de indice_image pour constituer la source complète.

GALERIE JAVASCRIPT MODIFIANT DU CSS

Nous allons ici illustrer une galerie photo suivant un principe déjà vu lors de la gestion des menus. Nous allons utiliser du javascript pour influencer des propriétés CSS d'éléments.

L'idée est de créer un conteneur très très large (assez pour contenir toutes les photos mises cotes à cotes) dont la hauteur est celle des photos.
Les différentes photos sont alors insérées au sein du conteneur les unes à coté des autres.

Par simplicité nous mettons les photos en positionnement absolu de manière à les rendre indépendantes de la position des autres photos et ainsi pouvoir agir sur le positionnement de chaque photo de manière séparée.
Pour éviter un positionnement absolu « pur », le conteneur des photos se trouve en positionnement relatif. Ainsi, le positionnement « absolu » des différentes photos reste lié au conteneur des photos.

L'idée sera de créer des miniatures cliquables pour sélectionner la photo à afficher.
La photo correspondante sera alors affichée en appelant la fonction javascript correspondante qui positionnera cette photo sur le bord gauche de son conteneur. Les positions des autres photos seront également mises à mises à jour.

L'arborescence interne du dossier racine est exactement le même que pour l'exemple précédant.
Voici ci-dessous son code HTML :

Code de la page index.html

```
<!doctype html>
<html>
<head>
<meta charset="UTF-8">
<title>Exemple de galerie</title>
<link rel="stylesheet" href="style.css"/>
<script type="text/javascript" src="galerie.js"></script>
</head>

<body>

    <hl> Voici un exemple simple de galerie photos </hl>
    <br>

    <div id="conteneur_images">
        <img id="image1" src= "images/bateau.jpg"/>
        <img id="image2" src= "images/maison.jpg"/>
        <img id="image3" src= "images/oiseaux.jpg"/>
        <img id="image4" src= "images/plage.jpg"/>
    </div>

    <div id="conteneur_minis">
        <img ONCLICK="clic_mini1()" src= "images/mini_bateau.jpg"/>
        <img ONCLICK="clic_mini2()" src= "images/mini_maison.jpg"/>
        <img ONCLICK="clic_mini3()" src= "images/mini_oiseaux.jpg"/>
        <img ONCLICK="clic_mini4()" src= "images/mini_plage.jpg"/>
    </div>

</body>
</html>
```

Quelques explications

L'en-tête fait apparaître une balise <script> qui permet de préciser la source du fichier javascript.

Le conteneur des images contient les 4 images de la galerie.
Chaque image est identifiable grâce à un « id » unique.

Les miniatures sont également contenues au sein d'un conteneur.
Chaque miniature précise une fonction qui lui est propre lorsqu'on clique dessus avec la souris.

Et voici le code de la feuille de style :

Code de style.css

```
body {
    margin: 0;
    padding: 20px;
    background-color:#999;
}

#conteneur_images{
    position:relative;
    height:430px;
    margin-left:0px;
    }

#image1,#image2, #image3, #image4{
    position:absolute;
    top:0px;
    margin:0px;
    padding:0px;
    width:640px;
    height:430px;
    transition:1s;
    }
```

Suite du code

```
#image1{
    left : 0px;
}
#image2{
    left : 640px;
}
#image3{
    left : 1280px;
}
#image4{
    left : 1920px;
}

#conteneur_minis{
    margin-top:20px;
}
```

Quelques explications

Le conteneur_images a une hauteur égale à celle des images.
Sa position est en relatif pour que les images se positionnent en « absolu » par rapport à lui.

Pour augmenter la lisibilité, les propriétés communes aux différentes images sont mises ensemble. Les images sont positionnées à 0px par rapport au bord supérieur de leur conteneur.

Chaque image se positionne alors à un endroit précis par rapport au bord gauche de leur conteneur.

Au départ l'aperçu est comme ci-dessous :

Et voici le code javascript :

Code de galerie.js

```
function clic_mini1()
    {
    document.getElementById("image1").style.left = "0";
    document.getElementById("image2").style.left = "640px";
    document.getElementById("image3").style.left = "1280px";
    document.getElementById("image4").style.left = "1920px";
    }

function clic_mini2()
    {
    document.getElementById("image1").style.left = "-640px";
    document.getElementById("image2").style.left = "0";
    document.getElementById("image3").style.left = "640px";
    document.getElementById("image4").style.left = "1280px";
    }
```

Suite du code

```
function clic_mini3()
    {
    document.getElementById("image1").style.left = "1280px";
    document.getElementById("image2").style.left = "-640px";
    document.getElementById("image3").style.left = "0";
    document.getElementById("image4").style.left = "640px";
    }

function clic_mini4()
    {
    document.getElementById("image1").style.left = "-1920px";
    document.getElementById("image2").style.left = "-1280px";
    document.getElementById("image3").style.left = "-640px";
    document.getElementById("image4").style.left = "0";
    }
```

Chaque fonction positionne son image sur le bord gauche du conteneur (« left » à 0px) et la rend visible. Toutes les autres images sont positionnées en fonction.

Dynamisme et police personnalisée

LES TRANSITIONS

Les transitions permettent de faire apparaître du mouvement au travers de changement de propriétés CSS. Une transition est paramétrée au sein d'un sélecteur avec le mot clé « transition » suivi d'une valeur en secondes.

Ex : transition : 1s ;

Nous avons déjà vu des exemples d'utilisation des transitions lors des gestions de menus et des galeries. Par exemple, dans la dernière galerie, nous avons paramétré une transition d'une seconde pour chaque image ; ainsi, lorsqu'on modifie une propriété CSS d'une image (l'opacité dans ce cas), il apparaît une durée d'une seconde avant que la nouvelle propriété CSS soit atteinte.

Une bonne utilisation des transitions ajoute du dynamisme au site.

SURVOL DE LIENS

Le survol des liens ajoute une interaction au site.
Le survol est géré grâce à la pseud-classe « hover ».

Exemple :
```
nav h6 :hover{
background-color : #AAA ;
}
```

L'idée est que le mot clé « hover » complète un sélecteur pour définir les propriétés CSS liées à ce sélecteur survolé par la souris. L'utilisation la plus courante est le fait de prendre en charge le survol d'un lien de navigation.

POLICE PERSONNALISEE

Les polices disponibles de base sont assez limitées. D'ailleurs si on veut s'assurer que le tout le monde puisse visualiser notre site avec la bonne police il faut prendre soin d'insérer le fichier de la police au sein de notre dossier racine.
Les fichiers de police ont des formats différents et ne sont pas tous compris par les différents navigateurs. Pour assurer une compatibilité maximale nous utiliserons des polices d'extension « .ttf » mais d'autres formats utiles existent.
Il faut aussi savoir que toutes les polices ne sont pas libres d'utilisation, pour télécharger des polices que vous pouvez utiliser librement vous pouvez par exemple vous rendre sur fontsquirrel.com
Vous pouvez alors déclarer les différentes polices que vous comptez utiliser au sein de votre CSS avec l'outil « @font-face »

Vous pourrez par exemple déclarer deux polices comme indiquées ci-dessous :

```
@font-face
        {
        font-family: 'nob';         /*nom au choix*/
        src: url('police/nobile/nobile.ttf') format('truetype');
        }
```

```
@font-face
        {
        font-family: 'calig';        /*nom au choix*/
        src: url('police/calligraffiti/calligraffiti.ttf') format('truetype');
        }
```

On donne donc un nom au choix pour chaque police (ici les noms sont « nob » et « calig ») et précisant le chemin relatif des fichiers contenant les polices.

Pour ensuite utiliser une police personnalisée au sein d'une zone il faut préciser comme dans l'exemple ci-dessous :

```
#emballage
        {
        font-family:'nob';
        }
.ma_zone
        {
        font-family:'calig ';
        }
```

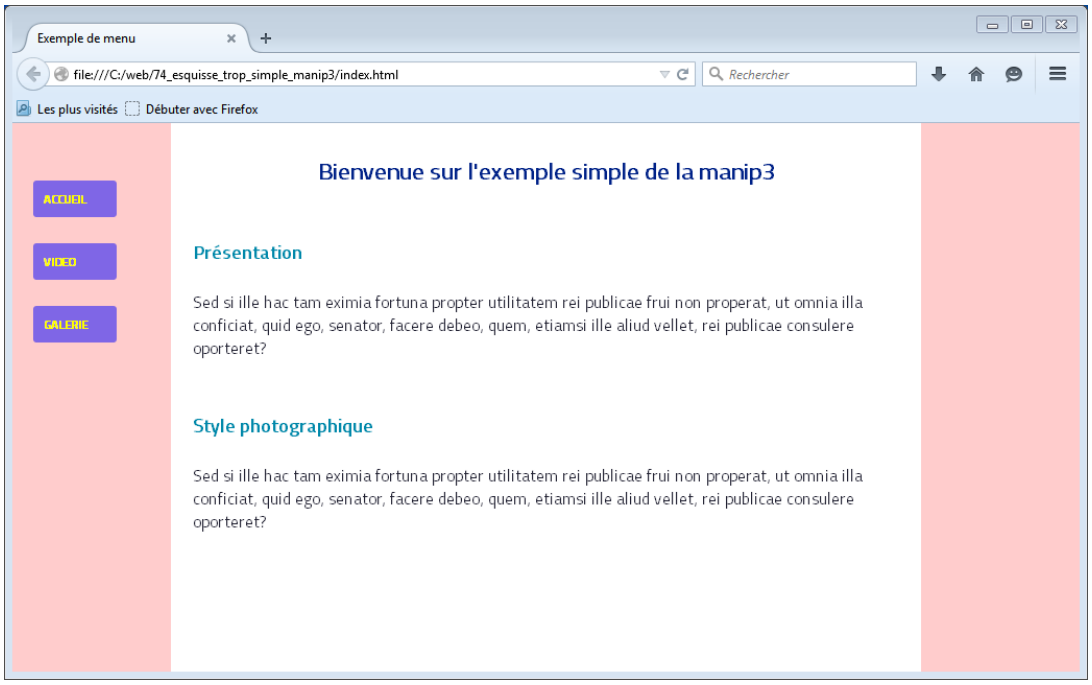
Un bon choix de police apporte un aspect visuel important à un site Web. Il est d'ailleurs conseillé d'y attacher une grande importance et de s'assurer que le rendu sera correct sur différents navigateurs.
Pour ce faire, il est intéressant de proposer des polices de caractères de réserve et de proposer différents formats de police.

Comme d'habitude, il est conseillé de tester le rendu de votre site sur différents navigateurs et avec des tailles différentes d'écrans.

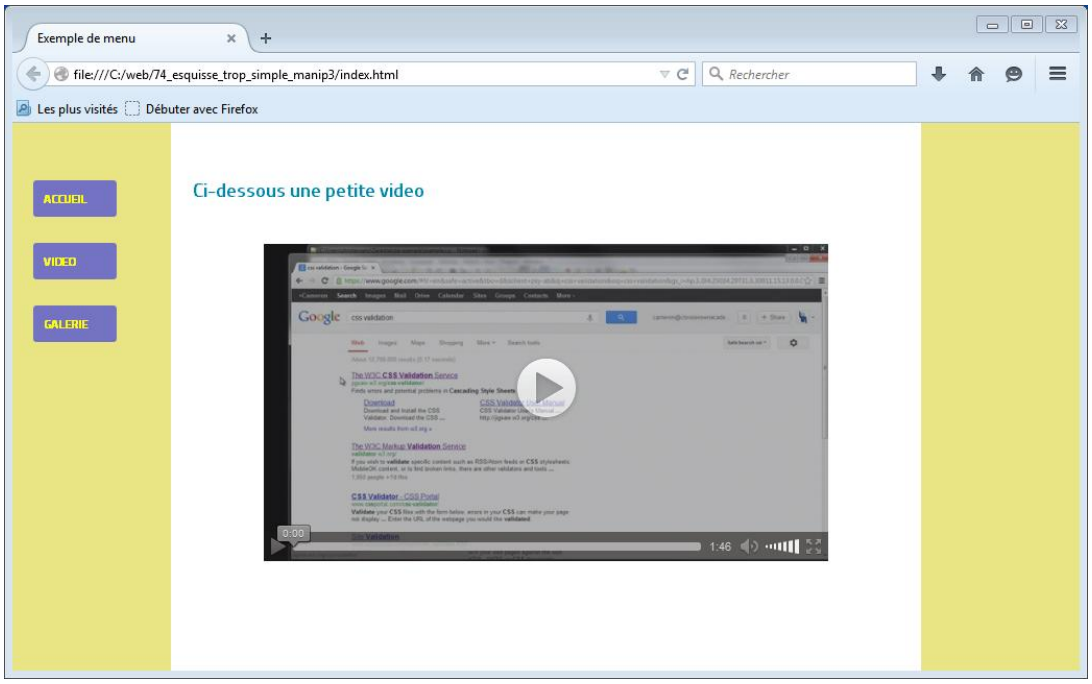
Exemple de solution de la manipulation 3

L'exemple donné ici est volontairement trop simple. Vous trouverez les aperçus des pages suivis de ses codes.

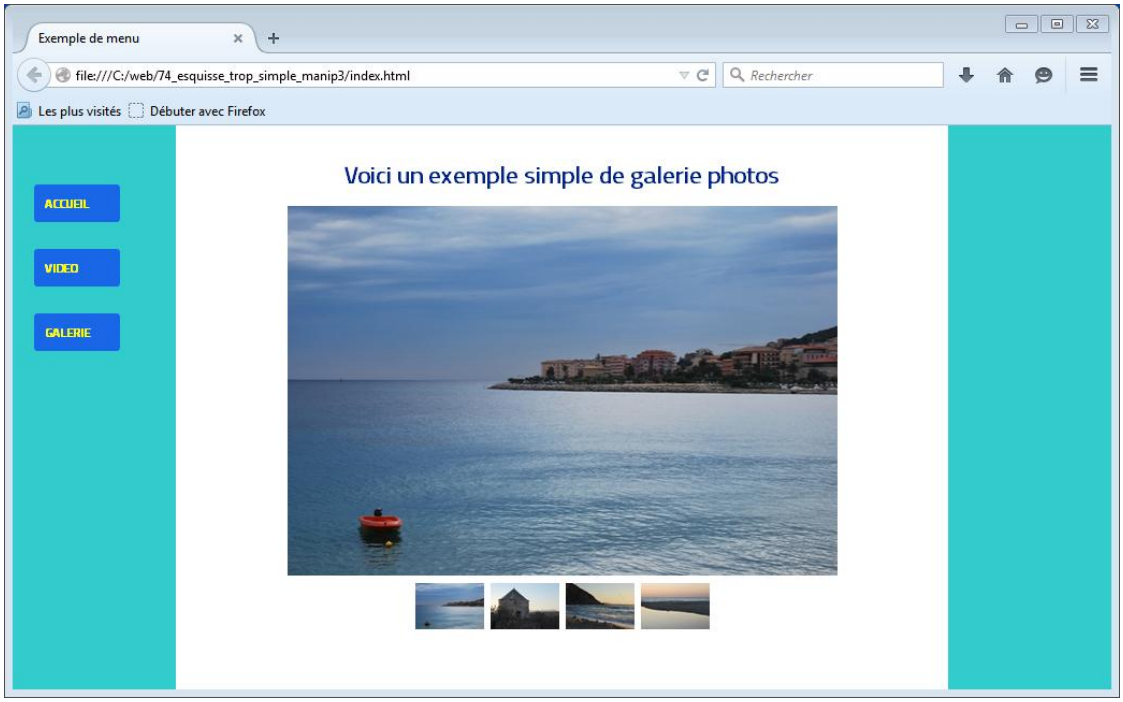
Aperçu de la page 1 :



Aperçu de la page 2 :



Aperçu de la page 3 :



Structure générale de « index.html » :

```
<!doctype html>
<html>
<head>
<meta charset="UTF-8">
<title>Exemple de menu</title>
<link rel="stylesheet" href="style.css"/>
<script type="text/javascript" src="transitions_menus.js"></script>
<script type="text/javascript" src="galerie.js"></script>
</head>

<body>

<nav>

<div id="conteneur_pages">
    <div id="page1">
        <div class ="central">
    </div>

    <div id="page2">
        <div class ="central">
    </div>

    <div id="page3">
</div>

</body>
</html>
```

CSS de mise en forme de la navigation et d'éléments généraux :

```css
@font-face {
    font-family: my_font;
    src: url('polices/titillium/titillium_regular.otf'),
        url('polices/pt_sans/PTS55F.ttf');
}

html,body {
    width:100%;height:100%;
    margin: 0px;padding:0px;
    overflow:hidden;    /*désactiver pendant le debug*/
    font-family:my_font, Arial;
}
nav {
    max-width:10%;
    position: fixed;
    top: 30px;left: 20px;
    z-index: 999; /* grande valeur pour la garder au dessus */
}

nav h6 {
    width: 60px;padding : 10px;
    background-color:rgba(0,0,255,0.5);
    border-radius: 3px;
    color:#FFFF00;
    transition:0.4s;
}

nav h6:hover {
    background-color:rgba(255,255,0,0.5);
    color:#0000FF;
    margin-left:10px;
}
#conteneur_pages{
    position:absolute;      /*influence dimensions de ses div internes*/
    width:100%;             /*pour servir de référence aux pages*/
    height:100%;            /*le débordement des pages est prévu*/
    overflow:visible;       /* on maintient la valeur par défaut*/
    top:0%;left:0%;
    transition:1s;
}

#page1, #page2, #page3 {
    position: absolute;
    width: 100%;height: 100%;
    padding:0px;
}
```

Suite du CSS (lié aux pages) :

```css
#page1{
    background:#FCC;
    left: 0%;
}

#page2 {
    background:#E9E484;
    left: 100%;
}

#page3 {
    background:#3CC;
    left: 200%;
}

.central{
width:70%;height:100%;
margin:auto;
background-color:#fff;
padding:1px;            /*evite certaines fusion de marges*/
}

.contenu{
margin : 30px 20px 30px 20px;
}
```

HTML de « nav » et Javascript « transition_menus.js » :

```html
<nav>
<h6 ONCLICK="click_bouton_page1()"> ACCUEIL </h6>
<h6 ONCLICK="click_bouton_page2()"> VIDEO </h6>
<h6 ONCLICK="click_bouton_page3()"> GALERIE </h6>
</nav>
```

```javascript
function click_bouton_page1()
    {
    document.getElementById("conteneur_pages").style.left= "0%";
    }

function click_bouton_page2()
    {
    document.getElementById("conteneur_pages").style.left = "-100%";
    }

function click_bouton_page3()
    {
    document.getElementById("conteneur_pages").style.left = "-200%";
    }
```

Code HTML lié à la page1 :

```
<div id="conteneur_pages">
    <div id="page1">
        <div class ="central">
            <h1> Bienvenue sur l'exemple simple de la manip3 </h1>

            <div class="contenu">
            <h2>Présentation</h2>
                <p>Sed si ille hac tam eximia fortuna propter utili
                rei publicae frui non properat, ut omnia illa confi
                quid ego, senator, facere debeo, quem, etiamsi ille
                rei publicae consulere oporteret?</p>

            <h2>Style photographique</h2>
                <p>Sed si ille hac tam eximia fortuna propter utili
                rei publicae frui non properat, ut omnia illa confi
                quid ego, senator, facere debeo, quem, etiamsi ille
                rei publicae consulere oporteret?</p>
            </div>

        </div>
    </div>
```

Code CSS de titres et paragraphe :

```
h1{
    font-size:22px;
    text-align:center;
    margin-top:29px;
    color:#028;
    }

h2{
    font-size:18px;
    margin-top:50px;
    margin-bottom:25px;
    color:#08A;
    }

p{
color:#223;
}
```

Codes HTML et CSS liés à la page video :

```
<div id="page2">
    <div class ="central">

        <div class="contenu">
            <h2> Ci-dessous une petite video </h2>

            <video controls id="ma_video">
                <source src="video/css3_validator.mp4" type="video/mp4" />
                <source src="video/css3_validator.ogg" type="video/ogg" />
            </video>
        </div>
    </div>
</div>
```

```
#ma_video{
    display:block;
    min-width:300px;width: 80%;
    margin:auto;margin-top:40px;
    }
```

Code HTML lié à la page galerie :

```
<div id="page3">
    <div class ="central">
        <h1> Voici un exemple simple de galerie photos </h1>

        <div id="conteneur_images">
            <img id="image1" />
            <img id="image2" />
            <img id="image3" />
            <img id="image4" />
        </div>

        <div id="conteneur_minis">
            <img ONCLICK="clic_mini1()" id = "mini1" />
            <img ONCLICK="clic_mini2()" id = "mini2" />
            <img ONCLICK="clic_mini3()" id = "mini3" />
            <img ONCLICK="clic_mini4()" id = "mini4" />
        </div>
    </div>

</div>
```

Code CSS lié à la galerie :

```
#conteneur_minis{
    position:relative;
    width:280px; /* 4 * 70px par mini */
    height:50px;
    margin:auto;
}
```

```
#mini1, #mini2, #mini3, #mini4 {
width:64px;      /* 70 px par mini avec espaces */
height:43px;
position:absolute;
top:7px;
}

#mini1{
left:3px; /* 3px+64px+3px -> 70px */
background-image:url("images/mini_bateau.jpg");
}
#mini2{
left:73px;
background-image:url("images/mini_maison.jpg");
}
#mini3{
left:143px;
background-image:url("images/mini_oiseaux.jpg");
}
#mini4{
left:213px;
background-image:url("images/mini_plage.jpg");
}
```

Code Javascript lié à la galerie :

```
function clic_mini1()
    {
    document.getElementById("image1").style.opacity = "1";
    document.getElementById("image2").style.opacity = "0";
    document.getElementById("image3").style.opacity = "0";
    document.getElementById("image4").style.opacity = "0";
    }

function clic_mini2()
    {
    document.getElementById("image1").style.opacity = "0";
    document.getElementById("image2").style.opacity = "1";
    document.getElementById("image3").style.opacity = "0";
    document.getElementById("image4").style.opacity = "0";
    }
function clic_mini3()
    {
    document.getElementById("image1").style.op
    document.getElementById("image2").style.op
    document.getElementById("image3").style.op
    document.getElementById("image4").style.op
    }

function clic_mini4()
    {
    document.getElementById("image1").style.op
    document.getElementById("image2").style.op
    document.getElementById("image3").style.op
    document.getElementById("image4").style.op
    }
```

Manipulation 4 : Site complet amélioré

Objectif général

- Renforcer les notions vues précédemment et consolider votre site (manipulation 3)
- Ajouter des apports personnels qui permettront d'améliorer votre site
- Mettre en ligne le site finalisé
- Rédiger un rapport explicatif

Durée de la manipulation

- Consignes et théorie : 2 h
- Travail en classe : 8 h
- Travail à la maison : 10 h environ

Notions associées

En plus des notions vues précédemment, les nouvelles notions que vous mettrez en œuvre seront issues de recherches ou de tests personnels.
L'idée est de mettre en œuvre des nouvelles notions ou des nouveaux outils qui n'ont pas été vus au cours et qui améliorent votre site.

Objectifs spécifiques

A. CONSOLIDER VOTRE SITE :

Tout comme pour la manipulation précédente, testez votre site sur différents navigateurs et arrangez-vous pour qu'il soit fonctionnel, fluide, bien structuré et esthétique.
Prévoyez également que votre site convienne pour différentes tailles d'écrans.

B. AMELIORER VOTRE SITE GRACE A DES APPORTS PERSONNELS

Dans l'optique de personnaliser et d'améliorer votre site, faites des recherches et des tests sur des notions qui n'ont pas été étudiées en classe. Ci-dessous quelques pistes :

- Nouvelles propriétés CSS3
- Tweenmax
- Utilisation d'un CMS (Wordpress, Joomla,…)
- JQuery

C. REDIGER UN RAPPORT EXPLICATIF

Vous devez rédiger un rapport consistant expliquant votre site pour quatre parties différentes :

- Les éléments principaux de mise en page et de positionnement
- Les éléments esthétiques primordiaux
- L'aspect dynamique du site (navigation, galerie, transitions,…)
- Les apports personnels

Si les explications sont consistantes et concises, chaque partie peut contenir environ une page.

Théorie partie 4

Il n'y a pas réellement de théorie pour cette partie dans la mesure où l'idée est d'encourager les apports personnels.

Quelques pistes avaient été mentionnées, elles sont reprises ci-dessous :

- Nouvelles propriétés CSS3
- Tweenmax
- Utilisation d'un CMS (Wordpress, Joomla,…)
- JQuery

Nouvelles propriétés CSS3

Il existe en CSS3 des propriétés qui permettent d'avoir des rendus intéressants. Notamment les propriétés de transformation de blocs qui peuvent ajouter des effets originaux lorsqu'on les accompagne de transitions.

Tweenmax

TweenMax est une librairie qui facilite l'écriture du code pour mettre en œuvre des fonctionnalités gérées en javascript.

L'idée est d'ajouter un fichier tout fait à notre arborescence pour pouvoir utiliser les fonctionnalités qui s'y trouvent.
Avec TweenMax on ajoutera à notre dossier racine le fichier « TweenMax.js ».

Vous trouverez ci-dessous un aperçu (partiel et non commenté) d'utilisation de TweenMax :

Code index.html

Code transition_menus.js

```
<!doctype html>
<html>
<head>
<meta charset="UTF-8">
<title>Exemple de menu</title>
<link rel="stylesheet" href="style.css"/>
<script type="text/javascript" src="TweenMax.js"></script>
</head>

<body>

<nav>
<h6 id="lien1"> PAGE1 </h6>
<h6 id="lien2"> PAGE2 </h6>
<h6 id="lien3"> PAGE3 </h6>
</nav>

<div id="page1"></div>
<div id="page2"></div>
<div id="page3"></div>
<script type="text/javascript" src="transitions_menus.js"></script>
</body>
</html>
```

```
lien_page1= document.getElementById("lien1");
lien_page2= document.getElementById("lien2");
lien_page3 = document.getElementById("lien3");

zone_page1 = document.getElementById("page1")
zone_page2 = document.getElementById("page2")
zone_page3 = document.getElementById("page3")

lien_page1.onclick = function ()
    {
    TweenMax.to ( zone_page1,1,{top:"00%"});
    TweenMax.to ( zone_page2,1,{top:"100%"});
    TweenMax.to ( zone_page3,1,{top:"200%"});
    }

lien_page2.onclick = function ()
    {
    TweenMax.to ( zone_page1,1,{top:"-100%"});
    TweenMax.to ( zone_page2,1,{top:"00%"});
    TweenMax.to ( zone_page3,1,{top:"100%"});
    }

lien_page3.onclick = function ()
    {
    TweenMax.to ( zone_page1,1,{top:"-200%"});
    TweenMax.to ( zone_page2,1,{top:"-100%"});
    TweenMax.to ( zone_page3,1,{top:"000%"});
    }
```

Utilisation d'un CMS

CMS signifie « Content Management System », ce qui se traduit par un « système de gestion du contenu ». Il s'agit dans notre cas d'un programme que l'on installe sur notre ordinateur qui va se charger de faciliter la conception et la mise à jour de sites Web.

Ce programme va alors générer du code au travers d'une interface de conception de site Web.
En général, le « CMS » orienté Web reposera en partie sur l'utilisation de bases de données et d'un langage de programmation orienté serveur (traditionnellement le PHP).
L'utilisation d'un tel programme implique généralement l'installation et le paramétrage de composants supplémentaires (gestion de base de données, simulateur de serveur local,…).

Deux « CMS » orientés Web très populaires sont Wordpress et Joomla ; ils sont également gratuits.

JQuery

Tout comme « TweenMax », JQuery est une librairie qui facilite l'écriture du code pour mettre en œuvre des fonctionnalités gérées en javascript.

Questions de Synthèse

Questions élémentaires

- Qu'appelle-t-on le dossier racine ?
- Qu'est-ce qu'un navigateur ?
- Qu'appelle-t-on « URL » ?
- A quoi sert la balise <html> ?

- A quoi sert la balise <head> ?
- A quoi sert la balise <body> ?
- Comment assurer le lien entre l'HTLM et le CSS ?
- Comment fait-on un saut de ligne en HTML ?
- Qu'est-ce qu'un sélecteur CSS ?
- Comment change-t-on la taille de la police ?
- Comment change-t-on la couleur de la police ?
- Comment change-t-on la police de caractère ?
- Comment change-t-on la couleur de fond d'un bloc ?
- Comment fixe-t-on la largeur d'un bloc ?
- Comment fixe-t-on la hauteur d'un bloc ?
- Comment détecte-t-on le survol d'un élément par la souris ?

Questions simples / intermédiaires

- Quelles sont les recommandations vues pour le choix des noms de fichiers/dossiers ?
- Où doit-on insérer le code pour mettre un titre ?
- Le titre d'une page HTML est-il affiché ? Si oui, où ?
- Qu'est-ce que l'indentation ? A quoi ça sert ?
- Qu'est-ce qu'une balise « DIV » ?
- Quelle est la différence entre « class » et « id » ?
- Qu'appelle-t-on une balise orpheline ?
- Que peut-on imaginer comme alternative à la balise
 ?
- Décrivez brièvement la syntaxe de base CSS.
- Comment centre-t-on du texte au sein d'un bloc ?
- Qu'est-ce que l'indentation ? A quoi ça sert ?
- Comment fonctionne la propriété « z-index » ?
- Comment insérer une image de fond à un bloc ?
- Qu'est-ce qu'un gestionnaire FTP ?
- Comment centrer un bloc horizontalement au sein d'un autre bloc ?
- Comment peut-on insérer une police personnalisée ?
- Que signifie « display : inline-block ;» ?
- Comment fonctionne le positionnement absolu de base ?
- Qu'est-ce qui différencie un élément de type « bloc » et un élément de type « en-ligne » ?
- Comment gérer des marges extérieures à un bloc ?
- Comment gérer des marges intérieures à un bloc ?
- Décrivez brièvement le modèle de la boîte.
- Comment insère-t-on une vidéo ?
- A quoi sert la propriété « text-transform » ?
- A quoi sert la propriété « text-align » ?
- A quoi sert la propriété « font-weight » ?
- Comment gérer les espaces entre les lettres des mots ?
- Décrivez brièvement la propriété « transition ».

Questions intermédiaires / avancées

- Comment centrer un bloc verticalement au sein d'un autre bloc ?
- A quoi peut servir l'ajout de la propriété « position relative ; » ?
- Comment utiliser l'utilitaire « @font-face » ?
- Pourquoi est-il parfois plus prudent de tabler sur l'utilisation du CSS 2.1 ?
- Qu'appelle-t-on « CMS » pour le développement web ?
- Citez un exemple de metadonnée.
- A quoi sert le paramétrage du « charset » ?
- Décrivez brièvement suivant quoi le positionnement absolu peut être nuancé.
- Qu'est-ce que le mécanisme de fusion des marges ?
- Décrivez un truc qui permet d'éviter l'influence du « padding » sur la dimension d'un bloc.
- Comment fonctionne la propriété « display » ?
- Décrivez différentes philosophies de gestion de menus.
- Comment peut-on agir sur du CSS à l'aide de Javascript ?

Bibliographie / Webographie

Sites Web

	Petite description
https://openclassrooms.com	Cours gratuits et payants
http://www.w3schools.com	Cours / tutoriels gratuits
http://tutsplus.com/	Cours gratuits et payants
https://www.codecademy.com/fr	Tutoriels ludiques et gratuits
https://validator.w3.org	Validation HTML
http://www.css-validator.org	Validation CSS
http://www.awwwards.com/	Exemple de sites qui se sont démarqués
http://www.dafont.com/	Polices de caractères à télécharger
http://www.fontsquirrel.com/	Polices de caractères à télécharger
http://multimedia.inraci.be	Source d'informations liées au cours
http://infographie.inraci.be	Source d'informations supplémentaires

Quelques ouvrages

	Petite description
Apprenez à créer votre site Web avec HTML5 et CSS3 Format PDF – Edition 2012 – par Mathieu NEBRA	Notions de base bien expliquées et illustrées
HTML5 et CSS3 : Faites évoluer le design de vos sites web Editions ENI – Janvier 2012 – par Christophe AUBRY	Illustre un bel éventail de possibilités
CSS : Techniques professionnelles pour une mise en page moderne Editions Pearson – 2011 – par Eric A. MEYER	Notions plus avancées, surtout sur la mise en page
CSS3 : Le design web moderne Editions Dunod – 2012 – par V. DE OLIVEIRA et C. ESNAULT	Notions plus avancées sur de nombreux sujets
Cascading Style Sheets Level 2 Revision 1 (CSS 2.1) Specification Format PDF –W3C Recommandation 07 June 2011	Ouvrage consultatif, CSS2.1 en état de recommandation
CSS, précis et concis Editions O'REILLY – 2001 – par Eric A. MEYER	Ouvrage plutôt « mémo » pour consulter la syntaxe.