


Variables (déclaration et typage)

 Pour faciliter le repérage des exercices JavaScript, la numérotation vue dans le chapitre Développement à partir d'algorithmes sera conservée.

1. Exercice n°2 : Surfaces de cercles

Pour débiter en douceur notre apprentissage, reprenons l'algorithme de l'exercice n°2 (Surfaces de cercles) vu au chapitre Développement à partir d'algorithmes. Exceptionnellement pour ce premier exercice, l'algorithme sera rappelé.

Sujet

Calculer (et afficher à l'écran) la surface de deux cercles de rayons prédéterminés (5.5 mètres et 3.5 mètres par exemple) ainsi que de la différence entre ces deux surfaces

Corrigé en langage descriptif algorithmique

Début

```
Réel RAYON1, RAYON2, PI, SURFACE1, SURFACE2, DIFFERENCE
RAYON1 <- 5.5
RAYON2 <- 3.5
PI <- 3.14
SURFACE1 <- PI * RAYON1 * RAYON1
SURFACE2 <- PI * RAYON2 * RAYON2
DIFFERENCE <- SURFACE1 - SURFACE2
Ecrire("Surface 1 = ", SURFACE1, Alaligne, "Surface 2 = ", SURFACE2, Alaligne, "Différence = ",
DIFFERENCE)
```

Fin

Corrigé en JavaScript

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">

<!--
NOM DU SCRIPT : CH3_2.htm
REALISATION INFORMATIQUE : Christian VIGOUROUX
DATE DE CREATION : 01/01/2014
DATE DE DERNIERE MODIFICATION : 01/01/2014
OBJET : Calcul (et affichage) de la surface de deux cercles de rayons
        prédéterminés (5.5 mètres et 3.5 mètres par exemple)
        ainsi que de la différence entre ces deux surfaces
-->

<!-- 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>CH3_2</title>

</head>

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

    <!-- Titre du traitement -->
    <h1>Editions ENI - JavaScript - Exercice CH3_2</h1>

    <!-- Début script JavaScript -->
    <script>

        /* Affichage du nom du script */
        alert("Exercice CH3_2");

        /* Déclaration de variables locales */
        var rayon1, rayon2, pi, surfacel, surface2, difference;

        /* Initialisation des variables */
        rayon1 = 5.5;
        rayon2 = 3.5;
        pi = 3.14;

        /* Calculs */
        surfacel = pi * rayon1 * rayon1;
        surface2 = pi * rayon2 * rayon2;
        difference = surfacel - surface2;

        /* Affichage des résultats */
        document.write("Surface 1 : " + surfacel + "<br />");
        document.write("Surface 2 : " + surface2 + "<br />");
        alert("Différence : " + difference);

    </script>

    <!-- Affichage du code source -->
    <br /><br /><br />
    <center >
    <a href="JavaScript:window.location=
'view-source:'+window.location">
        Code source
    </a>
    </center>

</body>

</html>

```

Dans le script HTML présenté, le code JavaScript a été intégré dans la section `<body> ... </body>`. Nous verrons ultérieurement que le code JavaScript aurait pu être placé à d'autres endroits du code HTML. En résumé, voici comment positionner le code JavaScript dans un développement Web en HTML :

- Cas 1 : dans un jeu de balises `<script> ... </script>` dans la section `<body> ... </body>` du document HTML principal, ce qui est le cas dans le script commenté.
- Cas 2 : dans un jeu de balises `<script> ... </script>` dans la section `<head> ... </head>` du document HTML principal, ce qui sera le cas pour les fonctions personnelles que nous développerons ultérieurement.
- Cas 3 : dans un fichier externe (d'extension `.js`) pour les fonctions que l'on souhaite pouvoir appeler depuis de multiples scripts Web.

N'insistons pas pour l'instant sur les cas 2 et 3, ces modes d'implémentation du code JavaScript seront vus du point de vue syntaxique ultérieurement dans ce livre. Des exemples seront aussi présentés à cette occasion.

Le code HTML DOCTYPE ci-après :

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
```

est la déclaration du type de document. Elle sert à préciser au navigateur le type de code HTML dans lequel le script (page) HTML est écrit, cela peut être HTML-3.2, HTML-4, XHTML...

Dans sa formulation détaillée, cette directive a la structure suivante :

```
<!DOCTYPE HTML PUBLIC "type_HTML" "adresse_DTD">
```

où :

- `type_HTML` est l'identificateur de la version du HTML utilisé,
- `adresse_DTD` donne l'URL d'un *Document Type Definition* (DTD), c'est-à-dire d'un document précisant les propriétés de chaque élément (balises et attributs) pour ce type de code HTML.

La séquence HTML suivante est un exemple typique de commentaires :

```
<!-- NOM DU SCRIPT : CH3_2.htm
    REALISATION INFORMATIQUE : Christian VIGOUROUX
    DATE DE CREATION : 01/01/2014
    DATE DE DERNIERE MODIFICATION : 01/01/2014
    OBJET : Calcul (et affichage) de la surface de deux cercles de rayons
            prédéterminés (5.5 mètres et 3.5 mètres par exemple)
            ainsi que de la différence entre ces deux surfaces
-->
```

Rappelons que les commentaires sont des explications intégrées au script qui ne seront pas interprétées par le navigateur. Ces lignes sont par contre fondamentales dans la compréhension des différentes séquences de code HTML du script. Ici le commentaire a tout simplement pour objectif de rappeler le traitement effectué par le script.

Nous considérons que le lecteur a tout de même des connaissances minimales en HTML et les balises <html>, </html>, <head>, </head>, <title>, </title>, <body> et </body> ne sont pas commentées ici. Le cas échéant, vous pourrez accéder à de multiples ressources sur le langage HTML sur Internet.

Attardons-nous sur la balise meta HTTP-equiv :

```
<meta HTTP-equiv="Content-Type" content="text/html; charset=utf-8" />
```

La syntaxe générale de cette balise est :

```
<meta HTTP-equiv="mot clé ici" content="valeur ici" />
```

Cette balise va permettre :

- de signaler l'encodage de la page Web par l'intermédiaire du mot clé Content-Type. Par exemple pour un encodage au format UTF-8 :

```
<meta HTTP-equiv="Content-Type" content="text/html; charset=UTF-8" />
```

- d'indiquer qu'il ne faut pas que la page soit enregistrée dans le cache du navigateur (la page ne sera pas sauvegardée dans les fichiers temporaires de l'internaute) par l'intermédiaire du mot clé pragma :

```
<meta http-equiv="pragma" content="no-cache" />
```

- de signaler qu'après une date d'expiration la page ne sera plus disponible, ceci par l'intermédiaire du mot clé expires :

```
<meta HTTP-equiv="expires" content="Tue, 20 Aug 1996 14:25:27 GMT" />
```

- de requérir l'actualisation de la page toutes les n secondes par le mot clé refresh. Par exemple, pour une réactualisation toutes les 15 secondes :

```
<meta http-equiv="refresh" content="15" />
```

- de rediriger l'internaute vers une autre page après n secondes par l'intermédiaire du mot clé refresh. Par exemple, une redirection vers la page yahoo.fr sera obtenue après l'affichage du script en cours pendant deux secondes :

```
http-equiv="refresh" content="2;URL=http://www.yahoo.fr/" />
```

Intéressons-nous maintenant à notre code JavaScript. Le script débute par une directive <script> (précédé d'un commentaire exprimé en HTML) :

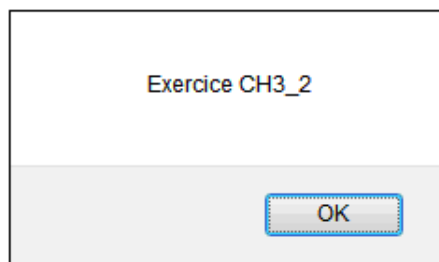
```
<!-- Début script JavaScript -->  
<script>
```

La commande alert (il s'agit, nous le verrons ultérieurement, d'une méthode) affiche un message d'annonce de

l'intitulé de l'exercice sous la forme d'une fenêtre "pop up" :

```
alert("Exercice CH3_2");
```

La fenêtre se présente comme ceci à l'exécution du script :



➤ Vous aurez sans doute noté que chaque instruction JavaScript est terminée par un point-virgule.

Ensuite un jeu de variables est déclaré. Ces variables correspondent bien évidemment à celles vues dans le cadre de l'algorithme. Il n'y a pas de raisons fondamentales de modifier la syntaxe de celles-ci. Tout au plus des changements de casse peuvent être envisagés pour respecter une normalisation des noms des variables, méthodes... (normalisation camelCase par exemple).

```
var rayon1, rayon2, pi, surfacel, surface2, difference;
```

Les déclarations se sont ici faites sur une seule ligne avec le préfixe `var` qui n'est pas une annonce de type. En JavaScript, le type a priori n'existe pas, les variables sont associées à un type lors des affectations de contenu.

Pour les initialisations, le signe `=` est utilisé. Pour le premier rayon, cela donne :

```
rayon1 = 5.5;
```

➤ Le signe d'affectation `=` utilisé ici ne doit pas être confondu avec le signe `==` qui servira à effectuer des comparaisons en égalité.

Les calculs ne posent pas de difficultés particulières, à titre d'exemple pour la première surface, le code est :

```
surfacel = pi * rayon1 * rayon1;
```

Passons enfin à l'affichage de résultat. La commande `document.write` permet d'afficher le résultat des calculs dans la fenêtre du navigateur. À titre indicatif pour la première surface, cela donne :

```
document.write("Surface 1 : " + surfacel + "<br />");
```

➤ `document.write` est une syntaxe orientée objet dans laquelle la méthode `write` est appliquée à un objet `document`. Nous reviendrons longuement sur la programmation orientée objet (POO) dans le chapitre Approche "objet" en JavaScript.

Les différents éléments à afficher sont séparés les uns des autres par un `+` et il n'y a pas de contraintes de conversion des valeurs numériques en chaînes de caractères comme cela est souvent le cas dans d'autres langages de programmation.

➤ La chaîne `
` montre qu'il est possible d'intégrer des balises HTML dans un affichage JavaScript. Ici le `
` provoque un changement de ligne avant l'affichage suivant.

L'affichage de la différence entre les deux surfaces est assuré par la méthode `alert` que nous connaissons déjà :

```
alert("Différence : " + difference);
```

➤ Les éléments affichés par la méthode `alert` s'organisent comme avec la méthode `document.write`.

La section écrite en JavaScript se termine par `</script>` :

```
</script>
```

La séquence suivante (non commentée dans le détail ici) permettra à l'internaute de visualiser le code HTML/JavaScript de la page par un clic sur un lien hypertexte :

```
<!-- Affichage du code source -->
<br /><br /><br />
<center >
<a href="JavaScript:window.location='view-source:'+window.location">
    Code source
</a>
</center>
```

2. Exercice n°3 : Surface et volume d'une sphère

Sujet

Calculer et afficher la surface et le volume d'une sphère dont la valeur du rayon est saisie au clavier

La nouveauté par rapport à l'exercice précédent (Exercice n°2) est la saisie du rayon au clavier.

Corrigé en JavaScript

Le corrigé est, ici encore, reproduit intégralement (ce ne sera plus le cas dans les exemples ultérieurement).

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
```

```

<!--
NOM DU SCRIPT : CH3_3htm
REALISATION INFORMATIQUE : Christian VIGOUROUX
DATE DE CREATION : 01/01/2014
DATE DE DERNIERE MODIFICATION : 01/01/2014
OBJET : Calcul (et affichage) de la surface et du volume d'une sphère
        dont le rayon est saisi au clavier
-->

<!-- 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>CH3_3</title>

    </head>

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

        <!-- Titre du traitement -->
        <h1>Editions ENI - JavaScript - Exercice CH3_6</h1>

        <!-- Début script JavaScript -->
        <script>

            /* Affichage du nom du script */
            alert("Exercice CH3_3");

            /* Déclaration de variables locales */
            var pi, rayon, surface, volume;

            /* Initialisation des variables */
            pi = 3.14;
            rayon = prompt("Rayon de la sphère :");

            /* Calculs */
            surface = 4 * pi * rayon * rayon;
            volume = surface * rayon / 3;

            /* Affichage des résultats */
            document.write("Rayon de la sphère : "
            + rayon + "<br />");
            document.write("Surface : " + surface
            + "<br />Volume : " + volume);
        </script>
    </body>
</html>

```

```

</script>

<!-- Affichage du code source -->
<br /><br /><br />
<center>
<a href="JavaScript:window.location=
'view-source:'+window.location">
    Code source
</a>
<center>

</body>

</html>

```

Commentaires du code JavaScript

Au niveau des initialisations, la variable `pi` prend la valeur 3.14 (valeur approximative) et la variable `rayon` est initialisée par une saisie au clavier (méthode `prompt`). Le seul paramètre de cette méthode est un libellé.

```

/* Initialisation des variables */
pi = 3.14;
rayon = prompt("Rayon de la sphère :");

```

3. Exercice n°4 : Nombre de lettres d'un mot

Sujet

Écrire un algorithme permettant la saisie au clavier d'un nom et d'afficher le nombre de lettres

Corrigé (partiel) en JavaScript

```

/* Déclaration de variables locales */
var nom;

/* Saisie du nom au clavier */
nom = prompt("Nom :");

/* Affichage du résultat */
document.write("Le nom " + nom + " contient " + nom.length + " lettre(s)");

```

Commentaires du code JavaScript

Anticipons sur l'approche "Programmation orientée objet" (POO) en JavaScript. La variable `nom` est de type `String`. Il est possible sur ce type d'objet d'utiliser des méthodes spécifiques ou d'accéder à des valeurs de propriétés comme `length` (nombre de caractères).

La valeur de la propriété `length` est ici directement affichée par l'appel à la méthode `document.write`. Il aurait aussi été envisageable de déclarer une variable mémoire supplémentaire, comme `longueurNom` par exemple et de lui affecter la valeur de `nom.length` avant également de l'afficher par le `document.write`.

4. Exercice n°5 : Détermination des initiales

Sujet

Écrire un algorithme qui permettra l'extraction des initiales d'une personne dont le prénom et le nom seront saisis au clavier (exemple CV pour Christian VIGOUROUX)

Corrigé (partiel) en JavaScript

```
/* Déclaration de variables locales */
var prenom, nom, initiale_prenom, initiale_nom;

/* Saisie du prénom */
prenom = prompt("Prénom :");

/* Saisie du nom */
nom = prompt("Nom :");

/* Détermination des initiales */
/* NB : La numérotation des lettres dans une chaîne débute à 0 */
initiale_prenom = prenom.charAt(0);
initiale_nom = nom.substring(0, 1);

/* Affichage des initiales */
document.write("Les initiales de " + prenom + " " + nom + " sont " +
initiale_prenom + initiale_nom);
```

Commentaires du code JavaScript

Alors que nous avons utilisé la propriété `length` dans l'exemple précédent, nous employons ici la méthode `charAt` pour déterminer l'initiale du prénom et la méthode `substring` pour l'initiale du nom.

La méthode `charAt` avec le paramètre 0 extrait le premier caractère de la variable `prenom`, c'est-à-dire l'initiale (dans une chaîne de caractères la numérotation des caractères débute à zéro).

La méthode `substring` requiert deux paramètres, le premier indique la position du premier caractère à extraire (0 comme pour la méthode `charAt`), le second le nombre de caractères à extraire (1 ici pour une seule lettre, c'est-à-dire l'initiale).

La méthode `substring` peut donc remplacer la méthode `charAt` dans tous les cas.