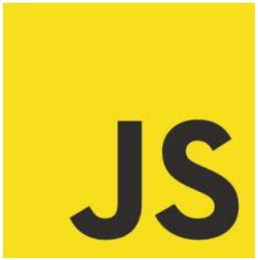
El Mamoun SOUIDI

JavaScript

Cours et exercices corrigés

Module Programmation Web 2 : Langage Javascript
en S4 Tronc commun : Informatique (MIP)



Année 2024

Table des matières

1

Introduction à Javascript

```
<script type="text/javascript">
<!--
. . . code de javascript . . .
-->
</script>
```

Dans ce chapitre, nous ...

Pour apprendre JavaScript il faut être familiarisé avec le langage HTML et des notions du langage CSS. En plus l'utilisation d'un éditeur de texte et d'un navigateur.

Un script est un (petit) programme ...

Exemple d'éditeurs : gedit sous Linux, Sublime, Visual Studio, Notepad, TextPad, etc.

Exemple de navigateurs : Google Chrome, Mozilla Firefox, Microsoft Edge ou autres

On peut aussi utiliser des environnement intégré de développement comme NetBeans, Eclipse, Visual Studio, and Brackets et autres.

1.1 Architecture client serveur

Une architecture client/serveur est un réseau formé de deux types d'ordinateurs : les clients et les serveurs.

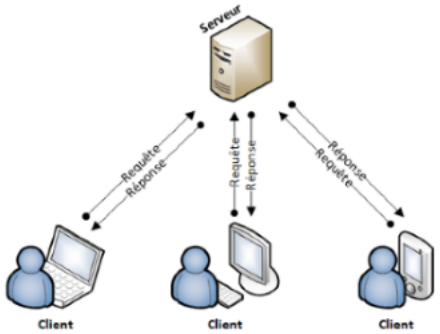


FIGURE 1.1 – Architecture client serveur

Un client est un ordinateur équipé de programmes, il est actif : il envoie des requêtes au serveur pour demander des services.

Un serveur est une machine généralement puissante en terme de capacités d'entrée-sortie, il est passif, à l'écoute et prêt à répondre aux requêtes envoyées par des clients. Un serveur est généralement capable de servir plusieurs clients à la fois.

Un serveur peut être spécialisé en serveur d'applications, de fichiers, de terminaux, ou encore de messagerie électronique.

Les programmes de machines clientes communiquent avec les programmes de machines de type serveurs.

Par exemple le navigateur Web d'un client qui envoie une requête (le contenu d'une page Web) à un serveur Web qui lui renvoie la réponse.

un serveur de base de données permet aux clients de récupérer des données stockées dans une base de données.

Un internaute connecté à l'internet par son ordinateur et un navigateur web est le client, le serveur est constitué par le ou les ordinateurs contenant les documents visités.

Les courriels sont envoyés et reçus par des clients et gérés par un serveur de messagerie.

L'internet est une architecture client serveur. Lorsque vous visitez un site web vous êtes client. Et les documents que vous visitez sont dans un serveur. Un serveur est un grand ordinateur équipé de logiciels qui répondent aux requêtes émis par le navigateur.

1.2 Langage client/serveur

Pour accéder à données côté serveur, nous avons besoin de lancer des applications. Comme langage de programmation pour réaliser ces applications PHP, ASP, JSP et bien

d'autres.

Lorsqu'un document HTML est demandé par le navigateur, le serveur le lui envoie. C'est le navigateur qui se charge alors de l'interprétation du contenu HTML CSS et JavaScript . Par contre si ce document HTML contient du code PHP alors c'est le serveur qui l'interprète et envoie le résultat sous forme de code HTML au navigateur.

PHP est un langage de programmation dont le code source s'insère aussi dans le code HTML. Par contre, ce code PHP est interprété par le serveur. PHP permet par exemple de traiter, d'enregistrer et d'accéder à des données dans un serveur. Il permet aussi de traiter des données à distance.

Un langage côté serveur est exécuté par le serveur. Il peut être utilisé pour le traitement de données reçues via un navigateur. Comme exemples de langage côté serveur on site PHP, Perl, Python, Ruby, Java. Lorsque Node.js est arrivé, il a permis d'utiliser JavaScript côté serveur en plus du côté client. Ce livre couvre aussi Node.js.

1.3 Fonctionnement de JavaScript

Les langages HTML et CSS offrent des fonctionnalités limitées d'interactivité avec l'utilisateur ; comme les liens, le survole d'une zone et les pages qu'ils réalisent sont statiques. Le langage JavaScript vient les compléter pour rendre les pages Web dynamiques et interactives. Il permet entre autres de faire des calculs, interactifs avec l'utilisateur et émettre des commentaires, gérer le temps : au bout d'un temps précis émettre un spot publicitaire, afficher l'heure sur une page Web, faire défiler des images dans un ordre programmé, afficher des images aléatoirement, modifier le contenu d'une page Web en fonction du temps, faire des traitements côté client par le navigateur et l'ordinateur du client pour soulager les serveurs. Le contrôle de saisie, un champ où on ne peut saisir que les nombres, ou que les lettres etc. Un autre exemple par exemple JavaScript peut éviter à un serveur de messagerie de vérifier si le mot de passe et le login ont été saisis et que l'adresse électronique saisie contient le `@` et le point ou non.

JavaScript peut aussi être utilisé pour écrire des programmes de grande taille : programme de chiffrement et de déchiffrement de données, etc...

Le langage JavaScript est un langage de programmation. Il permet d'écrire des programmes qui font des tâches bien précise. Le code source est inséré dans le code HTML grâce à la balise `<script>...</script>`. Lorsque qu'on envoie une requête demandant une page web contenant du code source JavaScript , le serveur renvoie le code source HTML et JavaScript au navigateur qui se charge de les interpréter. Par contre si le

fichier HTML contient du code PHTP par exemple, c'est le serveur qui se charge de son interprétation et envoie au navigateur uniquement le résultat du traitement au format HTML

```
<html>
  <head> ... </head>
  <body>
    code HTML
    ...
    <script type="text/javascript">
    . . . code de JavaScript . . .
    interprété par le navigateur
    </script>
    ...code HTML
    <?php
    . . . code PHP . . .
    interprété par le serveur et le résultat envoyé au format HTML
    ?>
    code HTML
  </body>
</html>
```

1.4 Langages du Web

Pour créer un site web professionnel il faut au moins maîtriser quatre langages :

– HTML HyperText Markup language, langage déclaratif de balisage standard pour créer des pages web.

– CSS Cascading StyleSheet langage déclaratif de feuille de style en cascade. Il permet notamment d'améliorer la mise en forme de document HTML et de séparer totalement le contenu d'un document de sa mise en forme.

– JavaScript langage de programmation côté client. Ce sont les navigateurs qui interprète le code JavaScript . Ce qui permet de soulager les serveurs. Vu sa popularité, tous les navigateurs sont équipés d'interpréteurs JavaScript.

– au moins un langage de programmation côté serveur. Comme PHP, ASP, JSP ou autres.

1.5 Qu'est ce que **JavaScript** ?

JavaScript est un langage de programmation qui permet d'apporter du dynamisme et de l'interactivité au langage HTML qui est statique.

JavaScript a été initialement élaboré en mai 1995 en 10 jours, sous le nom Mocha par Brendan Eich de chez Netscape pour son navigateur Web, Netscape Navigator.

En septembre 1995, dans Netscape Navigator 2.0 beta, Mocha fût nommé LiveScript. et En décembre 1995, dans Netscape Navigator 2.0 beta 3 LiveScript prit son nom définitif JavaScript .

JavaScript a été standardisé deux fois :

– ECMA-262 par Ecma International [1].

– ISO/IEC 16262 par l'Organisation internationale de normalisation (ISO) et la Commission électrotechnique internationale (CEI).

sous le nom ECMAScript et non JavaScript , car ce dernier nom a été une marque déposée de la société Sun devenue par la suite Oracle

En principe, JavaScript et ECMAScript désignent la même chose. Mais parfois :

– Le terme JavaScript fait référence au langage et à ses implémentations.

– Le terme ECMAScript fait référence au standard de langage et aux versions linguistiques.

Depuis 2015, des versions majeures sont publiées chaque mois de juin. Voir la liste [2] des versions depuis 1997 et les modifications apportées par une version par rapport à sa prédécesseur.

TC39 est le comité qui développe JavaScript . Ses membres sont, à proprement parler, des compa- niés : Adobe, Apple, Facebook, Google, Microsoft, Mozilla, Opera, Twitter et autres.

Tous les deux mois, le TC39 organise des réunions. Les procès-verbaux de ces réunions sont publics. [3].

Et plus tard co-fondateur du projet Mozilla, de la Mozilla Foundation et de la Mozilla Corporation. JavaScript est reconnu par tous les navigateurs depuis 1996. Pour garder la compatibilité, et tout comme, le langage HTML ou CSS, JavaScript est standardisé en 1997 par le comité spécialisé, ECMA [4] et en 1998 par ISO-16262.

JavaScript a été créé à l'origine pour valider les entrées de formulaire.

1. ECMA, un acronyme pour European Computer Manufacturers Association
2. https ://en.wikipedia.org/wiki/ECMAScript_version_history
3. https ://github.com/tc39/notes
4. European Computer Manufactures Association est un organisme européen de standardisation pour les systèmes d'information et de communication.

Microsoft a de son côté développé, pour son navigateur Internet Explorer, en 1996, le langage JScript, similaire à JavaScript et correspondant également à un dialecte du langage ECMAScript. Il en est de même pour FireFox.

Ainsi, chaque navigateur supporte ECMAScript par l'intermédiaire de son propre dialecte. Le terme JavaScript désigne en fin de compte par abus de langage, tous ces dialectes.

ECMAScript est, en fin de compte, le nom standard et international des spécifications utilisées pour JavaScript.

JavaScript est un langage de script interprété, non typé et orienté objet. Son code est intégré dans le langage HTML. Il est différent du langage Java et sont à des finalités différentes. Sa syntaxe est proche de celle du langage C.

JavaScript est toujours utilisé conjointement avec HTML. Il ne peut être utilisé pour réaliser des applications indépendantes.

JavaScript permet d'exécuter des programmes du côté client (navigateur) et ainsi de soulager les serveurs qui hébergent les pages web visitées. En plus, l'exécution de JavaScript est rapide.

Par exemple JavaScript permet de vérifier qu'un champ n'a pa été saisi et en avertir l'utilisateur, au lieu de faire cette vérification par le serveur. Faire des calculs, afficher des messages au visiteur d'une page, modifier le contenu de page web, défiler une série d'images sur l'écran. Déclencher une réaction à une action de l'utilisateur, comme un clique, un survol de souris etc. JavaScript permet aussi de développer des applications Internet.

Cependant, JavaScript ne permet aucune confidentialité au niveau des codes, car celui-ci est accessible à tout visiteur de page Web. Il suffit alors d'afficher le code source [5] pour voir le code en JavaScript . En JavaScript et pour la sécurité de votre ordinateur, il est impossible de lire ou d'écrire dans un fichier.

le code source

JavaScript est un langage interprété. Le code source est interprété, par un logiciel qui s'appelle interpréteur intégré dans les navigateurs. L'interpréteur interprète les lignes du code source une par une.

5. avec FireFox cliquer sur le bouton droit de la souris puis choisir View Page Source

1.6 Caractéristiques du langage JavaScript

JavaScript est un langage à typage dynamique, il n'est pas nécessaire de déclarer le type d'une variable. En écriant dans un code JavaScript `x=5` alors x est considéré comme un nombre et si on écrit `x='Bonjou'` x est considéré comme une chaîne de caractère.

La syntaxe ressemble à celle du langage C

JavaScript possède des fonctionnalités orientées objet : notion d'objets, héritage, classes, etc.

La programmation orientée objet est un paradigme de programmation développé dans les années 1990. Voir Chap 7 pour plus de détails.

1.6.1 Langages interprété ou compilé

Le code source est d'abord transformé en un fichier binaire par un logiciel qui s'appelle compilateur. Le système d'exploitation va utiliser le code binaire et les données d'entrée pour calculer les données de sortie.

Dans un langage interprété, le même code source pourra être utilisé sous différentes plate-formes. Alors que pour un langage compilé, il faudra compiler le code source sous une plate-forme pour l'utiliser avec celle-ci. Dans un langage compilé, le programme est plus rapide que le même programme dans un langage interprété.

Le code binaire est indépendant du code source et du compilateur. Par contre, un programme interprété a besoin l'interpréteur à chaque lancement du programme.

Les langages C, C++, sont des langages interprétés. JavaScript , PHP, Python sont des langages compilés.

Java fonctionne autrement. C'est un langage compilé en langage intermédiaire byte-code, qui fonctionne sous toutes les platforme à condition que l'application JVM (Java Virtual Machine) soit installé.

1.6.2 JavaScript et les autres langages

Le langage JavaScript conserve la tête du classement RedMonk [6] version (juin 2021)

1. JavaScript, 2. Python, 2. Java, 4. PHP, 5. CSS, 5. C++, 5. C#, 8. TypeScript, 9. Ruby, 10. C.

6. RedMonk est une société d'analystes spécialisée dans les développeurs de logiciels https ://redmonk.com

1.6.3 JavaScript en évolution

JavaScript est utilisé en blockchain !

W3C établit non seulement des normes pour HTML et CSS, mais également pour la façon dont JavaScript interagit avec les pages Web à l'intérieur d'un navigateur Web.

Les mise à jour sont maintenant identifiés par année de sortie. ES2016, ES2017, ES2018, ES2019, et ES2020

1.6.4 TypeScript

Le langage TypeScript, rendu public en octobre 2012, de Microsoft est un sur-ensemble de JavaScript (tout code JavaScript correct peut être utilisé avec TypeScript) pour les grands projets. Il est libre et open source. Il supporte la spécification ECMAScript 6. La version 4.0 de TypeScript a été publiée en février 2021.

1.7 JQuery

JQuery, (http ://jquery.com/) est une bibliothèque JavaScript libre et multi-plateforme créée pour faciliter la programmation en javascript. JQuery est facile à maîtriser.

Ajax (Asynchronous JavaScript and XML) n'est pas un nouveau langage de programmation mais un ensemble de technologies qui permettent la mise à jour du contenu d'une page Web d'une manière rapide et sans chargement complet de celle ci. Ajax combine HTML, JavaScript, CSS, XML, DOM et XMLHttpRequest.

1.8 Que peut-on faire avec JavaScript ?

JavaScript est principalement utilisé pour interagir avec les utilisateurs.

Nous pouvons développer des applications avec JavaScript qui rivalisent avec la vitesse et la fonctionnalité des applications de bureau.

1.9 Mise en pratique de JavaScript

Pour les débutants en JavaScript il faut lire séquentiellement ce livre. Saisir le code et le tester. Vous pouvez modifier le code et continuer à tester. Cette démarche permet aussi me mémoriser le vocabulaire et la syntaxe du langage.

Pour pratiquer du JavaScript il suffit d'avoir un éditeur de texte et un navigateur. Tous les navigateurs (sauf les très anciens) sont équipés d'interpréteurs de code JavaScript .

Pour voir les erreurs, vous pouvez utiliser la console de votre navigateur. Par exemple avec FireFox Menu , More Tooms, Web developpers Tools, puis console Ou simplement appuyer sur la touche F12

Node.js est un environnement de serveur open source. Il permet d'interpréter le code JavaScript . Il fonctionne sous Windows, Linux, Unix, Mac OS X, etc.

Node.js permet la programmation hors les navigateurs.

Node.js permet la programmation côté serveur.

céder à la console : F12, Ctrl-Shift-I,

1.10 Console JavaScript de navigateur

Chaque navigateurs Web dispose d'une console, nommée aussi outils de développements, qui permet de tester des morceaux de code JavaScript . Pour accéder à la console de votre navigateurs :

– Firefox : Shift+Ctrl+K ou voir menu

ou il suffit de chercher sur le net "console nom de votre navigateur"

1.11 Contenu de ce livre

1.12 Exercices

Exercice 1. Définir une architecture cliet/serveur.

Exercice 2. Qu'est ce qu'un langage de programmation interprété ? compilé ?

Exercice 3. Que peut-on faire avec JavaScript ?

Exercice 4. Quel est la différence entre JavaScript , ECMAScript et TypeScript ?

Exercice 5. Qui développe JavaScript ?

Exercice 6. Comment et où place-t-on le code source JavaScript ?

Dans ce chapitre nous apprenons les bases du langage JavaScript . Puis nous écrirons des script simples d'affichage et de lecture de données saisies au clavier.

1.13 Où placer le code source Javascript ?

Le code JavaScript est inséré directement dans le document HTML, autannt de fois que nécessaire ou comme événement. ou mis dans un fichier externe et appelé dans le code HTML. Il y a trois trois façons d'insérer le code JavaScript .

Les programmes JavaScript sont écrits avec le codage Unicode ce qui permet d'utiliser n'importe quel caractère Unicode dans les chaînes et les commentaires.

Unicode est adopté comme standard par la plupart des systèmes d'exploitation modernes pour la représentation de texte.

Unicode comprend l'intégralité des caractères de toutes les langues du monde.

La variante Unicode UTF-8 s'est imposée comme le standard pour le codage de documents HTML. Dès 2016, plus de 80 % des sites Web les plus visités au monde utilisaient UTF-8 pour leurs documents HTML.

Rappelons que l'Unicode[7] est un système de codage universel de caractères, il regroupe plus de 143 859 caractères de différent alphabets (version 2020), des symboles mathématiques, chinois, etc. Ce codage est sur deux octets ce qui autorise le codage de $2^{16} = 65536$ caractères.

1) Grâce à la balise **<script>** de HTML à insérer dans la balise **<head>**, dans la balise **<body>** Cette balise signale au navigateur qu'il s'agit d'un script JavaScript à interpréter. Une page HTML peut contenir plusieurs balises **<script>** mais elles ne doivent pas être imbriquées. La sytaxe est comme, il suit,

```
<!DOCTYPE html>
<html lang="fr">
<head>
<script type="text/javascript">
. . . code en javascript . . .
</script>
<script type="text/javascript" src="chemin/fichier.js"></script>
</head>
<body>
. . . code HTML. . .
<script type="text/javascript">
. . . code en javascript . . .
</script>
. . . code HTML. . .
<script type="text/javascript">
. . . code en javascript . . .
</script>
```

7. https ://unicode-table.com/fr/

```
. . . code HTML. . .
</body>
</html>
```

Ou

```
<script language="javascript">
. . . code de javascript . . .
</script>
```

Pour les anciens navigateurs qui n'interprètent pas JavaScript on met le code en commentaire :

```
<script type="text/javascript">
<!--
. . . code de javascript . . .
-->
</script>
```

Rappelons que <!-- ... --> est pour mettre des commentaires en HTML.

Lorsque le navigateur repère les balises ¡script¿, au lieu d'afficher leur contenu, il lance son moteur JavaScript pour interpréter les instructions du code.

Le code JavaScript n'est jamais montré à l'utilisateur. Mais peut être facilement consulté en accédant au code source de la page dans le navigateur.

type de ¡script¿ peut être omit, par défaut c'est JavaScript .

2) Grâce aux événements. Un événement est réaction à une action de l'utilisateur. Par exemple un clique souris est une action. Un exemple de réaction à un clique sur un lien ouvrira la page web indiquée par ce lien.

```
<balise onEvent ="code JavaScript à insérer">...</balise>
```

Comme exemple d'événement onEvent, on cite onClik (clique souris), onMouseover (survole de souris). Lorsque l'action onEvent est faite par l'utilisateur, le code JavaScript correspondant est déclenché. Nous allons voir les événements plus tard.

3) Insérer du code JavaScript à partir d'un fichier externe (avec l'extension .js) :

```
<script type="text/javascript" src="chemin/fichier.js"></script>
```

Le fichier contenant le code JavaScript , peut être sur une machine locale ou à distance, hébergé dans un site web. Un fichier externe ne doit pas contenir la balise **<script>** maid directement le code JavaScript . Son avantage est la possibilité de le partager avec plusieurs page web.

1.14 Bases du langage JavaScript

Contrairement aux langages HTML et CSS, JavaScript est sensible à la casse, c'est à dire il y a une différence entre majuscule et minuscule.

Chaque instruction se termine par un point-virgule (comme en langage C, C++, Java etc.).

Pour mettre en commentaire le reste d'une ligne on la précède par double slash // et pour mettre en commentaire plusieurs lignes on les met entre /* et */ (comme en C, C++, Java etc.). Les commentaires ne peuvent être imbriqués.

Une chaîne de caractère est une suite de caractères entre les guillemets anglaise simple ' et ' ou entre les guillemets anglaises double " et ".

En JavaScript , le symbole d'addition + sert pour l'addition de nombres mais aussi pour concaténer deux chaîne de caractères.

1.15 La méthode document.write()

Nous allons expliquer cette nouvelle notation dans le chapitre intitulé programmation orienté objet. Pour le moment write() est une méthode qui agit sur l'objet et document est un objet.

La méthode document.write() permet l'affichage de ses arguments dans le corps du document HTML. Elle prend un ou plusieurs arguments séparés par virgule. Elle accepte les balises HTML. La syntaxe est assez simple soit

```
<script type="text/javascript">
  document.write("texte1", "texte2", ... ");
</script>
```

ou

```
<script type="text/javascript">
  document.write("texte1"+"texte2"+... ");
</script>
```

On peut aussi écrire une variable, soit la variable

```
<script type="text/javascript">
  resultat=123;
  document.write(resultat);
</script>
```

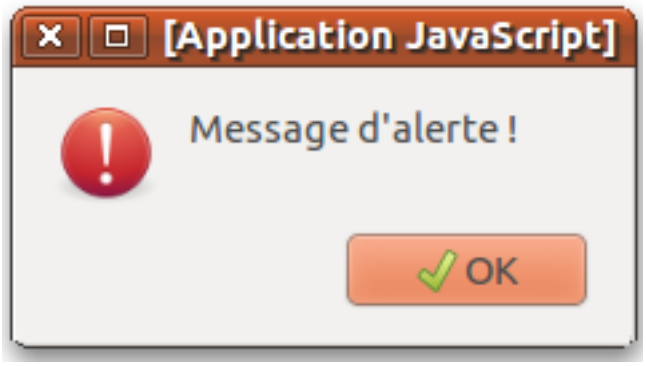


FIGURE 1.2 – Fenêtre alert()

On peut utiliser les balises HTML dans document.write() pour produire la mise en forme de son contenu.

```
<script type="text/javascript">
  document.write("<br><b>Le résultat est : </b>");
</script>
```

La méthode writeln() est proche de write() à ceci près qu'elle ajoute un retour chariot à la fin des caractères affichés par l'instruction. Ce qui n'a aucun effet en HTML. On utilise alors **
** dans document.write() pour retour à la ligne. Mais comme un retour à la ligne n'a aucun effet en HTML, ces deux méthodes se comportent identiquement.

1.16 Les fenêtres de dialogue

1.16.1 La méthode alert()

La méthode alert() de l'objet window permet d'afficher un message d'alert dans une fenêtre voir Figure 1.16.1 qui peut être le résultat d'un traitement ou un simple message à l'utilisateur. La syntaxe est

```
<script type="text/javascript">
  alert("Message d'alerte");
</script>
```

```
<script type="text/javascript">
  alert( "Bienvenue\nJavaScript est facile ! ");
</script>
```

\ permet de créer une nouvelle ligne.

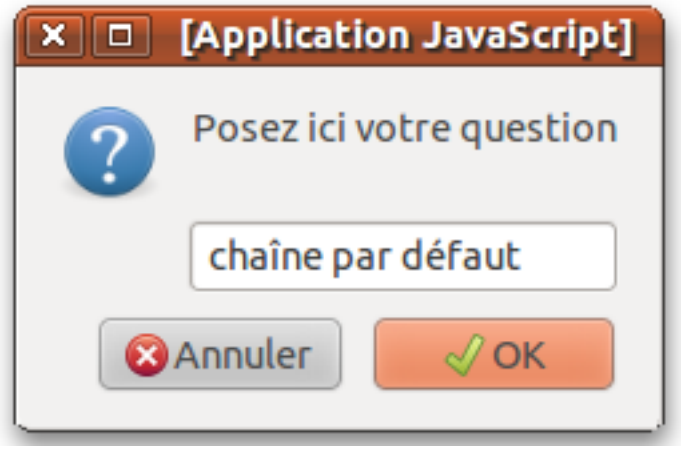


FIGURE 1.3 – Fenêtre prompt()

La méthode alert() permet aussi d'afficher le résultat d'un script.

```
<script type="text/javascript">
  res=123;
  alert( "Le résultat est :\n"+res );
</script>
```

1.16.2 La méthode prompt()

La méthode prompt() permet de récupérer une information provenant de l'utilisateur et lui donner un nom pour l'utiliser comme variable par la suite. Voir Figure 1.16.3. Elle requiert deux arguments : le texte d'invite destiné à l'utilisateur et la chaîne de caractère par défaut dans le champ de saisie. La syntaxe est :

```
<script type="text/javascript">
  var v=prompt(message[,message par defaut optionnel])
</script>
```

Le deuxième argument est optionnel. Exemple :

```
<script type="text/javascript">
  var reponse = prompt('Posez ici votre question','chaîne par défaut');
</script>
```

La valeur récupérée par la méthode prompt() est toujours de type chaîne de caractère, comme le montre le script suivant

```
<script type="text/javascript">
  n=prompt( "saisir un entier ");
  t=typeof(n); // affiche le type
  alert("ce que vous avez saisi est de type : "+t )
</script>
```

Si on désire convertir cette valeur en nombre, on peut utiliser la fonction parseInt() (pour la convertir en un nombre entier) ou parseFloat() (pour la convertir en nombre décimal).

1.16.3 La méthode confirm()

La méthode confirm() prend un seul argument de type chaîne de caractère, et affiche une fenêtre de confirmation avec un message et deux boutons au choix : ”OK” et ”Annuler”. Voir Figure 1.16.3. Cette méthode retourne true si on clique ”OK” et false si on clique ”Annuler”. Syntaxe

```
<script type="text/javascript">
  window.confirm("Message");
</script>
```

confirm() est souvent utilisé dans une structure conditionnelle par exemple :

```
<script type="text/javascript">
  if(confirm("Voulez-vous visiter\n le site de la FSR ?"))
  {document.location.href='http://www.fsr.ac.ma'; }
</script>
```

permet de choisir une option : ”OK” on est amené sur le lien de la FSR et ”Annuler” rester sur la page courante.

1.17 Fonctions de base

parseFloat(chaîne) : permet de transformer une chaîne de caractères en un nombre flottant si possible, sinon renvoie NaN (Not a Number)

Syntaxe : parseFloat(string)

parseFloat(”3.14”);

parseFloat(”AB2”); // retourne NaN

parseInt(string[, base]) : Analyse une chaîne de caractères et retourne un nombre entier dans la base spécifiée. Si aucune base n’est spécifiée, la base décimale est alors considérée par défaut. La base peut être 2 (binaire), 8 (octal), 10 (décimal), 16 (hexadécimal).

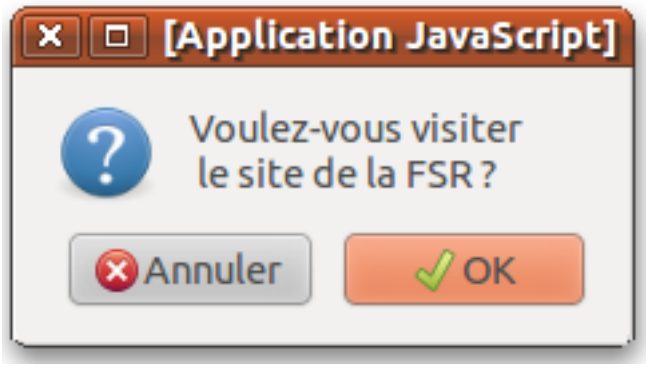


FIGURE 1.4 – Fenêtre confirm()

```
<script type="text/javascript">
  c=parseInt('100',16); document.write(c) //retourne 256
  c=parseInt('A',16); document.write(c) //retourne 10
  c=parseInt('A',8); document.write(c) //retourne NaN
  c=parseInt('16',8); document.write(c) //retourne 14
  c=parseInt(65,2); document.write(c) //retourne NaN
  c=parseInt(1000001,2); document.write(c) //retourne 65
</script>
```

eval : évalue une chaîne de caractère représentant une expression JavaScript, une instruction ou une suite d'instructions JavaScript.

Par exemple : `eval("2 + 3");` retourne 5.

1.18 Exercices

Exercice 7.

Exercice 8.

Exercice 9.

Exercice 10.

Exercice 11.

Exercice 12.

2

Variables et types de données

Dans ce chapitre, nous allons comprendre ce qu'est une variable, comment la déclarer Nous allons voir aussi les types reconnus dans JavaScript . les nombres number, les chaînes de caractères string, et d'autres.

2.1 Variables

2.1.1 Nom de variable

Une variable stocke des données qui peuvent être modifiées au cours de l'exécution d'un programme. Le nom d'une variable ne doit pas commencer par un chiffre et ne doit contenir que des lettres non accentuées, des chiffres, le symbole tiret bas (underscore) ou le dollar symbole $ (mais jamais d'espace, de caractère spécial comme , # etc, de symbole de ponctuation ou autres).

Nous rappelons que JavaScript dépend de la casse. Les deux variable suivantes sont tout à fait différentes `Rayon` et `rayon` .

Les caractères accentués comme à, é, è, ç etc ne peuvent être employés dans les noms de variables mais uniquement dans les chaînes de caractères.

De plus, un nom de variable ne peut être un des mots réservé suivants du langage

JavaScript : abstract, arguments, boolean, break, byte, case catch char class, default do double, else extends, eval, false final finally float, goto, if, implements, import, in, instanceof, int, interface, let, long, native, new, null, package, private, protected, public, return, short, static, super, switch, synchronized, this, throw, throws, transient, true, try, var, void, while, with.

Donner aux variables des noms significatifs. Il vaut beaucoup mieux appeler `rayon` une variable qui indique un rayon, que de l'appeler simplement `r` . Si plusieurs rayons `petitRayon` , `grandRayon` etc `numeroCompte`

2.1.2 Déclaration de variable

La déclaration de variable peut se faire à l'aide de mots clé var et let (avant il se faisait par le mots clé var).

– let : permet de déclarer des variables mutables (qui changent de valeurs). Les anciennes version de JavaScript utilisent var.

– const déclare des constantes (variables immuables).

JS n'impose pas l'initialisation des variables au moment de leur création. JS est non typé. Il n'est pas nécessaire de déclarer le type d'une variable en JS. On peut déclarer une variable par

Nous pouvons attribuer à une variable une nouvelle valeur qui est le résultat d'une expression impliquant sa valeur précédente. Par exemple `x=x+1;` . La variable x prend alors l'ancienne valeur de x augmentée de 1. Peut s'écrire avec le code `x++;` ou `++x;` . Il en est de même pour `x--;` ou `--x;` pour la décrémentation.

```
<script>
  let x=5;
  x++;
  document.write(x)
</script>
```

Le code suivant est correcte `y= x++ - 3` retranche 3 de x et puis incrémente x.

```
<script language="javascript">
  document.write('\u0641'+'\u0627'+'\u0637'+'\u0645'+'\u0629'+'<br>');
  let x=5;
  y=x++ -2;
  document.write(y+'<br>'); //y prend comme valeur 3
  document.write(x) //x prend comme valeur 6
</script>
```

Il vaut mieux éviter cette syntaxe.

`y= ++x - 3` x est d'abord incrémenté puis 3 est retranché.

`y+=3` est équivalent à `y=y+3` .

Il en est de même pour *=, /=, -=

Les opérateurs + et - ont même priorité. Comme il en est pour / et *. JavaScript les calcul de gauche à droite.

'abc"+12 JavaScript convertit 12 en chaîne de caractère et puis concatène pour obtenir abc12

1+2+"abc" donne 3abc

"abc"+1+2 donne abc12

`parseInt("123abc")` retourne 123

`parseInt("abc")` `parseInt("z123abc")` ou retourne NaN pour toute chaine qui ne commence pas par un nombre.

`parseFloat()`

isNaN() vérifie si son argument est un nombre.

`isNaN("abc")` retourne true

`isNaN("123")` retourne false

L'instruction var

Introduite depuis la version 1.0 de JavaScript .

Dans un script, les variables déclarées avec le mot clef var ou non, en dehors de toute fonctions, seront toujours globales. C'est à dire, on peut les exploiter partout dans le document.

Il n'est pas obligatoire de déclarer une variable avec sa valeur initiale. `let x;` . Dans ce cas elle a la valeur undefined. Comme nous pouvons le tester avec le code suivant :

```
<script>
  let x;
document.write(x)
</script>
```

retourne undefined.

Ce code permet de vérifier si une variable à déjà une valeur ou non.

Nous pouvons déclarer une variable et lui affecter une valeur en même temps. L'opérateur d'affectation étant le symbole =.

Si nous réaffectons une nouvelle valeur à une variable, celle-ci garde la dernière valeur et "oublie" définitivement la première. L'espace mémoire occupé est libéré automatiquement.

L'instruction let

Introduite dans ECMAScript 6 de 2015.

let permet de déclarer et éventuellement d'initialiser une variable dont la portée est celle du bloc courant. Un bloc en Javascript est délimité par { }.

Dans une fonction (définit par le mot clé function, voir chapitre sur les fonctions), une variable déclarée par le mot clé var aura une portée limitée à cette seule fonction. On ne pourra donc pas l'exploiter ailleurs dans le document. Par contre, dans une fonction, si la variable est déclarée sans le mot var, alors sa portée sera globale.

Il n'est pas obligatoire de déclarer une variable avec sa valeur initiale. `let x;`

La déclaration sans instruction var est identique à la déclaration avec var.

L'instruction const

Permet de déclarer une constante disponible uniquement en lecture.

Nous ne pouvons déclarer deux constantes avec le même nom dans un même bloc.

let fonctionne avec les portées de bloc et var avec les portées des fonctions :

```
let x ;
```

Le symbole d'affectation est le signe =.

Il est possible de définir plusieurs variables en une seule instruction, en les séparant par virgule :

```
var x=3, y=6 ;
```

La déclaration de variable peut se faire de deux façons :

- Explicitement, par le mot clef let.

Par exemple :

var Numero = 1 ; var nom = ”espace” ;

- Implicitement : sans utiliser var. Par exemple :

Numero = 1 ; Prenom = ”Eve” ;

Utiliser var ou non ?

2.2 Types de données

La tâche d'un ordinateur est le traitement de données. Ces données peuvent être de différents types : chaîne de caractères, nombres décimaux ou entiers etc.

Du moment que les données sont différemment traitées, il est important de comprendre la notion de type. Il n'a aucun sens de multiplier deux chaînes de caractère, cependant ce n'est pas le cas de nombres.

\b	touche de suppression,	\f	formulaire plein
\n	nouvelle ligne,	\r	appui sur la touche ENTREE
\t	tabulation,	\"	guillemets doubles
\'	guillemets simples,	antianti	caractère antislash
\xNN	caractère de code NN dans Latin-1	xNNN	caractère de code NN en Unicode

TABLE 2.1 – Caractères spéciaux en JavaScript

JavaScript est un langage faiblement typé. Contrairement à d'autres langages, notamment C, nous n'avons pas besoin de déclarer le type d'une variable pour l'utiliser. quand nous écrivons `let x=5;` alors x est considéré comme nombre. De même quand nous déclarons `let C='Bonjour';` alors C est considéré comme chaîne de caractère.

Toute valeur en JavaScript, est de l'un des types suivants :

Le type Number

nombres entiers ou décimaux. Par exemple 21 ou 3.14 (le séparateur décimal étant le point). JavaScript ne fait pas la distinction entre entier et décimaux. Il est capable de manipuler les nombres positifs ou négatifs entre -2^{53} et $+2^{53}$

Le type String

ou chaîne de caractère : c'est une suite de caractères à placer toujours entre ' ' ou " ", (sinon cette chaîne de caractère sera considérée comme une variable). Nous ne pouvons inclure " dans " ou "" dans "". Le ' dans une chaîne de caractère doit être précédé d'un \ pour avertir l'interpréteur que ' n'est pas la fermeture. `C='aujourd'hui'` est incorrecte, par contre `C='aujourd'hui'` ou `C="aujourd'hui"` est correcte.

Pour inclure " dans "" il suffit de précéder les "" internes par \.

C="William Shakespeare a dit T̈o be, or not to be, that is the question".

Le tableau 2.2 liste d'autres caractère spéciaux en JavaScript .

Dans le tableau 2.2, `NN`est un nombre hexadécimal qui représente un caractère de l'ensemble Latin-1 appelé aussi ISO/CEI 8859-1 ou simplement ISO 8859-1. Voir l'ensemble complet [1] Par exemple :

`document.write("\x41")` produit A

`document.write("\xE6")` produit æ

`document.write("\xA9")` produit ©

1. https ://fr.wikipedia.org/wiki/ISO/CEI_8859-1

De même nous pouvons écrire des caractères Unicode en utilisant leur code sous la forme `\uNNNN`. L'Unicode[2] code

`document.write('\u0641'+'\u0627'+'\u0637'+'\u0645'+'\u0629')` produit فاطمة

Le type **Boolean**

type dont les valeurs possibles sont true et false. Un traitement se fait souvent en fonction d'une condition, si celle ci est vraie voilà ce qu'il faut faire, sinon faire autre chose.

Le type **Null**

indique l'absence de valeur : aucune valeur pour l'objet n'est présente

Le type **Undefined**

l'unique valeur possible est undefined. C'est le type d'une variable déclarée avant qu'une valeur ne lui soit affectée.

Object :

Symbol :

L'opérateur typeof permet de déterminer le type d'une variable ou d'une expression.

Syntaxe : typeof(x) ou typeof x

```
let i = 1;
typeof i; //retourne number
let titre="JavaScript est génial";
typeof titre; //retourne string
var choix = true;
typeof choix; //retourne boole
var V;
alert(" type de V : "+(typeof V)); // retourne undefined
```

2.2.1 Conversion de types

.

2. https ://unicode-table.com/fr/

Les conversions possibles sont de types primitifs en chaînes de caractères et de chaînes de caractères en nombres entiers ou réels. Il est aussi possible de conversion entre différentes bases de système de numération.

Les types primitifs correspondent à des pseudo- objets. Ils peuvent posséder des méthodes,

```
var b = true;
alert(" type de b : "+(typeof b));
var v = b.toString();
alert(" type de v : "+(typeof v));
```

Pour les nombres, la méthode toString peut être utilisée avec un paramètre pour spécifier la base souhaité. Par défaut, la base décimale est utilisée. Il est possible de spécifier, la base binaire par 2, la base octale par 8, la base hexadécimale par 16 ou n'importe qu'elle autre base.

```
var N = 5;
var v1 = N.toString();
var v2 = N.toString(2); // 101
```

Les fonctions parseInt et parseFloat convertissent les chaînes de caractères représentant des nombres en nombres entiers ou décimaux. ParseInt peux prendre 2 arguments : la chaine à convertir et sa base.

```
var v = parseInt("11001"); //retourne 11001
var v = parseInt("11001",2); //retourne 25
```

Les opérateurs + et - permettent de convertir une chaîne de caractères en nombre de la même façon que la méthode parseInt . Le - change en plus le signe du nombre.

2.3 Constantes

Une constante est une variable qui ne change pas de valeur. Pour définir une constante, il suffit d'utiliser un nom de constante et de lui affecter une valeur au moyen de l'opérateur d'affectation =. Par exemple `pi=3.14 ;` ou `const pi=3.14;` .

Il existe des constantes prédéfinies telles que dans la Table 2.2 :

Toute valeur différente de null, NaN, undefined, 0 et la chaîne de caractères vide est évaluée par défaut par JavaScript comme true. Var x=3 ; if(x) ?

Constante	Explication
undefined	la variable qui prend cette valeur a été déclarée mais n'a pas été initialisée.
null	la variable n'existe pas.
Infinity	représente l'infini positif. Permet notamment de vérifier si il y a une division par zéro.
NaN	équivaut à la définition de l'IEEE de Not a Number .

TABLE 2.2 – Constantes prédéfinies de JS

Signe	Nom	Signification	Exemple	Résultat
+	plus	addition	x + 3	14
-	moins	soustraction	x - 3	8
*	multiplié par	multiplication	x*2	22
/	divisé	par division	x /2	5.5
%	modulo	reste de la division par	x%7	4
=	a la valeur	affectation	x=5	5

TABLE 2.3 – Symboles arithmétique en JavaScript

2.4 Opérateurs arithmétiques, booléens et de comparaison

Pour développer des programmes, en plus des variables à manipuler, nous disposons de nombreux opérateurs pour les traiter.

2.4.1 Les opérateurs arithmétiques

JavaScript et comme tout langage de programmation permet de faire tous les calculs arithmétiques. JavaScript utilise les symboles arithmétiques classiques. + pour l'addition, * pour la multiplication, / pour la division, ** pour la puissance, % pour le modulo,

Voir Tableau 2.4.1

Dans les exemples, la valeur initiale de x sera toujours égale à 11.

2.4.2 Les opérateurs de comparaison

JavaScript utilise les opérateurs suivants pour la comparaison de valeurs. Ces opérateurs sont utilisés dans les structures conditionnelles que nous allons voir plus tard.

Pour les exemple, on suppose x=11.

Signe	Nom	Exemple	Résultat
==	comparaison de valeurs	`x==11`	true
===	comparaison de valeur et type	`x=='11'`	false
<	inférieur	`x<11`	false
<=	inférieur ou égal	`x<=11`	true
>	supérieur	`x>11`	false
=<	supérieur ou égal	`x>=11`	true
!=	différent	`x!=11`	false
!==	différence de valeur et type	`x!=='11'`	true

TABLE 2.4 – Symbole de comparaaison en JavaScript

Signe	Description	Exemple	Signification	Résultat
+=	plus égal	x += y	x = x + y	16
-=	moins égal	x -= y	x = x - y	6
*=	multiplié égal	x *= y	x = x * y	55
/=	divisé égal	x /= y	x = x / y	2.2

TABLE 2.5 – Symboles d'incrémentation en JavaScript

```
lry b=2>5;
alert(" type de b : "+(typeof b));  \\ retourne boolean
```

Important. Il ne faut pas confondre le = et le == (deux signes =). Le = est un opérateur d'affectation de valeur tandis que le == est un opérateur de comparaison. Cette confusion est une source classique d'erreur de programmation.

2.4.3 Les opérateurs associatifs

On appelle ainsi les opérateurs qui réalisent un calcul dans lequel une variable intervient des deux côtés du signe = (ce sont donc en quelque sorte également des opérateurs d'attribution). Dans les exemples suivants x vaut toujours 11 et y aura comme valeur 5.

2.4.4 Les opérateurs d'incrémentation

Ces opérateurs vont augmenter ou diminuer la valeur de la variable d'une unité. Ce qui sera fort utile, par exemple, pour mettre en place des boucles. Dans les exemples x vaut 3.

Signe	Description	Exemple	Signification	Résultat
x++	incrémentation (x++ est le même que x=x+1)	y = x++	euis plus 1	4
x- -	décrémentation (x- - est le même que x=x-1)	y= x- -	3 puis moins 1	2

TABLE 2.6 – Symboles d'incrémentation en JavaScript

Signe	Nom	Exemple	Signification
&&	et	(condition1) && (condition2)	est vrai si les 2 conditions sont vraies, sinon fausse
——	ou	(condition1) —— (condition2)	est vrai si au moins une des 2 conditions est vraie, sinon fausse
!()	non logique	!(condition)	donne vrai si 'condition' est fausse, sinon fausse

TABLE 2.7 – Symboles logiques en JavaScript

En incrémentant ou décrémentant avec le double signe (++ ou - -) avant le nom de la variable, l'opération d'incrémentation ou de décrémentation sera prioritaire sur l'assignation. Par contre, si le double signe est après le nom de la variable, l'assignation sera prioritaire sur l'incrémentation ou la décrémentation.

2.4.5 Les opérateurs logiques

Aussi appelés opérateurs booléens, ses opérateurs servent à composer deux ou plusieurs conditions.

2.4.6 La priorité des opérateurs Javascript

Les opérateurs s'effectuent dans l'ordre suivant de priorité (du degré de priorité le plus faible ou degré de priorité le plus élevé). Dans le cas d'opérateurs de priorité égale, de gauche à droite.

Méthode	Description
isArray	Détermine si le paramètre est un tableau.
isBoolean	Détermine si le paramètre est un booléen.
isEmpty	Détermine si un tableau est vide.
isFinite	Détermine si le paramètre correspond à un nombre fini.
isFunction	Détermine si le paramètre est une fonction.
isNaN	Détermine si la valeur du paramètre correspond à NaN (Not a Number).
isNull	Détermine si le paramètre est null.
isNumber :	Détermine si le paramètre est un nombre.
isObject	Détermine si le paramètre est un objet.
isString	Détermine si le paramètre est une chaîne de caractères.
isUndefined	Détermine si le paramètre est indéfini, c'est-à-dire une référence non initialisée.

TABLE 2.8 – Constantes prédéfinies

Opération	Opérateur
,	virgule ou séparateur de liste
= += -= *= /= %=	affectation
? :	opérateur conditionnel
——	ou logique
&&	et logique
==!=	égalité
¡ ¡= ¿= ¿	relationnel
+ -	addition soustraction
* /	multiplier diviser
! - ++ –	unaire
( )	parenthèses

2.4.7 Opérateur sur les chaînes de caractères

L'opérateur + permet aussi de concaténer deux chaînes de caractères. Par exemple ″bon″+″jour″ donne ″bonjour″.

JavaScript offre un ensemble de méthodes permettant de détecter le type et la validité de la variable passée en paramètre. Ces méthodes commence par is.

Les méthodes escape et unescape offrent respectivement la possibilité d'encoder et de décoder des chaînes afin qu'elles puissent être utilisées dans des pages HTML.

2.5 Structures conditionnelles

La notion de condition existe dans tous les langages de programmation.
Seul le premier if et le bloc qui le qui sont nécessaires.

2.5.1 La structure if

```
if (condition1)
  { code qui sera interpété si la condition1 est vraie }
else if (condition2)  //optionnel
    { code qui sera interpété si la condition2 est vraie }
    // autant de else if que nécéssaire
else      //optionnel
   { code qui sera interpété si toutes
    les conditions ci-dessus sont fausses }
```

La condition doit être toujours entourée de parenthèses ( ). Le code qui la suit entre accolades est exécute si la condition est vraie

Les accolades { } ne sont obligatoires qu'en cas d'instructions multiples.

Il est possible d'utiliser autant de blocs `else if` que nécessaire.

La séquence else (optionnelle) est interprétée si toutes les 'condition' sont fausses.

Exemple.

```
<script type="text/JavaScript">
  var n=prompt('Saisir votre âge');
  if (n>=0 && n<=14){alert('Vous êtes un enfant')}
  else if  (n>=15 && n<=24){alert('Vous êtes un adolescent')}
  else if  (n>=25 && n<=64){alert('Vous êtes un adulte')}
  else if  (n>=65 && n<=120){alert('Vous êtes un aînés')}
  else {alert('âge érroné')}
</script>
```

2.5.2 Une autre structure conditionnelle

```
var variable=(condition) ? Valeur1 : Valeur2 ;
```

Si 'condition' est vraie, alors valeur1 est affectée à 'variable', sinon c'est valeur2 qui lui est affectée.

Par exemple :

```
<script type="text/JavaScript">
  x=(x>=0)?x:(-x) ;
  tranche=(age>18) ? 'majeur' :'mineur' ;
  salut=(heure<=18)?"Bonjour":"Bonsoir";
</script>
```

2.5.3 switch

La syntaxe est

```
switch(variable)
{
case valeur1 : instructions à faire si variable=valeur1 ; break ;
case valeur2 : instructions à faire si variable=valeur2  ; break ;
...
default : instructions à faire,  par défaut, si
     variable est différente de valeur1, valeur2 etc ;
}
```

switch ne peut tester que l'égalité de valeurs et non pas l'inégalité. Variable doit être une valeur et non une expression à évaluer.

Le contenu des case (valeur1, etc) doit être un élément déjà évalué. On ne peut y insérer des expressions. Pour éviter de tester tous les cas, l'utilisation de break permet de sortir du switch dès l'action terminée. Le default permet d'effectuer une action si la variable testée ne coïncide avec aucune des valeurs proposées dans les case précédent.

Il faut préciser un break pour chaque case afin de sortir du switch, sinon le switch va lire et exécuter le code contenu dans chaque case à la suite.

Exemple :

```
<script type="text/JavaScript">
  var n=parseFloat(prompt('Saisir un nombre'));
  switch (n){
    case 1: alert('Janvier'); break;
    case 2: alert('Février'); break;
    case 3: alert('Mars'); break;
    ...
    default :alert('Saisie érronée')
  }
</script>
```

2.6 Structures itératives

Cette notion appelée aussi boucles existe dans tous les langages de programmation. La résolution de plusieurs problèmes se ramène à un même traitement sur différentes valeurs.

Les structures itératives ou boucles permettent d'exécuter un bloc de code un certain nombre de fois, tant qu'une certaine condition est vraie. La condition donnée doit être fausse à un moment donné, sinon la boucle devient infinie ! JavaScript définit quatre types de boucles, for , for...in , while et do...while .

2.6.1 La boucle for

est utilisé quand nous connaissons à l'avance le nombre d'itérations

```
for (départ_compteur; condition_continuation; incrémentation)
{
code à  interpréter  ;
}
```

Exemple :

```
<script type="text/javascript">
document.write("voici les multiples de :")
for (var k=1; k<=10; k++)
  {
  document.write(7*k+"<br>");
  };
</script>
```

2.6.2 La boucle for in

est utilisé quand on parcourt un ensemble d'objet

```
for (variable in object)
  { code à  interpréter  ; }
```

```
<script type="text/javascript">
var personne={Nom:"Ali", Prénom:"Mohammed", age:25};
for (x in personne){
document.write(personne[x] + " ");
}
</script>
```

2.6.3 while

est utilisé quand

```
while (condition)
{
code à exécuter à chaque passage ;
}
```

```
<script type="text/javascript">
k=1;
while (k<=10)
  {document.write(7*k+' ');
     k++;
  };
</script>
```

2.6.4 do while

est utilisé quand

La boucle do – while est intéressante si besoin pour une raison ou pour une autre d'effectuer au moins un passage dans une boucle pour faire fonctionner un script. L'initialisation est exécutée avant tout passage dans la boucle for, tandis que l'incrémentation est exécutée à la fin de chaque passage dans la boucle for.

```
do
{
   code qui sera interprété à chaque itération
}
while (condition);
```

JavaScript définit les mots-clé break et continue afin de modifier l'exécution des boucles. Le premier offre la possibilité d'arrêter l'itération d'une boucle et de sortir de son bloc d'exécution, et le second de forcer le passage à l'itération suivante. Les traitements suivants de l'itération courante ne sont alors pas effectués.

break : permet d'interrompre prématurément une boucle for ou while .

Continue : permet de sauter une instruction dans une boucle for ou while et de continuer ensuite le bouclage (sans sortir de celui-ci comme le fait break).

Exemple

```
<script type="text/JavaScript">
var res = '';
```

```
for (i = 0; i < 10; i++) {
  if (i >= 3 && i<=5) { continue; }
  text = res + i+' ';}
document.write(res)
</script>
```

```
<script type="text/JavaScript">
var i = 0;
while (i < 10) {  if (i == 5) {break;}
  i = i + 1;
  document.write(i)}
</script>
```

2.7 Exercices

Exercice 13.

Exercice 14.

Exercice 15.

3

Les fonctions

La notion de fonction existe dans tous les langages de programmation. Une fonction est un sous-programme qui porte un nom et qui peut être utilisé plusieurs fois, par simple appel de son nom. Le nom de fonction suit les mêmes règles que celles de variables.

Avant de l'appeler, une fonction doit être définie. Une fonction peut être placée n'importe où dans un document HTML ; mais il est conseillé de la placer entre ¡head¿ ... ¡/head¿. Les arguments de fonctions doivent être indépendants du contexte d'un document HTML. Pour rappel, Javascript est sensible à la casse. Ainsi, fonct() et Fonc() sont différentes. En outre, les noms de fonctions dans un même script doivent être différents par exemple.

Les instructions composant la définition d'une fonction ne sont interprétées que jusqu'à l'appel de la fonction.

En Javascript, il existe deux types de fonctions :

- les fonctions prédéfinies dans le langage Javascript, comme parseInt(), eval() etc. Les fonctions membres appelées aussi "méthodes". Chacune d'entre elles est associées à un objet de JavaScript . Par exemple la méthode alert() de l'objet window, document.write de document etc.

- les fonctions écrites par le programmeur.

3.1 Programmation de fonction

Pour définir une fonction, on utilise le mot réservé function. La syntaxe de définition d'une fonction est la suivante.

```
function maFonction(argument1, argument2, ...) {
  ... instructions ... ;
  return resultat
}
```

La mention d'arguments est facultative dans la définition d'une fonction. En cas d'utilisation d'arguments, ils sont séparés par la virgule. Les parenthèses qui suivent le nom de fonction sont obligatoires, même en cas d'absence d'arguments. Ces parenthèses permettent à l'interpréteur JavaScript de distinguer les fonctions des variables. La définition d'une fonction est groupée dans un bloc entre accolades { }.

Pour renvoyer un résultat, il suffit d'écrire le mot clé return suivi de l'expression à renvoyer, sans parenthèses. L'instruction return est facultative et on peut utiliser plusieurs return dans une même fonction.

Une fois une fonction définie, on pourra l'appeler partout dans la suite par son nom. Une fonction peut appeler une autre fonction déjà définie.

Rappelons que le code JavaScript peut être placé comme valeur d'attribut d'événement onClick, onMouseOver, onFocus etc.

¡balise onEvent=" code JavaScript "¿ ... ¡/balise¿

Quand ce code JavaScript est long, on l'écrit sous forme d'une fonction nommée maFonction par exemple et on l'appelle comme il suit :

¡balise onEvent="maFonction()"¿ ... ¡/balise¿

```
<script type="text/javascript">
function fac(n){
  if (n < 2) {
    return 1;
  } else {
    return n * fac(n-1);
  }
}
</script>
```

On pourra par la suite appeler cette fonction

```
<script type="text/javascript">
var N= fac(7);
document.write(N);
```

```
</script>
```

Les arguments passés à une fonction constituent automatiquement un tableau nommé arguments.

Par exemple : une fonction qui fait la somme d'un nombre indéterminé de nombres.

```
<script>
function somme() {
    var s = 0;
    for (i = 0; i < arguments.length; i++) {
        s += arguments[i];
    }
    return s;
}
x = somme(2,4,6,8,10);
y = somme(1,3,5);
document.write(x);
document.write(y);
</script>
```

Écrire Une fonction qui affiche ses arguments.

```
<script type="text/javascript">
function F()
{for(i=0;i<arguments.length;i++)
  document.write(arguments[i]);
}
F("a","b", 123); // retourne ab123
</script>
```

Variable Locale. Une variable déclarée dans une fonction en utilisant var, n'est reconnue que à l'intérieur cette fonction. Cette variable n'est pas reconnue en dehors de sa fonction. On dit que cette variable est locale. Le navigateur efface de la mémoire toute variable locale à une fonction dès que cette fonction est utilisée.

Deux fonctions différentes peuvent utiliser un même nom de variable.

Variable Globale. Une variable globale peut être déclarée n'importe où dans un script en-dehors de toute fonction. On dit qu'elle a une portée globale. Une variable globale est stockée en mémoire, ce qui veut dire qu'elle prend plus de mémoire qu'une variable locale. C'est pourquoi, il faut privilégier l'utilisation des variables locales autant que possible.

3.2 Fonctions de haut niveau prédéfinies

Les fonctions de haut niveau ne sont associées à aucun objet particulier.

Fonction	Signification
encodeURI(string)	Encode l'URI de telle façon que tous les caractères spéciaux sont transformés en séquences de signes ASCII.
decodeURI(string)	Décode une URI qui a été encodé avec encodeURI.
eval()	Evalue et exécute le JS contenu dans une chaine de caractère. Var x=7 ; chaine="x= x+ 3" ; eval(chaine) ;
IsFinite()	Retourne true si la valeur passée en paramètre fait partie de la plage de nombre que peut traiter JavaScript, sinon retourne false.
isNaN()	Retourne true si la valeur passés en paramètre est un nombre, sinon retourne false.
Number()	convertit un objet en un nombre.
ParseFloat()	convertit une chaîne de caractères en valeur réelle flottante
parseInt()	convertit une chaîne de caractères en valeur entière dans la base spécifiée
String()	convertit un objet en une chaîne.

TABLE 3.1 – Fonctions de haut niveau prédéfinies en JavaScript

La fonction eval() permet d'évaluer une chaîne de caractère sous forme de valeur numérique. On peut stocker dans une chaîne des opérations numériques, des opérations de comparaison, des instructions et même des fonctions.

Par exemple C='3 + 7' ; eval(C) donne comme résultat 10.

Voir les autres fonctions dans le Tableau 3.1. Certaines fonctions comme escape(), unescape() sont devenues obsolètes.

3.3 Exercices

4
Objets et tableaux

Dans la Figure 7.3 du chapitre précédant, nous avons vu tous les objets prédéfinis de JavaScript . Dans ce chapitre, nous allons voir les propriétés et les méthodes importantes de chacun de ces objets.

4.1 L'objet Array

4.1.1 Tableaux

Array veut dire tableau en français. Un tableau est une suite finie de variables auxquels on peut accéder par indice (au lieu de leur accéder par leur nom). L'indexation des tableaux débute à 0. Cette notion de tableau existe dans tous les langages de programmation.

En JavaScript l'objet Array est utilisé pour créer des tableaux. La syntaxe est

1) Si on connaît le contenu du tableau

```
var monTableau = new Array(E0,  E1, ...,  En);
```

de façon équivalente, on peut aussi déclarer un tableau avec la syntaxe suivante :

```
var monTableau=[E0,  E1, ...,  En];
```

on a monTableau[0]=E0, monTableau[1]=E1, etc

```
<script type="text/javascript">
T=new Array('a','b',3,5);
document.write(T[0],T[2]) //retourne a 3
T[0]='A'; // écrase l'ancienne valeur et la remplace par A
document.write(T[0]) //retourne A
</script>
```

2) on connaît le nombre des éléments du tableau.

```
  Tableau  = new Array(nombre);// ou  Tableau=[nombre]
```

pour compléter ce tableau on fait des affectations Tableau[0]=V0, Tableau[1]=V1, ...

```
<script type="text/javascript">
T=new Array(3);
T[0]='pomme'; T[1]='banane';T[2]='poire';
U=[3];
U[0]='fraise'; U[1]='orange'; U[2]='mangue';
document.write(T[0],T[2]) //retourne  pomme poire
document.write(U[0],U[2]) //retourne  fraise mangue
</script>
```

3) Si on ne connaît pas le nombre des éléments du tableau à l'avance.

```
  Tableau = new Array(); ou Tableau=[]
```

pour compléter ce tableau on fait des affectations : Tableau[0]=E0, Tableau[1]=E1, ...

```
<script type="text/javascript">
T=new Array();
T[0]='pomme'; T[2]='poire';
U=[];
U[0]='fraise'; U[1]='orange'; U[2]='mangue';
document.write(T[0],T[1]) //retourne  pomme undefined
document.write(U[0],U[2]) //retourne  fraise mangue
</script>
```

On peut parcourir les tableaux en utilisant une boucle. La propriété length retourne la longueur de tableau.

```
  <script language="javascript">
  var T = new Array();
  T[0] = "orange";
  T[1] = "pomme";
  T[2] = "kiwi";
```

```
for (i=0;i<T.length;i++){document.write(T[i] + "<br />");}
</script>
```

4.1.2 Tableaux associatifs

Les tableaux associatifs sont des tableaux indexés par des chaînes de caractères et non pas par des indices numériques.

```
var T = new Array();    T['Clé1'] = 'Val1';    T['Clé2'] = 'Val2 '; ...
```

ou en utilisant les parenthèses

```
var T = {"clé1" :"val1" , "clé2" :"val2" ,  ... };
```

```
<script type="text/javascript">
var T = new Array();
T['Clé1'] = 'Val1';
T['Clé2'] = 'Val2 ';
for( x in T){document.write(x + ' : ' + T[x] + '  '+'<br/>')};

var Notes = {'Ali':[12,13,10,16,13], 'Med':[15,10,8,12,14],
  'Saloua':[10,12,15,4,18]};
document.write(Notes['Ali'],'<br>');  // retourne 12,13,10,16,13
document.write(Notes['Ali'][3]); // retourne 16
</script>
```

4.2 Tableaux prédéfinis de JavaScript

4.2.1 Objet prédéfini images[]

images : est un tableau contenant toutes les images de document HTML. Il a comme propriétés src, width, height, id, ...

document.images[i] indique la (i+1)eme image du document en cours.

document.images.length : retourne le nombre d'image dans le document HTML en cours.

document.images[i].src : contient le le chemin et nom de fichier de la (i+1)eme image dans le document HTML en cours.

Au lieu d'utiliser les indices pour le tableau images on peut utiliser l'attribut name de la balise ¡img¿ et utiliser document.images['nomImage'].

On peut aussi utiliser l'attribut id pour la balise ¡img¿ et la méthode getElementById() de document : document.getElementById('nomId') pour indiquer l'image ayant l'Id='nomId'.

```
<img src='img2.jpeg' id="abc">
<script type="text/javascript">
p=document.getElementById('abc').src
document.write(p); //retourne img2.jpeg
</script>
```

Propriété de l'objet images[]

alt contient le texte alternatif de l'image.

border Contient la valeur de border.

complete Contient un indicateur de fin de chargement de l'image. Vaut true si l'image est complètement chargée et false sinon.

fileSize Contient la taille en octets de l'image (ne fonctionne pas avec tous les navigateurs).

height contient la valeur définie par le paramètre height de la balise ¡img¿ en son absence indique la hauteur réelle de l'image.

id contient la valeur définie par le paramètre id de la balise ¡img¿. Cet identifiant s'utilise avec la méthode getElementById() de document. Attention de ne pas confondre id avec name.

name contient la valeur définie par le paramètre name de la balise ¡img¿.

src La propriété src contient le chemin et le nom de fichier de l'image.

width contient la valeur définie par le paramètre width de la balise ¡img¿ en son absence indique la largeur réelle de l'image.

4.2.2 L'objet prédéfini links[]

links : est un tableau contenant tous les liens du document HTML en cours. Il a comme propriétés href, target, id, ...

document.links[i] indique le (i+1)eme lien du document en cours.

document.links.length : retourne le nombre de liens dans le document HTML en cours.

document.links[i].href : contient l'URL du (i+1)eme lien dans le document HTML en cours.

Au lieu d'utiliser les indices pour le tableau links on peut utiliser l'attribut name de la balise ¡a¿ ...¡/a¿ et utiliser document.links['nomLien']

On peut aussi utiliser l'attribut id pour la balise ¡a¿ ...¡/a¿ et la méthode getElementById() de document : document.getElementById('nomId') pour indiquer le lien ayant l'Id='nomId'.

```
<a href="http://www.google.com" id='abc'> Google </a> <br>
<a href="http://www.yahoo.com" target="blank"> Yahoo </a> <br>
<script type="text/javascript">
document.write('URL du 1er lien est : ',document.links[0].href,'<br>');
document.write('La cible du 2e lien : ',document.links[1].target,'<br>');
document.links[0].href='http://www.youtube.com'; // modifie URL
document.getElementById('abc').textContent='Youtube';
</script>
```

4.2.3 L'objet prédéfinis forms[]

document.forms[i] indique le (i+1)eme formulaire du document en cours.

Au lieu d'utiliser les indices pour le tableau forms on peut utiliser l'attribut name de ¡form¿ ...¡/form¿ et utiliser document.forms['nomFormulaire'] pour indiquer le formulaire nommé 'nomFormulaire'. On peut aussi utiliser l'attribut id="monId" dans la balise ¡form¿ et document.getElementById('monId') pour repérer le formulaire dont l'Id="monId".

document.forms.length : retourne le nombre de formulaire dans le document HTML.

voir exemple ci-dessous.

Propriétés de l'objet forms[]

action Contient l'action définie pour un formulaire document.forms[X].action ; (action utile pour PHP).

elements Tableau de tous les élément d'un formulaire.

encoding Contient le type de " ENCTYPE" des données du formulaire. document.forms[X].encoding.

length Nombre de formulaires que contient le document. document.forms.length

method Contient la "méthode de transmission des données" (get/post) du formulaire

document.forms[X].method ; X est un indice ou nom de formulaire donné par l'attribut name de la balise ¡form¿.

name Contient le nom du formulaire document.forms[X].name

target Fenêtre cible du formulaire. document.forms[X].target

Méthodes de l'objet forms[]

reset() Réinitialise un formulaire : document.forms[X].reset()
submit() Soumet un formulaire : document.forms[X].submit()

4.2.4 L'objet prédéfinis elements[]

elements : est un tableau contenant tous les éléments d'un formulaire. Il s'agit de tous les types dans ¡input¿, ¡select¿, ¡textarea¿, etc

Au lieu d'utiliser les indices pour le tableau elements on peut utiliser l'attribut name de ¡input¿ et utiliser document.forms[i].elements['nomElement']. On peut aussi utiliser l'attribut id avec document.getElementById() pour repérer l'élément dont l'id indiqué.

document.forms["nomForm"].elements['nomElement'].value ; valeur de l'élément nommé 'nomElement' du formulaire nommé "nomForm".

On peu mixer la désignation par nom et par indice.

```
<form name='ff'>
<input type='text' name="cc">+
<input type='text' id="abc">=
<input type='text' onClick="
a=parseFloat(document.forms['ff'].elements[0].value);
b=parseFloat(document.getElementById('abc').value);
document.forms[0].elements[2].value=a+b;">
</form>
```

4.2.5 Localisation de balise

En plus des attributs name et id, JavaScript permet de localiser des balises d'après leurs noms ou la valeur de leur attribut class. Nous allons voir :

getElementsByTagName

getElementsByTagName(nomBalise) : renvoie tous les éléments du document dont le nom de balise est nomBalise. (remarquer le s dans getElementsByTagName).

```
<p> Paragraphe Paragraphe Paragraphe </p>
<p> texte  texte texte </p>
<script type="text/javascript">
var c=document.getElementsByTagName("p");
document.write(c[1].innerHTML); // retourne : texte  texte texte
</script>
```

getElementsByClassName

getElementsByClassName(maClasse) renvoie tous les éléments du document ayant un attribut HTML class dont la valeur est maClasse.

```
<div class='abc'> Paragraphe Paragraphe </div>
<div class='xyz'> texte  texte texte </div>
<div class='abc'> phrase phrase phrase </div>
<script type="text/javascript">
var c=document.getElementsByClassName("abc");
document.write(c[1].innerHTML); // retourne : phrase phrase phrase
</script>
```

4.3 Exercices

5
Manipulation du DOM

5.1 Qu'est-ce que le DOM ?

DOM (Document Object Model ou modèle objet de document en français) est standard développé par W3C.

Le modèle DOM consiste à représenter, de manière orienté objet, le contenu d'un document HTML comme structure arborescente composée de nœuds. Est considéré comme nœud dans ce modèle, le document lui même, les éléments, le contenu textuel d'un élément, et les commentaires. Chaque nœud est un objet pour lequel il existe des propriétés et des méthodes permettant de lire ou modifier dynamiquement la structure, le contenu et le style de ce document, via un langage de programmation comme JavaScript, PhP 5, Java, Python, Perl, ou autres.

Ces langages permettent la manipulation de document HTML à travers leur représentation DOM. Les noms des interfaces, classes, méthodes et propriétés du DOM sont indépendantes des langages.

`<h1>DOM est génial ! </h1>` cet élément créé deux nœuds : le nœud élément h1 et le nœud texte.

DOM est un standard qui définit :
– Les éléments HTML en tant qu'objets ;

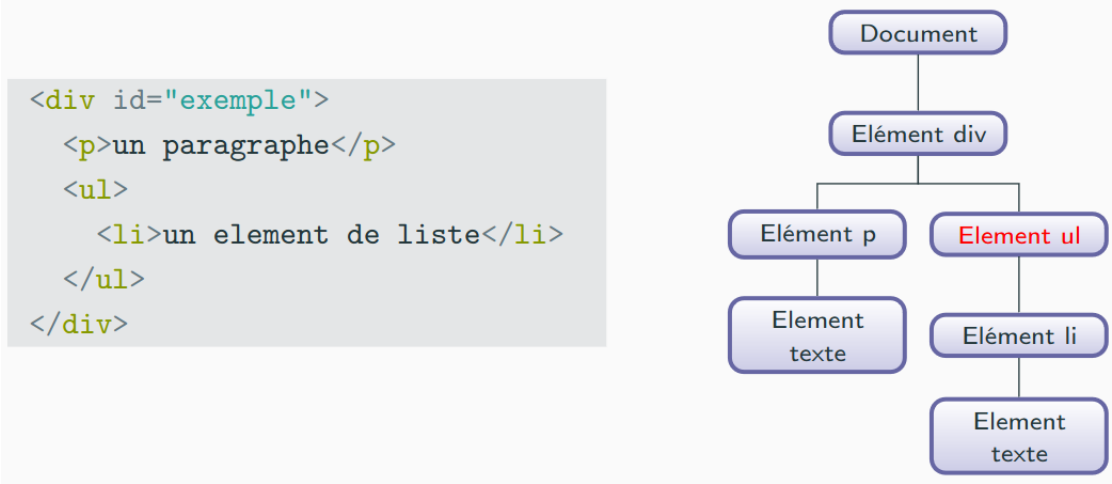


FIGURE 5.1 – Hiérarchie DOM

– Les propriétés de tous les éléments HTML ;
– Les méthodes pour accéder à tous les éléments HTML ;
– Les événements pour tous les éléments HTML ;

Avec DOM, JavaScript peut :
– accéder et modifier tous les éléments, attributs et styles CSS d'un document HTML.
– supprimer les éléments, et les attributs existant d'un document HTML ou en ajouter de nouveau.
– réagir à tous les événements d'un document HTML ou en créer de nouveaux.
C'est ce que nous allons apprendre à faire dans ce chapitre.

Pour rappel : Une propriété est une valeur qu'on peut obtenir ou définir. Alors qu'ne méthode est une action qu'on peut effectuer (comme ajouter ou supprimer un élément HTML).

5.2 Sélection des nœuds du DOM

Une fois qu'un nœud est sélectionné, par les méthodes vues dans la section précédente, un certain nombre de propriétés permettent de parcourir l'arborescence DOM à partir du nœud sélectionné.

Les propriétés qui suivent prennent également en compte les nœuds de texte et de commentaires, pas seulement les nœuds d'éléments.

Rappelons qu'un) nœud dans le modèle DOM, est l'un des objet suivants : le document lui même, les éléments, le contenu textuel d'un élément, et les commentaires.

Pour manipuler les nœuds d'un document HTML il faut savoir les sélectionner. La sélection d'un élément peut se faire :

– par son attribut `id` en utilisant la méthode getElementById() qui sélectionne l'unique élément du document dont l'id est fourni en paramètre.

– par son attribut class en utilisant la méthode getElementsByClassName() qui retourne tous les éléments dont la classe fournie en paramètre, sous forme d'un tableau.

– par le nom de l'élément en utilisant la méthode getElementsByTagName() qui sélectionne tous les éléments dont le nom est fournie en paramètre, sous forme d'un tableau.

– par un sélecteur CSS en utilisant la méthode querySelectorAll() sélectionne les éléments retenus par le sélecteur CSS fourni en paramètre, sous forme d'un tableau. querySelector est similaire mais ne fournit que le premier élément.

Exemples de sélecteur CSS

sélecteur de classe : document.querySelector(".nomClasse");

sélecteur d'élément : document.querySelector("div");

sélecteur id : document.querySelector("#main");

Propriétés

La propriété innerHTML ou textContent est utile pour récupérer ou remplacer le contenu des éléments HTML.

element.attribute = NouvelleValeur : Changer la valeur d'attribut d'un élément HTML

Sélection d'autres nœuds :

Le modèle DOM presente un document HTML comme une arborescence. Chaque nœud est un objet et possède un ensemble de propriétés :

– nodeName : renvoie le nom de l'élément du nœud (ou `#text` pour les nœuds de texte)

– nodeType : type du nœud.

– nodeValue : renvoie la valeur d'un nœud

– parentNode : permet d'accéder au nœud parent.

– childNodes : retourne tous les nœuds enfant (sous forme de tableau) – fitstChild : permet d'accéder au premier nœud enfant;

– lastChild : permet d'accéder au dernier nœud enfant;

– previouSibling : nœud précédent au même niveau (à gauche).

– nextSibling : nœud suivant au même niveau (à droite);

Le Tableau 5.1 contient une liste de toutes les valeurs possibles de nodeType avec leur significations.

Type de nœud	Description
1	un nœud élément (comme p ou div)
2	Attribut
3	Texte (espaces compris)
8	un nœud Commentaire HTML
9	nœud Document lui même
10	le nœud doctype
11	Fragment

TABLE 5.1 – Signification de type de nœuds retournée parnodeType

La propriété childNodes renvoie un objet NodeList qui est la liste de nœuds enfants d'un élément, en lecture seule. childNodes[0] est identique à firstChild

childNodes renvoie les nœuds enfants : nœuds d'élément, nœuds de texte et nœuds de commentaire. Les espaces entre les éléments sont également des nœuds de texte.

Siblings sont les ″frères″ et les ″soeurs″.

Siblings sont des nœuds avec le même parent (dans la même liste childNodes).

Récupérer tous les éléments ayant 'abc' and 'xyz' comme classes : document.getElementsByClassName xyz″);

Récupère tous les éléments qui ont une classe ″abc″, à l'intérieur d'un élément qui a l'ID ″xyz″ : document.getElementById(″xyz″).getElementsByClassName(″abc″);

– sélection par attribut name sur certains éléments

5.2.1 propriétés

L'objet document possède également des propriétés :

document.documentElement;

document.head;

document.body;

5.3 Manipulation des attributs des éléments

Une fois nous avons sélectionné un nœud élément, les méthodes que nous allons voir permettent d'agir sur ses attributs.

Les nœuds d'attribut sont accessibles via leur élément conteneur, au lieu d'être des nœuds enfants de cet élément.

Tous les attributs sont accessibles par les méthodes suivantes :

elem.hasAttribute(nomAtt) : vérifie l'existance de l'attribut nomAtt et retourne un booléen.

elem.getAttribute(nomAtt) : récupère la valeur de l'attribut nommé nomAtt.

elem.setAttribute(nomAtt,valeur) : affecte valeur comme valeur de l'attribut nommé nomAtt.

elem.removeAttribute(nomAtt) : supprime l'attribut nommé nomAtt.

La méthode setAttribute() ajout un attribut à un élément. La syntaxe est

element.setAttribute("attribut","valeurAttribut")

permet de définir ou modifier la propriété "style" d'un élément et là on travaille directement avec CSS.

element.setAttribute("style,"définition CSS .....") fournir le nom de l'attribut et sa valeur

Mais attention : cette opération écrase tous les autres styles inline associés à cet élément (à vérifier si cela comprend les styles hérités, sorry !). A utiliser avec prudence en tout cas. var el = document.getElementById("some-element"); el.setAttribute("style", "background-color :darkblue ;");

getAttribute()

```
<script type="text/javascript">
var link = document.getElementById('myLink');
var href = link.getAttribute('href'); // On récupère l'attribut "href"
alert(href);
link.setAttribute('href','http://www.google.com'); // on édite
</script>
```

Remplacement d'éléments replaceChild(newNode, list.firstChild);

5.4 Manipulation du contenu

la propriété innerHTML représente le contenu HTML d'un élément.

la propriété textContent représente le contenu textuel d'un élément.

Méthode d'abonnement addEventListener La méthode addEventListener permet d'abonner à l'objet sur lequel elle est invoquée une fonction pour l'événement précisé. objet.addEventListener(eventType, listenerFunction)

Insertion – noeudParent.insertBefore(noeudInséré,noeudRéférence) : insère noeudInséré avant noeudRéférence comme fils de noeudParent

– parent.appendChild(noeudAjouté) : le nœud noeudAjouté est ajouté à la fin des fils de parent

si le nœud inséré ou ajouté existe dans le document, il est alors déplacé (donc supprimé de la position existante et inséré/ajouté à la position demandée).

5.5 Suppression de nœuds

Suppression et remplacement

– parent.removeChild(noeud) : noeud est supprimé des fils de parent. appeler cette méthode à partir de l'élément parent du nœud à effacer et fournir le nœud en paramètre.

– parent.replaceChild(remplac¸ant,remplacé) : remplaçant prend la place de remplacé comme fils de paren

Ajout d'éléments

La méthode createElement() créé un nouvel élément, en utilisant le nom de l'élément fourni en paramètre.

La méthode appendChild() : ajoute un nouvel enfant à la fin de la liste.

La méthode insertBefore() : permet d'insérer un nouveau nœud avant.

La méthode createTextNode() crée ce nœud de texte ; il suffit de fournir le texte lui-même.

5.6 Création de nœuds

la méthode createTextNode() permet de placer du texte dans un élément.

1. Arbre initial

```
<div id="mon_div">
<ul><li>toto</li></ul>
</div>
```

2. Execution du code

```
var zone=document.getElementById("mon_div");
var p=document.createElement("p");
var texte=document.createTextNode("blablabla ...");
p.appendChild(texte);
zone.appendChild(p);
```

3. Nouvel arbre

```
<div id="mon_div">
<ul><li>toto</li></ul>
<p>blablablabla ...</p>
</div>
```

La méthode :

cloneNode() : permet de cloner un nœud existant.

cloneNode(true) : clone le nœud et ses enfants.

cloneNode(false) : clone sans les enfants.

Si vous fournissez true en paramètre à cloneNode(), une ”copie en profondeur” est effectuée, laquelle prend en compte également les enfants ; false ne copie que le nœud lui-même.

5.7 Changement de style

Pour changer le style d'un élément HTML, on utilise la syntaxe :

element.style.property = nouveauStyle

Agir sur les propriétés CSS

Tout élément HTML dispose d'un attribut style en JavaScript. Les noms des propriétés CSS doivent être convertis en camel case. Par exemple :

– la propriété style d'un élément permet d'agir sur les propriétés CSS de cet élément définies via l'attribut `style` ou dans le document HTML, mais elle ne permet pas d'accéder aux valeurs des propriétés définies dans une feuille de style.

– on utilise directement le nom de la propriété CSS après ”conversion camelback” si nécessaire c'est à dire font-size devient fontSize, border-right-style devient borderRight-Style, etc.

– les valeurs sont toujours des chaînes de caractères

– les unités doivent être précisées

```
<script type="text/javascript">
let e = document.getElementById("abc") ;
e.style.fontWeight = " bold " ;
e.style.fontSize = " 12 px " ; // l'unité est obligatoire
e.style.marginRight = " 10 px " ;
e.style.marginTop = " 2% " ;
e.style.backgroundColor = " rgba (128 ,0 ,0 ,0.5) " ;
let r = element.style.marginRight ;
let R = parseInt(r)+50;
e.style.marginRight = R + "px" ;
```

```
  </script>
```

getComputedStyle la méthode getComputedStyle de l'objet window permet d'obtenir les valeurs des propriétés CSS appliquées par le navigateur

```
<html>
<body>
  <div id="abc">DOM est génial, non !</div>

<script>
  x=document.getElementById("abc")
  x.style.color = "blue";
  x.style.fontSize = "28pt";
  x.style.backgroundColor = "red";
</script>

</body>
</html>
```

5.8 Exercices

Exercice 16. Écrire un script qui alterne la couleur des éléments d'une liste.

6

Gestion des événements

Un événement est une réaction à l'aide de scripts JavaScript, à une action de l'utilisateur sur un document HTML. Par exemple un clic de la souris (action) sur un lien ouvre (réaction) une autre page Web.

Exemples d'action : clique sur un bouton, survoler une image, cliquer dans un champs texte, quitter un champs texte, ouverture d'un document HTML, fermeture d'un document HTML, envoyer un formulaire, appuyer sur une touche du clavier, redimensionner une fenêtre du navigateur.

etc ...

Exemples de réaction : affichage d'un message d'alerte, ouverture d'une fenêtre Pop-up, fermeture d'une fenêtre, effectuer un calcul, etc...

Les événements JavaScript permettent l'interactivité avec la machine. La plupart des langages de programmation ont un certain type de modèle d'événement.

Pour certaines balises HTML est associé un ensemble d'événements. Un événement en JavaScript est un attribut de balise. Tous les événements commence par 'on'. Par exemple : OnClick, onMouseOver, ondblclick, etc ...

Le Tableau 6.1 donne la liste des événements et les balises auquelles ils sont associées. Certains événements peuvent ne pas exister dans toutes les versions de JavaScript .

La syntaxe est

```
<balise onEvenement= "code js ou fonction js"> ... </balise>
```

Exemple : Quand on clique sur l'image "img2.jpeg", elle est remplacée par l'image "img32.jpeg".

```
<img src="img2.jpeg" onClick="document.images[0].src='img3.jpeg'"/>
```

Losque le code JavaScript est assez long, il est préférable de l'écrire sous forme d'une fonction et d'appeler cette fonction dans l'événement.

Exemple :

```
<script type="text/javascript">
function effacerImg(){
  for(i=0;i<document.images.length;i++){document.images[i].src='';}
}
</script>
Cliquez pour cacher toutes les images de ce document
<form>
<input type="button" value="Cacher toutes les images" onClick="effacerImg()
    ">
</form>
<img src="img2.jpeg"> <br>
<img src='img3.jpeg'>
```

Dans l'exemple suivant, on utilise l'événement onKeyUp. Le contenu du champs text est mis en majuscule.

```
<script type="text/javascript">
function Maj() {
  var C = document.getElementById("abc");
  C.value = C.value.toUpperCase();}
</script>
<form>
Nom : <input type="text" id="abc"  onKeyUp="Maj()">
</form>
```

L'exemple suivant utilise l'événement onResize pour afficher la taille de la fenêtre en cours.

```
<body onResize="modifier()">
<script>
function modifier() {
  var w = window.outerWidth;
  var h = window.outerHeight;
  var texte = "<br> Largeur=" + w +',<br>'+ " hauteur=" + h;
```

```
  document.getElementById("abc").innerHTML = texte;
}
</script>
<p>Redimensionner votre fenêtre pour voir sa taille </p>
```

Événements en JavaScript

6.1 Exercices

Événement	**Se produit quand**	**S'applique à** :
OnAbort	l'utilisateur a arrêté le chargement de l'image.	images
onBlur	on quitte la fenêtre ou un objet de formulaire	Button, Checkbox, FileUpload, Layer, Password, Radio, Reset, Select, Submit, Text, TextArea, window
onChange	un élément de formulaire est modifié	FileUpload, Select, Submit, Text, TextArea
onClick	on clique dans ou sur un élément	Button, document, Checkbox, Link, Radio, Reset, Select, Submit
dblclick	double clique de souris	document, Link
onDragDrop	on glisse un élément sur une fenêtre à l'aide la souris	fenêtres
onError	le chargement de l'image ou de la fenêtre provoque une erreur	Images, fenêtres
onFocus	on sélectionne la fenêtre ou l'objet formulaire	Button, Checkbox, FileUpload, Layer, Password, Radio, Reset, Select, Submit, Text, TextArea, window
onDlClick	on fait un double clic du bouton gauche de la souris	Boutons, boutons radio, boutons submit et reset, liens
onKeyDown	une touche du clavier est pressée	Documents, images, liens, zones texte
onKeyPress	on appuie et maintient une touche du clavier	Documents, images, liens, zones texte
onKeyUp	on relâche une touche du clavier.	Documents, images, liens, zones texte
onLoad	le document se charge	Documents, images, window
onMouseDown	on clique avec le bouton de la souris	Documents, boutons, liens
onMouseMove	on bouge la souris	rien par défaut
onMouseOut	le pointeur de la souris sort d'une zone de sélection graphique ou un lien	Cartes, liens
onMouseOver	le curseur passe au dessus d'un lien	area, liens
onMouseUp	on relâche le bouton de la souris	Documents, boutons, liens
onMove	l'utilisateur ou un script bouge une fenêtre	fenêtres
onReset	on réinitialise un formulaire	Formulaires
onResize	l'utilisateur ou un script change la taille d'une fenêtre	window
onSelect	on sélectionne une zone ou un champ texte (clavier ou souris)	Champs ou zones texte
onSubmit	on envoie un formulaire.	Formulaire
onUnLoad	on ferme la fenêtre.	Documents

TABLE 6.1 – Événements en JavaScript

Événement	se produit quand	s'applique à
onAbort	l'utilisateur a arrêté le chargement de l'image.	images
onBlur	on quitte la fenêtre ou un objet de formulaire	Fenêtres, éléments de formulaire
onChange	un élément de formulaire est modifié	Champs texte, zones texte, listes de sélection
onClick	on clique dans ou sur un élément	Boutons, boutons radio, boutons submit et reset, liens
onDragDrop	on glisse un élément sur une fenêtre à l'aide la souris	fenêtres
onError	le chargement de l'image ou de la fenêtre provoque une erreur	Images, fenêtres
onFocus	on sélectionne la fenêtre ou l'objet formulaire	Fenêtres ,éléments de formulaire
onDblClick	on fait un double clic du bouton gauche de la souris	Boutons, boutons radio, boutons submit et reset, liens
onKeyDown	une touche du clavier est presséeDouble-click this paragraph to trigger a function.Double-click this paragraph to trigger a function.	Documents, images, liens, zones texte
onKeyPress	on appuie et maintient une touche du clavier	Documents, images, liens, zones texte
onKeyUp	on relâche une touche du clavier.	Documents, images, liens, zones texte
onLoad	le document se charge	Documents
onMouseDown	on clique avec le bouton de la souris	Documents, boutons, liens
onMouseMove	on bouge la souris	rien par défaut
onMouseOut	le pointeur de la souris sort d'une zone de sélection graphique ou un lien	Cartes, liens
onMouseOver	le curseur passe au dessus d'un lien	liens
onMouseUp	on relâche le bouton de la souris	Documents, boutons, liens
onMove	l'utilisateur ou un script bouge une fenêtre	fenêtres
onReset	on réinitialise un formulaire	Formulaires
onResize	l'utilisateur ou un script change la taille d'une fenêtres	fenêtres
onSelect	on sélectionne une zone ou un champ texte (clavier ou souris)	Champs ou zones texte
onSubmit	on envoie un formulaire.	Formulaires
onUnLoad	on ferme la fenêtre.	Documents

TABLE 6.2 – Événements en JavaScript

Objet	**Événements associables**
Lien hypertexte	onClick, onMouseOver, onMouseOut
Fenêtre	onLoad, onUnload
Bouton, Case à cocher, Boutons : radio, submit, Reset	onClick
Liste de sélection d'un formulaire	onBlur, onChange, onFocus
Bouton Submit	onSubmit
Champ de texte et zone de texte	onBlur, onChange, onFocus, onSelect

TABLE 6.3 – Événements associables à un objet

7

Programmation orientée objet en Javascript

Il y a trois paradigmes (façons de programmer) de programmation largement utilisés : La programmation procédurale : le code est un enchaînement de procédures pour résoudre le problème. La programmation fonctionnelle : le code est un enchaînement de fonctions pour résoudre le problème. La programmation orientée objet (POO) : on identifie les acteurs (=objets) du problème puis on détermine la façon dont ils doivent interagir pour résoudre le problème.

La programmation orientée objet est une nouvelle façon de concevoir et de développer des applications informatiques. Elle permet une plus grande modularité, ainsi qu'une meilleure lisibilité du code et une meilleure maintenabilité du code.

7.1 Défitions d'objet et classe

La POO consiste à représenter des objets du monde réel dans un programme informatique.

Pour représenter un objet du monde réel dans un programme informatique, on considère un ensemble de données appelées aussi attributs ou propriétés de cet objet et un ensemble d'actions appelées aussi méthodes de cet objet.

Classe : voiture
Attribus ou propriétés
Marque
Modèle
Couleur
...
Action ou méthodes
Démarrer
rouler
accélérer
...

TABLE 7.1 – Une classe voiture

Chaque attribut ou propriété est caractérisé par sa valeur. Une méthode s'appelle aussi fonction membre, est une action applicable à son objet.

Exemples d'objet :

– compte bancaire : a pour attributs ou propriétés numéro, nom de son propriétaire, solde, type de compte etc et comme méthodes débiter, accréditer, afficher solde, bloquer, etc

– footballeur : a pour attributs ou propriétés nom, numéro qu'il porte etc et a pour actions ou méthodes dribbler, courir, attraper le ballon, marquer le but etc

– voiture : a pour attributs marque, modèle, couleur, et a comme actions ou méthodes démarrer, rouler, accélérer etc...

Représentation graphique d'une classe. Voir Table 7.1

– Etudiant : nom, code, filière, modules suivis, notes etc et a comme actions ou méthodes valider, passer session de rattrapage, etc

Une classe est une description d'un ensemble d'objets ayant une structure de données commune (attributs ou propriétés) et disposant des mêmes méthodes (ou actions). Ainsi, tous les objets d'une même classe ont en commun les mêmes propriétés et les mêmes méthodes.

Une classe est caractérisée par ses attributs ou propriété et ses méthodes.

Une classe peut être vue comme un modèle d'objet ou comme un moule (voir Figure 7.2) qui permet de créer autant d'objets de même type et de même structure. L'instanciation est l'opération de créer un objet d'une classe. La notion de classe généralise celle de type.

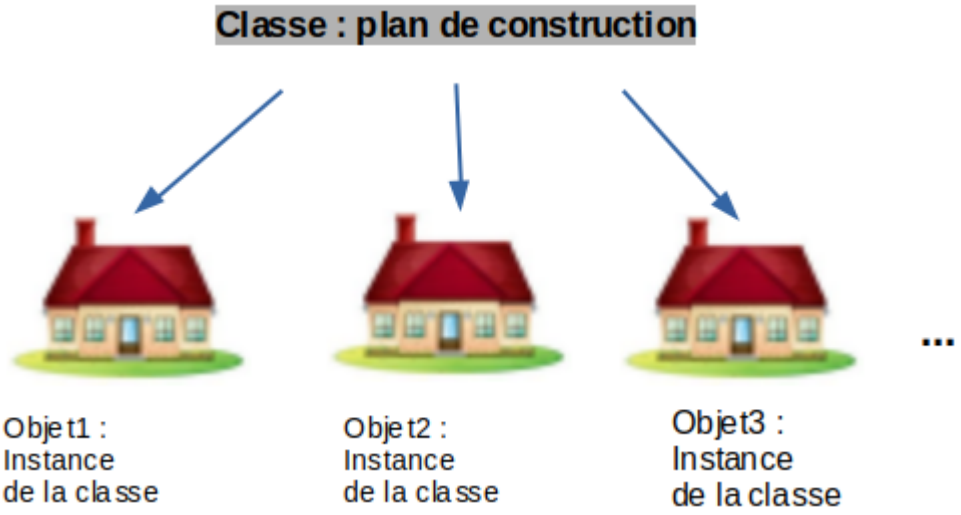


FIGURE 7.1 – Instanciation : création d'objets à partir d'une classe

FIGURE 7.2 – Le moule (la classe) permet de créer autant d'objets qu'on veut

Dans la pratique, on commence par créer une classe en déclarant ses propriétés, et programmant ses méthodes. Lorsqu'on déclare qu'un objet est de cette classe, alors cet objet obtient automatiquement toutes les propriétés et méthode définis dans sa classe. Il ne reste plus qu'à spécifier les valeurs des propriétés pour chaque objet.

Certains langages de programmation supportent la POO et d'autres non. Par exemple C n'est pas orienté objet. Par contre son extention C++ l'est.

Il y a deux grandes catégories de langages à objets :

– les langages à classes, comme C++, Java, Python et d'autres.

– les langages à prototypes, comme JavaScript, NewtonScript, Self, Lua et d'autres.

La programmation orientée prototype est un style de programmation orientée objet qui n'utilise pas les classes.

7.2 JavaScript et programmation orienté objet

Comme JavaScript est spécialement fait pour améliorer le langage HTML, fait que la plus part de ses objets utilisés sont prédéfinis ainsi que leur propriétés et méthodes. La Figure 7.3 présente tous les objets prédéfinis en JavaScript . On peut exceptionnellement définir des classes et des objets en JavaScript , mais ceux prédéfinis sont largement suffisants.

Les mécanismes relatifs à la programmation orientée objet sont différents de ceux de Java ou C++. En JS la notion de classe et le concept d'héritage n'existent.

Les méthodes se terminent toujours avec des parenthèses. Elles peuvent recevoir ou non des paramètres. Les propriétés ne supportent pas les parenthèses.

En Javascript, pour accéder à une propriété d'un objet, on utilise la syntaxe :

```
Objet.propriété
```

et pour modifier une propriété

```
Objet.propriété=nouvelleValeurDePropriété
```

Par exemple

```
<html>
<body><head><title>Ma page Web </title></head></body>
<script type="text/javascript">
document.write(document.title); // retourne 'Ma page Web'
document.title='Ma nouvelle page Web'
// le titre de la fenêtre devient 'Ma nouvelle page Web'
</script>
```

```
</body>
</html>
```

Certaines propriétés sont modifiables, par contre d'autres sont en lecture seule, elles permettent uniquement de récupérer des informations sur les objets.

Pour définir un nouveau objet

```
var monObjet = new Object();
```

Par exemple

```
<script type="text/javascript">
var maDte=new Date();//instanciation de l'objet maDte
document.write(maDte);
// retourne Thu Jan 28 2021 14:23:24 GMT+0100 (UTC+01:00)
document.write(maDte.getFullYear());
// retourne 2021
maDte.setFullYear(2029);
document.write(maDte.getFullYear());
// retourne 2029
</script>
```

```
Objet.methode()
```

Par exemple :

alert() est une méthode de l'objet window, on écrit window.alert("text d'alerte").

write() est une méthode de l'objet document, on écrit document.write("text1","text2",..).

Dans le prochain chapitre, nous allons voir pour chaque objet ses propriétés et ses méthodes.

7.3 Exercices

8

Fonctions avancées

9

Gestion des erreurs et débogage

10

Programmation asynchrone en Javascript

11

Les modules en JavaScript