

```
<!DOCTYPE html>
<html lang="fr">
```

```
<!--
    HTML5 - CSS3 - JAVASCRIPT

    source      : FicheAscii.html
    Copyright   : Guy REBOULET
    date        : 2013
    Version     : 1

    Affiche la tables des caractères
    ASCII gérés par votre ordinateur
```

```
-->
```

```
<head>
    <title>Formation HTML5-CSS3-JAVASCRIPT</title>

    <script>
```

```
    const GAUCHE   = 0;
    const DROITE   = 1;

    /*
        StrBinaire() permet d'afficher le résultat binaire
        d'une valeur transmise en argument en fonction
        d'un nombre de digits souhaités.

        Le traitement avec les opérateurs binaires
        procure un gain de temps machine par rapport à des
        divisions successives

        en entrée :
            i_Val      => Valeur à convertir
            i_NbrBits  => Nbre de digits à traiter
        en sortie
            Chaîne représentant la valeur binaire
            de i_Val

    */

    function StrBinaire( i_Val, i_NbrBits )
    {
        var l_Mask;
        var l_Ret=" ";

        for ( l_Mask = 1 << (i_NbrBits - 1); l_Mask; l_Mask >>=1 )
            l_Ret += ( i_Val & l_Mask ) ? '1' : '0' ;
        return l_Ret;
    }

    /*
        StrPadding() - Permet de compléter une valeur par une
```

```

        suite de caractères à droite ou à gauche
        sur une longueur suivant une directive.

    en entrée :
        i_Val    => Valeur à padder
        i_Car    => Caractère à appliquer
        i_Taille => Taille de la chaine à construire
        i_Sens   => Directive qui donne le sens de
                    la concatenation DROITE ou GAUCHE.

    en sortie :
        Chaîne de caractères résultat de la concaténation.

*/

function StrPadding ( i_Val, i_Car, i_Taille, i_Sens )
{
    var l_Val=" ";
    var l_Car=" ";
    var l_i;
    l_Val = i_Val.toString();
    l_Car = i_Car.toString();
    // Concaténe i_Val avec i_Car suivant une directive à gauche ou à droite
    for ( l_i=1; l_Val.length < i_Taille; l_i++)
        (i_Sens == DROITE)    ? (l_Val=l_Val+l_Car)
                               : (l_Val=l_Car+l_Val);

    return l_Val;
}

/*
AfficheAscii() - Affiche suivant différents
formats les caractères ASCII

    en entrée :
        i_Debut  => Valeur debut du traitement
        i_Fin    => Valeur Fin du traitement
    en sortie :
        Chaîne résultat de la composition

*/

function AfficheAscii( i_Debut, i_Fin )
{
    var l_i;
    var l_Ascii;
    var l_Binaire;
    var l_NumEntite;
    var l_Ret=" ";

    for ( l_i =i_Debut; l_i <= i_Fin; l_i++)
    {
        l_Ascii = String.fromCharCode( l_i );
        l_Binaire = StrBinaire( l_i, 8 );
        l_NumEntite=StrPadding('&#38;&#35;' + l_i, ' ', 13, GAUCHE);
        l_Ret += StrPadding ( l_i, '0', 3, GAUCHE )
                    + ' - ' + l_Binaire
                    + ' - ' + l_NumEntite
                    + ' - ' + l_Ascii + '\n' ;
    }
}

```

```
        return l_Ret;  
    }
```

```
document.writeln("<pre>");  
document.writeln( AfficheAscii( 32, 55));  
document.writeln("</pre>");
```

```
</script>
```

```
</head>
```

```
<body>
```

```
</body>
```

```
</html>
```