Chapitre 6
Tableaux

1. Tableaux à dimension unique

Nous avons entrevu dans le chapitre Développement à partir d'algorithmes le potentiel des tableaux à dimension unique et à dimensions multiples, voyons leur prise en compte sous JavaScript.

1.1 Syntaxe

En JavaScript, un tableau à dimension unique est une variable mémoire "composite" dans laquelle il va être possible de stocker plusieurs données indépendantes, y compris de types différents, avec une indexation de chacune des valeurs par un numéro (ou indice).

L'accès à chaque donnée du tableau se fera donc par l'intermédiaire de cette valeur d'indice.

Une particularité quant à cet indice, sa valeur pour la première cellule du tableau est 0.

Le langage JavaScript fournit plusieurs façons de créer un tableau :

– la syntaxe littérale,

– la syntaxe dite "Programmation orientée objet".

Avec une syntaxe littérale, la déclaration d'un tableau de nom `tabSemaine` de sept cellules contenant les libellés des jours d'une semaine se fait comme suit :

```
var tabSemaine = ["Lundi", "Mardi", "Mercredi", "Jeudi",
"Vendredi", "Samedi", "Dimanche"];
```

Vous noterez que la déclaration s'est accompagnée de l'initialisation de chacune des cellules du tableau `tabSemaine` (de la cellule d'indice 0 à la cellule d'indice 6).

Avec une syntaxe "Programmation orientée objet", vous auriez :

```
var tabSemaine = new Array("Lundi", "Mardi", "Mercredi", "Jeudi",
"Vendredi", "Samedi", "Dimanche");
```

Nous aurions pu déclarer le tableau `tabSemaine` sans lui affecter des valeurs. Des affectations ultérieures peuvent être envisagées, comme par exemple pour le Lundi :

```
tabSemaine[0] = "Lundi";
```

Ce qui est vraiment particulier dans la gestion des tableaux sous JavaScript est l'extrême souplesse autorisée :

– pas de dimensionnement a priori (il est toujours possible d'étendre la taille du tableau en fonction des besoins),
– possibilité de mélanger dans un même tableau des données de types différents,
– possibilité d'utiliser des tableaux associatifs (tableaux pour lesquels les indices sont remplacés par des valeurs textuelles).

Pour accéder dans un traitement au contenu d'une valeur de tableau rangée à une position d'indice particulière, la syntaxe sera :

```
document.write("Le 4ième jour de la semaine est " + tabSemaine[3]);
```

Remarque

Il faut toujours se rappeler que la numérotation des indices débute à zéro.

Enfin, sachez que JavaScript propose une multitude de méthodes s'appliquant sur les tableaux (`Array`). Vous pourrez facilement par ces méthodes insérer, supprimer, repérer des éléments d'un tableau. Il existe même des méthodes de tri (`sort`, `reverse`) pour classer facilement les valeurs contenues dans un tableau sans avoir recours à l'écriture fastidieuse d'un algorithme de tri.

1.2 Exercice n°14 : Décompte des nombres pairs dans un tableau

Sujet

Détermination du nombre de nombres pairs dans un tableau (saisie préalable des valeurs à prévoir au clavier).

Corrigé (partiel) en JavaScript

```
/* Déclaration de variables locales */
/*
i          : Compteur de boucle
nbPairs    : Cumul du nombre de nombres pairs
tableau    : Tableau des nombres
*/
var i, nb_pairs;
var tableau = new Array;

/* Initialisations */
nbPairs = 0;
for (i=1; i<=5; i++)
{
    tableau[i] = parseInt(prompt("tableau[" + i +"] : "));
}

/* Détermination du nombre de nombres pairs dans le tableau */
for (i=1; i<=5; i++)
{
    if (tableau[i]%2 == 0)
    {
        nbPairs = nbPairs + 1;
    }
}

/* Affichage du résultat */
document.write("Le tableau contient " + nbPairs + " nombres pairs");
```

Commentaires du code JavaScript

Rien de vraiment nouveau n'est présenté dans ce script hormis le calcul du modulo. Ce calcul sert ici à déterminer la parité de chaque contenu de cellules du tableau. Il est réalisé par l'intermédiaire l'opérateur `%`.

Vous aurez peut-être noté que dans ce script la cellule d'indice 0 n'a pas été utilisée (la numérotation par la boucle `for` débute à 1). Ce choix rend sans doute plus compréhensible l'algorithme (il n'y a que les informaticiens qui s'accommodent de la numérotation à partir de zéro !).

2. Tableaux à dimensions multiples

Il est fréquent que l'on ait besoin de tableau à dimensions multiples pour gérer des problématiques, notamment en mathématique, en statistique...

JavaScript offre cette possibilité.

2.1 Syntaxe

Comme pour les tableaux à dimension unique, JavaScript permet de déclarer les tableaux à dimensions multiples de plusieurs façons :

– avec une syntaxe littérale,

– avec une syntaxe dite "Programmation orientée objet".

Avec une syntaxe dite "Programmation orientée objet" (encore appelée JSON - *JavaScript Object Notation*), la déclaration d'un tableau de nom `tabMatrice` de deux lignes subdivisées en quatre colonnes avec initialisation se fait comme suit :

```
/* Déclaration du tableau tabMatrice */
var tabMatrice tableau = new Array();

/* Déclaration de la première "ligne" du tableau tabMatrice */
tabMatrice[0]=new Array()

/* Initialisation des 4 "colonnes" de la première "ligne" */
tabMatrice[0][0] = "Un";
```

```
tabMatrice[0][1] = "Deux";
tabMatrice[0][2] = "Trois";
tabMatrice[0][3] = "Quatre";

/* Déclaration de la deuxième "ligne" du tableau tabMatrice */
tabMatrice[1]=new Array()

/* Initialisation des 4 "colonnes" de la deuxième "ligne" */
tabMatrice[1][0] = "Onze";
tabMatrice[1][1] = "Douze";
tabMatrice[1][2] = "Treize";
tabMatrice[1][3] = "Quatorze";
```

2.2 Exercice n°15 : Mini-tableur

Sujet

Soit le tableau `tb` à deux dimensions comportant quatre lignes et cinq colonnes. Réaliser les traitements suivants :

– saisir au clavier des valeurs dans les trois premières lignes et les quatre premières colonnes (on conserve la dernière ligne et la dernière colonne libres pour des additions de lignes et de colonnes),
– additionner les colonnes en dernière ligne et les lignes en dernière colonne.

Corrigé (partiel) en JavaScript

```
 /* Déclaration de variables locales */
var tb = new Array(5);
var numLigne, numColonne;
var valeur;

/* Déclaration de 5 "colonnes" par "ligne" pour le tableau tb */
for (var numLigne=1; numLigne<tb.length; numLigne++)
{
    /* Création des "colonnes" (numérotées de 0 à 5) */
    tb[numLigne]=new Array(6);
}

/* Initialisation du tableau tb (3 lignes * 4 colonnes) par une saisie clavier */
for (numLigne = 1; numLigne <= 3; numLigne++) {
    for (numColonne = 1; numColonne <= 4; numColonne++) {
        valeur = parseInt(prompt("tableau[" + numLigne + "][" + numColonne + "] = "));
        tb[numLigne][numColonne] = valeur;
    }
}

/* Mise à zéro des totaux en ligne n°4 */
```

```
for (numColonne=1; numColonne<=5; numColonne++)
{
    tb[4][numColonne] = 0;
}

/* Mise à zéro des totaux en colonne n°5 */
for (numLigne=1; numLigne<=4; numLigne++)
{
    tb[numLigne][5] = 0;
}

/* Détermination des totaux en ligne n°4 et en colonne n°5 */
for (numLigne=1; numLigne<=3; numLigne++)
{
    for (numColonne=1; numColonne<=4; numColonne++)
    {
        /* Totalisation en ligne n°4 */
        tb[4][numColonne] = tb[4][numColonne]
        + tb[numLigne][numColonne];
        /* Totalisation en colonne n°5 */
        tb[numLigne][5] = tb[numLigne][5]
        + tb[numLigne][numColonne];
        /* Totalisation générale en ligne n°4-colonne n°5 */
        tb[4][5] = tb[4][5] + tb[numLigne][numColonne];
    }
}

/* Affichage du total général */
/* NB : Total de 78 étant donné la technique de remplissage retenue
du tableau tb */
document.write("Total général en tb[4][5] = " + tb[4][5]);
```

Commentaires du code JavaScript

Le tableau `tb` est dans un premier temps déclaré comme étant un tableau à une seule dimension avec cinq cellules, implicitement numérotées de 0 à 4 comme suit :

```
var tb = new Array(5);
```

Vous noterez que la ligne de numéro zéro ne sera pas utilisée par la suite. Ce choix a été fait car l'utilisation ultérieure de cette ligne ne serait pas très intuitive.

Dans un second temps les lignes de ce tableau sont elles-mêmes subdivisées en colonnes, comme ceci :

```
/* Déclaration de 5 "colonnes" par "ligne" pour le tableau tb */
for (var numLigne=1; numLigne<tb.length; numLigne++)
{
    /* Création des "colonnes" (numérotées de 0 à 5) */
    tb[numLigne]=new Array(6);
}
```

Chapitre 3-2
Définir les styles CSS

1. La structure d'une règle de style

1.1 La terminologie des CSS

Nous parlerons de CSS, de style, de règle, de déclaration, de propriété et de valeur. Il convient de bien définir cette terminologie afin d'utiliser les bons termes.

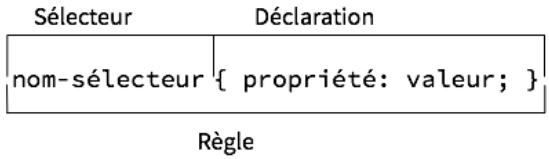
Les **CSS**, pour **Cascading Style Sheets**, sont une technologie développée par le W3C qui permet de mettre en forme et de mettre en page les pages des sites web structurés en HTML.

Un **style** est une mise en forme, ou une mise en page utilisant les CSS, qui est mémorisée et qui, ensuite, peut être appliquée à un ou plusieurs éléments HTML.

Une **règle** CSS permet la création d'un style. Cette règle est créée avec une syntaxe précise utilisant un **sélecteur** et une **déclaration**. Cette dernière est constituée de **propriétés** et de **valeurs**.

1.2 Définir une règle de style

Un style CSS est bâti avec une règle. Cette règle est constituée de plusieurs parties. Voici un petit schéma illustrant une règle CSS :



– La **règle** est constituée d'un sélecteur et d'une déclaration.

– Le **sélecteur** indique la portée du style, c'est-à-dire sur quel élément HTML peut s'appliquer le style créé. Il existe beaucoup de sélecteurs, nous les étudierons dans un prochain chapitre.

– La **déclaration** est indiquée entre accolades.

  C'est dans cette déclaration que sont indiquées la ou les **propriétés** CSS utilisées. Chaque propriété utilise une ou plusieurs **valeurs**. La propriété est séparée de la ou des valeurs par le caractère deux-points **:**. Chaque ligne dans la déclaration se termine par le caractère point-virgule **;**.

Pour plus de visibilité et de lisibilité, il est d'usage d'aller à la ligne après l'accolade ouvrante et avant l'accolade fermante. Ainsi chaque couple propriété/valeur se trouve sur une seule ligne.

Voici une règle CSS n'utilisant aucun espace ni retour à la ligne. La visibilité n'est pas au rendez-vous :

```
nom-selecteur{propriete1:valeur1;propriete2:valeur2;propriete3: valeur3;}
```

Voici une règle CSS utilisant plusieurs lignes dans sa déclaration :

```
nom-selecteur {
   propriete1: valeur1 ;
   propriete2: valeur2 ;
   propriete3: valeur3 ;
}
```

Les "puristes" pourront utiliser cette syntaxe :

```
nom-selecteur
   {
       propriete1: valeur1 ;
       propriete2: valeur2 ;
       propriete3: valeur3 ;
   }
```

C'est à vous d'organiser au mieux la structure des règles CSS pour avoir la meilleure lecture possible, tant que la syntaxe est respectée.

1.3 Les règles de nommage

Le nom des sélecteurs doit respecter quelques règles :

– Il ne doit pas commencer par un chiffre.

– Il ne doit pas contenir d'espace, de caractères accentués et de caractères spéciaux, comme : +, *, /…

– Il peut contenir les caractères tiret – et tiret bas (ou *underscore*) _, n'importe où dans le nom.

– Il est sensible à la casse. Si le nom du sélecteur est dans la règle, `.Mon-Selecteur`, son application dans le code HTML doit être identique : `class="Mon-Selecteur"`, et pas `class="mon-Selecteur"` ou `class="Mon-selecteur"`.

L'exception de la sensibilité à la casse est pour les sélecteurs de type qui utilisent le nom d'un élément HTML. En effet, vous le savez, le langage HTML est insensible à la casse. Pour cette raison de nombreux designers n'utilisent que des minuscules pour éviter toute ambiguïté entre les CSS et le HTML.

2. Les unités de mesure

2.1 L'utilisation des unités de mesure

De très nombreuses propriétés utilisent des valeurs numériques. Citons par exemple la taille des caractères, les dimensions des blocs participant à la mise en page, la largeur des écrans de diffusion… Les unités de mesure sont une spécification du W3C qui est presque achevée puisqu'elle possède le statut de **Candidate Recommendation** au 14 août 2018 (en novembre 2018, au moment de la rédaction de ce livre).
Voici son URL : https://www.w3.org/TR/css-values-3/

Notez bien qu'il n'est ni nécessaire ni utile d'indiquer une unité si la valeur est égale à `0`.

Voici un exemple inutile, mais valide :

```
.nav-bar {
   margin: 0px ;
}
```

Voici un exemple correct :

```
.nav-bar {
   margin: 0 ;
}
```

2.2 Les valeurs initiale et héritée

De nombreuses propriétés CSS peuvent utiliser trois valeurs textuelles :

– `initial` indique aux navigateurs qu'il faut utiliser la valeur par défaut de la propriété.
– `inherit` informe que la valeur calculée à utiliser est celle de l'élément HTML parent de l'élément concerné.
– `unset` spécifie que la valeur est `initial` ou `inherit`, selon que la propriété est héritée ou non.

2.3 Les valeurs numériques

Des propriétés peuvent utiliser des valeurs numériques de type entier (nommé *integer* dans les propriétés CSS) ou des valeurs décimales (nommé *number* dans les propriétés CSS). Attention, le séparateur décimal est le point `.` et pas la virgule `,`. Cette valeur est correcte : `1.2`, et celle-ci est incorrecte : `1,2`.

D'autres propriétés utilisent des pourcentages. L'unité est le classique `%` qui n'utilise pas d'espace après le nombre. Cette valeur est correcte : `25%` et celle-ci est incorrecte : `25 %`.

2.4 Les unités de longueur

De très nombreuses propriétés CSS utilisent des valeurs de longueur, nommées *length* dans les spécifications du W3C. Vous avez à votre disposition deux grands types de valeurs : les valeurs relatives et les valeurs absolues.

Les unités des valeurs relatives peuvent être :

- `em` pour indiquer la taille des caractères par rapport à l'élément parent.
- `rem` spécifie la taille des caractères dans l'élément racine `<html>` de la page web.
- `ex` donne la taille des caractères par rapport à la hauteur de la minuscule du caractère `x`.
- `vw` attribue la valeur proportionnellement à la largeur de l'écran du média de sortie (*viewport* en anglais).
- `vh` attribue la valeur proportionnellement à la hauteur de l'écran du média de sortie (*viewport* en anglais).

Les unités des valeurs absolues peuvent être :

- `cm` pour des centimètres.
- `mm` pour des millimètres.
- `in` pour des inches (1 inche = 2,54 cm).
- `pc` pour des picas (1 pica = 1/16 in).
- `pt` pour des points (1 point = 1/72 in).

– `px` pour des pixels (1 pixel = 1/96 in).

Pour les unités des angles, nous avons :

– `deg` pour les degrés.
– `grad` pour les gradients.
– `rad` pour les radiants.
– `turn` pour 1 tour complet, soit 360°.

Pour les unités de temps, nous avons :

– `s` pour les secondes.
– `ms` pour les millisecondes.

2.5 Les valeurs calculées

Certaines mises en forme ou mises en page peuvent dépendre de calculs où interviennent la largeur de l'écran de diffusion, la taille de caractère d'un élément parent… Pour effectuer des calculs, le W3C met à disposition l'expression `calc()`.

Voici un exemple simple dans lequel la largeur de l'élément HTML `<section>` est calculée à partir de trois valeurs :

```
section {
   float: left;
   margin: 1em;
   border: solid 1px;
   width: calc(100%/3 - 2*1em - 2*1px);
}
```

Voici un deuxième exemple d'une taille de caractère calculée proportionnellement à la largeur du viewport :

```
.texte {
   font-size: calc(100vw / 40);
}
```