

Initiation à la programmation à l'aide de JavaScript

Edité en décembre 2018
Cours donné par Technofutur TIC en formation à distance

PLUS HAUT
ET PLUS **PROCHE**
L'UNION EUROPÉENNE ET LA WALLONIE INVESTISSENT
DANS VOTRE AVENIR



Table des matières

Introduction

| | | |
|------|---|-------|
| 1 | Objectifs | p. 7 |
| 2 | Ce que vous allez voir dans ce cours..... | p. 7 |
| 3 | Ce que vous n'allez pas voir dans ce cours..... | p. 7 |
| 4 | Le déroulement du cours..... | p. 8 |
| 5 | Fonctionnement du cours..... | p. 8 |
| 6 | Naviguer dans le cours..... | p. 8 |
| 7 | Progression dans le cours..... | p. 10 |
| 8 | A qui s'adresse ce cours ?..... | p. 11 |
| 9 | Les prérequis du cours..... | p. 11 |
| 10 | Le matériel..... | p. 11 |
| 11 | Les exercices du cours..... | p. 12 |
| 12 | Test de prérequis..... | p. 13 |
| 13 | Test de prérequis (suite)..... | p. 13 |
| 14 | MEMO..... | p. 14 |
| 15.1 | Compléments..... | p. 15 |

Module 1 - Un premier programme

| | | |
|------|---|-------|
| 1 | Exercice : Ecrire dans le document..... | p. 16 |
| 2 | Objectifs..... | p. 16 |
| 3 | Préparation..... | p. 16 |
| 4 | Afficher du texte..... | p. 17 |
| 5 | Ecriture du code..... | p. 17 |
| 6 | L'exécution du programme..... | p. 18 |
| 7 | Quizz..... | p. 19 |
| 8 | Ce que l'on a fait..... | p. 19 |
| 9 | Explication 1/5 - Le code complet..... | p. 19 |
| 10 | Explication 2/5 - les balises HTML..... | p. 20 |
| 11 | Explication 3/5 - le script..... | p. 21 |
| 12 | Explication 4/5 - la fonction Main();..... | p. 21 |
| 13 | Explication 5/5 - l'instruction alert();..... | p. 22 |
| 14 | Synthèse..... | p. 23 |
| 15 | Appliquez : afficher une alerte..... | p. 23 |
| 16 | Afficher du texte dans le document..... | p. 24 |
| 17 | Pourquoi Main() ?..... | p. 24 |
| 18 | Fonction principale..... | p. 25 |
| 19 | Quizz..... | p. 25 |
| 20 | Synthèse..... | p. 25 |
| 21 | Mémo..... | p. 26 |
| 22.1 | Compléments..... | p. 29 |
| 22.2 | Rappel : Structure HTML de base..... | p. 29 |
| 22.3 | Exécuter un programme JavaScript..... | p. 29 |
| 22.4 | Une solution encore plus simple !..... | p. 29 |
| 22.5 | Les événements..... | p. 30 |
| 22.6 | Les commentaires JavaScript..... | p. 32 |
| 22.7 | Ecrire dans le document..... | p. 33 |

Module 2 - Communiquer avec la machine

| | | |
|---|---|-------|
| 1 | Objectifs..... | p. 34 |
| 2 | Communiquer à l'aide des formulaires..... | p. 34 |
| 3 | Rappel sur les formulaires..... | p. 34 |
| 4 | Les balises des formulaires..... | p. 35 |
| 5 | Un programme qui écrit..... | p. 35 |
| 6 | Le code du programme..... | p. 35 |
| 7 | Appliquez : dire bonjour dans un champ texte..... | p. 36 |

| | | |
|-------|--|-------|
| 8 | Conclusion..... | p. 37 |
| 9 | Quizz..... | p. 38 |
| 10 | Un programme qui lit..... | p. 38 |
| 11 | Exemple de programme qui lit..... | p. 38 |
| 12 | Appliquez : demander le prénom..... | p. 39 |
| 13 | Conclusion..... | p. 40 |
| 14 | Quizz..... | p. 41 |
| 15 | Synthèse..... | p. 41 |
| 16 | Exercice récapitulatif..... | p. 41 |
| 17 | Mémo..... | p. 43 |
| 18.1 | Compléments..... | p. 46 |
| 18.2 | Comment passer de l'énoncé au programme ?..... | p. 46 |
| 18.3 | Ce que l'on a fait..... | p. 47 |
| 18.4 | Langage machine et langage de programmation..... | p. 47 |
| 18.5 | Conclusion..... | p. 48 |
| 18.6 | Quizz..... | p. 48 |
| 18.7 | Historique des langages de programmation..... | p. 48 |
| 18.8 | Le JavaScript, un langage de programmation..... | p. 49 |
| 18.9 | Quizz..... | p. 49 |
| 18.10 | JavaScript et l'interactivité..... | p. 49 |
| 18.11 | Quizz..... | p. 50 |
| 18.12 | La fenêtre "prompt"..... | p. 50 |
| 18.13 | La fenêtre "prompt" (suite)..... | p. 51 |
| 18.14 | Quizz..... | p. 51 |
| 18.15 | Les événements d'un formulaire..... | p. 51 |
| 18.16 | Appliquez : une fenêtre prompt..... | p. 52 |
| 18.17 | Quizz..... | p. 54 |
| 18.18 | Appliquez : l'événement onblur..... | p. 54 |

Module 3 - Un peu plus sur le JavaScript

| | | |
|------|--|-------|
| 1 | Objectifs | p. 56 |
| 2 | Qui interprète un programme JavaScript ?..... | p. 56 |
| 3 | Les particularités du JavaScript..... | p. 56 |
| 4 | La sensibilité à la casse..... | p. 57 |
| 5 | Appliquez : la sensibilité à la casse..... | p. 57 |
| 6 | Un ";" à la fin de chaque instruction..... | p. 58 |
| 7 | Appliquez ! Un ";" à la fin de chaque instruction..... | p. 58 |
| 8 | Les mots réservés..... | p. 59 |
| 9 | Les caractères spéciaux..... | p. 59 |
| 10 | Quizz..... | p. 60 |
| 11 | Accès à un objet du document..... | p. 60 |
| 12 | Appliquez : accès à un objet du document..... | p. 60 |
| 13 | Conclusion..... | p. 61 |
| 14 | L'erreur est humaine..... | p. 61 |
| 15 | Comment détecter une erreur ?..... | p. 61 |
| 16 | Comment retrouver une erreur ?..... | p. 62 |
| 17 | Les points importants à vérifier..... | p. 63 |
| 18 | Conclusion : Les bonnes manières pour programmer..... | p. 63 |
| 19 | Mémo..... | p. 65 |
| 20.1 | Compléments..... | p. 67 |
| 20.2 | Langage compilé / Langage interprété..... | p. 67 |
| 20.3 | Quizz..... | p. 67 |
| 20.4 | Liste des mots réservés..... | p. 67 |
| 20.5 | Liste des caractères spéciaux..... | p. 68 |
| 20.6 | Savoir lire les messages d'erreur..... | p. 69 |
| 20.7 | Les erreurs JavaScript avec Firefox..... | p. 70 |

Module 4 - Stocker des valeurs

| | | |
|---|----------------|-------|
| 1 | Objectifs..... | p. 71 |
|---|----------------|-------|

| | | |
|------|--|-------|
| 2 | Dire plusieurs fois la même chose..... | p. 71 |
| 3 | Solution de l'exercice..... | p. 71 |
| 4 | Conclusion..... | p. 72 |
| 5 | Une solution plus "propre"..... | p. 72 |
| 6 | Ce que l'on a fait..... | p. 73 |
| 7 | Un programme qui lit (cf. module 2)..... | p. 73 |
| 8 | Bien déclarer une variable..... | p. 74 |
| 9 | Quizz..... | p. 75 |
| 10 | Appliquez : utiliser les variables..... | p. 75 |
| 11 | Quizz..... | p. 77 |
| 12 | Manipulez plusieurs variables..... | p. 77 |
| 13 | Explication du code 1/3..... | p. 77 |
| 14 | Explication du code 2/3..... | p. 78 |
| 15 | Explication du code 3/3..... | p. 78 |
| 16 | Conclusion..... | p. 79 |
| 17 | Appliquez : la concaténation..... | p. 79 |
| 18 | Quizz..... | p. 81 |
| 19 | Synthèse..... | p. 81 |
| 20 | Mémo..... | p. 82 |
| 21.1 | Compléments..... | p. 84 |
| 21.2 | Déclarer une variable ?..... | p. 84 |
| 21.3 | Quizz..... | p. 85 |
| 21.4 | Texte, nombres, booléens, | p. 85 |
| 21.5 | Travailler sur du texte..... | p. 86 |
| 21.6 | Quizz..... | p. 87 |
| 21.7 | Encore du travail sur le texte !..... | p. 87 |

Module 5 - Travailler avec des nombres

| | | |
|------|--|--------|
| 1 | Objectifs..... | p. 89 |
| 2 | Des variables numériques..... | p. 89 |
| 3 | Des opérations mathématiques avec des variables..... | p. 90 |
| 4 | L'addition..... | p. 90 |
| 5 | Appliquez : l'addition..... | p. 90 |
| 6 | L'addition 1/4..... | p. 91 |
| 7 | L'addition 2/4..... | p. 91 |
| 8 | L'addition 3/4..... | p. 92 |
| 9 | L'addition 4/4..... | p. 93 |
| 10 | Quizz..... | p. 94 |
| 11 | Soustraction..... | p. 94 |
| 12 | Appliquez : la soustraction..... | p. 94 |
| 13 | Multiplication..... | p. 95 |
| 14 | Division..... | p. 96 |
| 15 | Appliquez : la calculatrice..... | p. 96 |
| 16 | La calculatrice, les étapes du programme..... | p. 96 |
| 17 | Conclusion..... | p. 99 |
| 18 | Mémo..... | p. 100 |
| 19.1 | Compléments..... | p. 103 |
| 19.2 | L'opérateur +..... | p. 103 |
| 19.3 | Utilisation de la commande parseFloat()..... | p. 103 |
| 19.4 | Convertir un nombre en une chaîne de caractères..... | p. 103 |
| 19.5 | Convertir une chaîne de caractères en un nombre..... | p. 104 |
| 19.6 | Type texte ou type numérique ?..... | p. 105 |
| 19.7 | Quizz..... | p. 106 |
| 19.8 | Exercice de synthèse..... | p. 106 |
| 19.9 | Quizz..... | p. 106 |

Module 6 - Ajouter une condition

| | | |
|---|----------------------------------|--------|
| 1 | Objectifs..... | p. 107 |
| 2 | Une condition dans l'énoncé..... | p. 107 |

| | | |
|------|--|--------|
| 3 | Une structure conditionnelle..... | p. 107 |
| 4 | Le code du programme..... | p. 108 |
| 5 | Conclusion..... | p. 109 |
| 6 | Exercice : une condition..... | p. 109 |
| 7 | Comparer 2 valeurs..... | p. 110 |
| 8 | Quizz..... | p. 110 |
| 9 | Appliquez : comparer des valeurs numériques..... | p. 110 |
| 10 | Travailler avec plusieurs conditions..... | p. 111 |
| 11 | Un nombre au hasard entre 0 et 100..... | p. 112 |
| 12 | La solution de l'exercice..... | p. 113 |
| 13 | Conclusion..... | p. 114 |
| 14 | Appliquez : plusieurs conditions..... | p. 115 |
| 15 | Synthèse..... | p. 116 |
| 16 | Mémo..... | p. 117 |
| 17.1 | Compléments..... | p. 120 |
| 17.2 | Imbrication de structures conditionnelles..... | p. 120 |
| 17.3 | Conclusion..... | p. 120 |
| 17.4 | Plusieurs comparaisons à la fois..... | p. 121 |
| 17.5 | Encore des comparaisons..... | p. 122 |
| 17.6 | En résumé..... | p. 122 |
| 17.7 | Défi ! Un petit quizz..... | p. 123 |

Module 7 - Répéter une action

| | | |
|------|--|--------|
| 1 | Objectifs..... | p. 124 |
| 2 | Dire plusieurs fois la même chose..... | p. 124 |
| 3 | Conclusion..... | p. 125 |
| 4 | Appliquez : dire plusieurs fois la même chose..... | p. 125 |
| 5 | Quizz..... | p. 126 |
| 6 | Le squelette d'une boucle..... | p. 126 |
| 7 | Répéter 5 fois un traitement..... | p. 127 |
| 8 | Quizz..... | p. 128 |
| 9 | Exercice : compter jusqu'à 5..... | p. 128 |
| 10 | Exercice : compter à rebours..... | p. 128 |
| 11 | Quizz..... | p. 129 |
| 12 | Exemple : sauter des valeurs..... | p. 129 |
| 13 | Conclusion..... | p. 130 |
| 14 | Quizz..... | p. 130 |
| 15 | Synthèse..... | p. 130 |
| 16 | Mémo..... | p. 132 |
| 17.1 | Compléments..... | p. 135 |
| 17.2 | L'incrément..... | p. 135 |
| 17.3 | Quizz..... | p. 135 |
| 17.4 | Une condition dans une boucle..... | p. 135 |
| 17.5 | Conclusion..... | p. 137 |
| 17.6 | Appliquez : un if dans un for..... | p. 137 |
| 17.7 | Sortir d'une boucle..... | p. 138 |
| 17.8 | Défi ! Ecrire de plus en plus grand..... | p. 138 |

Module 8 - Grouper plusieurs valeurs

| | | |
|----|---|--------|
| 1 | Objectifs | p. 140 |
| 2 | Une collection de bandes dessinées..... | p. 140 |
| 3 | 2e solution : utilisation d'un tableau..... | p. 141 |
| 4 | Atteindre une donnée d'un tableau..... | p. 142 |
| 5 | Conclusion..... | p. 143 |
| 6 | Quizz..... | p. 143 |
| 7 | La propriété "length"..... | p. 143 |
| 8 | Quizz..... | p. 144 |
| 9 | Afficher toutes les valeurs d'un tableau..... | p. 144 |
| 10 | Quizz..... | p. 145 |

| | | |
|------|---|--------|
| 11 | Synthèse..... | p. 145 |
| 12 | Mémo..... | p. 147 |
| 13.1 | Compléments..... | p. 150 |
| 13.2 | Trois manières de créer un tableau..... | p. 150 |
| 13.3 | Trier un tableau..... | p. 150 |
| 13.4 | Afficher la date..... | p. 151 |

Module 9 - Utiliser des fonctions

| | | |
|------|---------------------------------------|--------|
| 1 | Objectifs | p. 152 |
| 2 | La fonction Main()..... | p. 152 |
| 3 | Créer une fonction..... | p. 152 |
| 4 | Quizz..... | p. 153 |
| 5 | Appeler une fonction..... | p. 153 |
| 6 | Exemple : une collection de BD..... | p. 154 |
| 7 | Un programme - Deux traitements..... | p. 154 |
| 8 | Le code détaillé du programme..... | p. 154 |
| 9 | Variable locale..... | p. 156 |
| 10 | Quizz..... | p. 156 |
| 11 | Les parenthèses d'une fonction..... | p. 156 |
| 12 | Conclusion..... | p. 157 |
| 13 | Appliquez : le jeu du bon numéro..... | p. 157 |
| 14 | Synthèse..... | p. 159 |
| 15 | Quizz (partie 1)..... | p. 159 |
| 16 | Quizz (partie 2)..... | p. 160 |
| 17 | Conclusion du cours..... | p. 160 |
| 18 | Ce qu'on a vu..... | p. 160 |
| 19 | Mémo..... | p. 162 |
| 20.1 | Compléments..... | p. 164 |
| 20.2 | Quelques ressources..... | p. 164 |

Réponses aux Quizz

| | |
|-------|--------|
| | p. 170 |
|-------|--------|

Introduction

1 : Objectifs

L'objectif principal de ce cours est de vous familiariser avec les **principes de base de la programmation**.

Comprendre les **concepts de la programmation** est l'étape primordiale d'un futur développeur, vient ensuite l'apprentissage d'un **langage de programmation** et de son **environnement**.

Pour éviter un cours trop théorique, nous avons choisi d'introduire dès le départ un **langage de programmation**.

Les exemples et les exercices vous permettront de comprendre et d'appliquer les principes de base dans des cas **concrets**.

Pour ce cours, nous avons fait le choix d'un langage de programmation simple et attrayant : le **JavaScript**.

Le JavaScript est un langage de programmation créé pour le Web et destiné à rendre les sites Internet plus **interactifs** (messages d'alerte, validation de formulaires, menus déroulants, images interverties, ...).

2 : Ce que vous allez voir dans ce cours

Ce cours a pour but de vous donner une première approche de la programmation, de manière très **pratique**.

Nous utiliserons le langage JavaScript pour les exemples et exercices.

A la fin de ce cours, vous aurez acquis des **connaissances générales** sur la programmation et vous serez capable d'utiliser le langage Javascript pour **créer de simples programmes**.

3 : Ce que vous n'allez pas voir dans ce cours

Ce cours n'a pas la prétention de vous rendre **EXPERT en programmation**, ni en un langage en particulier.

Il nous faudrait consacrer beaucoup plus de temps à cette tâche !

De même, ce cours n'a pas pour but de vous apprendre **toutes les particularités** du langage JavaScript, ni de ce que l'on appelle le **"Dynamic HTML"**.

Il est toutefois envisageable, suite à cette formation, d'approfondir ces matières...

4 : Le déroulement du cours

Dans les premiers modules, vous appréhendez la **programmation en général**.

A l'aide d'exemples et d'exercices, vous allez directement approcher "l'univers" de la programmation.

Ensuite, toujours par la pratique, vous approcherez plus précisément le **langage JavaScript** tout en continuant à apprendre les **principes de base de la programmation**.

5 : Fonctionnement du cours

Le cours est centré sur la **mise en pratique** des notions exposées : nous vous proposerons aussi souvent que possible d'exercer ce que vous avez appris, au travers de **questionnaires et d'exercices pratiques**.

Bien sûr, **plus** vous ferez d'exercices, **plus** vous retiendrez ce qui est présenté.

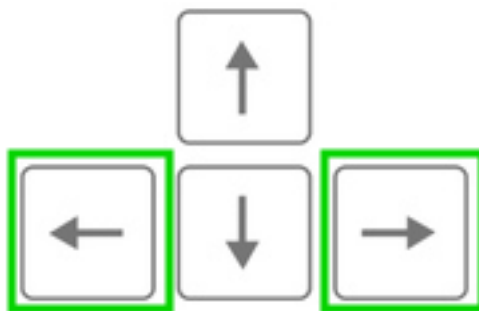
Nous vous donnerons aussi accès à des **ressources** pour aller plus loin dans les différents thèmes abordés.

Bien que le cours soit compatible avec les tablettes, **nous vous recommandons d'effectuer les exercices pratiques sur un ordinateur** (de bureau ou portable).

Pour plus de facilité, nous vous proposons de créer un dossier de départ appelé "monsite". Vous pouvez placer celui-ci sur votre bureau. Ce dossier sera utilisé durant toute la formation.

6 : Naviguer dans le cours

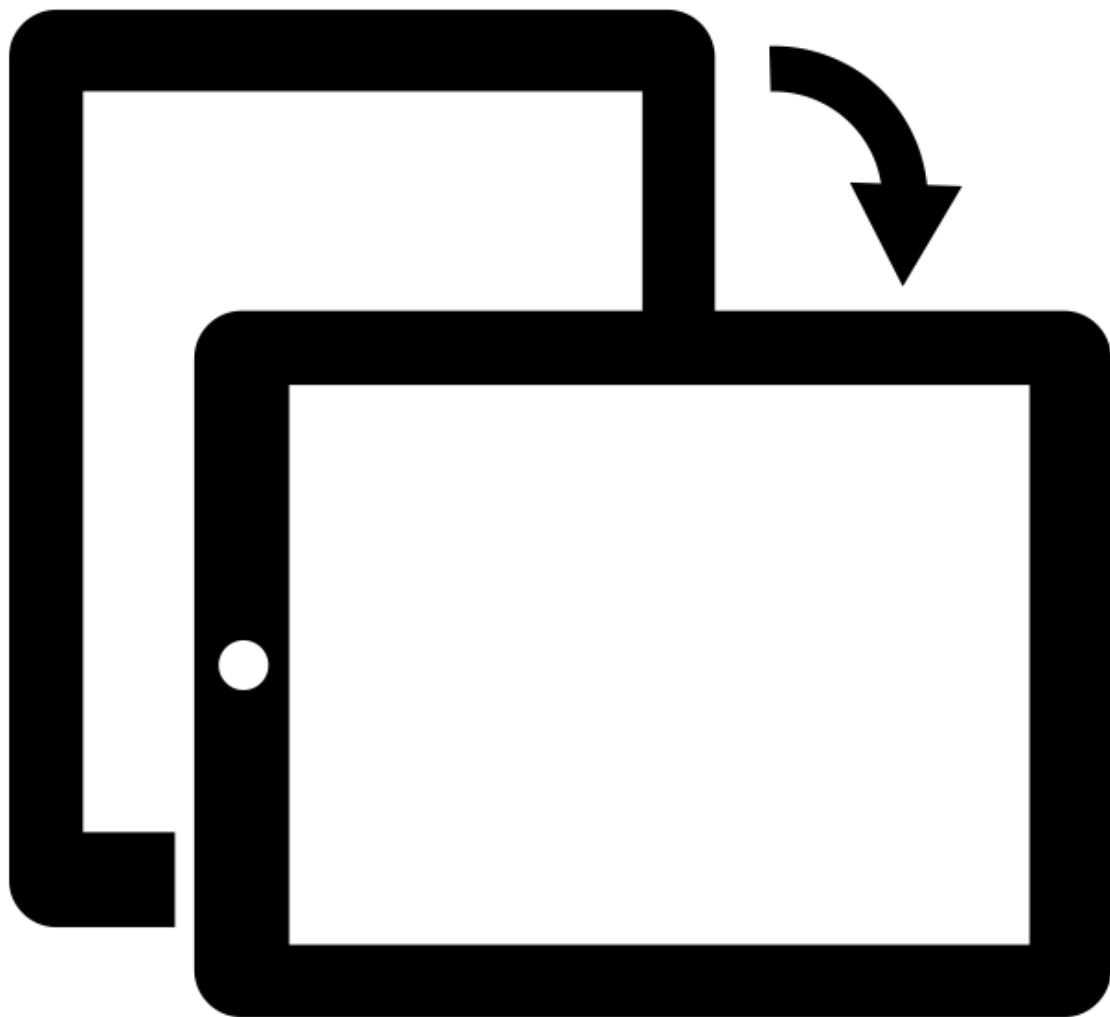
Vous pouvez également naviguer dans le cours en utilisant **les touches directionnelles (gauche et droite) de votre clavier**.



Le menu qui affiche la table des matières pour le module en cours est également cliquable.

Le cours est **responsive** : son format s'adapte à l'appareil sur lequel vous le consultez (ordinateur ou tablette) et suivant l'orientation (portrait ou paysage) de celui-ci.

Ce cours est consultable sur **tablette**, en mode **portrait**, ou **paysage**.



En mode portrait, le menu est accessible par le biais de cette icône.



Sur **tablette**, vous **pouvez balayer les contenus** vers la gauche ou vers la droite, pour avancer ou reculer dans le cours.



Vous pouvez **quitter le module** à tout moment en utilisant le bouton « **Quitter** » qui se trouve en haut à droite, ou en **fermant l'onglet** dans lequel le module est affiché.

7 : Progression dans le cours

Le cours **enregistre votre progression** dans la matière du cours, et vous informe de ce qui est vu ou ne l'est pas encore.

La progression est renseignée dans chaque module par le biais d'une **barre de progression**.



Dans la table des matières générale de la formation, les **boutons** d'accès aux modules vous indiquent votre progression pour chaque module. Ces éléments affichent le **pourcentage de matière vue**.



Les points des modules et les évaluations deviennent **accessibles au fur et à mesure** que vous progressez dans la formation. Les compléments ne font pas partie de la matière, et sont disponibles en tout temps, et ce peu importe la progression.

Il vous est possible de **désactiver cet outil de suivi** à tout moment, en cliquant sur l'**icône des engrenages** qui se trouve à droite dans la barre de progression.



Une fois le **suivi désactivé**, vous aurez le loisir de naviguer librement au sein de la formation. Sachez néanmoins qu'une fois désactivé, le suivi **n'enregistrera plus votre progression dans la matière**. Les évaluations soumises et notées ne sont pas affectées par cette désactivation.

Si vous réactivez le suivi de progression, le cours chargera votre **dernière progression enregistrée**.

8 : A qui s'adresse ce cours ?

Ce cours s'adresse aux personnes débutant en informatique et désireuses de se lancer dans le développement informatique par un apprentissage simple et concret.

Pas besoin de connaissances accrues dans le domaine de l'informatique, quelques prérequis suffisent pour commencer ce cours...

9 : Les prérequis du cours

Il est certain que vous ne pouvez pas commencer ce cours sans avoir quelques connaissances de base.

Que devez-vous savoir ?

Une bonne connaissance de l'**environnement Web** :

- Qu'est-ce qu'une page Web ?
- Qu'est-ce qu'un navigateur ?

Une bonne connaissance du **langage HTML** est nécessaire !

- Savoir créer une page HTML
- Connaître les principales balises HTML (<body>, <head>, <p>, <input>, ...)
- Savoir lire du code HTML simple

(N'hésitez pas à vous reporter au cours "Initiation au langage HTML" !)

10 : Le matériel

Voici ce dont vous avez besoin pour le cours:

- Un éditeur de texte comme le **Bloc-notes**, par exemple
Vous trouverez le Bloc-notes dans le menu **Démarrer > Programmes > Accessoires**
- Un navigateur (**Internet Explorer** de préférence)

Remarque : Ce cours a été conçu plus spécifiquement pour le navigateur Internet Explorer sur PC. Le langage JavaScript, au même titre que le langage HTML, peut présenter quelques différences d'un navigateur à l'autre (Google Chrome, Internet Explorer, Firefox...) et d'une plateforme à l'autre (PC, Mac, ...).

11 : Les exercices du cours

Les exercices du cours sont toujours présentés par un énoncé. A la suite de celui-ci, vous trouverez le nom du fichier entre parenthèses :

(nom du fichier : **ecrire.htm**)

Ce qui signifie que vous développerez vous-même l'exercice et l'enregistrerez sous le nom proposé dans un dossier particulier créé au préalable.

!!!! N'oubliez pas les **règles** pour les noms de fichiers :

A éviter :

- Espaces
- Accents
- Caractères spéciaux (# " @ . ; ? -)

(les majuscules/minuscules n'ont pas d'importance dans l'environnement Windows)

12 : Test de prérequis

1. Qu'est-ce qu'une page Web ?

- a) Un fichier qui commence par http://
- b) Un fichier texte dont l'extension est généralement .htm ou .html
- c) N'importe quelle ressource trouvée sur Internet

2. Qu'est-ce qu'un navigateur ?

- a) Un outil qui permet de visualiser des pages Web
- b) Un outil qui permet de créer des pages Web
- c) Un outil de traitement de texte

3. Lequel de ces logiciels n'est pas un navigateur ?

- a) Internet Explorer
- b) Firefox
- c) Wordpad

13 : Test de prérequis (suite)

1. Quelle balise trouve-t-on en premier dans un document HTML ?

- a) <html>
- b) <head>
- c) <title>

2. Laquelle de ces balises n'est pas obligatoire ?

- a) <html>
- b) <title>
- c) <body>

3. Que représente à l'écran ce morceau de code HTML ?<form><input type="Text"></form>

- a) Un simple texte
- b) Un champ texte dans un formulaire
- c) Rien, ces balises sont invisibles à l'écran

Si vous répondez facilement à ces questions, vous pouvez sans problème suivre ce cours.

Par contre, si vous éprouvez quelques difficultés pour certaines réponses, nous vous conseillons vivement de retourner voir le cours "Initiation au langage HTML " !

Bonne formation !

14 : MEMO

Il n'y a pas de mémo dans ce module.

15. 1 : Compléments

Il n'y a pas de compléments dans ce module.

Module 1 - Un premier programme

1 : Exercice : Ecrire dans le document

"Ecrire un programme qui affiche le texte "Bienvenue dans le monde de la bd !" au format titre1 (<h1>)."

```
<html>
\\r\\n\\t<head>\\r\\n\\t\\t<title></title>\\r\\n\\t\\t*script=\\r\\n\\t\\t\\tfunction  Main(){  \\r\\n\\t\\t\\t\\t
\\tdocument.write("Bienvenue dans le monde de la BD!")\\r\\n\\t\\t\\t\\t}\\r\\n\\t\\t=script*\\r\\n\\t</head>
\\r\\n\\t<body onload="Main();">\\r\\n\\t \\r\\n\\t</body>\\r\\n</html> <html>
\\r\\n\\t<head>\\r\\n\\t\\t<title></title>\\r\\n\\t\\t*script=\\r\\n\\t\\t\\tfunction  Main(){  \\r\\n\\t\\t\\t\\t
\\tdocument.write("<h1>Bienvenue dans le monde de la BD !</h1>")\\r\\n\\t\\t\\t\\t}\\r\\n\\t\\t=script*\\r
\\n\\t</head>\\r\\n\\t<body onload="Main();">\\r\\n\\t \\r\\n\\t</body>\\r\\n</html>
```

2 : Objectifs

Pour vous plonger directement dans l'univers de la programmation, nous vous proposons d'exécuter un premier exercice que vous allez développer de A à Z. Vous allez apprendre à **créer, éditer et visualiser** un document HTML contenant du **code JavaScript**. Bien sûr, toutes les étapes seront détaillées et expliquées !

Dans un deuxième temps, vous aurez l'occasion de découvrir quelques **exemples de programmes** écrits en JavaScript dans des pages HTML.

3 : Préparation

Tout au long de ce cours, vous allez appliquer la matière en effectuant des exercices ; c'est pourquoi nous vous proposons **d'organiser** votre travail en créant un dossier sur votre disque dur. Vous pourrez ainsi stocker et retrouver tous les exercices. Le nom du fichier sera précisé entre parenthèses après chaque énoncé.

Ouvrez l'explorateur Windows (raccourci : touche Windows + E) et placez-vous sur le disque C. Dans la barre de menu, choisissez l'option **Fichier > Nouveau > Dossier**. Nommez votre dossier "coursJavascript".

Quels sont les outils dont vous allez avoir besoin pour ce cours ?

Le JavaScript est un langage qui utilise le même environnement que l'HTML, vous utiliserez donc les mêmes outils :

- le Bloc-notes (pour ouvrir le Bloc-notes : menu Démarrer > Programmes > Accessoires > Bloc-notes)
- Microsoft Internet Explorer (version 5 ou supérieure)

Le **Bloc-notes** (Notepad) sera utilisé pour **éditer** les documents HTML.

Remarque : Tout éditeur de code HTML peut faire l'affaire ! Sous Windows, il y a l'excellent Notepad++ !!

Le navigateur Microsoft **Internet Explorer** sera utilisé pour **visualiser** le résultat.

4 : Afficher du texte

Voici l'énoncé d'un programme simple :

Ecrire un programme qui affiche une fenêtre d'alerte contenant le texte "Bienvenue dans l'univers de la BD !" (**nom du fichier : alerter.htm**).

5 : Ecriture du code

Pour commencer, ouvrez le Bloc-notes et recopiez le code ci-dessous.

Faites bien attention à l'écriture du code, celui-ci doit être en tout point **IDENTIQUE** à celui proposé ci-dessous!

Remarque : Pour copier ce code dans son intégralité, sélectionnez-le et copiez-le dans le presse-papier (clipboard) dont le raccourci clavier est Ctrl + C (ou clic droit > copier). Il ne vous restera plus qu'à le coller dans le Bloc-notes (Ctrl + V ou clic droit > coller)

```
<html>
  <head>
    <title>
      Ecrire un programme qui affiche le texte "Bienvenue dans
l'univers de la BD !"
    </title>
    <script>
      function Main(){
        alert("Bienvenue dans l'univers de la BD !") ;
      }
    </script>
  </head>
  <body onload="Main();" >
  </body>
</html>
```

Enregistrez le document dans le dossier "coursJavascript" sous le nom "alerter.htm".

Remarque : Attention à l'**extension ".htm"** du fichier ! Lorsque vous enregistrez votre document, choisissez "tous les fichiers" dans la liste "type" sinon votre document risque d'être sauvé sous le nom "alerter.htm.txt" et sera pris comme un fichier **texte** et non un **document HTML** !!!

Dans l'Explorateur Windows, il est facile de reconnaître une page HTML d'un simple fichier texte : l'icône du fichier est différente.

6 : L'exécution du programme

Visualisons maintenant le résultat du premier exercice, dans le jargon, on appelle cela exécuter le programme.

Comme pour une simple page HTML, ouvrez le navigateur et allez **dans Fichier > Ouvrir... > Parcourir** et allez rechercher le fichier "alerter.htm".

Remarque : Vous pouvez également double-cliquer dans l'Explorateur Windows sur le fichier "alerter.htm", il s'ouvrira tout seul dans votre navigateur.

SUPER ! Vous venez d'exécuter votre premier programme !

Si tout se passe bien, vous devriez avoir une fenêtre d'alerte vous affichant le texte "Bienvenue dans l'univers de la BD !".

Une fenêtre d'alerte est une simple fenêtre d'avertissement contenant un message. Le message est le **seul** élément paramétrable d'une fenêtre d'alerte.

Si vous n'avez pas de fenêtre d'alerte, c'est qu'il y a une erreur dans votre code, il faut éditer le document dans le Bloc-notes et vérifier le code. Faites très attention aux points suivants :

- Les majuscules / minuscules ;
- Les points-virgules ;
- Les guillemets ;
- L'orthographe des mots-clés.

Une fois le code modifié, enregistrez à nouveau votre travail et recommencez cette étape.

Vous pouvez également visualiser le résultat attendu en ligne :

Voyons le résultat !

Lorsque vous cliquerez sur le bouton ci-dessus, une nouvelle fenêtre s'ouvrira dans laquelle une alerte affichera le texte "Bienvenue dans l'univers de la BD". [Voir le résultat](#)

7 : Quizz

1. Quel outil utilise-t-on pour écrire du code JavaScript ?

- a) Un éditeur de texte
- b) Un navigateur

2. Quel outil utilise-t-on pour "exécuter un programme" écrit en JavaScript ?

- a) Un éditeur de texte
- b) Un navigateur

8 : Ce que l'on a fait

Pour que vous puissiez suivre la démarche du développement de ce code, nous allons revoir ce code pourvus de commentaires.

Note : *Qu'est-ce qu'un commentaire ?* C'est une annotation du développeur qui est complètement ignorée par le programme (ou par le navigateur dans le cas d'un commentaire HTML) et qui est utilisée pour rendre le code plus clair ou pour donner quelques explications sur celui-ci.

Testez l'utilisation des commentaires! Ajoutez-en dans le fichier "alerter.htm", vous verrez que les commentaires n'ont aucune influence sur le programme!

Remarque :

En HTML, les commentaires s'écrivent comme suit :

```
<!-- commentaire HTML -->
```

En JavaScript, les commentaires sur UNE ligne s'écrivent comme suit :

```
// commentaire JavaScript sur une ligne
```

Quelques exemples de commentaires en JavaScript se trouvent dans les compléments.

Continuons pour voir le code en détail.

9 : Explication 1/5 - Le code complet

Voici le code complet et commenté de cette première page.

```
<html>
  <head>
    <title>
```

```

        Ecrire un programme qui affiche le texte "Bienvenue dans
l'univers de la BD !"
    </title>
    <!-- début du script -->
    <script>
        // fonction PRINCIPALE appelée au chargement de la page
        function Main(){
            // ouverture de la fenêtre d'alerte
            alert("Bienvenue dans l'univers de la BD !");
        }
    </script>
</head>
<!-- exécution du script au CHARGEMENT de la page (onload="") -->
<body onload="Main();" >
</body>
</html>

```

Voyons-le plus en détail.

10 : Explication 2/5 - les balises HTML

En surbrillance, lignes 1 à 7 et 13 à 18, ce sont les balises HTML de la page.

Vous retrouvez les balises `<html>`, `<head>`, `<title>` et `<body>` que vous utilisez à chaque création de document HTML.

Sont comptés parmi ces éléments HTML :

- la balise `<script></script>` : c'est la balise HTML qui permet de placer le Javascript au sein du HTML,
- les commentaires HTML : `<!-- COMMENTAIRE -->`

```

<html>
    <head>
        <title>
            Ecrire un programme qui affiche le texte "Bienvenue dans
l'univers de la BD !"
        </title>
        <!-- début du script -->
        <script>
            // fonction PRINCIPALE appelée au chargement de la page
            function Main(){
                // ouverture de la fenêtre d'alerte
                alert("Bienvenue dans l'univers de la BD !");
            }
        </script>
    </head>

```

```
<!--exécution du script au CHARGEMENT de la page (onload="") -->
<body onload="Main();">
</body>
</html>
```

Remarque : en complément, vous pouvez revoir la structure de base d'un document HTML.

11 : Explication 3/5 - le script

La balise `<script>` est une **balise HTML** qui signifie l'intégration d'un script dans le document HTML : **tout** ce qui se trouve entre les balises `<script>` et `</script>` **doit** être écrit en **JavaScript**.

Qu'est-ce qu'un script ?

Un script est une suite d'**instructions** (ordres) interprétées à l'exécution d'un programme. (vous verrez cela plus en détail dans le module suivant).

```
<html>
  <head>
    <title>
      Ecrire un programme qui affiche le texte "Bienvenue dans
l'univers de la BD !"
    </title>
    <!-- début du script -->
    <script>
      // fonction PRINCIPALE appelée au chargement de la page
      function Main(){
        // ouverture de la fenêtre d'alerte
        alert("Bienvenue dans l'univers de la BD !");
      }
    </script>
  </head>
  <!--exécution du script au CHARGEMENT de la page (onload="") -->
  <body onload="Main();">
</body>
</html>
```

12 : Explication 4/5 - la fonction Main();

function Main(){...} est la **fonction PRINCIPALE** du programme. **Tout** ce qui se trouve à l'intérieur des accolades `{ }` fait partie de la fonction Main().

Cette fonction est appelée à un certain moment. Ce moment est, dans ce cas-ci, le CHARGEMENT de la page, ligne 16 : `<body onload="Main();">`

Remarque : "onload" signifie "au chargement".

```
<html>
  <head>
    <title>
      Ecrire un programme qui affiche le texte "Bienvenue dans
l'univers de la BD !"
    </title>
    <!-- début du script -->
    <script>
      // fonction PRINCIPALE appelée au chargement de la page
      function Main(){
        // ouverture de la fenêtre d'alerte
        alert("Bienvenue dans l'univers de la BD !");
      }
    </script>
  </head>
  <!--exécution du script au CHARGEMENT de la page (onload="") -->
  <body onload="Main();">
</body>
</html>
```

13 : Explication 5/5 - l'instruction alert();

`alert("Bienvenue dans l'univers de la BD !");` est la commande qui dit au programme d'ouvrir une fenêtre d'alerte et d'y afficher le texte "Bienvenue dans l'univers de la BD !".

Remarquez que la commande se termine par un ; .

Vous pouvez revoir l'exécution du programme à la page suivante.

```
<html>
  <head>
    <title>
      Ecrire un programme qui affiche le texte "Bienvenue dans
l'univers de la BD !"
    </title>
    <!-- début du script -->
    <script>
      // fonction PRINCIPALE appelée au chargement de la page
      function Main(){
        // ouverture de la fenêtre d'alerte
        alert("Bienvenue dans l'univers de la BD !");
      }
    </script>
  </head>
  <!--exécution du script au CHARGEMENT de la page (onload="") -->
```

```
<body onload="Main();">
</body>
</html>
```

Exécuter le script

Pour revoir le fonctionnement de cette page et du script qui s'y trouve, cliquez sur ce bouton. [Voir la page](#)

14 : Synthèse

Pour écrire un programme en JavaScript, il suffit d'ajouter les balises `<script>`/`</script>` dans un document HTML.

Ces balises `<script>`/`</script>` sont généralement placées dans l'en-tête d'une page, cela permet de regrouper le code et de ne pas trop mélanger l'HTML et le JavaScript.

Le code JavaScript sera inséré à l'intérieur de ces balises.

Comme le code JavaScript se trouve dans un document HTML, on peut l'éditer dans n'importe quel éditeur de document HTML. Le plus simple des programmes est le Bloc-notes.

Pour exécuter le programme, il suffit d'ouvrir le document HTML dans un navigateur.

Le langage JavaScript est sensible à la casse, c'est-à-dire aux **majuscules/minuscules**. La commande `alert("Bonjour")` est exécutée normalement alors que la commande `Alert("Bonjour")` produit une erreur.

15 : Appliquez : afficher une alerte

Appliquez directement cette matière en créant un programme similaire à l'aide du bouton "action" :

"Créer un simple programme qui affiche le texte "Bonjour tout le monde !" dans une fenêtre d'alerte."

Le squelette du "programme" est déjà écrit.

Pour vous aider, récupérez le code du programme `alerter.htm`.

Attention ! Faites bien attention aux :

- fautes de frappe
- majuscules / minuscules

```
<html>|r|n|t<head>|r|n|t|t<title></title>|r|n|t|t*script=|r|n|t|t|t|r|n|t|t=script*|r|n|t</head>|r|n|t<body>|r|n|t|t|r|n|t</body>|r|n</html> <html>|r|n|t<head>|r|n|t|t<title></title>|r|n|t|t*script=|r|n|t|t|t|tfunction Main(){ |r|n|t|t|t|t|talert("Bonjour tout le monde !");|r|n|t|t|t|t|t|t|t|t|t=script*|r|n|t</head>|r|n|t<body onload="Main();">|r|n|t |r|n|t</body>|r|n</html>
```

16 : Afficher du texte dans le document

Voici une variante du premier exercice :

Plutôt que d'afficher le texte "Bienvenue dans l'univers de la BD !" dans une fenêtre d'alerte, affichons-le dans le **document** (nom du fichier : **ecrire.htm**).

Pour ce faire, il suffit de changer la commande `alert("Bienvenue dans l'univers de la BD !")` en `document.write("Bienvenue dans l'univers de la BD !")`

Voici le code modifié :

```
<html>
  <head>
    <title>
      Ecrire un programme qui affiche le texte "Bienvenue dans
l'univers de la BD !"
    </title>
    <script>
      // Intégration d'un script dans le document
      function Main(){
        // fonction PRINCIPALE appelée au chargement de la page
        document.write("Bienvenue dans l'univers de la BD !") ;
        // écriture dans le document
      }
    </script>
  </head>
  <body onload="Main();" >
  </body>
</html>
```

Enregistrez ce code sous "ecrire.htm" et ouvrez le fichier...

Exécuter le script

Plutôt que de voir le texte s'afficher dans une fenêtre d'alerte, il s'affiche dans le document. [Voir la page](#)

17 : Pourquoi Main() ?

Voici encore un commentaire sur le code que nous venons de découvrir.

Nous avons vu que le code était englobé dans la fonction **Main(){ }**.

Cette fonction est utilisée dans un but pratique : elle permet d'exécuter le programme lors de différents **événements**. Ainsi par exemple, si vous remplacez **onload** par **onunload** dans le code, le programme s'exécutera non pas au chargement de la page mais lorsque l'utilisateur quitte celle-ci. Dans les compléments, vous verrez différents exemples de ce type.

D'autre part, au module 2, vous verrez que le programme peut aussi être exécuté lorsque l'utilisateur clique sur un bouton, dans un formulaire.

appeler la fonction `cocotte()` au lieu de `main()` fonctionnerait tout autant. Mais par convention, il est conseillé d'utiliser `main()`. À vous de voir...

18 : Fonction principale

L'utilisation de la fonction `Main()` est également **utile** pour la prise en main d'un langage de programmation car de nombreux langages, comme par exemple le langage Java, **nécessitent** l'utilisation d'une fonction dite "principale".

Dans le complément "une solution encore plus simple", vous verrez qu'on peut cependant travailler sans cette fonction.

19 : Quizz

1. La commande qui permet d'afficher un texte dans une fenêtre d'alerte est...

- a) `Main()`
- b) `alert()`
- c) `document.write()`

2. La commande qui permet d'afficher un texte dans le document est ...

- a) `Main()`
- b) `alert()`
- c) `document.write()`

20 : Synthèse

Un programme écrit en JavaScript est en fait une simple page HTML contenant :

- des balises HTML
- du code JavaScript

On différencie le code HTML et le code JavaScript en plaçant ce deuxième entre les balises HTML `<script>` et `</script>`.

Pour acquérir de bonnes habitudes de programmation, on utilise une fonction "**principale**" que l'on appelle communément "**Main()**". Cette fonction est appelée à un certain moment dans l'affichage de la page (par ex. : `onload` qui est l'événement "au chargement de la page").

21 : Mémo

Préparons-nous

Afin de travailler dans les meilleures conditions possibles, voici ce dont nous aurons besoin pour ce cours:

- Le **bloc notes** (ou Notepad++, disponible ici: <https://notepad-plus-plus.org/fr/>)
- Un **navigateur** (Internet Explorer, Chrome, Firefox, Safari, ...)

N'oubliez pas qu'une fois votre fichier sauvegardé, vous devez modifier son extension en **.htm** à la place de **.txt** (le **.html** est également autorisé).

Lancez votre premier programme

Une fois votre première page créée, vous pouvez avoir envie de la tester. Il vous suffit pour cela de **double-cliquer** sur le fichier html, et votre premier programme s'exécutera.

Si rien ne se passe, n'oubliez pas de **vérifier les points suivants**:

- Les **majuscules / minuscules**, le langage Javascript est en effet sensible à la casse
- Les **points-virgules**
- Les **guillemets**
- L'**orthographe** des mots-clés

Les commentaires

Ceux-ci sont différents, en fonction de la partie de la page dans laquelle on se trouve.

Si vous êtes dans le code HTML, les commentaires s'écrivent comme ceci :

```
<!-- commentaire HTML -->
```

En JavaScript, les commentaires sur UNE ligne s'écrivent comme suit :

```
// commentaire JavaScript sur une ligne
```

Sur plusieurs lignes, il suffit de commencer par **/*** et de finir par ***/** comme dans cet exemple:

```
/*
```

```
Si vous lisez ceci  
votre cours de JavaScript  
avance énormément!
```

```
*/
```

Le code

Vous avez vu tout au long de ce premier chapitre comment le **JavaScript** se marie au langage **HTML**.

Pour rappel, voici le squelette d'une page HTML contenant du **JavaScript** :

```
<html>
  <head>
    <title>
      Titre de la page
    </title>
    <script>
    </script>
  </head>
  <body>
  </body>
</html>
```

La différence avec une "simple" page HTML vient donc de ces balises: `<script></script>`

La fonction Main();

Choisir le nom `Main()` n'est qu'une convention, vous pouvez l'appeler `LaFraise()`, ou comme il vous plaira, tant que vous respectez son nom (**et les majuscules**) quand vous y faites référence dans votre programme.

Cette fonction **englobe** votre code. Elle est utilisée dans un but pratique : elle permet d'exécuter le programme lors de différents **événements**. Si vous remplacez par exemple `onload` par `onunload` dans le code, le programme s'exécutera non pas au **chargement de la page** mais lorsque l'utilisateur **quitte celle-ci**.

Afficher du texte

1. Pour afficher du texte dans une fenêtre d'alerte (attention aux majuscules) : `alert("");` ;
2. Pour afficher du texte dans le document : `document.write("");` ; C'est tout simple, on part du "**document**", et on utilise sa fonction "**write**". Ce qui est inscrit entre guillemets sera affiché dans le document (la page web).

Les événements

Dans la partie HTML, on peut choisir de déclencher une fonction Javascript lors d'un événement. Voici quelques exemples d'événements :

- `<body onload="Main() ;">` // appel de la fonction `Main()` au **chargement** de la page
- `<body onunload="Main() ;">` // appel de la fonction `Main()` au **déchargement** de la page
- `` // appel de la fonction `Main()` lors du **survol de l'image**
- `` // appel de la fonction `Main()` lors du **clic sur l'image**

- `<input type="button" value="quitter" onclick="Main() ;">` // appel de la fonction Main() lors du clic sur le bouton

Bon à savoir

Le JavaScript est **interprété** et non **compilé**. C'est le **navigateur** qui se charge de cette interprétation. De ce fait, il peut y avoir des **différences d'interprétation** entre différents navigateurs.

22. 1 : Compléments

En complément, vous trouverez des informations plus détaillées sur les notions vues, des exemples de programmes exécutés lors de différents événements et un exercice récapitulatif.

22. 2 : Rappel : Structure HTML de base

Certaines balises sont nécessaires à la bonne interprétation d'une page.

Voici le squelette de base d'un document HTML :

```
<html>
  <head>
    <title>
    </title>
  </head>
  <body>
  </body>
</html>
```

(la balise <title> est requise pour toutes les pages HTML)

22. 3 : Exécuter un programme JavaScript

Pour **exécuter** un programme JavaScript, il suffit d'**ouvrir la page HTML** qui contient le code JavaScript dans un navigateur. Ce code est **interprété** en même temps que l'ouverture du fichier par le navigateur, comme l'HTML.

Il y a donc **cohabitation** entre le code **JavaScript** et le code **HTML** dans un même fichier, c'est ce qui fait toute la particularité de ce langage !

Il y a tout de même un petit "HIC" à tout cela : c'est le navigateur qui interprète le langage HTML et les scripts JavaScript. Il existe donc quelques petites différences d'interprétation entre les différents navigateurs.

Dans ce cours, nous nous limitons à montrer des exercices utilisant des commandes "compatibles". Il y a toutefois quelques cas particuliers dans les compléments de modules.

22. 4 : Une solution encore plus simple !

Reprenons l'énoncé du premier exercice : "Ecrire un programme qui affiche une fenêtre d'alerte contenant le texte "Bienvenue dans l'univers de la BD !"". "

Pour le même résultat, il existe un code plus simple :

```

<html>
  <head>
    <title>
      Ecrire un programme qui affiche le texte "Bienvenue dans
l'univers de la BD !"
    </title>
    <!--début du script JavaScript -->
    <script>
      // Intégration d'un script dans le document
      alert("Bienvenue dans l'univers de la BD !") ;
      // ouverture de la fenêtre d'alerte
    </script>
  <!--fin du script JavaScript -->
</head>
<body>
</body>
</html>

```

On a retiré la fonction "Main()" et le programme s'exécute correctement. La commande JavaScript "alert()" est exécutée **au même moment** que son **interprétation**. En clair, cela signifie que le navigateur lit les instructions du document HTML de haut en bas et va donc exécuter l'instruction "alert()" dès le début de notre exemple puisque cette instruction est située avant le <body>.

Cette fonction Main() n'est en fait pas **obligatoire** mais simplement **utile** car elle permet d'associer une **action** (par ex. alert(...)) à un **événement** (par ex. onload="Main() ;").

Pour une meilleure approche de programmation, nous continuerons à travailler avec cette fonction Main(), cela vous permettra de passer plus facilement à d'autres langages de programmation comme, par exemple, Java.

Notez que le nom n'a pas d'importance ici : vous pourriez remplacer Main() par Zozo() ou Pomme() dans le code de la page sans modifier le résultat.

22. 5 : Les événements

Les **événements** ont été créés pour ajouter de **l'interactivité** entre un visiteur et une page Web.

Voici quelques exemples d'événements :

- <body onload="Main() ;"> // appel de la fonction Main() au **chargement de la page**
- <body onunload="Main() ;"> // appel de la fonction Main() au **déchargement de la page**
- // appel de la fonction Main() lors du **survol de l'image**
- // appel de la fonction Main() lors du **clic sur l'image**
- <input type="button" value="quitter" onclick="Main() ;"> // appel de la fonction Main() lors du **clic sur le bouton**

Ces événements se retrouvent toujours dans une **balise HTML**. A ces événements, on **associe** l'appel d'une fonction JavaScript, par exemple la fonction **Main()** ; .

Voici quelques exemples (code + exécution) de programmes appelant la fonction **Main()** lors de différents événements.

1. Afficher un message d'au revoir

Un programme affiche "Merci de votre visite. A bientôt !" lorsque l'utilisateur quitte la page.

L'événement utilisé est **onunload**, cet événement peut uniquement être placé dans la balise **<body>**, il signifie "au déchargement de la page".

```
<html>
  <head>
    <title>
      Ecrire un programme qui affiche le texte "Bienvenue dans
l'univers de la BD !"
    </title>
    <!-- début du script -->
    <script>
      // fonction PRINCIPALE appelée
      function Main(){
        // ouverture de la fenêtre d'alerte
        alert("Bienvenue dans l'univers de la BD !");
      }
    </script>
  </head>
  <!--exécution du script au DECHARGEMENT de la page (onunload="") -->
  <body onunload="Main();">
</body>
</html>
```

Au lieu d'appeler la fonction **Main()** au **chargement de la page (onload)**, on l'appelle au **déchargement de la page (onunload)**. Vous ne verrez l'alerte que lorsque vous quitterez la page.

[Exécuter le code](#)

2. Afficher un message lors du survol d'une image

Un programme affiche "Voici le dernier tome de Spirou et Fantasio !" lorsque l'on survole l'image avec le curseur de la souris.

L'événement utilisé est **onmouseover**. Cet événement se déclenche lorsque l'on survole l'élément dans lequel il a été placé.

```
<html>
  <head>
    <title>
```

```
    Un programme qui alerte "Voici le dernier tome de Spirou et
Fantasio !" lorsque l'on survole l'image
</title>
<script>
    // Intégration d'un script dans le document
    function Main(){
        alert("Voici le dernier tome de Spirou et Fantasio !") ;
        // ouverture de la fenêtre d'alerte
    }
</script>
</head>
<body>
    
</body>
</html>
```

La fonction **Main()** est appelée lors du **survol de l'image** (**onmouseover**).

[Exécuter le code](#)

22. 6 : Les commentaires JavaScript

Il existe deux moyens d'écrire des commentaires en JavaScript :

Commentaires sur une ligne :

```
// ceci est un commentaire sur une ligne
```

Commentaires sur plusieurs lignes :

```
/*
Ceci
est
un
commentaire
sur
plusieurs
lignes
*/
```


En résumé :

// ... pour une ligne de commentaires

/* ... */ pour plusieurs lignes de commentaires

1. Lequel de ces commentaires est incorrect ?

- a) // ce programme alerte l'utilisateur d'un message de bienvenue // cette alerte s'affiche automatiquement au chargement de la page
- b) /* ce programme alerte l'utilisateur d'un message de bienvenue cette alerte s'affiche automatiquement au chargement de la page */
- c) // ce programme alerte l'utilisateur d'un message de bienvenue cette alerte s'affiche automatiquement au chargement de la page //

22. 7 : Ecrire dans le document

La commande `document.write()` est bien pratique pour afficher du contenu dans le document. Ce contenu peut contenir des **balises HTML** :

```
document.write("<b>Bienvenue dans le monde de la BD !</b>") ;
```

Attention ! N'essayez pas d'écrire au format HTML dans une fenêtre d'alerte. Une fenêtre d'alerte n'affiche que du texte et pas d'HTML.

Module 2 - Communiquer avec la machine

1 : Objectifs

Dans ce module, vous allez apprendre à :

- **afficher** du texte dans un champ texte d'un formulaire.
- **recupérer** de l'information à partir d'un champ texte d'un formulaire.

C'est grâce aux formulaires qu'un utilisateur peut communiquer avec un programme.

C'est cette **communication** entre un programme et l'utilisateur qui rend un outil **interactif** et qui permet une gestion personnalisée des actions menées.

2 : Communiquer à l'aide des formulaires

Un formulaire permet une **communication** entre l'utilisateur et le site (ou les pages du site).

Les données sont envoyées au serveur, généralement dans une base de données via un programme ou une application Web.

Le JavaScript est utilisé, entre autres, pour éviter l'envoi d'un formulaire vide au serveur en **validant** le formulaire avant l'envoi.

Un formulaire peut aussi fonctionner **sans** base de données, simplement pour rendre une page plus interactive. Nous en verrons quelques exemples dans ce cours.

3 : Rappel sur les formulaires

Un formulaire peut contenir différents types de champs : champs texte, cases à cocher, boutons radio, boutons...

Dans le cadre de ce cours, nous travaillerons uniquement avec des champs texte et des boutons.

Voici un exemple de formulaire :

Le code est le suivant :

```
<form name="info" action="mailto:spirou@franquin.be">
  Votre nom : <input type="text" name="nom">
  Votre prénom : <input type="text" name="prenom">
  <input type="button" value="Envoyer" onclick=" Main( );">
</form>
```

4 : Les balises des formulaires

Pour insérer un formulaire dans une page Web, on utilise la balise suivante :

```
<form name="info" action="mailto:spirou@franquin.be">
<!-- Contenu du formulaire -->
</form>
```

Dans ce cas-ci, le formulaire est envoyé à une adresse email, spirou@franquin.be.

Dans ce cours, nous ne nous intéresserons qu'à l'interaction entre une page Web et l'utilisateur, nous ne préciserons donc **pas** l'attribut **action**.

Par contre, l'attribut "**name**" est très important : en JavaScript, il faut **nommer** un formulaire et les champs qui y sont inclus pour pouvoir les atteindre. Dans ce cas-ci, le nom du formulaire est **info**.

La balise pour un champ texte est : `<input type="text" name="nom">`.

Dans ce cas-ci, le nom du champ texte est **nom**.

La balise pour un bouton est:

```
<input type="button" value="Envoyer" onclick="Main();">
```

Attention ! Les noms du champ texte et du formulaire sont très importants !

Le nom d'un élément HTML (un formulaire ou un champ) doit **toujours**:

- être en un seul mot,
- être sans accent,
- commencer par une lettre ou " _ ".

5 : Un programme qui écrit

Vous allez voir comment un programme peut **transmettre** des messages à l'utilisateur à l'aide des formulaires...

Reprenons l'énoncé de "alerter.htm" avec une petite modification :

"Créer un programme qui affiche le texte "Bienvenue dans l'univers de la BD !" dans un champ texte d'un formulaire" (**nom du fichier : écrireForm.htm**).

6 : Le code du programme

Voici le code du programme :

```

<html>
  <head>
    <title>
      Créer un programme qui affiche le texte "Bienvenue dans
l'univers de la BD !" dans un champ texte d'un formulaire
    </title>
    <script>
      function Main(){
        document.accueil.message.value = "Bienvenue dans l'univers
de la BD !" ;
      }
    </script>
  </head>
  <body onload="Main();" >
    <form name="accueil">
      <input type="Text" name="message" size="40">
    </form>
  </body>
</html>

```

Recopiez ce code dans le Bloc-notes, enregistrez-le sous "ecrireForm.htm" et exécutez-le.

Ce programme écrit un message dans un champ texte se trouvant dans un formulaire.

Ce code mérite quelques explications supplémentaires :

1. On a créé un formulaire portant le nom "accueil"
2. Dans ce formulaire, on a créé un champ texte portant le nom "message"
3. Dans le programme, on **atteint la valeur du champ "message"** se trouvant dans le **formulaire "accueil"** par la commande suivante : **document.accueil.message.value**. Cette valeur est égale à "Bienvenue dans l'univers de la BD !".

document.accueil.message.value : on atteint le document

document.accueil.message.value : on atteint le formulaire nommé "accueil"

document.accueil.message.value : on atteint le champ texte nommé "message"

document.accueil.message.value : on atteint la valeur du champ texte

La propriété **value** permet d'assigner une valeur dans le champ texte (vous allez voir plus loin que, grâce à la propriété **value**, on peut aussi récupérer le texte se trouvant dans un champ).

Exécuter le programme

Cliquez sur le bouton pour afficher la page et voir son fonctionnement. [Voir la page](#)

7 : Appliquez : dire bonjour dans un champ texte

Appliquez directement la matière vue en effectuant un exercice similaire à l'aide du bouton "action":

"Ecrire un programme qui affiche "Bonjour tout le monde !" dans un champ texte nommé "message" se trouvant dans un formulaire nommé "accueil". Ce texte doit s'afficher lors d'un clic sur un bouton (<input type="button" value="Dis Bonjour !" onclick="Main() ;">)."

- 1 Créer le formulaire contenant un champ texte et un bouton,
- 2 Ajouter le contenu de la fonction Main,
- 3 Ajouter l'événement qui appelle la fonction Main.

- N'oubliez pas de bien nommer le formulaire et le champ texte,
- Pour que le texte s'affiche entièrement, n'hésitez pas à agrandir la taille du champ texte :
`<input type="Text" name="message" size="40">`,
- Faites attention aux majuscules/minuscules,
- N'oubliez pas de guillemet, d'accolade, de parenthèse, ...

8 : Conclusion

- La méthode `"alert(...)"` : cette méthode affiche du texte dans une fenêtre d'alerte
- La méthode `"document.write(...)"` : cette méthode écrit du texte dans le document
- La propriété `"document.formulaire.champ.value = "..."` : on écrit dans un champ texte d'un formulaire (`formulaire` étant le nom du formulaire et `champ` étant le nom du champ texte).

9 : Quizz

1. Lequel de ces formulaires HTML porte un nom correct ?

- a) <form name="l'accueil">
- b) <form name="accueil">
- c) <form name="accueil site">

2. Laquelle de ces commandes permet de récupérer la valeur du champ "montant" du formulaire "calcul" ?

- a) document.calcul.montant.valeur
- b) document.montant.value
- c) document.calcul.montant.value

10 : Un programme qui lit

Voyons maintenant un exemple de communication de l'utilisateur vers le programme.

L'utilisateur va écrire dans un formulaire. Afin de nous assurer que le programme a bien reçu l'information, nous allons lui demander de répéter l'information reçue.

(pour les besoins de cet exemple, nous utiliserons la notion de variable, qui sera expliquée au module 4)

11 : Exemple de programme qui lit

"Ecrire un programme qui demande à l'utilisateur de proposer une injure du capitaine Haddock dans un champ texte. Lorsque l'utilisateur clique sur le bouton "Répéter l'injure", la proposition s'affiche dans un autre champ texte du formulaire" (nom du fichier : demanderForm.htm)

```
<html>
  <head>
    <title>
      Programme qui lit
    </title>
    <script>
      var prop;
      // on déclare la variable proposition
      function Main(){
        prop = document.echo.proposition.value ;
        // on enregistre la proposition dans la variable
proposition

        document.echo.injure.value = prop;
        // on écrit la valeur proposée dans le champ texte " injure
"
      }
    </script>
  </head>
</html>
```

```

        </script>
    </head>
    <body>
        <form name="echo">
            <input type="text" name="proposition">
                <input type="button" value="Répéter la proposition"
onclick="Main()">
                Le capitaine Haddock dit :
                <input type="text" name="injure">
            </form>
        </body>
    </html>

```

Remarque : Dans ce programme, **prop** est le nom d'une **variable**. Cette variable va contenir une **valeur**. Cette notion de variable est expliquée plus en détail dans le module 4.

Enregistrez ce programme sous "demanderForm.htm" et exécutez-le.

Le résultat :

Vous obtenez un formulaire.

Vous insérez votre proposition dans le champ texte et cliquez sur le bouton "Répéter l'injure".
Automatiquement, votre proposition s'affiche dans le deuxième champ du formulaire.

Ce programme lit un message inséré par l'utilisateur dans un champ de formulaire.

Exécuter la page

Lorsque l'utilisateur clique sur le bouton "Répéter la proposition", la proposition s'affiche dans le document. Vous pouvez voir l'exécution de ce programme en ligne. [Voir la page](#)

12 : Appliquez : demander le prénom

Appliquez directement la matière vue en effectuant un exercice :

"Voici un programme qui demande votre prénom via un formulaire et qui affiche le texte "Bonjour [PRENOM]" dans une fenêtre d'alerte. Modifiez ce programme pour que ce texte s'affiche dans un 2ème champ du formulaire placé après le bouton (Code du deuxième champ :
<input type="text" name="echo" size="40">").

```

<html>
\\r\\n\\t<head>
\\r\\n\\t\\t<title>Afficher le texte "Bonjour [PRENOM]" dans un champ texte du formulaire</title>
\\r\\n\\t\\t<script=
\\r\\n\\t\\t\\tvar proposition;
\\r\\n\\t\\t\\tfunction Main(){
\\r\\n\\t\\t\\t\\tproposition = document.questionnaire.prenom.value ;
\\r\\n\\t\\t\\t\\talert("Bonjour " + proposition) ;
\\r\\n\\t\\t\\t}
\\r\\n\\t\\t</script>
\\r\\n\\t</head>

```

```

\\r\\n\\t<body>
\\r\\n\\t\\t<h1>Quel est votre prénom? </h1>
\\r\\n\\t\\t<form name="questionnaire">
\\r\\n\\t\\t\\t<input type="Text" name="prenom">
\\r\\n\\t\\t\\t<input type="Button" value="Envoyer" onclick="Main();">
\\r\\n\\t\\t</form>
\\r\\n\\t</body>
\\r\\n\\t</html> <html>
\\r\\n\\t<head>
\\r\\n\\t\\t<title>Afficher le texte "Bonjour [PRENOM]" dans un champ texte du formulaire</title>
\\r\\n\\t\\t<script=
\\r\\n\\t\\t\\tvar proposition;
\\r\\n\\t\\t\\tfunction Main(){
\\r\\n\\t\\t\\t\\tproposition = document.questionnaire.prenom.value ;
\\r\\n\\t\\t\\t\\tdocument.questionnaire.echo.value="Bonjour "+proposition;
\\r\\n\\t\\t\\t\\t}
\\r\\n\\t\\t\\t=script*
\\r\\n\\t</head>
\\r\\n\\t<body>
\\r\\n\\t\\t<h1>Quel est votre prénom? </h1>
\\r\\n\\t\\t<form name="questionnaire">
\\r\\n\\t\\t\\t<input type="Text" name="prenom">
\\r\\n\\t\\t\\t<input type="Button" value="Envoyer" onclick="Main();">
\\r\\n\\t\\t\\t<br>
\\r\\n\\t\\t\\t<input type="text" name="echo" size="40">
\\r\\n\\t\\t</form>
\\r\\n\\t</body>
\\r\\n\\t</html>

```

13 : Conclusion

Communication utilisateur => programme :

Le programme récupère de l'information de l'utilisateur en lisant des messages.

En JavaScript, on peut récupérer du texte à l'aide de l'instruction

"document.formulaire.champ.value" qui permet la récupération de la valeur insérée dans le champ nommé "champ" du formulaire nommé "formulaire".

Il existe une autre méthode permettant de récupérer de l'information provenant de l'utilisateur: la méthode **"prompt"**. Cette méthode est expliquée dans les compléments de ce module.

14 : Quizz

1. Laquelle de ces commandes récupère le contenu du champ "montant" du formulaire "calcul" dans la variable "prix" ?

- a) prix = document.calcul.montant.value ;
- b) prix = document.montant.calcul.value ;
- c) document.calcul.montant.value = prix ;

15 : Synthèse

Informar l'utilisateur et récupérer de l'information provenant de l'utilisateur rend votre programme **interactif**.

Le dialogue **programme/utilisateur** permet la transmission à l'utilisateur d'information comme des messages d'avertissement, des messages d'erreur, ... C'est cette interactivité qui rendra votre programme attrayant, abordable et accessible à tous.

Le dialogue **utilisateur/programme** permet la récupération d'information provenant de l'utilisateur. En JavaScript, on utilise cette communication pour valider les données d'un formulaire, récupérer le nom d'une personne afin de personnaliser un message, afficher ou traiter des données fournies par l'utilisateur comme des calculs (âge, TVA, année bissextile, ...), des petits jeux (serpent, puissance 4, démineur, ...), ...

16 : Exercice récapitulatif

Cet exercice vous permettra de mettre en pratique les notions vues au cours de ce module :

Ecrire un programme qui demande un mot de passe via un formulaire, ce mot de passe est récupéré lors du clic sur le bouton "connexion". Le programme affiche ensuite une alerte pour remercier l'utilisateur (**nom du fichier : login.htm**).

Etapes du développement :

1. Créer le formulaire contenant

un titre : `<h3>Ce site est sécurisé !</h3>`

un champ texte : `<input type="password" name="motdepasse">`

Attention ! Au lieu de mettre `<input type="text">`, il vaut mieux utiliser la balise `<input type="password">` afin que le mot de passe soit masqué lors de l'encodage.

un bouton : `<input type="button" value="connexion">`

2. Dans la fonction Main() :

Récupérer le mot de passe, ce mot de passe doit être enregistré dans une variable.

Afficher un message de remerciement dans une alerte.

3. Appeler la fonction Main() lors du clic sur le bouton

Conseil : Reportez-vous aux exemples de ce module !

Remarque : Cet exercice n'est bien sûr pas complet, il faudrait ajouter une vérification du mot de passe. Pour un maximum de sécurité, cette vérification s'effectue côté serveur.

Consulter la solution

Si votre programme ne tourne pas correctement, vous pouvez voir la page attendue. Le programme affiche une **alerte** concernant la sécurisation de la page. Le programme **demande à l'utilisateur de fournir un mot de passe**. Pour voir le code, affichez la source de la page (Clic droit > **Code source** ou **Afficher la source**). [Voir la page](#)

17 : Mémo

Les formulaires

Un **formulaire** permet d'établir une **communication** entre le site et le visiteur. Le JavaScript permet entre autres de vérifier que **le formulaire n'est pas vide** au moment de l'envoi des données. Les formulaires contiennent plusieurs **types de champs**: du texte, des boutons, des cases à cocher, ... Les balises d'un formulaire sont `<form></form>`

Voici un formulaire avec quelques champs :

```
<form name="quizz" >
  <input type="text" name="q1">
    <input type="button" value="Envoyer" onclick="Main()">
</form>
```

N'oubliez pas de lui donner un nom (ici "quizz") pour pouvoir exploiter son contenu plus tard. Ceci est également valable pour les champs insérés.

Ces noms doivent **toujours** :

- être en un seul mot,
- être sans accent,
- commencer par une lettre ou " _ ".

Pour écrire une information

Voici les trois notions vues jusqu'ici pour **afficher quelque chose à l'écran** :

On peut afficher du contenu dans une **pop-up** qui va s'ouvrir dans le navigateur.

```
alert("information");
```

On peut également afficher ce contenu **dans la page**.

```
document.write("information");
```

Et pour finir, on peut afficher une **valeur dans un formulaire**.

C'est assez simple, il suffit d'écrire dans le document (la page web), qui contient un formulaire (ici appelé "formul1"), lui-même contenant un champ texte "message", auquel nous allons assigner une valeur :

```
document.formul1.message.value = "information";
```

Pour lire une information

C'est simplement le contraire, on prend la **variable** "information" et on y glisse le contenu de la valeur qui est dans le champ "message" du formulaire "formul1", lui-même stocké sur ce **document** (la page web) :

```
information = document.formul1.message.value;
```

(information est une **variable**. La notion de variable sera expliquée dans le module 4.)

Il existe une autre méthode permettant de récupérer de l'information provenant de l'utilisateur: la méthode "prompt".

On s'en sert de la façon suivante :

```
proposition = prompt("Quel est votre fruit préféré? :", "Ecrivez votre  
réponse ici");  
document.write(proposition);
```

Les actions et événements

Dans un programme contenant des **formulaires**, il y a souvent :

- un **événement** : l'utilisateur **clique** sur un bouton (par exemple: "paiement")
- une **action** : par exemple, **alerter** l'utilisateur avec le texte : "Cette page est sécurisée !"

Cette action doit être exécutée lorsque l'événement se produit.

Voici le code adéquat :

```
<form>
  <input type="Button" value="paiement" onclick="Main();" >
</form>
```

Ainsi que quelques exemples d'événements

- **onclick** : lorsque l'on clique (sur un bouton, par exemple)
- **onchange** : lorsque l'on effectue un changement (dans une liste déroulante, par exemple)
- **onfocus** : lorsque l'on "active" un élément (clic dans un champ texte, par exemple)
- **onblur** : lorsque l'on "désactive" un élément (utilisation de la touche "tabulation" pour sortir d'un champ texte, par exemple)
- **onsubmit** : lorsque l'on soumet le formulaire (se place dans la balise <form>)

Les différents langages rencontrés

Toutes ces informations semblent compréhensibles car écrites dans un langage "**humain**", mais ce n'est qu'une traduction du **binaire**, composé de **0** et de **1**, vers quelque chose de plus lisible pour un être humain normalement constitué.

On appelle cela un **langage de programmation**, à l'opposé du **langage machine**, **binaire**. Le HTML, quant à lui, **n'est pas un langage de programmation** car il ne crée pas des programmes, il décrit la façon dont une page doit s'afficher, c'est un **langage de description**.

Un **langage de programmation** est composé de **mots-clés** et d'une **structure logique**. Actuellement, les **langages de programmation** ont atteint une certaine maturité : on y retrouve toujours les mêmes notions de base : **variables**, **structures itératives**, **structures conditionnelles**, **fonctions**, ...

Le DHTML

Le JavaScript permet de rendre une page web **interactive**. Cette technologie qui permet la **communication** entre le langage **JavaScript** et le langage **HTML** s'appelle **DOM** pour **Document Object Model (Modèle Objet du document)**.

L'ensemble des technologies rendant une page plus dynamique s'appelle le **DHTML (dynamic HTML)**. Ce n'est pas un langage, mais plutôt le résultat de la collaboration entre 4 technologies : **HTML**, **CSS (les feuilles de style)**, **DOM** et **JavaScript**.

18. 1 : Compléments

Dans les compléments, nous allons voir plus en détails ce qu'est un programme. Nous découvrirons aussi un autre moyen de communication utilisateur-programme : la fenêtre **prompt**. Enfin, nous verrons des exemples intéressants d'événements dans des formulaires.

18. 2 : Comment passer de l'énoncé au programme ?

Maintenant que vous commencez à programmer, vous vous posez peut-être une des questions suivantes :

- Par quoi commencer ?
- Comment réaliser ceci ?
- Comment récupérer l'information ?

Je vous propose de prendre un énoncé et de l'analyser pour déterminer les étapes du développement du programme...

Voici l'énoncé d'un programme à réaliser :

"Ecrire un programme qui alerte l'utilisateur que la page est sécurisée lorsque l'utilisateur clique sur le bouton "Païement" ". (nom du fichier : **securite.htm**)

Mettez-vous à la place de **l'utilisateur** du programme :

En tant qu'utilisateur, vous devriez voir s'afficher une fenêtre d'alerte contenant le texte suivant :

" Cette page est sécurisée ! ". Cette alerte devrait s'afficher au moment où l'utilisateur **clique** sur le bouton "Païement" se trouvant dans la page.

Dans ce programme, il y a :

- **un événement** : l'utilisateur **clique** sur le bouton "païement"
- **une action** : **alerter** l'utilisateur avec le texte : "Cette page est sécurisée !"

Cette **action** doit être exécutée lorsque l'**événement** se produit.

1. L'**événement** qui déclenche l'action est le "click" du bouton "païement"

```
<input type="Button" value="païement" onclick="Main();">
```

2. L'**action** du programme est :

```
function Main(){  
alert("Cette page est sécurisée !") ;  
}
```

Assemblons ces morceaux de code pour former le programme.

Voici le code du programme répondant aux spécifications de l'énoncé :

```
<html>  
  <head>  
    <title>  
      Ecrire un programme qui alerte l'utilisateur que la page est  
sécurisée lorsque l'utilisateur clique sur le bouton  
    </title>  
    <script>  
      function Main(){
```

```

        alert("Cette page est sécurisée !");
    }
</script>
</head>
<body>
    <form>
        <input type="Button" value="paiement" onclick="Main();" >
    </form>
</body>
</html>

```

Remarque : Ce code est similaire au programme "alerter.htm" réalisé au module 1. La seule différence est que l'action est exécutée à l'événement **onclick d'un bouton** et non à l'événement **onload de la page**.

Enregistrez ce code sous "securite.htm". Vous pouvez exécuter votre programme...

Résultat attendu

Le programme alerte que la page est sécurisée lorsque l'on clique sur le bouton "paiement". [Voir la page](#)

18. 3 : Ce que l'on a fait

Techniquement, vous venez de **traduire** une spécification exprimée en **langue française** en une suite d'**instructions** compréhensibles par une machine.

Pour vous faciliter la tâche de programmation, il faut structurer les idées de manière à pouvoir les représenter sous forme schématique.

La représentation logique d'un programme s'appelle l'**algorithmique** : un algorithme est une suite d'opérations visant à la résolution d'un problème.

18. 4 : Langage machine et langage de programmation

Il est certain qu'une machine (à moins qu'elle ne soit programmée pour ça) ne peut pas **comprendre** un ordre écrit en français. La langue française est composée d'une grammaire et d'une conjugaison trop complexes pour une machine.

Un ordinateur est une machine somme toute très basique puisqu'elle ne comprend que des **suites de 0 et de 1** ! La plus grande difficulté est de "traduire" ce que l'on veut faire en une suite de 0 et de 1 compréhensibles par la machine...pas si simple !

Afin d'éviter de programmer en une suite de 0 et de 1, on a créé des **langages intermédiaires** qui sont

- compréhensibles à l'être humain de par ses mot-clés (vocabulaire) en anglais et sa structure logique (grammaire)
- compréhensibles par la machine car chaque instruction correspond à une suite de 0 et de 1.

Ces langages intermédiaires sont appelés " langages de programmation ".

18. 5 : Conclusion

La traduction entre le "langage humain" et le "langage machine" s'effectue via un **langage de programmation**.

Un langage de programmation est un langage qui, comme son nom l'indique, permet de créer des programmes.

Remarque : Le langage HTML **n'est pas un langage de programmation** car il ne crée pas des programmes, il décrit la façon dont une page doit s'afficher, **c'est un langage de description !**

Un langage de programmation est composé de **mots-clés** et d'une **structure logique**.

18. 6 : Quizz

1. Un langage de programmation est

- a) une suite de 0 et de 1 compréhensibles par la machine
- b) est composé d'instructions qui correspondent à une suite de 0 et de 1 compréhensibles par la machine
- c) est composé d'une grammaire et d'une conjugaison compréhensibles par la machine

18. 7 : Historique des langages de programmation

Le premier langage de programmation "officiel" est le **FORTRAN**, apparu en 1958. Depuis, beaucoup d'autres langages ont été créés, chacun différent selon l'environnement d'utilisation. Certains n'ont pas eu de succès et ont disparu ; d'autres, par contre, sont toujours utilisés de nos jours.

Voici quelques langages de programmation assez connus :

- COBOL
- BASIC
- Pascal
- Visual Basic
- C,C++
- Perl
- Java

Actuellement, les langages de programmation ont atteint une certaine **maturité** : on y retrouve toujours les mêmes **notions de base** : variables, structures itératives, structures conditionnelles, fonctions, ...

Pour en savoir plus sur l'histoire des langages de programmation :

<http://www.commentcamarche.net/langages/langages.php3>

18. 8 : Le JavaScript, un langage de programmation

Le JavaScript a été conçu par **Netscape** en collaboration avec **Sun** dans le but de créer un langage simple, léger, s'exécutant côté client (exécuté par le navigateur sur votre propre PC), permettant **l'interactivité** entre une page Web et un utilisateur.

Le JavaScript est également utilisé par des technologies côté serveur comme ASP (Active Server Pages).

Ce langage est devenu "standard" en 1998 sous le nom **d'EcmaScript** (ECMA262).

Pour en savoir plus sur les spécifications du langage :

<http://www.ecma-international.org/>

18. 9 : Quizz

1. EcmaScript...

- a) Est un langage tout à fait différent de JavaScript
- b) Est le nom repris pour la standardisation du JavaScript
- c) N'existe pas

2. On utilise le JavaScript

- a) Principalement dans les technologies du Web
- b) Dans divers domaines de l'informatique : Applications Windows, Applications pour PDA, ...

18. 10 : JavaScript et l'interactivité

La puissance de JavaScript vient de sa capacité à **communiquer avec les éléments d'une page Web** et à les modifier (changer le contenu, changer de taille, changer de couleur, ...). Cette technologie qui permet la **communication** entre le langage JavaScript et le langage HTML s'appelle **DOM** pour Document Object Model (Modèle Objet du document).

L'ensemble des technologies rendant une page **plus dynamique** s'appelle le **DHTML** (dynamic HTML).

Le DHTML n'est donc pas un langage, il est le résultat de la collaboration entre **4 technologies** : HTML, CSS, DOM et JavaScript.

Pour en savoir plus sur le DHTML :

<http://www.commentcamarche.net/contents/242-introduction-au-dynamic-html-dhtml>

18. 11 : Quizz

1. La technologie servant à rendre les pages Web plus dynamiques s'appelle

- a) Le JavaScript
- b) Le DHTML
- c) Le DOM

2. La technologie permettant l'accès aux éléments (objets) du document s'appelle

- a) Le JavaScript
- b) Le DHTML
- c) Le DOM

18. 12 : La fenêtre "prompt"

Vous avez vu comment **afficher** du texte dans le document, dans une fenêtre d'alerte et dans un champ du formulaire.

Vous avez également vu comment **recupérer** de l'information à partir d'un champ texte.

Voici un autre moyen qui va vous permettre de récupérer de l'information : **La fenêtre "prompt"**.

Voici l'énoncé d'un programme :

" Ecrire un programme qui demande à l'utilisateur de proposer une injure du capitaine Haddock dans une fenêtre prompt. Lorsque l'utilisateur clique sur le bouton "Répéter l'injure", la proposition s'affiche dans le document" (**nom du fichier : demander.htm**)

```
<html>
  <head>
    <title>
      Ecrire un programme qui demande à l'utilisateur de proposer une
      injure du capitaine Haddock dans une fenêtre prompt. Lorsque l'utilisateur
      clique sur le bouton "Répéter l'injure", la proposition s'affiche dans le
      document
    </title>
    <script>
      var proposition;
      //déclaration de la variable proposition
      function Main(){
        proposition = prompt("Proposez une injure du capitaine
Haddock :", "Ecrivez votre proposition ici") ;
        // stockage de la valeur proposée dans la variable
proposition

        document.write("Le capitaine Haddock dit : ") ;
        document.write(proposition) ;
        // on écrit la valeur de la proposition dans le document
      }
    </script>
  </head>
  <body onload="Main();">
```

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

Enregistrez ce programme sous "demander.htm" et exécutez-le.

Remarque : Dans ce programme, **proposition** est le nom d'une **variable**. Cette variable va contenir une **valeur**. Cette notion de variable est expliquée plus en détail dans le module 4 mais vous l'utilisez déjà pour les besoins de l'exercice.

Résultat attendu

Une fenêtre s'ouvre et vous demande de proposer une injure du capitaine Haddock. Cette fenêtre contient un champ texte permettant d'insérer du contenu. Lorsque votre proposition est insérée, vous cliquez sur OK. Automatiquement, votre proposition s'affiche dans le document. [Voir la page](#)

18. 13 : La fenêtre "prompt" (suite)

Ce programme **lit** un message inséré par l'utilisateur dans une fenêtre prompt

La commande " prompt " a deux paramètres :

prompt("Quel est votre âge ?", "insérez votre âge ici")

1. la question ou l'introduction (**Quel est votre âge ?**)
2. la phrase d'aide se trouvant dans le champ texte de la fenêtre (**insérez votre âge ici**)

La commande prompt la plus simple serait **prompt("", "")** mais n'aurait **aucun** intérêt.

Par contre, la phrase d'aide est optionnelle, comme ceci : **prompt("Quel est votre âge ?", "")**

18. 14 : Quizz

1. Quelle commande ne récupère pas l'âge de l'utilisateur ?

- a) `age = prompt("Quel est votre âge ?", "");`
- b) `age = prompt("Quel est votre âge ?", "insérez votre âge ici");`
- c) `age = alert("Quel est votre âge ?");`

2. Trouvez la commande erronée :

- a) `age = prompt("Quel est votre âge ?", "insérez votre âge ici", "");`
- b) `age = prompt("Quel est votre âge ?", " insérez votre âge ici ");`
- c) `age = prompt("Quel est votre âge ?", "");`

18. 15 : Les événements d'un formulaire

Nous avons vu dans ce module comment exécuter un programme en **cliquant sur un bouton**. En fait, il est possible d'exécuter un programme lors de différents événements.

Voici les principaux événements que l'on retrouve dans les éléments d'un formulaire :

- **onclick** : lorsque l'on clique (sur un bouton, par exemple)
- **onchange** : lorsque l'on effectue un changement (dans une liste déroulante, par exemple)
- **onfocus** : lorsque l'on "active" un élément (clic dans un champ texte, par exemple)
- **onblur** : lorsque l'on "désactive" un élément (utilisation de la touche "tabulation" pour sortir d'un champ texte, par exemple)
- **onsubmit** : lorsque l'on soumet le formulaire (se place dans la balise <form>)

Voici quelques exemples en ligne utilisant ces différents événements :

L'événement "onchange"

Ce formulaire présente une liste déroulante. Lorsque vous sélectionnez un élément de la liste, une alerte vous répète votre choix.

Remarque : Pour voir le code de ce programme, il vous suffit de réaliser un clic droit sur la page et de choisir l'option "Afficher la source". [Voir la page](#)

L'événement "onfocus"

Ce formulaire présente un champ de type texte contenant une valeur par défaut. Lorsque l'on clique dans ce champ, la valeur par défaut disparaît, évitant à l'utilisateur de l'effacer ! [Voir la page](#)

L'événement "onblur"

Ce formulaire présente un champ mémo. Lorsque vous y insérez du contenu et que vous sortez du champ, vous obtenez un petit message d'alerte. [Voir la page](#)

L'événement "onsubmit"

Ce formulaire présente un champ et un bouton d'envoi (submit). Lorsque vous soumettez le formulaire, le programme vérifie la valeur insérée dans le champ. C'est ce procédé qui est souvent utilisé pour vérifier si le formulaire a été bien rempli avant de l'envoyer. [Voir la page](#)

18. 16 : Appliquez : une fenêtre prompt

Appliquez directement la matière vue en effectuant un exercice similaire :

"Ecrire un programme qui demande le prénom de l'utilisateur à l'aide d'une fenêtre prompt et qui l'affiche dans le document."

La variable doit être nommée "proposition".

La fenêtre prompt doit comporter ces textes :

1. "Quel est votre prénom ?"
2. "Inscrivez votre prénom ici"

```
<html>
\\r\\n\\t<head>
\\r\\n\\t\\t<title>Ecrire un programme qui demande le prénom du visiteur via une fenêtre prompt et
qui affiche son nom dans le document</title>
\\r\\n\\t\\t*script=
\\r\\n\\t\\tfunction Main(){
```

```

\\r\\n\\t\\t\\t
\\r\\n\\t\\t\\t}
\\r\\n\\t\\t=script*
\\r\\n\\t</head>
\\r\\n\\t<body onLoad="Main();">
\\r\\n\\t</body>
\\r\\n</html>
<html>
\\r\\n\\t<head>
\\r\\n\\t\\t<title>Ecrire un programme qui demande le prénom du visiteur via une fenêtre prompt et
qui affiche son nom dans le document</title>
\\r\\n\\t\\t*script=
\\r\\n\\t\\t\\tvar proposition;
\\r\\n\\t\\t\\tfunction Main(){
\\r\\n\\t\\t\\t\\tproposition = prompt("Quel est votre prénom ?", "Inscrivez votre prénom ici");
\\r\\n\\t\\t\\t\\tdocument.write("Bonjour ");
\\r\\n\\t\\t\\t\\tdocument.write(proposition);
\\r\\n\\t\\t\\t\\t}
\\r\\n\\t\\t=script*
\\r\\n\\t</head>
\\r\\n\\t<body onLoad="Main();">
\\r\\n\\t</body>
\\r\\n</html>

```

Exécuter le code

Au chargement de la page, une fenêtre "prompt" demande à l'utilisateur de donner son prénom. La page affiche le message Bonjour [prénom]. [Voir la page](#)

18. 17 : Quizz

1. Lequel de ces événements ne retrouve-t-on pas dans un élément de formulaire ?

- a) onclick
- b) onchange
- c) onload

2. L'événement que l'on utilise pour vérifier s'il y a eu un changement de valeur est

- a) onclick
- b) onchange
- c) onsubmit

3. A quoi sert l'événement onsubmit ?

- a) A soumettre le formulaire
- b) A effectuer une action juste avant l'envoi du formulaire
- c) A effectuer une action juste après l'envoi du formulaire

18. 18 : Appliquez : l'événement onblur

Voici un exercice permettant de mettre en pratique ce que nous avons vu dans le complément qui précède.

"Ecrire un programme qui affiche le message de remerciement dans une alerte, lorsque l'on perd le focus d'un formulaire (onblur)"

```
<html>
\\r\\n\\t<head>
\\r\\n\\t\\t<title>Ecrire un programme qui affiche un message de remerciement dans une alerte lors de
la perte de focus dans un formulaire (onblur)</title>
\\r\\n\\t\\t*script=
\\r\\n\\t\\t\\tfunction Main(){
\\r\\n\\t\\t\\t\\talert("Merci pour votre remarque");
\\r\\n\\t\\t\\t}
\\r\\n\\t\\t=script*
\\r\\n\\t</head>
\\r\\n\\t<body>
\\r\\n\\t\\t<form name="accueil">
\\r\\n\\t\\t\\t<h3>Veuillez entrer votre remarque:</h3>
\\r\\n\\t\\t\\t<input type="text" name="remarque" >
\\r\\n\\t\\t</form>
\\r\\n\\t</body>
\\r\\n</html> <html>
\\r\\n\\t<head>
\\r\\n\\t\\t<title>Ecrire un programme qui affiche un message de remerciement dans une alerte lors de
la perte de focus dans un formulaire (onblur)</title>
\\r\\n\\t\\t*script=
\\r\\n\\t\\t\\tfunction Main(){
\\r\\n\\t\\t\\t\\talert("Merci pour votre remarque");
```

```
\\r\\n\\t\\t\\t}
\\r\\n\\t\\t=script*
\\r\\n\\t</head>
\\r\\n\\t<body>
\\r\\n\\t\\t<form name="accueil">
\\r\\n\\t\\t\\t<h3>Veuillez entrer votre remarque:</h3>
\\r\\n\\t\\t\\t<input type="text" name="remarque" onblur="Main();">
\\r\\n\\t\\t</form>
\\r\\n\\t</body>
\\r\\n</html>
```

Exécuter le code

Le programme affiche le message de remerciement, lorsque le champ du formulaire perd le focus.

[Voir la page](#)

Module 3 - Un peu plus sur le JavaScript

1 : Objectifs

Dans ce module, vous apprendrez à mieux connaître le langage JavaScript, quelles sont ses particularités, comment il est interprété, ...

En réalisant les exercices des modules précédents, vous avez sans doute remarqué qu'une faute de frappe, et donc une erreur, est vite arrivée. Il est donc important de savoir détecter et corriger une erreur...

2 : Qui interprète un programme JavaScript ?

Comme pour une page HTML, c'est le navigateur qui interprète le code JavaScript. eef

C'est donc le navigateur qui traduit, au fur et à mesure, le code JavaScript en un format compréhensible par l'ordinateur.

Remarque : Contrairement au langage Java, le Javascript n'est pas un langage compilé, il est interprété. Un programme écrit dans un langage "compilé" est traduit dans un langage compréhensible par la machine "une fois pour toutes" avant de pouvoir être utilisé, cette étape s'appelle la compilation. Les instructions du langage "interprété" seront par contre traduites au fur et à mesure lors de l'exécution du programme.

L'avantage du JavaScript : l'utilisateur n'a pas besoin d'installer un plugin (contrairement à une animation Flash), tout ce dont il a besoin est un navigateur qui interprète le JavaScript (c'est-à-dire pratiquement tous les navigateurs actuels : Internet Explorer, Firefox, Chrome, etc.).

L'inconvénient du JavaScript : Comme pour le code HTML, le code JavaScript peut être interprété différemment en fonction des navigateurs. Cela signifie que l'on pourra constater des différences à l'écran lors de l'exécution de certaines instructions.

3 : Les particularités du JavaScript

JavaScript est un langage de programmation. Chaque langage de programmation possède

- un ensemble de mots représentant le "vocabulaire" du langage
- une structuration logique représentant la "grammaire" du langage

Voici quelques particularités propres au langage JavaScript...

4 : La sensibilité à la casse

Le langage JavaScript est **sensible à la casse**. Cela signifie qu'il fait la différence entre un "A" majuscule et un "a" minuscule.

La fonction :

```
function main(){  
...  
}
```

est différente de la fonction :

```
function Main(){  
...  
}
```

Remarque : Vous pouvez bien évidemment utiliser les majuscules/minuscules dans le contenu textuel : "bienvenue dans l'univers de la bd !" ou "BIENVENUE DANS L'UNIVERS DE LA BD !". Les majuscules/minuscules n'auront d'importance au niveau du texte que lorsque l'on comparera deux valeurs. Par exemple : "TINTIN" est différent de "tintin".

5 : Appliquez : la sensibilité à la casse

Il y a 3 erreurs dans ce code. Corrigez-le afin qu'il s'exécute correctement.

```
<html>  
\\r\\n\\t<head>  
\\r\\n\\t\\t<title>Casse</title>  
\\r\\n\\t\\t*script=  
\\r\\n\\t\\t\\tfunction main(){  
\\r\\n\\t\\t\\t\\t\\tdocument.write("Bienvenue dans le monde de la BD !") ;  
\\r\\n\\t\\t\\t\\t}  
\\r\\n\\t\\t=script*  
\\r\\n\\t</head>  
\\r\\n\\t<body onload="Main();">  
\\r\\n\\t</body>  
\\r\\n</html> <html>  
\\r\\n\\t<head>  
\\r\\n\\t\\t<title>Casse</title>  
\\r\\n\\t\\t*script=  
\\r\\n\\t\\t\\tfunction Main(){  
\\r\\n\\t\\t\\t\\t\\tdocument.write("Bienvenue dans le monde de la BD !") ;  
\\r\\n\\t\\t\\t\\t}  
\\r\\n\\t\\t=script*  
\\r\\n\\t</head>  
\\r\\n\\t<body onload="Main();">  
\\r\\n\\t</body>
```

\\r\\n</html>

6 : Un ";" à la fin de chaque instruction

Vous avez certainement remarqué qu'en fin de ligne, on ajoute un point-virgule.
Ce point-virgule sert à séparer les **instructions**.

Qu'est-ce qu'une instruction ?

Une instruction, c'est **l'ordre** (**l'action**) que le programme va commander à la machine.
Un programme est composé d'une suite d'instructions.

Pour un code plus lisible, on écrit une instruction par ligne :

```
alert("Bienvenue dans l'univers de la BD !");  
alert("Merci de votre visite !");
```

Dans ce cas, les points-virgules en fin d'instructions ne sont **pas obligatoires**.

Pour un code plus compact, on peut écrire plusieurs instructions sur une ligne :

```
alert("Bienvenue dans l'univers de la BD !"); alert("Merci de votre visite !");
```

Dans ce cas, le point-virgule est **nécessaire entre deux instructions** !

Remarque : Pour un code plus homogène et bien écrit, on met un point-virgule à la fin de chaque instruction, même s'il y a un retour à la ligne.

7 : Appliquez ! Un ";" à la fin de chaque instruction

Ajoutez les points-virgules à la bonne exécution du programme.

```
<html>  
\\r\\n\\t<head>  
\\r\\n\\t\\t<title>Casse</title>  
\\r\\n\\t\\t*script=  
\\r\\n\\t\\t\\tfunction Main(){  
\\r\\n\\t\\t\\t\\t\\tdocument.write("Bienvenue dans le monde de la BD!<br>")  
\\r\\n\\t\\t\\t\\t\\tdocument.write("Sur ce site, vous trouverez toute information concernant la bande  
dessinée...<br>")  
\\r\\n\\t\\t\\t\\t\\tdocument.write("...Bonne visite!")  
\\r\\n\\t\\t\\t\\t}  
\\r\\n\\t\\t\\t=script*  
\\r\\n\\t</head>  
\\r\\n\\t<body onload="Main();">  
\\r\\n\\t</body>  
\\r\\n</html> <html>  
\\r\\n\\t<head>  
\\r\\n\\t\\t<title>Casse</title>  
\\r\\n\\t\\t*script=  
\\r\\n\\t\\t\\tfunction Main(){  
\\r\\n\\t\\t\\t\\t\\tdocument.write("Bienvenue dans le monde de la BD!<br>");document.write("Sur  
ce site, vous trouverez toute information concernant la bande  
dessinée...<br>");document.write("...Bonne visite!");
```


10 : Quizz

1. Pour qu'un guillemet soit interprété comme du texte, on ajoute devant celui-ci...

- a) message="Vous avez dit \"bizarre\" !" ;
- b) message="Vous avez dit /"bizarre/" !" ;
- c) message="Vous avez dit //"bizarre//" !" ;

11 : Accès à un objet du document

En JavaScript, on travaille en "collaboration" avec une page HTML.

Comme vous l'avez vu dans le module 2, il est possible de communiquer avec des éléments d'une page HTML comme par exemple un champ texte :

document.accueil.message.value = "Bienvenue dans l'univers de la BD !" ;

(accueil étant le nom du formulaire et message étant le nom du champ texte)

Chaque élément d'une page HTML est accessible via un **chemin hiérarchique logique** : dans le **document**, on trouve un formulaire **accueil** dans lequel on trouve le champ **message**. On atteint donc la valeur du champ par le chemin : **document.accueil.message.value**.

A ne pas oublier : Pour atteindre un élément d'un document, il faut lui donner un nom. C'est par son nom que l'on atteindra l'élément.

12 : Appliquez : accès à un objet du document

Complétez le HTML afin que le script s'exécute correctement.

```
<html>
\\r\\n\\t<head>
\\r\\n\\t\\t*script=
\\r\\n\\t\\t\\tfunction Main(){
\\r\\n\\t\\t\\t\\t\\tdocument.quizz.question1.value = "Quel est le nom du chien de Boule ?";
\\r\\n\\t\\t\\t\\t}
\\r\\n\\t\\t=script*
\\r\\n\\t</head>
\\r\\n\\t<body onload="Main();">
\\r\\n\\t\\t<form>
\\r\\n\\t\\t\\t<input type="text">
\\r\\n\\t\\t</form>
\\r\\n\\t</body>
\\r\\n</html> <html>
\\r\\n\\t<head>
\\r\\n\\t\\t*script=
\\r\\n\\t\\t\\tfunction Main(){
\\r\\n\\t\\t\\t\\t\\tdocument.quizz.question1.value = "Quel est le nom du chien de Boule ?";
\\r\\n\\t\\t\\t\\t}
\\r\\n\\t\\t=script*
\\r\\n\\t</head>
\\r\\n\\t<body onload="Main();">
```

```
\\r\\n\\t\\t<form name="quizz">
\\r\\n\\t\\t\\t<input type="text" name="question1">
\\r\\n\\t\\t</form>
\\r\\n\\t</body>
\\r\\n</html>
```

13 : Conclusion

Chaque langage de programmation possède des particularités qu'il faut maîtriser pour éviter des **erreurs de syntaxe**.

Il existe de nombreux livres et sites Internet concernant la "**grammaire**" du langage JavaScript. Vous trouverez des ressources dans les compléments de ce module.

14 : L'erreur est humaine

Toutes ces particularités du langage peuvent vous amener à faire des erreurs, par exemple mettre un "A" majuscule pour la commande "alert()".

Il faut bien admettre que **l'erreur est humaine** et que, si ce n'est pas la première, ce ne sera pas non plus la dernière !

Dans un programme, on peut retrouver deux types d'erreurs :

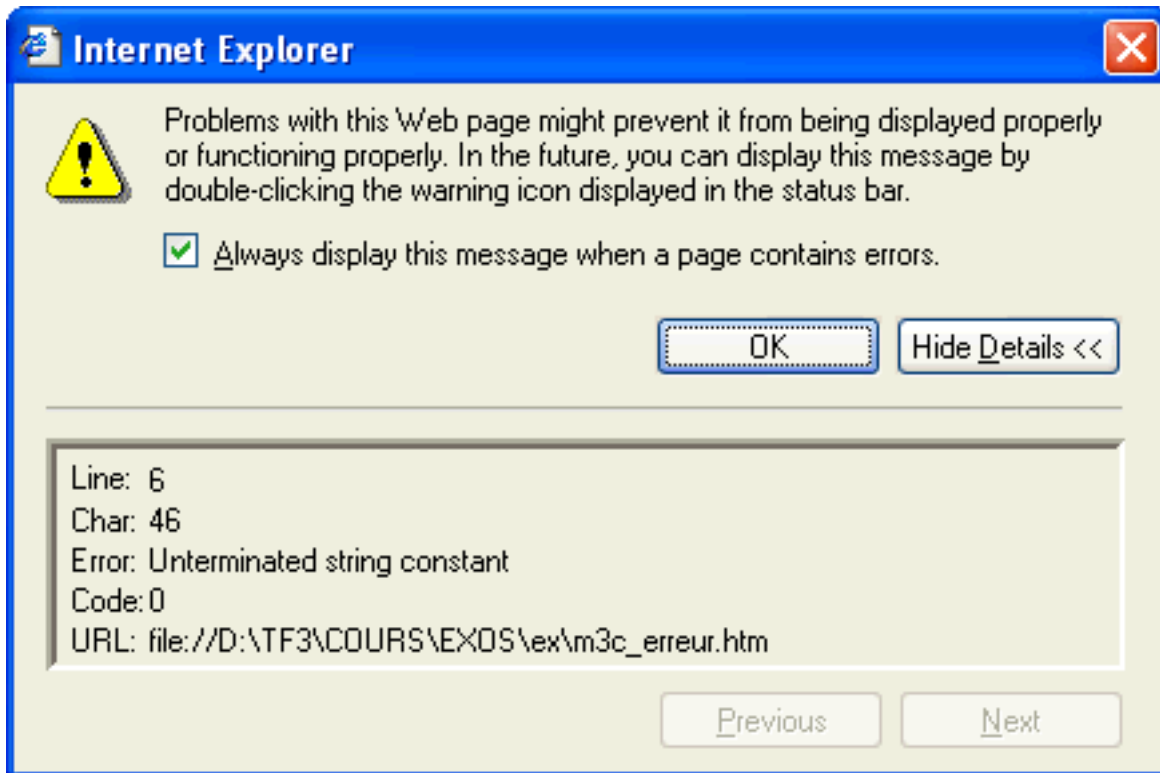
- **les erreurs de syntaxe** : faute de frappe, oubli de guillemet, ...
- **les erreurs logiques** : appel de la fonction Main() alors qu'elle n'existe pas, oubli d'appel de la fonction Main() au chargement de la page, ...

Voici comment gérer les erreurs d'un programme.

15 : Comment détecter une erreur ?

Une **erreur** se détecte facilement dans un programme car généralement, il ne tourne pas correctement ;-)

Néanmoins, le navigateur prévient lorsqu'il rencontre une erreur dans un programme : généralement, vous obtenez un " horrible " message d'alerte ...



... et un picto d'alerte s'affiche dans la barre d'état (en bas à gauche) du navigateur.



Généralement, une erreur entraîne une autre : vous pouvez obtenir **plusieurs messages d'alerte** à la suite !

Remarque : Si vous n'obtenez pas la grande fenêtre d'alerte, il vous suffit de double-cliquer sur le picto d'alerte en bas à gauche de l'écran du navigateur.

16 : Comment retrouver une erreur ?

Les messages d'erreur du navigateur Internet Explorer précisent la ligne et l'endroit sur la ligne où il trouve l'erreur... ATTENTION ! L'emplacement réel de l'erreur n'est **pas toujours celui proposé** par le navigateur !

Le meilleur moyen de détecter l'emplacement exact d'une erreur est de mettre la ligne suspecte en **commentaire**, comme ceci :

```
<html>
  <head>
    <title>
```

```

        Comment retrouver une erreur
    </title>
    <script>
        function Main(){
            // alert("Bienvenue dans le monde de la "BD" !");
        }
    </script>
</head>
<body onload="Main();" >
</body>
</html>

```

Si la page s'affiche sans erreur, c'est que l'erreur se trouve bien au niveau de la ligne en commentaire.

Comme il n'existe pas d'outils de **débogage** (= détecteur de bug) vraiment performants pour le langage JavaScript, cette astuce de programmeur va vous être bien utile !

=> Il ne reste plus qu'à trouver l'erreur dans la ligne...

17 : Les points importants à vérifier

Quelles sont les erreurs possibles dans un programme ?

- Erreur d'écriture :

`functoin Main()` au lieu de `function Main()`

- Problème de majuscule/minuscule

`document.Write()` au lieu de `document.write()`

- Oubli de fermeture de parenthèses ou d'accolades

`alert("Bienvenue"` au lieu de `alert("Bienvenue")`

- Conflit d'utilisation de guillemets

`alert("Qui a dit : "Je pense donc je suis ! " ?")` au lieu de `alert("Qui a dit:\ "Je pense donc je suis ! \ " ?")`

- Oubli de point-virgule entre deux instructions se trouvant sur la même ligne

`document.write("Bonjour !
") document.write("Comment allez-vous ?")` au lieu de `document.write("Bonjour !
"); document.write("Comment allez-vous ?")`

- Utilisation d'un mot réservé pour le nom d'une fonction

`function function(){ ... }` au lieu de `function messageBienvenue(){ ... }`

18 : Conclusion : Les bonnes manières pour programmer

Pour éviter un maximum **d'erreurs**, il faut être **systématique** :

- Chaque fois que vous ouvrez une accolade, fermez-la directement. Idem pour une parenthèse ou un guillemet. Cela vous évitera d'omettre un élément fermant.
- Placez **une instruction par ligne**. Le code sera plus lisible et retrouver les erreurs sera plus simple.
- **Indentez** votre code (ajoutez des retraits), cela permet également de rendre le code plus clair.

19 : Mémo

Interprété ou compilé?

A la différence d'autres langages comme le **Java**, le **JavaScript** n'est pas **compilé** mais **interprété** par le navigateur. De ce fait, son code est **portable** sur plusieurs **navigateurs** sans la moindre modification, mais son interprétation peut être légèrement différente d'un à l'autre.

Les particularités du JavaScript

Etant donné qu'il s'agit d'un **langage de programmation**, il possède des mots qui forment son "**vocabulaire**", et une structuration logique, qui représente sa "**grammaire**".

Ce langage est **sensible à la casse** (majuscule / minuscule): **Main()** et **main()** sont donc deux fonctions différentes.

Il est également **sensible à la ponctuation**, n'oubliez jamais votre ; à la fin d'une instruction.

Les **caractères spéciaux**, quant à eux, doivent être précédés d'un \ pour s'afficher sans être interprétés par le navigateur.

Voici comment on ajoute des guillemets dans du texte :

```
alert("Bienvenue dans l'univers de la \"BD\" !")
```

Les mots réservés

Le **JavaScript** n'autorise pas l'usage de certains mots pour nommer une fonction ou une variable, comme **alert**, **function**, ou encore **prompt**.

La liste des mots réservés (déjà utilisés ou à venir) peut être trouvée sur ce site:

https://developer.mozilla.org/fr/docs/Web/JavaScript/Reference/Mots_r%C3%A9serv%C3%A9s

Une fois tous vos éléments **nommés correctement**, vous pouvez enfin y accéder, voici la façon pour y parvenir:

Si on suit un chemin hiérarchique logique dans le document, on trouve un formulaire "**accueil**" dans lequel est placé le champ "**message**".

On atteint donc la valeur du champ "**message**" par le chemin : **document.accueil.message.value**

Ca ne marche pas?

Pas de stress, la plupart des erreurs viennent de **trois sources** :

- les **erreurs de syntaxe** : faute de frappe, oubli de guillemets, de point-virgule, utilisation d'un mot réservé, ...
- les **fautes de frappe** : différence entre majuscules/minuscules, inversion de lettres, ...
- les **erreurs logiques** : appel de la fonction Main() alors qu'elle n'existe pas, oubli d'appel de la fonction Main() au chargement de la page, ...

Pour voir la source de l'erreur, vous pouvez vous inspirer du **message d'alerte** ou du **picto** présent dans la barre d'état du navigateur.

En général, les **messages d'erreur** sont assez incompréhensibles. On peut toutefois s'en sortir avec **un peu de logique**: tout ce qui a été ouvert doit être fermé, les majuscules respectées, les ; bien placés, les noms d'objets bien écrits, ...

Ensuite, rien de plus simple, vous mettez les lignes suspectes **en commentaire** (avec le // comme vu précédemment), et vous relancez la page incriminée.

Pensez également à **changer de navigateur**, Firefox vous offre une description des bugs plus claire. En règle générale, il est souvent recommandé de tester son programme dans plusieurs navigateurs.

Améliorez-vous

Pour finir, n'oubliez pas les bonnes manières pour programmer: tout ce qui est ouvert doit être fermé directement, aérez votre code, commentez-le au besoin et indentez-le.

20. 1 : Compléments

Voici quelques explications supplémentaires concernant les particularités du langage JavaScript

20. 2 : Langage compilé / Langage interprété

Il existe **deux types de langages** :

- les langages interprétés (par exemple le JavaScript)
- les langages compilés (par exemple le Java).

Un langage **interprété** est un langage qui est traduit en un langage compréhensible par la machine au moment de son exécution.

Un langage **compilé** est un langage qui est traduit en un langage compréhensible par la machine lors de la compilation, ce qui rend l'exécution du programme plus rapide car il n'y a plus de "traduction" à effectuer pendant l'exécution du programme.

Les **avantages** et les **inconvénients** des deux types de langages :

Contrairement au langage interprété, chaque modification d'un langage compilé nécessite une **recompilation** du programme pour que les modifications soient prises en compte à l'exécution de celui-ci...

Pour un langage compilé, les erreurs de syntaxe sont détectées à la compilation et non à l'exécution. Vous obtiendrez donc moins d'erreurs à l'exécution de votre programme puisqu'elles auront été détectées avant !

Remarque : Le fait que le JavaScript soit un langage interprété est un choix délibéré : cela le rend

- **plus accessible** car il se trouve directement dans une page HTML,
- **plus léger** car le format du code est de type ASCII (on peut visualiser le code avec le Bloc-notes).

20. 3 : Quizz

1. Le langage JavaScript est un langage

- a) interprété par le navigateur
- b) compilé et exécuté par le navigateur

20. 4 : Liste des mots réservés

Voici la liste des mots à éviter en tant que nom de fonction ou nom de variable.

Sachez que certains mots ne sont pas encore utilisés par le langage mais sont prévus pour une utilisation future. C'est le cas pour le mot réservé "debugger" qui n'a pas encore d'utilisation mais qui pourrait en avoir dans le futur...

| Liste des mots reserves en Javascript | | |
|---------------------------------------|------------|--------------|
| abstract | final | public |
| boolean | finally | return |
| break | float | short |
| byte | for | static |
| case | function | super |
| catch | goto | switch |
| char | if | synchronized |
| class | implement | this |
| const | import | throw |
| continue | in | throws |
| debugger | instanceof | transient |
| default | int | true |
| delete | interface | try |
| do | long | typeof |
| double | native | var |
| else | new | void |
| enum | null | volatile |
| export | package | while |
| extends | private | with |
| false | protected | |

20. 5 : Liste des caractères spéciaux

Pour éviter des conflits de guillemets (guillemets déclarant du contenu textuel et guillemets faisant partie du contenu textuel), on utilise les caractères spéciaux en précédant le guillemet d'un antislash (\) .

```
alert("Dans quelle BD trouve-t-on la phrase : \"Je ne suis pas gros, je suis juste un petit peu enveloppé !\" ?")
```

Une fenêtre d'alerte n'accepte pas le format HTML. Impossible, donc, d'écrire en couleur ou en gras... Par contre, les caractères spéciaux permettent l'ajout d'un **retour à la ligne** avec le caractère `\n` (l'équivalent du `
` en HTML).

Pour formater le contenu textuel d'une fenêtre d'alerte, on utilise également les caractères spéciaux :

alert ("Bienvenue dans l'univers de la BD !\nSur ce site, vous trouverez toute l'information concernant la bande dessinée...\n...Bonne visite !")

| Liste des caractères spéciaux | |
|-------------------------------|---|
| <code>\b</code> | touche de suppression |
| <code>\f</code> | saut de ligne d'impression |
| <code>\n</code> | retour à la ligne |
| <code>\r</code> | appui sur la touche ENTREE (retour chariot) |
| <code>\t</code> | tabulation |
| <code>\"</code> | guillemets doubles |
| <code>\'</code> | guillemets simples |
| <code>\\</code> | caractère antislash |

20. 6 : Savoir lire les messages d'erreur...

Il n'est pas simple de comprendre les messages d'erreurs affichés par les navigateurs. En effet, ceux-ci sont des **messages génériques** et traduits de l'anglais... Ce qui les rend donc moins lisibles !

Voici quelques erreurs possibles en fonction des différents messages obtenus avec Internet Explorer :

Objet attendu :

- Vous avez fait une faute de frappe : Alert au lieu de alert
- Vous faites appel à un objet qui n'existe pas : `<body onload="fctPrincipale() ;">` alors que la fonction `fctPrincipale()` n'existe pas

')' attendu :

- Il y a un conflit de guillemets dans une chaîne de caractères : `alert("Qui a dit : "Je pense donc je suis" ?")`

' ;' attendu

- Il y a un conflit de guillemets dans une chaîne de caractères : `message ="Qui a dit : "Je pense donc je suis" ?" ;`

Constante de chaîne non terminée

- Il manque les guillemets de fermeture de la chaîne de caractères

'document.form.message.valeur' a la valeur NULL ou n'est pas un objet

- La propriété "valeur" n'existe pas pour un élément du formulaire

'document.form.messagesssss.value' a la valeur NULL ou n'est pas un objet

- Le champ messagesssss n'existe pas ou bien le formulaire form n'existe pas

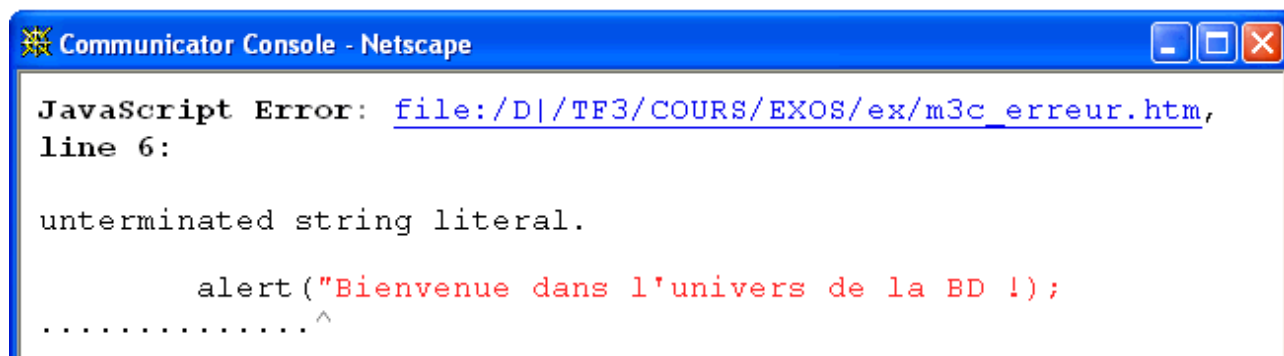
Cet objet ne gère pas cette propriété ou cette méthode

- La propriété ou la méthode de l'objet est incorrecte : document.Write()

20. 7 : Les erreurs JavaScript avec Firefox

Un des atouts de Firefox est son **débogueur JavaScript**. Beaucoup plus puissant que celui d'Internet Explorer, il décrit les "bugs" plus clairement.

Voici le message obtenu lors de l'exécution du programme erreur.htm (voir complément précédent) :



Ce message est clair : le programme n'arrive pas à trouver la fin de la chaîne de caractères. Le pointeur est placé au début de la chaîne de caractères en question.

Pour obtenir cette petite fenêtre appelée **Console JavaScript**, il vous suffit de taper dans la barre des adresses (à la place de l'URL) : **javascript:**

Remarque : Lorsque vous êtes bloqué sur une erreur, essayez votre programme dans différents navigateurs, les différents messages pourront peut-être vous mettre sur la voie...

Module 4 - Stocker des valeurs

1 : Objectifs

Dans les modules précédents, vous avez appris à afficher du contenu et à récupérer des informations provenant de l'utilisateur.

Dans ce module, vous apprendrez à **stocker ces données** afin de les **réutiliser**.

2 : Dire plusieurs fois la même chose

Dans le premier module, vous avez affiché le texte "Bienvenue dans l'univers de la BD !" dans le document.

Reprenons cet énoncé avec une petite amélioration :

"Afficher 3 fois le texte "Bienvenue dans l'univers de la BD" dans le document" (**nom du fichier : écrire3fois.htm**)

Ecrivez le code du programme répondant à cette spécification et enregistrez-le sous le nom "ecrire3fois.htm".

Vous trouverez la solution de l'exercice à la page suivante.

3 : Solution de l'exercice

Le code est assez simple :

```
<html>
  <head>
    <title>
      Afficher 3 fois le texte "Bienvenue dans l'univers de la BD"
dans le document
    </title>
    <script>
      function Main(){
        document.write("Bienvenue dans l'univers de la BD !<br>") ;

        // afficher une 1re fois
        document.write("Bienvenue dans l'univers de la BD !
<br>") ;

        // afficher une 2e fois
        document.write("Bienvenue dans l'univers de la BD !
<br>") ;

        // afficher une 3e fois
      }
    </script>
```

```
</head>
<body onload="Main()">
</body>
</html>
```

Remarque: A la suite de "Bienvenue dans l'univers de la BD", on a ajouté la balise HTML **
. Cette balise permet d'effectuer un **retour à la ligne. Sans cette balise, le texte se serait affiché comme suit : **Bienvenue dans l'univers de la BD !Bienvenue dans l'univers de la BD !Bienvenue dans l'univers de la BD !** (les trois phrases se suivent sans retour chariot !)

Exécuter le programme

Voici le programme qui affiche trois fois le même texte dans le document. [Voir la page](#)

4 : Conclusion

Le texte est bien affiché **3 fois**, mais le code est "lourd" :

1. Il est inutilement long, vous avez dû recopier le texte plusieurs fois
2. Si le texte de bienvenue change, vous voilà bon pour modifier plusieurs fois le texte !

5 : Une solution plus "propre"

Voici un moyen d'éviter la répétition du texte : stockons-le dans une **variable** (nom du fichier : **ecrire3foisV2.htm**, **V2** pour version 2).

```
<html>
  <head>
    <title>
      Afficher 3 fois le texte "Bienvenue dans l'univers de la BD"
dans le document
    </title>
    <script>
      var message;

      function Main(){
        message="Bienvenue dans l'univers de la BD !<br>";
        document.write(message);
        document.write(message);
        document.write(message);
      }
    </script>
  </head>
  <body onload="Main()">
</body>
```



```
</html>
```

Enregistrez ce code sous le nom "ecrire3foisV2.htm" et exécutez le programme.

Voilà qui résout les problèmes :

Le code est moins long et dans le cas de la modification du texte, le changement est **automatique**.

Exécuter le programme

Voici le programme qui affiche trois fois le même texte dans le document à l'aide d'une variable.

[Voir la page](#)

6 : Ce que l'on a fait

Plutôt que d'écrire à chaque fois une valeur, vous l'avez **stockée** dans une variable.

Une variable est une **étiquette** qui permet au programme de **référencer** une valeur.

Dans l'exemple précédent, la valeur est

"Bienvenue dans l'univers de la BD !"

et l'étiquette de cette valeur (le nom de la variable) est

message.

Cette variable a été **définie** au préalable avec la commande: **var message ;**.

Et on lui a **associé une valeur** avec la commande : **message = "Bienvenue dans l'univers de la BD !"** ; .

7 : Un programme qui lit (cf. module 2)

Rappelez-vous, dans le module 2, vous avez **déjà** utilisé la notion de variable en stockant le contenu inséré par l'utilisateur dans la variable **prop** :

```
<html>
  <head>
    <title>
      Lecture de la valeur d'une variable
    </title>
    <script>
      var prop;
      // on déclare la variable proposition
      function Main(){
        prop = document.echo.proposition.value;
        // on enregistre la proposition dans la variable
        proposition
        document.echo.injure.value = prop;
        // on écrit la valeur proposée dans le champ texte "injure"
      }
    </script>
  </head>
  <body>
    <div>
      <input type="text" value="proposition" />
      <input type="button" value="OK" />
    </div>
    <div>
      <input type="text" value="injure" />
    </div>
  </body>
</html>
```

```

        </script>
    </head>
    <body>
        <form name="echo">
            <input type="text" name="proposition">
                <input type="button" value="Répéter la proposition"
onclick="Main()">
                Le capitaine Haddock dit : <input type="Text"
name="injure">
            </form>
        </body>
    </html>

```

8 : Bien déclarer une variable

1. Généralement , on **déclare** (crée) les variables en début de programme .

Remarque : il est possible de déclarer une variable dans une fonction. Dans ce cas, elle n'existe **que dans cette fonction** et n'est pas "utilisable" en dehors de celle-ci (vous verrez ceci plus en détail dans le module 9 : Les fonctions).

```

<html>
    <head>
        <title>
            Déclaration de variable
        </title>
        <script>
            var message;
            function Main(){
                //instructions de la fonction
            }
        </script>
    </head>
    <body onload="Main()">
    </body>
</html>

```

2. N'oubliez jamais de refermer les guillemets, vous obtiendriez une erreur !

```
var message = "coucou;
```

3. Le nom d'une variable doit être écrit :

- en un seul mot,
- sans accent
- et commencer par une lettre ou par "_" (underscore).

```
var message ; // nom de variable correct
var _message ; // nom de variable correct
var lmessage ; // nom de variable incorrect
```

9 : Quizz

1. Pour les réutiliser, on stocke les données dans une

- Variable
- Valeur

2. Parmi ces 3 noms de variable, lequel est correct ?

- catégorie
- 1reProposition
- _couleur

3. Laquelle de ces déclarations est correcte ?

- Var texte ;
- var var ;
- var mode ;

10 : Appliquez : utiliser les variables

Ce programme affiche 4 fois le texte "Bienvenue dans l'univers de la BD!" de plus en plus grand. Modifiez le code de ce programme afin de stocker le message dans une variable nommée "texte".

```
<html>
\\r\\n\\t<head>
\\r\\n\\t\\t<title>Ecrire un programme qui affiche 4 fois le texte "Bienvenue dans le monde de la BD!"
de plus en plus grand </title>
\\r\\n\\t*script=
\\r\\n\\t\\tfunction Main(){
\\r\\n\\t\\t\\t\\tdocument.write("<h4>");
\\r\\n\\t\\t\\t\\tdocument.write("Bienvenue dans le monde de la BD!");
\\r\\n\\t\\t\\t\\tdocument.write("</h4>");
\\r\\n\\t\\t\\t\\tdocument.write("<h3>");
\\r\\n\\t\\t\\t\\tdocument.write("Bienvenue dans le monde de la BD!");
\\r\\n\\t\\t\\t\\tdocument.write("</h3>");
```

```

\\r\\n\\t\\t\\tdocument.write("<h2>");
\\r\\n\\t\\t\\tdocument.write("Bienvenue dans le monde de la BD!");
\\r\\n\\t\\t\\tdocument.write("</h2>");
\\r\\n\\t\\t\\tdocument.write("<h1>");
\\r\\n\\t\\t\\tdocument.write("Bienvenue dans le monde de la BD!");
\\r\\n\\t\\t\\tdocument.write("</h1>");
\\r\\n\\t\\t}
\\r\\n\\t\\t=script*
\\r\\n\\t</head>
\\r\\n\\t<body onload="Main();">
\\r\\n\\t</body>
\\r\\n</html> <html>
\\r\\n\\t<head>
\\r\\n\\t\\t<title>Ecrire un programme qui affiche 4 fois le texte "Bienvenue dans le monde de la BD!"
de plus en plus grand </title>
\\r\\n\\t*script=
\\r\\n\\t\\tvar texte;
\\r\\n\\t\\ttexte="Bienvenue dans le monde de la BD!";
\\r\\n\\t\\tfunction Main(){
\\r\\n\\t\\t\\t\\tdocument.write("<h4>");
\\r\\n\\t\\t\\t\\tdocument.write(texte);
\\r\\n\\t\\t\\t\\tdocument.write("</h4>");
\\r\\n\\t\\t\\t\\tdocument.write("<h3>");
\\r\\n\\t\\t\\t\\tdocument.write(texte);
\\r\\n\\t\\t\\t\\tdocument.write("</h3>");
\\r\\n\\t\\t\\t\\tdocument.write("<h2>");
\\r\\n\\t\\t\\t\\tdocument.write(texte);
\\r\\n\\t\\t\\t\\tdocument.write("</h2>");
\\r\\n\\t\\t\\t\\tdocument.write("<h1>");
\\r\\n\\t\\t\\t\\tdocument.write(texte);
\\r\\n\\t\\t\\t\\tdocument.write("</h1>");
\\r\\n\\t\\t}
\\r\\n\\t\\t=script*
\\r\\n\\t</head>
\\r\\n\\t<body onload="Main();">
\\r\\n\\t</body>
\\r\\n</html>

```

11 : Quizz

1. Laquelle de ces instructions ne récupère PAS la valeur du champ proposition dans la variable injure ?

- a) injure = document.form.proposition.value ;
- b) document.form.proposition.value = injure ;

12 : Manipulez plusieurs variables

Jusqu'à présent, vous avez appris à stocker une valeur dans une variable et à afficher la valeur de la variable telle quelle. Maintenant, vous allez travailler avec les variables . Vous allez les réutiliser pour afficher du texte dynamique.

Insérer, dans une page HTML, le texte lacunaire suivant :

Les ... de cette bande ... sont des petits ... tout bleus

"Créer un programme qui demande les mots manquants (à l'aide d'un formulaire) à l'utilisateur et affiche le texte complété avec les mots proposés par l'utilisateur dans une fenêtre d'alerte." (nom du fichier : txtLacun.htm)

Si vous vous sentez d'attaque, essayez de faire l'exercice seul(e) !

Traduisons cet énoncé et trouvons les étapes principales du développement du programme :

1. La page HTML contient le texte " Les ... de cette bande ... sont des petits ... tout bleus".
2. Le programme récupère LES propositions de l'utilisateur dans des variables à l'aide d'un formulaire.
3. Le programme affiche (dans une fenêtre d'alerte) le texte lacunaire complété par les propositions de l'utilisateur.

Les étapes de développement de ce programme sont décrites dans les 3 points suivants.

13 : Explication du code 1/3

Voici le code de départ.

```
<html>
  <head>
    <title>Texte lacunaire</title>
  </head>
  <body>
    <h1>Complétez le texte lacunaire suivant:</h1>
    <p>Les ... de cette bande ... sont des petits ... tout bleus</p>
  </body>
</html>
```

Continuons !

14 : Explication du code 2/3

Demandons les mots manquants par le biais d'un formulaire (lignes 18 à 23) et stockons-les dans des variables : prop1, prop2, prop3.

Remarque : pour la première fois, vous avez utilisé 3 variables. Un programme peut contenir plusieurs variables.

```
<html>
  <head>
    <title>Texte lacunaire</title>
    <script>
      var prop1;
      var prop2;
      var prop3;
      function Main(){
        prop1=document.form.prop1.value;
        prop2=document.form.prop2.value;
        prop3 = document.form.prop3.value;
      }
    </script>
  </head>
  <body>
    <h1>Complétez le texte lacunaire suivant:</h1>
    <p>Les ... de cette bande ... sont des petits ... tout bleus</p>
    <form name="form">
      1er mot manquant: <input type="text" name="prop1"/><br>
      2ème mot manquant: <input type="text" name="prop2"><br>
      3ème mot manquant: <input type="text" name="prop3"><br>
      <input type="button" value="Compléter le texte" onClick="Main();">
    </form>
  </body>
</html>
```

Continuons !

15 : Explication du code 3/3

Affichons la phrase complétée par l'utilisateur (ligne 12).

```
<html>
```

```

<head>
<title>Texte lacunaire</title>
<script>
    var prop1;
    var prop2;
    var prop3;
    function Main(){
        prop1=document.form.prop1.value;
        prop2=document.form.prop2.value;
        prop3 = document.form.prop3.value;
        alert("Les " + prop1 + " de cette bande " + prop2 + " sont des
petits " + prop3 + " tout bleus " ) ;
    }
</script>
</head>
<body>
<h1>Complétez le texte lacunaire suivant:</h1>
<p>Les ... de cette bande ... sont des petits ... tout bleus</p>
<form name="form">
1er mot manquant: <input type="text" name="prop1"/><br>
2ème mot manquant: <input type="text" name="prop2"><br>
3ème mot manquant: <input type="text" name="prop3"><br>
    <input type="button" value="Compléter le texte" onClick="Main();">
    </form>
</body>
</html>

```

Recopiez l'entièreté du code dans votre éditeur et enregistrez votre document sous le nom "textLacun.htm". Il vous reste à exécuter ce programme.

Voir le résultat

Pour voir le comportement attendu de cette page, suivez ce lien. [Voir la page](#)

16 : Conclusion

Dans la commande "alert("Les " + prop1 + " de cette bande " + prop2 + " sont des petits " + prop3 + " tout bleus ")", on a "collé" des chaînes de caractères et des variables à l'aide de l'opérateur "+" : c'est ce qu'on appelle la concaténation.

N'oubliez pas d'ajouter un espace autour de la variable car vous pourriez obtenir le résultat suivant : Leshérosde cette bandedessinéesont des petitsbonshommetout bleus

17 : Appliquez : la concaténation

Ecrire un programme qui demande un prénom via un formulaire et qui affiche le texte : "Il était une fois l'histoire de [PRENOM]. [PRENOM] se promenait sur la plage. Soudain, [PRENOM] entendit

une personne appeler : [PRENOM] ![PRENOM] !... [PRENOM] se retourna mais se rendit vite compte que la personne était en train de rappeler son chien [PRENOM] !!!".

[PRENOM] étant la valeur de la variable prenom définie au préalable." (nom du fichier : [concat.htm](#))

Etapes du développement :

1. créer le formulaire (name="form") demandant un prénom (name="prenom")
2. récupérer la proposition dans une variable (prenom)
3. afficher dans le document, avec **write()**, le texte "Il était une fois l'histoire de [PRENOM]. [PRENOM] se promenait sur la plage lorsque, soudain, [PRENOM] entendit une personne appeler : [PRENOM] ![PRENOM] !... [PRENOM] se retourna mais se rendit vite compte que la personne qui venait d'appeler était en train de rappeler son chien [PRENOM] !!!"

```
<html>
\\r\\n\\t<head>
\\r\\n\\t\\t<title>Concaténation</title>
\\r\\n\\t*script=
\\r\\n\\t\\tfunction Main(){
\\r\\n\\t\\t}
\\r\\n\\t=script*
\\r\\n\\t</head>
\\r\\n\\t\\t\\t<input type="button" value="Afficher" onClick="Main();">
\\r\\n\\t</body>
\\r\\n</html> <html>
\\r\\n\\t<head>
\\r\\n\\t\\t<title>Concaténation</title>
\\r\\n\\t*script=
\\r\\n\\t\\tvar prenom;
\\r\\n\\t\\tfunction Main(){
\\r\\n\\t\\t\\tprenom = document.form.prenom.value;
\\r\\n\\t\\t\\tdocument.write("Il était une fois " + prenom + ". " + prenom + " se promenait sur la plage
lorsque, soudain, "+prenom+" entendit une personne appeler : " + prenom + " !" + prenom + "!! "
+ prenom + " se retourna mais se rendit vite compte que la personne était en train de rappeler son
chien " + prenom + " !!!");
\\r\\n\\t\\t\\t}
\\r\\n\\t\\t=script*
\\r\\n\\t</head>
\\r\\n\\t\\t<form name="form">
\\r\\n\\t\\t\\tVotre prénom: <input type="text" name="prenom">
\\r\\n\\t\\t\\t<input type="button" value="Afficher" onClick="Main();">
\\r\\n\\t\\t</form>
\\r\\n\\t</body>
\\r\\n</html>
```


18 : Quizz

1. Pour "coller" du texte et des variables, on a utilisé l'opérateur :

- a) +
- b) "
- c) *

2. Quel terme utilise-t-on pour signifier que l'on "colle" du texte et des variables ensemble ?

- a) La déclaration
- b) La concaténation
- c) La collation

3. Laquelle de ces instructions est correcte ?

- a) alert("Les " + prop1 + " de cette bande " + prop2 + " sont des petits " + prop3 + " tout bleus ") ;
- b) alert("Les + prop1 + " de cette bande " + prop2 + " sont des petits " + prop3 + " tout bleus ") ;
- c) alert("Les " + prop1 + " de cette bande " prop2 + " sont des petits " + prop3 + " tout bleus ") ;

4. Sachant que la variable "compteur" vaut 3, laquelle de ces instructions s'affichera sans erreur ?

- a) alert("Vous avez gagné " + compteur + BD d'Astérix et Obélix") ;
- b) alert("Vous avez gagné " + compteur + " BD d'Astérix et Obélix") ;
- c) alert("Vous avez gagné " + compteur + " BD d'Astérix et Obélix") ;

19 : Synthèse

Dans tout programme, dès que l'on travaille avec des données, on utilise des **variables** pour les **stocker** afin de pouvoir les réutiliser.

Les variables rendent le code plus clair, plus réutilisable, plus générique.

Les variables sont faites pour être manipulées, la **concaténation** est un exemple de manipulation de variables.

20 : Mémo

Les variables

Une variable permet de **stocker une valeur**. Cela limite les **répétitions**, allège **le code**, et diminue le nombre d'erreurs dues à une **faute de recopiage**.

Il s'agit donc d'une **étiquette** qui permet au programme de **référer une valeur**.

Le **nom d'une variable** doit être écrit :

- en un seul mot
- sans accent
- et commencer par une lettre ou par "_" (underscore).

La **déclaration** (ou création) d'une variable se fait généralement en **début de programme**. On peut la définir **dans une fonction**, mais elle n'existerait **que dans cette fonction**, et pas en dehors.

Les **types de variables** les plus fréquents:

- texte (on entoure le contenu de guillemets)
- numérique
- tableau de données, ce point sera abordé dans un chapitre ultérieur
- booléen (vrai/faux)

Définir une variable

var proposition;

Et on lui attribue une **valeur** comme ceci:

```
proposition = "The quick brown fox jumps over the lazy dog";
```

Lire du contenu

Pour lire le **contenu** d'une variable, il suffit d'utiliser son nom sans guillemets, de cette façon:

```
alert ("Nous sommes le " + jour);
```

Ceci affichera une **pop-up** indiquant la phrase "Nous sommes le " suivie du contenu de la variable **jour**.

Pour ajouter une variable dans du texte, nous avons fait une **concaténation** entre une **chaîne de caractères** et une **variable** grâce au signe "+".

Voici un second exemple

```
document.write ("Nous sommes le " + NumJour + " jour de l'année.");
```

Peut-on utiliser une variable non déclarée ?

Oui, on appelle cela une **déclaration implicite**. Cela n'est toutefois **pas conseillé**, pour deux raisons:

- la plupart des autres langages **nécessitent** la déclaration des variables dans leur code, autant directement prendre de **bonnes habitudes**.
- en déclarant vos variables, vous avez un endroit où elles sont toutes reprises, leur type (entier, booléen, ...) ainsi que leur nom, ceci permet de **limiter les erreurs dans votre code**.

Travailler sur du texte

En plus de la **concaténation**, il existe d'autres fonctions permettant de traiter du texte.

Une d'entre-elles est le **"substring"**, qui isole une partie d'une chaîne de caractères:

```
var prop ;  
var prop6let ; // proposition 6 lettres  
prop = document.BD.prop.value; // on récupère la proposition dans le formulaire  
prop6let = prop.substr(0,6); // on part de la lettre 0, on reprend les 6 premières lettres, et on stocke  
tout ça dans la variable prop6let
```

Si d'autres méthodes permettant de traiter les chaînes de caractères vous intéressent, vous pouvez lire les compléments de ce cours, qui vous parlent de l'objet **String()** plus en détail.

21. 1 : Compléments

Dans les compléments, vous découvrirez plus d'informations sur les variables et le travail avec du texte. Vous apprendrez ainsi à manipuler des chaînes de caractères, les tronquer, les mettre en gras ou en majuscules...

21. 2 : Déclarer une variable ?

Comme il a été dit dans ce module, une variable doit en principe être déclarée avant d'être utilisée.

Voici cependant un exemple de programme utilisant une variable sans la déclarer :

```
<html>
  <head>
    <title>
      Pas de déclaration de variable
    </title>
    <script>
      function Main(){
        injure = document.BD.prop.value;
        // récupération de la valeur insérée
        // la variable injure n'a pas été déclarée
        document.BD.echo.value = injure;
        // affichage de la variable injure dans le champ echo
      }
    </script>
  </head>
  <body>
    <form name="BD">
      <h1>Proposez une des injures du capitaine Haddock</h1>
      <input type="text" name="prop">
        <input type="button" value="r&eacute;p&eacute;ter"
onclick="Main()">
        <input type="text" name="echo">
      </form>
    </body>
  </html>
```

Recopiez ce code et enregistrez-le sous "echoV2.htm".

Exécuter le programme

Voici un programme qui utilise une variable sans la déclarer. Vous remarquez qu'omettre une déclaration de variable n'apporte aucun message d'erreur puisque le programme tourne correctement. [Voir la page](#)

Faut-il vraiment déclarer une variable ?

La réponse est NON ... en JavaScript !

Lorsqu'on ne déclare pas une variable, la déclaration s'effectue d'elle-même. C'est ce que l'on appelle la déclaration implicite.

Voici encore une souplesse du JavaScript : la déclaration de variables n'est pas obligatoire.

Sachez que peu de langages acceptent cette méthode et que l'on vous conseille de l'éviter si vous souhaitez prendre de bonnes habitudes pour la suite ;-)

21. 3 : Quizz

1. En JavaScript, la déclaration d'une variable

- a) Est automatique
- b) Est nécessaire et obligatoire

2. 2. La déclaration implicite de variables existe

- a) Dans tous les langages
- b) Dans quelques langages

21. 4 : Texte, nombres, booléens, ...

Jusqu'à présent, vous avez rencontré un seul type de variable : le type texte (ou chaîne de caractères).

Ex : `var message="Bonjour tout le monde" ;`

Bien sûr, il n'y a pas que des variables de type texte, il existe d'autres types de variables :

- type numérique
- type booléen
- type tableau de données

Pour différencier le type d'une variable de type texte, il suffit "d'entourer" le contenu par des guillemets :

```
var message="Bonjour tout le monde" ;
```

Sans les guillemets, vous auriez obtenu une erreur...

Dans le module suivant, vous allez apprendre à déclarer et utiliser des variables de type **numérique**. Par la suite, vous apprendrez également à déclarer et à utiliser des **tableaux** de données.

21. 5 : Travailler sur du texte

"Ecrire un programme qui demande à l'utilisateur (via un formulaire) de proposer le nom d'un héros de BD qui a 6 lettres. Cette proposition est répétée dans un deuxième champ du formulaire."
(nom du fichier : tronquerTxt.htm)

Explication :

Peu importe la proposition de l'utilisateur, le programme n'affichera que les **6 premières lettres**.

Par exemple :

L'utilisateur propose capitaine Haddock.

Le programme répète : capita.

L'utilisateur propose spirou.

Le programme répète spirou.

=> Le programme doit tronquer la proposition de l'utilisateur en ne prenant que les 6 premières lettres.

Voici la commande (ligne 5) qui permet de tronquer une chaîne de caractères :

```
var prop;  
var prop6let;  
prop=document.BD.prop.value;  
// récupération de la proposition  
prop6let = prop.substr(0,6);  
// troncature de la variable prop
```

prop.substr(0,6) est la **méthode** qui permet de **tronquer** la chaîne de caractères contenue dans la variable **prop** en commençant **au début de la chaîne** (0) et en prenant **exactement 6 lettres** (6).

substr signifie *substring*, *sous-chaîne de caractères* ou *morceau de chaîne de caractères*.

Vous pouvez essayer de faire l'exercice seul. Voici le code du programme .

```
<html>  
  <head>  
    <title>Tronquer du texte</title>  
    <script>  
      var prop;  
      var prop6let;  
      function Main(){  
        prop = document.BD.prop.value; // récupération de la proposition  
        prop6let = prop.substr(0,6); // troncature de la variable prop  
        document.BD.echo.value = prop6let; // affichage de la variable  
prop6let  
      }  
    </script>  
  </head>  
  <body>
```

```
<form name="BD">
<h1>Proposez le nom d'un héros de BD contenant 6 lettres</h1>
<input type="text" name="prop">
  <input type="button" value="Répéter" onclick="Main();">
    <input type="Text" name="echo">
  </form>
</body>
</html>
```

Exécuter le code

Ce programme n'affiche que les six premiers caractères de la proposition introduite par l'utilisateur. [Voir la page](#)

21. 6 : Quizz

1. Soit la variable message contenant le texte "Bienvenue dans l'univers de la BD !". Laquelle de ces commandes récupère uniquement le mot "Bienvenue" ?

- a) message.substr(0,8)
- b) message.substr(0,9)
- c) message.substr(0,10)

2. Toujours avec la même variable message, quel mot récupère la commande message.substr(31,2) ? (Conseil : faites le test dans un petit programme !)

- a) BD
- b) BD !
- c) la

21. 7 : Encore du travail sur le texte !

Il existe une grande quantité de commandes permettant de travailler sur une chaîne de caractères. Voici quelques exemples de ce qu'il est possible de réaliser avec celles-ci.

Il existe encore d'autres méthodes permettant de travailler sur une chaîne de caractères. Vous trouverez la liste de ces méthodes dans une référence du langage JavaScript en regardant à l'objet String() (qui signifie "chaîne de caractères").

<http://devguru.com/content/technologies/javascript/home.html>

Aggrandir des caractères

Ce programme affiche une chaîne de caractères en plus grand. [Voir la page](#)

Mettre en majuscules

Ce programme affiche une chaîne de caractères en majuscules. [Voir la page](#)

Compter les caractères

Ce programme affiche la longueur d'une chaîne de caractères. [Voir la page](#)

Modifier une chaîne de caractères

Ce programme modifie le contenu d'une chaîne de caractères. [Voir la page](#)

Module 5 - Travailler avec des nombres

1 : Objectifs

Après avoir fait connaissance avec la notion de **variable**, vous allez travailler avec elles. Il existe plusieurs **types** de variables, ce module est consacré aux variables de **type numérique**.

2 : Des variables numériques

Jusqu'à présent, vous avez travaillé avec des variables contenant du texte. Vous allez maintenant travailler avec des variables contenant d'autres **types de données** : des nombres !

ex : compteur = 3 ;

Lorsque l'on ne met **pas** de guillemets, cela signifie qu'on a un nombre.

compteur = "3" est donc différent de **compteur = 3**. Dans le premier cas, vous avez le **texte 3** alors que dans le deuxième, vous avez le **nombre 3**.

Comment faire pour convertir le texte "3" en un nombre 3 ?

Attention ! Le contenu que l'on récupère d'un champ texte d'un formulaire est de type **texte**. Si l'on souhaite récupérer une valeur numérique, il faudra donc convertir le contenu.

Il existe une commande qui permet la **conversion** de contenu de **type textuel** en contenu de **type numérique** : **parseFloat()**.

Voici un exemple simple de l'utilisation de cette commande :

```
texte="3";
nombre=parseFloat(texte);
// la variable nombre a une valeur de 3
//cette valeur est de type numérique
document.write(nombre);
// la valeur de la variable nombre s'affiche : 3
```

Remarque : N'essayez pas de convertir le texte "Bonjour" en type numérique ! Voici la démonstration de cette erreur :

=> Lorsque le programme ne parvient pas à convertir du contenu textuel en nombre, il donne une valeur spéciale qui est **NaN** et qui signifie **Not A Number** (n'est pas un nombre).

parseFloat() est une commande qui permet de **convertir du contenu textuel en contenu numérique**.

3 : Des opérations mathématiques avec des variables

Avec ces variables de type **numérique**, on peut effectuer toute une série d'opérations mathématiques comme l'addition, la soustraction, la division et la multiplication.

4 : L'addition

Le signe utilisé pour additionner deux nombres est "+". Voici un exemple très simple d'addition de variables numériques :

```
var nombre1 = 123;  
var nombre2 = 321;  
var resultat = nombre1 + nombre2;  
alert(resultat);
```

5 : Appliquez : l'addition

"Créer un programme qui affiche la somme de deux nombres. Ces deux nombres sont insérés dans des champs de formulaire par l'utilisateur. Lorsque l'utilisateur clique sur le bouton "=", le programme affiche le résultat de l'addition dans un troisième champ du formulaire." (**nom du fichier : addition.htm**)

Les étapes du développement :

1. créer le formulaire HTML (avec 3 champs et 1 bouton)



2. effectuer l'addition
3. afficher la somme dans le 3ème champ du formulaire

Il y a toutefois une particularité à cet exercice : les valeurs insérées dans les **champs "texte"** sont de type **chaîne de caractères** (comme le nom l'indique : champ **texte**).

Il faudra donc utiliser la commande **parseFloat()** pour convertir le contenu du champ texte en nombre, comme ceci :

```
nombre1 = parseFloat(document.calculatrice.nombre1.value);
```

Vous pouvez essayer de faire l'exercice seul ou voir le développement du programme détaillé dans les pages suivantes.

6 : L'addition 1/4

Créons d'abord le formulaire HTML dont voici le code (lignes 6 à 8).

```
<html>
  <head>
    <title>Additionner des nombres</title>
  </head>
  <body>
    <form name="calculatrice">
      <input type="text" name="nombre1"> + <input type="text"
name="nombre2">
      <input type="button" value=" = ">
      <input type="text" name="resultat">
    </form>
  </body>
</html>
```

7 : L'addition 2/4

Récupérons ensuite les valeurs (lignes 4 à 16) des champs "nombre1" et "nombre2" (dans des variables du même nom) lors de l'événement **onclick** sur le bouton "=".

N'oublions pas de convertir les valeurs récupérées en nombre à l'aide de la commande `parseFloat()`.

```
<html>
  <head>
    <title>Additionner des nombres</title>
    <script>
      var nombre1; // déclaration de la variable nombre1
      var nombre2; // déclaration de la variable nombre2
      function Main(){
        // fonction principale appelée au chargement de la page
        nombre1 = parseFloat(document.calculatrice.nombre1.value);
        // récupération du premier nombre inséré dans le formulaire dans la
variable nombre1
        // et conversion en type numérique
        nombre2 = parseFloat(document.calculatrice.nombre2.value);
        // récupération du deuxième nombre inséré dans le formulaire dans
la variable nombre2
```

```

        // et conversion en type numérique
    }
</script>
</head>
<body>
<form name="calculatrice">
    <input type="text" name="nombre1"> + <input type="text" name="nombre2">
        <input type="button" value=" = ">
            <input type="text" name="resultat">
        </form>
    </body>
</html>

```

8 : L'addition 3/4

Calculons la somme des deux nombres (lignes 7,16 et 26).

```

<html>
  <head>
    <title>Additionner des nombres</title>
    <script>
      var nombre1; // déclaration de la variable nombre1
      var nombre2; // déclaration de la variable nombre2
      var resultat; // déclararation de la varaible resultat
      function Main(){
        // fonction principale appelée au chargement de la page
        nombre1 = parseFloat(document.calculatrice.nombre1.value);
        // récupération du premier nombre inséré dans le formulaire dans la
variable nombre1
        // et conversion en type numérique
        nombre2 = parseFloat(document.calculatrice.nombre2.value);
        // récupération du deuxième nombre inséré dans le formulaire dans
la variable nombre2
        // et conversion en type numérique
        resultat = nombre1 + nombre2; // calcul de la somme des deux
nombres
      }
    </script>
  </head>
  <body>
    <form name="calculatrice">
      <input type="text" name="nombre1"> + <input type="text" name="nombre2">
        <input type="button" value=" = ">
          <input type="text" name="resultat">
        </form>
    </body>

```

</html>

9 : L'addition 4/4

Affichons le résultat dans le champ adéquat ((lignes 17 et 18) et ajoutons une instruction **onclick** sur le bouton qui chargera la fonction **Main()** (ligne 26).

```
<html>
  <head>
    <title>Additionner des nombres</title>
    <script>
      var nombre1; // déclaration de la variable nombre1
      var nombre2; // déclaration de la variable nombre2
      var resultat; // déclararation de la varaible resultat
      function Main(){
        // fonction principale appelée au chargement de la page
        nombre1 = parseFloat(document.calculatrice.nombre1.value);
        // récupération du premier nombre inséré dans le formulaire dans la
variable nombre1
        // et conversion en type numérique
        nombre2 = parseFloat(document.calculatrice.nombre2.value);
        // récupération du deuxième nombre inséré dans le formulaire dans
la variable nombre2
        // et conversion en type numérique
        resultat = nombre1 + nombre2; // calcul de la somme des deux
nombres
        document.calculatrice.resultat.value = resultat ;
        // affichage du résultat dans le champ "resultat"
      }
    </script>
  </head>
  <body>
    <form name="calculatrice">
      <input type="text" name="nombre1"> + <input type="text" name="nombre2">
      // ajout de onclick="Main();" sur le bouton
      <input type="button" value=" = " onclick="Main();">
      <input type="text" name="resultat">
    </form>
  </body>
</html>
```

Il ne vous reste plus qu'à enregistrer le code sous "addition.htm" et à l'exécuter.

Voir le résultat

Ouvrez cette page pour voir le résultat attendu. [Voir la page](#)

10 : Quizz

1. La commande permettant de convertir du texte en nombre est

- a) int()
- b) var
- c) parseFloat()

2. La valeur récupérée d'un champ de formulaire est de type

- a) numérique
- b) texte
- c) variable

3. Pour additionner deux nombres, on utilise l'opérateur

- a) Add
- b) *
- c) +

11 : Soustraction

Le signe utilisé pour soustraire deux nombres est "-". Voici un exemple très simple de soustraction de variables numériques :

```
var nombre1 = 123;
var nombre2 = 321;
var resultat = nombre2-nombre1;
alert(resultat);
```

12 : Appliquez : la soustraction

"Ce programme effectue l'addition de deux nombres insérés dans un champ texte, modifiez-le pour qu'il effectue la différence entre les deux nombres."

```
<html>
\\r\\n\\t<head>
\\r\\n\\t\\t<title>Addition / soustraction</title>
\\r\\n\\t\\t*script=
\\r\\n\\t\\t\\tvar nombre1;
\\r\\n\\t\\t\\tvar nombre2;
\\r\\n\\t\\t\\tvar resultat;
\\r\\n\\t\\t\\tfunction Main(){
\\r\\n\\t\\t\\t\\tnombre1 = parseFloat(document.calculatrice.nombre1.value);
\\r\\n\\t\\t\\t\\tnombre2 = parseFloat(document.calculatrice.nombre2.value);
```

```

\\r\\n\\t\\t\\t\\tresultat = nombre1 + nombre2;
\\r\\n\\t\\t\\t\\tdocument.calculatrice.resultat.value = resultat ;
\\r\\n\\t\\t\\t\\t}
\\r\\n\\t\\t=script*
\\r\\n\\t</head>
\\r\\n\\t<body>
\\r\\n\\t\\t<form name="calculatrice">
\\r\\n\\t\\t\\t<input type="Text" name="nombre1"> + <input type="Text" name="nombre2"><input
type="Button" value=" = " onclick="Main();"> <input type="Text" name="resultat">
\\r\\n\\t\\t</form>
\\r\\n\\t</body>
\\r\\n</html> <html>
\\r\\n\\t<head>
\\r\\n\\t\\t<title>Addition / soustraction</title>
\\r\\n\\t\\t*script=
\\r\\n\\t\\t\\tvar nombre1;
\\r\\n\\t\\t\\tvar nombre2;
\\r\\n\\t\\t\\tvar resultat;
\\r\\n\\t\\t\\tfunction Main(){
\\r\\n\\t\\t\\t\\tnombre1 = parseFloat(document.calculatrice.nombre1.value);
\\r\\n\\t\\t\\t\\tnombre2 = parseFloat(document.calculatrice.nombre2.value);
\\r\\n\\t\\t\\t\\tresultat = nombre1 - nombre2;
\\r\\n\\t\\t\\t\\tdocument.calculatrice.resultat.value = resultat ;
\\r\\n\\t\\t\\t\\t}
\\r\\n\\t\\t=script*
\\r\\n\\t</head>
\\r\\n\\t<body>
\\r\\n\\t\\t<form name="calculatrice">
\\r\\n\\t\\t\\t<input type="Text" name="nombre1"> - <input type="Text" name="nombre2"><input
type="Button" value=" = " onclick="Main();"> <input type="Text" name="resultat">
\\r\\n\\t\\t</form>
\\r\\n\\t</body>
\\r\\n</html>

```

Exécuter le code

Si votre programme ne tourne pas correctement, vous pouvez voir la solution en ligne. [Voir la page](#)

13 : Multiplication

Le signe utilisé pour multiplier deux nombres est "*". Voici un exemple très simple de multiplication de variables numériques :

```

var prixHTVA = 999;
var Taux = 1,21;
var prixTVAC=prixHTVA*Taux;
alert(prixTVAC);

```

14 : Division

Le signe utilisé pour diviser un nombre est "/". Voici un exemple très simple de division d'une variable numérique par une autre variable numérique :

```
var nombre1 = 2500;
var nombre2 = 5;
var resultat = nombre1/nombre2;
alert(resultat) ;
```

15 : Appliquez : la calculatrice

"Créer une calculatrice : Deux nombres sont insérés dans des champs de formulaire par l'utilisateur. Lorsque l'utilisateur clique sur le bouton "+", "-", "*" ou "/", le programme affiche le résultat de l'opération dans un troisième champ du formulaire." (nom du fichier : [calculatrice.htm](#))

Voici à quoi le formulaire de la calculatrice devrait ressembler :



The diagram shows a horizontal layout for a calculator. It starts with a rectangular input field, followed by four square buttons containing the operators '+', '-', '*', and '/'. After these buttons is another rectangular input field, followed by an equals sign '=', and finally a third rectangular field for the result.

Essayez de créer ce programme. Pour plus de détails sur les étapes de développement ou pour voir la solution de l'exercice, passez à la page suivante.

16 : La calculatrice, les étapes du programme

Voici quelques précisions sur le développement du programme :

Attention ! Dans ce programme, il y a **4 actions** pour **4 événements** :

- clic sur le bouton " + " : action **additionner**
- clic sur le bouton " - " : action **soustraire**
- clic sur le bouton " * " : action **multiplier**
- clic sur le bouton " / " : action **diviser**

Vous allez donc devoir créer **4 fonctions** :

```
function Addition(){
    nombre1=document.formulaire.nombre1.value;
    nombre2=document.formulaire.nombre2.value;
    resultat=nombre1+nombre2;
    document.formulaire.resultat.value = resultat;
}
```



```
function Soustraction(){
    nombre1=document.formulaire.nombre1.value;
    nombre2=document.formulaire.nombre2.value;
    resultat = nombre1 - nombre2;
    document.formulaire.resultat.value = resultat;
}
```

Chaque bouton fera appel à la fonction adéquate :

```
<input type="button" value="+" onclick="Addition()">
  <input type="button" value="-" onclick="Soustraction()">
```

Remarque : vous aurez l'occasion d'approfondir la notion de **fonction** dans le module 9.

Voici le code de l'application "calculatrice".

```
<html>
  <head>
    <title>Créer une calculatrice</title>
    <script>
      var nombre1;
      var nombre2;
      var resultat;
      function Addition(){
        nombre1 = parseFloat(document.calculatrice.nombre1.value);
        // récupération du premier nombre inséré dans le formulaire dans la
variable nombre1
        // et conversion en type numérique
        nombre2 = parseFloat(document.calculatrice.nombre2.value);
        // récupération du deuxième nombre inséré dans le formulaire dans la
variable nombre2
        // et conversion en type numérique
        resultat = nombre1 + nombre2;
        // calcul de la somme des deux nombres
        document.calculatrice.resultat.value = resultat;
        // affichage du résultat dans le champs "resultat"
      }
      function Soustraction(){
        nombre1 = parseFloat(document.calculatrice.nombre1.value);
        // récupération du premier nombre inséré dans le formulaire dans la
variable nombre1
        // et conversion en type numérique
        nombre2 = parseFloat(document.calculatrice.nombre2.value);
        // récupération du deuxième nombre inséré dans le formulaire dans la
variable nombre2
        // et conversion en type numérique
        resultat = nombre1 - nombre2;
```

```

        // calcul de la différence des deux nombres
        document.calculatrice.resultat.value = resultat ;
        // affichage du résultat dans le champs "resultat"
    }
    function Multiplication(){
        nombre1 = parseFloat(document.calculatrice.nombre1.value);
        // récupération du premier nombre inséré dans le formulaire dans la
variable nombre1
        // et conversion en type numérique
        nombre2 = parseFloat(document.calculatrice.nombre2.value);
        // récupération du deuxième nombre inséré dans le formulaire dans la
variable nombre2
        // et conversion en type numérique
        resultat=nombre1*nombre2;
        // calcul de le produit des deux nombres
        document.calculatrice.resultat.value=resultat;
        // affichage du résultat dans le champs "resultat"
    }
    function Division(){
        nombre1 = parseFloat(document.calculatrice.nombre1.value);
        // récupération du premier nombre inséré dans le formulaire dans la
variable nombre1
        // et conversion en type numérique
        nombre2 = parseFloat(document.calculatrice.nombre2.value);
        // récupération du deuxième nombre inséré dans le formulaire dans la
variable nombre2
        // et conversion en type numérique
        resultat=nombre1/nombre2;
        // calcul du quotient des deux nombres
        document.calculatrice.resultat.value=resultat;
        // affichage du résultat dans le champ "resultat"
    }
}
</script>
</head>
<body>
<form name="calculatrice">
<input type="text" name="nombre1"> <input type="button" value="
+ " onclick="Addition();"> <input type="button" value=" - "
onclick="Soustraction();"> <input type="button" value=" * "
onclick="Multiplication();"> <input type="button" value=" / "
onclick="Division();"> <input type="text" name="nombre2"> = <input
type="text" name="resultat">
</form>
</body>
</html>

```

Exécuter le programme.

Il vous suffit d'insérer deux nombres et de cliquer sur l'opération souhaitée. [Voir la page](#)

17 : Conclusion

1. Quelle commande permet de convertir du contenu de type textuel en contenu de type numérique ?

- a) parseFloat()
- b) prompt()
- c) Aucune, la conversion est toujours automatique

2. Sachant que a = 3 et b = 5, laquelle de ces instructions représente l'addition de a et b ?

- a) $a + b = \text{resultat}$;
- b) $\text{resultat} = a + b$;
- c) $\text{resultat} = "a + b"$;

3. Trouvez l'instruction qui convertit correctement des EUR en BEF:

- a) $\text{eur} = \text{bef} / 40,3399$;
- b) $\text{eur} = \text{bef} * 40,3399$;
- c) $\text{bef} = \text{eur} / 40,3399$;

18 : Mémo

Les variables numériques

Jusqu'ici, nous avons travaillé avec des variables de type textuelles:

```
nom = "Capitaine Haddock" ;
```

Nous allons aborder les variables numériques:

```
Pointure = 45 ;
```

Des opérations mathématiques

Le signe utilisé pour **additionner** deux nombres est "+"

```
var resultat = nombre1 + nombre2 ;
```

Le signe utilisé pour **soustraire** deux nombres est "-"

```
var resultat = nombre2 - nombre1 ;
```

Le signe utilisé pour **multiplier** deux nombres est "*"

```
var prixTVAC = prixHTVA * 1,21;
```

Le signe utilisé pour **diviser** un nombre est "/"

```
var resultat = nombre1 / nombre2 ;
```

Passer du textuel au numérique

Si vous avez récupéré une **valeur textuelle** et que vous désirez faire une opération arithmétique dessus, il vous faut la **convertir en numérique**.

Cette opération s'appelle "**parser**" une valeur grâce à la commande `parseFloat()`;

```
texte = "3" ; // les guillemets indiquent une valeur "String"
```

```
nombre = parseFloat(texte) ; // on insère dans la variable nombre le contenu de la variable texte transformé en nombre
```

Remarque : N'essayez pas de convertir le texte "Bonjour" en type numérique, vous provoqueriez une erreur.

Le signe + et ses différentes utilisations

Nous avons vu dans le chapitre précédent que le signe "+" permettait de **concaténer deux chaînes de caractères**, et dans celui-ci qu'il permettait d'**additionner deux valeurs numériques**.

Faites donc bien attention au **type de variable employé**, ainsi qu'à leur **contenu**. Par défaut, ce qui est récupéré dans un champ de formulaire est considéré **comme du texte**.

Ainsi, cet exemple vous montrera l'utilisation du "+" dans les deux contextes:

```
nombre1="123"; // contenu textuel
nombre2="321" ; // contenu textuel
resultat1=nombre1+nombre2;
// resultat1 vaut "123321" car 123 et 321 sont considérés comme du texte
// on a concaténé le texte 123 au texte 321
resultat2 = parseFloat(nombre1) + parseFloat(nombre2) ;
// resultat2 vaut 444 car 123 et 321 sont des nombres, on a additionné les
deux nombres
```

La conversion implicite

Si vous concaténez un **nombre** et une **chaîne de caractères**, et que vous placez le résultat dans une variable, une **conversion implicite** s'exécute.

Le programme **convertit** lui-même la variable de type numérique en type texte.

```
var nombre;
var texte;
var resultat;
nombre=3; // variable de type numérique
texte = "Nombre de BD gagnées : " ; // variable de type texte
resultat = texte + nombre ; // resultat sera de type texte
```

Le contraire est également vrai, on peut **convertir implicitement** une chaîne de caractères en nombre:

```
var nombre;
var texte;
var resultat;
nombre=100; // variable de type numérique
texte = "40.3399" ; // variable de type texte
resultat = texte * nombre ; // resultat sera de type numérique
```

Bien entendu, il n'y a **pas de conversion explicite** avec le signe "+" qui fera plutôt une concaténation.

Connaître le type d'une variable

Il peut être utile de vérifier un type de valeur entrée par un visiteur, afin par exemple de contrôler la valeur saisie dans un champ.

Nous utiliserons dès lors l'opérateur **typeof** qui affiche le type d'une variable:

```
- var num ;  
- num = "3";  
- alert(typeof num); // la variable num est de type TEXTE
```

19.1 : Compléments

En complément, vous découvrirez plus d'informations sur les types de variables ainsi que deux exercices pratiques destinés à approfondir les notions vues jusqu'ici.

19.2 : L'opérateur +

L'opérateur "+" est utilisé dans 2 opérations différentes :

- la concaténation de chaînes de caractères
- l'addition de nombres

Comment savoir quelle opération va être effectuée ?

- la concaténation : variables de type **texte**
- l'addition : variables de type **numérique**

Pour être certain d'effectuer une addition de deux nombres, il suffit de convertir les variables en numérique à l'aide de la commande `parseFloat()`.

19.3 : Utilisation de la commande `parseFloat()`

Voici un exemple de l'utilisation de cette commande :

```
nombre1="123"; // contenu textuel
nombre2="321" ; // contenu textuel
resultat1=nombre1+nombre2;
// resultat1 vaut "123321" car 123 et 321 sont considérés comme du texte
// on a concaténé le texte 123 au texte 321
resultat2=parseFloat(nombre1)+parseFloat(nombre2);
// resultat2 vaut 444 car 123 et 321 sont des nombres, on a additionné les
deux nombres
```

19.4 : Convertir un nombre en une chaîne de caractères

Voici un exemple de concaténation de chaînes de caractères un peu particulier : vous allez **concaténer** un **nombre** et une **chaîne de caractères**...

```
var nombre;
var texte;
var resultat;
nombre=3; // variable de type numérique
```

```
texte="Nombre de BD gagnées : " ; // variable de type texte
resultat=texte+nombre ;
```

1. A votre avis, le résultat est de type

- a) Texte
- b) Numérique
- c) On obtient une erreur

Lorsque le programme effectue l'opération "+", il **convertit** lui-même la variable de type **numérique** en type **texte**. C'est ce que l'on appelle la **conversion implicite**.

La variable **resultat** est de type **texte** et sa valeur est "Nombre de BD gagnées : 3"

Mémo :

[texte] + [texte] = **concaténation**

[nombre] + [nombre] = **addition**

[nombre] + [texte] = **concaténation** avec **conversion implicite** du nombre en texte

19.5 : Convertir une chaîne de caractères en un nombre

Vous avez vu que le JavaScript peut convertir automatiquement un nombre en texte grâce à la **conversion implicite**. Voici un exemple provoquant l'effet inverse : une conversion implicite de texte en nombre.

```
var nombre;
var texte;
var resultat;
nombre = 100; // variable de type numérique
texte = "40.3399" ; // variable de type texte
resultat = texte * nombre ;
```

1. A votre avis, le résultat est de type

- a) Texte
- b) Numérique
- c) On obtient une erreur

Lorsque le programme effectue l'opération "*", il **convertit** lui-même la variable de type **texte** en type **numérique**. Il y a là conversion implicite d'un texte en nombre.

La variable **resultat** est de type **numérique** et sa valeur est **4033.99**.

Mémo :

Pour les opérations mathématiques (sauf addition, "+"):

[nombre] * [nombre] = opération mathématique

[nombre] * [texte] = opération mathématique avec conversion implicite du texte en nombre

Remarque : si le texte ne peut pas être converti, vous obtiendrez la valeur NaN, Not A Number (pas un nombre)

19. 6 : Type texte ou type numérique ?

Vous n'êtes plus certain du type d'une variable ? L'opérateur typeof vous affiche le type d'une variable :

```
var num ;  
num = "3";  
alert(typeof num); // la variable num est de type TEXTE  
num = parseFloat(num);  
alert(typeof num); // la variable num est de type NUMERIQUE
```

L'opérateur typeof peut retourner les types suivants :

number: type numérique

string: type texte

boolean: type booléen (vrai ou faux)

object: objet

function: fonction

undefined: déclaré mais non défini (pas de valeur assignée)

19. 7 : Quizz

1. Soit le code suivant : `var num ; alert(typeof num);` Quelle est la valeur retournée par l'opérateur `typeof` ?

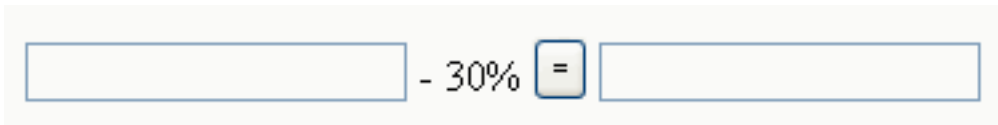
- a) number
- b) string
- c) undefined

19. 8 : Exercice de synthèse

"Ecrire un programme qui calcule la réduction du prix d'une BD (-30%). L'utilisateur encode le prix d'une BD dans le champ adéquat et clique sur le bouton "=" pour calculer la réduction" (**nom du fichier : `reducBD.htm`**)

Vous devrez donc :

1. créer un formulaire similaire à celui-ci :



The image shows a form layout on a light gray background. It consists of a rectangular text input field, followed by the text "- 30%", then a small square button with an equals sign "=", and finally another rectangular text input field.

2. récupérer la valeur du champ "prix"
3. calculer la réduction
4. afficher le prix réduit dans le champ "prixreduit"

Remarque : $\text{prixBD} - 30\%(\text{prixBD})$ est équivalent à $\text{prixBD} * 0.7$

Excuter le programme

L'utilisateur insère un prix dans le premier champ texte et clique sur le bouton "=". Le résultat est affiché dans un deuxième champ du formulaire. Consultez le code source pour voir le code. [Voir la page](#)

19. 9 : Quizz

1. Pour récupérer la valeur insérée dans le champ "prix" du formulaire "formBD", la commande est :

- a) `Document.formBD.prix.value`
- b) `document.formBD.prix`
- c) `document.formBD.prix.value`

Module 6 - Ajouter une condition

1 : Objectifs

Ce module montre comment gérer des **conditions** et **cas particuliers** d'un programme. Vous apprendrez à **tester**, à **comparer** certaines informations avant d'effectuer des instructions ce qui, par exemple, vous évitera d'effectuer des **traitements inutiles** ou d'obtenir des **erreurs**.

2 : Une condition dans l'énoncé

Reprenons le programme login.htm (cf. module 2) et améliorons l'énoncé :

"Ecrire un programme qui récupère un mot de passe dans un formulaire de connexion sécurisée. **Si** le mot de passe est égal à "bachibouzouk", une alerte prévient l'utilisateur qu'il a l'autorisation d'entrer. **Sinon**, une alerte prévient que l'utilisateur n'a pas l'autorisation d'entrer." (**nom du fichier : Securite2.htm**)

Les mots importants de cet énoncé sont en rouge : **Si, Sinon**.

Ces mots marquent clairement **une condition de traitement**.

Cela signifie que le programme va effectuer **l'un ou l'autre** traitement en fonction d'une **condition** :

La condition :

Le mot de passe donné par l'utilisateur **est égal à** "Bachibouzouk"

Si la condition est vraie (traitement A) :

L'utilisateur peut entrer

Sinon (traitement B):

L'utilisateur ne peut pas entrer

3 : Une structure conditionnelle

Pour traduire cette étape en langage de programmation, on respecte une syntaxe prédéfinie appelée **structure conditionnelle**.

Voici comment on représente (en JavaScript) une structure conditionnelle :

```
if (condition){  
    // traitement A  
}else{  
    // traitement B  
}
```

Remarque : Pour éviter les erreurs de syntaxe (oubli de "}", par exemple), on **indente** correctement le code pour qu'il soit clair. Prenez la bonne habitude d'ajouter directement une accolade fermante

lorsque vous venez d'insérer une accolade ouvrante. Comme cela, vous n'oublierez jamais d'accolade !

Le "if" et le "else" doivent impérativement être écrits en **minuscules**.

4 : Le code du programme

Voici le code du programme qui récupère le mot de passe inséré par l'utilisateur dans le formulaire de connexion.

Commençons d'abord par récupérer le mot de passe :

```
<html>
  <head>
    <title>
      Ecrire un programme qui récupère un mot de passe dans un
      formulaire
    </title>
    <script>
      var motpropose;
      function Main(){
        motpropose = document.securite.motdepasse.value;
        // on récupère la valeur du champ texte dans une variable
      }
    </script>
  </head>
  <body>
    <h3>Zone Sécurisée</h3>
    <p>Veuillez entrer votre mot de passe:</p>
    <form name="securite">
      <input type="text" name="motdepasse">
      <input type="button" value="ENTRER" onclick="Main()">
    </form>
  </body>
</html>
```

Le mot de passe inséré par l'utilisateur est récupéré dans la variable **motpropose**.
Il ne reste plus qu'à comparer la valeur de **motpropose** à "bachibouzouk" :

```
if(motpropose=="bachibouzouk"){ // si le mot proposé est égal à
  "bachibouzouk"
  alert("Vous avez le droit d'entrer!");
  // alors on affiche le texte "Vous avez le droit d'entrer !"
}else{ // sinon
  alert("Vous n'avez PAS le droit d'entrer");
  // on affiche le texte "Vous n'avez PAS le droit d'entrer"
```

```
}
```

Pour **comparer l'égalité** de **motpropose** et "bachibouzouk", on a utilisé un **double égal "=="** et non un simple "**=**". Lorsque l'on veut comparer l'égalité de deux éléments, on utilise toujours le "**==**". **N'oubliez pas que "**=**" signifie que l'on assigne une valeur à une variable.**

Rappel : N'oubliez pas que le langage JavaScript est **sensible à la casse**. La valeur "**bachibouzouk**" en minuscules est donc **différente** de la valeur "**BACHIBOUZOUK**" en majuscules !

Voilà! Il ne vous reste plus qu'à insérer ce morceau de code dans votre programme et à l'enregistrer sous "securite2.htm".

Exécuter le programme

Lorsque l'utilisateur insère le mot de passe adéquat, il obtient le message "Vous avez le droit d'entrer !". [Voir la page](#)

5 : Conclusion

Dans un programme, il peut y avoir deux traitements différents en fonction d'une **condition**. Si la condition est VRAIE, on effectue un traitement A ; sinon (la condition est FAUSSE), on effectue un traitement B.

Voici le "squelette" de la structure conditionnelle "if" :

```
if(condition){  
    // traitement A  
}else{  
    // traitement B  
}
```

Pour comparer l'égalité de 2 valeurs, on utilise "**==**" et non "**=**" !!! Faites bien attention à ne pas vous tromper !

6 : Exercice : une condition

Reprenez votre programme demanderForm.htm (voir l'exercice du module 2.11). Ce programme demande une injure du capitaine Haddock. Améliorez ce programme pour qu'il vérifie si le champ n'est pas vide. Si l'injure est égale à "" (rien), avertir l'utilisateur qu'il n'a rien inséré avec le message "Veuillez proposer une injure du capitaine Haddock.", sinon afficher le message "Le capitaine Haddock a dit [INJURE]." (**nom du fichier : injureV2.htm**)

Exécuter le programme

Si l'utilisateur ne donne aucune proposition, un message d'erreur apparaît. Par contre, s'il propose une injure, celle-ci s'affichera dans un deuxième champ du formulaire. Vous remarquez que la structure conditionnelle est utilisée ici pour la gestion d'**erreurs** dans le programme. [Voir la page](#)

7 : Comparer 2 valeurs

Jusqu'à présent, vous avez comparé l'**égalité** de deux valeurs (motproposé == "bachibouzouk"). Il est également possible d'effectuer d'autres opérations de comparaison. On les utilise principalement pour **comparer des valeurs numériques** :

Valeur1 "**plus petit que**" valeur2 :
`nombre < 100`

Valeur1 "**plus grand que**" valeur2 :
`nombre > 100`

Valeur1 "**plus petit ou égal**" valeur2 :
`nombre <= 100`

Valeur1 "**plus grand ou égal**" valeur2 :
`nombre >= 100`

Valeur1 "**différent de**" valeur2 :
`nombre != 100`

8 : Quizz

1. if(prop <= 100){...}Dans cette structure conditionnelle, la condition est

- a) "Si la variable prop est plus petite que 100"
- b) "Si la variable prop est plus grande ou égale à 100"
- c) "Si la variable prop est plus petite ou égale à 100"

9 : Appliquez : comparer des valeurs numériques

"Ecrire un programme qui demande (dans un champ de formulaire) le nombre de BD de Spirou et Fantasio que vous avez dans votre bibliothèque. Si le nombre est plus grand que 45, on obtient l'alerte suivante : "Vous avez presque toute la collection !". Sinon, on obtient l'alerte : "Il vous manque encore quelques tomes avant que la collection soit complète"."

```
<html>
\\r\\n\\t<head>
\\r\\n\\t\\t<title>Collection</title>
\\r\\n\\t\\t*script=
\\r\\n\\t\\t\\tvar total;
\\r\\n\\t\\t\\tfunction Main(){
\\r\\n\\t\\t\\t\\ttotal = parseFloat(document.BD.spirou.value);
```

```

\\r\\n\\t\\t}
\\r\\n\\t\\t=script*
\\r\\n\\t</head>
\\r\\n\\t<body>
\\r\\n\\t\\t<h3>Spirou et Fantasio</h3>
\\r\\n\\t\\t<p>Combien de BD de Spirou & Fantasio avez-vous?</p>
\\r\\n\\t\\t<form name="BD">
\\r\\n\\t\\t\\t<input type="Text" name="spirou">
\\r\\n\\t\\t\\t<input type="Button" value="ENREGISTRER" onclick="Main();">
\\r\\n\\t\\t</form>
\\r\\n\\t</body>
\\r\\n</html> <html>
\\r\\n\\t<head>
\\r\\n\\t\\t<title>Collection</title>
\\r\\n\\t\\t*script=
\\r\\n\\t\\t\\tvar total;
\\r\\n\\t\\t\\tfunction Main(){
\\r\\n\\t\\t\\t\\ttotal = parseFloat(document.BD.spirou.value);
\\r\\n\\t\\t\\t\\tif(total > 45){
\\r\\n\\t\\t\\t\\t\\talert("Vous avez presque toute la collection");
\\r\\n\\t\\t\\t\\t}else{
\\r\\n\\t\\t\\t\\t\\talert("Il vous manque encore quelques tomes avant que la collection soit complète");
\\r\\n\\t\\t\\t\\t}
\\r\\n\\t\\t\\t}
\\r\\n\\t\\t=script*
\\r\\n\\t</head>
\\r\\n\\t<body>
\\r\\n\\t\\t<h3>Spirou et Fantasio</h3>
\\r\\n\\t\\t<p>Combien de BD de Spirou & Fantasio avez-vous?</p>
\\r\\n\\t\\t<form name="BD">
\\r\\n\\t\\t\\t<input type="Text" name="spirou">
\\r\\n\\t\\t\\t<input type="Button" value="ENREGISTRER" onclick="Main();">
\\r\\n\\t\\t</form>
\\r\\n\\t</body>
\\r\\n</html>

```

Exécuter le programme

L'utilisateur reçoit deux messages différents en fonction de la valeur insérée dans le formulaire.

[Voir la page](#)

10 : Travailler avec plusieurs conditions

Il est également possible d'avoir plus d'une condition dans un programme.

Voici l'énoncé d'un programme de jeu:

"L'ordinateur choisit un nombre au hasard entre 0 et 100. L'utilisateur doit le deviner et proposer un nombre. Le programme vérifie si la proposition de l'utilisateur est égale à son nombre... Si c'est le cas, l'utilisateur a gagné ! Si la proposition est plus petite que le nombre, on aide l'utilisateur

avec l'indice "plus grand". Si la proposition est plus grande que le nombre, on aide l'utilisateur avec l'indice "plus petit". (nom du fichier : jeu.htm)

Remarquez que l'on retrouve 3 "si" dans l'énoncé, il y a donc **plusieurs conditions et plusieurs traitements** !

Voici la syntaxe d'une structure conditionnelle à plusieurs conditions :

```
if (condition1){
    // traitement A
}else if(condition2){
    // traitement B
}else{
    // traitement C
}
```

Remarque : Vous pouvez ajouter autant de blocs "else if(...){...}" que vous le souhaitez.

A la page suivante, nous vous expliquons comment choisir un nombre au hasard entre 0 et 100.

11 : Un nombre au hasard entre 0 et 100

Voici l'instruction qui permet de choisir un nombre au hasard entre 0 et 100 :

```
var nombre;
nombre=Math.round(Math.random()*100);
```

Math.random() est une commande qui prend un nombre décimal au hasard entre 0 et 1 :
0.7813682387203784

Ensuite, on multiplie ce nombre par 100 :
78,13682387203784

Et on arrondit ce résultat à l'aide de la commande Math.round() :
78

Avec cette méthode, on obtiendra toujours un nombre entre 0 et 100. C'est une simple formule mathématique, ne vous inquiétez pas si vous ne vous sentez pas à l'aise devant cette instruction. Courage !

Vous avez maintenant toutes les clés en main pour développer le programme. Essayez de créer ce programme de jeu, le détail de la solution se trouve à la page suivante.

12 : La solution de l'exercice

Plusieurs conditions dans une structure conditionnelle

Voici le code de la structure conditionnelle qui compare la valeur de la proposition à la valeur du nombre choisi au hasard :

```
if(proposition==nombre){
    // si la proposition est égale au nombre, c'est gagné !
    alert("gagné !");
}else if(proposition < nombre){
    // si la proposition est inférieure au nombre, l'indice est : " plus
grand ".
    alert("Le nombre à trouver est plus grand !");
}else{
    // si la proposition n'est ni égale, ni inférieure,
    // c'est qu'elle est supérieure au nombre.
    // Dans ce cas, l'indice est : " plus petit ".
    alert("Le nombre à trouver est plus petit ! ");
}
```

Mettons tous ces morceaux de codes bout à bout pour former le programme :

```
<html>
  <head>
    <title>
      Le jeu du bon numéro
    </title>
    <script>
      var nombre;
      var proposition;
      // on choisit le nombre au hasard en début de programme
      nombre = Math.round(Math.random()*100);
      function Main(){
        proposition=parseFloat(document.jeu.prop.value);
        if(proposition==nombre){
          // si la proposition est égale au nombre, c'est gagné !
          alert("gagné !") ;
        }else if(proposition<nombre){
          // si la proposition est inférieure au nombre, l'indice
est : "plus grand".
          alert("Le nombre à trouver est plus grand !");
        }else{
          // si la proposition n'est ni égale, ni inférieure,
          // c'est qu'elle est supérieure au nombre.
          // Dans ce cas, l'indice est : "plus petit".
          alert("Le nombre à trouver est plus petit !");
        }
      }
    </script>
  </head>
</html>
```

```

    }
  }
</script>
</head>
<body>
<h1>Le jeu du bon numéro</h1>
<form name="jeu">
  <input type="text" name="prop">
    <input type="button" value="verifier!" onclick="Main()">
  </form>
</body>
</html>

```

Enregistrez ce code sous "jeu.htm" et exécutez-le.

Résultat attendu

Voyons le résultat :

1. L'utilisateur introduit une proposition
- 2a. La proposition est égale au nombre choisi au hasard. Le message "gagné !" s'affiche.
- 2b. La proposition est inférieure au nombre choisi au hasard. Le message "plus grand !" s'affiche.
- 2c. La proposition est supérieure au nombre choisi au hasard. Le message "plus petit !" s'affiche.
3. On recommence jusqu'à obtenir le message " gagné ! ". [Voir la page](#)

13 : Conclusion

Dans un programme, il peut y avoir plusieurs traitements différents en fonction de plusieurs conditions. Si la condition 1 est VRAIE, on effectue le traitement A, si la condition 2 est VRAIE, on effectue le traitement B, si la condition 3 est VRAIE, on effectue le traitement C, ..., sinon on effectue un traitement X.

Voici le "squelette" de la structure conditionnelle "if" avec plusieurs conditions:

```

if (condition1 ){
  traitement A
}
else if (condition2 ){
  traitement B
}
else if (condition3 ){
  traitement C
}
...
}else {
  traitement X
}

```

Remarque : On peut insérer autant de blocs "else if" que l'on veut, mais il faut éviter d'avoir trop de conditions différentes car cela complique le code...

Les structures conditionnelles sont souvent utilisées pour éviter des erreurs dans le programme :

- **Si** l'utilisateur n'a pas encodé de valeur
- **Si** la valeur encodée n'est pas correcte
- ...

14 : Appliquez : plusieurs conditions

Créez un programme qui demande à l'utilisateur la quantité de chocolats mangée chaque jour.

SI la réponse est "égal à 0" => alerter "Vous devriez ! Le chocolat est bon pour la santé !"

SI la réponse est "plus petit que 5" => alerter "Vive le chocolat !"

SINON => afficher le message : "La gourmandise est un vilain défaut !"

```
<html>
\\r\\n\\t<head>
\\r\\n\\t\\t<title>Aimez-vous le chocolat?</title>
\\r\\n\\t\\t*script=
\\r\\n\\t\\t\\tvar reponse;
\\r\\n\\t\\t\\tfunction Main(){
\\r\\n\\t\\t\\t\\treponse = parseFloat(document.choco.reponse.value);
\\r\\n\\t\\t\\t}
\\r\\n\\t\\t=script*
\\r\\n\\t</head>
\\r\\n\\t<body>
\\r\\n\\t\\t<form name="choco">
\\r\\n\\t\\t\\t<h3>Combien de fois par jour mangez-vous du chocolat ?</h3>
\\r\\n\\t\\t\\t<input type="text" name="reponse"> fois<br>
\\r\\n\\t\\t\\t<input type="button" value="Enregistrer la réponse" onClick="Main();">
\\r\\n\\t\\t</form>
\\r\\n\\t</body>
\\r\\n</html> <html>
\\r\\n\\t<head>
\\r\\n\\t\\t<title>Aimez-vous le chocolat?</title>
\\r\\n\\t\\t*script=
\\r\\n\\t\\t\\tvar reponse;
\\r\\n\\t\\t\\tfunction Main(){
\\r\\n\\t\\t\\t\\treponse = parseFloat(document.choco.reponse.value);
\\r\\n\\t\\t\\t\\tif(reponse == 0){
\\r\\n\\t\\t\\t\\t\\talert("Vous devriez en manger ! Le chocolat est bon pour la santé!");
\\r\\n\\t\\t\\t\\t} else if(reponse < 5){
\\r\\n\\t\\t\\t\\t\\talert("Vive le chocolat !");
\\r\\n\\t\\t\\t\\t} else{
\\r\\n\\t\\t\\t\\t\\t\\talert("La gourmandise est un vilain défaut!");
\\r\\n\\t\\t\\t\\t\\t}
\\r\\n\\t\\t\\t\\t}
\\r\\n\\t\\t\\t=script*
\\r\\n\\t\\t</head>
\\r\\n\\t<body>
```

```
\\r\\n\\t\\t<form name="choco">
\\r\\n\\t\\t\\t<h3>Combien de fois par jour mangez-vous du chocolat ?</h3>
\\r\\n\\t\\t\\t<input type="text" name="reponse"> fois<br>
\\r\\n\\t\\t\\t<input type="button" value="Enregistrer la réponse" onClick="Main();">
\\r\\n\\t\\t</form>
\\r\\n\\t</body>
\\r\\n</html>
```

Exécuter le programme

L'utilisateur obtient des messages différents en fonction de la valeur insérée dans le formulaire.

[Voir la page](#)

15 : Synthèse

Dans ce module, vous avez appris à créer des programmes qui effectuent différents traitements en fonction d'une ou plusieurs **conditions**.

Pour effectuer différents traitements en fonction d'une condition, on utilise une **structure conditionnelle**.

Dans une condition, on compare la valeur de deux éléments.

16 : Mémo

Les conditions

Les **structures conditionnelles** permettent de faire une action ou une autre **en fonction d'un résultat**.

Grâce à elles, nous pouvons faire des **comparaisons** ou des **tests**, afin d'éviter d'effectuer des traitements inutiles ou d'obtenir des erreurs.

Dans un programme, il peut y avoir deux traitements différents **en fonction d'une condition**. Si la condition est **VRAIE**, on effectue un **traitement A** ; sinon la condition est **FAUSSE**, on effectue donc un **traitement B**.

Une condition se présente de cette façon:

Si la condition est vraie

On effectue le traitement A

Sinon

On effectue le traitement B

Voici la version en code:

```
if (motpropose == "bachibouzouk") { //condition
  alert("Vous avez le droit d'entrer!") // traitement A
} else {
  alert("Vous n'avez PAS le droit d'entrer"); // traitement B
```

Pensez à **indenter** votre code, et à **fermer toute accolade** ouverte directement. Egaleme^{nt}, le "if" et le "else" doivent être **écrits en minuscules**.

La **comparaison** se fait grâce au "==", tandis qu'une **assignation** de valeur se fait avec un simple "=".

La comparaison:

Egalité: ==
nombre == 100

Plus petit que: <
nombre < 100

Plus grand que: >
nombre > 100

Plus petit ou égal: <=
nombre <= 100

Plus grand ou égal: >=

nombre >= 100

Différent de: !=
nombre != 100

Plusieurs conditions imbriquées:

Il est possible de faire des **conditions à plusieurs niveaux**, en les **imbriquant** les unes dans les autres:

SI la **condition 1** est remplie, effectuer le **TRAITEMENT A**, **SINON SI** la **condition 2** est remplie effectuer **TRAITEMENT B** (et ainsi de suite...), **SINON** effectuer le **TRAITEMENT X**.

Voici cet exemple illustré en code:

```
if (condition1){  
  traitement A  
}  
  
else if (condition2){  
  traitement B  
}  
  
else if (condition3){  
  traitement C  
}  
  
...  
  
else {  
  traitement X  
}
```

On peut imbriquer autant de "if" que l'on veut, mais attention à garder un code lisible!

Toute la difficulté du jeu consiste à **ne pas commettre d'erreur de syntaxe**. Pensez à traiter vos structures conditionnelles **une à la fois**, afin d'éviter toute erreur éventuelle.

Les opérateurs logiques

Grâce à l'opérateur logique || (OU), vous pouvez diminuer le code nécessaire pour vos structures conditionnelles, en **reprenant plusieurs conditions** en un seul test:

```
if(proposition == "" || proposition > 100 || proposition < 0){ traitement }
```

On teste 3 choses: si la valeur entrée est vide, si elle est supérieure à 100, ou si elle est inférieure à 0, alors on effectue le traitement.

L'opérateur logique && (ET) permet également de grouper des conditions en les cumulant:

```
if(proposition != "" && proposition >= 0 && proposition <= 100){ traitement }
```

Si la proposition n'est pas vide, ET si elle est supérieure ou égale à 0, ET si elle est inférieure ou égale à 100, on effectue le traitement.

Instructions mathématiques

Afin de faire des tests plus poussés, il peut être utile de connaître les **instructions suivantes**:

Math.random() est une commande qui prend un nombre décimal au hasard entre 0 et 1

Math.round() permet d'arrondir un nombre décimal en entier.

17. 1 : Compléments

En complément, vous pourrez approfondir le travail avec plusieurs conditions. Nous vous proposons aussi, comme défi, de réaliser vous-même un quizz interactif.

17. 2 : Imbrication de structures conditionnelles

Voici une spécification supplémentaire pour le programme "jeu.htm" :

"Si l'utilisateur n'a inséré aucune valeur dans le formulaire, afficher l'alerte suivante : "Veuillez introduire un nombre entre 0 et 100 !", sinon on effectue les tests de comparaison." (nom du fichier : jeuV2.htm)

En d'autres termes :

Si la proposition est vide

Alors afficher "Veuillez introduire un nombre entre 0 et 100 !"

Sinon

Si la proposition est égale au nombre

Alors afficher "gagné !"

Sinon Si la proposition est inférieure au nombre

Alors afficher "Le nombre à trouver est plus grand !"

Sinon (la proposition est supérieure au nombre)

Afficher "Le nombre à trouver est plus petit !"

Il y a deux **conditions imbriquées**, voici le code correspondant à ces conditions :

```
if(proposition==""){
    alert("Veuillez introduire un nombre entre 0 et 100 ! ") ;
}else{
    if (proposition==nombre){
        alert("gagné !");
    }else if(proposition < nombre){
        alert("Le nombre à trouver est plus grand !");
    }else{
        alert("Le nombre à trouver est plus petit ! ");
    }
}
```

Complétez votre code et enregistrez-le sous "jeuV2.htm".

Exécuter le programme

Ce programme vérifie tout d'abord si l'utilisateur a bien inséré une proposition. [Voir la page](#)

17. 3 : Conclusion

On peut utiliser plusieurs **structures conditionnelles imbriquées (emboîtées)**.

La difficulté est de ne pas commettre **d'erreur de syntaxe**. Pour cela, soyez systématique : commencez par écrire le squelette de la première structure conditionnelle avant de vous attaquer à la seconde.

Indentez votre code, il sera beaucoup plus clair !

```
if(condition A){
    // Instruction A
}else{
    if(condition B){
        // Instruction B1
    }else{
        // Instruction B2
    }
}
```

17. 4 : Plusieurs comparaisons à la fois

Encore une spécification supplémentaire pour le jeu du bon numéro :

"Si l'utilisateur n'a inséré aucune valeur ou si la proposition est plus grande que 100 ou si la proposition est plus petite que 0, afficher l'alerte "Veuillez introduire un nombre entre 0 et 100 !""
(nom du fichier : jeuV3.htm)

Il est possible de vérifier ces 3 conditions à la suite en utilisant "else if" :

Si la proposition est vide

Alors afficher l'alerte d'erreur

Sinon Si la proposition est supérieure à 100

Alors afficher l'alerte d'erreur

Sinon Si la proposition est inférieure à 0

Alors afficher l'alerte d'erreur

Remarquez que ces 3 conditions mènent à la même erreur : "Veuillez fournir un nombre entre 0 et 100 !" Il y a donc répétition de l'instruction `alert("Veuillez fournir un nombre entre 0 et 100 !");` ;

Groupons ces 3 conditions :

```
if(proposition== "" || proposition>100 || proposition<0){
    alert("Veuillez fournir un nombre entre 0 et 100 !") ;
} else{
    // traitement B
}
```

Voici la traduction de cette condition :

Si la proposition est vide **OU** si la proposition est supérieure à 100 **OU** si la proposition est inférieure à 0

Alors afficher l'alerte d'erreur

|| signifie "OU", c'est un **opérateur logique** : il permet de **grouper** des conditions

Complétez votre code et enregistrez votre programme sous "jeuV3.htm".

Exécuter le programme

Ce programme vérifie tout d'abord si l'utilisateur a inséré une proposition correcte. [Voir la page](#)

17. 5 : Encore des comparaisons

Il existe d'autres opérateurs logiques : **&&** signifie "ET" et permet également de grouper des conditions.

Exemple :

Si la proposition n'est pas vide **ET si** la proposition est plus grande ou égale à 0 **ET si** la proposition est inférieure ou égale à 100

Alors effectuer la comparaison entre la proposition et le nombre.

... Et le code correspondant :

```
if(proposition != "" && proposition >= 0 && proposition <= 100){  
... traitement A  
}
```

Comparaison multiple

Ce programme vérifie tout d'abord si l'utilisateur a inséré une proposition correcte. [Voir la page](#)

17. 6 : En résumé

Le squelette de structures conditionnelles imbriquées :

```
if(condition A){  
// Instruction A  
}else{  
if(condition B){  
// Instruction B1  
}else{  
// Instruction B2  
}  
}
```

Les opérateurs conditionnels :

"ET"

```
if(a == b && b == c){  
  // alors a est égal à c  
}
```

"OU"

```
if(a < b || a > b){  
  // alors a est soit plus grand, soit plus petit que b (mais n'est pas  
  égal à b)  
}
```

17. 7 : Défi ! Un petit quizz

Voici l'énoncé du défi :


"Ecrire un programme qui présente un petit questionnaire (voir illustration ci-dessous). Lorsque l'utilisateur envoie ses réponses, une fenêtre d'alerte affiche son score."

MiniQuizz

Combien y a-t-il de Dalmatiens?

Combien de nains accueillent blanche-neige?

Combien Picsou a-t-il de neveux ?



Module 7 - Répéter une action

1 : Objectifs

Dans ce module, vous apprendrez à créer des structures dites "itératives" pour **répéter** un traitement. Ce type de **structure** possède quelques paramètres de configuration que vous apprendrez à manipuler pour créer une répétition personnalisée.

2 : Dire plusieurs fois la même chose

Rappelez-vous le module 4 ! Voici à nouveau l'énoncé de l'exercice :

"Afficher 3 fois le texte "Bienvenue dans l'univers de la BD" dans le document" (**nom du fichier : repeter3fois.htm**)

Vous allez **encore** améliorer le code de ce programme de manière à le rendre le plus concis possible.

Vous remarquez tout de suite qu'il y a une **répétition** (Afficher 3 fois un texte). Plutôt que d'écrire plusieurs fois **document.write**, vous allez créer un programme qui **répète** le traitement **document.write(message)**.

Pour répéter un traitement, on utilise une **structure itérative**, on parle également de **boucle**.

Voici le code du programme qui effectue une répétition de traitement :

```
<html>
  <head>
    <title>
      Afficher 3 fois le texte "Bienvenue dans l'univers de la BD"
dans le document
    </title>
    <script>
      var message;
      function Main(){
        message="Bienvenue dans l'univers de la BD !<br>";
        for(compteur=1;compteur<4;compteur=compteur+1){
          document.write(message);
        }
      }
    </script>
  </head>
  <body onload="Main()">
  </body>
</html>
```

Voici la traduction en français du code for... :

"Pour un compteur valant 1, tant que le compteur est inférieur à 4, j'effectue le traitement `document.write(message)` et j'augmente le compteur de 1 à chaque tour"

Enregistrez ce code sous "repeter3fois.htm" et exécutez-le.

Résultat attendu

Ce programme affiche 3 fois le même message à l'aide d'une structure itérative. [Voir la page](#)

3 : Conclusion

Lorsque l'on souhaite répéter plusieurs fois un traitement, on utilise une **structure itérative** (ou **boucle**).

Par exemple :

```
for (compteur=1;compteur<4;compteur=compteur+1) {  
    document.write(message);  
}
```

Pour déterminer le nombre de répétitions, on utilise une variable qui va jouer le rôle de **compteur** (c'est pourquoi on appelle souvent cette variable "**compteur**". On peut la nommer également "**i**").

A chaque répétition, le compteur va augmenter automatiquement. On dit que le compteur **s'incrémente** automatiquement.

4 : Appliquez : dire plusieurs fois la même chose

"Modifier le code du programme repeter3fois.htm de manière à afficher 5 fois le texte "La BD est à la fête !" dans le document".

```
<html>  
\\r\\n\\t<head>  
\\r\\n\\t\\t<title>Répéter 3 fois</title>  
\\r\\n\\t\\t<script=>  
\\r\\n\\t\\t\\tvar message;  
\\r\\n\\t\\t\\tfunction Main(){  
\\r\\n\\t\\t\\t\\tmessage = "La BD est à la fête !<br>" ;  
\\r\\n\\t\\t\\t\\tfor(compteur=1 ;compteur<4 ;compteur=compteur+1){  
\\r\\n\\t\\t\\t\\t\\tdocument.write(message) ;  
\\r\\n\\t\\t\\t\\t\\t}  
\\r\\n\\t\\t\\t\\t}  
\\r\\n\\t\\t\\t=script*  
\\r\\n\\t</head>  
\\r\\n\\t<body onload="Main();">  
\\r\\n\\t</body>  
\\r\\n</html> <html>
```

```

\\r\\n\\t<head>
\\r\\n\\t\\t<title>Répéter 3 fois</title>
\\r\\n\\t\\t<script=
\\r\\n\\t\\t\\tvar message;
\\r\\n\\t\\t\\tfunction Main(){
\\r\\n\\t\\t\\t\\tmessage = "La BD est à la fête !<br>" ;
\\r\\n\\t\\t\\t\\tfor(compteur=1 ;compteur<6 ;compteur=compteur+1){
\\r\\n\\t\\t\\t\\t\\t\\tdocument.write(message) ;
\\r\\n\\t\\t\\t\\t\\t}
\\r\\n\\t\\t\\t\\t}
\\r\\n\\t\\t=script*
\\r\\n\\t</head>
\\r\\n\\t<body onload="Main();">
\\r\\n\\t</body>
\\r\\n</html>

```

Exécuter le programme

Le message "La BD est à la fête !" s'affiche 5 fois dans le document. [Voir la page](#)

5 : Quizz

1. Une structure itérative est également appelée

- a) Boucle
- b) Bloc
- c) Structure

2. Une structure itérative permet :

- a) D'effectuer un traitement en fonction d'une condition
- b) D'effectuer plusieurs fois le(s) même(s) traitement(s)

6 : Le squelette d'une boucle

Afin de n'oublier aucun paramètre, il est conseillé de toujours écrire le squelette d'une structure itérative :

```

for(valeur initiale;test de répétition;incrément){
// traitement
}

```

Pour effectuer le compte des répétitions, on donne 3 paramètres :

1. une valeur de départ du compteur (**valeur initiale**) :

compteur=1

2. une condition de fin de répétition (**test de répétition**) :

compteur<4 qui signifie : **tant que** le compteur est inférieur à 4, on continue la répétition

3. la façon dont le compteur va augmenter, s'incrémenter (**incrément**) :

compteur= compteur+1 qui signifie : le compteur s'augmente de 1 à chaque fois.

Ces 3 paramètres sont séparés par un point-virgule ;

En résumé :

Valeur initiale : compteur=1

Test de répétition : compteur < 4

Incrément : compteur = compteur + 1

Il ne vous reste plus qu'à **remplacer** les commentaires en **rouge** par le contenu adéquat.

7 : Répéter 5 fois un traitement

Comment créer une boucle qui répète 5 fois un traitement ?

Partons du squelette de la structure itérative "for" :

```
for(i=valeur initiale;test de répétition;incrément){  
    // traitement  
}
```

Pour commencer, on va initialiser le compteur à 1 :

```
for(i=1; test de répétition;incrément){  
// traitement  
}
```

Ensuite, on augmente le compteur i de 1 à chaque tour :

```
for(i=1;test de répétition;i=i+1){  
traitement  
}
```

Pour n'effectuer le traitement **que 5 fois**, la valeur du compteur doit rester inférieure à 6 :

```
for(i=1;i<6;i=i+1){  
traitement  
}
```

En résumé :

Valeur initiale : 1

Test de répétition : i < 6

Incrément : i = i + 1

Le compteur passera par les valeurs suivantes : 1,2,3,4,5,6 **STOP**

8 : Quizz

1. Dans une boucle, le compteur permet de

- a) Compter le nombre de répétitions à effectuer
- b) Numéroté les traitements

2. Le paramètre qui permet au compteur d'augmenter automatiquement s'appelle

- a) Le compteur
- b) L'incrément
- c) Le test

9 : Exercice : compter jusqu'à 5

"Ecrire un programme qui énumère les nombres de 1 à 5 dans le document" (nom du fichier : **compter.htm**)

C'est facile ! Il vous suffit de créer une boucle qui affiche la valeur du compteur dans le document !

Un petit coup de pouce...

Valeur initiale : 1

Test de répétition : $i < 6$

Incrément : $i = i + 1$

```
for(i=valeur initiale ; test de répétition ; incrément){  
    document.write(i + "&nbsp;");  
    // on ajoute un espace après pour ne pas coller les nombres  
}
```

Enregistrez votre code sous "compter.htm" et exécutez-le.

Résultat attendu

Les nombres de 1 à 5 s'affichent dans le document. [Voir la page](#)

10 : Exercice : compter à rebours

"Ecrire un programme qui énumère les nombres de 5 à 0 dans le document" (nom du fichier : **compteAREbours.htm**)

Remarquez qu'ici il faut compter "à l'envers".

Valeur initiale : 5

Test de répétition : $i > -1$

Incrément : $i = i - 1$

Un petit coup de pouce...


```
for(i=valeur initiale ; test de répétition ; incrément){
    document.write(i + "&nbsp;");
    // on ajoute un espace après pour ne pas coller les nombres
}
```

Enregistrez votre code sous "compteAREbours.htm" et exécutez-le.

Remarque : Le test de répétition aurait pu être **i >= 0**, il est équivalent à **i > -1**.

Résultat attendu

Les nombres de 5 à 0 s'affichent dans le document. [Voir la page](#)

11 : Quizz

1. Laquelle de ces boucles ne compte pas de 0 à 100 ?

- a) for(i=0;i<101;i=i+1){document.write(i + " "); }
- b) for(i=0;i<100;i=i+1){document.write(i + " "); }
- c) for(i=0;i<=100;i=i+1){document.write(i + " "); }

12 : Exemple : sauter des valeurs

Jusqu'à présent, on a incrémenté le compteur de 1. Voici un exemple qui incrémente le compteur de 5 :

"Ecrire un programme qui donne tous les multiples de 5 jusqu'à 100" (nom du fichier : **tableDe5.htm**)

Remarquez qu'ici, il faut compter en sautant des valeurs. C'est l'**incrément** qui va vous permettre de sauter des valeurs :

Un petit coup de pouce...

Valeur initiale : 5

Test de répétition : i < 101 (ou i <= 100)

Incrément : i = i + 5

```
for(i=valeur initiale ; test de répétition ; incrément){
    document.write(i + "&nbsp;");
    // on ajoute un espace après pour ne pas coller les nombres
}
```

Enregistrez votre code sous "tableDe5.htm" et exécutez-le.

Résultat attendu

Les multiples de 5 s'affichent dans le document. [Voir la page](#)

13 : Conclusion

Les paramètres d'une boucle permettent de personnaliser le compteur.

La valeur du compteur peut augmenter (compte) ou diminuer (décompte) selon la définition de l'incrément.

Faites très attention aux **boucles infinies** !

Exemple de boucle infinie :

```
for(i=1;i>0;i=i+1){  
    document.write(i+"&nbsp;");  
}
```

Ici, la condition **i > 0** est **toujours vraie** : la répétition ne s'arrêtera **jamais** ! Le navigateur va se "planter" (comme on dit dans le jargon !) car cela lui prend trop de ressource d'écrire à l'infini !

Pour éviter les boucles infinies, il faut que la condition de répétition devienne, à un moment donné, **fausse**.

14 : Quizz

1. La boucle for(i=0 ; i < 100 ; i = i - 1){...}

- a) Est infinie
- b) Se répète 100 fois
- c) Se répète 101 fois

2. La boucle for(i=0; i < 5; i = i + 5){...} se répète

- a) Une fois
- b) Quatre fois
- c) Cinq fois

15 : Synthèse

Pour éviter d'écrire plusieurs fois le même traitement, on utilise une **structure itérative** (ou boucle) qui effectuera ce traitement plusieurs fois d'affilée.

La structure itérative "for" permet de répéter le même traitement plusieurs fois grâce à l'utilisation d'un **compteur** qui permet d'évaluer le nombre de répétitions à effectuer.

Dans ce module, vous avez appris à répéter plusieurs fois un traitement comme `document.write(message)` ou `document.write(compteur)`. Dans le module suivant, vous apprendrez à utiliser la structure itérative `for` pour parcourir les données d'un tableau.

16 : Mémo

Répéter une action

Les **structures itératives** (ou boucles) nous permettent de **répéter un traitement automatiquement**, en diminuant le risque d'erreurs et en rendant son programme plus concis.

Voici le code du programme qui effectue une répétition de traitement:

```
for(compteur=1 ;compteur<4 ;compteur=compteur+1){  
document.write(message) ;  
}
```

Ce qui pourrait être traduit par: "Pour un **compteur** valant 1, tant que le **compteur est inférieur** à 4, j'**effectue le traitement** document.write(message) et j'**augmente le compteur** de 1 à chaque tour"

Le rôle du compteur est de **déterminer le nombre de répétitions**. A chaque passage dans la boucle, celui-ci s'incrémente de 1 dans l'exemple ci-dessus. **Sans lui, on ne sortirait jamais de la boucle.**

Notez qu'on peut également **décrémenter** le compteur, lui additionner des valeurs 2 par 2, tant qu'on arrive à un moment à la valeur de sortie.

Dès que la **valeur de sortie** est atteinte (ici quand compteur<4 devient faux), on **quitte la boucle** sans effectuer le traitement.

Attention aux boucles infinies!

```
for(i=1;i>0;i=i+1){  
    document.write(i+"    ");  
}
```

Dans ce cas de figure, la condition sera **toujours vraie**, vous ne sortirez jamais de votre boucle, ce qui peut causer un **plantage** de votre navigateur.

Pour éviter les boucles infinies, il faut que la condition de répétition devienne, à un moment donné, **fausse**.

Syntaxe avancée

Pour incrémenter le compteur de 1, on effectue l'opération suivante :

```
i = i + 1;
```

On peut aussi l'écrire des façons suivantes:

```
i+=1;  
i++;
```

Ce sont trois syntaxes pour le même résultat. Vous pouvez remplacer le + par un - si vous souhaitez décrémenter votre compteur.

Combiner une répétition et une action

On peut très bien combiner une boucle **for** et une condition **if**:

```
for(i=1;i<= 50;i++){  
    if(i/7==parseInt(i/7) ){  
        document.write("<b>" + i + "</b>&nbsp;");  
    }else{  
        document.write(i + "&nbsp;");  
    }  
}
```

parseInt() transforme une valeur **décimale en entier**.

Cet exemple compte de 1 à 50, et vérifie si le compteur divisé par 7 donne un entier, auquel cas il l'affiche en gras, sinon il l'affiche en normal.

Sortir d'une boucle

Pour sortir d'une boucle, il suffit d'utiliser l'instruction **break** ;

Break est un mot réservé qui permet de sortir d'une boucle:

```
for( ){  
    if( ){  
        break ; // on sort de l'instruction  
    }else{  
        // instruction  
    }
```

```
}
```

17. 1 : Compléments

Dans les compléments, vous découvrirez ce qu'est `i++` et apprendrez à combiner boucles et conditions. Vous pourrez aussi mettre en pratique la matière de ce chapitre au travers de plusieurs exercices supplémentaires.

17. 2 : L'incrément

Pour **incrémenter** le compteur de `1`, on effectue l'opération suivante :

`i = i + 1` qui signifie que la variable `i` augmente de `1`.

Voici d'autres opérations équivalentes à `i = i + 1` :

`i += 1`

ou encore

`i++`

`++` est un **opérateur** qui augmente automatiquement une variable de `1`.

De même pour `i = i + 5` :

`i += 5`

On retrouve les mêmes équivalences pour la soustraction :

`i = i - 1` est équivalent à

`i -= 1`

`i--`

17. 3 : Quizz

1. Quelle opération n'est pas équivalente à `i++` ?

- a) `i += 1`
- b) `i = 1 + i`
- c) `i = 1`

2. Quelle opération est équivalente à `i -= 5` ?

- a) `i = i - 5`
- b) `i = 5 - i`
- c) `i == 5`

17. 4 : Une condition dans une boucle

"Ecrire un programme qui affiche tous les nombres de 1 à 50. A chaque fois que le nombre est un multiple de 7, il est écrit en gras." (nom du fichier : [tableDe7V2.htm](#))

Voici le début du résultat affiché dans le document : `1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 ...`

Comment vérifier si un nombre est multiple de 7 ?

Il suffit de vérifier que le quotient de la division du nombre par 7 est bien un nombre entier :

```
if(nombre/7==parseInt(nombre/7)){
    // le nombre est un multiple de 7
}
```

nombre/7 : quotient en nombre décimal (à virgule)

parseInt(nombre/7) : prend uniquement le nombre entier du quotient

Par exemple, vérifions si 10 est un multiple de 7 :

10/7 = 1,42857... (le quotient en nombre décimal)

parseInt(10/7) = 1 (uniquement le nombre entier du quotient)

=> 1,42857... n'est pas égal à 1, donc le nombre 10 n'est pas divisible par 7 car le quotient n'est pas un **nombre entier** !

Vérifions si 14 est multiple de 7 :

14/7 = 2

parseInt(14/7) = 2

=> 2 est égal à 2, donc le nombre 14 est divisible par 7 car le quotient est un **nombre entier** !

Remarque : parseInt() est une commande qui convertit n'importe quel nombre en nombre entier.

Ex : parseInt(3,1415) = 3

parseInt(1234,56789) = 1234

Voici la boucle qui affiche tous les nombres de 1 à 50 :

```
for(i=1;i<= 50;i++){
    document.write(i + "&nbsp;");
}
```

Il reste à vérifier si le nombre est un multiple de 7 :

```
for(i=1;i<= 50;i++){
    if(i/7==parseInt(i/7)){
        document.write("<b>" + i + "</b>&nbsp;");
    }else{
        document.write(i + "&nbsp;");
    }
}
```

Complétez ce code et enregistrez votre programme sous "tableDe7V2.htm".

Résultat attendu

Ce programme affiche tous les nombres de 1 à 50. Lorsque le programme rencontre un multiple de 7, il l'affiche en gras. [Voir la page](#)

17.5 : Conclusion

On peut **imbriquer** une structure conditionnelle if dans une boucle for.

Voici le squelette d'une structure conditionnelle imbriquée dans une boucle for :

```
for( ){
    if( ){
        // instruction
    }else{
        // instruction
    }
}
```

Soyez attentif ! Ne faites pas d'**erreur de syntaxe** en oubliant une parenthèse, une accolade ou un point-virgule.

17.6 : Appliquez : un if dans un for

"Ecrire un programme qui affiche tous les nombres de 1 à 100. Tous les multiples de 9 doivent s'afficher en gras".

```
<head>
\\r\\n\\t<title>Afficher tous les nombres de 1 à 100. Les multiples de 9 doivent être affichés en gras.</
title>
\\r\\n\\t\\t*script=
\\r\\n\\t\\t\\tfunction Main(){
\\r\\n\\t\\t\\t\\tfor(){
\\r\\n\\t\\t\\t\\t\\t\\tif( ){
\\r\\n\\t\\t\\t\\t\\t\\t\\t
\\r\\n\\t\\t\\t\\t\\t\\t\\t}else{
\\r\\n\\t\\t\\t\\t\\t\\t\\t\\td
\\r\\n\\t\\t\\t\\t\\t\\t\\t}
\\r\\n\\t\\t\\t\\t\\t\\t\\t}
\\r\\n\\t\\t\\t\\t\\t}
\\r\\n\\t\\t\\t\\t\\t}
\\r\\n\\t\\t\\t=script*
\\r\\n\\t</head>
\\r\\n\\t<body onload="Main();">
\\r\\n\\t</body>
\\r\\n</html> <head>
\\r\\n\\t<title>Afficher tous les nombres de 1 à 100. Les multiples de 9 doivent être affichés en gras.</
title>
\\r\\n\\t\\t*script=
\\r\\n\\t\\t\\tfunction Main(){
\\r\\n\\t\\t\\t\\tfor(i=1 ; i<101 ; i=i+1){
\\r\\n\\t\\t\\t\\t\\t\\tif(i/9 == parseInt(i/9)){
```

```

\\r\\n\\t\\t\\t\\t\\t\\t\\tdocument.write("<b>" + i + "</b> ");
\\r\\n\\t\\t\\t\\t\\t\\t\\t}else{
\\r\\n\\t\\t\\t\\t\\t\\t\\tdocument.write(i + " ");
\\r\\n\\t\\t\\t\\t\\t\\t\\t}
\\r\\n\\t\\t\\t\\t\\t\\t\\t}
\\r\\n\\t\\t\\t\\t\\t\\t\\t}
\\r\\n\\t\\t=script*
\\r\\n\\t</head>
\\r\\n\\t<body onload="Main();">
\\r\\n\\t</body>
\\r\\n</html>

```

Exécuter le code

Le programme affiche tous les nombres de 1 à 100, les multiples de 9 sont affichés en gras. [Voir la page](#)

17. 7 : Sortir d'une boucle

"Ecrire un programme qui demande (via un formulaire) un nombre à l'utilisateur. Le programme se met à afficher tous les nombres de 1 à 100. S'il rencontre le nombre introduit par l'utilisateur, il s'arrête, sinon il continue jusqu'à 100." (Nom du fichier : **arreteCompter.htm**)

Pour sortir d'une boucle, il suffit d'utiliser l'instruction `break` ; .

Break est un mot réservé qui permet de sortir d'une boucle.

```

for( ){
    if( ){
        break; // on sort de l'instruction
    }else{
        // instruction
    }
}

```

Essayez d'écrire le code de ce programme et enregistrez-le sous "arreteCompter.htm".

Résultat attendu

L'utilisateur insère un nombre dans le formulaire. Le programme se met à compter à partir de 1. S'il rencontre le nombre inséré par l'utilisateur, il s'arrête. Sinon, il s'arrête à 100. [Voir la page](#)

17. 8 : Défi ! Ecrire de plus en plus grand

Voici l'énoncé de ce défi :

"Ecrire un programme qui affiche le texte "Bienvenue dans l'univers de la BD !" 5 fois et de plus en plus grand."

Le résultat doit ressembler à ceci :

Bienvenue dans l'univers de la BD

Bienvenue dans l'univers de la BD

Bienvenue dans l'univers de la BD

Bienvenue dans l'univers de la BD

Bienvenue dans l'univers de la BD

Module 8 - Grouper plusieurs valeurs

1 : Objectifs

L'objectif de ce module est de stocker des **collections de valeurs** sous **un seul nom de variable**. Ce type de variable permettant de **grouper plusieurs valeurs** sous un même nom s'appelle **tableau**.

Dans ce module, vous apprendrez à créer des variables de type **tableau**, à stocker un ensemble de valeurs dans un tableau et à afficher les différentes valeurs d'une collection de données.

2 : Une collection de bandes dessinées

"Ecrire un programme qui affiche le titre d'une BD du petit Spirou en fonction du numéro introduit par l'utilisateur dans un champ de formulaire". (**nom du fichier : afficheBD.htm**)

Comment s'y prendre ?

Il faudrait stocker tous les titres de la série BD dans le programme pour ensuite pouvoir les récupérer via leur numéro.

1re solution :

On stocke chaque titre dans une variable :

```
BD1="Dis Bonjour à la dame !";  
BD2="Tu veux mon doigt ?";  
// etc.
```

Ensuite, on récupère le numéro dans le champ du formulaire, sans oublier de convertir cette valeur en nombre :

```
numero=parseFloat(document.BD.numero.value);
```

Et on va chercher le titre correspondant au numéro :

```
if (numero==1){  
    alert("Le titre correspondant au numéro est : "+BD1);  
}else if (numero==2){  
    alert("Le titre correspondant au numéro est : "+BD2);  
}
```

Résultat attendu

L'utilisateur insère un nombre et le programme affiche le titre correspondant au numéro. [Voir la page](#)

Les désavantages de cette méthode :

- Le code est long
- Plus il y a de tomes, plus il faut rajouter des lignes de code

Voyons une solution plus optimale...

3 : 2e solution : utilisation d'un tableau

Les titres "Dis bonjour à la dame", "Tu veux mon doigt ?", ... sont liés entre eux car ils font **tous** partie d'une même série de bandes dessinées. **Groupons** ces valeurs dans **une seule** variable **spirou** de type tableau.

Un tableau est une variable qui permet de **stocker plusieurs valeurs sous un même nom avec une indexation** (numérotation) :

```
var spirou=new Array();  
// déclaration de la variable spirou qui est de type tableau  
spirou=["Dis Bonjour à la dame !","Tu veux mon doigt ?","Mais ! Qu'est-ce  
que tu fabriques ?","C'est pour ton bien !","Merci qui ?","N'oublie pas ta  
capuche !","Demande à ton père !","T'as qu'à t'retenir !","C'est pas de ton  
âge !","Tu comprendras quand tu seras grand !"];  
// assignation de l'ensemble des valeurs dans le tableau
```

Voici une représentation visuelle du tableau spirou :

| Spirou | |
|--------|---|
| 0 | "Dis Bonjour à la dame !" |
| 1 | "Tu veux mon doigt ?" |
| 2 | "Mais ! Qu'est-ce que tu fabriques ?" |
| 3 | "C'est pour ton bien !" |
| 4 | "Merci qui ?" |
| 5 | "N'oublie pas ta capuche !" |
| 6 | "Demande à ton père !" |
| 7 | "T'as qu'à t'retenir !" |
| 8 | "C'est pas de ton âge !" |
| 9 | "Tu comprendras quand tu seras grand !" |

Attention ! Remarquez que l'**indexation** d'un tableau commence à 0. Un tableau qui contient 10 valeurs a un **indice** qui va de 0 à 9 !

4 : Atteindre une donnée d'un tableau

Pour accéder à un titre de BD, il suffit d'afficher la variable **spirou** avec l'**indice** souhaité.

ATTENTION ! Comme le **premier album** est positionné à l'**indice 0**, si on souhaite atteindre le titre correspondant au **tome 3**, il faudra demander la valeur correspondant à l'**indice 2**.

Il y a donc une petite **astuce** : pour atteindre le titre d'un album, il faut **diminuer** le numéro de 1 pour connaître son **indice** dans le tableau ;-)

Accédons par exemple au titre correspondant au tome 3 :

```
alert("Le titre correspondant au numéro est : " + spirou[2]) ;
```

Donne comme résultat :

Le titre correspondant au numéro est : "Mais ! Qu'est-ce que tu fabriques ?"

Pour afficher le titre correspondant à la valeur insérée par l'utilisateur dans le formulaire:

```
numero=parseFloat(document.BD.numero.value)-1;
// on diminue le nombre introduit pour connaître l'indice
```

```
alert("Le titre correspondant au numéro est : "+spirou[numero]);
```

Assemblez ces morceaux de code pour former le programme qui affiche le titre correspondant au numéro inséré par l'utilisateur dans le formulaire.

Enregistrez le code sous "AfficheBD.htm" et exécutez-le.

Résultat attendu

L'utilisateur insère un nombre et obtient le titre de la BD correspondant au numéro. [Voir la page](#)

5 : Conclusion

Lorsque l'on a un ensemble de valeurs qui forment une **liste** (comme la liste des mois, liste des membres d'un club, liste des tomes d'une BD, ...), on les stocke ensemble dans une seule variable de type **tableau**.

On accède à différentes valeurs du tableau par un **indice** (position de l'élément dans le tableau).

L'indice d'un tableau commence toujours à 0. La **1re valeur** d'un tableau a donc **l'indice 0**.

6 : Quizz

1. L'indice minimum d'un tableau est :

- a) Défini en fonction du nombre de valeurs se trouvant dans le tableau
- b) 0
- c) 1

2. Si on veut atteindre le titre du tome 3 du petit Spirou, quelle commande doit-on utiliser ?

- a) spirou[2]
- b) spirou["3"]
- c) spirou[3]

7 : La propriété "length"

La variable de type "tableau" présente la propriété **length** qui signifie **longueur**:

```
var spirou=new Array();  
// déclaration de la variable spirou qui est de type tableau  
var numero;  
spirou=["Dis Bonjour à la dame !","Tu veux mon doigt ?","Mais ! Qu'est-ce  
que tu fabriques ?","C'est pour ton bien !","Merci qui ?","N'oublie pas ta  
capuche !","Demande à ton père !","T'as qu'à t'retenir !","C'est pas de ton  
âge !","Tu comprendras quand tu seras grand !"];  
// assignation de l'ensemble des valeurs dans le tableau
```

```
alert(spirou.length);
```

La propriété **length** de la variable **spirou** renvoie la **taille** du tableau. Dans ce cas-ci, la valeur retournée est **10** car il y a 10 données dans le tableau.

Cette propriété permet de calculer **l'indice maximum** d'un tableau.

Voici le calcul qui permet de trouver cet indice maximum :

on prend la taille du tableau (10) et on lui retire 1 (**10-1=9**).

Règle générale :

Indice minimum : 0

Indice maximum : taille du tableau – 1

Vous allez voir maintenant à quoi sert cette propriété **length** d'une variable de type **tableau**.

Mais avant cela... un petit quizz !

8 : Quizz

1. Soit un tableau contenant 12 valeurs, quelle est la valeur de l'indice maximum ?

- a) 11
- b) 12
- c) 13

**2. var jour= new Array() ; jour =
["lundi","mardi","mercredi","jeudi","vendredi","samedi","dimanche"] ; Quelle est la
taille de ce tableau ?**

- a) 6
- b) 7
- c) 8

3. Quelle commande permet de trouver l'indice maximum d'un tableau ?

- a) IndiceMax = tableau.length
- b) IndiceMax = tableau.length - 1
- c) IndiceMax = tableau.length + 1

9 : Afficher toutes les valeurs d'un tableau

"Ecrire un programme qui affiche la collection complète des albums du Petit Spirou dans la page HTML." (nom du fichier : **AfficheListe.htm**)

Jusqu'à présent, vous n'avez fait qu'afficher une valeur du tableau à la fois. Il vous faut maintenant afficher TOUTES les valeurs du tableau spirou.

Pour cela, vous allez utiliser la **structure itérative "for"** pour parcourir le tableau et afficher toutes ses valeurs :

Valeur initiale du compteur = indice minimum du tableau : **i = 0**

Test de répétition de la boucle : $i < \text{longueurTableau}$

Incrément : $i = i + 1$

Avec une telle boucle, le compteur de la boucle passera par les valeurs suivantes :

0,1,2,3,4,5,6,7,8,9,10

Ce qui correspond exactement aux différents indices du tableau !

Voici le code de la boucle qui parcourt le tableau **spirou** :

```
for( i=0;i<spirou.length;i=i+1){  
    document.write(spirou[i]+"<br>");  
}
```

Il ne reste plus qu'à mettre cette boucle dans la fonction Main() de votre programme et appeler la fonction principale au chargement de la page.

Enregistrez le code sous afficheListe.htm et exécutez votre programme.

Résultat attendu

La liste des BD du Petit Spirou est affichée dans le document. [Voir la page](#)

10 : Quizz

1. Pour parcourir toutes les données d'un tableau, on utilise

- a) Une structure itérative
- b) Une structure conditionnelle
- c) Une structure indicative

2. Pour parcourir TOUTES les données d'un tableau, on utilise l'incrément :

- a) $i = i - 1$
- b) $i = i + 1$
- c) $i = i + 5$

11 : Synthèse

Grâce à l'utilisation des variables de type "tableau", vous pouvez grouper plusieurs données qui sont liées entre elles.

Cela vous évite la création d'un nombre important de variables de type "chaîne de caractères" ou de type "numérique".

Grâce à l'**indexation** des données dans un tableau, on peut aisément retrouver une donnée, il suffit de connaître sa **position** dans le tableau.

On peut connaître la **taille** d'un tableau grâce à la propriété **length**.

Pour parcourir TOUTES les valeurs d'un tableau, on utilise une **structure itérative "for"** dont le compteur vaut successivement les valeurs des indices du tableau.

12 : Mémo

Les tableaux

Jusqu'ici, les **variables** permettaient de stocker une seule valeur à la fois.

Les **tableaux** permettent d'en grouper **plusieurs** en un seul endroit, afin d'**alléger son code** et de le rendre plus **dynamique**.

Voici la déclaration d'un tableau:

```
var spirou = new Array();
```

Et la façon de lui assigner des valeurs:

```
spirou=["Dis Bonjour à la dame !","Tu veux mon doigt ?","Mais ! Qu'est-ce  
que tu fabriques ?","C'est pour ton bien !","Merci qui ?","N'oublie pas ta  
capuche !","Demande à ton père !","T'as qu'à t'retenir !","C'est pas de ton  
âge !","Tu comprendras quand tu seras grand !"];
```

Voici deux autres façons de créer **le même tableau**:

La première:

```
var spirou=new Array("Dis Bonjour à la dame !","Tu veux mon doigt ?","Mais !  
Qu'est-ce que tu fabriques ?","C'est pour ton bien !","Merci  
qui ?","N'oublie pas ta capuche !","Demande à ton père !","T'as qu'à  
t'retenir !","C'est pas de ton âge !","Tu comprendras quand tu seras  
grand !");
```

Et la seconde:

```
var spirou=new Array();  
spirou[0]="Dis Bonjour à la dame !";  
spirou[1]="Tu veux mon doigt ?";  
spirou[2]="Mais ! Qu'est-ce que tu fabriques ?";  
spirou[3]="C'est pour ton bien !";  
// ...  
spirou[9]="Tu comprendras quand tu seras grand !";
```

A vous de voir celle qui vous convient le mieux.

Afficher le contenu d'un tableau

Attention, **les tableaux commencent à compter à 0**, le 1er élément sera donc le 0, le second le 1, etc...

Un tableau de 10 éléments ira donc **de 0 à 9**, il faut donc diminuer le numéro de la ligne désirée d'une valeur pour connaître **son indice dans le tableau**.

Pour afficher le troisième titre "**Mais ! Qu'est-ce que tu fabriques ?**", nous utiliserons donc la syntaxe suivante:

```
document.write("Le troisième tome de Spirou est: "+spirou[2]);
```

Si on désire afficher le titre correspondant à la valeur insérée par l'utilisateur dans le formulaire:

```
numero=parseFloat(document.BD.numero.value)-1;  
// on diminue le nombre introduit pour connaître l'indice  
alert("Le titre correspondant au numéro est : "+spirou[numero]);
```

Connaître la taille d'un tableau

Les variables de type **Array()** bénéficient de la propriété **length**, qui permet de connaître la taille du tableau:

```
taille = spirou.length;
```

Vu que les indices des tableaux commencent à 0, le dernier élément du tableau aura pour indice la **longueur du tableau - 1**.

Afficher tous les éléments d'un tableau

Maintenant que l'on connaît la taille d'un tableau, nous pouvons en afficher tous les éléments grâce à une **structure itérative "for"**.

Il suffit de déterminer la **longueur du tableau**, et de le parcourir tant qu'on est inférieur à cette longueur (ou qu'on est égal à cette longueur -1) vu que les indices commencent à 0 jusqu'à la longueur -1.

```
for(i=0;i<spirou.length;i=i+1){  
    document.write(spirou[i]+"<br>");  
}
```

Vous pouvez également vouloir **trier vos valeurs** avant leur affichage. Pour cela, il existe une méthode toute faite: **sort()**

```
spirou = spirou.sort() ;
```

Afficher la date

Il est possible d'utiliser une méthode **affichant la date**, par exemple pour créer un tableau contenant tous les mois de l'année:

```
var aujourd'hui=new Date();  
document.write(aujourd'hui) ;
```

La date récupérée est au format "complet", on peut la formater pour n'en récupérer qu'une partie:

- **aujourd'hui.getDate()** : Récupère le jour (ex : 24)
- **aujourd'hui.getMonth()** : Récupère le numéro du mois, 0 correspondant à janvier, 11 correspondant à décembre (ex : 03)
- **aujourd'hui.getFullYear()** : Récupère l'année en 4 chiffres (ex : 2003)
- **aujourd'hui.getDay()** : Récupère le numéro du jour de la semaine, 0 correspondant à dimanche, 6 correspondant à samedi

De cette façon, la création de tableaux contenant des dates est simplifiée.

13. 1 : Compléments

En complément, vous découvrirez trois méthodes pour créer un tableau. Vous apprendrez aussi à trier des données et à afficher la date.

13. 2 : Trois manières de créer un tableau

Il existe différentes méthodes de création de tableau. Voici 3 méthodes équivalentes :

```
var jour=new Array();
jour=["dimanche","lundi","mardi","mercredi","jeudi","vendredi","samedi"];
```

OU

```
var jour=new
Array("dimanche","lundi","mardi","mercredi","jeudi","vendredi","samedi");
```

OU

```
var jour = new Array() ;jour[0] = "dimanche" ;jour[1] = "lundi" ;jour[2]
= "mardi" ;jour[3] = "mercredi" ;jour[4] = "jeudi" ;jour[5] =
"vendredi" ;jour[6] = "samedi" ;
```

La dernière méthode est la plus longue mais aussi la plus lisible. A vous de choisir la méthode qui vous convient le mieux

13. 3 : Trier un tableau

"Ecrire un programme qui affiche la liste des BD du Petit Spirou triées par ordre alphabétique."
(nom du fichier : tri.htm)

Pour trier les éléments d'un tableau, il suffit de reprendre le tableau et d'appeler la méthode sort() :

```
spirou = spirou.sort() ;
```

Ensuite, il ne vous reste plus qu'à parcourir le tableau et à afficher chaque valeur dans le document.

Résultat attendu

La liste des BD du Petit Spirou s'affiche dans l'ordre alphabétique. [Voir la page](#)

13. 4 : Afficher la date

Avec le JavaScript, il est possible d'afficher la **date du jour** dans le document.

```
var aujourd'hui=new Date();
document.write(aujourd'hui);
```

Voici le résultat obtenu à l'écran :

Thu Apr 24 11:47:14 UTC+0200 2003

Comme vous pouvez le remarquer, le format de la date n'est pas très joli !

On peut néanmoins effectuer des traitements sur la date pour obtenir un format plus adéquat.

- **aujourd'hui.getDate()** : Récupère le jour (ex : 24)
- **aujourd'hui.getMonth()** : Récupère le numéro du mois, 0 correspondant à janvier, 11 correspondant à décembre (ex : 03)
- **aujourd'hui.getFullYear()** : Récupère l'année en 4 chiffres (ex : 2003)
- **aujourd'hui.getDay()** : Récupère le numéro du jour de la semaine, 0 correspondant à dimanche, 6 correspondant à samedi

1. "Ecrire un programme qui affiche la date au format suivant : Nous sommes le [libellé du jour] JJ [libellé du mois] AAAA (par exemple : Nous sommes le 31/12/1999). Les libellés des jours de la semaine et des mois seront stockés dans une variable de type tableau" (**nom du fichier : date.htm**)

Si votre programme ne tourne pas correctement, vous pouvez voir la solution en ligne.

2. "Ecrire un programme qui affiche la date au format suivant : Nous sommes le dimanche 31 décembre 1999)." (**nom du fichier : dateV2.htm**)

Pour ce faire, il vous faudra créer un tableau **tabMois** contenant tous les libellés des mois et un tableau **tabSemaine** contenant tous les libellés des jours de la semaine (conseil : commencez par dimanche car les calendriers anglais commencent les semaines à dimanche).

Si votre programme ne tourne pas correctement, vous pouvez voir la solution en ligne.

Solution du Programme 1

Ce programme récupère la date du système à l'aide de la commande **var aujourd'hui = new Date()** ;

La date du jour s'affiche au format JJ/MM/AAAA. [Voir la page](#)

Solution du Programme 2

Ce programme récupère la date du système à l'aide de la commande **var aujourd'hui = new Date()** ;

La date du jour s'affiche au format suivant : "Nous sommes le Libellé du jour JJ Libellé du mois AAAA". [Voir la page](#)

Module 9 - Utiliser des fonctions

1 : Objectifs

So far, so good ! Dans les modules précédents, vous avez appris les bases de la programmation à l'aide d'exemples et d'exercices. Dans ce module, vous apprendrez à grouper un traitement dans une **fonction**. Plus qu'un élément de base en programmation, les fonctions vont vous permettre d'avoir un code plus **clair** et **réutilisable**.

Dans ce but, vous allez reprendre des exercices existants pour les améliorer.

2 : La fonction Main()

Depuis le début de ce cours, vous utilisez les fonctions !

La fonction **Main()** est une fonction créée par le programmeur dans le but de pouvoir exécuter un traitement au moment souhaité : **onclick** d'un bouton, **onload** de la page, ...

Sans l'utilisation de fonctions, les instructions JavaScript seraient exécutées directement **au moment** de la lecture par le navigateur.

Comme il était dit dans le premier module, cette fonction n'est pas obligatoire mais juste utile.

Vous avez donc déjà une idée de ce que représente une fonction : elle regroupe un ensemble d'**instructions**.

Un des intérêts d'une fonction est qu'on peut l'appeler à certains moments bien précis grâce aux événements !

3 : Créer une fonction

Voici le squelette d'une fonction :

```
function NomDeLaFonction(){  
    // instructions de la fonction  
}
```

function est un **mot réservé** qui doit être écrit en minuscules.

Les parenthèses sont **obligatoires**, elles ont un rôle important dans une fonction pour le passage de valeurs à une fonction.

Comme pour les noms de variables, le **nom d'une fonction** doit être :

- en un seul mot
- commencer par une lettre ou " _ "

Conseil : Evitez les mots réservés et des noms identiques pour une fonction et une variable, vous obtiendriez une erreur ! Généralement, on différencie un nom de fonction d'un nom de variable en le commençant par une **majuscule**. Ce n'est pas une obligation, c'est une convention de programmeur...

4 : Quizz

1. Lequel de ces noms de fonction est incorrect ?

- a) Fonction
- b) function
- c) fonction

5 : Appeler une fonction

Pour **appeler** une fonction, il suffit de l'appeler par son nom (sans oublier les parenthèses) :

```
NomDeLaFonction();
```

Une fonction peut être appelée :

- dans un événement se trouvant dans une balise HTML

```

```

- dans le code Javascript

```
<script>
  function Main(){
    Affiche() ; // on a appelé la fonction Affiche()
  }
  function Affiche(){
    alert("Bienvenue dans l'univers de la BD !");
  }
</script>
```

1. Lequel de ces morceaux de code n'appelle pas la fonction "AfficheMessage" ?

- a) <input type="Button" value=" " onclick="AfficheMessage;">
- b) <body onload="AfficheMessage();">
- c) <script>...AfficheMessage() ; ...</script>

6 : Exemple : une collection de BD

"Ecrire un programme qui affiche le titre d'une BD du Petit Spirou en fonction du numéro introduit par l'utilisateur dans un champ de formulaire. Le programme affiche la collection complète des albums du Petit Spirou lorsque l'utilisateur clique sur le bouton "Affiche liste". (nom du fichier : **collectionBD.htm**)

Ce programme réunit les deux exercices du module précédent.

Il y aura donc deux traitements différents correspondant à deux événements différents :

Action 1 : clic sur le bouton "Afficher le titre correspondant"

Traitement 1 : affichage du titre correspondant au numéro introduit dans le formulaire

Action 2 : clic sur le bouton "Afficher la liste complète"

Traitement 2 : affichage de la liste complète des albums du Petit Spirou

La page suivante explique comment séparer des traitements en utilisant plusieurs fonctions.

7 : Un programme - Deux traitements

Comment séparer deux traitements différents ? En utilisant les **fonctions** !

Vous allez créer deux fonctions :

- **RecupTitre()**
- **AfficheListe()**

Chacune de ces fonctions effectuera un traitement bien précis : soit l'affichage d'un titre par rapport au tome, soit l'affichage de la liste complète des albums.

Vous pouvez essayer de faire l'exercice seul. Allez à la page suivante pour voir le détail du programme

8 : Le code détaillé du programme

```
<html>
  <head>
    <title>
      Collection de BD "Le Petit Spirou"
    </title>
    <script>
      var spirou=new Array()
```

```

        // déclaration de la variable spirou qui est de type tableau
        spirou=["Dis Bonjour à la dame !","Tu veux mon doigt ?","Mais !
Qu'est-ce que tu fabriques ?","C'est pour ton bien !","Merci
qui ?","N'oublie pas ta capuche !","Demande à ton père !","T'as qu'à
t'retenir !","C'est pas de ton âge !","Tu comprendras quand tu seras
grand !"] ;

        // assignation de l'ensemble des valeurs dans le tableau
        function RecupTitre(){
            var numero;
            numero=parseFloat(document.BD.numero.value)-1;
            // récupération du nombre introduit par l'utilisateur
            if((numero>=0)&&(numero<=9)){
                alert("Le titre correspondant au numéro est : " +
spirou[numero]);
                // affichage du titre correspondant au numéro introduit
dans le formulaire
            }else{
                alert("Il n'existe aucun titre correspondant au numéro :
"+ numero);
                // affichage d'un message d'erreur
            }
        }
        function AfficheListe(){
            for(i=0;i<spirou.length;i=i+1){
                // parcours de toutes les valeurs du tableau
                document.write(spirou[i]+"<br>");
            }
        }
    </script>
</head>
<body>
    <form name="BD">
        <input type="text" name="numero">
        <input type="button" value="Afficher le titre
correspondant" onclick="RecupTitre();">
        <input type="button" value="Afficher la liste
complète" onclick="AfficheListe();">
    </form>
</body>
</html>

```

Enregistrez ce code sous CollectionBD.htm et exécutez le programme.

Résultat attendu

Ce programme permet de :

1. trouver le titre d'une BD à partir du numéro du tome
2. afficher la liste complète de la collection de BD

[Voir la page](#)

9 : Variable locale

Remarquez que, dans cet exercice, on a déclaré la variable **numero** à l'intérieur de la fonction **RecupTitre()**.

La variable **numero** est une **variable locale** (en opposition avec une **variable globale**), ce qui signifie que cette variable n'est déclarée que pour la fonction **RecupTitre()** et ne pourra pas être utilisée en dehors de celle-ci.

Une variable locale **est définie à l'intérieur de la fonction** dans laquelle elle va être utilisée (généralement au début de la fonction).

Une **variable globale** est une variable qui est **accessible de partout**. Pour définir une variable globale, il suffit de la déclarer en dehors de toute fonction (généralement en début de programme).

Important : si vous oubliez de déclarer une variable, la variable sera **automatiquement** déclarée en tant que variable globale.

10 : Quizz

1. Dans le code suivant : `function Main(){var i = 0 ;...}` La variable `i` ...

- a) N'est pas accessible en dehors de la fonction `Main()`
- b) Est accessible de partout mais uniquement si on appelle la fonction `Main()`
- c) Est accessible en dehors de la fonction `Main()`

2. Dans ce code, laquelle de ces variables n'est pas globale ? ...`eur = 40,3399 ; function Main() {var message ; message = "Bienvenue" ; message2 = "Au revoir ; "}`...

- a) `message`
- b) `eur`
- c) `message2`

11 : Les parenthèses d'une fonction

Jusqu'ici, vous avez utilisé des fonctions sans jamais utiliser les parenthèses.

Par exemple :

```
function Main(){  
    // instructions  
}
```

Les parenthèses d'une fonction sont utilisées pour transmettre un ou plusieurs paramètres :

Exemple :

```
//définition de la fonction
function Affiche(message){
    alert(message);
}
```

```
//Appel de la fonction
Affiche("Bienvenue dans l'univers de la BD !");
Affiche("Comment allez-vous ?");
```

Grâce aux paramètres, on peut passer des informations différentes à la **même** fonction. Dans ce cas-ci, la fonction **Affiche** est appelée deux fois. La première fois, la variable **message** vaut "Bienvenue dans l'univers de la BD" et la deuxième fois, la variable **message** vaut "Comment allez-vous ?".

message est une **variable locale** de la fonction **Affiche**. La **valeur** de cette variable dépend de ce qui est transmis dans les parenthèses lors de l'appel de la fonction.

Résultat attendu

Le programme affiche l'une après l'autre les deux valeurs du message. [Voir la page](#)

12 : Conclusion

Les fonctions permettent de **séparer** des parties de code pour rendre le programme plus **interactif**.

Au lieu de **grouper** tout le code dans une seule fonction, on essaie de le **séparer** selon l'action à effectuer.

13 : Appliquez : le jeu du bon numéro

"Reprendre le jeu du bon numéro. Ajouter un bouton permettant de recommencer le jeu : créer une fonction **NouveauJeu()** qui détermine un nombre au hasard entre 0 et 100. Cette fonction est appelée lorsque l'utilisateur clique sur le bouton "nouveau jeu"."

```
<html>
\\r\\n<head>
\\r\\n\\t<title>Le jeu du bon numéro</title>
\\r\\n\\t*script=
\\r\\n\\t\\tvar nombre;
\\r\\n\\t\\tvar proposition;
\\r\\n\\t\\tfunction NouveauJeu(){
\\r\\n\\t\\t}
\\r\\n\\t\\tnombre = Math.round(Math.random()*100);
\\r\\n\\t\\tfunction Main(){
\\r\\n\\t\\t\\tproposition = parseFloat(document.jeu.prop.value);
\\r\\n\\t\\t\\tif(proposition == "" || proposition > 100 || proposition < 0){
```



```
\\r\\n\\t\\t<input type="button" value="Nouveau jeu" onClick="NouveauJeu();"><br>
\\r\\n\\t\\t<input type="Text" name="prop"> <input type="Button" value=" verifier!"
onclick="Main();">
\\r\\n\\t</form>
\\r\\n</body>
\\r\\n</html>
```

Exécuter le programme

Lorsque l'utilisateur clique sur le bouton "nouveau jeu", un nouveau jeu commence avec un nouveau nombre à trouver. [Voir la page](#)

14 : Synthèse

Les fonctions sont utilisées pour **séparer** différents traitements d'un programme. Elles permettent également d'exécuter des instructions lors **d'événements** prédéfinis.

Une variable définie dans une fonction est appelée **variable locale** et n'existe que dans le contexte de la fonction.

Une variable définie en dehors de toute fonction est appelée **variable globale** et reste accessible dans tout le programme.

15 : Quizz (partie 1)

Ce quizz vous permettra de tester vos connaissances sur les fonctions :

1. Lequel de ces noms de fonction est correct ?

- a) _var
- b) lvar
- c) var

2. Quelle balise appelle la fonction nommée "affiche" au chargement de la page ?

- a) <body onload="function affiche() ;">
- b) <body onload="affiche ;">
- c) <body onload="affiche() ;">

3. Comment déclare-t-on la variable message en tant que variable locale à la fonction NouveauJeu()?

- a) var message à placer dans la fonction Main()
- b) var message à placer dans la fonction NouveauJeu()
- c) var message à placer en dehors de toute fonction

16 : Quizz (partie 2)

1. Comment déclare-t-on la variable message en tant que variable globale?

- a) var message à placer dans la fonction Main()
- b) var message à placer dans la fonction NouveauJeu()
- c) var message à placer en dehors de toute fonction

2. Si on oublie de déclarer une variable (oubli du mot-clé "var"), la variable non déclarée est de type :

- a) Globale ou locale selon l'endroit où elle a été créée
- b) Locale
- c) Globale

17 : Conclusion du cours

Et voilà! Nous arrivons au terme de ce parcours de formation.

Nous espérons que le chemin réalisé ensemble fut agréable et qu'il vous donnera envie d'aller plus loin...

18 : Ce qu'on a vu

Tout au long des modules, vous avez appris à créer de **simples programmes**.

Grâce à ces exemples et exercices, vous êtes plus familier avec les notions de programme et de langage de programmation.

Vous avez vu que dans un langage de programmation, on trouve :

- **des variables**

- des opérateurs
- des fonctions
- des structures (itératives, conditionnelles)
- des mots réservés

Maintenant, vous êtes capables d'utiliser toutes ces notions dans un programme.

Vous avez vu comment créer des programmes en JavaScript et comment communiquer avec l'utilisateur.

Vous êtes capable de créer des programmes qui **s'exécutent sur une page Web** et qui communiquent avec l'utilisateur.

Principalement dans le module 3, vous avez rencontré les particularités du langage JavaScript et les **messages d'erreur** possibles.

Vous connaissez les particularités syntaxiques et environnementales du langage JavaScript et vous avez appris à retrouver les erreurs dans un programme.

19 : Mémo

Utiliser des fonctions

Jusqu'ici, nous avons utilisé des **fonctions** sans le savoir, comme **Main()** ou les fonctions associées à des objets comme un bouton (**onclick**), une page (**onload**), ...

Voici comment créer nos propres fonctions, qui contiendront un ensemble d'instructions appelables n'importe où dans notre code:

```
function NomDeLaFonction(){  
    // instructions  
}
```

Encore une fois, **attention aux majuscules**. Vous devez également respecter les parenthèses, qui permettent de **passer une valeur** à la fonction, qui la traitera.

Les noms de fonctions sont soumis **aux mêmes règles que ceux des variables**.

Appeler une fonction

Il suffit de l'appeler par son nom:

- dans la partie HTML:

```

```

- dans le code JavaScript:

```
Affiche(); // on a appelé la fonction Affiche()
```

La portée des variables

Plus tôt dans ce cours, nous avons parlé des variables **locales** et **globales**.

Une variable **locale** n'est utilisable **que dans la fonction dans laquelle elle a été déclarée**, on ne peut pas y accéder en dehors de cette fonction.

Une variable **globale** est utilisable **partout dans le code**, il suffit de la déclarer en dehors de la fonction.

Attention, si vous ne déclarez pas votre variable (vous vous souvenez de la **déclaration implicite**?), celle-ci aura une portée **globale**.

Les parenthèses

Dans votre fonction, les parenthèses servent à transmettre un ou plusieurs paramètres, afin de les traiter:

```
function Affiche(message){  
    alert(message); //message est une variable locale  
}  
//Appel de la fonction  
Affiche("Comment allez-vous ?");
```

La fonction traitera la chaîne de caractères qui lui est envoyée, en l'affichant dans une popup. On peut appeler cette fonction **autant de fois qu'on le désire**, en lui passant des paramètres différents à chaque fois.

20. 1 : Compléments

En complément, nous vous proposons quelques ressources pour aller plus loin.

20. 2 : Quelques ressources

Si cela vous tente d'aller plus loin, il y a encore des sentiers à découvrir ou à approfondir ;-)

Avant tout, avez-vous entendu parler des formations qui se donnent en salle à Technofutur TIC?

Vous trouverez le catalogue sur notre site :

<http://www.technofuturtic.be>

Vous voulez des références sur la programmation en général, sur la logique de programmation ?

<http://www.netalya.com/fr/Algo-intro.asp>

<http://lwh.free.fr/>

Vous voulez apprendre la programmation orientée objet ?

<http://www.commentcamarche.net/poo/poointro.php3>

http://fr.wikipedia.org/wiki/Programmation_orient%C3%A9e_objet

Vous voulez développer vos connaissances en JavaScript et ce que l'on appelle le DHTML (dynamic-HTML) ?

<http://www.toutjavascript.com/savoir/savoir.php3>

<http://www.laltruiste.com/document.php?compteur=3&page=1&rep=10>

A bientôt:-)

Réponses aux quizz

Introduction

- Quizz n° 1 p. 13
- 1. Qu'est-ce qu'une page Web ?**
b) - Un fichier texte dont l'extension est généralement .htm ou .html
 - 2. Qu'est-ce qu'un navigateur ?**
a) - Un outil qui permet de visualiser des pages Web
 - 3. Lequel de ces logiciels n'est pas un navigateur ?**
c) - Wordpad
- Quizz n° 2 p. 13
- 1. Quelle balise trouve-t-on en premier dans un document HTML ?**
a) - <html>
 - 2. Laquelle de ces balises n'est pas obligatoire ?**
b) - <title>
 - 3. Que représente à l'écran ce morceau de code HTML ?<form><input type="Text"></form>**
b) - Un champ texte dans un formulaire

Module 1 - Un premier programme

- Quizz n° 1 p. 19
- 1. Quel outil utilise-t-on pour écrire du code JavaScript ?**
a) - Un éditeur de texte
 - 2. Quel outil utilise-t-on pour "exécuter un programme" écrit en JavaScript ?**
b) - Un navigateur
- Quizz n° 2 p. 25
- 1. La commande qui permet d'afficher un texte dans une fenêtre d'alerte est...**
b) - alert()
 - 2. La commande qui permet d'afficher un texte dans le document est ...**
c) - document.write()
- Quizz n° 3 p. 32
- 1. Lequel de ces commentaires est incorrect ?**
c) - // ce programme alerte l'utilisateur d'un message de bienvenue cette alerte s'affiche automatiquement au chargement de la page //

Module 2 - Communiquer avec la machine

- Quizz n° 1 p. 38
- 1. Lequel de ces formulaires HTML porte un nom correct ?**
b) - <form name="accueil">
 - 2. Laquelle de ces commandes permet de récupérer la valeur du champ "montant" du formulaire "calcul" ?**
c) - document.calcul.montant.value

- Quizz n° 2 p. 41
 - 1. Laquelle de ces commandes récupère le contenu du champ "montant" du formulaire "calcul" dans la variable "prix" ?**
 - a) - prix = document.calcul.montant.value ;
- Quizz n° 3 p. 48
 - 1. Un langage de programmation est**
 - b) - est composé d'instructions qui correspondent à une suite de 0 et de 1 compréhensibles par la machine
- Quizz n° 4 p. 49
 - 1. EcmaScript...**
 - b) - Est le nom repris pour la standardisation du JavaScript
 - 2. On utilise le JavaScript**
 - a) - Principalement dans les technologies du Web
- Quizz n° 5 p. 50
 - 1. La technologie servant à rendre les pages Web plus dynamiques s'appelle**
 - b) - Le DHTML
 - 2. La technologie permettant l'accès aux éléments (objets) du document s'appelle**
 - c) - Le DOM
- Quizz n° 6 p. 51
 - 1. Quelle commande ne récupère pas l'âge de l'utilisateur ?**
 - c) - age = alert("Quel est votre âge ?") ;
 - 2. Trouvez la commande erronée :**
 - a) - age = prompt("Quel est votre âge ?", "insérez votre âge ici", "");
- Quizz n° 7 p. 54
 - 1. Lequel de ces événements ne retrouve-t-on pas dans un élément de formulaire ?**
 - c) - onload
 - 2. L'événement que l'on utilise pour vérifier s'il y a eu un changement de valeur est**
 - b) - onchange
 - 3. A quoi sert l'événement onsubmit ?**
 - b) - A effectuer une action juste avant l'envoi du formulaire

Module 3 - Un peu plus sur le JavaScript

- Quizz n° 1 p. 60
 - 1. Pour qu'un guillemet soit interprété comme du texte, on ajoute devant celui-ci...**
 - a) - message="Vous avez dit \"bizarre\" !" ;
- Quizz n° 2 p. 67
 - 1. Le langage JavaScript est un langage**
 - a) - interprété par le navigateur

Module 4 - Stocker des valeurs

- Quizz n° 1 p. 75
 - 1. Pour les réutiliser, on stocke les données dans une**

a) - Variable

2. Parmi ces 3 noms de variable, lequel est correct ?

c) - `_couleur`

3. Laquelle de ces déclarations est correcte ?

c) - `var mode ;`

- Quizz n° 2 p. 77

1. Laquelle de ces instructions ne récupère PAS la valeur du champ proposition dans la variable injure ?

b) - `document.form.proposition.value = injure ;`

- Quizz n° 3 p. 81

1. Pour "coller" du texte et des variables, on a utilisé l'opérateur :

a) - `+`

2. Quel terme utilise-t-on pour signifier que l'on "colle" du texte et des variables ensemble ?

b) - La concaténation

3. Laquelle de ces instructions est correcte ?

a) - `alert("Les " + prop1 + " de cette bande " + prop2 + " sont des petits " + prop3 + " tout bleus ") ;`

4. Sachant que la variable "compteur" vaut 3, laquelle de ces instructions s'affichera sans erreur ?

c) - `alert("Vous avez gagné " + compteur + " BD d'Astérix et Obélix") ;`

- Quizz n° 4 p. 85

1. En JavaScript, la déclaration d'une variable

a) - Est automatique

2. La déclaration implicite de variables existe

b) - Dans quelques langages

- Quizz n° 5 p. 87

1. Soit la variable message contenant le texte "Bienvenue dans l'univers de la BD !". Laquelle de ces commandes récupère uniquement le mot "Bienvenue" ?

b) - `message.substr(0,9)`

2. Toujours avec la même variable message, quel mot récupère la commande `message.substr(31,2)` ? (Conseil : faites le test dans un petit programme !)

a) - BD

Module 5 - Travailler avec des nombres

- Quizz n° 1 p. 94

1. La commande permettant de convertir du texte en nombre est

c) - `parseFloat()`

2. La valeur récupérée d'un champ de formulaire est de type

b) - texte

3. Pour additionner deux nombres, on utilise l'opérateur

c) - `+`

- Quizz n° 2 p. 99

1. Quelle commande permet de convertir du contenu de type textuel en contenu de type numérique ?

a) - parseFloat()

2. Sachant que a = 3 et b = 5, laquelle de ces instructions représente l'addition de a et b ?

b) - resultat = a + b;

3. Trouvez l'instruction qui convertit correctement des EUR en BEF:

a) - eur = bef / 40,3399 ;

- Quizz n° 3 p. 103

1. A votre avis, le résultat est de type

a) - Texte

- Quizz n° 4 p. 104

1. A votre avis, le résultat est de type

b) - Numérique

- Quizz n° 5 p. 106

1. Soit le code suivant : var num ; alert(typeof num); Quelle est la valeur retournée par l'opérateur typeof ?

c) - undefined

- Quizz n° 6 p. 106

1. Pour récupérer la valeur insérée dans le champ "prix" du formulaire "formBD", la commande est :

c) - document.formBD.prix.value

Module 6 - Ajouter une condition

- Quizz n° 1 p. 110

1. if(prop <= 100){...} Dans cette structure conditionnelle, la condition est

c) - "Si la variable prop est plus petite ou égale à 100"

Module 7 - Répéter une action

- Quizz n° 1 p. 126

1. Une structure itérative est également appelée

a) - Boucle

2. Une structure itérative permet :

b) - D'effectuer plusieurs fois le(s) même(s) traitement(s)

- Quizz n° 2 p. 128

1. Dans une boucle, le compteur permet de

a) - Compter le nombre de répétitions à effectuer

2. Le paramètre qui permet au compteur d'augmenter automatiquement s'appelle

b) - L'incrément

- Quizz n° 3 p. 129

1. Laquelle de ces boucles ne compte pas de 0 à 100 ?

b) - for(i=0;i<100;i=i+1){ document.write(i + " "); } }

- Quizz n° 4 p. 130
 1. La boucle `for(i=0 ; i < 100 ; i = i - 1){...}`
 - a) - Est infinie
 2. La boucle `for(i=0; i < 5; i = i + 5){...}` se répète
 - a) - Une fois

- Quizz n° 5 p. 135
 1. Quelle opération n'est pas équivalente à `i++` ?
 - c) - `i = 1`
 2. Quelle opération est équivalente à `i -= 5` ?
 - a) - `i = i - 5`

Module 8 - Grouper plusieurs valeurs

- Quizz n° 1 p. 143
 1. L'indice minimum d'un tableau est :
 - b) - 0
 2. Si on veut atteindre le titre du tome 3 du petit Spirou, quelle commande doit-on utiliser ?
 - a) - `spirou[2]`

- Quizz n° 2 p. 144
 1. Soit un tableau contenant 12 valeurs, quelle est la valeur de l'indice maximum ?
 - a) - 11
 2. `var jour= new Array() ; jour = ["lundi","mardi","mercredi","jeudi","vendredi","samedi","dimanche"] ;` Quelle est la taille de ce tableau ?
 - b) - 7
 3. Quelle commande permet de trouver l'indice maximum d'un tableau ?
 - b) - `IndiceMax = tableau.length - 1`

- Quizz n° 3 p. 145
 1. Pour parcourir toutes les données d'un tableau, on utilise
 - a) - Une structure itérative
 2. Pour parcourir TOUTES les données d'un tableau, on utilise l'incrément :
 - b) - `i = i + 1`

Module 9 - Utiliser des fonctions

- Quizz n° 1 p. 153
 1. Lequel de ces noms de fonction est incorrect ?
 - b) - `function`
- Quizz n° 2 p. 153
 1. Lequel de ces morceaux de code n'appelle pas la fonction "`AfficheMessage`" ?
 - a) - `<input type="Button" value=" " onclick="AfficheMessage;">`
- Quizz n° 3 p. 156
 1. Dans le code suivant : `function Main(){var i = 0 ;...}` La variable `i` ...
 - a) - N'est pas accessible en dehors de la fonction `Main()`

2. Dans ce code, laquelle de ces variables n'est pas globale ?...eur = 40,3399 ; function Main(){var message ; message = "Bienvenue" ; message2 = "Au revoir ; "}...

a) - message

- Quizz n° 4 p. 159

1. Lequel de ces noms de fonction est correct ?

a) - _var

2. Quelle balise appelle la fonction nommée "affiche" au chargement de la page ?

c) - <body onload="affiche() ;">

3. Comment déclare-t-on la variable message en tant que variable locale à la fonction NouveauJeu()?

b) - var message à placer dans la fonction NouveauJeu()

- Quizz n° 5 p. 160

1. Comment déclare-t-on la variable message en tant que variable globale?

c) - var message à placer en dehors de toute fonction

2. Si on oublie de déclarer une variable (oubli du mot-clé "var"), la variable non déclarée est de type :

c) - Globale

