

# Apprentissage du modèle DOM

Dans une série d'exemples, voyons comment il est possible depuis JavaScript d'écrire du texte (puis des éléments DOM plus complexes) dans un document HTML via JavaScript.

## 1. Script "Hello World!"

Dans ce premier script, voyons comment afficher une chaîne de caractères "Hello World!" dans le corps du document HTML en y ajoutant aussi des balises de mise en forme (gras).

Le script n'est pas écrit ci-après dans son intégralité, seule la séquence JavaScript concernée, intégrée dans la section HTML <body> du script, est reproduite :

```
<!-- Début script JavaScript -->
<script type="text/JavaScript">

    /* Affichage du nom du script */
    alert("DOM_01");

    /* Affichage Hello World! par la méthode document.write */
    document.write("Hello World!<br />");

    /* Affichage Hello World! par la méthode document.write */
    /* avec mise en gras du mot World */
    document.write("Hello <b>World</b>!");

</script>
```

À l'exécution après l'affichage, par la méthode `alert`, d'une boîte de dialogue annonçant l'intitulé de l'exercice (DOM\_01), nous obtenons tout naturellement l'affichage suivant dans la section `body` :



Sur cet exemple, vous voyez la possibilité d'intégrer dans la séquence à afficher des balises de mise en forme comme `<b> ... </b>` (mise en gras). En réalité toutes les balises et attributs de balises HTML sont intégrables par la méthode `document.write`. Dans l'absolu, l'intégralité du script HTML pourrait de ce fait être programmée via JavaScript.

## 2. Différence entre `write` et `writeln`

Dans ce petit exemple, voyons la différence entre les méthodes `write` et `writeln`.

```
<!-- Début script JavaScript -->
<script type="text/JavaScript">

    /* Affichage du nom du script */
    alert("DOM_02");

    /* Mise en évidence de la différence
    entre document.write et document.writeln */
    /* NB1 : Avec la méthode writeln le retour chariot
    en fin de ligne est implicite */
    /* NB2 : La méthode writeln requiert l'encadrement
    de la séquence par un jeu de <pre> ... </pre> */
    document.write("<pre>");
    document.write("Ligne n°1<br />");
    document.write("Ligne n°2<br />");
    document.write("Ligne n°3<br /><br />");
    document.writeln("Ligne n°4");
    document.writeln("Ligne n°5");
    document.write("Ligne n°6");
    document.write("</pre>");

</script>
```

L'affichage obtenu dans la section `body` à l'exécution est :



Avec la méthode `write` pour obtenir un passage à la ligne suivante, il faut explicitement intégrer une balise `<br />` (break) en fin de message à écrire. Cette contrainte n'est pas nécessaire pour la méthode `writeln`.

Vous noterez aussi que la méthode `writeln` requiert l'encadrement de la séquence par un jeu de `<pre> ... </pre>`.

### 3. Gestion des liens hypertextes

Dans ce nouvel exemple, apprenons à gérer les liens hypertextes. La séquence de code reproduite est intégrée une

fois de plus dans la section HTML <body> du script :

```
<!-- Définition de plusieurs liens "hypertexte" consécutifs
dans le document -->
Moteur de recherche <a href="http://www.google.fr"
id="google">Google</a><br />
Site d'informations <a href="http://www.yahoo.fr" id="yahoo">Yahoo</a><br />
Encyclopédie <a href="http://www.wikipedia.fr"
id="wikipedia">Wikipedia</a>

<!-- Début script JavaScript -->
<script type="text/JavaScript">

    /* Saut de lignes */
    document.write("<br /><br />");

    /* Affichage du nombre de lien(s) 'hypertexte' du document */
    document.write("Nombre de lien(s) 'hypertexte' du document : "
+ document.links.length + "<br />");

    /* Affichage de l'identifiant du 2ème lien 'hypertexte' du document */
    document.write("Identifiant du 2ème lien 'hypertexte' du document : "
+ document.links[1].id + "<br />");

    /* Affichage de l'URL du 3ème lien 'hypertexte' du document */
    document.write("URL du 3ème lien 'hypertexte' du document : "
+ document.links[2].href);

</script>
```

Trois liens hypertextes (google, yahoo, wikipedia) sont tout d'abord programmés via des balises HTML <a href> ... </a>.

Ensuite le nombre de liens du document, l'identifiant du deuxième lien et l'URL du troisième lien sont affichés :



## 4. Gestion des images

Au travers d'un exemple, apprenons à gérer les images. La séquence de code reproduite est toujours intégrée dans la section HTML `<body>` du script :

```
<!-- Définition de plusieurs images consécutives dans le document -->

</img><br /><br />

</img><br /><br />

</img><br /><br />

</img><br /><br />

<!-- Début script JavaScript -->
<script type="text/Javascript">

  /* Affichage du nombre d'image(s) du document */
  document.write("Nombre d'image(s) du document : "
+ document.images.length + "<br />");

  /* Affichage de l'identifiant de la 1ère image */
  document.write("Identifiant de la 1ère image : "
+ document.images[0].id + "<br />");

  /* Affichage de l'attribut border de la 2ème image */
  document.write("Attribut border de la 2ème image : "
+ document.images[1].border + "<br />");

  /* Affichage de l'attribut src de la 3ème image */
  document.write("Attribut src de la 3ème image : "
+ document.images[2].src + "<br />");

  /* Affichage des attributs width et height de la 4ème image */
  document.write("Attribut width de la 4ème image : "
+ document.images[3].width + "<br />");
  document.write("Attribut height de la 4ème image : "
+ document.images[3].height + "<br />");

</script>
```

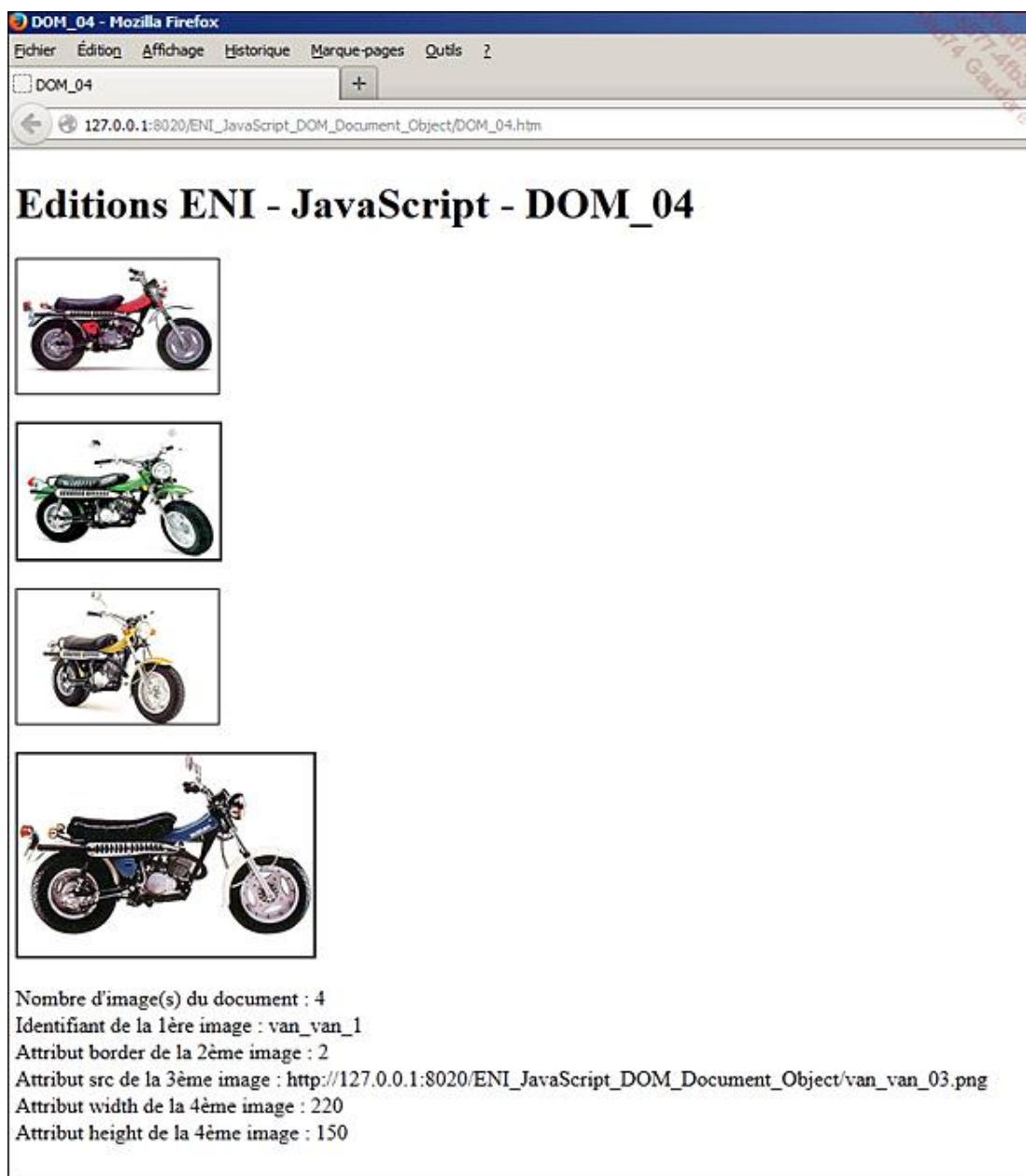
Quatre images sont tout d'abord présentées via des balises HTML <img> avec certains attributs (id, border, src, width, height).

Ensuite sont prévus différents affichages concernant ces images :

- le nombre d'images du document ;
- les attributs des images.

➤ La numérotation des images s'effectue de 0 à n-1.

L'exécution du script donne :



## 5. Gestion des formulaires et de leurs balises

Dans cet exemple, voyons comment gérer les formulaires et les balises intégrées dans ces formulaires. Pour limiter les explications, seules des balises HTML `input` seront placées dans les différents formulaires.

Le code (partiel) du script est présenté ci-après :

```
<!-- Définition de plusieurs formulaires consécutifs dans le document -->
<b>Formulaire n°1 :</b>
<form name="form1" id="idForm1" method="post">
    Champ 1 : <input type="text" id="champ1" size="20"><br />
    Champ 2 : <input type="text" id="champ1" size="20"><br />
</form>
<br />
<b>Formulaire n°2 :</b>
<form name="form2" id="idForm2">
    Champ 3 : <input type="text" id="champ3" size="20"><br />
    Champ 4 : <input type="text" id="champ4" size="20"><br />
</form>
<br />
<b>Formulaire n°3 :</b>
<form name="form3" id="idForm3">
    Champ 5 : <input type="text" id="champ5" size="20"><br />
    Champ 6 : <input type="text" id="champ6" size="20"><br />
</form>
<br /><br />

<!-- Début script JavaScript -->
<script type="text/JavaScript">

    /* Affichage du nombre de formulaire(s) du document */
    document.write("Nombre de formulaire(s) du document : "
+ document.forms.length + "<br />");

    /* Affichage du nom du 1er formulaire */
    document.write("Nom du 1er formulaire : "
+ document.forms[0].name + "<br />");

    /* Affichage de l'identifiant du 2ème formulaire */
    document.write("Identifiant du 2ème formulaire : "
+ document.forms[1].id + "<br />");

</script>
```

Dans ce script sont prévus différents affichages concernant les formulaires et les balises `input` intégrées dans ceux-ci :

- Nombre de formulaire(s) du document :

```
document.write("Nombre de formulaire(s) du document : "
+ document.forms.length + "<br />");
```

- Nom du premier formulaire :

```
document.write("Nom du 1er formulaire : "  
+ document.forms[0].name + "<br />");
```

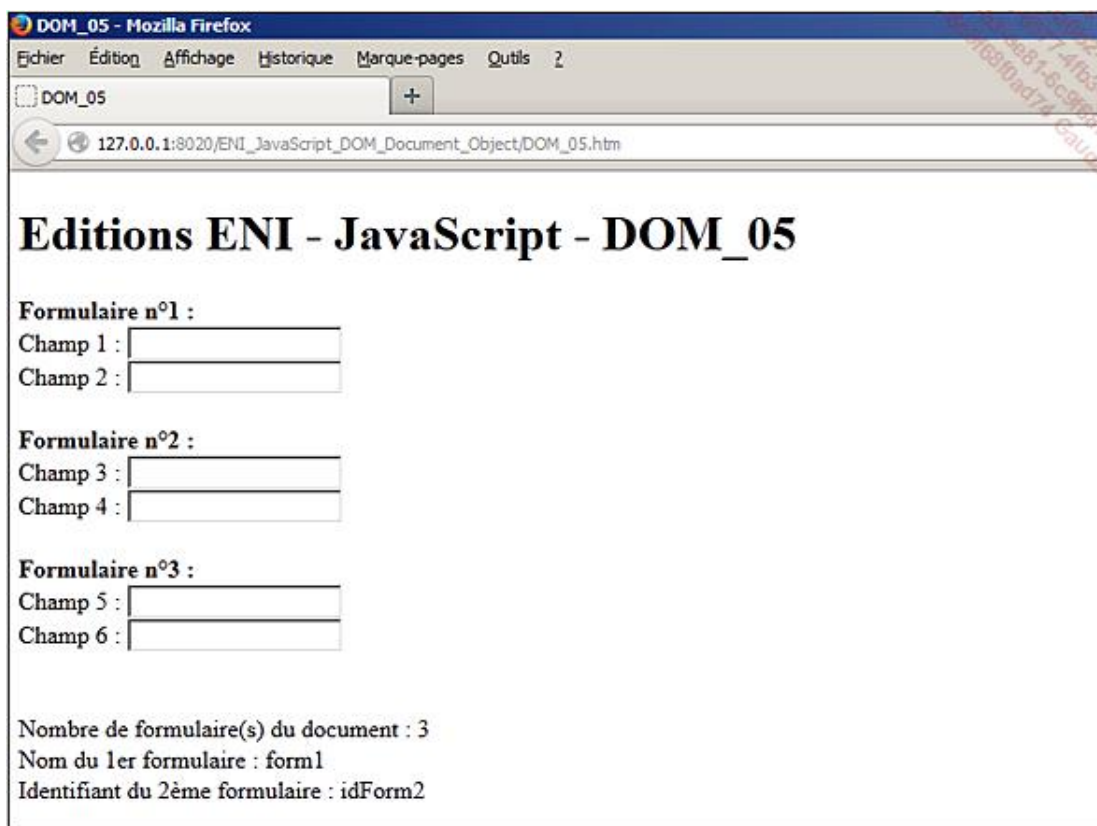
- Identifiant du deuxième formulaire :

```
document.write("Identifiant du 2ème formulaire : "  
+ document.forms[1].id + "<br />");
```

Le script ne présente pas de difficultés particulières. Prenez tout de même en compte les quelques remarques ci-après :

- La numérotation des formulaires s'effectue de 0 à n-1.
- La propriété length précise le nombre d'éléments de chaque collection (forms, input...).

L'exécution du script donne ceci :



## 6. Gestion des ancres

Voyons maintenant comment gérer les ancres au travers d'un petit exemple. Le code suivant (partiel) est placé dans la section HTML <body> de notre script :

```
<!-- Définition d'ancres (pouvant ensuite être atteintes par un href) -->  
<a name="html">Cours n°1 (HTML)</a><br />  
<a name="css">Cours n°2 (CSS)</a><br />  
<a name="xml">Cours n°3 (XML)</a><br />
```

```

<!-- Définition d'un lien hypertexte pointant sur google.fr (ne sera pas
comptabilisé comme une ancre) -->
Accès à un <a href="http://www.google.fr">complément de cours</a> (cf.
moteur de recherche google)

<!-- Début script JavaScript -->
<script type="text/JavaScript">

    /* Saut de lignes */
    document.write("<br /><br />");

    /* Affichage du nombre d'ancre(s) du document */
    document.write("Nombre d'ancre(s) du document : "
+ document.anchors.length + "<br />");

    /* Affichage de l'intitulé de la 1ère ancre du document */
    document.write("1ère ancre du document : "
+ document.anchors[0].text + "<br />");

    /* Affichage du nom de la 2ème ancre du document */
    document.write("2ème ancre du document : "
+ document.anchors[1].name);

</script>

```

Dans ce script trois ancres HTML sont programmées :

```

<a name="html">Cours n°1 (HTML)</a><br />
<a name="css">Cours n°2 (CSS)</a><br />
<a name="xml">Cours n°3 (XML)</a><br />

```

En règle générale, les liens hypertextes pointent vers des scripts externes (enchaînement entre pages d'un même site ou entre sites différents) avec une syntaxe `<a href="adresse absolue/adresse relative de la page à atteindre">intitulé du lien</a>`.

Dans notre script, il aurait aussi été possible de programmer des renvois vers les trois ancres (ceci n'a pas été prévu) avec une formulation du type `<a href="#nom_ancre">intitulé du lien</a>`, soit par exemple dans notre cas pour la première ancre :

```

<a href="#html">Accès à l'explication sur HTML</a>

```

Par contre un lien hypertexte pointant sur `http://www.google.fr` a été mis en place :

```

Accès à un <a href="http://www.google.fr">complément de cours</a>
(cf. moteur de recherche google)

```

Dans la section JavaScript de notre page, vous trouverez tout d'abord le décompte du nombre d'ancres de la page :



```
document.write("Nombre d'ancre(s) du document : "  
+ document.anchors.length + "<br />");
```

puis l'affichage de l'intitulé de la première ancre :

```
document.write("1ère ancre du document : "  
+ document.anchors[0].text + "<br />");
```

et du nom de la deuxième ancre :

```
document.write("2ème ancre du document : "  
+ document.anchors[1].name);
```

À l'exécution, l'affichage obtenu est :



➤ Le lien vers <http://www.google.fr> n'est pas comptabilisé dans les ancres.

## 7. Gestion de la navigation entre pages Web

Nous allons nous attacher à la problématique de la navigation entre les pages Web. Nous ne rentrerons pas dans les techniques les plus avancées de passage de paramètres entre pages et dans les mécanismes de mémorisation d'informations (persistance). À titre indicatif, vous trouverez plus loin dans ce livre un chapitre dédié à la gestion des cookies (cf. chapitre Gestion des cookies en JavaScript).

➤ Les cookies se présentent comme de petits fichiers texte qui peuvent entreposer un nombre limité de données. Cette technique peut se révéler précieuse pour conserver des contenus de variables mémoire réutilisables au fil de la navigation au travers des nombreuses pages de votre site Web (conservation d'identifiants, de mots de passe, de préférences utilisateur...).

Dans cadre de cet exemple, deux scripts vont être mis en œuvre.

Le code source (placé dans la section HTML <body>) du premier script est présenté ci-après :

```
<!-- Début script JavaScript -->
<script type="text/JavaScript">

    /* Affichage du nom du script */
    alert("DOM_07_1");

    /* Lien hypertexte donnant accès au document DOM_07_2.htm */
    document.write("Accès au document <a
href='DOM_07_2.htm'>DOM_07_2.htm</a><br />");

    /* Lien hypertexte donnant accès au document DOM_07_3.htm */
    document.write("Accès au document <a href='DOM_07_3.htm?livre=ENI
JavaScript'>DOM_07_3.htm (passage d'un paramètre)</a>");

</script>
```

Le script précédent ne présente aucune particularité. Bien évidemment, les deux liens définis donnant accès aux pages DOM\_07\_02.htm et DOM\_07\_03.htm auraient pu être programmés en HTML.

Le deuxième lien appelle le script DOM\_07\_03.htm en lui passant un paramètre nommé livre avec la valeur "ENI JavaScript" :

```
<a href='DOM_07_3.htm?livre=ENI JavaScript'>DOM_07_3.htm
(passage d'un paramètre)</a>
```

L'affichage donne ceci :



Passons maintenant au deuxième script (DOM\_07\_2.htm) appelé par le document principal DOM\_07\_1.htm. Le code (intégré à la section HTML <body>) est une fois de plus reproduit partiellement ci-après :

```
<!-- Début script JavaScript -->
<script type="text/JavaScript">
```

```

/* Affichage du nom du script */
alert("DOM_07_2");

/* Affichage de l'URL de la page donnant accès au document */
document.write("URL de la page donnant accès au document : "
+ document.referrer);

</script>

```

L'URL du document appelant est affichée dans ce script :

```

document.write("URL de la page donnant accès au document : "
+ document.referrer);

```

Dans notre cas l'affichage donne :



Terminons avec le troisième script (DOM\_07\_3.htm) appelé lui aussi depuis le document principal DOM\_07\_1.htm mais cette fois-ci avec un passage de paramètre. La reproduction du code source est plus complète car du JavaScript est placé à la fois dans la section HTML <head> et dans la section HTML <body> :

```

<!-- Début en-tête script HTML -->
<head>

    <!-- Balise meta -->
    <meta http-equiv="Content-Type"
    content="text/html; charset=utf-8" />

    <!-- Titre du script HTML -->
    <title>DOM_07_3</title>

    <!-- Début script JavaScript -->
    <script type='text/javascript'>

        /* Fonction historyBack */
        function historyBack()
        {
            /* Retour à la page précédente */
            window.history.back();
        }

    </script>

```

```

</head>

<!-- Début section body du script HTML -->
<body>

    <!-- Titre du traitement -->
    <h1>Editions ENI - JavaScript - DOM_07_3</h1>

    <!-- Début script JavaScript -->
    <script type='text/javascript'>

        /* Affichage du nom du script */
        alert("DOM_07_3");

        /*
        Affichage de l'URL de la page
        donnant accès au document
        */
        document.write("URL de la page donnant accès au document : "
        + document.referrer + "<br />");

        /*
        Affichage de la valeur du paramètre
        passé par la page précédente
        NB : param contiendra la chaîne ?param=Eni%20Javascript
        */
        var param = window.location.search;
        document.write("Paramètre (avant recodage) : "
        + param + "<br />");

        /*
        Décodage du paramètre (remplacement de %20
        par un espace dans notre cas)
        */
        param = unescape(param);
        document.write("Paramètre (après recodage) : "
        + param + "<br />");

        /*
        Suppression du nom du paramètre
        (livre dans notre cas)
        */
        param = param.substring(param.lastIndexOf("=")+1);
        document.write("Valeur du paramètre : "
        + param);

    </script>

    <!-- Formulaire -->
    <br /><br />
    <form>
        <input
            type="button"
            id="boutonHistoryBack"
            value="Retour à la page précédente"

```

```

        onclick="historyBack()"
    />
</form>

```

L'exécution du script suit la séquence suivante :

- Affichage de l'URL appelante (comme dans le document DOM\_07\_2.htm).
- Traitement et affichage du paramètre passé par le script DOM\_07\_1.htm :

```

/*
Affichage de la valeur du paramètre
passé par la page précédente
NB : param contiendra la chaîne ?param=Eni%20Javascript
*/
var param = window.location.search;
document.write("Paramètre (avant recodage) : "
+ param + "<br />");

/*
Décodage du paramètre (remplacement de %20
par un espace dans notre cas)
*/
param = unescape(param);
document.write("Paramètre (après recodage) : "
+ param + "<br />");

```

window.location.search isole dans l'URL la partie contenant le paramètre (?livre=Eni%20JavaScript). La fonction unescape permet ensuite de remplacer les caractères encodés lors de l'appel du document depuis DOM\_07\_1.htm. L'espace représenté dans l'URL par la séquence '%20' sera rétabli.

L'affichage donne ceci :



À la fin du script, vous trouvez un mini-formulaire avec un bouton appelant une fonction JavaScript de nom historyBack (placée dans la section HTML <head>). Cette fonction provoque le retour au script appelant (DOM\_07\_1.htm) :

```

window.history.back();

```

## 8. Affichage de caractéristiques générales du document

Sans vouloir être exhaustif, le script va vous montrer comment accéder à quelques caractéristiques générales (titre, URL, date de dernière modification...) du script en cours :

```
<!-- Début script HTML -->
<html>

  <!-- Début en-tête script HTML -->
  <head>

    <!-- Balise meta -->
    <meta HTTP-equiv="Content-Type"
    content="text/html; charset=utf-8" />
    <!-- Titre du script HTML -->
    <title>DOM_08</title>

  </head>

  <!-- Début section body du script HTML -->
  <body>

    <!-- Titre du traitement -->
    <h1>Editions ENI - JavaScript - DOM_06</h1>

    <!-- Début script JavaScript -->
    <script type="text/JavaScript">

      /* Affichage du nom du script */
      alert("DOM_08");

      /* Affichage du titre du document */
      document.write("Titre du document : "
      + document.title + "<br />");

      /* Affichage du nom de domaine du document */
      document.write("Domaine du document : "
      + document.domain + "<br />");

      /* Affichage de l'URL absolue du document */
      document.write("URL absolue du document : "
      + document.URL + "<br />");

      /* Affichage de la date de dernière modification du document */
      document.write("Date de dernière modification du document : "
      + document.lastModified);

    </script>
```

Différents affichages sont prévus concernant des caractéristiques générales du script :

- le titre du document :

```
document.write("Titre du document : "
+ document.title + "<br />");
```

- le domaine du document :

```
document.write("Domaine du document : "
+ document.domain + "<br />");
```

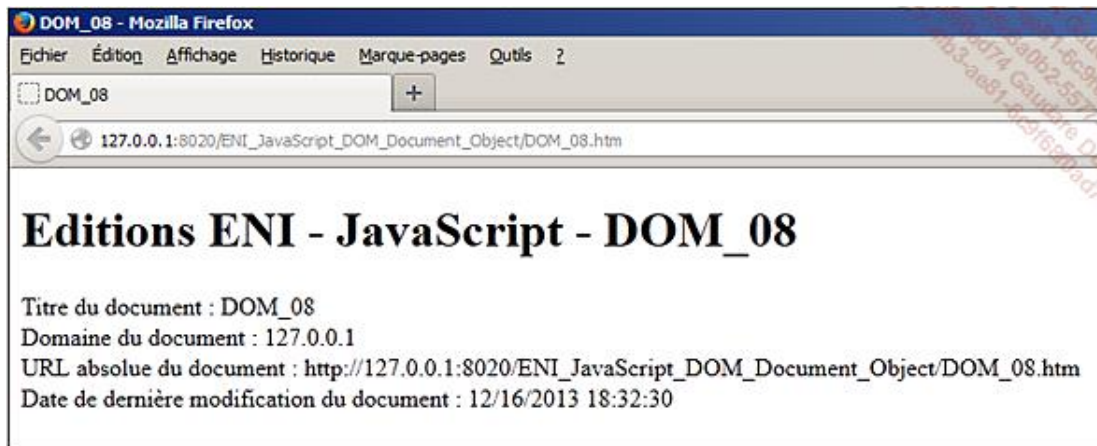
- l'URL (absolue) du document :

```
document.write("URL absolue du document : "
+ document.URL + "<br />");
```

- la date de dernière modification :

```
document.write("Date de dernière modification du document : "
+ document.lastModified);
```

L'exécution du script donne ceci :



## 9. Gestion des boutons dans les formulaires

Au travers de l'exemple proposé dans la section Gestion des formulaires et de leurs balises, nous avons étudié cette gestion en JavaScript.

Allons maintenant plus loin avec des formulaires dans lesquels nous allons positionner des champs de saisie (balises HTML `input`).

### Exemple 1

Dans un premier exemple, attachons-nous essentiellement à la récupération d'informations (nom, identifiant) sur les boutons de formulaires. Dans la section HTML `<body>` notre document contient (présentation partielle) la description HTML de trois formulaires :

```

<!-- Formulaire avec un bouton à désactiver -->
<form name="formulaire1">
    Un clic sur le bouton de confirmation va le désactiver<br><br />
    <input
        type="button"
        id="monBouton"
        value="OK"
        onclick="desactiverBouton();"
    />
</form>

<!-- Formulaire avec 2 boutons (obtention de l'intitulé) -->
<br /><hr><br />
<form name="formulaire2">
    Un clic sur l'un des boutons en affichera le nom (intitulé)<br /><br />
    <input
        type="button"
        id="monBouton1"
        value="Bouton1"
        onclick="afficherNomBouton(this.value);"
    /><br />
    <input
        type="button"
        id="monBouton2"
        value="Bouton2"
        onclick="afficherNomBouton(this.value);"
    /><br />
</form>

<!-- Formulaire avec 2 boutons et 1 lien hypertexte
(obtention de l'identifiant) -->
<br /><hr><br />
Un clic sur l'un des boutons ou le lien hypertexte
en affichera l'identifiant<br /><br />
<form name="formulaire3">
    <input
        type="button"
        id="monBouton3"
        value="Bouton3"
        onclick="afficherIdentifiantElement(this.id);"
    /><br />
    <input
        type="button"
        id="monBouton4"
        value="Bouton4"
        onclick="afficherIdentifiantElement(this.id);"
    /><br />
    <a
        href="#"
        id="monLien"
        onclick="afficherIdentifiantElement(this.id);"
        Lien hypertexte
    </a>

```



```
</form>
```

Chaque bouton (`<input type="button" ...>`) appelle une fonction JavaScript. À titre d'exemple, le premier bouton se trouvant dans le premier formulaire appelle la fonction JavaScript `desactiverBouton` lors d'un clic sur lui (déclencheur `onclick`).

Étudions maintenant dans le détail la programmation associée à chaque bouton.

Pour le bouton d'identifiant `monBouton` du premier formulaire, le code du bouton (section HTML `<body>`) est :

```
<input
  type="button"
  id="monBouton"
  value="OK"
  onclick="desactiverBouton();"
/>
```

et la fonction déclenchée a pour code source (section HTML `<head>`) :

```
/* Fonction desactiverBouton */
function desactiverBouton()
{
  document.getElementById("monBouton").disabled=true;
}
```

`document.getElementById("monBouton")` désigne le bouton identifié par un `id` égal à `monBouton` et `.disabled=true` rend ce bouton inaccessible.

Pour le bouton d'identifiant `monBouton1` du deuxième formulaire, le code du bouton (section HTML `<body>`) est :

```
<input
  type="button"
  id="monBouton1"
  value="Bouton1"
  onclick="afficherNomBouton(this.value);"
/>
```

et la fonction déclenchée a pour code source (section HTML `<head>`) :

```
/* Fonction afficherNomBouton */
function afficherNomBouton(nomBouton)
{
  alert("L'intitulé du bouton sélectionné est " + nomBouton);
}
```

Le paramètre `this.value` passé à la fonction `afficherNomBouton`, représentant le libellé affiché sur le bouton (`value="Bouton1"`), est affiché par la méthode `alert` de la fonction `afficherNomBouton`.

Le bouton identifié `monBouton2` n'est pas commenté ici, il appelle aussi la fonction `afficherNomBouton`.

Pour le bouton d'identifiant `monBouton3` du troisième formulaire, le code (section HTML `<body>`) est :

```
<input
  type="button"
  id="monBouton3"
  value="Bouton3"
  onclick="afficherIdentifiantElement(this.id);"
/>
```

et la fonction déclenchée a pour code source (section HTML `<head>`) :

```
/* Fonction afficherIdentifiantBouton */
function afficherIdentifiantBouton(idBouton)
{
  alert("Le bouton sélectionné a pour identifiant "+ idBouton);
}
```

Le paramètre `this.id` passé à la fonction `afficherIdentifiantBouton`, représentant l'identifiant du bouton (`id="monBouton3"`), est affiché par la méthode `alert` de la fonction `afficherIdentifiantBouton`.

Le bouton identifié par `monBouton4` n'est pas commenté ici, il appelle aussi la fonction `afficherIdentifiantBouton`.

Enfin, pour le lien hypertexte identifié `monLien` dans le troisième formulaire, le code du bouton (section HTML `<body>`) est :

```
<a
  href="#"
  id="monLien"
  onclick="afficherIdentifiantElement(this.id);">
  Lien hypertexte
</a>
```

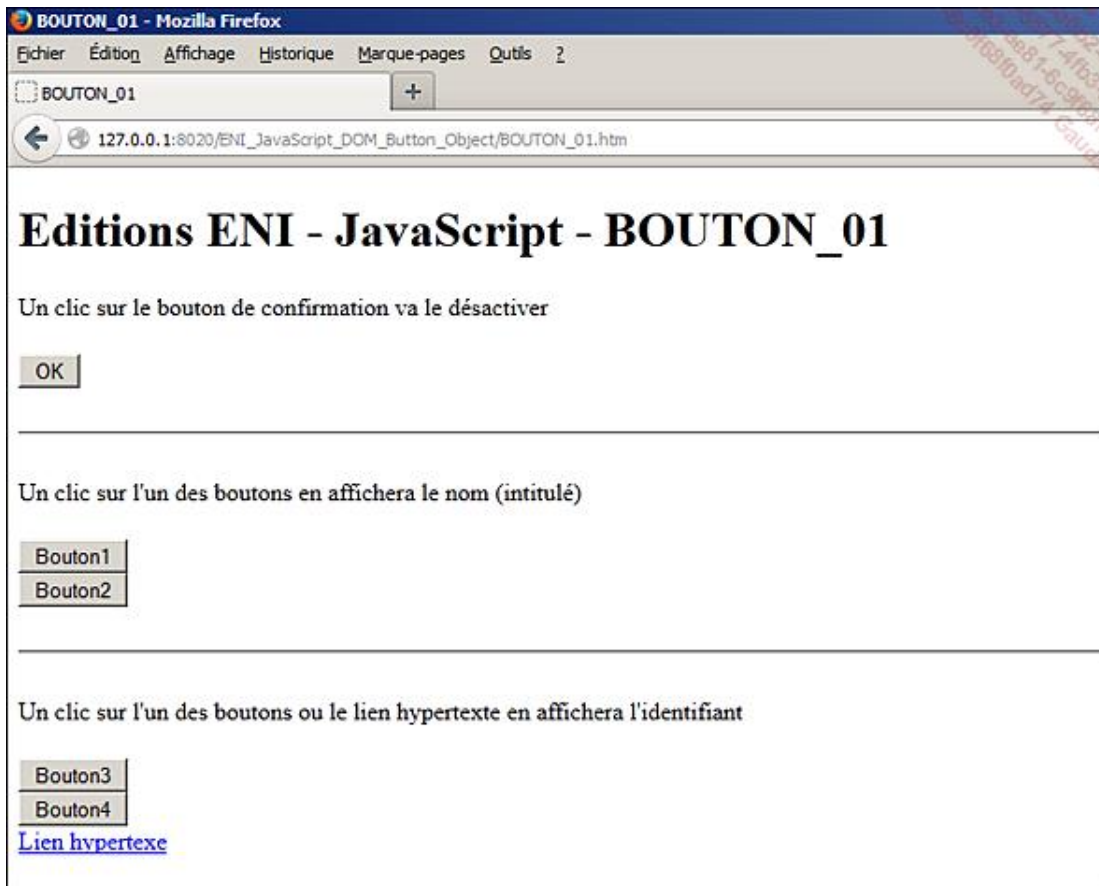
et la fonction déclenchée a pour code source (section HTML `<head>`) :

```
/* Fonction afficherIdentifiantElement */
function afficherIdentifiantElement(idElement)
{
  alert("Le lien sélectionné a pour identifiant "+ idElement);
}
```



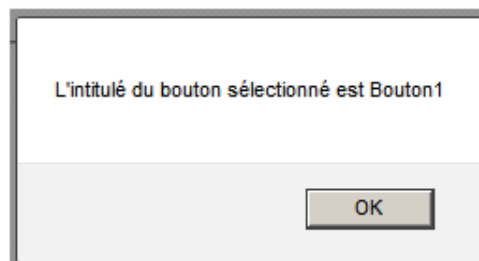
Vous avez suffisamment d'expérience pour détecter que la fonction `afficherIdentifiantElement` aurait également pu être utilisée pour les boutons vus précédemment.

Récapitulons ce qui se passe à l'exécution de ce script. Les formulaires s'affichent comme suit :

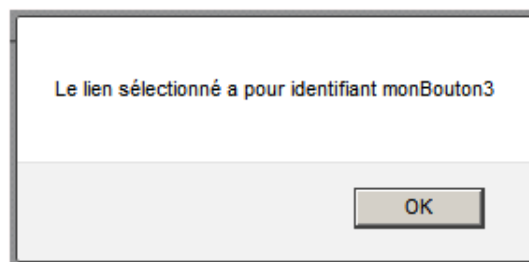


Un clic sur le bouton **OK** du premier formulaire désactive ce bouton.

Un clic sur le bouton **Bouton1** affiche ceci :



Un clic sur le bouton **Bouton3** déclenche cet affichage :



Enfin, un clic sur le lien hypertexte affiche ceci :



## Exemple 2

Étudions un deuxième exemple concernant les formulaires et leurs boutons avec un script un peu plus conséquent.

Dans la section HTML <body>, vous trouverez :

```
<!-- Formulaire formulaire1 avec trois boutons -->
Formulaire n°1 :<br />
<form name="formulaire1">
  <input
    type="button"
    id="monBouton1"
    value="Bouton1"
    onclick="afficherProprietesBouton(this.value, this.id);"
  /><br />
  <input
    type="button"
    id="monBouton2"
    value="Bouton2"
    onclick="afficherProprietesBouton(this.value, this.id);"
  /><br />
  <input
    type="button"
    id="monBouton3"
    value="Bouton3"
    onclick="afficherProprietesBouton(this.value, this.id);"
  />
</form>

<!-- Formulaire formulaire2 avec deux boutons -->
<br /><hr /><br />
Formulaire n°2 :<br />
<form name="formulaire2">
  <input
    type="button"
    id="monBouton4"
    value="Bouton4"
    onclick="afficherProprietesBouton(this.value, this.id);"
  /><br />
  <input
    type="button"
    id="monBouton5"
    value="Bouton5"
    onclick="afficherProprietesBouton(this.value, this.id);"
```

```

    />
</form>

<!-- Début script JavaScript -->
<script type="text/Javascript">

    /*
    Affichage du nombre de balises input
    dans l'ensemble des formulaires
    */
    var listeInput = document.getElementsByTagName('input');
    var nombreInput = listeInput.length;
    document.write("<br /><hr /><br />");
    document.write("Nombre de balises input dans l'ensemble des formulaires : "
    + nombreInput + "<br />");

    /*
    Affichage du nombre de balises input
    dans le 2ème formulaire
    */
    var liste2Input = document.forms[1].getElementsByTagName('input');
    var nombre2Input = liste2Input.length;
    document.write("Nombre de balises input dans le 2ème formulaire : "
    + nombre2Input);

</script>

```

L'affichage de deux formulaires est prévu. Le premier formulaire contient trois champs de saisie de type `<input type="button" ...>` et le second formulaire en contient deux.

Chaque bouton possède les attributs (propriétés) `id` et `value` et déclenche une même fonction `afficherProprietesNomBouton`. Pour chaque appel à la fonction `afficherProprietesNomBouton`, deux paramètres sont passés : le libellé du bouton (`this.value`) et l'identifiant (`this.id`).

À la suite de l'affichage des deux formulaires, les informations suivantes seront présentées :

- le nombre de balises `input` dans l'ensemble des formulaires ;
- le nombre de balises `input` dans le deuxième formulaire.

L'instruction :

```
var listeInput = document.getElementsByTagName('input');
```

crée un tableau (liste) des éléments DOM de nom `input` de notre document.

Puis le nombre d'éléments est déterminé :

```
var nombreInput = listeInput.length;
```

et affiché :

```
document.write("Nombre de balises input dans l'ensemble des  
formulaires : "  
+ nombreInput + "<br />");
```

Pour déterminer le nombre d'éléments DOM de nom `input` dans le deuxième formulaire, la syntaxe est assez proche :

```
var liste2Input = document.forms[1].getElementsByTagName('input');
```

Dans cette syntaxe `forms[1]` vient préciser que seuls les `input` du second formulaire (numéroté 1) doivent être comptabilisés.

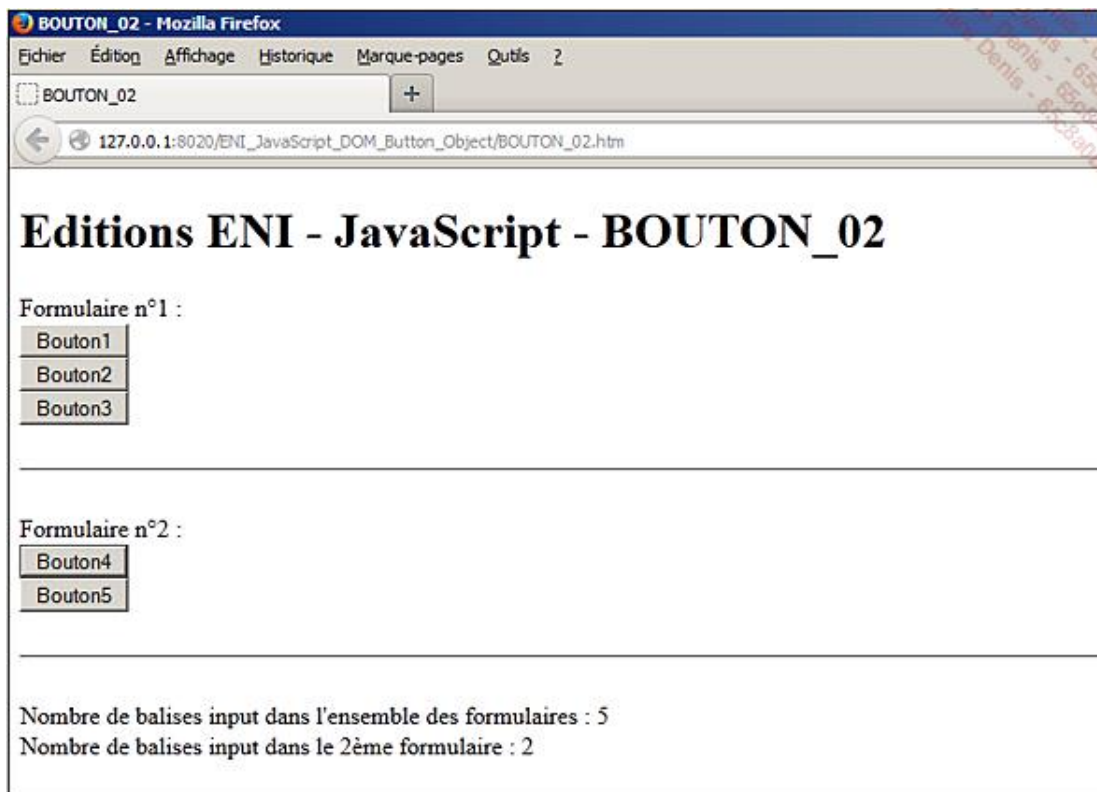
Reste à voir le contenu de la fonction `afficherProprietesBouton` (section HTML <head>) :

```
/* Fonction afficherProprietesBouton */  
function afficherProprietesBouton(nomBouton, idBouton)  
{  
    alert("L'intitulé du bouton sélectionné est  
    "+ nomBouton + " et ce bouton appartient au formulaire "  
    + document.getElementById(idBouton).form.name);  
}
```

Le traitement est sans surprise.

Notons juste que `document.getElementById(idBouton).form.name` affiche le nom du formulaire (`formulaire1` ou `formulaire2`) dans lequel se trouve le bouton sur lequel il a été cliqué.

Récapitulons une fois de plus ce qui se passe à l'exécution du script. Les formulaires s'affichent comme suit :



Un clic sur le bouton Bouton1 du premier formulaire affiche ceci :



Un clic sur le bouton Bouton4 du second formulaire provoque cet affichage :



## 10. Gestion des tableaux (balise HTML table)

La gestion des tableaux HTML (balise table) est une option des plus intéressantes.

Regardons une fois de plus au travers d'une série d'exemples ce que vous pourriez faire sur les tableaux HTML avec un minimum de connaissances en JavaScript.

## Exemple 1

L'objectif est d'apprendre à modifier les caractéristiques d'un tableau (border, cellspacing, cellpadding et caption) via du JavaScript.

Rappelons brièvement à quoi correspondent ces quatre attributs :

- attribut `border` : épaisseur du cadre (par défaut 1) exprimée en pixel(s),
- attribut `cellspacing` : taille de la bordure entre les cellules exprimée en pixel(s),
- attribut `cellpadding` : espace compris entre le contenu des cellules et les bordures du tableau exprimé en pixel(s),
- attribut `caption` : intitulé précédent le tableau.

Rassurez-vous, le résultat de l'exécution du script sera proposé à la présentation de cet exemple.

Regardons tout de suite le contenu de la section HTML `<body>` de notre exemple :

```
<!-- Tableau HTML -->
<table id="monTableau" border="1" cellspacing="1" cellpadding="1">
  <!-- Définition de la ligne n°0 du tableau (titre) -->
  <tr>
    <td>Modèle</td>
    <td>Puissance</td>
  </tr>
  <!-- Définition de la ligne n°1 du tableau -->
  <tr>
    <td>Ferrari 512 BB</td>
    <td>380</td>
  </tr>
  <!-- Définition de la ligne n°2 du tableau -->
  <tr>
    <td>Lamborghini Miura</td>
    <td>385</td>
  </tr>
  <!-- Définition de la ligne n°3 du tableau -->
  <tr>
    <td>Porsche 964 3.6</td>
    <td>360</td>
  </tr>
</table>

<!-- Formulaire avec boutons -->
<br /><br />
<form>
  Modification des valeurs d'attributs du tableau<br /><br />
  <input
    type="button"
    id="boutonBorder"
    value="Modification attribut border"
    onclick="modifierBorder();"
  /><br />
```



```



```

Le tableau (une ligne de titres de colonnes et trois lignes de données (deux colonnes intitulées *Modèle* et *Puissance*)) ne requiert pas de remarques. Vous noterez quand même les attributs par défaut associés à ce tableau :

- attribut `id` : `monTableau`
- attribut `border` : `1`
- attribut `cellspacing` : `1`
- attribut `cellpadding` : `1`

Sous ce tableau a été mis en place un formulaire à quatre boutons. Les boutons ont pour rôle (dans l'ordre de leur présentation dans le formulaire) :

- de modifier la valeur de l'attribut `border`,
- de modifier la valeur de l'attribut `cellspacing`,
- de modifier la valeur de l'attribut `cellpadding`,
- d'ajouter un titre (attribut `caption`).

Pour les quatre appels à des fonctions JavaScript, il n'est pas prévu de passage de paramètres. Les quatre appels se font sur l'événement `onclick`.

Passons maintenant à l'étude des quatre fonctions JavaScript (section HTML `<head>`).

Le code de la fonction `modifierBorder` est :

```

/* Fonction modifierBorder */
function modifierBorder()
{
    if (document.getElementById('monTableau').border == "1")
    {

```

```

        /* Passage à une épaisseur de 5 */
        document.getElementById('monTableau').border="5";

    }
    else
    {
        /* Passage à une épaisseur de 1 */
        document.getElementById('monTableau').border="1";
    }
}

```

Un test est effectué sur l'épaisseur de la bordure du tableau repéré par son identifiant (getElementById) :

```

if (document.getElementById('monTableau').border == "1")

```

Si l'épaisseur de la bordure est égale à 1 (la valeur par défaut dans le code HTML vu précédemment) alors l'épaisseur est portée à 5 (et inversement ramenée à 1 dans le cas contraire).

La fonction modifierBorder a pour code :

```

/* Fonction modifierCellspacing */
function modifierCellspacing()
{
    if (document.getElementById('monTableau').cellSpacing == "1")
    {
        /* Passage à un cellspacing à 40 */
        document.getElementById('monTableau').cellSpacing="40";
    }
    else
    {
        /* Passage à un cellspacing à 1 */
        document.getElementById('monTableau').cellSpacing="1";
    }
}

```

Vous le voyez, le code est semblable à celui de la fonction précédente, seul l'attribut cellSpacing est venu remplacer border.

Le code de la troisième fonction, modifierCellpadding, n'est pas reproduit ici. Seul l'attribut cellPadding vient remplacer le cellSpacing.

La dernière fonction ajouterCaption, est reproduite ci-après :

```

/* Fonction AjouterCaption */
function AjouterCaption()
{

    /* Instanciation et affichage du titre du tableau */
    var monTitre = document.getElementById('monTableau').createCaption();
    monTitre.innerHTML = "Voitures de sport";
}

```

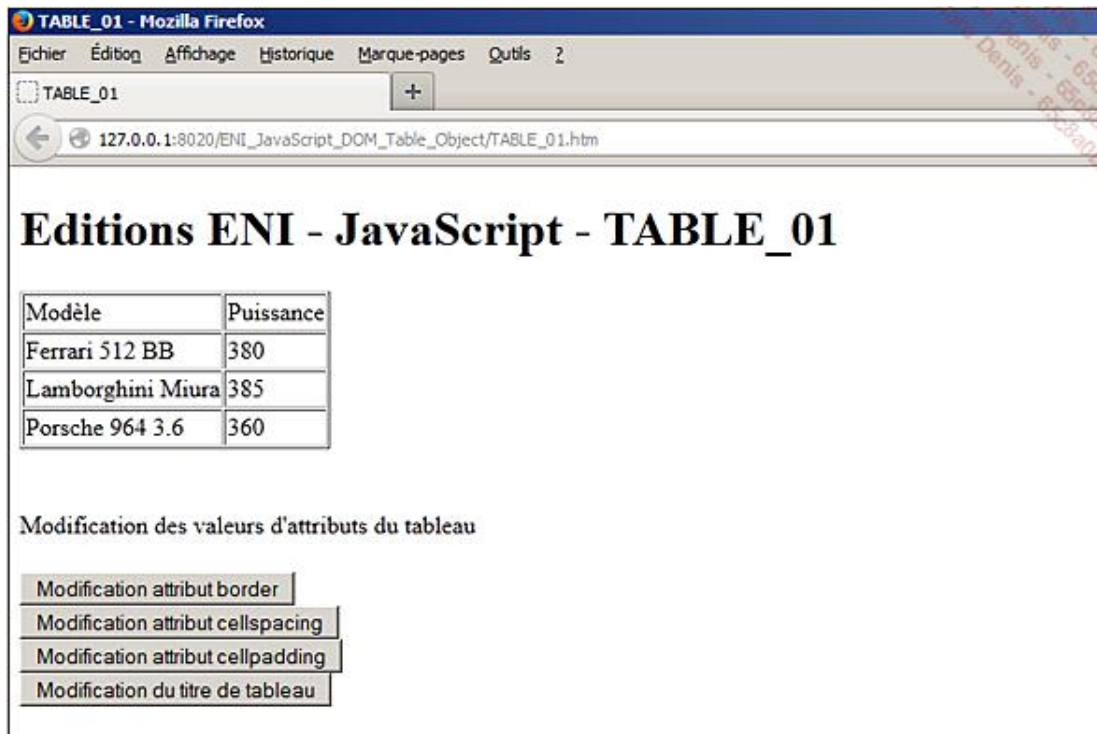
```
}
```

Un titre est créé pour le tableau d'identifiant `monTableau` et ceci par l'intermédiaire d'une fonction de nom `createCaption`.

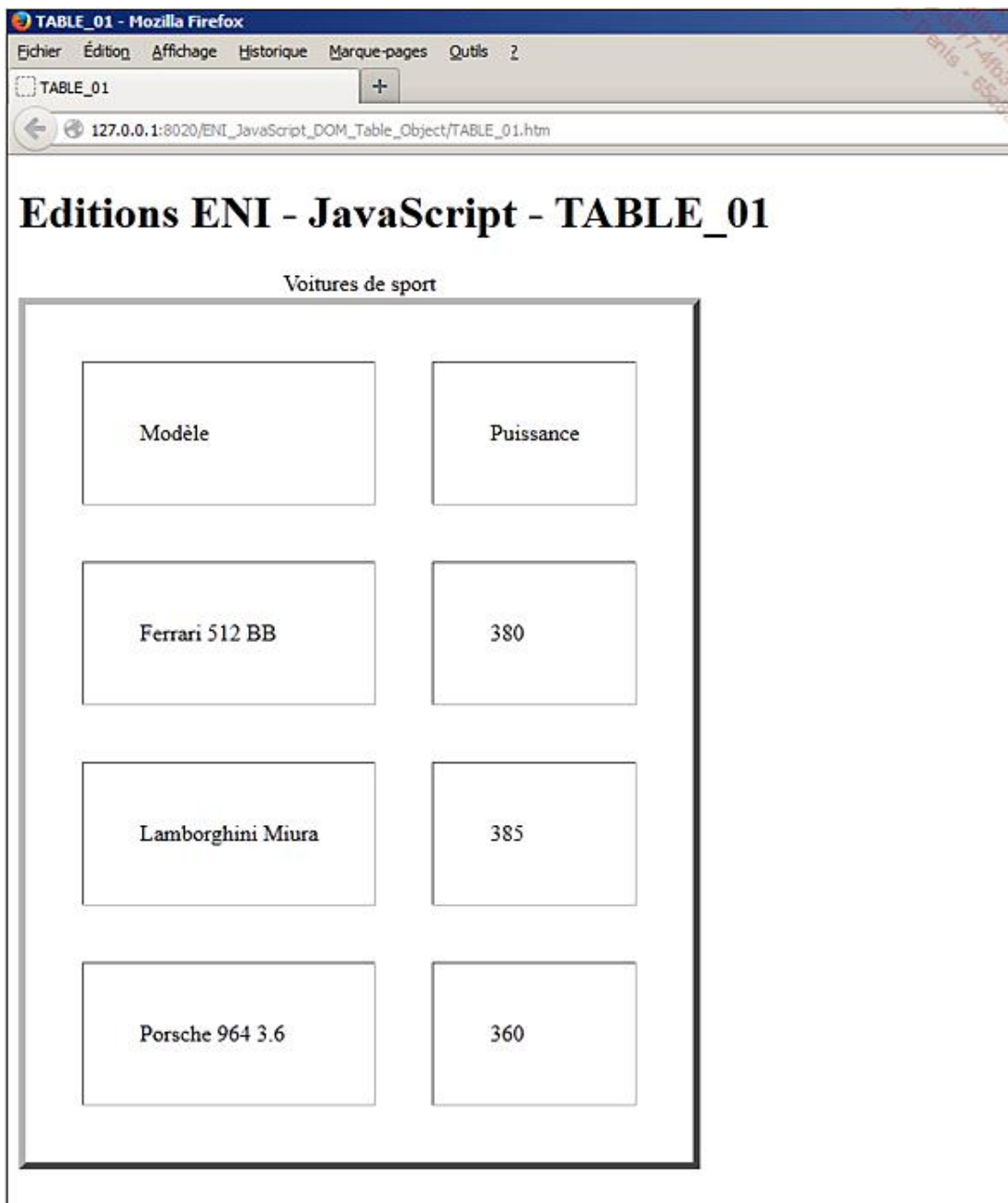
Ensuite l'affectation du titre sur le dessus du tableau s'effectue par :

```
monTitre.innerHTML = "Voitures de sport";
```

Voyons ce que donne l'exécution de ce script à l'affichage. Commençons par l'état initial (avant utilisation des boutons) :



Après utilisation des quatre boutons, l'affichage se présente comme suit :



## Exemple 2

Dans ce second exemple, attachons-nous à automatiser via JavaScript la mise en place d'un trait horizontal d'entête s'affichant au-dessus du tableau HTML (un trait de pied de tableau est également prévu). Il n'y a aucune difficulté point de vue technique, c'est par contre l'occasion de découvrir un attribut HTML assez méconnu, `frame`.

Rappelons que cet attribut HTML peut prendre deux valeurs :

- `above` : affichage de bordures externes en haut du tableau seulement,
- `below` : affichage de bordures externes en bas du tableau seulement.

À la suite du tableau HTML déjà vu au travers de l'exemple 1, un formulaire est prévu :

```
<!-- Formulaire avec boutons -->
```

```

<br /><br />
<form>
  Ajout/Suppression des traits de délimitation du tableau<br /><br />
  <input
    type="button"
    id="boutonTraitAuDessus"
    value="Trait au-dessus du tableau"
    onclick="placerTraitAuDessus();"
  /><br />
  <input
    type="button"
    id="boutonTraitEnDessous"
    value="Trait au-dessous du tableau"
    onclick="placerTraitEnDessous();"
  /><br />
</form>

```

Deux boutons `<input type="button" ...>` sont présents dans le formulaire, le premier appelant une fonction `placerTraitAuDessus` et le second une fonction `placerTraitEnDessous`.

La première procédure a pour code :

```

/* Fonction placerTraitAuDessus */
function placerTraitAuDessus()
{
  document.getElementById( 'monTableau' ).frame="above" ;
}

```

et la seconde :

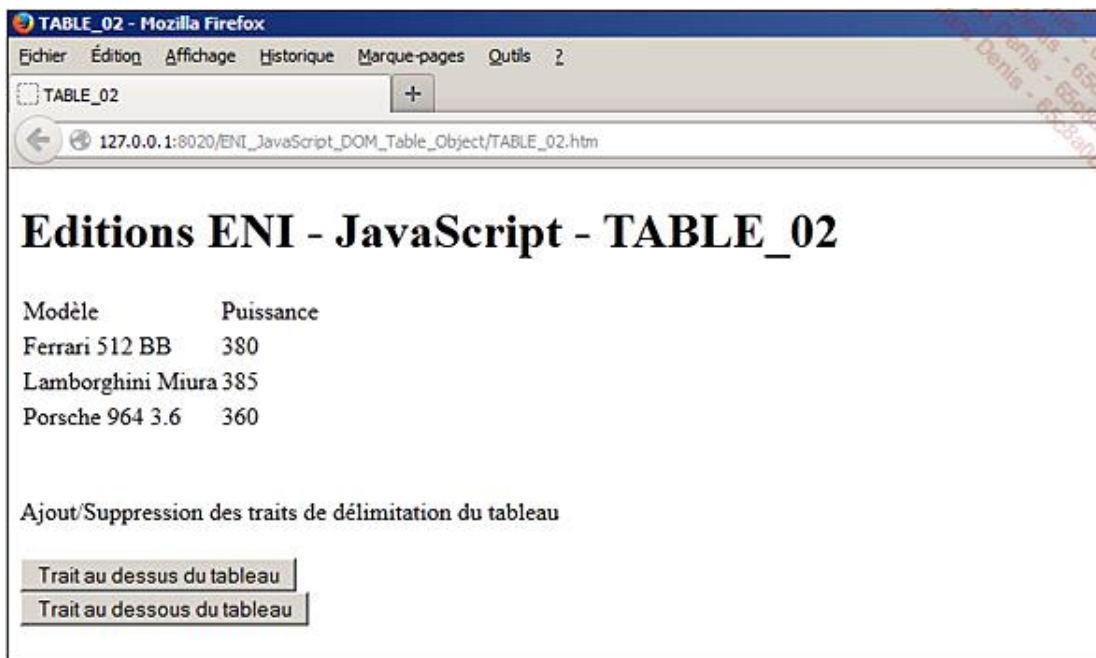
```

/* Fonction placerTraitEnDessous */
function placerTraitEnDessous()
{
  document.getElementById( 'monTableau' ).frame="below" ;
}

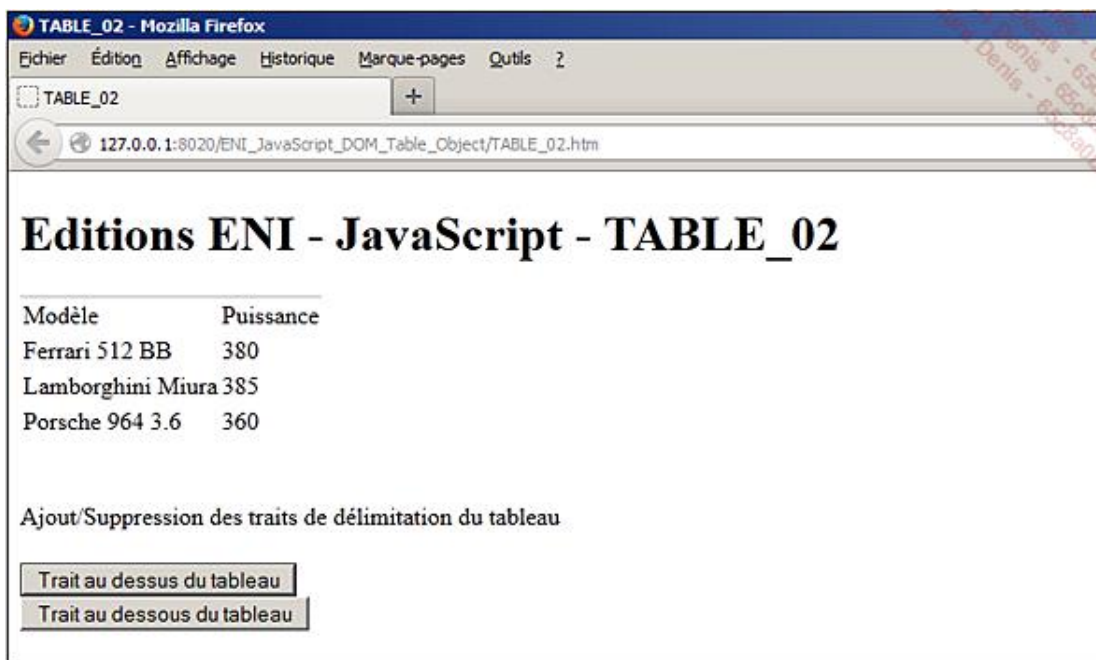
```

Dans les deux cas la propriété (attribut HTML) `frame` est paramétrée et ceci pour le tableau identifié par `monTableau`.

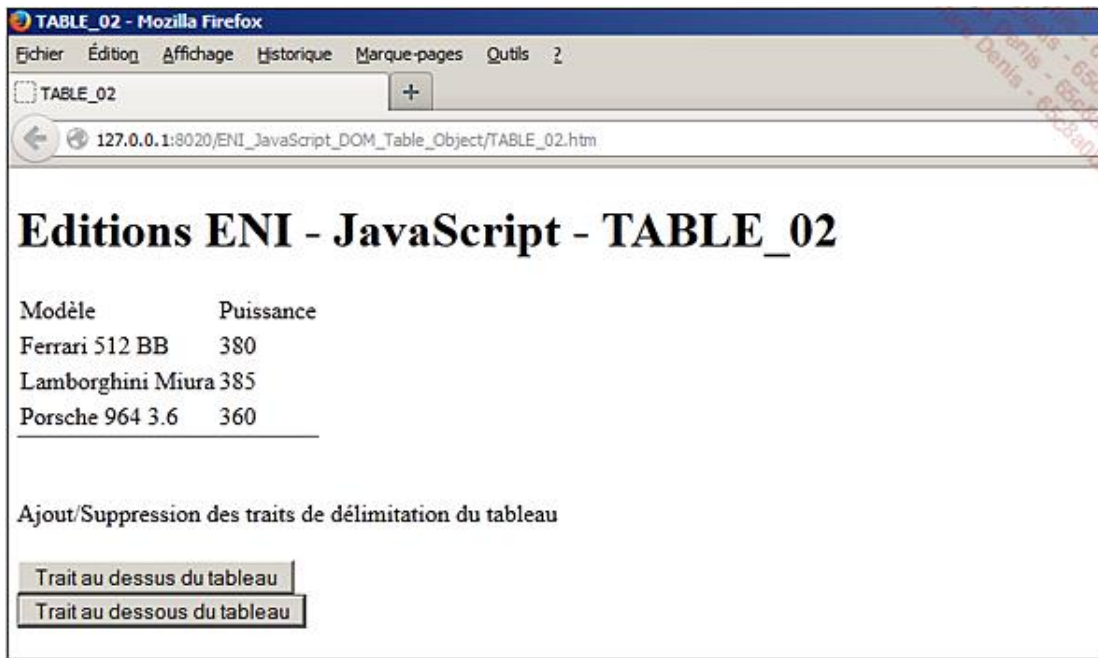
À l'exécution du script, l'état initial du tableau est le suivant :



Après sollicitation du premier bouton, l'affichage se présente comme suit :



et après sollicitation du deuxième bouton, l'affichage devient :



### Exemple 3

Basons nous une fois de plus sur le tableau proposé dans les exemples 1 et 2. À la suite (dans la section HTML <body>), deux formulaires sont installés :

```

<!-- Formulaire affichage -->
<br /><hr /><br />
<form name="affichage">
    Affichage d'une cellule du tableau<br /><br />
    N° de ligne (entre 0 et 3) :
    <input
        type="text"
        id="numLigne1"
        value=""
    /><br />
    N° de colonne (entre 0 et 1) :
    <input
        type="text"
        id="numColonne1"
        value=""
    /><br />
    <input
        type="button"
        id="boutonAfficherCellule"
        value="Affichage d'une cellule"
        onclick="afficherCellule()"
    /><br />
</form>

<!-- Formulaire modification -->
<br /><hr /><br />
<form name="modification">
    Modification d'une cellule du tableau<br /><br />

```

```

N° de ligne (entre 0 et 3) :
<input
    type="text"
    id="numLigne2"
    value=""
/><br />
N° de colonne (entre 0 et 1) :
<input
    type="text"
    id="numColonne2"
    value=""
/><br />
<input
    type="button"
    id=boutonAfficherCellule"
    value="Modification d'une cellule"
    onclick="modifierCellule()"
/><br />
</form>

```

Dans le premier formulaire, un bouton `<input type="button" ...>` est prévu pour solliciter une fonction de nom `afficherCellule` sur la base d'un numéro de ligne et d'un numéro de colonne saisis auparavant au travers de deux zones de saisie `<input type="text" ...>` (`numLigne1` et `numColonne1`).

Structurellement le second formulaire est identique, un bouton `<input type="button" ...>` est prévu pour solliciter la fonction de nom `modifierCellule`.

Passons au code JavaScript de la fonction `afficherCellule` (placée dans la section HTML `<head>`) :

```

/* Fonction afficherCellule */
function afficherCellule()
{
    /*
    Affichage de contrôle du numéro de la ligne
    sélectionnée dans le tableau
    */
    var numeroLigne =
    parseInt(document.getElementById('numLigne1').value);
    alert("N° de ligne : " + numeroLigne);
    /*
    Affichage de contrôle du numéro de la colonne
    sélectionnée dans le tableau
    */
    var numeroColonne =
    parseInt(document.getElementById('numColonne1').value);
    alert("N° de colonne : " + numeroColonne);
    /*
    Affichage du nombre de lignes du tableau
    */
    var nombreLignes =
    parseInt(document.getElementById('monTableau').rows.length);
    alert("Nombre de ligne(s) du tableau : " + nombreLignes);
    /*

```



```

Affichage du nombre de colonnes du tableau
(par détermination du nombre de cellules de la ligne de titre)
*/
var nombreColonnes =
parseInt(document.getElementById('monTableau')
.rows[0].cells.length);
alert("Nombre de colonne(s) du tableau : " + nombreColonnes);
/*
Affichage de la cellule
*/
if ((numeroLigne >= 0) && (numeroLigne <= (nombreLignes - 1))
&& (numeroColonne >= 0) && (numeroColonne <= (nombreColonnes - 1)))
{
    /* Affichage des cellules (colonnes) de la ligne */
    alert("Code HTML de la ligne : "
+ document.getElementById('monTableau')
.rows[numeroLigne].innerHTML);
    /* Mise en tableau des cellules de la ligne en cours */
    var tabLigne =
document.getElementById('monTableau')
.rows[numeroLigne].cells;
    /* Affichage */
    alert("Contenu de la cellule : "
+ tabLigne[numeroColonne].innerHTML);
}
else
{
    alert("Votre numéro de ligne ou votre numéro de colonne
est erroné");
}
}

```

Le premier objectif de la fonction est de récupérer dans des variables mémoire le numéro de la ligne et le numéro de la colonne dont l'affichage est souhaité :

```
var numeroLigne = parseInt(document.getElementById('numLigne1').value);
```

et

```
var numeroColonne = parseInt(document.getElementById('numColonne1').value);
```

Ensuite, pour qu'un contrôle de cohérence du numéro de la ligne et du numéro de la colonne saisis au travers du formulaire puisse être effectué, le nombre de lignes et de le nombre de colonnes du tableau sont déterminés :

```
var nombreLignes =
parseInt(document.getElementById('monTableau').rows.length);
```

```
var nombreColonnes =
```

```
parseInt(document.getElementById('monTableau').rows[0].cells.length);
```

Passons enfin à l’affichage du contenu de la cellule ciblée. Dans un premier temps, le code HTML complet de la ligne est affiché par la méthode `alert` :

```
/* Affichage des cellules (colonnes) de la ligne */  
alert("Code HTML de la ligne : " +  
document.getElementById('monTableau').rows[numeroLigne].innerHTML);
```

Par exemple pour la ligne n°2 du tableau, le résultat sera : `<td>Lamborghini Miura</td><td>385</td>`

Rappelons que la propriété `innerHTML` fournit le contenu HTML de l’élément de tableau (ici une ligne entière).

Les différents éléments de la ligne (cellules ou colonnes) sont ensuite placés dans un tableau mémoire :

```
var tabLigne =  
document.getElementById('monTableau').rows[numeroLigne].cells;
```

Ce qui permet ensuite d’afficher la valeur de la colonne ciblée (sans le code HTML) comme suit :

```
alert("Contenu de la cellule : " + tabLigne[numeroColonne].innerHTML);
```

Par exemple pour le numéro de ligne 2 et le numéro de colonne 1, le résultat affiché est 385.

Il ne vous aura pas échappé qu’un contrôle de validité a été fait sur le numéro de la ligne et le numéro de la colonne :

```
if ((numeroLigne >= 0) && (numeroLigne <= (nombreLignes - 1)) &&  
(numeroColonne >= 0) && (numeroColonne <= (nombreColonnes - 1)))
```

La seconde fonction `modifierCellule` a un code source assez proche de celui de la fonction `afficherCellule`. Attardons-nous sur les quelques différences.

Comme précédemment les cellules du tableau pour la ligne concernée sont placées dans un tableau mémoire :

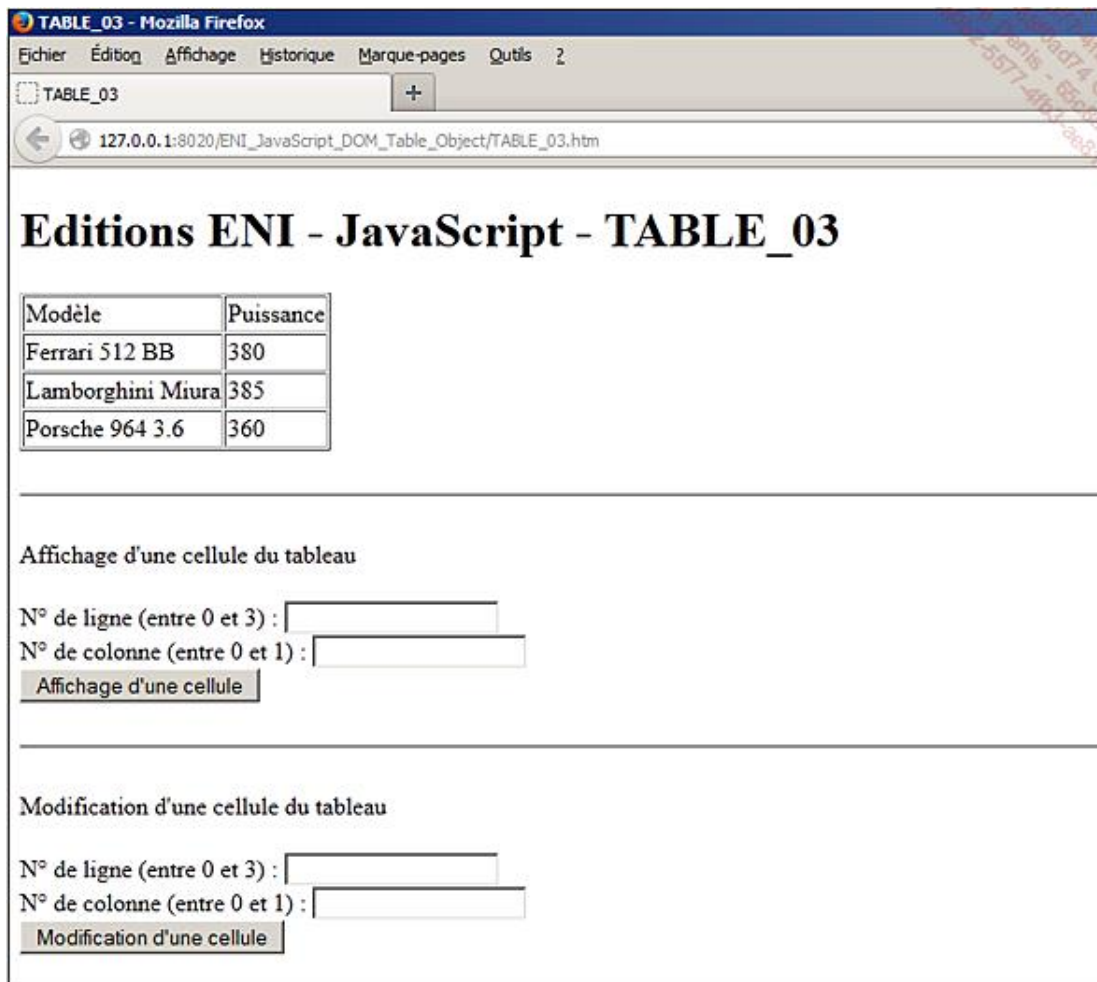
```
/* Mise en tableau des cellules de la ligne en cours */  
var tabLigne =  
document.getElementById('monTableau')  
.rows[numeroLigne].cells;
```

Ensuite il reste à modifier le contenu de la cellule ciblée. Dans notre exemple la valeur de remplacement est une constante ("Nouvelle valeur"). Bien évidemment dans une application plus évoluée cette valeur serait elle-même à saisir au travers d’un formulaire. Le code pour ce remplacement (modification) est le suivant :

```
/* Modification */
```

```
tabLigne[numeroColonne].innerHTML="Nouvelle valeur";
```

À l'exécution, l'affichage initial se présente comme suit :



**TABLE\_03 - Mozilla Firefox**

TABLE\_03

127.0.0.1:8020/ENI\_JavaScript\_DOM\_Table\_Object/TABLE\_03.htm

## Editions ENI - JavaScript - TABLE\_03

Modèle	Puissance
Ferrari 512 BB	380
Lamborghini Miura	385
Porsche 964 3.6	360

---

**Affichage d'une cellule du tableau**

N° de ligne (entre 0 et 3) :

N° de colonne (entre 0 et 1) :

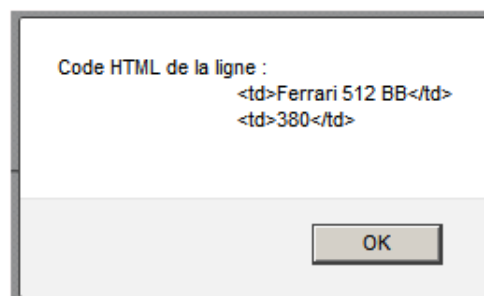
---

**Modification d'une cellule du tableau**

N° de ligne (entre 0 et 3) :

N° de colonne (entre 0 et 1) :

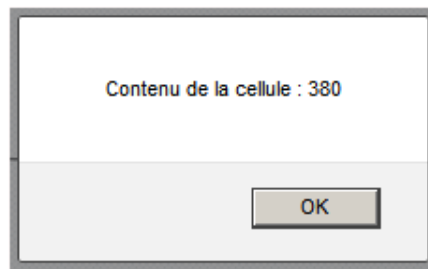
Si l'on saisit 1 en tant que numéro de ligne et 1 également en tant que numéro de colonne dans le premier formulaire, les affichages suivants apparaissent :



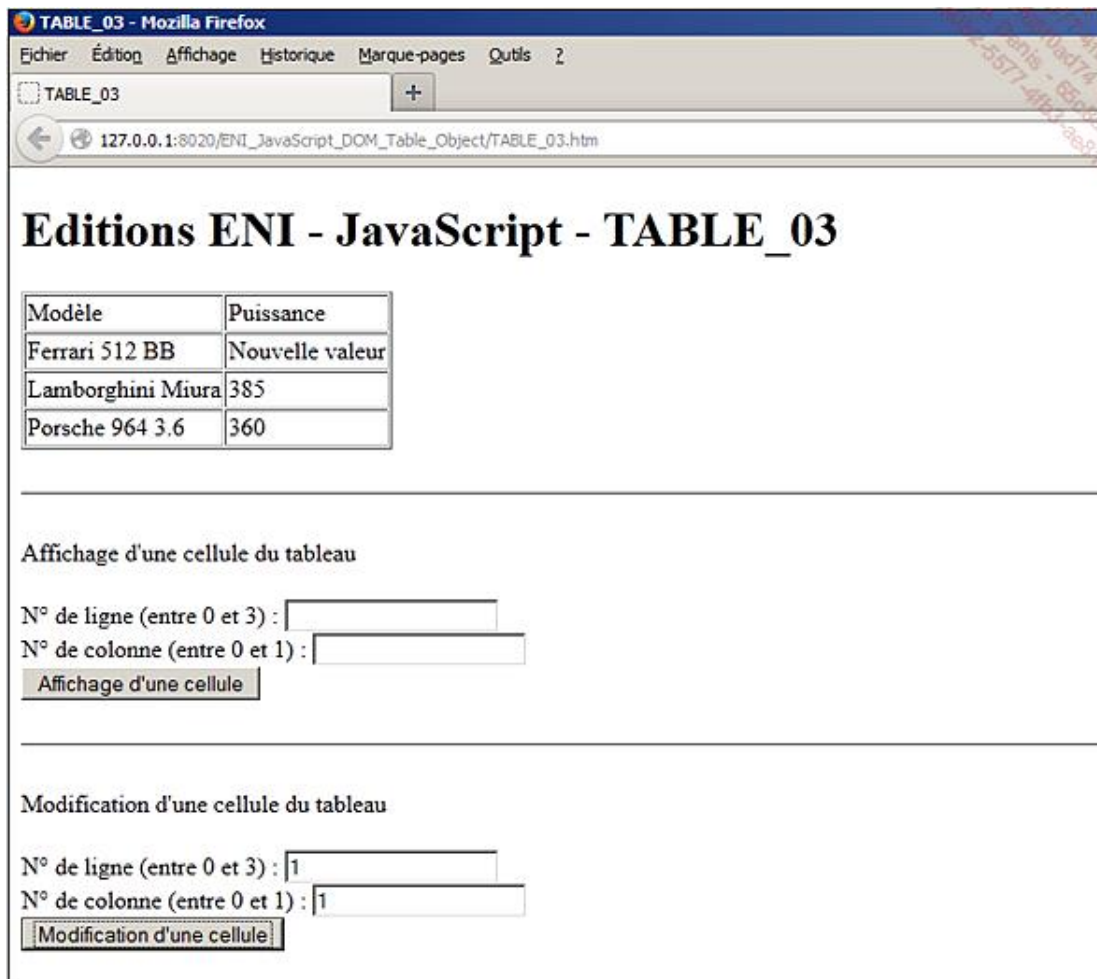
Code HTML de la ligne :

```
<td>Ferrari 512 BB</td>  
<td>380</td>
```

et



Enfin si l'on saisit 1 en tant que numéro de ligne et 1 également en tant que numéro de colonne dans le second formulaire, le tableau est modifié comme suit :



#### Exemple 4

Partons encore du tableau proposé dans les exemples précédents (comportant dans sa version initiale trois voitures de sport). L'objectif dans ce nouvel exemple va être d'insérer en première ligne (directement sous le titre) un nouveau véhicule ainsi qu'un second en fin de table. Pour ne pas trop alourdir le code de l'exemple (et les explications afférentes), les données insérées seront des valeurs constantes. Vous saurez aisément personnaliser ce code pour que les données puissent être saisies au clavier.

Pour l'ajout en tête de tableau (sous le titre) le formulaire est le suivant :

```
<!-- Formulaire d'ajout en première ligne ou en dernière ligne -->
```

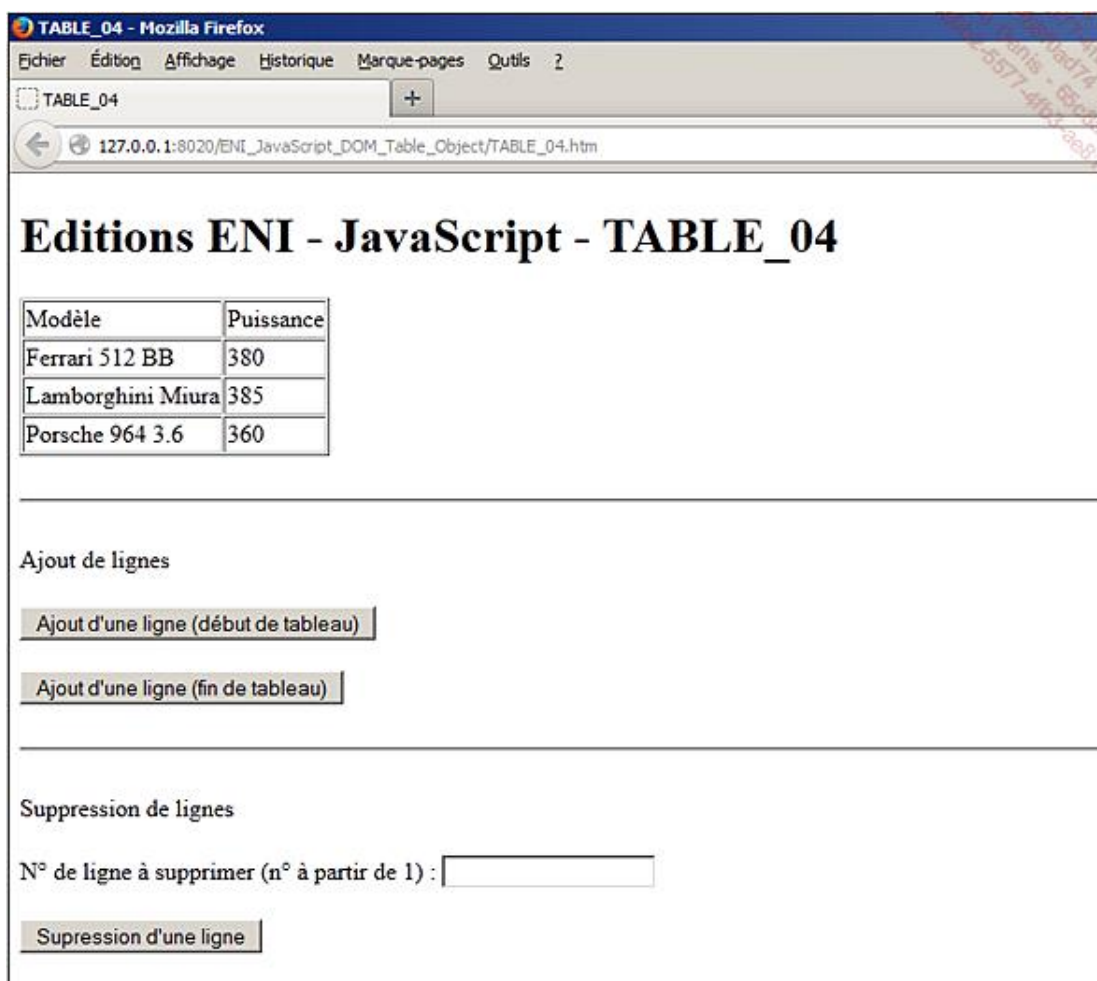
```

<br /><hr /><br />
Ajout de lignes<br /><br />
<form name="formulaireAjout">
  <input
    type="button"
    id="boutonAjouterLigneDebutTableau"
    value="Ajout d'une ligne (début de tableau)"
    onclick="ajouterLigneDebutTableau()"
  /><br /><br />
  <input
    type="button"
    id="boutonAjouterLigneFinTableau"
    value="Ajout d'une ligne (fin de tableau)"
    onclick="ajouterLigneFinTableau()"
  /><br />
</form>

```

Deux boutons `<input type="button" ... >` appellent les fonctions JavaScript `ajouterLigneDebutTableau` et `ajouterLigneFinTableau`.

À l'exécution l'état initial du tableau est le suivant :



**TABLE\_04 - Mozilla Firefox**

TABLE\_04

127.0.0.1:8020/ENI\_JavaScript\_DOM\_Table\_Object/TABLE\_04.htm

## Editions ENI - JavaScript - TABLE\_04

Modèle	Puissance
Ferrari 512 BB	380
Lamborghini Miura	385
Porsche 964 3.6	360

Ajout de lignes

Ajout d'une ligne (début de tableau)

Ajout d'une ligne (fin de tableau)

Suppression de lignes

N° de ligne à supprimer (n° à partir de 1) :

Suppression d'une ligne

Étudions le code JavaScript de la fonction `ajouterLigneDebutTableau` (placée dans la section HTML `<head>`) :

```

/* Fonction ajouterLigneDebutTableau */
function ajouterLigneDebutTableau()
{

    /*
    Ajout d'une ligne vide en position 1 (sous le titre)
    */
    var tabLigne=document.getElementById('monTableau').insertRow(1);

    /*
    Ajout d'une colonne 0 (Modèle) dans la ligne créée
    et stockage d'une valeur
    */
    var colonne0=tabLigne.insertCell(0);
    colonne0.innerHTML="Ferrari F40";

    /*
    Ajout d'une colonne 1 (Puissance) dans la ligne créée
    et stockage d'une valeur
    */
    var colonnel=tabLigne.insertCell(1);
    colonnel.innerHTML="478";
}

```

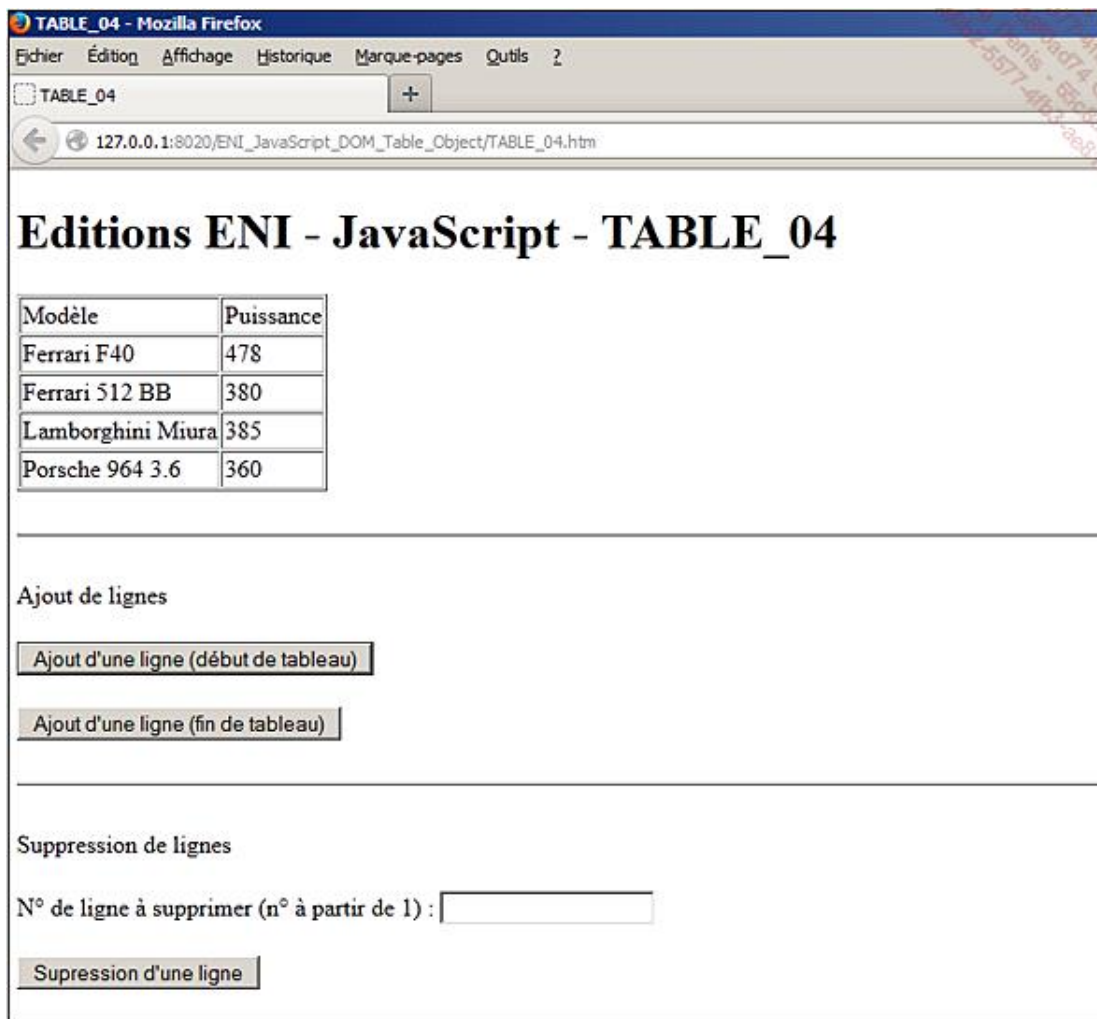
Il convient tout d'abord d'insérer une ligne (vide pour l'instant) sous le titre dans le tableau 'monTableau'. Vous noterez que la numérotation des lignes commence dans les tableaux HTML à 0, la ligne insérée sera donc la n°1. La méthode est insertRow.

```
var tabLigne=document.getElementById('monTableau').insertRow(1);
```

Une fois la ligne insérée, il faut prévoir la description des colonnes (deux dans notre cas) et y placer des données. Pour la première colonne (numérotée 0 également), le code est :

```
var colonne0=tabLigne.insertCell(0);
colonne0.innerHTML="Ferrari F40";
```

Le tableau HTML après cette insertion se présente comme suit :



Le code JavaScript de la fonction `ajouterLigneFinTableau` est assez proche du précédent :

```
/* Fonction ajouterLigneFinTableau */
function ajouterLigneFinTableau()
{
    /*
    Affichage du nombre de lignes du tableau
    */
    var nombreLignes =
    parseInt(document.getElementById('monTableau').rows.length);
    /* alert("Nombre de ligne(s) du tableau : " + nombreLignes); */

    /*
    Ajout d'une ligne vide après la dernière ligne
    actuelle du tableau
    */
    var tabLigne=document.getElementById('monTableau')
    .insertRow(nombreLignes);

    /*
    Ajout d'une colonne 0 (Modèle) dans la lignée créée
    et stockage d'une valeur
    */
    var colonne0=tabLigne.insertCell(0);
```

```
colonne0.innerHTML="Porsche 959";

/*
Ajout d'une colonne 1 (Puissance) dans la lignée créée
et stockage d'une valeur
*/
var colonne1=tabLigne.insertCell(1);
colonne1.innerHTML="450";
}
```

Il faut dans un premier temps déterminer le nombre de lignes présentes dans le tableau HTML :

```
var nombreLignes =
parseInt(document.getElementById('monTableau').rows.length);
```

Puis ajoutons une ligne en fin de tableau, toujours via la méthode `insertRow` :

```
/*
Ajout d'une ligne vide après la dernière ligne
actuelle du tableau
*/
var tabLigne=document.getElementById('monTableau')
.insertRow(nombreLignes);
```

Il reste ensuite à placer un contenu dans les deux colonnes (comme précédemment pour la première ligne). Pour la seconde colonne (numérotée 1), correspondant à la puissance, le code est :

```
var colonne1=tabLigne.insertCell(1);
colonne1.innerHTML="450";
```

Le tableau HTML après cette insertion se présente désormais comme suit :



TABLE\_04 - Mozilla Firefox

Fichier Édition Affichage Historique Marque-pages Outils ?

TABLE\_04 +

127.0.0.1:8020/ENI\_JavaScript\_DOM\_Table\_Object/TABLE\_04.htm

## Editions ENI - JavaScript - TABLE\_04

Modèle	Puissance
Ferrari F40	478
Ferrari 512 BB	380
Lamborghini Miura	385
Porsche 964 3.6	360
Porsche 959	450

---

Ajout de lignes

Ajout d'une ligne (début de tableau)

Ajout d'une ligne (fin de tableau)

---

Suppression de lignes

N° de ligne à supprimer (n° à partir de 1) :

Suppression d'une ligne

Il reste à voir comment supprimer une ligne de tableau HTML pour laquelle le numéro de ligne est fourni au clavier au travers d'un champ `<input type="text" ...>`. Dans notre script HTML (section `<body>`) un second formulaire est prévu à la suite de celui ayant servi pour nos ajouts de lignes. Son code HTML est le suivant :

```
<!-- Formulaire de suppression de lignes -->
<br /><hr /><br />
Suppression de lignes<br /><br />
<form name="formulaireSuppression">
  N° de ligne à supprimer (n° à partir de 1) :
  <input
    type="text"
    id="numLigne"
    value=""
  /><br /><br />
  <input
    type="button"
    id="boutonSupprimerLigneTableau"
    value="Supression d'une ligne"
    onclick="supprimerLigneTableau()"
  />
</form>
```

Ce formulaire est sans surprise, il comprend un champ de saisie (identifiant numLigne) et un bouton d'appel à la fonction supprimerLigneTableau.

Étudions le code de cette fonction :

```
/* Fonction supprimerLigneTableau */
function supprimerLigneTableau()
{
    /*
    Affichage de contrôle du numéro de la ligne à supprimer
    dans le tableau
    */
    var numeroLigne =
    parseInt(document.getElementById('numLigne').value);
    /* alert("N° de ligne à supprimer : " + numeroLigne); */
    /*
    Affichage du nombre de lignes du tableau
    */
    var nombreLignes =
    parseInt(document.getElementById('monTableau').rows.length);
    /* alert("Nombre de ligne(s) du tableau : " + nombreLignes); */
    /*
    Suppression de la ligne
    */
    if ((numeroLigne >= 1) && (numeroLigne < nombreLignes))
    {
        /* Suppression */
        document.getElementById('monTableau').deleteRow(numeroLigne);
    }
    else
    {
        /* N° de ligne erroné */
        alert("N° de ligne erroné");
    }
}
```

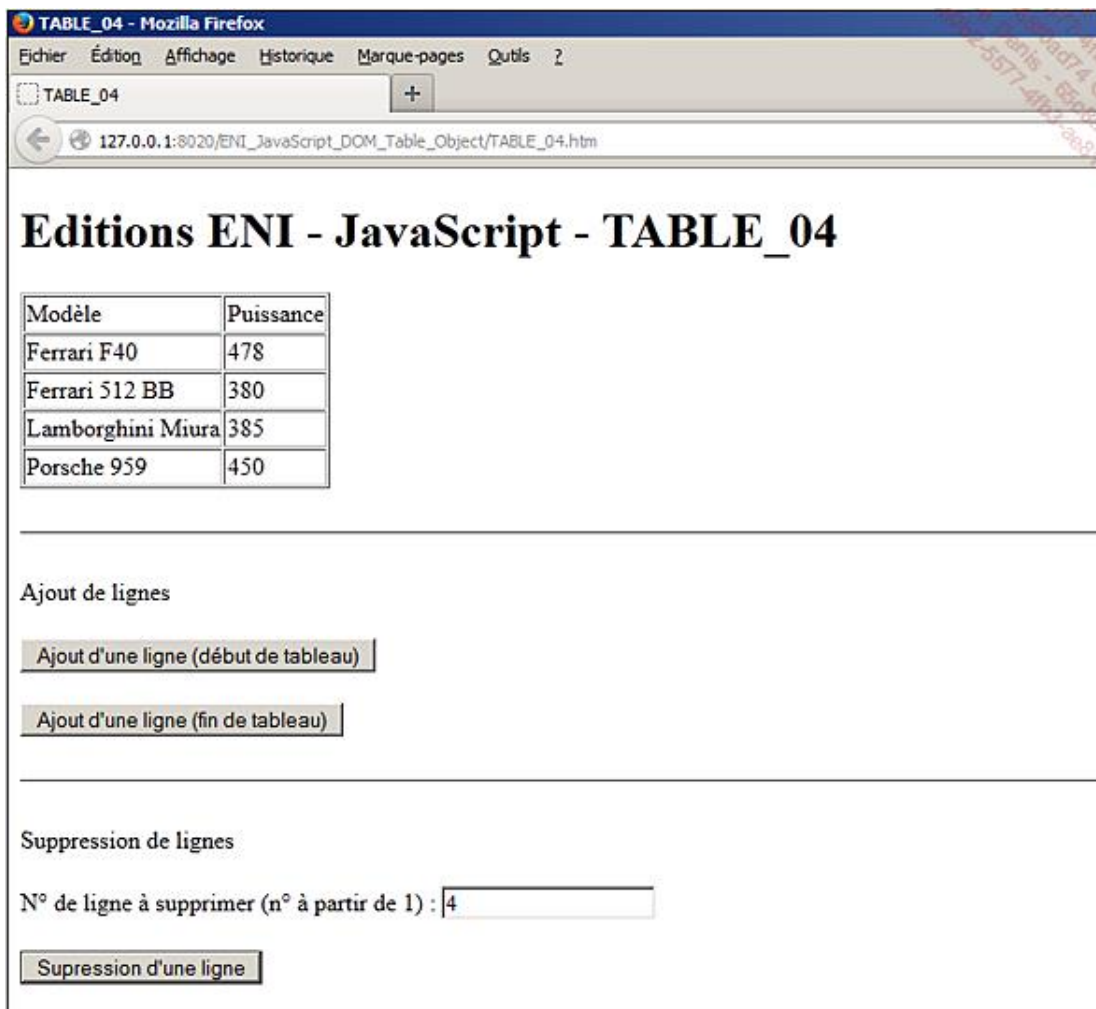
Pour qu'il soit possible d'annoncer un numéro de ligne éventuellement incorrect (inférieur à 1 ou supérieur ou égal au nombre de lignes) le nombre de lignes du tableau est déterminé :

```
var nombreLignes =
parseInt(document.getElementById('monTableau').rows.length);
```

La suppression s'effectue par l'intermédiaire de la méthode deleteRow, comme suit :

```
document.getElementById('monTableau').deleteRow(numeroLigne);
```

Si la suppression de la ligne n°4 est demandée, l'affichage du tableau devient :



### Exemple 5

Terminons nos exemples par un script dans lequel nous allons colorer les différentes lignes du tableau HTML via JavaScript :

- la ligne de titre en bleu,
- les lignes de données de numéro pair en argent (silver),
- les lignes de données de numéro impair en blanc (white).

Nous utiliserons toujours notre tableau HTML habituel (une ligne de titre et trois lignes de données) :

Modèle	Puissance
Ferrari 512 BB	380
Lamborghini Miura	385
Porsche 964 3.6	360

Le code HTML du formulaire (section <body>) comprenant les boutons d'appel aux fonctions JavaScript a le code suivant :

```
<!-- Formulaire -->
```

```

<br /><br />
<form>
    Coloration du titre (orange) et des lignes
    (alternativement argent/blanc) du tableau<br /><br />
    <input
        type="button"
        id="boutonColorerTitreTableau"
        value="Coloration du titre"
        onclick="colorerTitreTableau()"
    /><br />
    <input
        type="button"
        id="boutonColorerLignesTableau"
        value="Coloration des lignes"
        onclick="colorerLignesTableau()"
    />
</form>

```

Un premier bouton `<input type="button" ... >` d'identifiant `boutonColorerTitreTableau` appelle une fonction de nom `colorerTitreTableau` et un second bouton `<input type="button" ... >` d'identifiant `boutonColorerLignesTableau` appelle une fonction de nom `colorerLignesTableau`.

Analysons le code de la fonction `colorerTitreTableau` (section HTML `<head>`) :

```

/* Fonction colorerTitreTableau */
function colorerTitreTableau()
{
    /*
    Affichage du nombre de colonnes du tableau
    (par détermination du nombre de cellules de la ligne de titre)
    */
    var nombreColonnes =
    parseInt(document.getElementById('monTableau').rows[0].cells.length);
    alert("Nombre de colonne(s) du tableau : " + nombreColonnes);
    /*
    Mise en tableau mémoire (tabLigne) des cellules
    de la ligne n°0 (titre)
    */
    var tabLigne=
    document.getElementById('monTableau').rows[0].cells;
    /*
    Modification de la couleur des cellules
    de la ligne n°0 (titre)
    */
    for (numeroColonne=0; numeroColonne<nombreColonnes; numeroColonne++)
    {
        /*
        Coloration de la cellule en cours
        en orange (code RGB #FFA500)
        */
        tabLigne[numeroColonne].bgColor = "#FFA500";
    }
}

```

Le nombre de colonnes est déterminé dans un premier temps (en calculant le nombre de cellules de la ligne n°0) :

```
var nombreColonnes =  
parseInt(document.getElementById('monTableau').rows[0].cells.length);
```

Ensuite, également comme nous l'avons déjà fait dans des exemples précédents, les cellules de la ligne n°0 (ligne de titre) sont placées dans un tableau mémoire (tabLigne), comme suit :

```
var tabLigne=  
document.getElementById('monTableau').rows[0].cells;
```

Il reste ensuite à utiliser un traitement itératif (boucle for dans notre code) pour balayer les cellules de ce tableau pour les colorer (en orange pour le titre), comme suit :

```
tabLigne[numeroColonne].bgColor = "#FFA500";
```

Terminons par le code de la fonction colorerLignesTableau :

```
/* Fonction colorerLignesTableau */  
function colorerLignesTableau()  
{  
    /*  
    Affichage du nombre de lignes du tableau  
    */  
    var nombreLignes =  
    parseInt(document.getElementById('monTableau').rows.length);  
    alert("Nombre de ligne(s) du tableau : " + nombreLignes);  
    /*  
    Affichage du nombre de colonnes du tableau  
    (par détermination du nombre de cellules de la ligne de titre)  
    */  
    var nombreColonnes =  
    parseInt(document.getElementById('monTableau').rows[0].cells.length);  
    alert("Nombre de colonne(s) du tableau : " + nombreColonnes);  
    /*  
    Coloration des lignes (hors titre)  
    alternativement en gris/blanc  
    */  
    for (i=1; i<nombreLignes; i++)  
    {  
        /*  
        Mise en tableau mémoire (tabLigne)  
        des cellules de la ligne en cours  
        */  
        var tabLigne =  
        document.getElementById('monTableau').rows[i].cells;  
        /*
```

```

    Modification de la couleur des cellules
    de la ligne en cours
    */
    for (numeroColonne=0; numeroColonne<nombreColonnes; numeroColonne++)
    {
        if (i%2 == 0)
        {
            /*
            Coloration de la cellule en cours
            en argent (code RGB #C0C0C0)
            */
            tabLigne[numeroColonne].bgColor = "silver";
        }
        else
        {
            /*
            Coloration de la cellule en cours
            en blanc (code RGB #FFFFFF)
            */
            tabLigne[numeroColonne].bgColor = "white";
        }
    }
}

```

Là encore, beaucoup de points communs avec les exemples précédents. Le nombre de lignes du tableau HTML est déterminé :

```

var nombreLignes =
parseInt(document.getElementById('monTableau').rows.length);

```

Le nombre de colonnes est aussi calculé comme dans la fonction `colorerTitreTableau`.

Ensuite une boucle `for` balaie les lignes du tableau HTML en débutant le parcours à la ligne n°1 (première ligne des données) :

```

for (i=1; i<nombreLignes; i++)

```

La ligne en cours est stockée dans le tableau mémoire `tabLigne` déjà rencontré dans la fonction `afficherTitreTableau` :

```

var tabLigne =
document.getElementById('monTableau').rows[i].cells;

```

Une deuxième boucle `for` (identique à celle de la fonction `afficherTitreTableau`) parcourt les deux colonnes du tableau. À chaque tour de boucle un test de parité (`i%2 == 0`) est fait et pour les lignes de numération paire la couleur retenue (via la propriété `bgColor`) est la couleur argent (`silver`) :

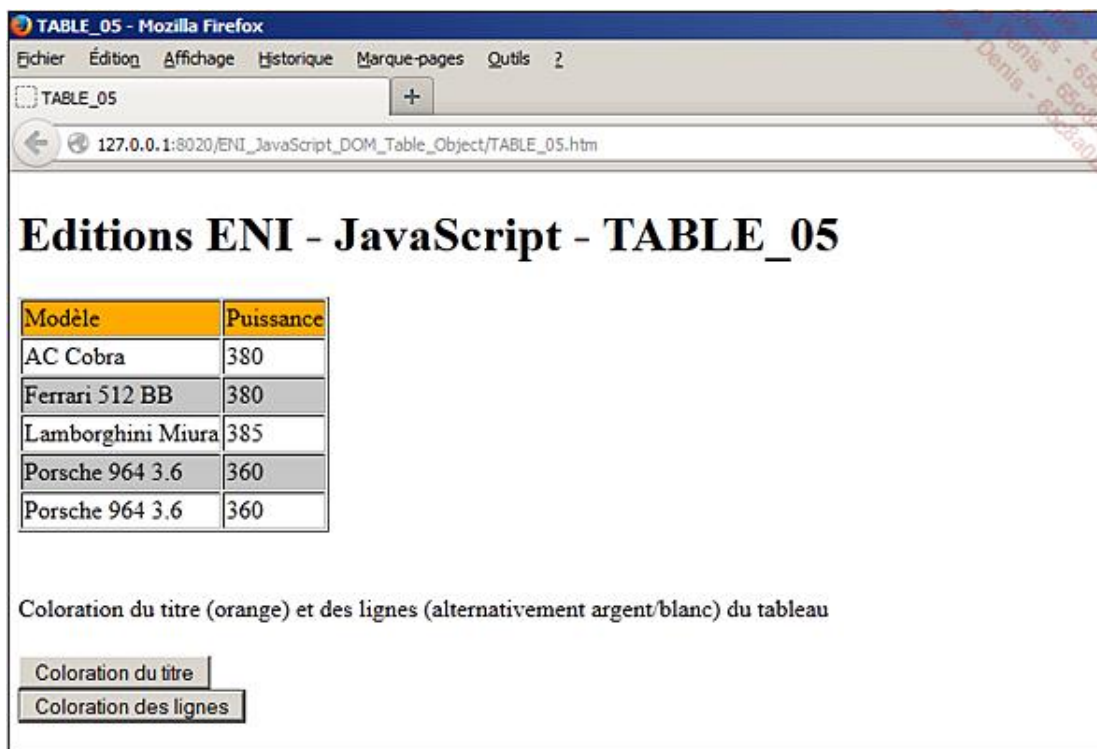
```
tabLigne[numeroColonne].bgColor = "silver";
```

et la couleur blanche (white) pour les lignes de numérotation impaire :

```
tabLigne[numeroColonne].bgColor = "white";
```

- Pour les codes des couleurs, il est possible d'utiliser le pseudonyme des couleurs (orange, silver, white...) ou le code RGB associé à la couleur (# suivi de six codes hexadécimaux, par exemple #FFA500 pour la couleur orange).

L'affichage du tableau se présente comme suit après application des colorations sur le titre et les lignes de données :



**TABLE\_05 - Mozilla Firefox**

Fichier Édition Affichage Historique Marque-pages Outils ?

TABLE\_05 +

127.0.0.1:8020/ENI\_JavaScript\_DOM\_Table\_Object/TABLE\_05.htm

## Editions ENI - JavaScript - TABLE\_05

Modèle	Puissance
AC Cobra	380
Ferrari 512 BB	380
Lamborghini Miura	385
Porsche 964 3.6	360
Porsche 964 3.6	360

Coloration du titre (orange) et des lignes (alternativement argent/blanc) du tableau

Coloration du titre

Coloration des lignes