

République Algérienne Démocratique et Populaire
Ministère de L'Enseignement Supérieur et de la Recherche Scientifique

Université Mouloud Mammeri De Tizi-Ouzou



Faculté De Génie Electrique Et D'informatique
DEPARTEMENT D'AUTOMATIQUE

**Mémoire de Fin d'Etudes
de MASTER ACADEMIQUE**
Spécialité : **Automatique et Informatique Industrielle**

Présenté par
Kemmoum NACER
Houchi YOUNES

Mémoire proposé et dirigé par Mr **Ahmed MAIDI**

Thème

**Développement d'un laboratoire virtuel
en utilisant les langages HTML et
JavaScript**

Mémoire soutenu publiquement le 27 / 06 / 2018 devant le jury composé de :

Mr Redouane KARA
Professeur, UMMTO, Président

Mr Ahmed MAIDI
Professeur, UMMTO, Rapporteur

Mme Ouerdia CHILALI
Maître de conférence B, UMMTO, Examineur

Mr M'hamed DAOUI
Maître de conférence A, UMMTO, Examineur

Table des matières

Introduction générale	8
1 Simulation et laboratoires virtuels	10
1.1 Introduction	10
1.2 Mise en œuvre et déploiement des VRLs (Virtual Remote Labs)	13
1.2.1 Laboratoires virtuels	13
1.2.2 Laboratoires à distance RLs (remote labs)	15
1.3 Améliorer l'interactivité du laboratoire	16
1.4 Augmenter les options de contrôle	17
1.5 Conclusion	22
2 Le langage HTML	24
2.1 Introduction	24
2.2 Définitions	24
2.2.1 HTML	24
2.2.2 Les version de HTML	25
2.2.3 Les balises	25
2.3 Création d'une simple page HTML	26
2.3.1 DOCTYPE	27
2.3.2 HEADING	28
2.3.3 Principaux types d'encodage	29
2.3.4 Les paragraphes	29
2.3.5 Saut de ligne et espace	30
2.3.6 Listes à puces ou numérotées	30
2.3.7 Formatage de texte	31
2.3.7.1 Surbrillance du texte (Highlighting)	31
2.3.7.2 Gras, Italique et Souligné	32
2.3.7.3 L'abréviation	33
2.3.7.4 Inséré, Supprimé et Barré	33
2.3.7.5 Exposant et Indice	33
2.4 Les Liens	34

2.4.1	Un lien vers un autre site	34
2.4.2	Lien vers une ancre dans la même page	35
2.4.3	Lien vers une ancre dans une autre page	36
2.4.4	Lien vers une page dans le même site	36
2.5	Les Tableaux	37
2.5.1	Portée des lignes et des colonnes	38
2.5.2	Tableau avec thead, tbody, tfoot et caption	39
2.6	Création des images	41
2.6.1	Utilisation de l'attribut de srcset	42
2.7	CSS	42
2.7.1	Emplacement des styles	42
2.7.2	Principes d'une bonne écriture HTML/CSS	43
2.7.3	L'utilisation de HTML avec CSS	43
2.7.4	Charger plusieurs feuilles de style	45
2.8	Conception page HTML/CSS	45
2.9	Conclusion	48
3	Le langage JavaScript	50
3.1	Introduction	50
3.2	Définitions	50
3.2.1	Langage de programmation	50
3.2.2	JavaScript	50
3.2.3	Exemple d'introduction	51
3.2.4	Commentaires	53
3.2.5	Variables	54
3.2.5.1	Types de variables	54
3.2.6	Opérateurs arithmétiques	55
3.2.7	Opérateur d'affectation	56
3.2.8	Opérateur d'incrément et de décrémentation	58
3.2.9	Opérateurs de comparaison	59
3.2.10	Structures conditionnelles	60
3.2.10.1	Tests conditionnels avec if et else	60
3.2.10.2	L'instruction switch/case	63
3.2.11	Boucles conditionnelles	65
3.2.11.1	boucle while	66
3.2.11.2	boucle do . . . while	66
3.2.11.3	boucle for	67
3.2.11.4	Instruction break	69
3.3	Tableaux	70

3.3.1	Accès aux valeurs d'un tableau	71
3.3.2	Des outils pratiques	72
3.3.3	Parcourir un tableau	73
3.4	Objets	74
3.4.1	Définition	74
3.4.2	Constructeur	75
3.4.2.1	Syntaxe d'un constructeur	75
3.4.2.2	Mot clé "this"	75
3.4.2.3	Mot clé "new"	76
3.4.3	Propriétés	77
3.4.3.1	Modifier les données	78
3.4.4	Méthodes	79
3.4.4.1	Ajout des méthodes	80
3.4.4.2	Définir une méthode dans le constructeur	80
3.4.4.3	Ajout d'une méthode via prototype	81
3.4.5	Accès aux propriétés et aux méthodes	81
3.4.6	Déclaration d'un objet littérale	81
3.4.6.1	Ajout des éléments	84
3.4.6.2	Parcourir un objet	84
3.5	Boite d'alerte	87
3.5.1	Caractéristiques	87
3.6	Boite de confirmation	89
3.6.1	Caractéristiques	89
3.7	Boite de saisie	91
3.7.1	Caractéristiques	91
3.8	Fonctions	92
3.9	Exemples de fonction mathématiques	95
3.10	Application JavaScript	98
3.11	Conclusion	101
4	Développement d'applications d'automatique	102
4.1	Introduction	102
4.2	Types généraux de régulateurs	102
4.2.1	Régulateur proportionnel	102
4.2.2	Régulateur proportionnel-intégral	103
4.2.3	Régulateur proportionnel-dérivé idéal	104
4.3	Régulateur proportionnel-intégral-dérivé idéal	104
4.4	Rôle du régulateur	105
4.5	L'algorithme de Runge-Kutta	105

4.6	Commande d'un système de premier ordre avec un PI	107
4.7	Simulation d'un système masse-ressort-amortisseur	111
4.8	Conclusion	112
Conclusion générale		113
Annexe		115

Table des figures

1.1	Les types d'environnements expérimentaux.	12
1.2	Laboratoire virtuel où le contrôleur est configuré par des paramètres fixes	
	[9].	18
1.3	Laboratoire virtuel flexible avec un interpréteur de langage de programma-	
	tion intégré [9].	19
1.4	Un éditeur d'expérimentation où les étudiants peuvent éditer et exécuter	
	des contrôleurs VRL [10].	20
1.5	Options de déploiement de contrôleur.	22
2.1	Exemple d'une simple page HTML.	27
2.2	La hiérarchie des en-têtes en HTML.	29
2.3	Large espace et saut de ligne en HTML.	30
2.4	Listes à puces et listes numérotées.	31
2.5	Surbrillance du texte.	32
2.6	Souligné.	33
2.7	Abréviation.	33
2.8	Liens vers un autre site.	34
2.9	Lien vers une ancre.	35
2.10	Tableau de Liens vers des ancres.	36
2.11	Tableau.	38
2.12	Porté des lignes et des colonnes dans un tableau.	39
2.13	Tableau avec thead,tbody,tfoot et caption.	41
2.14	Interface principale d'un laboratoire virtuel.	46
2.15	Page HTML sous forme d'un laboratoire virtuel.	47
2.16	Page HTML sous forme d'un laboratoire virtuel.	48
3.1	Explication du fonctionnement client-side.	51
3.2	La fonction console.log dans la console du navigateur.	52
3.3	Déclaration et affichage d'une variable.	52
3.4	Introduction de plusieurs valeurs.	53
3.5	Opérateur de concaténation « + ».	56

3.6	Opérateur de concaténation « += ».	56
3.7	Opérateur d'affectation.	57
3.8	Opérateur d'affectation pour une expression algébrique.	58
3.9	Distributeur Automatique.	62
3.10	Distributeur Automatique.	62
3.11	Instruction switch/case.	65
3.12	Instruction switch/case.	65
3.13	La boucle while.	66
3.14	La boucle do while.	67
3.15	Simple compteur.	69
3.16	L'instruction break.	70
3.17	Accès au valeur d'un tableau.	72
3.18	Accès au valeur d'un tableau.	72
3.19	Parcourir un tableau.	74
3.20	Le mot clé this.	76
3.21	Le mot clé instanceof.	77
3.22	Récupération du paramètre « nom » de l'objet « Personne ».	77
3.23	La propriété length.	78
3.24	Modification des données.	79
3.25	Affichage du résultat via la boîte alert.	80
3.26	Objet littérale.	82
3.27	Récupération de la valeur d'un élément.	83
3.28	Ajouter des éléments.	84
3.29	Parcourir un objet.	86
3.30	Boîte d'alerte avec Google Chrome.	87
3.31	Boîte d'alerte avec Mozilla Firefox.	88
3.32	Boîte d'alerte avec Internet Explorer.	88
3.33	Boîte de confirmation.	89
3.34	La fonction confirm renvoie true.	90
3.35	La fonction confirm renvoie fals.	90
3.36	Boîte de confirmation et alert combinée.	90
3.37	Boîte de confirmation et alert combinée.	91
3.38	Boîte de saisie.	92
3.39	Exemple d'une fonction.	93
3.40	Résultat de la fonction jour().	94
3.41	Fonction qui affiche une phrase.	94
3.42	Fonction de fibonnaci.	96
3.43	Aire et circonférence d'un cercle.	97
3.44	Calculateur de température de point de rosée.	98

3.45	Application JavaScript pour tracer les courbes des fonctions mathématiques.	100
4.1	Commande d'un système de premier ordre avec un PI.	107
4.2	Simulation d'un système de premier ordre en boucle ouverte avec un PI.	109
4.3	Simulation d'un système de premier ordre en boucle fermée avec un PI.	110
4.4	Système masse-ressort-amortisseur.	111
4.5	Application masse-ressort amortisseur.	112

Introduction générale

Un laboratoire virtuel (V.L) est défini comme "un espace de travail numérique pour la collaboration à distance et l'expérimentation dans la recherche ou dans d'autres activités créatives en vue de générer et de diffuser des résultats au moyen de technologies partagées de l'information et de la communication" [16].

Au sens le plus large du terme, un VL est une collaboration centrée sur la réalisation d'objectifs particuliers, créatifs et/ou destinés à étayer une décision. De la sorte, le VL peut englober pratiquement tous les domaines.

Les formes d'enseignement classiques, les infrastructures, les sollicitations extérieures (manque de matériel, nombre élevé d'étudiants, perturbations, greve ...) ou simplement les variations d'attention de l'étudiant (mise en place de cours sans pause pour "questions-réponse") peuvent nuire tant à son apprentissage qu'à la motivation de l'enseignant.

Les domaines scientifiques et techniques sont basés sur la mise en pratique d'acquis théoriques à travers la manipulation d'objets et d'instruments. Les travaux pratiques (T.P) prévus à cet effet nécessitent un temps privilégié, des laboratoires équipés avec des grandes infrastructures adaptées pour accueillir un grand nombre d'étudiant qui ne cesse de croître. Les grands effectifs inscrits dans les premières années d'études supérieures universitaires et les limitations matérielles (expériences coûteuses, des manipulations dangereuses, etc.) et humaines rendent délicate l'acquisition des savoirs par la réalisation des T.P.

Face aux difficultés que rencontrent les laboratoires classiques traditionnels (manque de temps et de moyens pour l'acquisition des savoirs en T.P), on propose une solution qui envisage une stratégie pédagogique intégrant un laboratoire virtuel (V.L). Le but est de pouvoir simuler, à domicile ou n'importe où, des expériences réelles distantes (ex : commande d'un robot à distance) ou virtuelles par le biais d'une communication par internet.

Pour mieux cerner notre étude, on a jugé utile de répartir notre travail comme suit :

— Le premier chapitre traite la simulation via des laboratoires virtuels, leur mise en

œuvre et leur déploiement ainsi que les objectifs qu'ils permettent d'atteindre.

- Le deuxième chapitre expose le langage de balisage HTML (HyperText Markup Language) conçu pour développer les pages web.
- Le troisième chapitre est consacré au langage de programmation JavaScript, sa structure et son rôle dans l'interactivité et la dynamique des sites web.
- Le quatrième chapitre concerne la partie pratique qui porte sur la mise au point de deux applications de simulation. La première concerne un système de premier ordre commandé par un PI. Et la deuxième concerne un système masse-ressort-amortisseur.
- Enfin nous terminons par une conclusion générale.

Chapitre 1

Simulation et laboratoires virtuels

1.1 Introduction

Chaque science est principalement basée sur deux approches [1] : la première est basée sur le domaine pratique, tandis que l'autre est basée sur la théorie et les mathématiques. De nos jours les ingénieurs en automatique ont besoin d'avoir à la fois une compréhension approfondie des mathématiques qui sont derrière des concepts de l'automatique, et une vaste expérience pratique pour implémenter les solutions théoriques pour résoudre des problèmes réels. L'approche basée sur l'expérience pratique repose sur l'idée que quelque chose a besoin d'être contrôlé, donc les programmes d'ingénierie des systèmes de contrôle doivent être pratiques et basés sur la pratique. Ceci a été une vision traditionnelle de l'ingénierie, jusqu'à ce que la deuxième approche basée sur la théorie et les mathématiques commence à gagner de l'importance.

La théorie est basée sur les quatre concepts abstraits identifiés par [1], comme des principaux concepts sur les systèmes de contrôle tel que : système dynamique, stabilité, retour d'état et compensation dynamique, qui sont mieux modélisés mathématiquement. Par conséquent l'expérimentation en laboratoire joue un rôle clé pour relier la théorie et la pratique entre eux. Entre autres les laboratoires de contrôle permettent d'atteindre les objectifs suivants :

- Démontrer, valider et motiver des concepts analytiques.
- Introduction au monde réel de control, la modélisation tel que la saturation, le bruit, capteur, dynamique des actionneurs, l'incertitude, etc.
- Fournir des systèmes avec des outils d'instrumentation et de mesure.
- Exposer les étudiants à des problèmes de conception plus large comme : problème de la spécification à l'implémentation du matériel et aux considérations économiques.
- Exposer les étudiants à la pratique professionnelle et la rédaction des rapports.

- Le travail en équipes et la résolution des problèmes.
- Comparer les résultats théoriques avec les résultats réels ce qui permet de valider la théorie.

Les laboratoires pratiques traditionnels impliquent des coûts élevés dû à l'équipement, l'espace et l'équipe de maintenance. C'est pour cela qu'une ligne de recherche qui est en croissance depuis les vingt dernières années cherche à réduire les coûts de ces laboratoires en profitant des avantages d'internet, c'est-à-dire en remplaçant les laboratoires traditionnels par des laboratoires virtuels. Pour caractériser les différentes modalités des environnements d'expérimentation et ainsi fournir une définition précise des laboratoires virtuel, deux critères ont été proposés dans des travaux antérieurs :

1. Selon la façon dont les ressources sont utilisées à des fins expérimentales, les environnements peuvent être locaux ou éloignés (à distance).
2. Selon la nature physique du laboratoire, les environnements peuvent être réels ou bien simulés.

En combinant ces deux critères, on peut avoir les quatre types pour environnement expérimental suivants (Figure 1.1) :

1. Accès local aux ressources réels : cette combinaison représente des laboratoires pratiques traditionnels ou l'étudiant est devant un ordinateur relié directement au système.
2. Accès local aux ressources simulées : tout l'environnement est du software (logiciel) et l'interface d'expérimentation travaille sur une ressource physiquement inexistante et virtuel qui est en même temps avec l'interface font partie de l'ordinateur.
3. Accès à distance aux ressources réels : l'équipement réel du système est accessible via internet. L'utilisateur opère et contrôle à distance un système réel via une interface d'expérimentation. Cette approche est dite laboratoire réel distant.
4. Accès à distances aux ressources simulées : cette forme d'expérimentation est similaire au point 3, la différence est qu'on remplace le système physique par un modèle mathématique. L'étudiant opère avec l'interface d'expérimentation sur un système virtuel accessible via internet. Comme il s'agit d'un processus simulé il peut être instancié pour servir quiconque le demande. C'est ce qu'on appelle éventuellement un laboratoire virtuel distant (VRLs). En comparaison avec les ressources locales simulées, les laboratoires virtuels permettent : Un découplage du modèle (qui peut marcher sur le serveur) et de la vue (qui peut marcher sur le client) ce qui supporte

l'introduction immédiate des expériences de contrôle avec des délais inconnus variant dans le temps et un travail collaboratif en ligne.

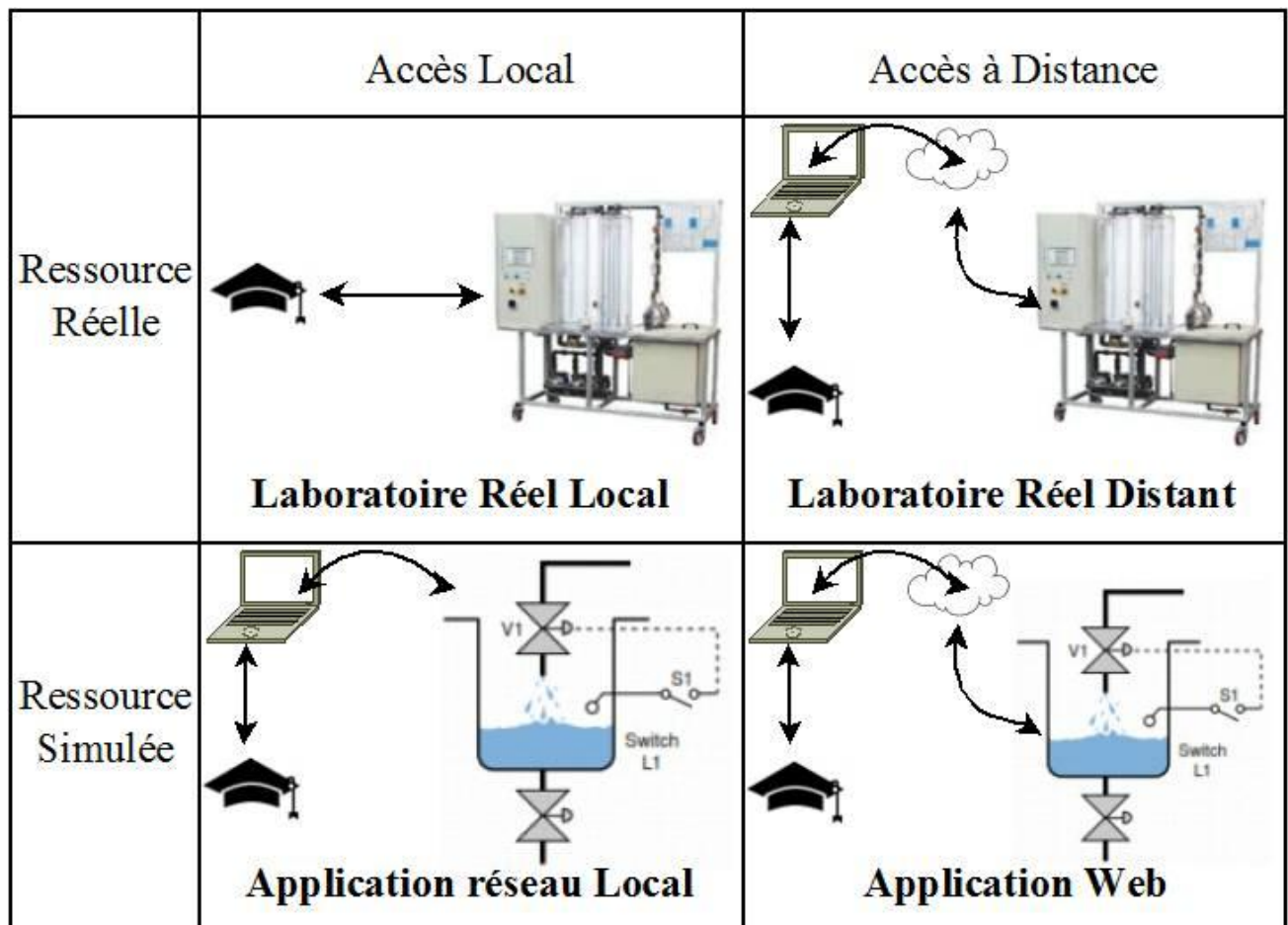


FIGURE 1.1 – Les types d’environnements expérimentaux.

L'importance et l'utilisation des laboratoires virtuels distants (VRLs) ont augmentés au fil des années au fur et à mesure que la technologie progresse. Lors de la conception initiale des laboratoires virtuels, l'une des préoccupations consiste à évaluer si les VRLs étaient capables de fournir des résultats d'apprentissage comparable à ceux des laboratoires pratiques traditionnels. La plupart des études ont montrées que les VRLs et les laboratoires traditionnels sont tout aussi efficaces. De plus les VRLs offrent des avantages supplémentaires suivants :

1. Disponibilité : Les VRLs peuvent être utilisés partout et à tout moment par les étudiants qui sont dispersés géographiquement et qui sont conditionnés à différents fuseaux horaires.
2. Observabilité : Les sessions de laboratoires peuvent être regardées par beaucoup de personnes et peuvent être enregistrées.

3. Accessibilité : Les laboratoires peuvent être accessibles aux personnes handicapés.
4. Sécurité : Les VRLs sont une meilleure alternative (aucun danger) aux laboratoires traditionnels pour les expériences dangereuses.

1.2 Mise en œuvre et déploiement des VRLs (Virtual Remote Labs)

Pendant des années, l'effort multidisciplinaire requis pour développer et construire des laboratoires virtuels éloignés, surtout avec le manque d'outils nécessaires pour leurs créations, a abouti à la création d'un seul VRL pour une expérience de système de contrôle complexe supposé comme un énorme effort dans le passé. Par exemple, [4] a développé un laboratoire virtuel distant pour le contrôle d'un pendule inversé.

Au fur et à mesure que le temps passait et que la technologie évoluait, la création des VRLs a cessé d'être un processus pénible dans le quel aucune réutilisation ne pouvait être obtenue, et permettait aux enseignants de se concentrer d'avantage sur l'aspect pédagogique du VRLs que sur l'aspect technique de sa construction.

1.2.1 Laboratoires virtuels

Les laboratoires virtuels peuvent être implémentés sous forme d'un programme de bureau (fonctionnant sur le système d'exploitation de l'utilisateur) ou sous forme d'application web (exécutés sur le navigateur de l'utilisateur). Par rapport aux applications web, les programmes de bureau possèdent les inconvénients suivants :

1. Ils sont moins portables : Les applications de bureau sont limitées à une plateforme particulière, par exemple : Un système d'exploitation, la machine virtuelle java, le NET Common Language Runtime, etc.
2. Ils sont moins sécurisé : Les applications de bureau ont constamment besoin d'un accès privilégié complet pour être installé, elles ont un accès complet au disque dur de l'utilisateur et elles peuvent ouvrir des connections illimités (libres) vers le monde extérieur, etc.

Par conséquent, la plupart des VLs (Virtual Labs) sont mis en œuvres en tant qu'application basé sur le web. En général, une fois que le VL est en cours d'exécution sur le navigateur de l'utilisateur, il ne demande pas d'établir de nouvelles connexions à l'extérieure. Néanmoins l'architecture du VL peut devenir plus complexe dans certains cas

exceptionnels :

- Si la simulation a un coût de calcul excessif pour l'ordinateur de l'étudiant (ou tablette, téléphone portable, etc.), alors une architecture client-serveur peut être utilisée. Par exemple [2] proposent de lancer une interface graphique légère sur le client qui permettra de contrôler les calculs effectués sur le serveur.
- Pour permettre aux étudiants dispersés géographiquement de travailler en toute collaboration sur le même VL, [3] proposent la création d'un réseau Peer to Peer (P2P) entre les navigateurs des étudiants. Dans chaque réseau, l'enseignant ou l'un des étudiants joue le rôle de directeur de session, étant responsable du démarrage et de la clôture de la session de collaboration, les participants restants sont autorisés seulement à voir en temps réel ce que le directeur fait dans la simulation partagée, à moins que le directeur ne cède momentanément le contrôle du VL à un autre participant. Le réseau P2P est centralisé autour du directeur de session, ainsi toutes les instances de simulation s'exécutant sur les navigateurs des participants sont synchronisées en fonction de l'instance du directeur. De cette façon, les conflits d'événements que peuvent entraîner des résultats non désirés et incohérents sont évitées.

Programmer un VL à partir de zéro nécessite un effort considérable et un travail répétitif. Par exemple, une exigence de base pour les VLs est d'être dynamique et interactif : tout changement dans les paramètres de simulation doit être immédiatement reflété dans l'interface utilisateur graphique. Ainsi, les étudiants peuvent visualiser à la volé comment le comportement du modèle évolue en fonction des valeurs des paramètres interactifs. La programmation de l'interactivité n'est pas triviale et elle doit être faite pour chaque VL. Pour éliminer ce travail redondant et augmenter la productivité dans le développement VL, il existe des outils de création, telle que l'EjsS qui offre des interfaces graphiques de haut niveau pour créer des simulations sur ordinateur. Grâce à EjsS, les utilisateurs faibles en programmation peuvent spécifier le modèle mathématique du VL, et la manière dont il doit être visualisé. Ensuite EjsS génère automatiquement le code qui implémente le laboratoire spécifié.

Afin de prendre en charge l'expérience réaliste de l'utilisateur, les VLs doivent fournir des médias enrichis, tels que les graphiques 3D, la vidéo, audio, etc. Malheureusement, ces fonctionnalités n'ont pas été pris en charge par HTML depuis longtemps. Pour surmonter les limitations de l'HTML, le choix le plus commun pour développer les VLs a été les applets (microprogramme) java. De nos jours, comme les applets java sont progressivement non supportées par la plupart des navigateurs web et que la nouvelle norme JavaScript (HTML5, WebGL, etc.) fournit des médias enrichis, la tendance actuelle s'est déplacée

vers le JavaScript. L'inconvénient de ce changement de technologie est qu'il existe de nombreuses bibliothèques java pour l'ingénierie des systèmes de contrôle qui facilitent la création des laboratoires virtuels dans ce domaine par exemple : [11] propose des classes pour la création des boucles ouvertes et fermées, les processus du premier et second ordre ou les contrôleurs PID n'existent pas encore en JavaScript.

1.2.2 Laboratoires à distance RLs (remote labs)

Puisque les laboratoires à distance (RLs) télé opère du matériel réel, ils sont généralement plus chers et plus complexes que les VLs. Pour surmonter une telle complexité les RLs suivent une architecture client-serveur.

Le coté client du RL est généralement une application web qui interagit avec le coté serveur pour télécommander la configuration actuelle et visualiser les informations du laboratoire (streaming vidéo, données de capteur, etc.).

Les VLs et le coté client des RLs partagent de nombreuses fonctionnalités communes (applications internet riche pour l'expérimentation en laboratoire), et donc les deux sont généralement implémentés avec la même technologie (java dans le passé et JavaScript de nos jours). En outre, certaines approches tirent parti de ces similitudes avec les laboratoires virtuels et distants de Blender en superposant une représentation graphique du comportement du VL à l'image fourni par la webcam du laboratoire actuel. Ceci permet une comparaison visuelle de la réponse théorique de l'équipement avec la réponse réelle. De telles approches sont appelés VRLs avec une réalité augmentée.

Traditionnellement, le côté serveur du RL (remote lab) a été implémenté sous la forme de programmes Lab-View ou Matlab / Simulink fonctionnant sur des ordinateurs PC. Cette approche est assez coûteuse, car elle nécessite :

1. De dédier un ordinateur PC pour chaque laboratoire.
2. De payer les licences logicielles de LabView ou Matlab (Simulink).

Heureusement, le scénario matériel change rapidement. Il est maintenant possible de remplacer l'ordinateur PC par des options beaucoup plus abordables telles que BeagleBone, Arduino ou Raspberry Pi. Par exemple, [2] décrivent le développement de VRLs à faible coût sur l'ingénierie de commande (deux entraînements électriques couplés, un moteur à courant continu et un système de flux de chaleur) utilisant une carte BeagleBone Black directement connectée via un serveur Node.js avec le coté-client implémenté avec EjsS. De cette façon, les coûts logiciels disparaissent car LabView et Matlab (Simulink) sont remplacés par Node.js et EjsS qui est gratuit. De même, [5] présentent également une

solution pour les laboratoires distants à faible coût utilisant Raspberry Pi et Node.js pour appliquer le contrôle PID à un mono-rotor vertical.

1.3 Améliorer l'interactivité du laboratoire

On distingue trois types d'expériences en ligne [6] :

1. Expérience par lots où les étudiants spécifient à l'avance l'expérience et soumettent la spécification comme une demande au serveur. Ainsi, la spécification est exécutée sans aucune intervention et le résultat est renvoyé à l'étudiant.
2. Expériences de capteurs où les étudiants surveillent ou analysent les flux de données en temps réel sans influencer les phénomènes mesurés.
3. Expériences interactives où les étudiants contrôlent et surveillent l'expérience pendant son exécution.

La majorité des VRLs sont conçus pour des expériences interactives. Il existe peu de laboratoires orientés vers des expériences par lots ou vers des capteurs, et la plupart d'entre eux ont été développés une décennie auparavant. De plus les expériences par lots et par capteurs n'ont pas vraiment de sens dans les applications et les expériences des systèmes de commande, dans lesquelles le besoin d'une boucle de contrôle implique une certaine forme d'interactivité. De plus la tendance progresse vers des niveaux d'interactivité croissants. Les points suivants décrivent certaines recherches qui mettent en évidence une telle tendance :

- Traitement des VRLs via les haptiques : les haptiques sont des dispositifs sensibles à la force capable d'actionner à distance du matériel réel ou d'interagir avec des modèles de réalité virtuelle, en renvoyant les informations de feedback de la force de réaction à l'action de l'utilisateur. L'introduction de l'Haptique dans l'expérimentation en ligne peut accroître l'attrait et l'intérêt des étudiants. Par exemple : [7] décrivent un VL contrôlé par un dispositif haptique pour expérimenter diverses forces de la physique, telles que la force de frottement, la force aérodynamique ou la force gravitationnelle. Traditionnellement, les haptiques étaient coûteux. Néanmoins, des recherches sont menées pour proposer des solutions moins coûteuses, par exemple : [8] décrivent la construction d'un dispositif haptique à retour de force à faible coût d'un « 1 » degré de liberté. Avec la publication open-source des conceptions haptiques et le prix réduit de l'impression 3D, les haptiques pourraient bientôt devenir une technologie très répandue pour gérer les VRLs, mais il reste encore à voir si les dispositifs haptiques peuvent s'avérer utiles ou non.

- Gamification des VRLs : L'application des principes du jeu dans les contextes non-jeu afin d'améliorer l'engagement de l'utilisateur, l'apprentissage, l'évaluation, etc., est connu sous le nom de gammification. Dans le contexte de l'éducation, certains exemples de la gamification sont *suralimenté* et *Robocode*, qui sont des jeux pour comprendre les concepts électromagnétiques et apprendre la programmation informatique, respectivement.

1.4 Augmenter les options de contrôle

Les principaux éléments d'une boucle de régulation : le modèle du système et le contrôleur (Correcteur) ont traditionnellement été fortement couplés dans les VRLs. Par exemple : la Figure 1.2 montre le laboratoire virtuel d'un système de premier ordre. Le laboratoire inclut un correcteur PI, et bien que les utilisateurs puissent configurer ses paramètres, ils ne peuvent pas modifier le correcteur lui-même. Dans les laboratoires dotés d'une telle architecture monolithique, les étudiants peuvent uniquement jouer avec un jeu de paramètres réduit pour influencer le comportement du système ou tout au plus, ils peuvent choisir entre différents correcteurs pré-implémentés. Du point de vue des développeurs de VRL, il s'agit d'une solution facile à mettre en œuvre. Cependant, cette configuration limitée n'est pas suffisante pour les étudiants qui doivent concevoir et tester des lois de contrôle avancées.

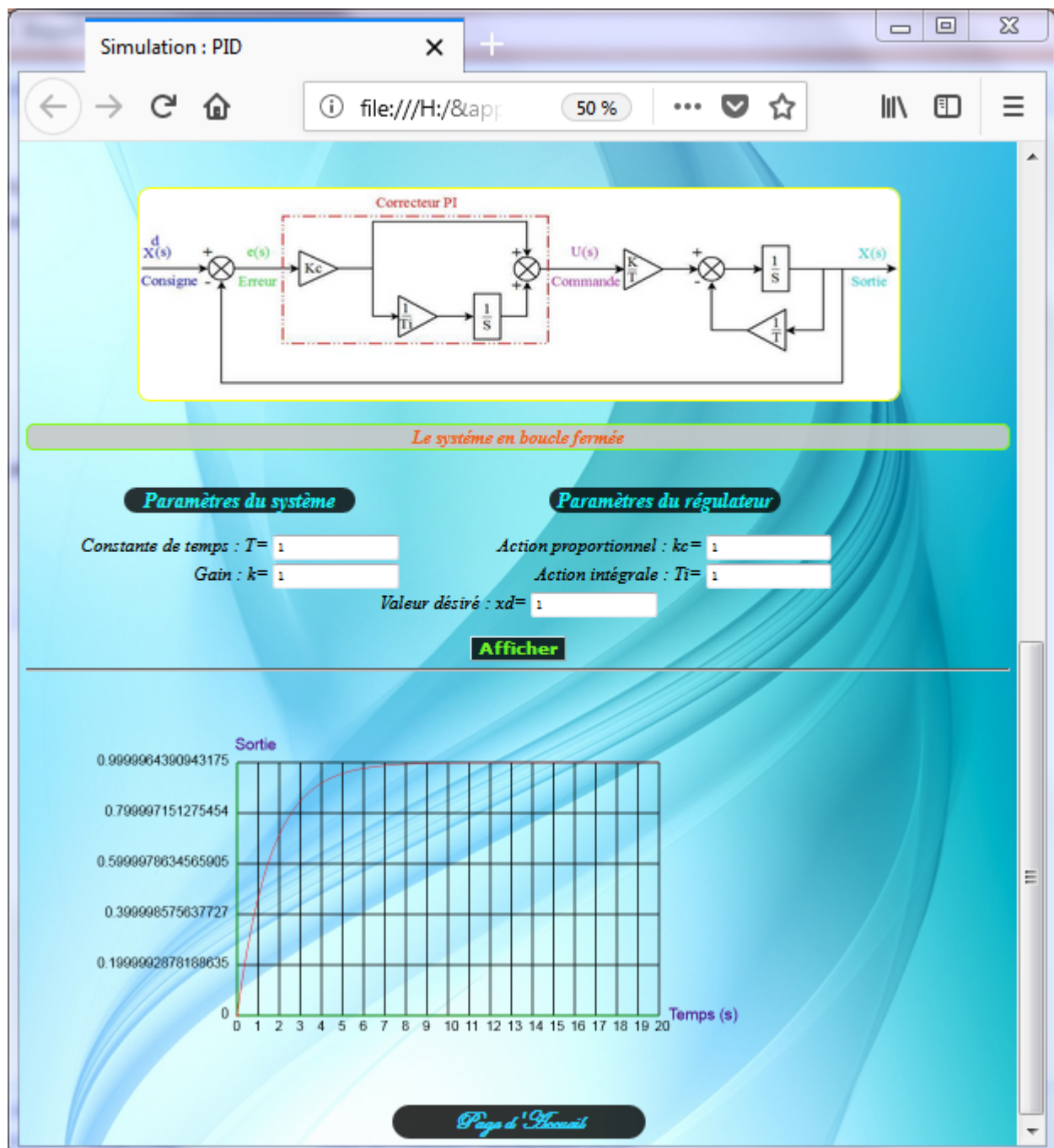


FIGURE 1.2 – Laboratoire virtuel où le contrôleur est configuré par des paramètres fixes [9].

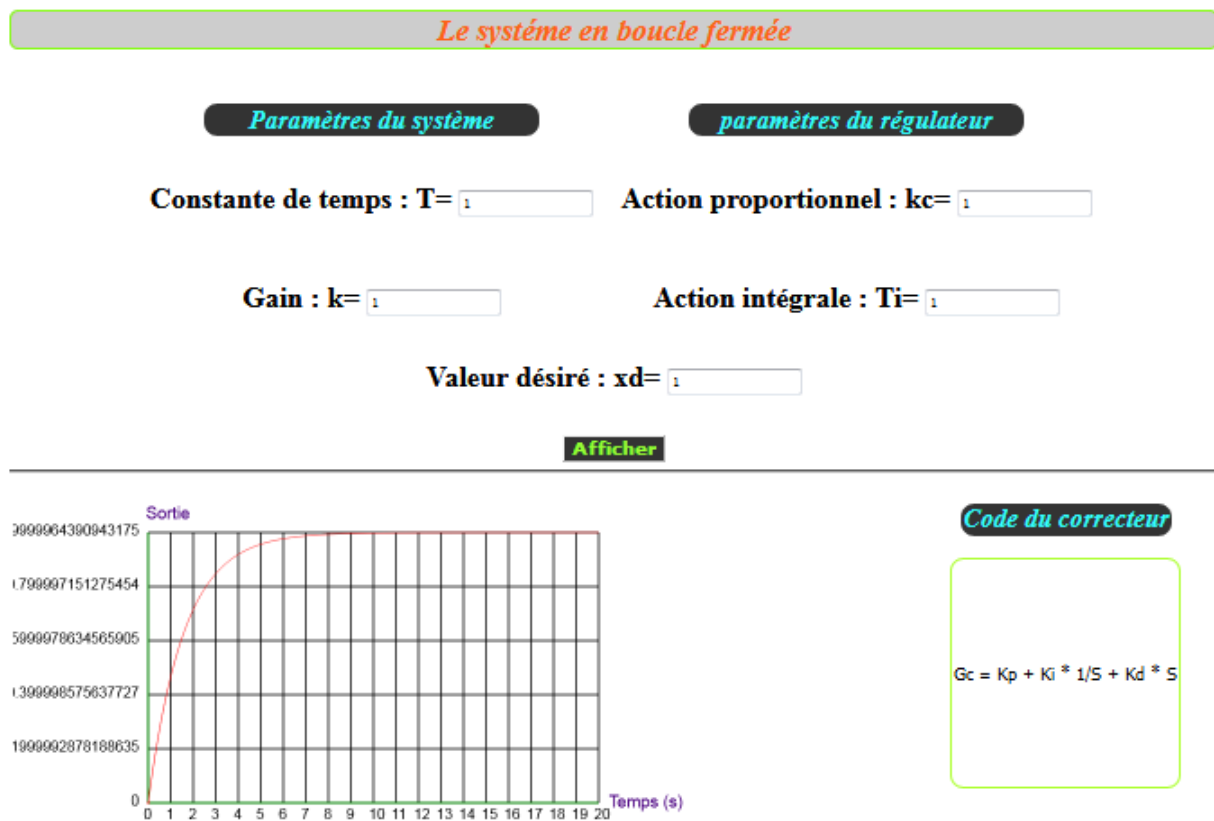


FIGURE 1.3 – Laboratoire virtuel flexible avec un interpréteur de langage de programmation intégré [9].

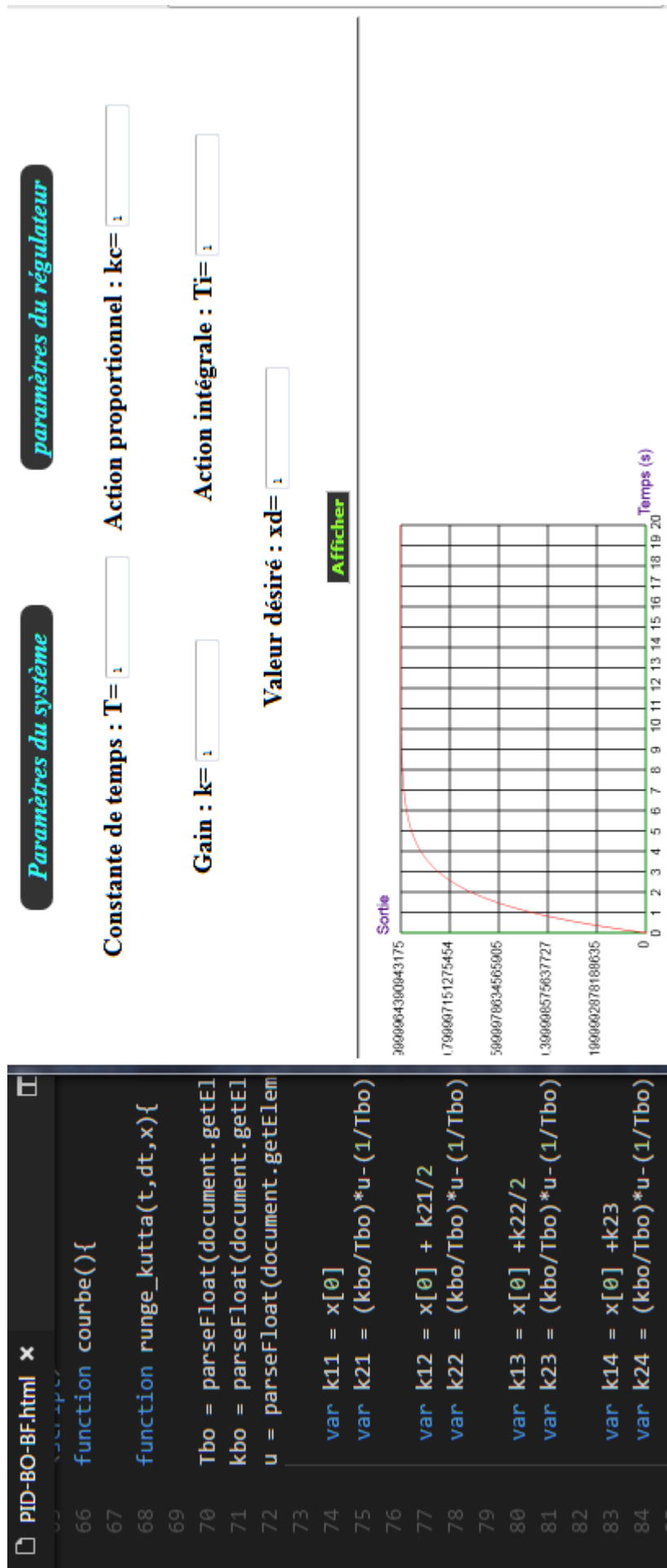


FIGURE 1.4 – Un éditeur d'expérimentation où les étudiants peuvent éditer et exécuter des contrôleurs VRL [10].

Récemment, deux approches ont été proposées pour flexibiliser les VRLs en découplant le contrôleur du modèle du système. Les deux approches aident les étudiants à écrire le code du contrôleur et à l'exécuter sans recompiler l'implémentation de VRL :

- [9] proposent d'intégrer des interprètes de la programmation dans les VRLs. Par exemple, le laboratoire virtuel de la Figure 1.3 comprend un champ de texte dans lequel l'étudiant peut introduire le type du contrôleur.
- [10] ont développé une application de bureau, appelée l'éditeur d'expérimentation, où l'étudiant peut charger des VRLs. Ensuite, en utilisant un langage interprété, l'étudiant peut écrire des expérimentations ou changer le comportement du laboratoire en ajoutant de nouvelles fonctionnalités ou en réimplantant une partie du laboratoire. Par exemple, sur la Figure 1.4, le laboratoire du contrôleur a été chargé dans l'éditeur d'expérimentation qui est ensuite utilisé pour modifier le comportement du laboratoire.

Comme le montre la figure 1.5, une fois que le contrôleur a été découplé du système, de nouvelles possibilités apparaissent pour l'architecture client-serveur. Bien que le système soit représentée avec un périphérique réel, elle peut aussi être un modèle (c'est-à-dire que la figure est valide pour les laboratoires virtuels et distants). Le contrôleur peut être placé soit sur le serveur, soit sur le côté client, comme les Figures 1.5(a) et 1.5(b) le représentent, respectivement. Traditionnellement, les contrôleurs ont été placés du même côté que le système dans les VRLs. Cependant, il peut être intéressant de séparer le contrôleur et de le placer du côté client de sorte que des retards incertains liés aux communications réseau apparaissent à la fois dans l'envoi de l'action de contrôle et dans la réception du retour du système. Enfin, comme le montre la figure 1.5(c), le contrôleur peut être placé non pas sur le serveur ou sur le côté client, mais ailleurs. Dans ce cas, le contrôleur est fourni en tant que service, ce qui le rend réutilisable et permet des scénarios intéressants tels que les expériences de contrôle distribuées.

Les boucles de contrôle qui sont fermées à travers les réseaux de communication deviennent de plus en plus fréquentes car le matériel requis devient moins cher et plus facile d'accès. Un système de commande qui communique avec des capteurs et des actionneurs à travers un ou plusieurs réseaux de communication est appelé en général, un système de commande distribué en temps réel. Il est bien connu que les réseaux de communication introduisent des retards dans la communication variables par nature en raison de la bande passante et de la surcharge des nœuds de communication. Pour cette raison, l'utilisation de techniques de contrôle, basées sur les événements, pour éviter les problèmes d'instabilité dans le processus découlant des retards dans le processus / capteur / contrôleur est un problème de recherche difficile qui peut être relié à la mise en œuvre de la nouvelle

génération de laboratoires de contrôle à distance.

En résumé, les laboratoires rigides existants ne fonctionnent que pour les cours d'introduction. Les étudiants en maîtrise avancée ont besoin d'approches plus souples, comme celles résumées dans cette section. Cette tendance à l'augmentation des options de contrôle pour les VRLs se poursuivra à l'avenir.

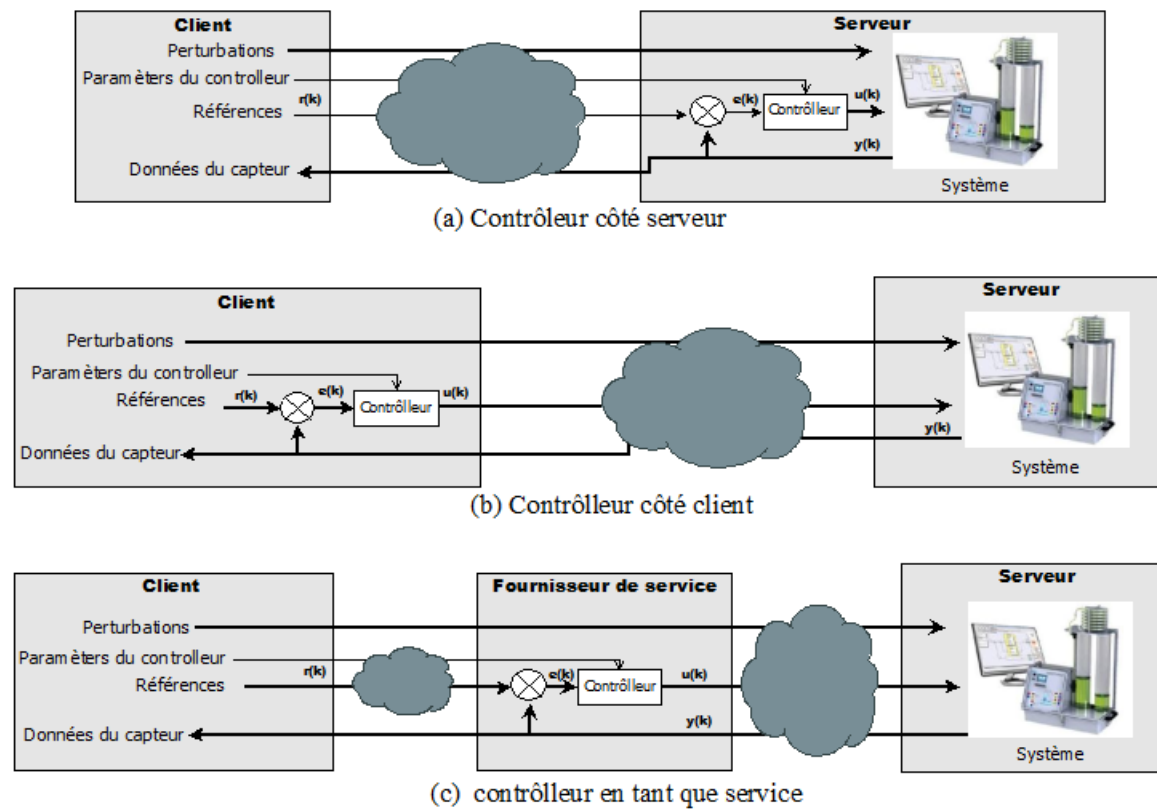


FIGURE 1.5 – Options de déploiement de contrôleur.

1.5 Conclusion

Les VRLs sont apparus il y a une vingtaine d'années, leur utilisation s'étendant progressivement à tous les niveaux d'enseignement de l'école primaire à l'enseignement supérieur, de l'apprentissage professionnel aux universités et de l'auto-éducation aux collèges techniques.

L'enseignement de la science de l'automatique est un défi puisqu'il s'agit d'une discipline à l'évolution très dynamique et dont la nature est transdisciplinaire. L'automatique est derrière le développement d'un large éventail d'applications, telles que le développement d'avions à haute performance, d'automobiles économes en carburant, d'usines de traitement industriel, de rovers planétaires, de réseaux de communication, etc. Les VRLs

offrent des opportunités sans précédent pour le partage à grande échelle des ressources d'expérimentation, et ils peuvent ainsi fournir aux étudiants l'accès à une grande variété de laboratoires.

Chapitre 2

Le langage HTML

2.1 Introduction

Au fur et à mesure de la croissance de la communauté du Web et de la diversification des capacités et aptitudes de ses membres, il devient crucial que les technologies employées soient appropriées à leurs besoins spécifiques. Le langage HTML a été conçu pour rendre les pages Web plus accessibles au grand public.

Dans ce chapitre, nous allons définir quelques notions de base du langage HTML.

2.2 Définitions

2.2.1 HTML

HTML est un acronyme qui signifie (HyperText Markup Language), c'est-à-dire Langage de Marquage HyperTexte qui utilise des instructions système composées d'éléments et de balises qui décrivent la structure et la présentation d'un document pour le web. Les instructions signifient qu'avec HTML vous déclarez qu'est ce qui est présenté au téléspectateur mais pas comment il est présenté. La présentation visuelle est définie par CSS (Cascading Style Sheets) et affichée par des navigateurs web [12]. Toutefois il existe des balises qui le permettent comme « font », mais elles sont entièrement obsolète, et ne devrait pas être utilisées par le développeur.

Le HTML est incontestablement le langage de base du web, c'est un langage de description permettant de structurer et d'afficher différents objets sur un écran (qu'on appellera abusivement : page). Ces objets peuvent être du texte, des tableaux, des images voire de la vidéo et des sons. Ils peuvent être aussi des éléments qui sont devenus classiques dans les environnements graphiques, à savoir les boutons, pop-listes, listes à ascenseurs, boîtes de dialogue [13].

2.2.2 Les version de HTML

- HTML 1 : C'est la toute première version créée par Tim Berners -Lee en 1991.
- HTML 2 : La deuxième version du HTML apparaît en 1994 et prend fin en 1996 avec l'apparition du HTML 3.0. C'est cette version qui posera en fait les bases des versions suivantes du HTML. Les règles et le fonctionnement de cette version sont donnés par le W3C.
- HTML 3 : Cette nouvelle version du HTML apparue en 1996, rajoute de nombreuses possibilités au langage comme les tableaux, les applets, les scripts, le positionnement du texte autour des images, etc.
- HTML 4 : Il s'agit de la version la plus répandue du HTML (plus précisément, il s'agit de HTML 4.01). Elle apparaît pour la première fois en 1998 et propose l'utilisation de frames (qui découpent une page web en plusieurs parties), des tableaux plus complexes, des améliorations sur les formulaires, etc. Mais surtout, cette version permet pour la première fois d'exploiter des feuilles de style, le fameux CSS !
- HTML 5 : C'est la dernière version. Encore assez peu répandue, elle fait beaucoup parler d'elle car elle apporte de nombreuses améliorations comme la possibilité d'inclure facilement des vidéos, un meilleur agencement du contenu, de nouvelles fonctionnalités pour les formulaires, etc [14].

2.2.3 Les balises

Les balises (tag) définissent un élément et ses propriétés. En HTML, ces balises sont les signes inférieur et supérieur par exemple `<element_nom>`.

Une balise définit un élément : un objet, texte ou autre, encadré par une balise d'ouverture et une balise de fermeture, qui est identique à la balise d'ouverture, à ceci près qu'une barre oblique (slash) est placée entre le caractère inférieur et le nom de l'élément. Entre les deux balises est placé le contenu. Si on veut utiliser les deux caractères `<` et `>` dans un texte et que le navigateur ne tente pas de l'interpréter comme balises, il faut les remplacer par leurs entités HTML équivalentes :

- `>` sera remplacé par `>` ;
- `<` sera remplacé par `<` ;

Exemple : `< element_nom >... Contenu ...</ element_nom >`.

On ne peut pas saisir des caractères accentués directement au clavier, il faut les remplacer par les codes HTML équivalents, il en va de même pour certains symboles. Les délimiteurs pour ces codes sont le caractère `&` (et commercial) au début du code et le

caractère (point-virgule) à la fin.

Exemple : pour écrire é (e avec accent aigu) en HTML il faut écrire :

```
&eacute;
```

Il y a quelques éléments HTML qui ont aucun contenu et ils sont dépourvu de balise de fermeture, et ceci lorsque l'absence de contenu est liée à la définition de ces éléments appelés éléments vide. Ils incluent : `<img>`, `<meta>`, `<link>` et `<input>`. Les noms des éléments peuvent être imaginés comme des mots clés qui nous renseignent sur le contenu qu'ils renferment, de tel que : « video, audio, table, footer ». La page HTML peut se composer de centaines d'éléments qui sont lus par le navigateur web, interprétés et traduits en contenu lisible et audible pour l'homme sur l'écran. La balise `<p>` représente un paragraphe ordinaire comme suit :

```
<p> Ceci est un paragraphe ordinaire. </p>
```

2.3 Création d'une simple page HTML

Les fichiers HTML peuvent être créés par un quelconque éditeur de texte mais doivent être sauvegardés sous l'extension « .html » ou « .htm ». Les fichiers HTML commencent par la balise `<html>` et se terminent par la balise `</html>` [12]. L'exemple suivant nous permet de créer une simple page web « bonjour tout le monde ». Une fois créé ce fichier peut être ouvert par quelconque navigateur web.

Exemple :

```
<!DOCTYPE html>
<html lang="fr">
  <head>
    <meta charset="UTF-8">
    <title> Bonjour tout le monde </title>
  </head>
  <body>
    <h1> Bonjour tout le monde ! </h1>
    <p> Ceci est un paragraphe </p>
  </body>
</html>
```



FIGURE 2.1 – Exemple d’une simple page HTML.

Décomposition et explication de l’exemple précédent :

- `<!DOCTYPE html>` : Définit la version du HTML utilisée dans le document. Dans ce cas, c’est le HTML5.
- `<html>` : Le premier langage utilisant le langage des codes ISO.
- `<head>` : L’information concernant le document HTML appelé (metadata), il peut aussi contenir des informations venant de l’extérieure comme le style de feuille et le type de texte.
- `<meta charset="UTF-8">` : Cette balise indique le type de codage utilisé. Le « charset » attribut et déclare le caractère d’encodage. Dans les documents HTML moderne on trouve « "UTF-8" » qui permet de taper tels quels les accents, le ç et le symbole €, sans avoir à les coder.
- `<title>` : C’est le titre de la page, et sera afficher dans l’onglet de la page du navigateur web ou dans la barre de titre du navigateur. Le texte est écrit entre cette balise d’ouverture et la balise de fermeture qui obligatoirement `</title>`.
- `<body>` : Ouvre la partie du document qui est affichée pour l’utilisateur, que ce soit du contenu visible ou audible. Après la balise de fermeture `</body>` on ne doit pas ajouter de contenu.
- `<h1>` : C’est le premier niveau de l’entête de la page.

2.3.1 DOCTYPE

C’est l’abréviation de (Document Type) qui signifie type du document. Il aide le navigateur à comprendre la version HTML du document qui est écrit pour être mieux interpréter. Les déclarations du doctype ne se fait pas par des balises HTML et elles doivent se trouvées au sommet du document avant la balise `<html>` [12].

2.3.2 HEADING

HTML ne fournit pas juste des balises des paragraphes mais aussi six balises d'en-tête (header) qui indiquent les différentes tailles et épaisseurs. En partant du premier en-tête vers le sixième les tailles diminuent et les épaisseurs deviennent plus fines. Les en-têtes peuvent être utilisés pour définir le sujet quels précédent, et ils sont définis par des balises de `<h1>` à `<h6>` [12].

`<h1>` : Définit l'en-tête le plus important.

`<h6>` : Définit l'en-tête le moins important.

Exemple de définition d'un en-tête : `<h1> en-tête 1 </h1>`.

Les moteurs de recherches et les utilisateurs intermédiaire indexent le contenu de la page en se basent sur les éléments d'en-tête. Par exemple pour créer un tableau, il est important d'utiliser la structure correcte pour l'en-tête. En général, un article doit avoir un élément `h1` pour le titre principale suivit d'un élément `h2` pour le sous-titre ainsi de suite si c'est nécessaire. Si il y a des éléments `h1` dans le niveau le plus haut, alors ils ne doivent pas être utilisés pour décrire le contenu des niveaux inférieures [12].

Exemple d'un document qui illustre la hiérarchie :

```
<h1>Titre principale</h1>
<p>Introduction</p>
<h2>Premier sous-titre</h2>
<h3>Deuxième sous-titre</h3>
<p>Paragraphe</p>
<h3>Troisième sous-titre</h3>
<p>Paragraphe</p>
<h2>Conclusion</h2>
<p>Paragraphe</p>
```

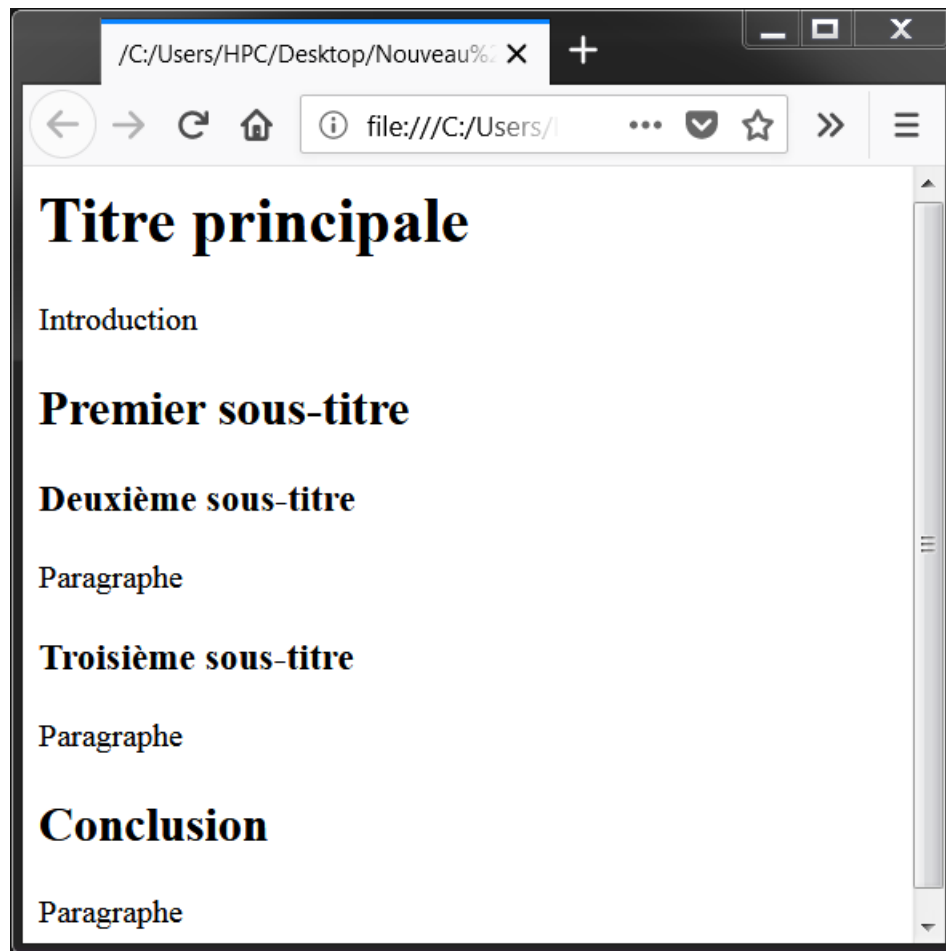


FIGURE 2.2 – La hiérarchie des en-têtes en HTML.

2.3.3 Principaux types d'encodage

- utf-8 (charset=utf-8) : codage universel Unicode (sur un ou deux octets).
- iso-8859-1 ou Latin-1 (charset=iso-8859-1) : codage occidental classique, souvent utilisé.
- iso-8859-15 ou Latin-9 (charset=iso-8859-15) : codage occidental iso-8859-1 plus quelques caractères dont €, Œ, œ, et Ÿ.
- Windows-1252 ou ANSI (charset=windows-1252) : codage provenant du iso-8859-1, comprenant également €, Œ, œ, et Ÿ.

2.3.4 Les paragraphes

Les paragraphes sont les éléments les plus basiques dans le HTML. On ne peut pas être sûr de la manière dont le HTML va être affiché, car la largeur, la taille des écrans et la résolution des fenêtres influent sur le résultat de l'affichage. Dans le code du HTML un large espace ou un saut de ligne ne sont pas pris en compte, le navigateur les supprime lors de l'affichage de la page [12].

Exemple :

```
<p> Voici un          exemple qui illustre          un          large espace
      Et un saut de ligne qui vont être supprimés par le navigateur web</p>
```

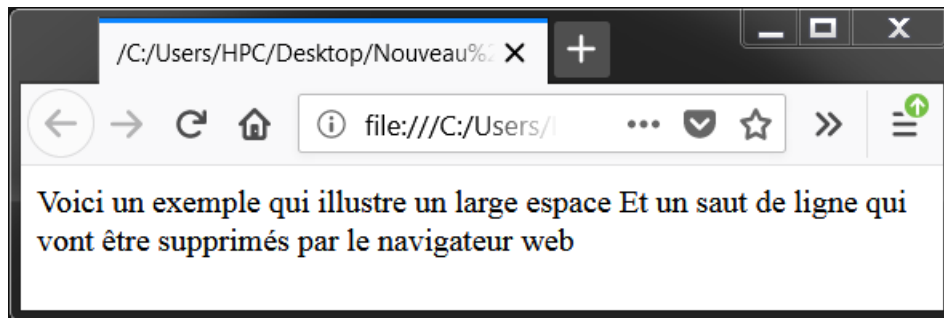


FIGURE 2.3 – Large espace et saut de ligne en HTML.

2.3.5 Saut de ligne et espace

Pour créer un saut de ligne qui sera effectivement affiché, il faut utiliser la balise `<br />` comme (break). Pour forcer l’affichage de plusieurs espaces successifs sur une ligne, il faut utiliser le caractère spécial ` ` ; (non breakable space) ou espace insécable en français.

2.3.6 Listes à puces ou numérotées

Une énumération gagne en clarté lorsque chacun des points est repéré par une puce ou un numéro. De telles listes sont délimitées par les balises :

- `<ul>...</ul>` comme unordered list pour une liste à puces.
- `<ol>...</ol>` comme ordered list pour une liste numérotée.

Dans les deux cas, chaque ligne est repérée par `<li>...</li>` à l’intérieur de ces balises.

Exemple :

```
<p>Liste à puces :</p>
<ul>
  <li>Premièrement</li>
  <li>Deuxièmement</li>
  <li>Troisièmement</li>
</ul>
```

```
<p>Liste numérotée :</p>
<ol>
  <li>Premièrement</li>
  <li>Deuxièmement</li>
  <li>Troisièmement</li>
</ol>
```

Résultat :

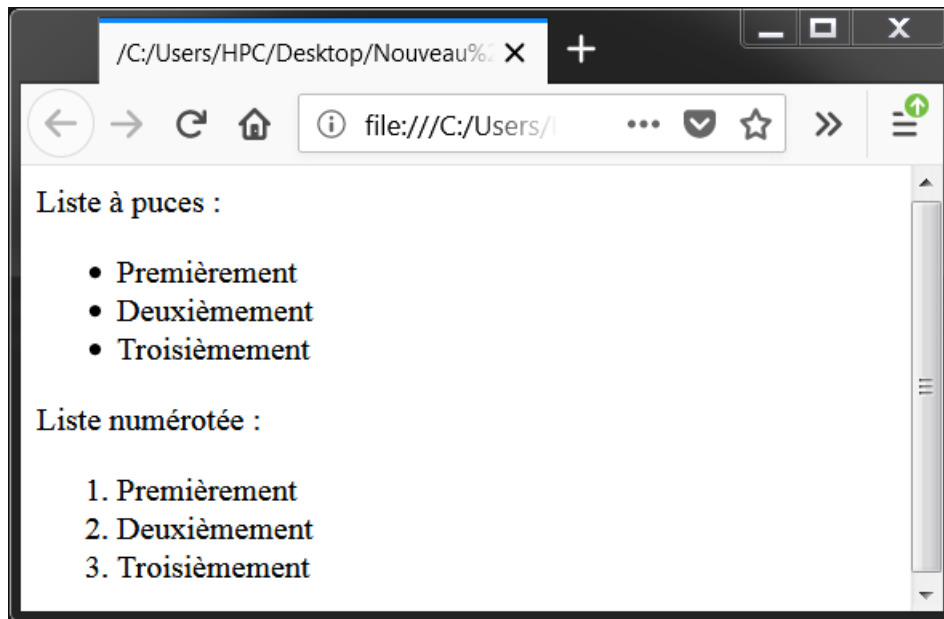


FIGURE 2.4 – Listes à puces et listes numérotées.

2.3.7 Formatage de texte

Alors que beaucoup de balises HTML sont utilisées pour créer des éléments, HTML fournit aussi pour le texte des balises de formatage qui permettent d'appliquer des styles spécifiques à des portions de textes.

2.3.7.1 Surbrillance du texte (Highlighting)

La balise `<mark>` est utilisée pour rendre le texte éclatant en le surlignant d'une couleur vive dans le document. La plupart des exemples usuels peuvent être le résultat d'une recherche ou l'utilisateur demande une requête, ce résultat est représenté par une surbrillance. Les formatages standards usuels sont du texte noir sur un arrière-plan jaune, mais cela peut être changé avec CSS [12].

Exemple :

```
<p>Voici une partie d'un article qui contient <mark>une requête</mark>
```

que nous cherchons.</p>

Résultat :



FIGURE 2.5 – Surbrillance du texte.

2.3.7.2 Gras, Italique et Souligné

- Gras : Pour mettre le texte en gras en utilise les balises `<strong>` ou `<b>` : `<strong>` Texte en gras `</strong>` ou `<b>` Texte en gras `</b>`. `<strong>` Correspondent par défaut au gras, mais avec CSS en peut remplacer le gras par d'autre forme ou couleurs. Tandis que `<b>` représente simplement ce qui devrait être en gras.
- Italique : Pour mettre le texte en italique, on utilise les balises `<em>` ou `<i>` : `<em>` Le texte est en italique `</em>` ou `<i>` texte est en italique `</i>`. `<em>` (emphase) : Exagération dans l'expression, le ton, la voix, le geste. La balise produit par défaut de l'italique, elle est utilisée pour indiquer si le texte ou une action à l'intérieure d'une phrase doit avoir une haute intonation et accentué, tandis que `<i>` représente simplement le texte qui doit être distingué du texte normale qui l'entoure comme le titre d'un journal ou d'un livre.
- Souligné : L'élément `<u>` représente ce qui n'est pas articulé, note explicative, non textuel on peut aussi l'utiliser pour une traduction pour indiquer une faute d'orthographe dans le texte de la page [12].

Exemple :

```
<p> Cette phrase contient quelque <u>fautes d'orthographe </u></p>
```

Résultat :

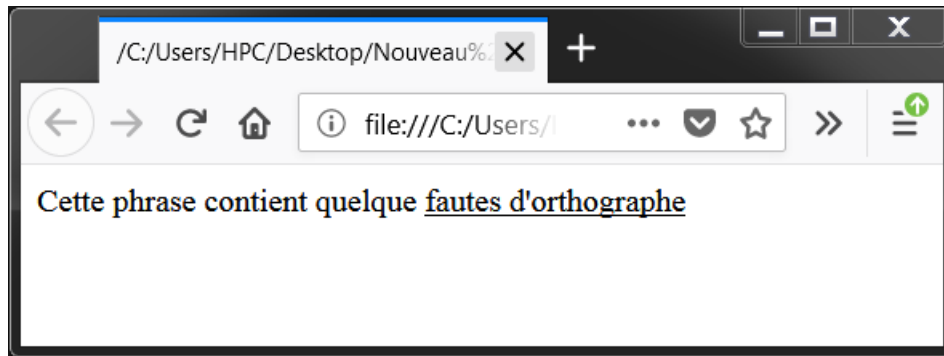


FIGURE 2.6 – Souligné.

2.3.7.3 L'abréviation

Pour marquer une expression comme abréviation, on utilise la balise `<abbr>`.

Exemple :

```
<p>C'est un langage <abbr titre="HyperText Markup Language">HTML</abbr>!  
</p>
```

Résultat :

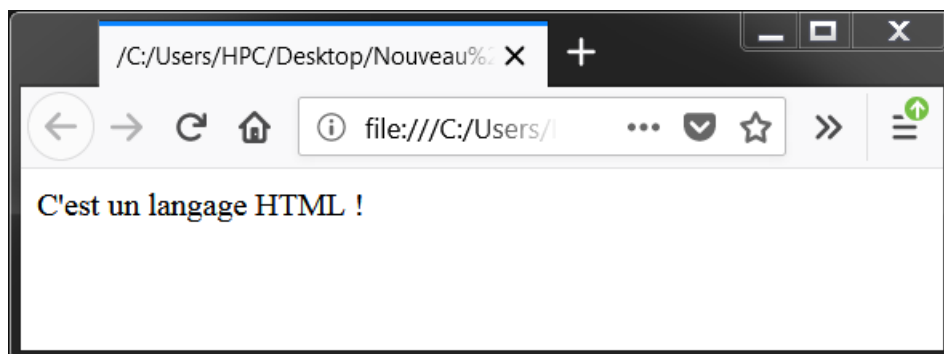


FIGURE 2.7 – Abréviation.

2.3.7.4 Inséré, Supprimé et Barré

- Inséré : Pour marquer un texte comme inséré, on utilise la balise `<ins>` : `<ins> Nouveau texte </ins>`.
- Supprimé : Pour marquer un texte comme supprimé, on utilise la balise `<del>` : `<del> Texte supprimé </del>`.
- Barré : Pour barrer le texte, on utilise la balise `<s>` : `<s> Le texte est barré </s>`.

2.3.7.5 Exposant et Indice

Pour mettre un texte ou bien exposant ou indice, on utilise la balise `<sup>` ou `<sub>`.

- Exposant : `<p>1 <sup> er </sup></p>`.
- Indice : `<p>x <sub> i,j </sub></p>`.

2.4 Les Liens

Un lien est une relation entre deux éléments d'information sensible au clic de la souris. Il est possible de placer des liens sur les objets structurer et afficher par HTML. Ce dernier nous permet de créer différents lien qui mènent vers différents endroits comme :

2.4.1 Un lien vers un autre site

Voici un simple exemple d'utilisation de la balise `<a>` qui permet de créer un lien :

```
<a href="http://facebook.com/">facebook.com</a>
```

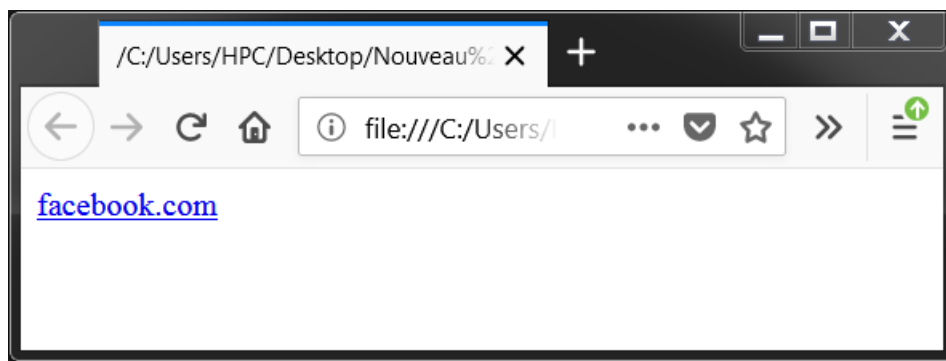


FIGURE 2.8 – Liens vers un autre site.

Le résultat obtenu est le lien créé pour l'URL `http ://facebook.com/` qui est spécifié dans le programme HTML par « href » (hypertext reference). L'attribut « href » est obligatoire, puisqu'il indique l'adresse de la page à afficher lors d'un clic sur ce lien. Si la page à atteindre se trouve dans le même dossier que la page en cours d'affichage, en utilise l'adresse internet ou un nom de fichier. Si le fichier est dans un sous-dossier, il faut taper le nom de ce dossier suivi d'un séparateur : « / » Pour indiquer que le lien amène vers un site web externe, on peut utiliser « external » [12].

Exemple :

```
<a href="http://facebook.com/" rel="external">site web externe</a>
```

Exemple :

```
<a href="ftp://exemple.com/">C'est un lien vers un site qui utilise  
le protocole ftp</a>
```

2.4.2 Lien vers une ancre dans la même page

Les ancres peuvent être utilisées pour des sauts dans les pages HTML avec des balises spéciales. La balise `<a>` peut pointer vers un quelconque élément si il est attribuer par un id.

Exemple :

```
<a href="#poissons" title="Les habitants de la mer">Accès au paragraphe
    Les habitants de la mer</a>
<h2 id="poissons">Les habitants de la mer</h2>
```

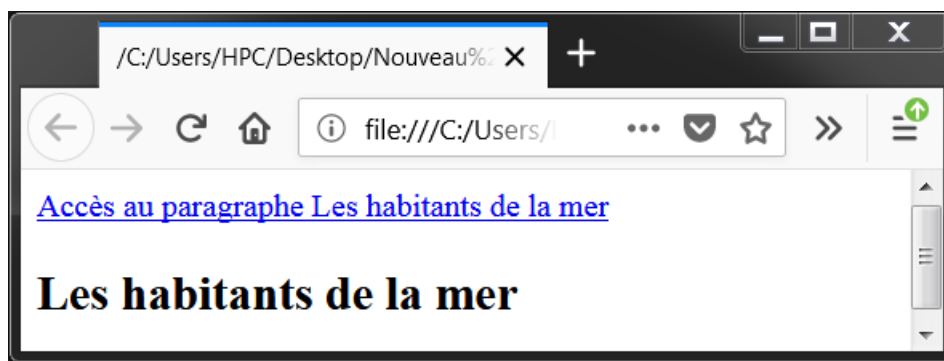


FIGURE 2.9 – Lien vers une ancre.

L'attribut « title » permet d'afficher un texte automatiquement dans une bulle au passage de la souris sur le lien dans l'exemple c'est Les habitants de la mer. Cette méthode permet également de créer un lien vers un endroit précis d'une autre page que celle affichée. Par exemple, le lien `<a href="oiseaux.html#rossignol">` affichera la page oiseaux.html et placera le curseur de la fenêtre sur la balise d'identifiant `id="rossignol"`. Les ancres sont surtout utilisées pour des sauts de sous-sections dans la page et pour l'union des balises d'en-têtes. Une fois qu'on a plusieurs sections, on peut créer un tableau de contenus avec des liens qui spécifient les sections au sommet de la page.

Exemple :

```
<h1>Tableau de Contenus</h1>
<a href='#Ancre1'>Cliquer pour sauter vers la première ancre</a><br />
<a href='#Ancre2'>Cliquer pour sauter vers la deuxième ancre</a>
<h2 id="Ancre1">Première ancre</h2>
<p>Contenu du la première ancre</p>
<h2 id="Ancre2">Deuxième ancre</h2>
<p>Contenu du la deuxième ancre</p>
```

Résultat :

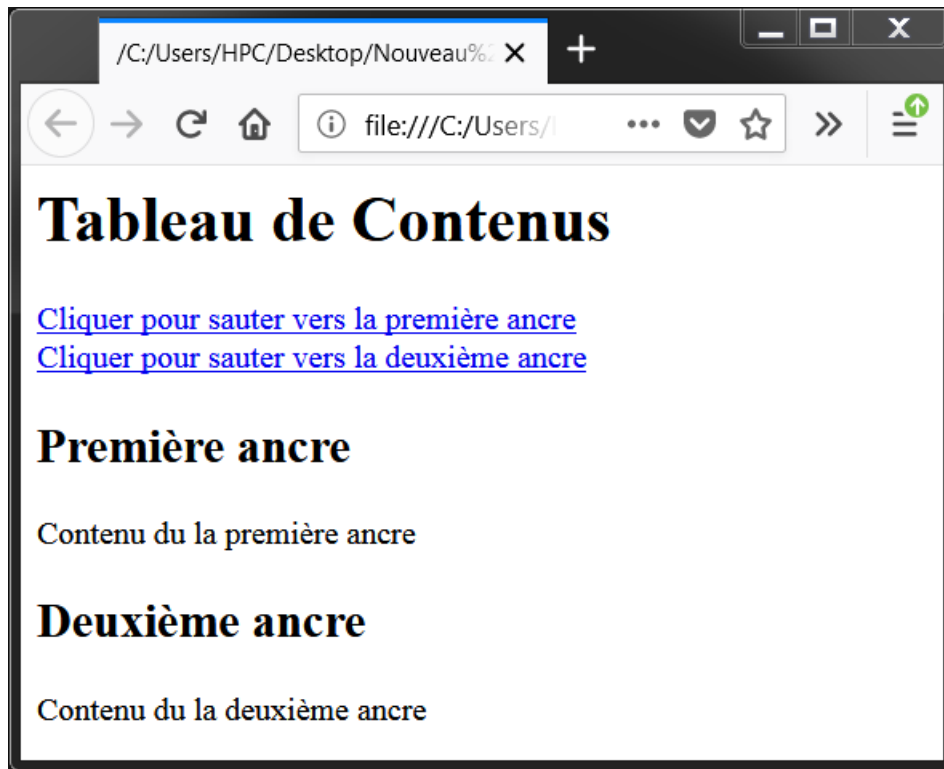


FIGURE 2.10 – Tableau de Liens vers des ancrs.

2.4.3 Lien vers une ancre dans une autre page

Une ancre est définie par la balise `<a>` permet d'écrire un lien Hypertext qui sert à relier les pages entre elles. Les ancrs peuvent être attachées à une autre page HTML, c'est-à-dire on peut créer des liens d'une page à une autre en spécifiant le nom de la page et le nom de l'ancre.

Exemple :

— On suppose qu'on a créé une page HTML (page1.html) :

```
<h2>Premier thème</h2>
<p>Contenu du premier thème</p>
<h2>Deuxième thème</h2>
<p>Contenu du deuxième thème</p>
```

— Maintenant, on va créer une autre page HTML (page2.html) :

```
<a href="page1.html">Aller vers le premier thème</a>
```

2.4.4 Lien vers une page dans le même site

On peut utiliser un chemin relatif pour un lien vers d'autres pages dans le même site web.

Exemple :

```
<a href="/exemple">C'est le lien</a>
```

L'exemple au-dessus le lien part vers le fichier exemple qui dans le répertoire racine (/) du serveur.

2.5 Les Tableaux

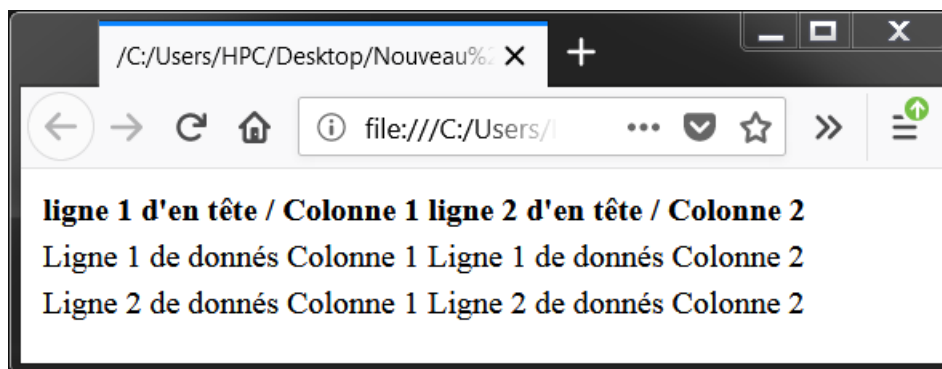
L'élément du HTML `<table>` autorise les auteurs web d'afficher des données tabulaires telles que texte, images, liens, autre tableau dans un tableau a deux dimensions qui est formé de lignes et de colonnes.

Exemple :

```
<table>
<tr> // ligne de tableau
<th> ligne 1 d'en tête / Colonne 1</th>
  //<th> pour ajouter une colonne en gras
<th> ligne 2 d'en tête / Colonne 2</th>
</tr>
<tr>
<td> Ligne 1 de donnés Colonne 1</td><td> pour ajouter une colonne
<td> Ligne 1 de donnés Colonne 2</td>
</tr>
<tr>
<td> Ligne 2 de donnés Colonne 1</td>
<td> Ligne 2 de donnés Colonne 2</td>
</tr>
</table>
```

Ou th signifie table header et td table data.

Résultat :



ligne 1 d'en tête	Colonne 1	ligne 2 d'en tête	Colonne 2
Ligne 1 de donnés	Colonne 1	Ligne 1 de donnés	Colonne 2
Ligne 2 de donnés	Colonne 1	Ligne 2 de donnés	Colonne 2

FIGURE 2.11 – Tableau.

Initialement, les bordures du tableau sont invisibles, donc il faut ajouter dans la balise `<table>` l'attribut " border " pour indiquer l'épaisseur de la bordure en pixels. Également inclus dans la balise `<table>`, les attributs " cellpadding " et " cellspacing " permettent de préciser en pixels, respectivement l'espacement entre cellules (traits d'encadrement simples si 0, doubles sinon) et les marges intérieures des cellules. La balise `<table>` permet de créer un tableau, dans l'exemple précédent, le tableau est constitué de trois lignes grâce à la balise `<tr>`. La première ligne est une ligne d'en-tête qui est constitué de deux colonnes qui sont en gras, les colonnes sont créées par la balise `<th>`. Les deux autres lignes sont des lignes de donnés constituées aussi de deux colonnes mais elles ne sont pas en gras elles sont créées par la balise `<td>`.

2.5.1 Portée des lignes et des colonnes

Les cellules du tableau peuvent prendre plusieurs lignes ou colonnes et cela se fait par « colspan » et « rowspan ». Ces attributs peuvent être appliqués aux éléments `<th>` et `<td>` [12].

Exemple :

```
<table border="1">
<tr>
<td>Ligne 1 colonne 1</td>
<td>Ligne 1 colonne 2</td>
<td> Ligne 1 colonne 3</td>
</tr>
<tr>
<td colspan="3">Cette deuxième Ligne Occupe toute
    les trois colonnes</td>
</tr>
<tr>
<td rowspan="2">Cette cellule occupe deux Lignes </td>
```

```

<td> Ligne 3 colonne 2</td>
<td> Ligne 3 colonne 3</td>
</tr>
<tr>
<td> Ligne 4 colonne 2</td>
<td> Ligne 4 colonne 3</td>
</tr>
</table>

```

Résultat :

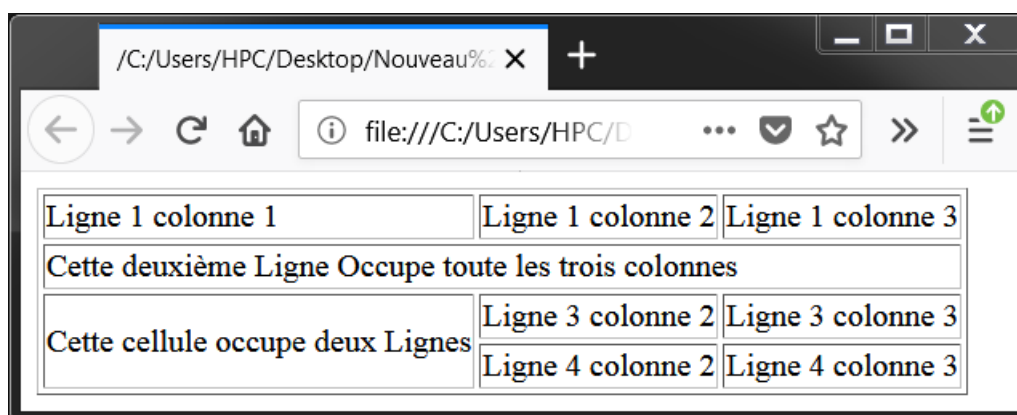


FIGURE 2.12 – Porté des lignes et des colonnes dans un tableau.

- « rowspan » : Est un entier non négatif qui spécifie le nombre de ligne occuper par la cellule, et la valeur attribuer par défaut est un (1). La valeur zéro (0) veut dire que la cellule va s'étendre de la ligne en cours jusqu'à la dernière ligne du tableau (<thead>, <tbody> et <tfoot>).
- « colspan » : Est un entier non négatif qui spécifie le nombre de colonnes occupées par la cellule, et la valeur attribuer par défaut est un (1). La valeur zéro (0) veut dire que la cellule va s'étendre de la colonne en cours jusqu'à la dernière colonne du groupe de colonne <colgroup> auquel la cellule est définit.

2.5.2 Tableau avec thead, tbody, tfoot et caption

HTML fournit aussi des tableaux avec les éléments <thead>, <tbody>, <tfoot> et <caption>, ces éléments sont utiles pour ajouter des valeurs et prévoir de la place pour le code CSS. Quand le tableau ne convient pas sur la page du papier lors de l'impression, la plupart des navigateurs modifie les contenus de <thead> dans chaque page [12].

La balise <thead> regroupe les informations d'en-tête (en général les titres des colonnes), la balise <tbody> contient les données à proprement parlé et la balise <tfoot>

réuni les informations relatives au pied de page (des références, remarques ou légendes par exemple).

Il y a un ordre spécifique auquel on doit adhérer, l'exemple suivant démontre comment quatre éléments doivent être placés :

Exemple :

```
<table>
<caption>Titre du tableau</caption>
<thead> <!-- thead est après caption |-->
<tr>
  <th>contenu d'en-tête1</th>
  <th>content d'en-tête2</th>
</tr>
</thead>
<tbody> <!-- tbody c'est après thead -->
<tr>
  <td>contenu du corps1</td>
  <td>contenu du corps2</td>
</tr>
</tbody>
<tfoot>
<tr>
  <td>contenu du pied1</td>
  <td>contenu du pied2</td>
</tr>
</tfoot>
</table>
```

Remarque : tfoot peut être placé avant ou après tbody, mais pas dans un groupe de tbody.

Résultat :

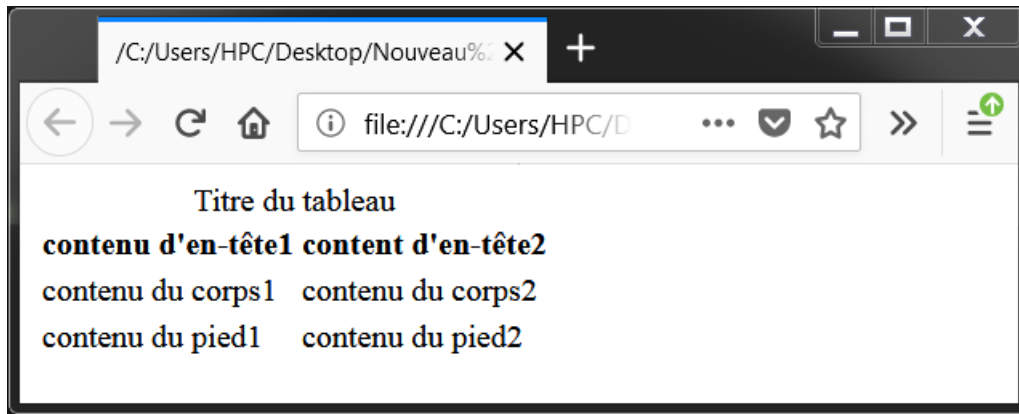


FIGURE 2.13 – Tableau avec thead,tbody,tfoot et caption.

2.6 Création des images

Pour ajouter une image à une page web, on utilise la balise `<img>`. Cette balise n'a pas besoin d'une balise de fermeture. Les attributs « `src` » et « `alt` » sont obligatoirement ajoutés à la balise `<img>`. « `src` » : Il spécifie l'URL de l'image. « `alt` » : Il spécifie un texte alternatif qui doit être affiché si l'image ne s'affiche pas pour une certaine raison.

Exemple :

```

```

On peut aussi prendre l'image depuis l'URL du web.

Exemple :

```

```

Techniquement les images ne sont pas insérées dans la page HTML, les images sont liées à la page HTML. La balise `<img>` crée un espace pour l'image référencée. Si l'attribut « `src` » ne contient qu'un nom de fichier, cela signifie que ce fichier image se trouve dans le même dossier que la page web qui l'utilise et si cette image se trouvait dans le sous-dossier `images`, la balise `<img>` s'écrirait : ``. Il est possible d'enfoncer les images directement à l'intérieur de la page en utilisant « `base64` » [12].

Exemple :

```

```

Pour lier une image vers un autre document, on ajoute simplement la balise `<a>` à l'intérieur de la balise `<img>`.

2.6.1 Utilisation de l'attribut de `srcset`

```

```

Les tailles sont comme `media`, elles décrivent combien d'espace va prendre l'image de viewport. Il est possible de préciser les dimensions que doit prendre l'image en utilisant les attributs « `width` » et « `height` » qui donnent respectivement sa largeur et sa hauteur. La manière la plus simple consiste à ne définir qu'un seul de ces attributs, car l'autre sera calculé automatiquement pour que les proportions soient respectées. Cependant, mieux vaut ne pas utiliser ces attributs et retailler préalablement l'image aux dimensions souhaitées l'image affichée sera de meilleure qualité et cela n'obligera pas les internautes à télécharger un gros fichier pour afficher seulement une petite image.

2.7 CSS

CSS est un acronyme qui signifie (Cascading Style Sheets), c'est-à-dire feuilles de style en cascade. Le terme « en cascade » indique que la mise en forme d'une page peut faire appel à plusieurs feuilles de style. Les différentes propriétés affectées à un même élément s'ajoutent pour lui donner sa mise en forme finale. Lorsque deux propriétés se contredisent, des règles de priorité s'appliquent et c'est généralement le dernier style défini qui est pris en compte.

2.7.1 Emplacement des styles

Les règles de style peuvent se trouver :

- Dans le code HTML, comme attributs de balises : ce sont des styles en ligne (utilisation déconseillée).
- Dans l'en-tête de la page web : feuille de style interne.

- Dans un fichier distinct : feuille de style externe à appeler dans l'en-tête de la page Web.

2.7.2 Principes d'une bonne écriture HTML/CSS

La meilleure méthode pour l'écriture du code HTML/CSS est celle qui sépare le contenu qui est codé en HTML et la mise en forme qui est codé par CSS. Cette méthode présente plusieurs avantages comme la clarté du code et la possibilité de définir des styles communs à plusieurs pages. La dissociation du contenu (HTML) et de la mise en forme (feuille de style) permet :

- De retrouver et corriger plus facilement le texte des pages.
 - D'utiliser une feuille de style externe commune aux différentes pages d'un site. Il en résulte une meilleure unité graphique entre ces pages et aussi des mises à jour plus simples par la suite. Une modification dans la feuille de style externe se répercute d'un seul coup sur toutes les pages du site. La mise en page est beaucoup plus légère car elle ne nécessite plus l'utilisation de tableaux. Les CSS permettent en effet de positionner les différentes parties d'une page web :
1. Soit de façon rigoureuse : blocs fixes dont les coordonnées sont choisies.
 2. Soit d'une manière souple : blocs flottants qui s'alignent les uns par rapport aux autres.

2.7.3 L'utilisation de HTML avec CSS

CSS fournit des styles pour les éléments du HTML dans une page et dans une ligne. La forme implique l'usage de style dans l'attribut des balises mais c'est déconseiller car c'est difficile.

- Feuilles de style internes : utilise la balise `<style>` qui est utilisée pour déclarer les règles pour diriger les portions de la page.

Lorsque les règles de styles sont regroupées dans l'en-tête de la page web, elles constituent une feuille de style interne. Les styles sont écrits entre les balises `<head>` et `</head>` à l'intérieur d'une balise `<style>`.

Exemple d'utilisation interne de Feuilles de style :

```
<head>
  <style type="text/css">
    body {background-color: gray;}
  </style>
```

```
</head>
```

Les styles ainsi déclarés sont de type texte d'où l'attribut `type="text/css"` dans la balise `<style>`.

— Feuilles de style externes : Lorsque des règles de styles sont applicables à plusieurs pages web, il est intéressant de les écrire dans un fichier à part. Cette feuille de style externe est appelée par chacune des pages concernées. Elle garantit l'unité graphique du site et facilite les modifications. Les feuilles de style peuvent être utilisées par la balise `<link>` qui prend un fichier externe de CSS et applique les règles au document. Une feuille de style externe est un fichier d'extension « .css » :

1. C'est un fichier texte qui contient l'ensemble des règles définies.
2. Il ne contient pas les balises `<style> ... </style>`.
3. Il n'y a pas non plus les symboles de commentaires `<!-- ... -->`.

Exemple d'utilisation externe de Feuilles de style :

L'utilisation de l'attribut de « link » dans la tête du document :

```
<head>
<link rel="stylesheet" type="text/css" href="mon_style.css">
</head>
```

Dans l'exemple précédent, la feuille de style s'appelle `mon_style.css` et se trouve dans le même dossier que la page web. Une multitude de « Stylesheet » interne peuvent être inclues dans un même programme comme le montre l'exemple suivant :

Exemple :

```
<head>
<style type="text/css">
body {background-color: gray;}
</style>
<style type="text/css">
p {background-color: blue;}
</style>
</head>
```

- Styles en ligne : Dans le corps de la page HTML (entre `<body>` et `</body>`), il est possible de préciser des styles qui prévaudront sur les feuilles de style interne et externe. Cette méthode est à éviter autant que possible, car elle revient à mélanger le contenu et la mise en forme. Cela enlève la clarté et l'homogénéité apportées par les feuilles de style. La maintenance des pages devient plus délicate. On peut créer un élément spécifique par l'utilisation de l'attribut de « style » :

```
<span style="color: red">Ce texte va apparaitre en rouge.</span>
```

2.7.4 Charger plusieurs feuilles de style

Il est possible charger plusieurs Feuilles de style :

Exemple :

```
<head>  
<link rel="stylesheet" type="text/css" href="general.css">  
<link rel="stylesheet" type="text/css" href="specific.css">  
</head>
```

2.8 Conception page HTML/CSS

Pour illustrer le fonctionnement du langage HTML nous avons développé deux page web avec le CSS inclus. L'interface principale qui présente l'accès a quatre autres pages sous forme d'un laboratoire virtuel et qui permettent d'effectuer un TP.

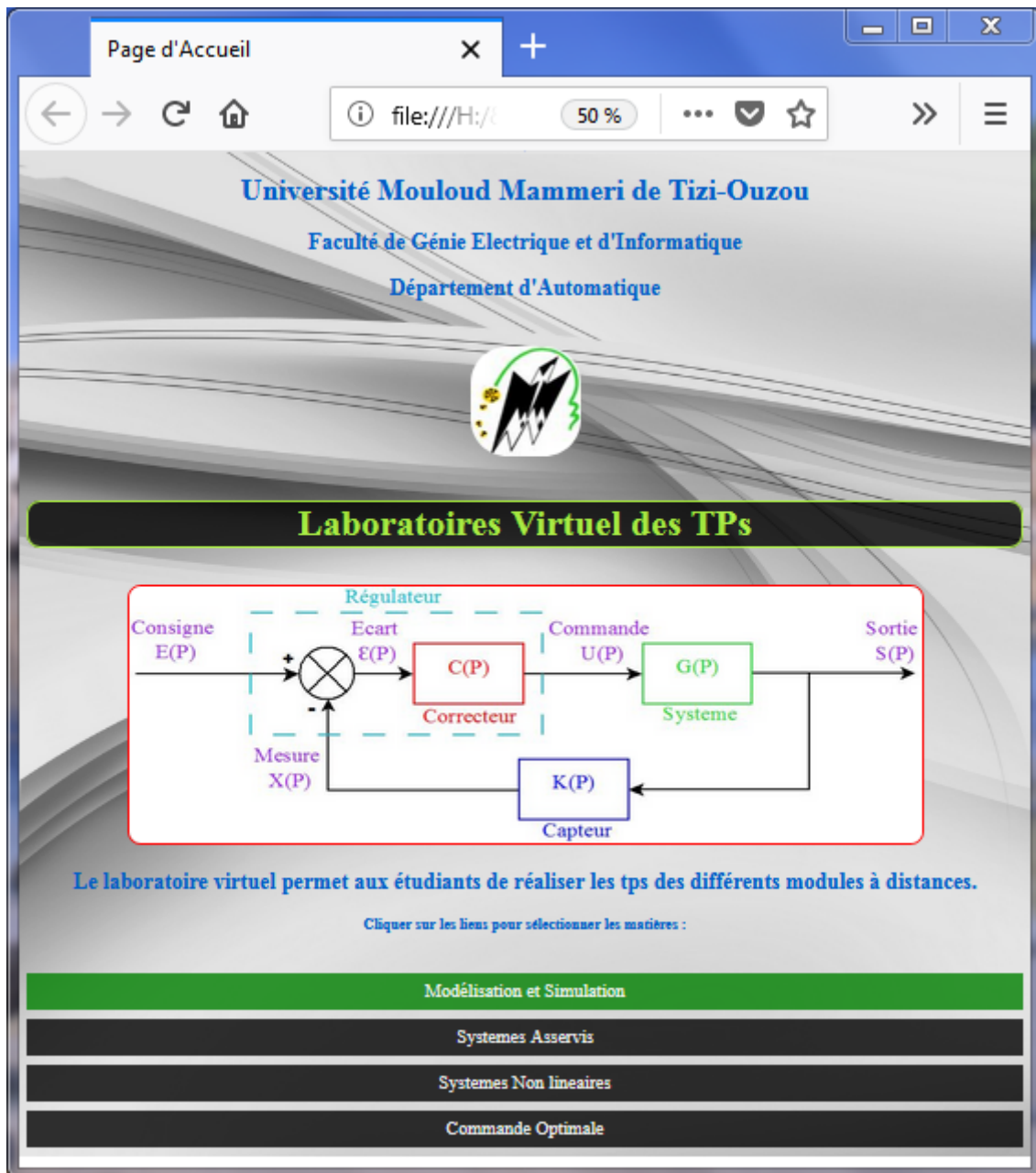


FIGURE 2.14 – Interface principale d'un laboratoire virtuel.

En cliquant sur le premier lien, l'étudiant est redirigé vers une autre page web (Modélisation et Simulation) et qui permet d'effectuer un TP sur la simulation. Le système figuré sur cette page est un système masse-ressort dont le schéma synoptique et les équations d'état sont afficher dans cette page. L'étudiant a la possibilité d'introduire sur les champs d'entré les valeurs des paramètres mais aussi il peut choisir l'état, l'entrée, la sortie et spécifié les conditions initiales. En cliquant sur le Button Cliquer ici les résultats de simulation seront affichés.

Modelisation Et Simulation

file:///H:/&PAGE D'ACEUILLE&/ 67 %

Université Mouloud Mammeri de Tizi-Ouzou
Faculté de Génie Electrique et d'Informatique
Departement d'Automatique

Laboratoires Virtuel des Tps
Modelisation et Simulation
Simulation d'un système Masse-Ressort

Equations :

$$\begin{aligned}\dot{\theta}_1 &= \theta_2 \\ \dot{\theta}_2 &= -\frac{kr^3}{m_1 l^2} (\sin(\theta_1(t)) - \sin(\theta_3(t)))^2 \cos(\theta_1(t)) - \frac{g}{l} \sin(\theta_1(t)) \\ \dot{\theta}_3 &= \theta_4 \\ \dot{\theta}_4 &= \frac{kr^3}{m_1 l^2} (\sin(\theta_1(t)) - \sin(\theta_3(t)))^2 \cos(\theta_3(t)) - \frac{g}{l} \sin(\theta_3(t))\end{aligned}$$

Shéma synoptique :

Les variables et leurs dimensions :

Variables	Dimensions

Parametres :

FIGURE 2.15 – Page HTML sous forme d'un laboratoire virtuel.

Modelisation Et Simulation

file:///H:/&PAGE D'ACEUILLE&/ 67 %

$$\begin{aligned} \dot{\theta}_1 &= \theta_2 \\ \dot{\theta}_2 &= -\frac{kr^3}{m_1 l^2} (\sin(\theta_1(t)) - \sin(\theta_3(t)))^2 \cos(\theta_1(t)) - \frac{g}{l} \sin(\theta_1(t)) \\ \dot{\theta}_3 &= \theta_4 \\ \dot{\theta}_4 &= \frac{kr^3}{m_1 l^2} (\sin(\theta_1(t)) - \sin(\theta_3(t)))^2 \cos(\theta_3(t)) - \frac{g}{l} \sin(\theta_3(t)) \end{aligned}$$

Les variables et leurs dimensions :

Variables	Dimensions
θ_1 (Position angulaire de m1)	rad
θ_2 (Vitesse angulaire de m1)	rad/s
θ_3 (Position angulaire de m2)	rad
θ_4 (Vitesse angulaire de m2)	rad/s
g (Accélération de la pesanteur)	m/s ²

Parametres :

Parametres	Dimensions
m1 (Poid de la masse 1)	kg
m2 (Poid de la masse 2)	kg
k (Raideur du ressort)	kg/s ²
l (Longueur de la tige)	m

m1 : m2 : k : l :

Choix de l'état :

☐ Etat 1 ☐ Etat 2

Choix de l'entrée :

Echelon unitaire ☐ Sinusoïdale ☐ Impulsion ☐ Rampe ☐

Choix de la sortie :

La sortie :

Simulation :

Conditions Initiales :

[Cliquez ici](#)

[Retour vers le menu principale](#)

FIGURE 2.16 – Page HTML sous forme d’un laboratoire virtuel.

2.9 Conclusion

Le langage de balisage HTML permet de structurer un document et de le mettre en forme. Cependant, ce langage utilisé intensément par les sites internet tend à devenir exclusivement un langage de structuration en se rapprochant du format XML, laissant la mise en forme aux feuilles de styles en cascade (CSS) et les animations au JavaScript.

Les feuilles de style simplifient le balisage HTML et soulagent grandement HTML des responsabilités de la présentation. Elles donnent aux auteurs comme aux utilisateurs le contrôle de la présentation des documents (informations sur les polices de caractères,

alignement, couleurs, etc.). Les informations de styles peuvent être spécifiées pour un élément ponctuel ou pour des groupes d'éléments.

Pour l'interactivité d'une page web, on utilise le JavaScript qui fera l'objet du chapitre suivant.

Chapitre 3

Le langage JavaScript

3.1 Introduction

Pour dynamiser une page web, les fonctionnalités offertes par le langage HTML sont insuffisante. Ainsi, des langages de programmation spécialisés sont apparus permettant d'animer la page web. Parmi ces langages, le JavaScript représente le langage le plus utilisés avec des bibliothèques très riches. Dans ce chapitre, on présente quelques notions de JavaScript.

3.2 Définitions

3.2.1 Langage de programmation

Un langage de programmation est un langage qui permet aux développeurs d'écrire du code source qui sera analysé par l'ordinateur. C'est un ensemble d'actions, appelées instructions, qui vont permettre de donner des ordres à l'ordinateur afin de faire fonctionner le programme. Le code source est celui qui régit le fonctionnement du programme [14].

3.2.2 JavaScript

Le JavaScript est un langage de scripts, cela signifie qu'il s'agit d'un langage interprété qui veut dire qu'il n'y a pas de compilation. Le code source reste tel quel, alors il est donc nécessaire de posséder un interpréteur qui se chargera de le lire et de réaliser les actions demandées pour faire fonctionner le code JavaScript.

Les interpréteurs sont inclus dans les navigateurs web et chaque navigateur web possède son propre interpréteur JavaScript comme SpiderMonkey de Mozilla Firefox et V8 de Google Chrome.

Le JavaScript s'inclut directement dans la page web ou dans un fichier externe. Il permet de dynamiser une page HTML en ajoutant des interactions avec l'utilisateur, des animations, de l'aide à la navigation comme par exemple [14] :

- Afficher ou masquer du texte.
- Faire défiler des images.
- Créer un diaporama avec un aperçu en grand des images.
- Créer des infobulles.

Il est important de préciser que le langage JavaScript est dit client-side (coté client), c'est-à-dire que les scripts sont exécutés par le navigateur chez l'internaute (client) et cela diffère des langages de scripts dit server-side qui sont exécutés par le serveur web. Ces derniers s'occupent de créer la page web qui sera ensuite envoyée au navigateur [14].

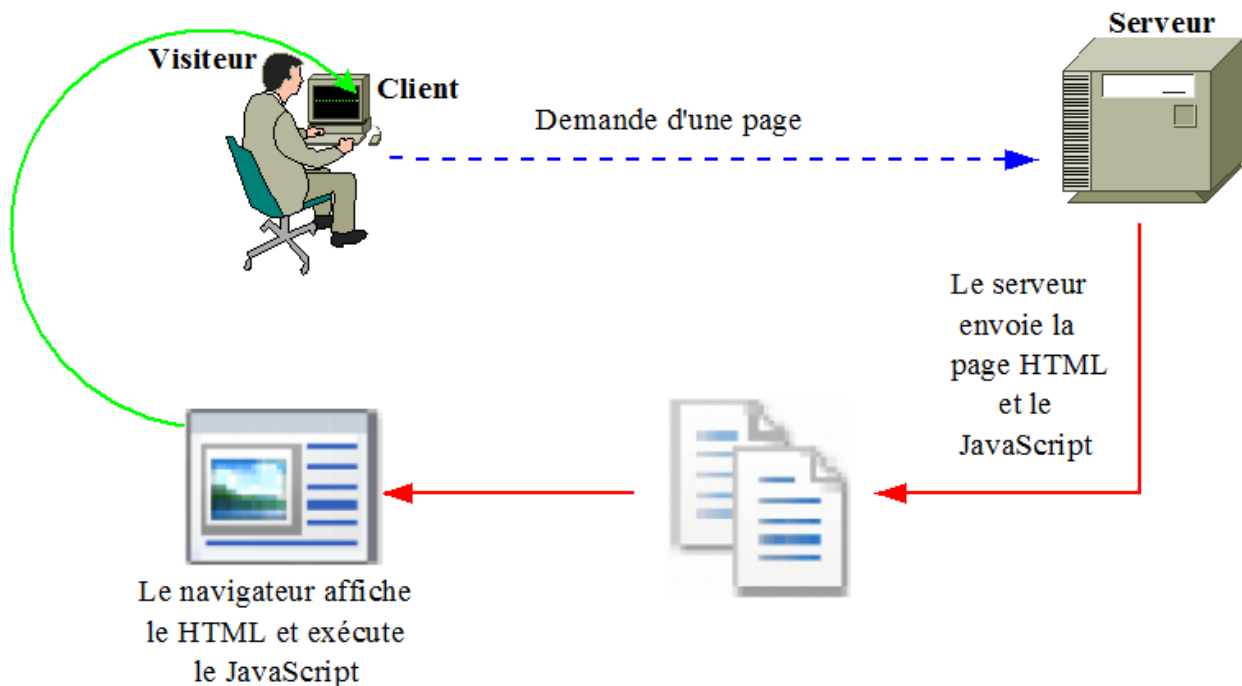


FIGURE 3.1 – Explication du fonctionnement client-side.

Le visiteur demande une page via le serveur, ce dernier répond à la demande et envoie la page HTML et le JavaScript sous forme de fichier à extension `.html` qui aboutira sur un poste client (micro-ordinateur) puis interprété grâce à un programme appelé navigateur.

3.2.3 Exemple d'introduction

Ouvrir les outils de développement du navigateur et aller à la console de JavaScript, ensuite taper « `console.log ("Bienvenu");` » et ensuite taper sur la touche du clavier

"entrer". Dans cet exemple, la fonction `console.log ( )` permet d'afficher "Bienvenu" de la console, et renvoi `undefined` car la fonction `console.log ( )` n'a pas de valeur explicite.

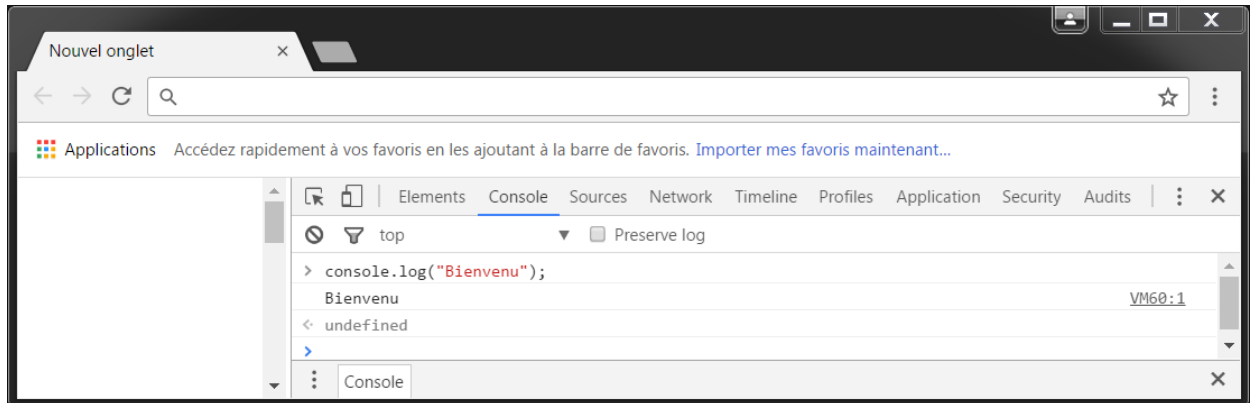


FIGURE 3.2 – La fonction `console.log` dans la console du navigateur.

La fonction `console.log ( )` peut être utilisée pour noter les variables de tout genre. Pour afficher une variable dans une console, il faut la déclarer avec le mot clé `var` [12].

Exemple :

```
var foo = "bar" ;  
console.log(foo) ;
```

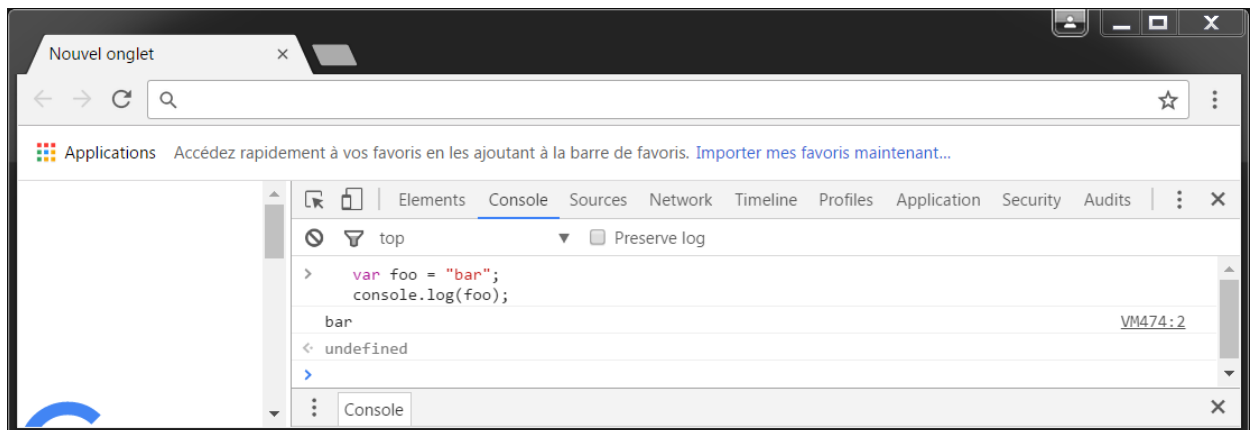


FIGURE 3.3 – Déclaration et affichage d'une variable.

Si on veut introduire plusieurs valeurs, on les sépare par une simple virgule. L'espace va être automatiquement ajouté entre chaque argument pendant leurs enchainement [12].

```
var thisVar = 'first value' ;  
var thatVar = 'second value' ;  
console.log("thisVar:", thisVar, "and thatVar:", thatVar) ;
```

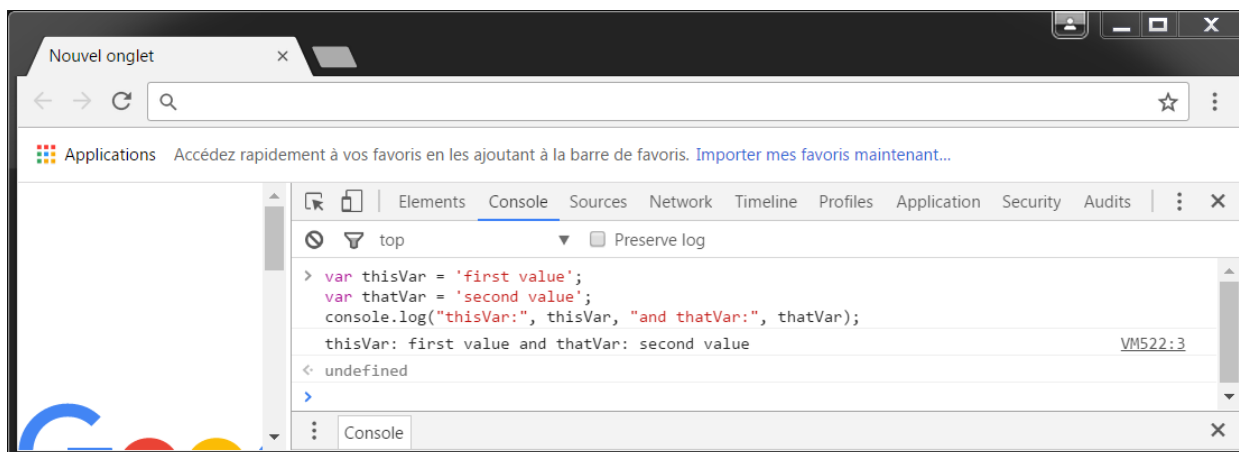


FIGURE 3.4 – Introduction de plusieurs valeurs.

3.2.4 Commentaires

Les commentaires sont une partie essentielle pour avoir un bon style de programmation. Un code intelligemment commenter est plus facile à comprendre et facile à examiner plus tard après l'avoir développé. Les commentaires sont insérés dans le code en utilisant certains caractères qui marquent le début et la fin des commentaires. Le JavaScript présente deux genres de commentaires : Les commentaires sur une ligne et les commentaires sur plusieurs lignes. On peut utiliser l'un ou l'autre ou les deux à la fois dans un même script.

```
// Deux slash pour un commentaire sur une ligne
/* Un slash et une étoile pour un commentaire
sur plusieurs lignes.
*/
```

Les commentaires sont ignorés et ne sont pas pris en compte lors de l'exécution. Cependant, les commentaires qui commencent par deux slash ne peuvent être placés au début de l'instruction car le JavaScript ne peut pas savoir où est la fin du commentaire et le début de l'instruction comme le montre l'exemple suivant :

```
// L'accélération gravitationnelle est : var g=9.8 ; //m/s^2
```

Les syntaxes correctes du code pour l'exemple précédent sont :

— La première :

```
// L'accélération gravitationnelle est :
var g=9.8 ; //m/s^2
```

— La deuxième :

```
/* L'accélération gravitationnelle est : */var g=9.8 ; //m/s^2
```

3.2.5 Variables

Dans le JavaScript le mot-clef `var` (instruction) est utilisé pour déclarer des variables et leur identificateur comme suit :

```
var nom-de-la-variable = valeur ;
```

Il est inutile de déclarer le type de variable, car on peut immédiatement déclarer une valeur par défaut.

Exemple :

```
var g=9.8 ;  
//Ou bien :  
var g; g=9.8 ;  
g="accélération gravitationnel." ;
```

L'instruction `var` demande à l'interpréteur de JavaScript de mettre un espace de côté pour la variable nommée « `g` », pendant la déclaration il n'est pas encore clair quelle genre d'information « `g` » va représenter.

En utilisant l'instruction `var`, on peut conserver tout ce qu'on veut dans le contenu (valeur) de l'identificateur « `g` », et il est possible de changer sa valeur et sa nature. Comme dans l'exemple précédent, on a d'abord attribué à « `g` » un flottant 9.8 ensuite une chaîne de caractère "accélération gravitationnel".

Les instructions de déclaration de données dans le JavaScript réservent le nom de l'identificateur associé à la valeur mais sans la nature de la donnée, car le JavaScript déduit le type de la donnée à partir du contenu de la variable en cours.

Exemple :

```
var nombre1 = 5 ;  
nombre1 = 3 ;  
nombre1 = nombre1 + 5 ; // 3 + 5 = 8  
nombre1 = nombre1 - 6 ; // 8 - 6 = 2  
nombre2 = nombre1 * 10 ; //2 (multiplier par)10=20  
nombre3 = nombre2/nombre1 ; //20(diviser par)2=10  
nombre3 = "C'est une chaîne de caractère" ; //"C'est une chaîne de caractère"
```

3.2.5.1 Types de variables

— `var Integer = 12; // nombre de 32-bit .`

```

— var Long = 9310141419482; // nombre de 64-bit .
— var Float = 5.5; // nombre (décimal) avec une virgule flottant de 32-bit .
— var Double = 9310141419482.22; // nombre avec une virgule flottant 64-bit.
— var Boolean1 = true; // 1-bit true/false (0 ou 1) .
— var Boolean2 = false; // false.
— var String = "C'est une chaine de caractère" ; // "C'est une chaine de caractère" .
— var NaN_Exemple = 0/0 ; // NaN .
— var Objet = null ; // null .

```

3.2.6 Opérateurs arithmétiques

Les opérateurs arithmétiques de JavaScript sont :

- L'addition s'effectue en utilisant le traditionnel symbole "+" (plus).
- La soustraction avec le symbole "-" (moins).
- La multiplication avec le symbole "*" (étoile) .
- La division se sert du symbole "/" (slash) .
- L'opérateur modulo qui donne le reste de la division en utilisant le symbole "%" (pour-cent).

Ces opérateurs sont appelés opérateurs binaires et signifie qu'ils exigent deux opérandes.

Exemple :

```

var a = b = c = 2 ; // On déclare et on initialise a, b et c à 2
var d = a / (b + c) - a * 4 ; // d = 2 / (2+2) - 2 * 4 = 1 / 2 - 8 = - 7.5
var e = 3 * (3 + 3) ; // e = 3 * 6 = 18

```

Il existe aussi les opérateurs abrégés : +=, -=, *=, /= et %= qui sont une combinaison entre un opérateur arithmétique et l'opérateur d'assignement « = ». Ils sont utilisés lorsque l'opération s'effectue sur une seule variable. Voici un exemple très simple :

Exemple :

```

var a = 3 ;
var b = a - 2 ; // b = 1
a -= b ; // a = a - b = 3 - 1 = 2
a += b ; // a = a + b = 2 + 1 = 3
b *= a ; // b = b * a = 1 * 3 = 3
b /= b ; // b = b / b = 3 / 3 = 1

```

L'opérateur d'addition est aussi utilisé comme opérateur de concaténation pour les chaînes de caractères, c'est-à-dire qu'il rassemble deux chaînes de caractères pour n'en former qu'une seule.

Exemple :

```
var a="Opérateur de" + " " + "concaténation" ;// "Opérateur de concaténation"
```

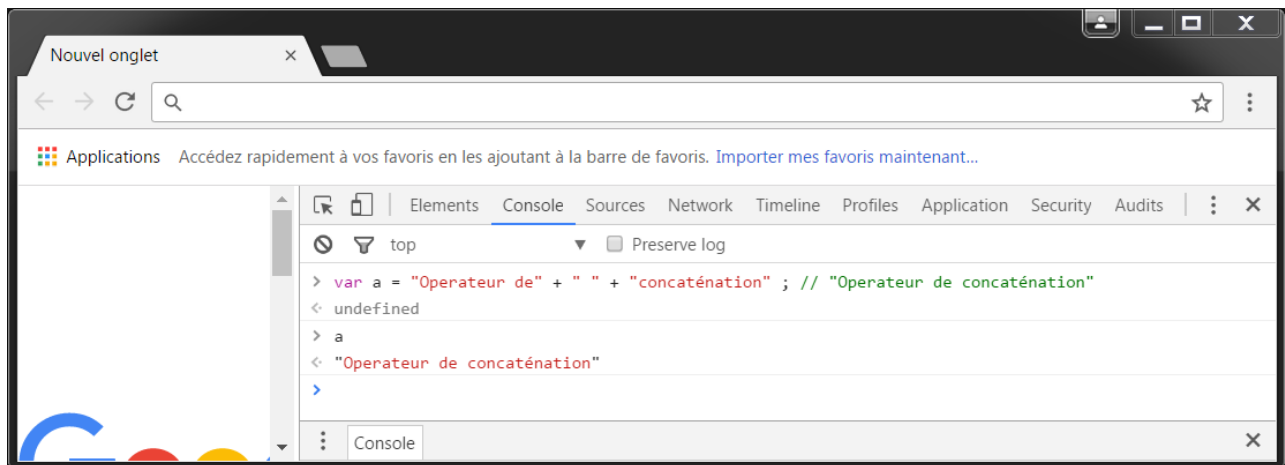


FIGURE 3.5 – Opérateur de concaténation « + ».

On peut aussi utiliser l'opérateur « += » pour la concaténation et le résultat sera pareil que le précédent, en utilisant l'opérateur « + ».

Exemple :

```
var chaine_a = "Bonjour" ;
var chaine_b = "tout le monde" ;
chaine_a += chaine_b ; // chaine_a vaut Bonjour tout le monde
```

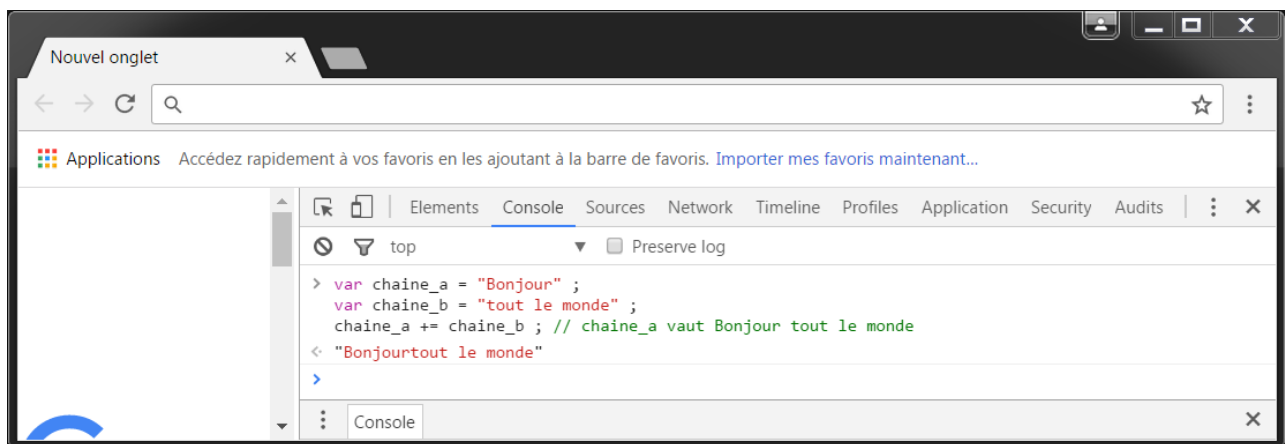


FIGURE 3.6 – Opérateur de concaténation « += ».

3.2.7 Opérateur d'affectation

L'opérateur d'affectation de JavaScript est symbolisé par « = », il permet d'affecter une valeur (nombre ou chaîne de caractères) à une variable. La fonction de l'opérateur d'affectation est d'évaluer l'expression qui se trouve à sa droite et d'affecter le résultat à

l'identificateur qui se trouve à sa gauche.

Par exemple, si on veut coder l'expression algébrique $a + b = x$ dans le JavaScript, il faut faire attention à ne pas la saisir de cette manière, car l'identificateur doit se trouver à gauche du symbole « = », si non la syntaxe va générer des erreurs. Alors l'équivalent de cette expression algébrique dans le JavaScript est : $x = a + b$

Exemple :

```
var nombre = 18 ;  
var chaine = "Bonjour tout le monde" ;
```

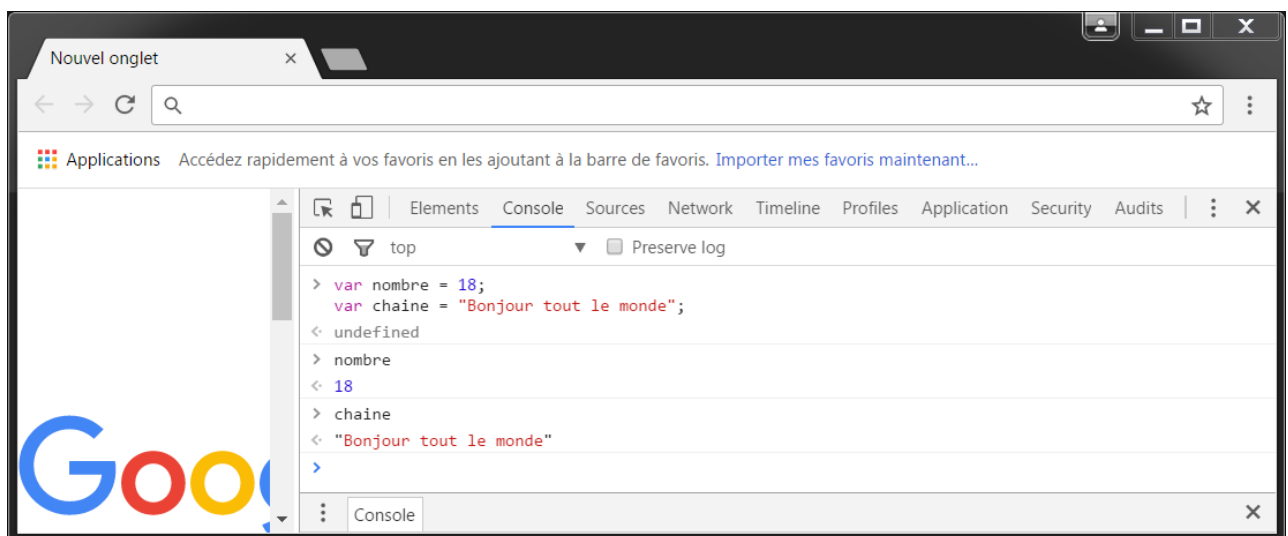


FIGURE 3.7 – Opérateur d'affectation.

Exemple :

```
var a = 3 ;  
var b = 2 ;  
x = a + b ;
```

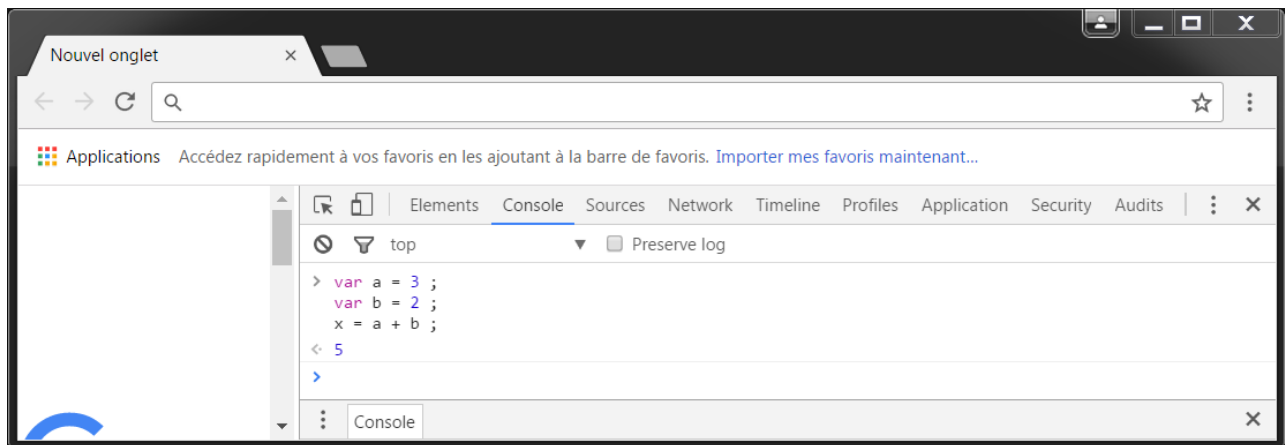


FIGURE 3.8 – Opérateur d’affectation pour une expression algébrique.

3.2.8 Opérateur d’incrément et de décrémentation

- L’opérateur d’incrément est « ++ », il permet d’ajouter « 1 » à la valeur de la variable à laquelle il est appliqué.

Exemple :

```
var i = 2 ;  
i++ // ou bien ++i
```

Dans la première ligne on a affecté à « i » la valeur « 2 ».

Dans la deuxième ligne on a incrémenté i de « 1 » c’est-à-dire la valeur i est « 3 ».

- L’opérateur de décrémentation est « -- », il permet de soustraire « 1 » de la valeur de la variable à laquelle il est appliqué.

Exemple :

```
var k = 3 ;  
k-- // ou bien --k
```

Dans la première ligne on a affecté à « k » la valeur « 3 ».

Dans la deuxième ligne on a décrémenté k de « 1 », c’est-à-dire la valeur k est « 2 ».

Remarque :

Dans les expressions algébrique « ++x » et « x++ » donnent deux résultats différents.

« ++x » : l’opération se fait immédiatement.

« x++ » : l’opération ne se fait pas immédiatement.

Par exemple :

```
var x = 3 ;  
y = (++x) + 3 ;
```

Dans cet exemple `x` est incrémenté avant que l'opérateur « `=` » affecte la valeur de l'expression à `y`, c'est-à-dire que `x` vaut 4 et ensuite `y` vaut 7. Par contre dans l'exemple suivant l'opérateur « `=` » affecte la valeur de l'expression à `y` avant que `x` soit incrémenté de 1, c'est-à-dire que `x` reste égale à 3 et ensuite `y` vaut 6 et ensuite `x` vaut 4.

Par exemple :

```
var x = 3 ;  
y = (x++) + 3 ;
```

3.2.9 Opérateurs de comparaison

Les opérateurs de comparaison permettent de comparer deux valeurs et renvoient une valeur de type booléen (`true` / `false`) ou (`vrai` / `faux`) selon les cas.

- `>` : strictement supérieur
- `<` : strictement inférieur
- `>=` : supérieur ou égal
- `<=` : inférieur ou égal
- `==` : strictement égal
- `!=` : strictement différent

Exemple :

```
var a = b = 10 ;  
document.write (a>b) ; // false  
document.write (a>=b) ; // true  
document.write ((a+1)>b) ; // true  
document.write (b!=a) ; // false  
document.write (b==a) ; // true  
document.write (b<=a) ; // true
```

L'instruction `document.write()` permet d'écrire une chaîne de caractères dans le document. Le paramètre de cette instruction peut être une chaîne de caractères classique, du code HTML et même du JavaScript, elle est utilisée pour afficher un résultat.

Remarque :

Il faut faire attention à ne pas confondre l'opérateur « `==` » qui est un opérateur de comparaison avec l'opérateur « `=` » qui est un opérateur d'affectation. Dans l'exemple précédent l'opérateur « `==` » a comparé entre la valeur de `a` et la valeur de `b`, et il a renvoyé « `true` » qui veut dire (`vrai`) en français c'est-à-dire que `a` est égale à `b`.

L'opérateur « `!` » est un opérateur de négation, on l'utilise en le combinant avec les autres opérateurs de comparaison ainsi le `vrai` devient `faux` et vice-versa.

Exemple :

```
var a = 3 ;  
var b = 5 ;  
document.write (a < b) ; //affiche true car (3 < 5)  
document.write(!(a < b)) ; //affiche false car l'expression(!(a<b))  
équivalent à(a>=b)
```

3.2.10 Structures conditionnelles

Il est possible de traduire le langage parlé vers la syntaxe des instructions programme, par exemple : Si x est supérieur à y, alors z = 10, ou si aujourd'hui c'est lundi, alors je pars à l'université.

3.2.10.1 Tests conditionnels avec if et else

La structure de contrôle « if » permet d'exécuter une instruction ou une suite d'instructions seulement si la condition est vraie.

La syntaxe d'une condition « if » est la suivante :

```
if (condition)  
{Bloc d'instructions}
```

Si la condition est vraie alors le bloc d'instruction sera exécuté.

Exemple :

```
var âge = 15 ;  
if (âge >= 18) {  
  document.write ("Vous pouvez entrer") ;  
}
```

La structure « if » peut être complétée d'un « else » permettant de traiter les cas où la condition sera fausse, la syntaxe est :

```
if (condition)  
{Bloc d'instructions 1}  
else  
{Bloc d'instructions 2}
```

Si la condition est vraie alors le Bloc d'instructions 1 sera exécuté, sinon la condition est fausse alors le Bloc d'instructions 2 sera exécuté.

Exemple :

```
var a = 15 ;
if (a >= 0)
{
document.write ("a est positif.") ;
}
else
{
document.write ("a est négatif.") ;
}
```

Remarque :

Les accolades ne sont pas nécessaires lorsque le bloc à exécuter ne nécessite qu'une seule ligne.

On peut aussi rajouter des conditions intermédiaires entre le « if » et le « else » pour tester plus de cas.

L'exemple suivant montre comment gérer simplement les conditions multiples en JavaScript.

Exemples :

```
<html>
<head>
  <meta charset = "utf-8"/>
  <title>Distributeur Automatique</title>
  <script language = "javascript" type = "text/javascript">
var choix = parseFloat (prompt ("Pour un café taper 1\nPour un soda taper 2
    \nPour un crème taper 3\nPour un thé taper 4")) ;
document.write ("Pour "+choix+",") ;
if (choix == 1)
document.write (" vous avez un café") ;
else if (choix == 2)
document.write (" vous avez un soda") ;
else if (choix == 3)
document.write (" vous avez un crème") ;
else if (choix == 4)
document.write (" vous avez un thé") ;
else
document.write (" il n ya rien") ;
document.write (".") ;
  </script>
```

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

Resultat :

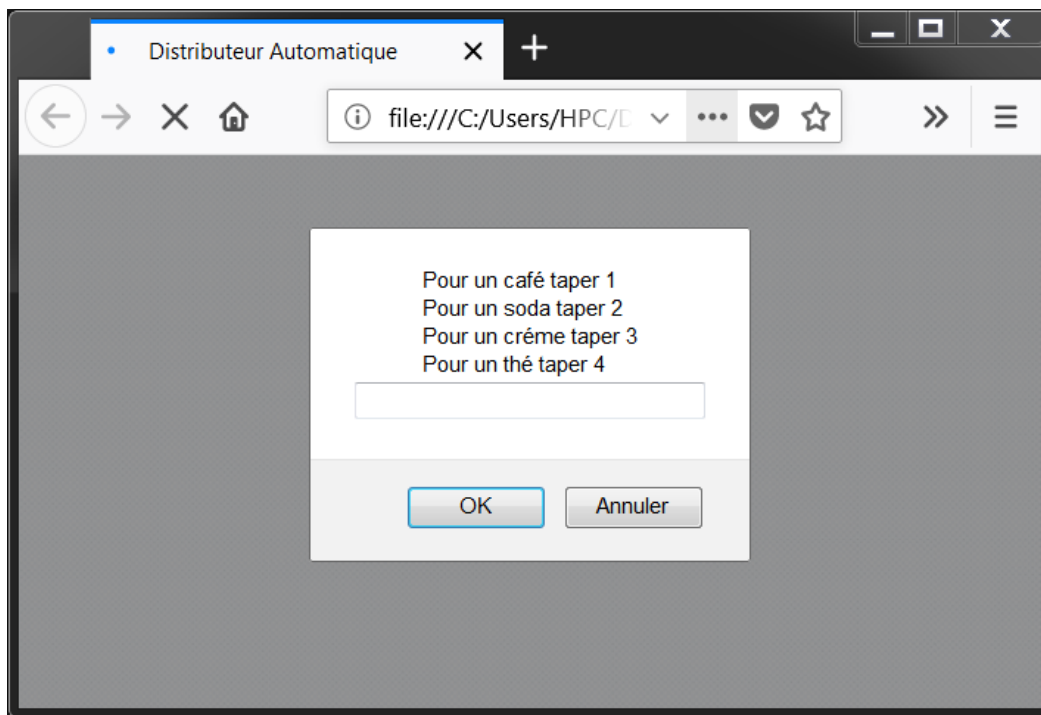


FIGURE 3.9 – Distributeur Automatique.

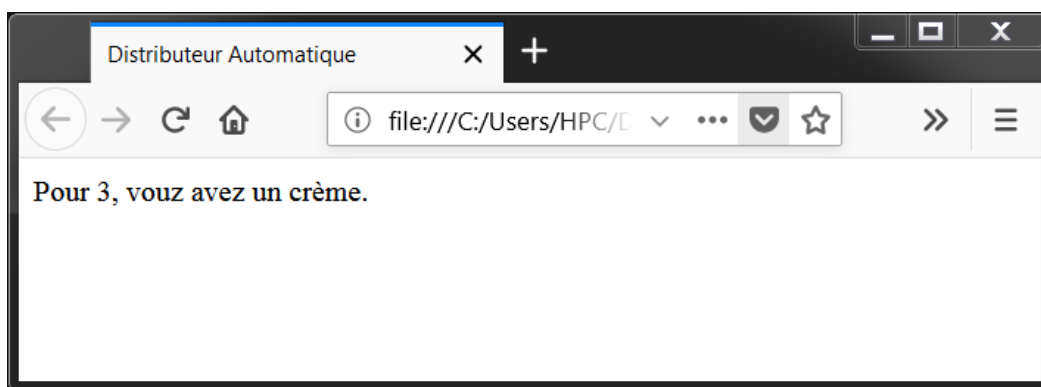


FIGURE 3.10 – Distributeur Automatique.

Pour rallier plusieurs conditions en une seule, il existe deux opérateurs qui le permettent, cela se révèle très pratique mais avec moins de précision.

- && : Veut dire ET.
- || : Veut dire OU.

Par exemple si on veut coder l'expression suivante : Si aujourd'hui c'est dimanche ou bien c'est lundi, je pars à l'université.

Exemple :

```
var a ;  
if ((a == "dimanche") || (a == "lundi"))  
document.write ("je pars à l'université.") ;
```

3.2.10.2 L'instruction switch/case

L'imbrication de « if/else » peut avantageusement être remplacée par une alternative intéressante. En effet, lorsqu'on a un nombre important de cas à vérifier l'instruction « switch » permet de tester plusieurs valeurs pour une expression sans passer par une cascade de « else if ».

Syntaxe :

```
Switch (variable) {  
Case "a" :  
Instruction 1 ;  
Instruction 2 ;  
Instruction 3 ;  
break ;  
case "b" :  
Instructions ;  
break ;  
default :  
Instructions ;  
break ;  
}
```

L'instruction break peut optionnellement être utilisée pour chaque cas et permet de s'assurer que seules les instructions associées à ce cas seront exécutées. Si break n'est pas utilisé, le programme continuera son exécution avec les instructions suivantes (des autres cas de l'instruction switch).

Exemple :

```
<html>  
<head>  
  <title> Nombre de jours </title>  
  <script language="javascript" type="text/javascript">
```

```
var mois=prompt ("Donnez un mois (1-12) :");
switch (mois) {
case "1" :
case "3" :
case "5" :
case "7" :
case "8" :
case "10" :
case "12" :
  alert ("Il y a 31 jours dans ce mois.") ;
  break ;
case "4" :
case "6" :
case "9" :
case "11" :
  alert ("Il y a 30 jours dans ce mois.") ;
  break ;
case "2" :
  alert ("Il y a 28 ou 29 jours dans ce mois.") ;
  break ;
default :
  alert ("Ce mois n'existe pas.") ;
}
</script>
</head>
<body> </body>
</html>
```

Résultat :

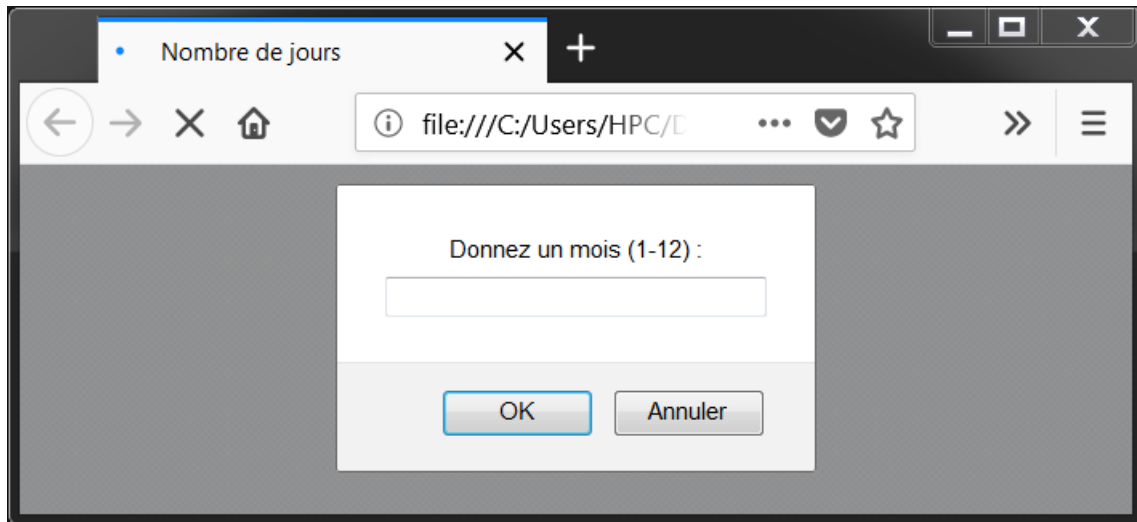


FIGURE 3.11 – Instruction switch/case.

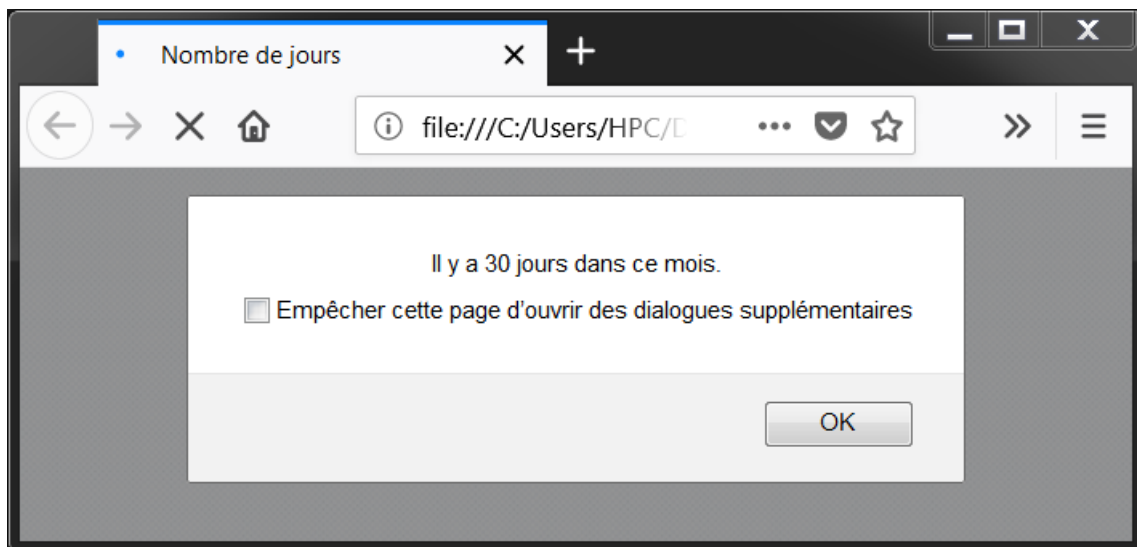


FIGURE 3.12 – Instruction switch/case.

3.2.11 Boucles conditionnelles

Les conditions qui contrôlent l'exécution ou la fin d'une boucle doivent être déterminées par les valeurs déterminées à l'intérieur de la boucle pendant le fonctionnement du Script. Il y a deux genres de boucles conditionnelles : boucles prétest et boucles posttest.

Les instructions des boucles prétest peuvent ou ne peuvent pas être exécutées du tout, car elles dépendent des valeurs originales des variables de la boucle.

Les boucles posttest sont toujours exécutées, et les valeurs des variables sont testées à la fin de la boucle.

3.2.11.1 boucle while

La boucle « while » est une boucle prétest, c'est la plus simple d'utilisation puisqu'elle ne comporte qu'un seul argument. Voici la syntaxe de « while » :

```
while (condition) {  
  (Instructions, incrémentation...)  
}
```

Exemple :

```
var i = 0 ;  
while (i < 5)  
{  
  document.write ("Ligne "+i+"<br />") ;  
  i++ ;  
}  
document.write ("Boucle terminée !") ;
```

Résultat :

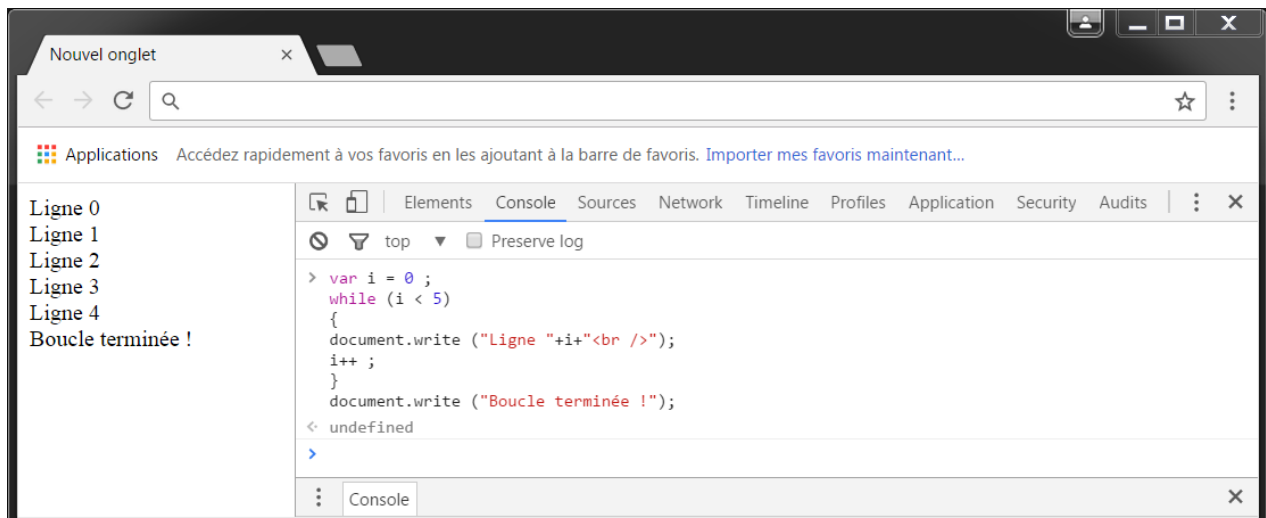


FIGURE 3.13 – La boucle while.

3.2.11.2 boucle do . . . while

La boucle « do . . . while » est une boucle posttest, proche de la boucle « while » puisqu'il s'agit de la même boucle sauf que le bloc d'instruction s'effectuera au minimum une fois et ce même si la condition de la boucle n'est pas vérifiée dès le début.

Exemple :

```
var i = 5 ;  
do { document.write ("Ligne "+i+"<br />") ;  
i++ ;  
}  
while (i < 5) ;  
document.write ("Boucle terminée !")
```

Ici, la condition vaut false dès le début. Un « while » normal aurait donc sauté la condition pour passer à la suite du code. Mais le « do...while » exécute une fois le code.

Résultat :

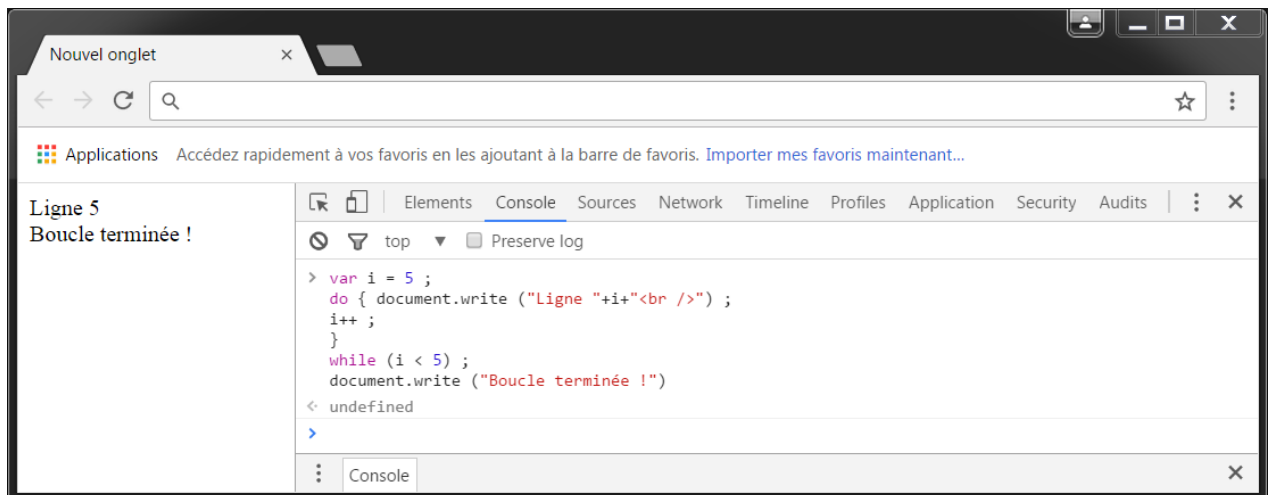


FIGURE 3.14 – La boucle do while.

3.2.11.3 boucle for

La boucle « for » est une structure répétitive et une boucle prétest. Le mot clef « for » est suivi de trois informations à l'intérieur des parenthèses. La première information c'est la valeur initiale du compteur, elle peut être identifiée par un nom. La deuxième information c'est la condition qui établit jusqu'à quand la boucle doit continuer à être exécuter, c'est-à-dire jusqu'à ce que la condition soit vraie. La troisième information contrôle combien de fois le compteur a été incrémenté ou décrémenté.

Syntaxe :

```
for (valeur_initiale=valeur initiale du compteur ;  
    Valeur la plus élevé ou la plus basse du compteur ;  
    Control de l'incrémentation ou de la décrémentation du conteur)  
{  
Instructions  
}
```

Ou bien :

```
for (initialisation ; condition ; incrémentation)
{
  Instructions
}
```

Exemple :

```
<html>
  <head>
    <title>Compteur</title>
    <script>
var k ;
document.write ("C'est un simple compteur : "+"<br />") ;
for (k=0 ; k<=10 ; k++)
document.write (k+"<br />") ;
    </script>
  </head>
<body> </body>
</html>
```

Résultat :

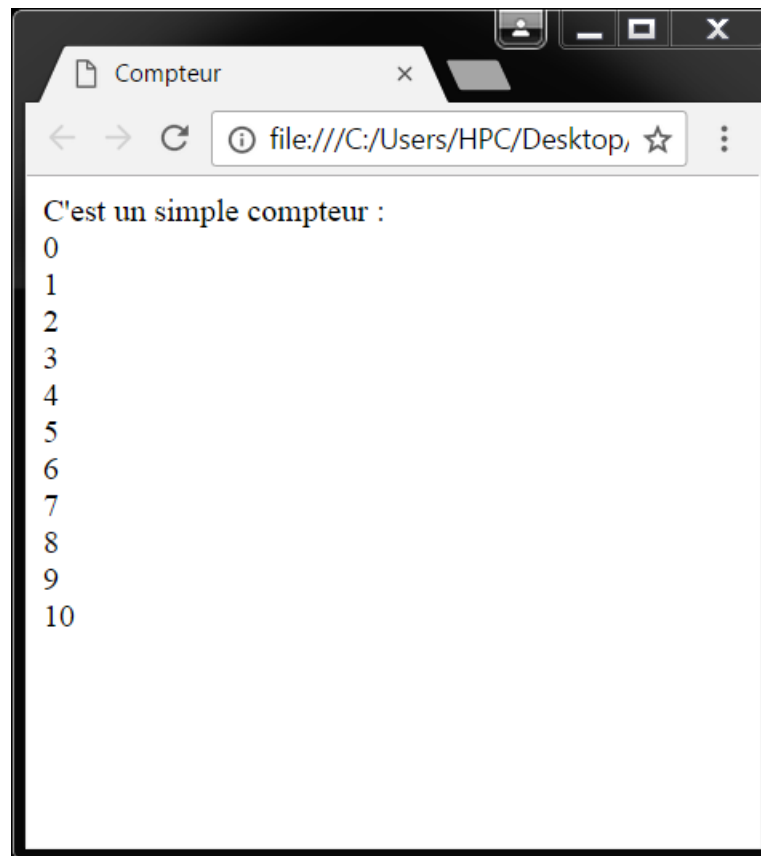


FIGURE 3.15 – Simple compteur.

3.2.11.4 Instruction break

Il arrive parfois que l'on veuille sortir d'une boucle alors même que la condition de la boucle est encore vraie. Pour cela, on peut faire appel à l'instruction « break ». Lorsque cette instruction est exécutée, la boucle se termine immédiatement et le code suivant est exécuté.

Exemple :

```
var i = 1 ;  
while (i <= 10)  
{  
  if (i == 5) break ;  
  i++ ;  
}  
document.write ("La valeur de i est "+i) ;
```

Résultat :

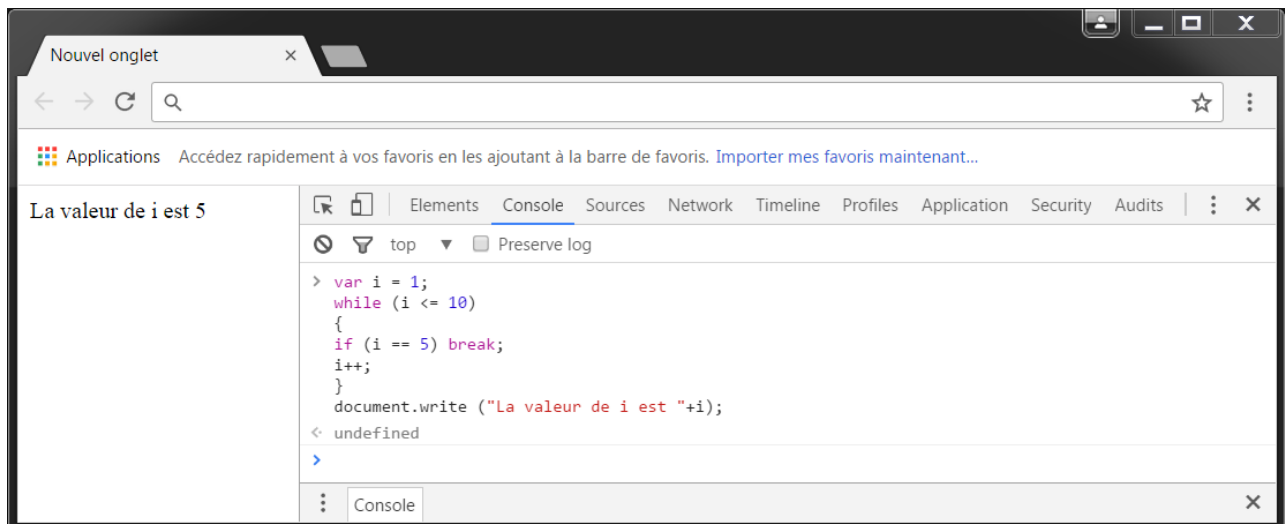


FIGURE 3.16 – L’instruction break.

3.3 Tableaux

Le concept des tableaux est important dans la programmation, car il nous donne le moyen d’organiser, d’accéder et de manipuler plusieurs quantités qui sont liées. Le nombre de cases d’un tableau correspond aux adresses appelées indices, et leurs contenu sont appelées éléments. Dans beaucoup de langage de programmation, notamment dans le JavaScript la numérotation des cases du tableau commence par zéro « 0 » plutôt que par un « 1 ». Il existe plusieurs méthodes pour créer un tableau, la plus importante c’est le constructeur `new Array ( )` et voici les syntaxes qui sont possibles :

```
// Première syntaxe
var tableau1 = new Array ( ) ;
// Deuxième syntaxe
var tableau2 = new Array (valeur 1, valeur 2, . . . , valeur n) ;
// Troisième syntaxe
var tableau3 = new Array (10) ;
```

La première syntaxe permet de créer un tableau vide nommé `tableau1`, la deuxième syntaxe permet de créer un tableau nommé `tableau2` de `n` éléments avec des valeurs initiales, la troisième syntaxe permet de créer un tableau nommé `tableau3` de 10 éléments sans spécification des valeurs. Il y a aussi une autre méthode intéressante pour créer un tableau, et qui ne fait pas appelle au constructeur `new Array ( )`, et voici les syntaxes possibles :

```
// Première syntaxe
```

```
var mon_tableau1 = [ ] ;  
// Deuxième syntaxe  
var mon_tableau2 = ['valeur 1', 'valeur 2', 'valeur 3', ..., 'valeur m'] ;  
// Troisième syntaxe  
var mon_tableau3 = [, , , , , , , , ,] ;
```

Dans la première et la deuxième syntaxe c'est le même principe que la première méthode avec le constructeur `new Array ( )`, par contre la troisième syntaxe est différente, car elle permet de créer un tableau de 10 éléments sans spécification des valeurs en utilisant neuf « 9 » virgules.

3.3.1 Accès aux valeurs d'un tableau

Pour accéder aux valeurs d'un tableau à indices numériques, la seule possibilité est de passer par l'indice de chacune des valeurs contenues dans ce tableau, sans oublier que l'indice de la première valeur du tableau est zéro « 0 ».

Exemple :

```
var mon_tableau4 = [16.5, 17.5, 20, 18, 19] ;  
mon_tableau4 [2] // accéder à la troisième valeur du tableau  
mon_tableau4 [0] = 31 // ranger le nombre 31 dans la première case  
                    du tableau  
ran = mon_tableau4 [1] // écrire dans la variable rang  
                    le contenu de la 2ème case du tableau  
var x ;  
x = mon_tableau4 [3] + mon_tableau4 [4] ;/*additionner la valeur  
    du quatrième élément du tableau avec la valeur  
    du cinquième élément du tableau*/
```

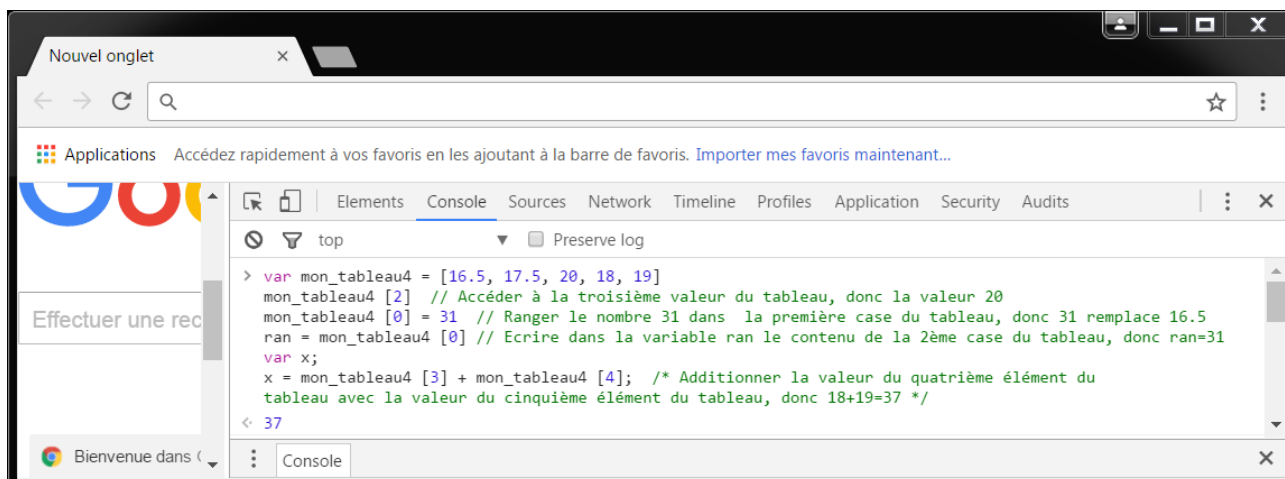


FIGURE 3.17 – Accès au valeur d'un tableau.

Résultat :

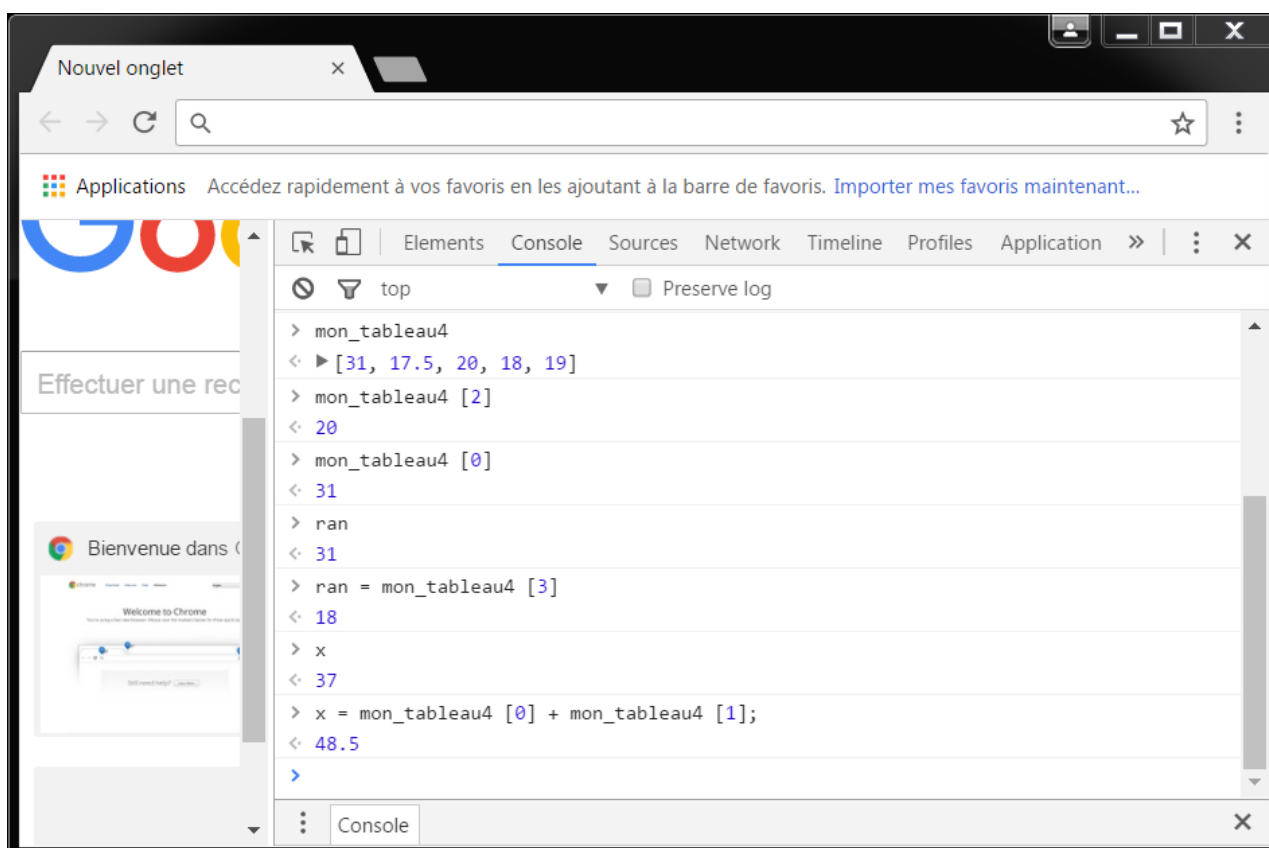


FIGURE 3.18 – Accès au valeur d'un tableau.

3.3.2 Des outils pratiques

Le JavaScript met à disposition des fonctions pour la gestion des tableaux. En voici la liste des fonctions principales :

- Concat (Tableau1, Tableau2, Tableau3,...) : cette méthode permet de concaténer (Coller) plusieurs tableaux pour n'en former plus qu'un seul.

- `join(Tableau)` ou `Tableau.join ( )` ou `Tableau.toString ( )` : renvoie tous les éléments (Valeurs) de `Tableau` sous forme d'une chaîne de caractères.
- `pop(Tableau)` ou `Tableau.pop ( )` : supprime le dernier élément de `Tableau` après avoir retourné sa valeur.
- `Tableau.shift ( )` : supprime le premier élément de `Tableau` après avoir retourné sa valeur.
- `Tableau.unshift (valeur1, valeur2, ...)` : permet d'ajouter des éléments au début.
- `Tableau.push (valeur1, valeur2, ...)` : permet d'ajouter des éléments à la fin de `Tableau`.
- `Tableau.sort ( )` : permet de trier tous les éléments de `Tableau`.
- `Tableau.reverse ( )` : inverse l'ordre des éléments de `Tableau`.
- `Tableau.splice (début, fin, valeur)` : écrase les valeurs des éléments du `Tableau` dont l'indice est compris entre `début` et `fin`.
- `Tableau.slice (début, fin)` : retourne les éléments dont l'indice est compris entre `fin` et `début`.

3.3.3 Parcourir un tableau

Pour lister l'intégralité du tableau, il nous faut utiliser une boucle. Il va nous être utile de connaître la longueur du tableau (le nombre d'indices qu'il possède). Pour cela, on fait appel à la méthode « `length` » de l'objet « `Array` ». Ainsi, on accède aux valeurs de notre tableau grâce à ses indices.

Le principe pour parcourir un tableau est simple : il faut faire autant d'itérations qu'il y a d'éléments. Le nombre d'éléments d'un tableau se récupère avec la propriété « `length` », exactement comme pour le nombre de caractères d'une chaîne de caractères. À chaque itération, on va avancer d'un élément dans le tableau, en utilisant la variable de boucle `i` : à chaque itération, elle s'incrémente, ce qui permet d'avancer dans le tableau élément par élément. Voici un exemple qui permet d'afficher tous les éléments d'un tableau un par un dans une fenêtre d'alerte.

Exemple :

```
var tab = [3, 4, 5, 6, 7, 8, 9] ;  
for (var i = 0 ; i < tab.length ; i++) {  
  alert (tab[i]) ;  
}
```

Exemple : Cet exemple permet de remplir un tableau avec une seule valeur.

```
var a = 13 ;  
var mon_tableau1 = new Array(10) ;  
for (i = 0 ; i < mon_tableau1.length ; i++)  
{  
mon_tableau1 [i] = a ;  
}
```

Résultat :



FIGURE 3.19 – Parcourir un tableau.

3.4 Objets

Le JavaScript possède des objets natifs, comme : String, Boolean et Array mais nous permet aussi de créer nos propres objets avec leurs propres méthodes et propriétés.

3.4.1 Définition

Un objet est un concept, une idée ou une chose. Un objet possède une structure qui lui permet de pouvoir fonctionner et d'interagir avec d'autres objets. Cette structure contient trois parties distinctes [14] :

- Un constructeur.
- Des propriétés.
- Des méthodes.

3.4.2 Constructeur

Le constructeur est un code qui est exécuté quand on utilise un nouvel objet, il va contenir la structure de base de l'objet. Un constructeur est donc une fonction invoquée au moment de la création de l'objet et qui permet d'effectuer des actions comme définir diverses variables dès le départ au sein même de l'objet (comme le nombre de caractères d'une chaîne de caractères). La syntaxe d'un constructeur est la même que celle d'une fonction, alors pour les différencier, on met en majuscule la première lettre d'un constructeur, exactement comme les objets natifs de JavaScript, par exemple : Array, Data ... [14]

3.4.2.1 Syntaxe d'un constructeur

```
function Personne ( ) {  
  // Code du constructeur  
}
```

3.4.2.2 Mot clé "this"

Le code du constructeur peut contenir des particularités comme le mot clé « this », qui fait référence à l'objet dans lequel il est exécuté. Grâce à « this », il est possible de définir les propriétés d'un objet. Les paramètres d'un constructeur vont être détruits à la fin de l'exécution de ce dernier, alors que les propriétés définies par le biais de « this » vont rester présentes.

Exemple :

```
// Définition de l'objet Person via un constructeur  
function Personne (nom, age, sex, parent, travail, amis) {  
  this.nom = nom ;  
  this.age = age ;  
  this.sex = sex ;  
  this.parent = parent ;  
  this.travail = travail ;  
  this.amis = amis ;  
}  
  
// On crée des variables qui vont contenir une instance de l'objet Personne :  
var seb = new Personne ('Sébastien', 23, 'm', 'aîné',  
                        'Javascipteur', 'ludovic');  
var lau = new Personne ('Laurence', 19, 'f', 'soeur',  
                        'Sous-officier', 'pauline') ;  
  
alert (seb.age) ;
```

Résultat :

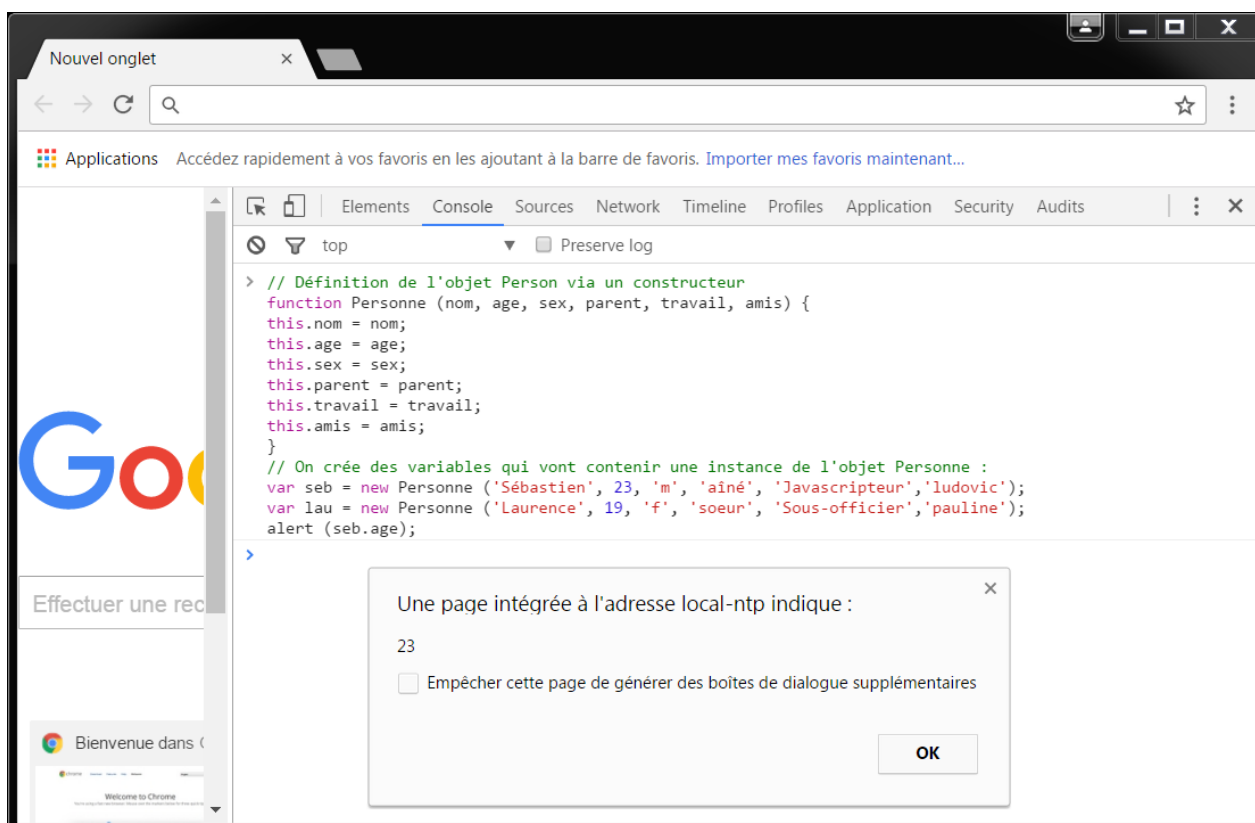


FIGURE 3.20 – Le mot clé this.

« this.amis » affecte une propriété « amis » à objet « personne », tandis que le paramètre « amis » n'est qu'une simple variable qui sera détruite à la fin de l'exécution du constructeur.

3.4.2.3 Mot clé "new"

Pour pouvoir utiliser un objet, on définit une variable qui va contenir une instance de l'objet c'est-à-dire une copie. Pour indiquer au JavaScript qu'il faut utiliser une instance, on utilise le mot-clé « new ».

Le mot-clé « new » ne signifie pas créer un nouvel objet mais signifie créer une nouvelle instance de l'objet, ce qui est très différent. On ne fait que créer une instance, une copie, de l'objet initial, ce qui nous permet de conserver l'objet en question.

Il est possible de faire un test pour savoir si la variable « seb » est une instance de « Personne ». Pour ce faire, il convient d'utiliser le mot-clé « instanceof », comme suit :

```
alert (seb instanceof Personne) ;
```

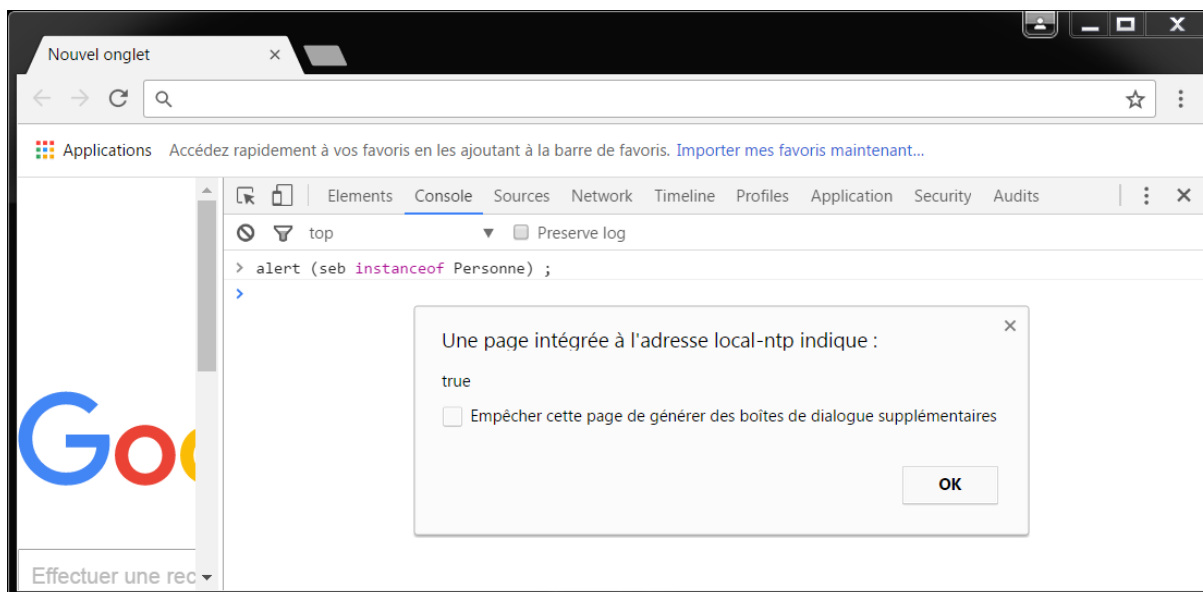


FIGURE 3.21 – Le mot clé instanceof.

Il est possible de récupérer le premier paramètre « Personne » qui est le « nom » en faisant :

```
lau.nom
```

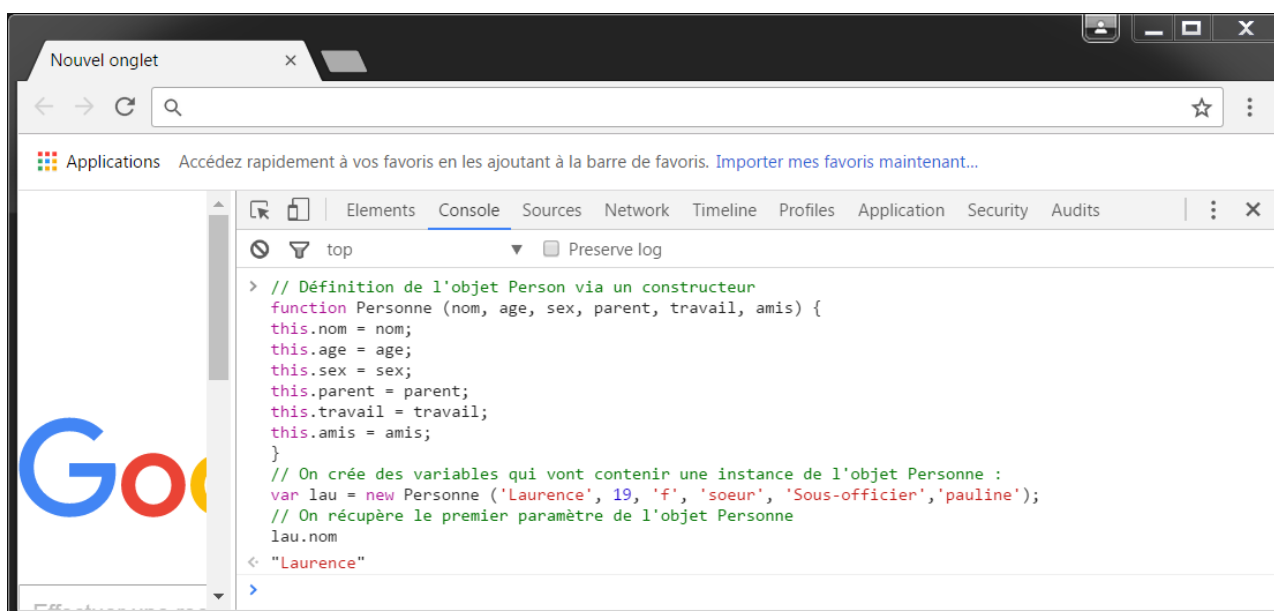


FIGURE 3.22 – Récupération du paramètre « nom » de l'objet « Personne ».

3.4.3 Propriétés

Une propriété est toute valeur placée dans une variable au sein de l'objet, donc une propriété est une variable contenue dans l'objet. Cette variable contient des informations

nécessaires au fonctionnement de l'objet. Le contexte d'usage de cette variable se limite à l'objet dans lequel elle est contenue, ce qui confère à l'objet un état [14].

Dans l'exemple suivant, on a créé un objet string à qui on peut afficher le nombre de ses caractères par la propriété `length` comme suit :

```
chaine_de_caractère = "Voici une chaine de caractère" ;  
alert (chaine_de_caractère.length) ;
```

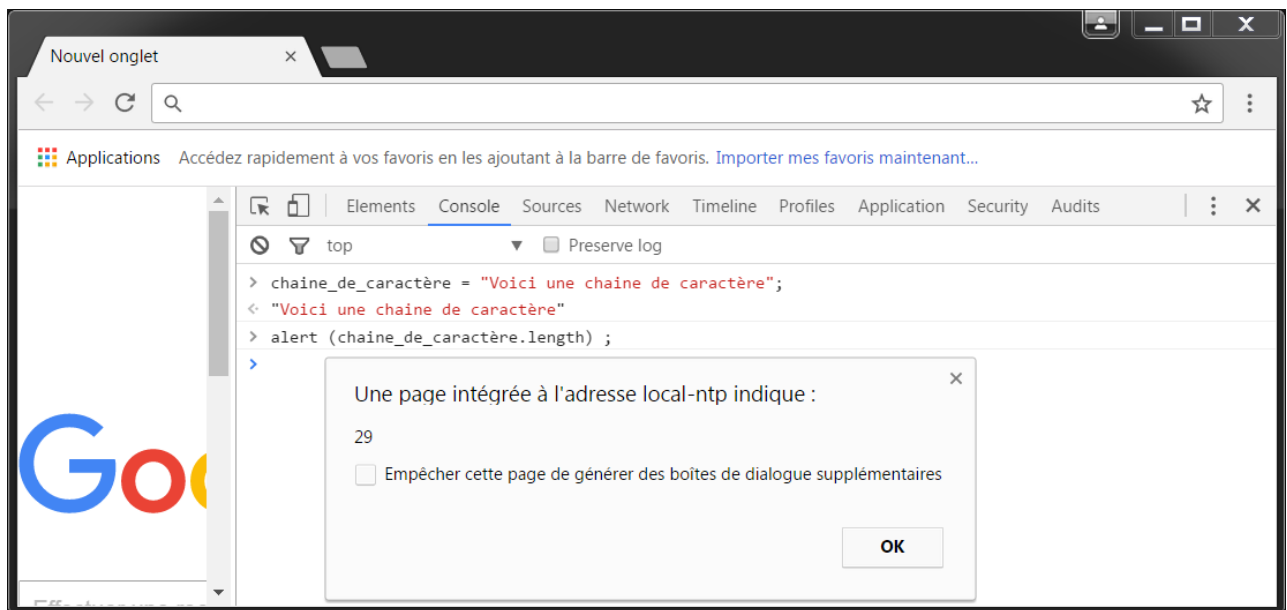


FIGURE 3.23 – La propriété `length`.

3.4.3.1 Modifier les données

Une fois la variable définie, on peut modifier les propriétés exactement comme s'il s'agissait d'un simple objet littéral.

Exemple :

```
var pie = new Personne ('pierre', 25, 'm', 'aîné', 'footballeur', [ ]) ;  
pie.age = 45 ; // On change l'âge  
pie.travail = 'acteur' ; // On change le travail
```

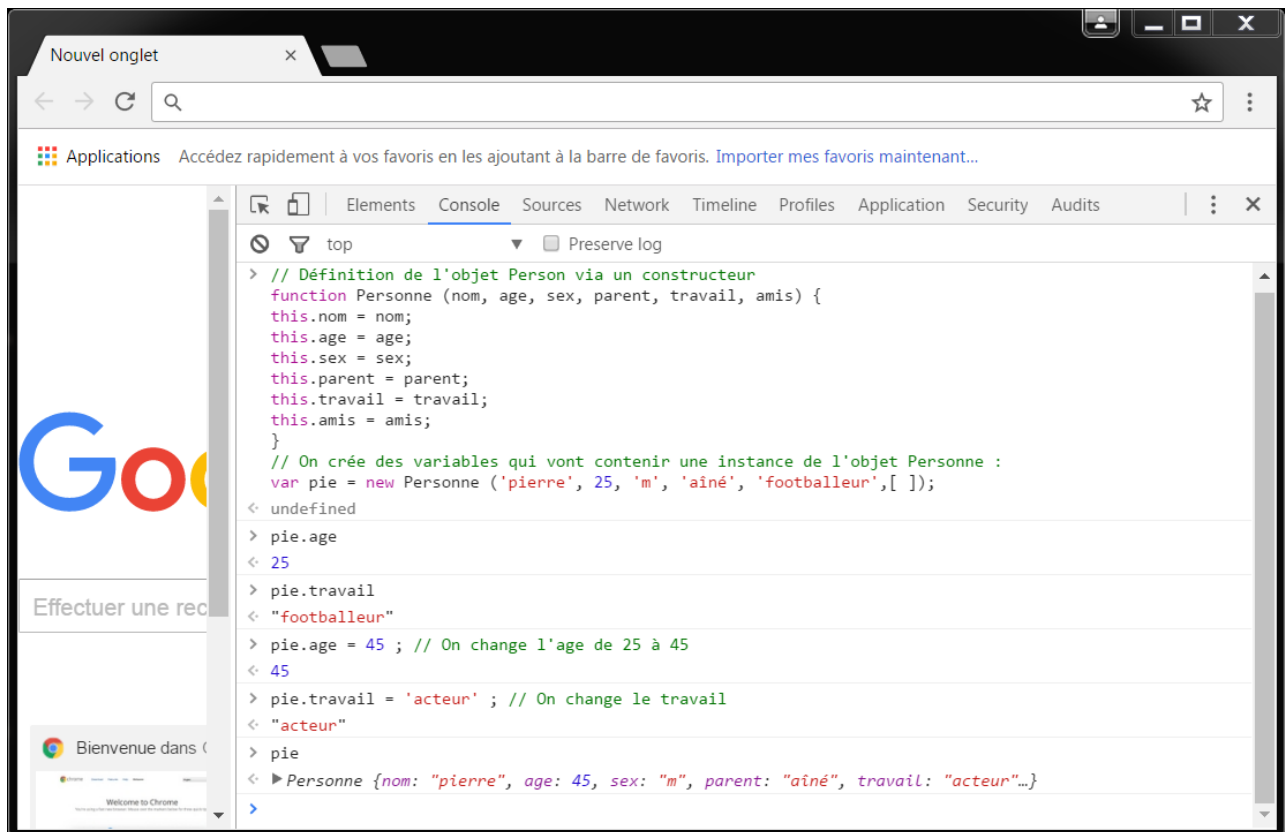


FIGURE 3.24 – Modification des données.

3.4.4 Méthodes

Il est possible de modifier l'objet. Cela se fait par l'intermédiaire des méthodes. Les méthodes sont des fonctions contenues dans l'objet et qui permettent de réaliser des opérations sur le contenu de l'objet, c'est-à-dire interagir avec l'objet [14].

Par exemple, dans le cas d'une chaîne de caractères, il existe une méthode qui permet de mettre la chaîne de caractères en majuscules.

Exemple :

```
var chaine_de_caractère = 'Ceci est une chaine de caractère' ;
alert(chaine_de_caractère.toUpperCase()) ;
```

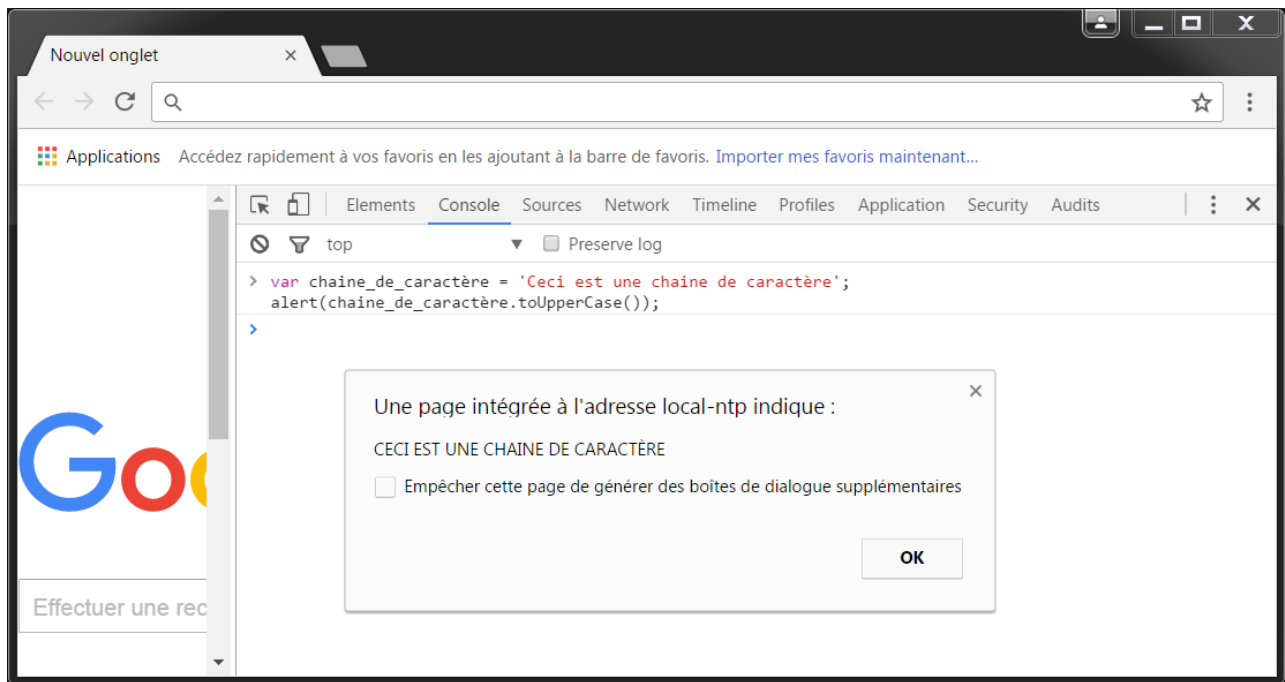


FIGURE 3.25 – Affichage du résultat via la boîte alert.

3.4.4.1 Ajout des méthodes

Il y a deux manières de définir une méthode pour un objet : dans le constructeur ou via prototype. Définir les méthodes directement dans le constructeur est facile puisque c'est nous qui créons le constructeur. La définition de méthodes via prototype (mécanisme au sein de JavaScript qui permet aux objets JavaScript d'hériter des propriétés d'autres objets) est utile surtout si on n'a pas créé le constructeur : ce sera alors utile pour ajouter des méthodes à des objets natifs comme « String » ou « Array ».

3.4.4.2 Définir une méthode dans le constructeur

```
function Personne (nom, age, sex, parent, travail, amis) {  
  this.nom = nom ;  
  this.age = age ;  
  this.sex = sex ;  
  this.parent = parent ;  
  this.travail = travail ;  
  this.amis = amis ;  
  this.addamis = function (nom, age, sex, parent, travail, amis)  
  {  
    this.amis.push (new Personne (nom, age, sex, parent, travail, amis)) ;  
  } ;  
}
```

Le code de cette méthode permet d'ajouter un objet « Personne » dans le tableau des amis.

3.4.4.3 Ajout d'une méthode via prototype

Lors de la définition d'un objet, un sous objet appelé prototype est attribué automatiquement à l'objet. Le prototype permet d'ajouter des méthodes aux objets. Cela dit, il ne faut pas confondre l'objet « prototype » avec la bibliothèque de JavaScript prototype [14].

Voici comment ajouter une méthode « addamis () » à l'objet « personne » déjà créé précédemment.

```
Personne.prototype.addamis = function(nom, age, sex,  
                                     parent, travail, amis){  
this.amis.push(new Personne(nom, age, sex,  
                             parent, travail, amis)) ; }
```

L'ajout de méthodes par prototype a l'avantage d'être indépendant de l'objet, c'est-à-dire qu'on peut définir un objet dans un fichier, et ajouter des méthodes dans un autre fichier (pour autant que les deux fichiers soient inclus dans la même page Web).

3.4.5 Accès aux propriétés et aux méthodes

La présence d'un point permet d'accéder aux propriétés et aux méthodes d'un objet. Ainsi, quand nous écrivons : « (chaine_de_caractère.toUpperCase()); » ou quand nous écrivons : « chaine_de_caractère.length », nous demandons au JavaScript d'accéder aux méthodes et aux propriétés de l'objet, pour fournir le nombre de caractères contenus dans « chaine_de_caractère », ou pour mettre le contenu de « chaine_de_caractère » en majuscule. La propriété « length » contient ce nombre et a été définie quand nous avons créé l'objet. Ce nombre est également mis à jour quand on modifie la chaîne de caractères [14].

3.4.6 Déclaration d'un objet littérale

En JavaScript, il est possible de créer un objet sans passer par un constructeur, simplement, en spécifiant ses propriétés et méthodes entre deux accolades ou entre deux parenthèses. Il s'agit d'une forme littérale.

On peut utiliser les deux syntaxes suivantes [14] :

```
var mon_objet = new Object (  
  nom1 : valeur1,  
  nom2 : valeur2  
);
```

Ou bien :

```
var mon_objet = {  
  nom1 : valeur1,  
  nom2 : valeur2  
};
```

Ainsi, l'objet est une sorte de liste d'association nom/valeur qui définit les initialisations des éléments à ajouter à l'objet, on écrit :

```
var mon_object = {  
  élément1 : 'Texte 1',  
  élément2 : 'Texte 2' } ;
```

La séparation des éléments se fait par des virgules comme pour un tableau.

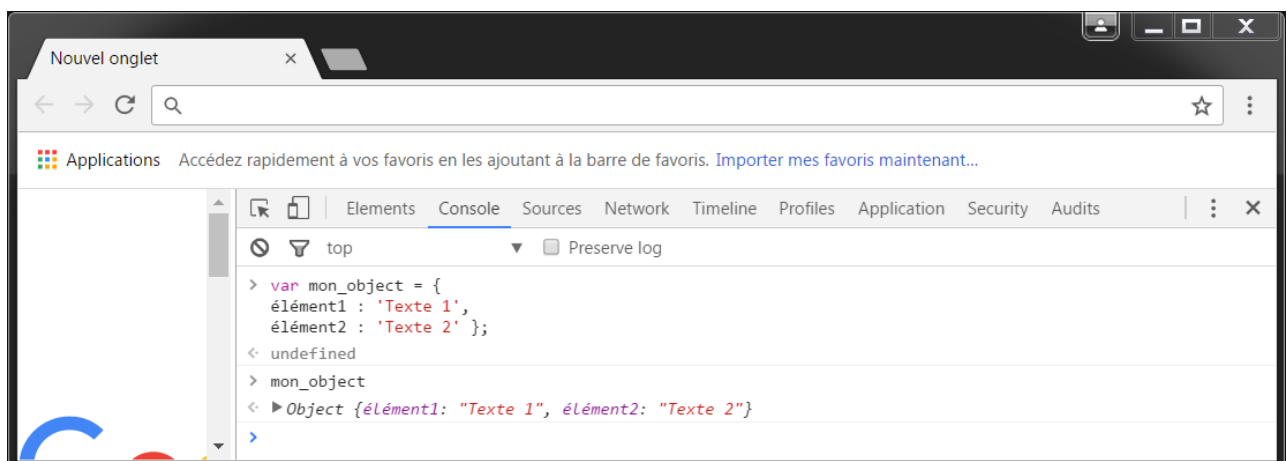


FIGURE 3.26 – Objet littérale.

Comme l'indique ce code, il suffit de taper l'identifiant souhaité suivi de deux points et de la valeur à lui attribuer.

Exemple :

```
var Classement = {  
  Premier : 'Sébastien',
```

```
Deuxième : 'Laurence',  
Troisième : 'Ludovic',  
Quatrième : 'Pauline',  
Cinquième : 'Guillaume'  
} ;
```

Résultat :

```
Object {Premier:"Sébastien", Deuxième:"Laurence",  
        Troisième:"Ludovic", Quatrième:"Pauline", Cinquième:"Guillaume"}
```

Les identifiants sont en réalité des propriétés, exactement comme la propriétés « length » d'un tableau ou d'une chaîne de caractère. Pour récupérer la valeur d'un élément, il suffit de saisir comme suit :

```
Classement.Premier ;
```

Résultat :

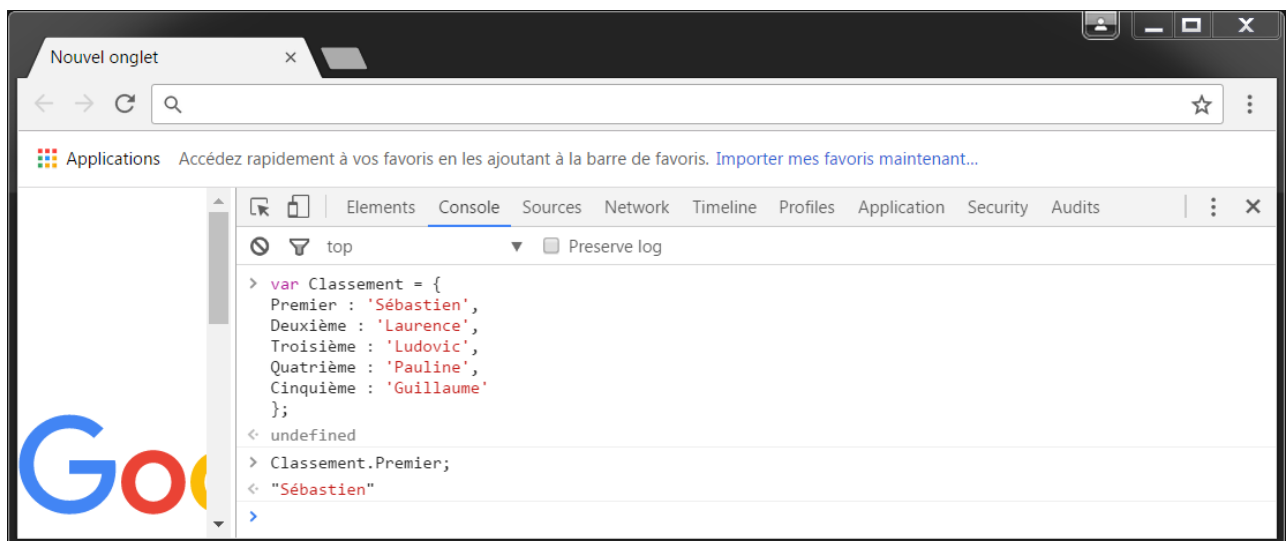


FIGURE 3.27 – Récupération de la valeur d'un élément.

Comme pour les tableaux, il existe une autre manière pour récupérer les éléments des objets qui est pareil à celle du tableau sauf qu'on remplace l'indice par le nom de la propriété. Par exemple :

```
Classement ['Premier'] ;
```

3.4.6.1 Ajout des éléments

Il est possible d'ajouter un élément en spécifiant un identifiant qui n'est pas encore présent dans l'objet, par exemple on veut ajouter un sixième au classement dans l'exemple précédent.

```
Classement ['sixième'] = 'Didier' ; //« Didier » est ajouté  
et est accessible via l'identifiant « uncle »
```

Résultat :

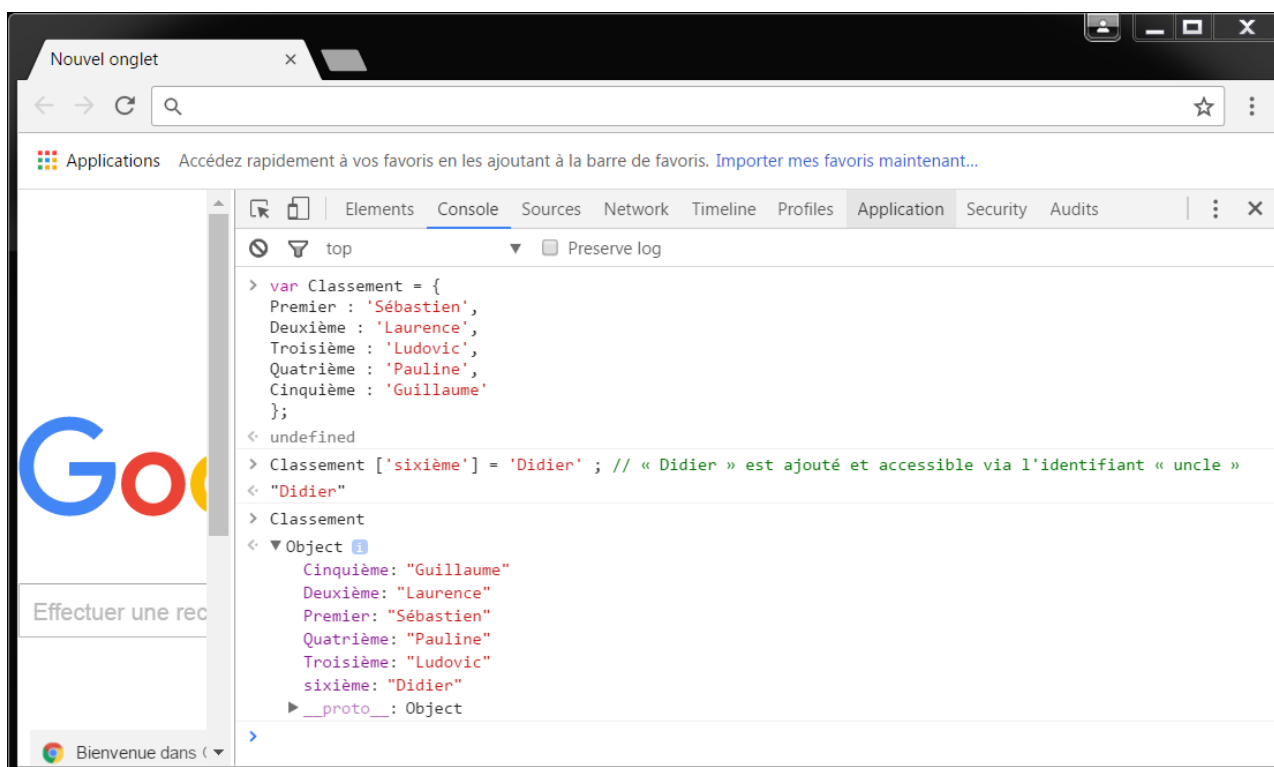


FIGURE 3.28 – Ajouter des éléments.

3.4.6.2 Parcourir un objet

Il n'est pas possible de parcourir un objet littéral avec une boucle `for`. Parce que une boucle `for` est surtout capable d'incrémenter une variable numérique, ce qui est d'aucune utilité dans le cas d'un objet littéral puisque nous devons posséder un identifiant. En revanche, la boucle « `for in` » se révèle très intéressante, et ne sert qu'à parcourir un objet [14].

Le fonctionnement est quasiment le même que pour un tableau excepté qu'ici il suffit de fournir une « variable clé » qui reçoit un identifiant (au lieu d'un index) et de spécifier l'objet à parcourir :

```
for (var nom_indice in Classement)
{
document.write (Classement [nom_indice]) ;
}
```

Résultat :



FIGURE 3.29 – Parcourir un objet.

Il existe deux méthodes peut affecter des valeurs :

— Soit directement en utilisant le nom de l'indice :

```
Classement ['Troisième'] = "Alex"
```

— Soit via la boucle « for in » :

```
for (var id in Classement)
{
  Classement [id] = 0 ;
}
```

3.5 Boite d'alerte

Il existe une boite de dialogue qui a pour principal objectif d'alerter le visiteur de quelque chose. Il s'agit de la boite d'alerte.

Voici la syntaxe de la boite d'alerte :

```
alert ( ' ' ) ;
```

La boite d'alerte affiche tous ce que le développeur met entre les parenthèses du code de la boite d'alerte. L'emploi du caractère spécial « \n » permet de faire des sauts de ligne.

3.5.1 Caractéristiques

La boite d'alerte est munie d'un bouton « OK » pour quitter lors du clique, et d'une croix de fermeture. Elle s'affiche différemment selon le navigateur utilisé.

Exemple : Navigateur « Google Chrome »

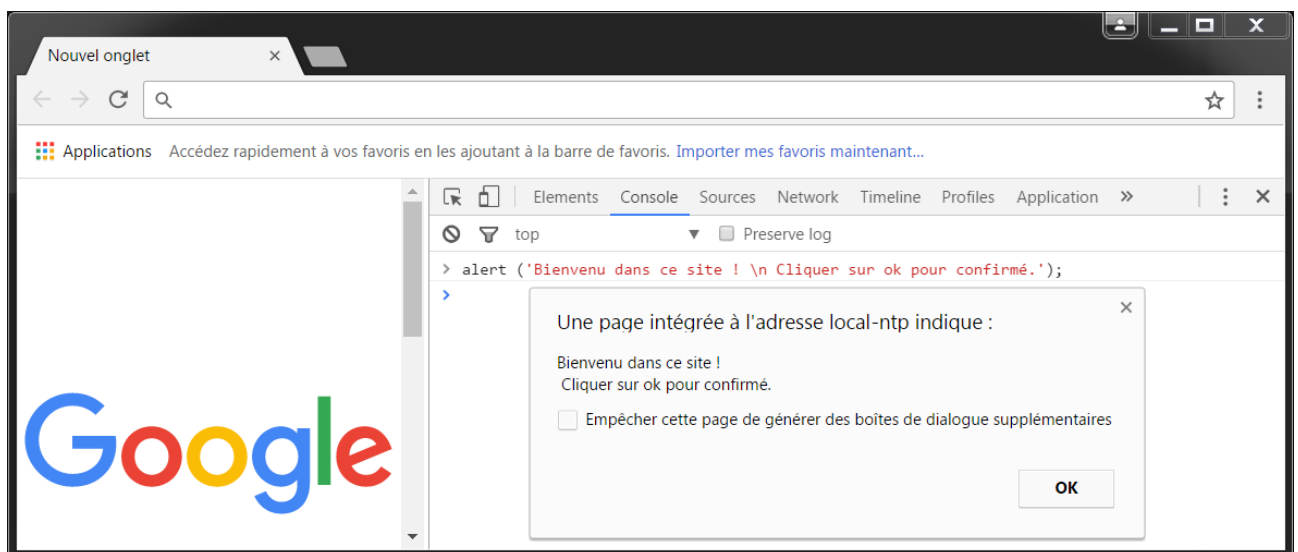


FIGURE 3.30 – Boite d'alerte avec Google Chrome.

Exemple : Navigateur « Mozilla Firefox »

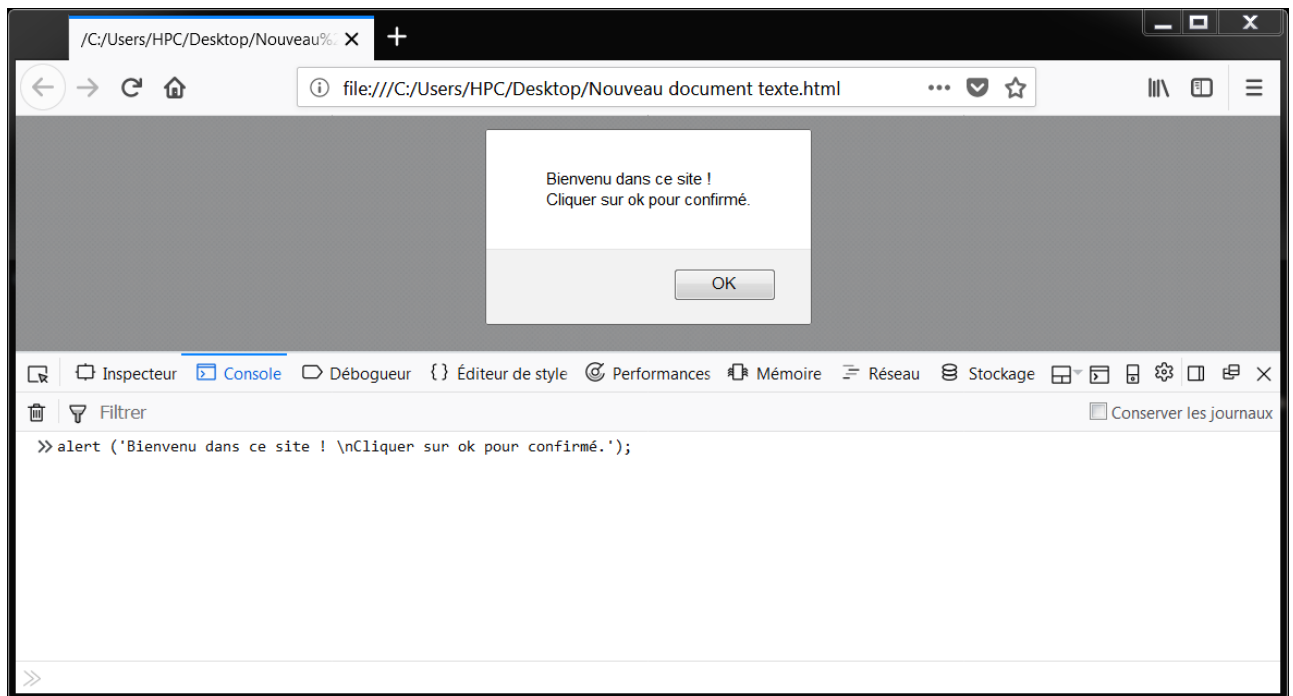


FIGURE 3.31 – Boîte d'alerte avec Mozilla Firefox.

Exemple : Navigateur « Internet Explorer »

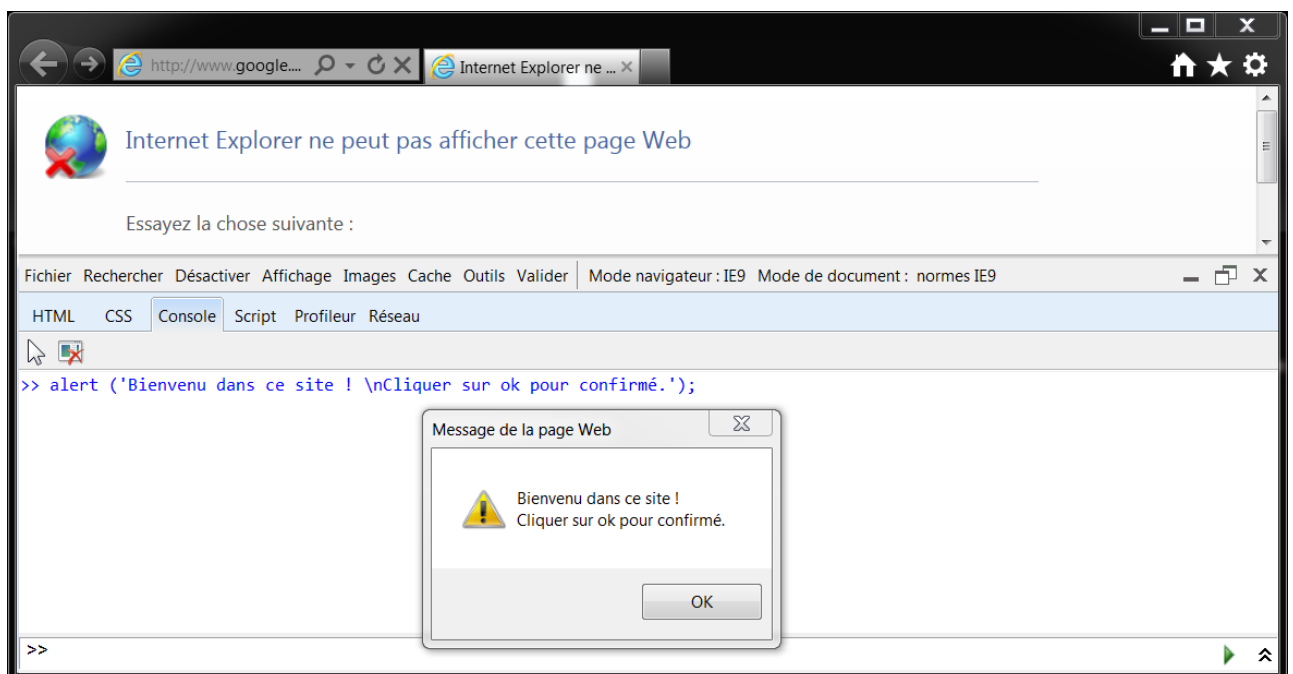


FIGURE 3.32 – Boîte d'alerte avec Internet Explorer.

3.6 Boite de confirmation

JavaScript met à disposition des développeurs une fonction nommée « `confirm` » qui permet d'afficher une boite de dialogue demandant une confirmation au visiteur. Par exemple : Voulez-vous supprimer tous les messages ?

Voici la syntaxe de la boite de confirmation :

```
confirm(" ") ;
```

3.6.1 Caractéristiques

La boite de confirmation est munie d'un bouton « OK » pour la confirmation lors du clique, et d'un bouton « Annuler » pour l'annulation lors du clique, et d'une croix de fermeture. Elle s'affiche différemment selon le navigateur utilisé.

La fonction `confirm` renvoie deux valeurs :

- « `true` » si le visiteur a cliqué sur « OK ».
- « `false` » si le visiteur a cliqué sur « Annuler » ou a cliqué sur la croix rouge de fermeture.

Exemple :Navigateur « Google Chrome »

```
confirm("Voulez-vous supprimer tous les messages ?") ;
```

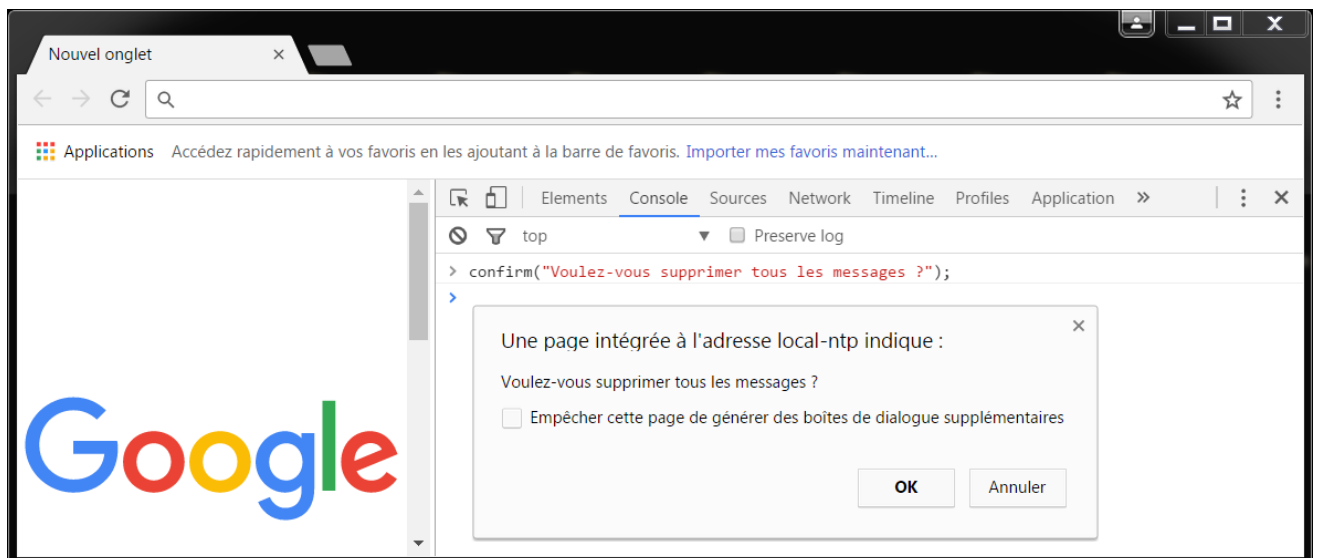


FIGURE 3.33 – Boite de confirmation.

- Un clique sur « OK » :

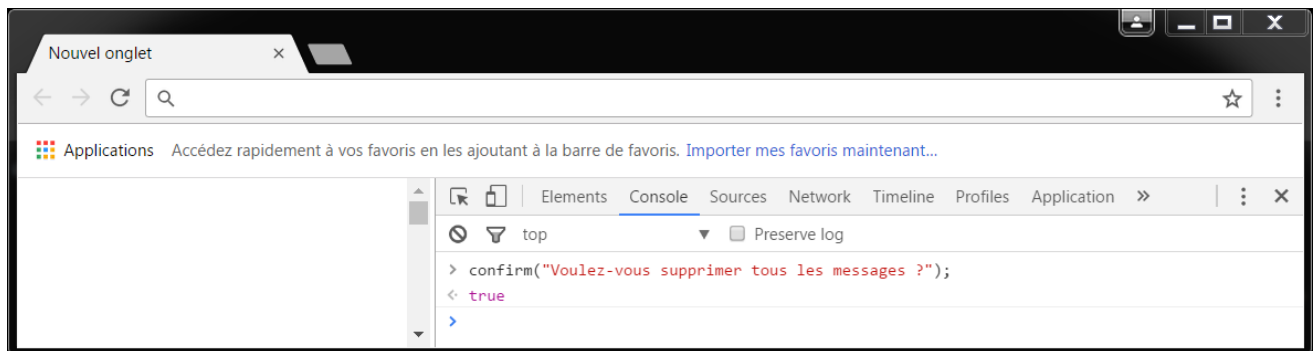


FIGURE 3.34 – La fonction confirm renvoie true.

— Un clique sur « Annuler » ou sur la croix de fermeture :

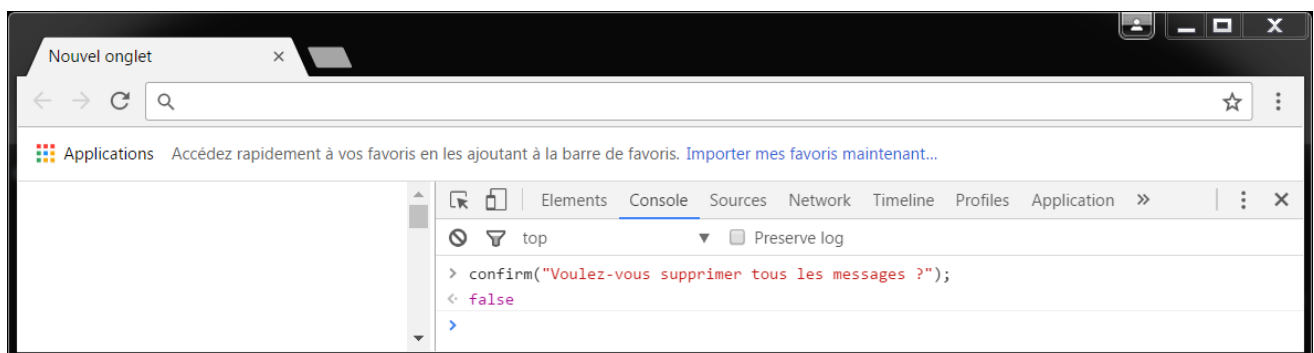


FIGURE 3.35 – La fonction confirm renvoie fals.

On peut combiner dans le code entre la boîte d’alerte et la boîte de confirmation pour afficher sur la boîte d’alerte la valeur que la fonction « confirm » renvoie.

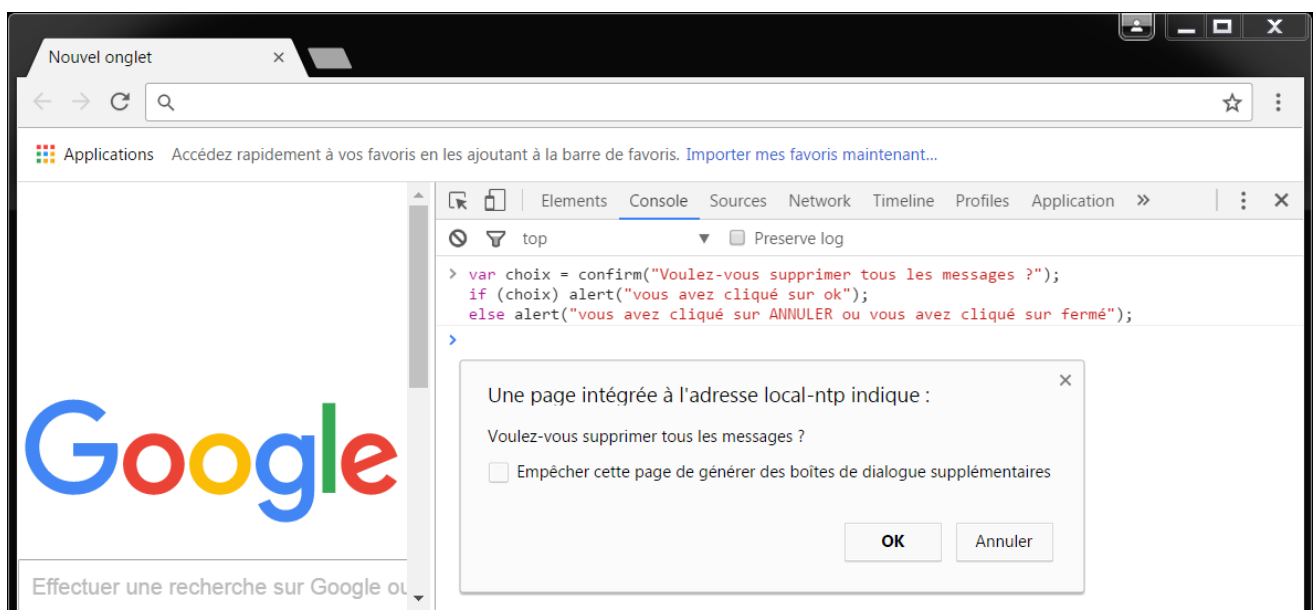


FIGURE 3.36 – Boîte de confirmation et alert combinée.

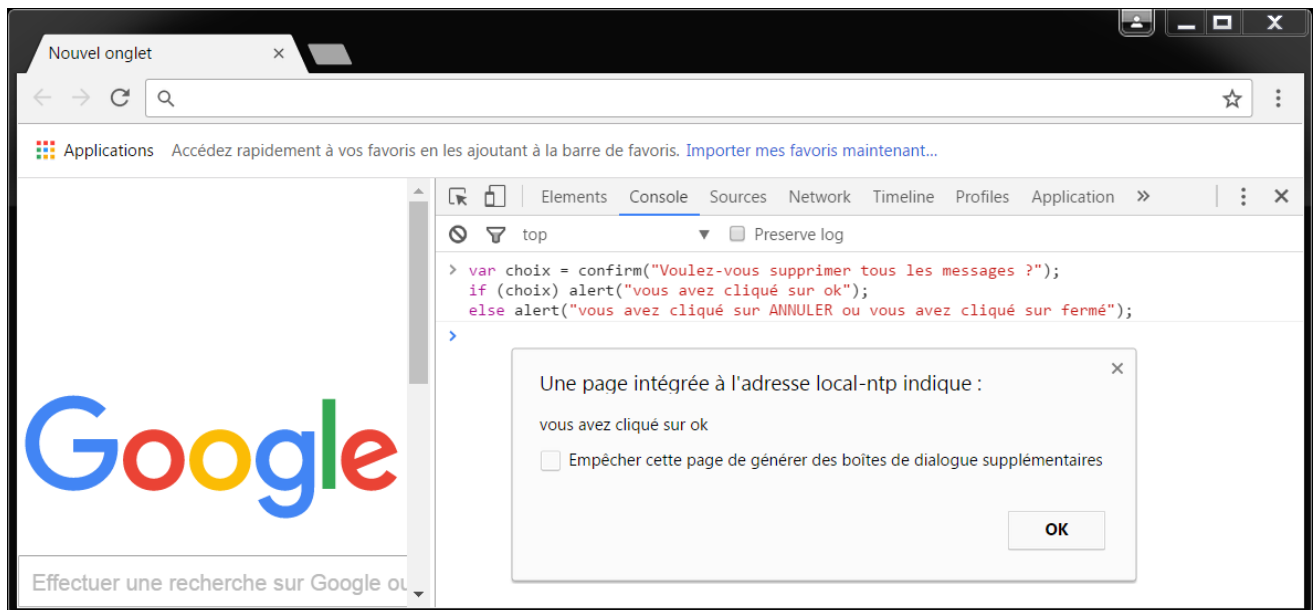


FIGURE 3.37 – Boîte de confirmation et alert combinée.

3.7 Boîte de saisie

La boîte de saisie permet de demander une saisie au visiteur.

3.7.1 Caractéristiques

La boîte de saisie est munie d'un bouton « OK » pour valider la saisie lors du clique, et d'un bouton « Annuler » pour l'annulation lors du clique, et d'une croix de fermeture. Elle possède aussi un champ de texte pour la saisie. Elle s'affiche différemment selon le navigateur utilisé.

Voici la syntaxe de la boîte de saisie :

```
prompt("texte de demande", valeur par défaut du textfield) ;
```

Exemple :

```
do  
{  
  choix = prompt("Veuillez entrer un nombre supérieur à cinq : ") ;  
}  
while (isNaN(choix) || !choix || Number(choix) < 5) ;  
document.write("Le nombre que vous avez entré est : "+choix) ;
```

Résultat :

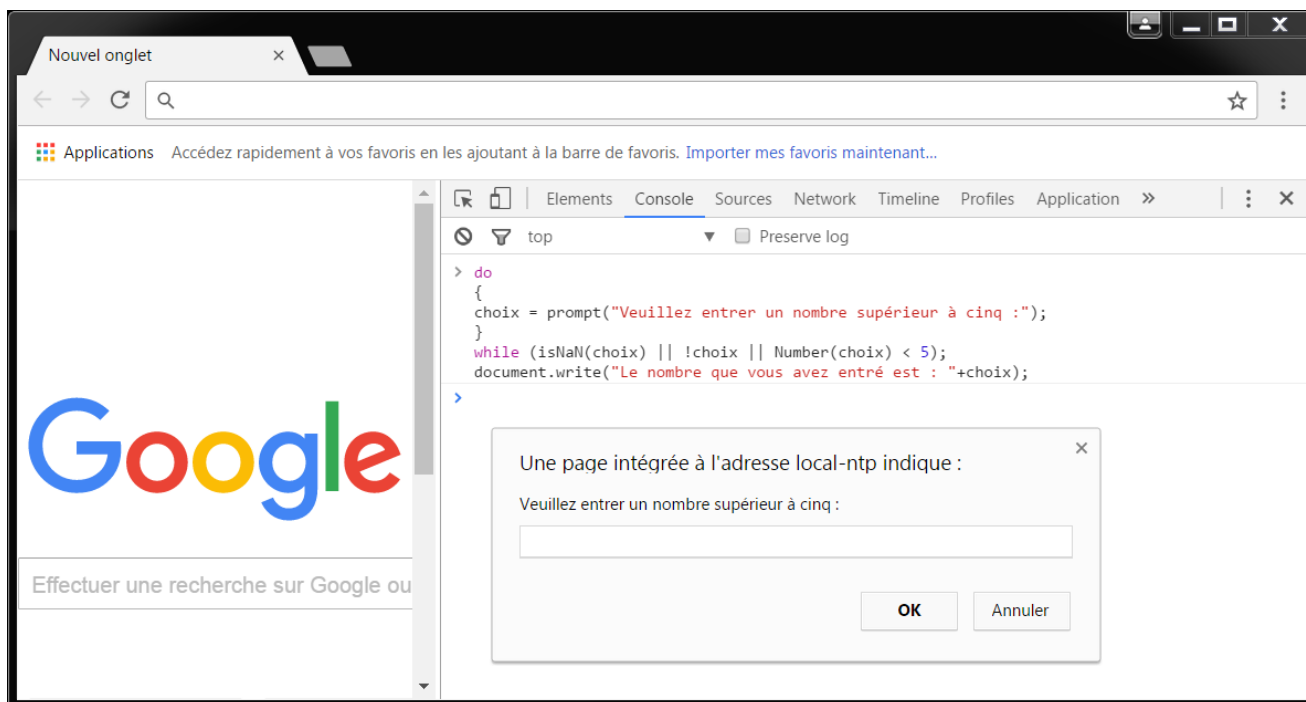


FIGURE 3.38 – Boîte de saisie.

En premier lieu, on utilise la boucle « do-while » pour être sûr que celle-ci sera parcourue au moins une fois. Avec la boucle « while », il faut d’abord initialiser le choix à « null » avant la boucle.

Dans la boucle, il est demandé d’entrer un nombre supérieur à cinq « 5 » via le prompt. Ensuite, la condition de « while » vérifie si la saisie est bien un nombre « `isNaN(choix)` ». NaN signifie en Anglais Not a Number, en Français (pas un nombre).

Puis, on vérifie que le visiteur n’a pas cliqué sur « Annuler » : « `!choix` ».

Enfin, on vérifie que le nombre est supérieur à 5 : « `Number(choix) < 5` », c’est-à-dire que tant que le nombre saisi est inférieur à cinq « 5 », la boîte de saisie demande au visiteur d’entrer un nombre supérieur à cinq.

3.8 Fonctions

Les fonctions sont définies comme un ensemble d’instructions sous forme de blocs de code qui accepte des données pour faire des opérations bien définie et ensuite retourner un ou plusieurs résultats. Pour déclarer une fonction, il faut le mot clé « function » suivi du nom de la fonction sans caractères spéciaux (pas d’espace, de virgule . . .).

Le nom de la fonction devient en quelque sorte une nouvelle commande du langage de JavaScript qui pourra être utilisé à tout moment comme une commande classique.

Pour définir une fonction, la syntaxe est la suivante :

```
function Nom_Fonction (Arg1,..., ArgN) {  
  Instruction 1 ;  
  InstructionM ;  
  return valeur ;  
}
```

Entre les parenthèses se trouvent les arguments (ou paramètres) qui sont des noms de variable de la fonction qui doivent être séparé par des virgules. Les arguments de la fonction ne sont pas typés lors de la définition. Il peut s'agir aussi bien de chaînes de caractères, de nombres, de booléens que d'objets. D'autre part, le nombre d'arguments d'une fonction n'est pas fixé lors de sa déclaration. On peut par exemple créer une fonction qui est appelée sans argument, et renvoie le jour de la semaine ou qui est appelée avec le paramètre date au format (Jour/Mois/Année) et renvoie le jour de la semaine pour cette date. Entre les accolades se trouvent le bloc d'instructions de la fonction.

Exemple :

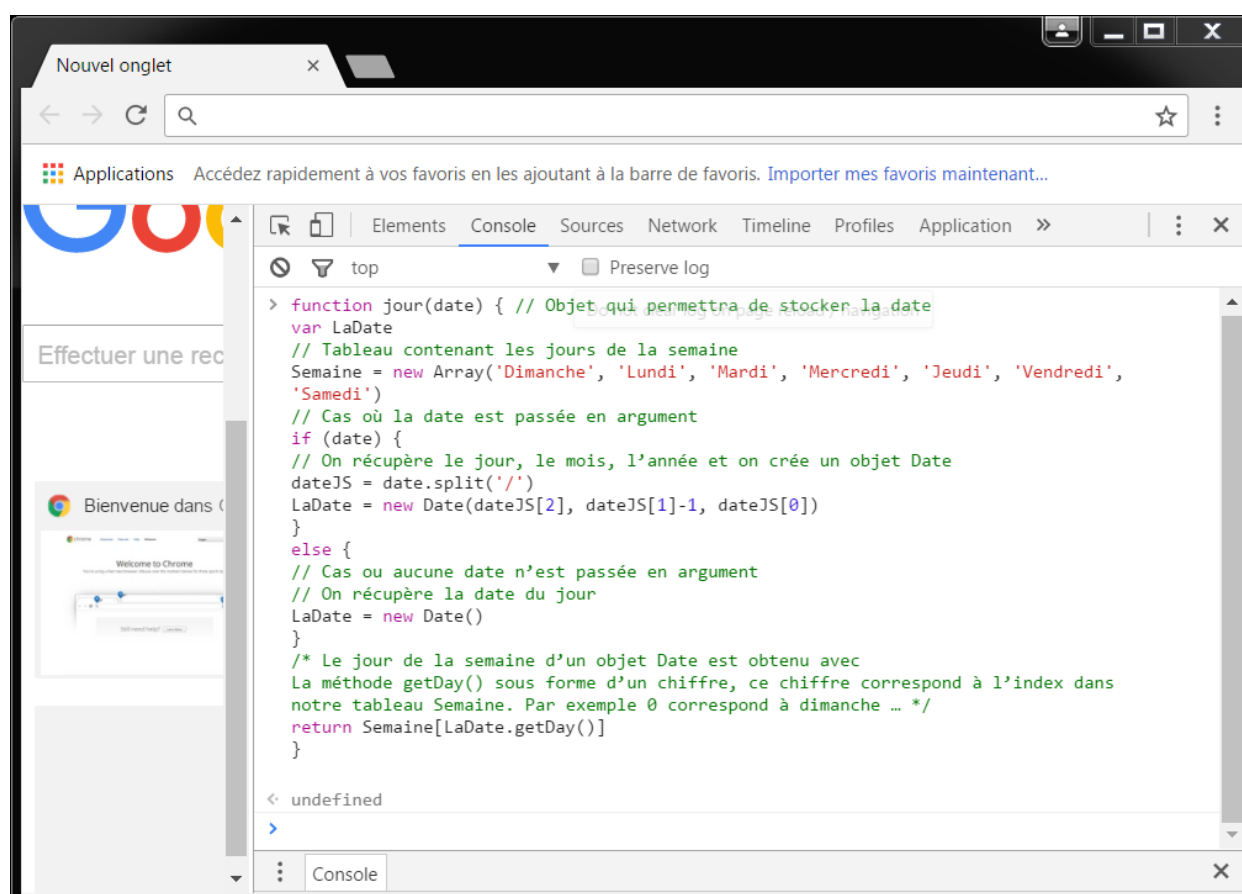


FIGURE 3.39 – Exemple d'une fonction.

Résultat :

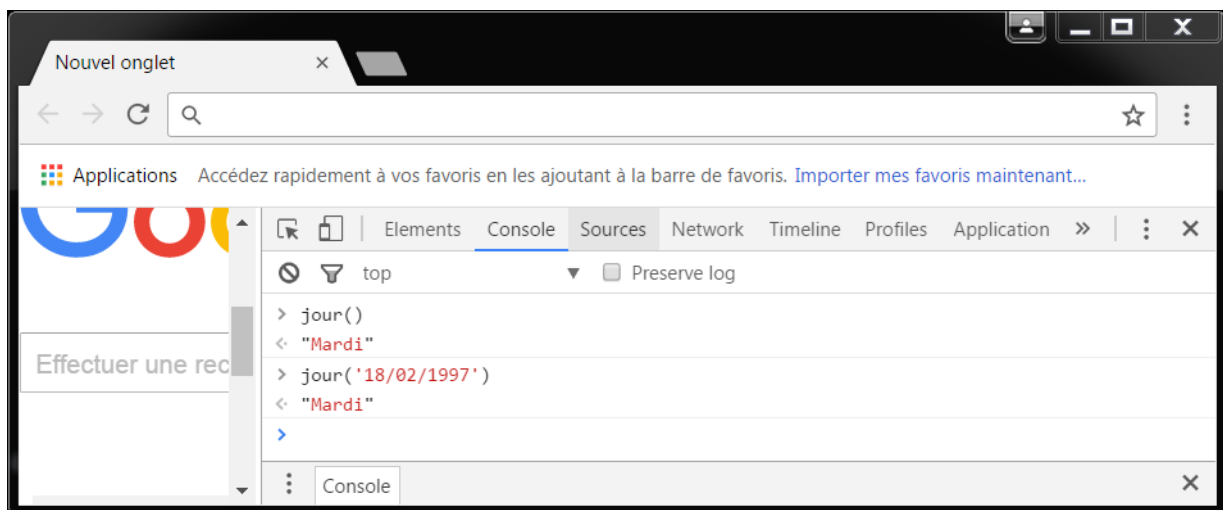


FIGURE 3.40 – Résultat de la fonction `jour()`.

Le mot clé "return" qui comme son nom l'indique veut dire (retourner) en anglais. Il est simplement suivi de la valeur à retourner. On peut créer une fonction qui ne retourne pas de valeur mais affiche une phrase. La fonction « `parseInt()` » permet de convertir une chaîne de caractère en un nombre entier.

Exemple :

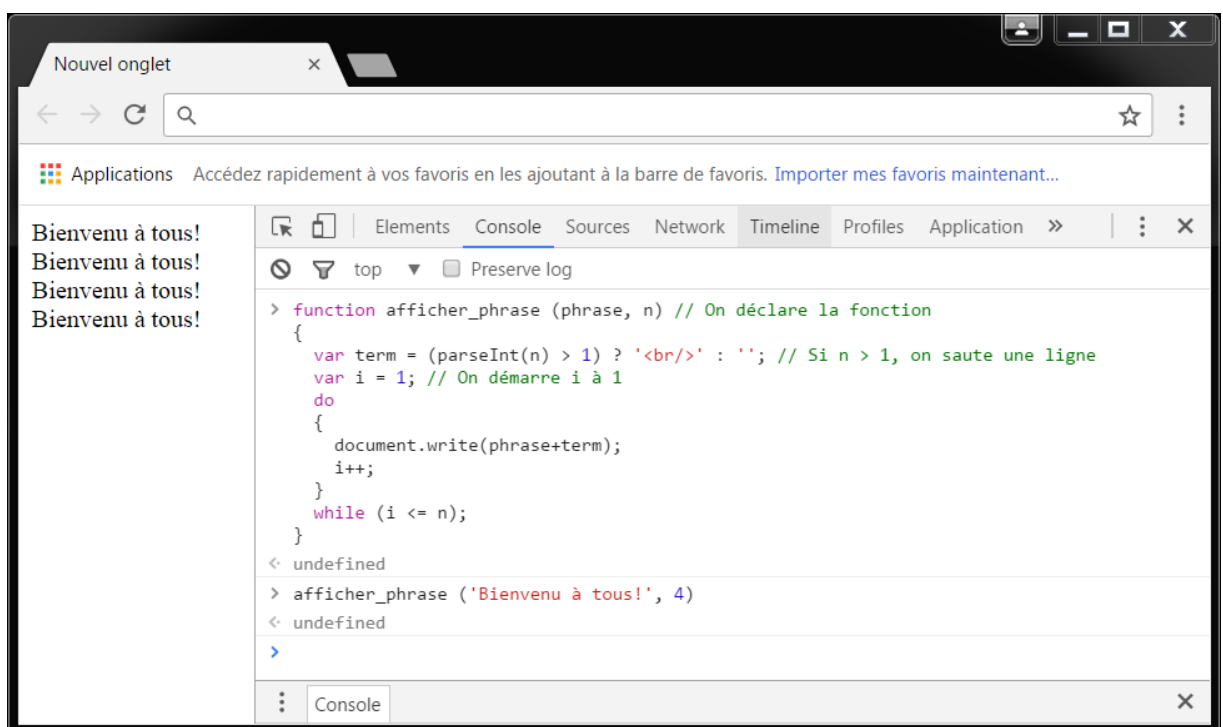


FIGURE 3.41 – Fonction qui affiche une phrase.

3.9 Exemples de fonction mathématiques

Exemple 1 : Calcul du n^{th} nombre de fibonnaci.

$$F_n = 1 \text{ si } n = 1 \text{ ou } n = 2$$

$$F_n = F_{n-1} + F_{n-2} \text{ si } n > 2$$

Le programme :

```
<html>
<head>
  <title>Calcul le nombre de fibonnaci</title>
  <script>
    function Fib(n){
      if (n<=2)
        return 1 ;
      else return Fib(n-1)+Fib(n-2) ;
    }
  </script>
</head>
<meta charset="UTF-8">
<body bgcolor="greenyellow">
<h1>Calculer le n <sup>th</sup> nombre de fibonnaci </h1>
<p>
<form> valeur de n ( un entier positif ) :
<input type="text" name="n" size="2" maxlength="3" value="1"
onblur="FibN.value=Fib(parseInt(n.value)) ; " />
(Cliquer sur tab pour avoir le n<sup>th</sup>nombre de fibonnaci) :
<input type="text" name="FibN" size="8" maxlength="8" value="1" />
</form>
</body>
</html>
```

Résultat :



FIGURE 3.42 – Function de fibonnaci.

Exemple 2 : Calcul de l'aire et le circonférence d'un cercle.

```
<html>
<head>
<title>Cercle</title>
<script name="javascript" type="text/javascript">
function moncercle(f){
var r=parseFloat(f.rayon.value) ;
f.aire.value=Math.PI*r*r;
f.circonference.value=2*Math.PI*r ;
}
</script>
</head>
<meta charset="UTF-8">
<body bgcolor="#99ccff">
<h1>Mon cercle</h1>
<form>
Entrer le rayon, puis appuyez sur tab ou cliquez sur la case"aire".<br/>
Rayon(cm) :
<input type="text" name="rayon" size="16" maxlength="7"
  onblur="moncercle(form) ; "/>
Aire(cms<sup>2</sup>) :
<input type="text" name="aire" size="16" maxlength="7"/>
Circonference(cm) :
<input type="text" name="circonference" size="16" maxlength="7"/>
</form>
</body>
</html>
```

Résultat :

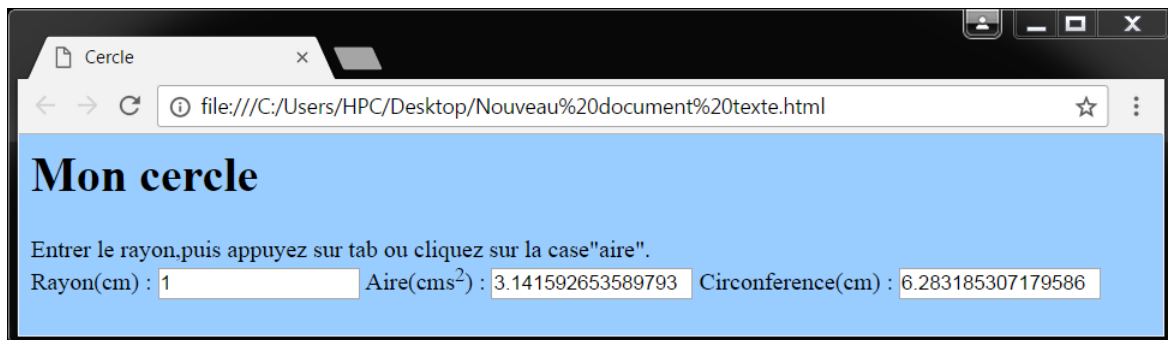


FIGURE 3.43 – Aire et circonférence d'un cercle.

Exemple 3 : La température du point de rosée est la température à laquelle la vapeur d'eau se condense de l'atmosphère. Il est lié à la température de l'air et l'humidité relative suivant ces équations :

$$a = 17.27$$

$$b = 237.7$$

$$\alpha = aT_a / (b + T_a) + \ln(RH)$$

$$T_{dp} = (b + \alpha) / (a - \alpha)$$

Le programme :

```
<html>
<head>
<meta charset="UTF-8">
<title>Calculateur de température du point de rosée</title>
<script language="javascript" type="text/javascript">
function getDewpoint(T,RH){
var a=17.28, b=237.7, alpha ;
var temp=parseFloat(T.value) ;
var rh=parseFloat(RH.value)/100 ;
alpha=a*temp/(b+temp)+Math.log(rh) ;
return Math.round(b+alpha/(a-alpha)*10)/10 ;
}
</script>
</head>
<body bgcolor="yellow">
<h1>Température du point de rosée</h1>
<p>
<form>
```

```

<input type="reset" value="Réinitialiser tout les champs"/><br/>
Température :
<input type="text" name="T" size="5" maxlength="6" value="17"/>&#176;c<br/>
Humidité relative :
<input type="text" name="RH" size="6" maxlength="6" value="60"/>%<br/>
<input type="button"
value="Cliquez ici pour obtenir la température du point de rosée."
onclick="DP.value=getDewpoint(T,RH)"/><br/>
La température du point de rosée :
<input type="text" name="DP" size="5" maxlength="6"/>&#176;c<br/>
</p>
</form>
</body>
</html>

```

Résultat :



FIGURE 3.44 – Calculateur de température de point de rosée.

3.10 Application JavaScript

Avant de conclure ce troisième chapitre nous avons réalisé une application web qui permet de tracer des fonctions mathématiques.

Cette application est intégrée dans une seule page web, on y trouve une liste de fonctions mathématiques qui sont déjà disponibles sur la page grâce à la balise « `<select>` » et que nous avons intégrées dans le code JavaScript grâce à la bibliothèque « `Math` ».

Ensuite on y trouve un champ de texte pour la saisie c'est-à-dire que le visiteur peut saisir à volonté des fonctions mathématiques à condition que ce soit en fonction de la variable « x » car on a programmé le JavaScript de manière à ce que « x » soit reconnu comme variable grâce à une fonction prédéfinie « Eval ».

Enfin la balise « <canvas> » qui est un élément de HTML trace les courbes des fonctions choisies et saisies par le visiteur dans une zone de dessin dont la hauteur et la largeur sont définies dans le code HTML comme attributs de la balise « <canvas> ».

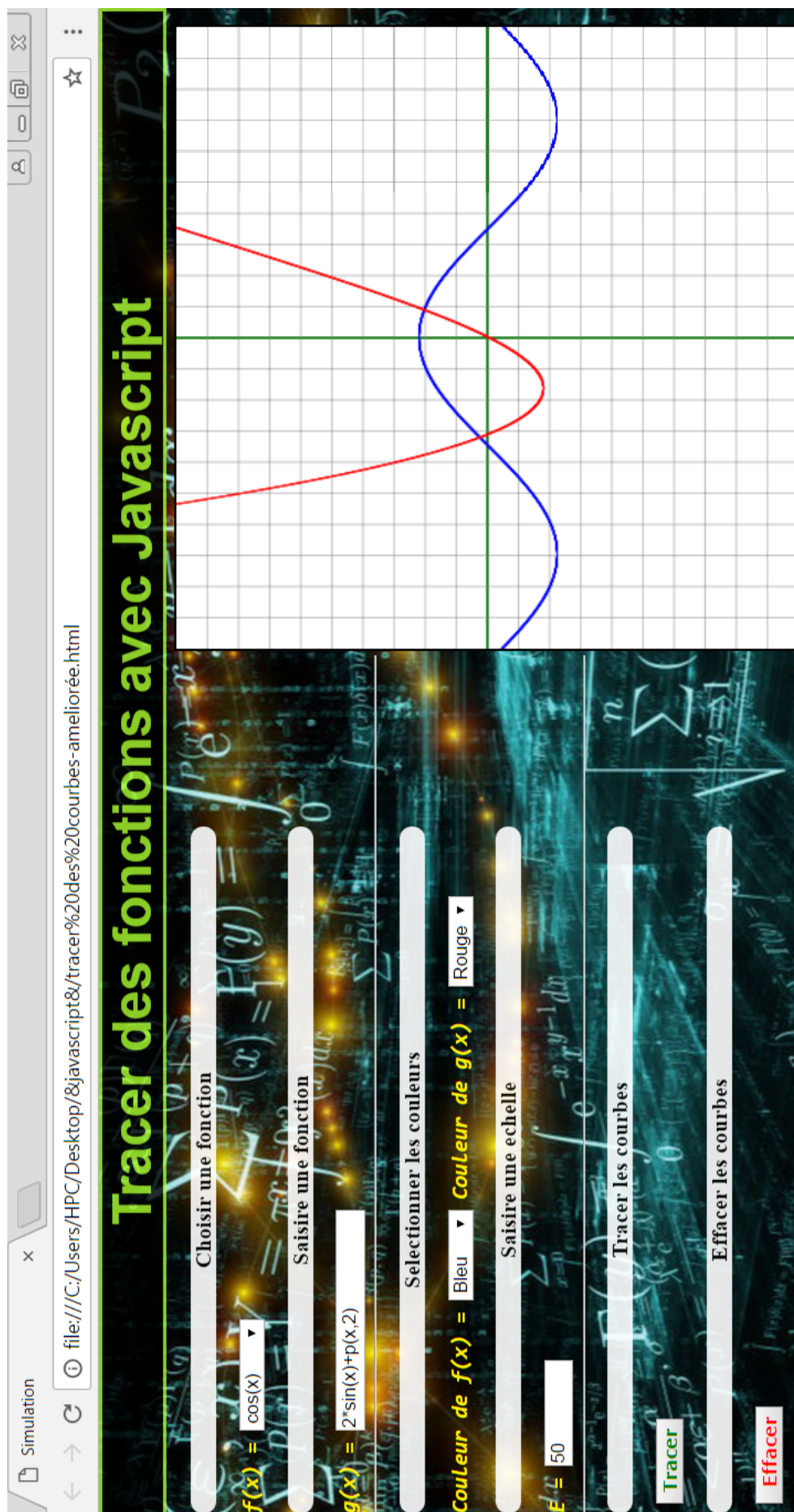


FIGURE 3.45 – Application JavaScript pour tracer les courbes des fonctions mathématiques.

3.11 Conclusion

Le JavaScript est un langage informatique utilisé sur les pages web. Ce langage à la particularité de s'activer sur le poste client, en d'autres mots c'est l'ordinateur qui va recevoir le code et qui devra l'exécuter. C'est en opposition à d'autres langages qui sont activé côté serveur.

L'exécution du code est effectué par le navigateur internet tel que Firefox ou Google Chrome. La particularité du JavaScript consiste à créer des petits scripts sur une page HTML dans le but d'ajouter une petite animation ou un effet particulier sur la page. Cela permet en général d'améliorer l'ergonomie ou l'interface utilisateur. L'intérêt du JavaScript est d'exécuté un code sans avoir à recharger une nouvelle fois la page.

Chapitre 4

Développement d'applications d'automatique

4.1 Introduction

Dans ce chapitre, nous allons développer un Laboratoire virtuel comprenant deux applications d'automatique. La première a pour objet de montrer les effets de différentes actions d'un régulateur PI. La deuxième concerne la simulation du comportement dynamique du système masse-ressort-amortisseur. Les deux applications ont été développées en HTML et JavaScript.

4.2 Types généraux de régulateurs

La méthode de régulation la plus employée dans l'industrie est de très loin le régulateur Proportionnel-intégral-dérivé ou PID. On assure couramment que 90 à 95% des problèmes de l'industrie peuvent être résolus par ce type de régulateur qui est facilement disponible sous forme d'un module électronique. Il allie une apparente simplicité de compréhension et une performance le plus souvent satisfaisante. Il est basé sur un principe quasiment naturel qui consiste à agir sur le procédé en fonction de l'erreur entre la consigne et la sortie mesurée [15].

Un régulateur peut prendre les formes les plus diverses. En fait, il représente une stratégie de commande, c'est-à-dire un ensemble de règles fournissant une valeur de l'action de commande lorsque la sortie dévie par rapport à la consigne. Un régulateur peut donc être constitué par une équation ou un algorithme.

4.2.1 Régulateur proportionnel

La sortie agissante du régulateur proportionnel est proportionnelle à l'erreur en entrée :

$$r(t) = K_r e(t) + r_b \quad (4.1)$$

où K_r est le gain proportionnel du régulateur et r_b est le signal de décentrement de l'actionneur (=signal agissant lorsque $e(t) = 0$) ajusté pour que la sortie coïncide avec la sortie désirée à l'état stationnaire. Le régulateur proportionnel est caractérisé par le gain proportionnel K_r et parfois par la bande proportionnelle BP définie par :

$$BP = \frac{100}{K_r} \quad (4.2)$$

Dans le cas d'un gain adimensionnel. En général, la bande proportionnelle est comprise entre 1 et 500 : elle représente le domaine de variation de l'erreur pour que le signal agissant couvre tout son domaine. Plus le gain est élevé, plus la bande proportionnelle est faible, et plus le régulateur est sensible. Un régulateur peut saturer, lorsque sa sortie $r(t)$ atteint une valeur maximale r_{max} ou minimale r_{min} . La fonction de transfert du Régulateur est simplement égale au gain du régulateur :

$$G_r(s) = K_r \quad (4.3)$$

Le régulateur proportionnel présente l'inconvénient de toujours provoqué un écart de la sortie avec la consigne.

4.2.2 Régulateur proportionnel-intégral

La sortie agissante du régulateur proportionnel-intégral est liée à l'erreur en entrée par :

$$r(t) = K_r(e(t) + \frac{1}{T_I} \int_0^t e(x)dx) + r_b \quad (4.4)$$

Pour les procédés chimiques, la constante de temps intégral T_I est en général de l'ordre de : $0.1 \leq T_I \leq 60$ min.

L'action intégral a pour effet de modifier la sortie du régulateur $r(t)$ tant qu'une erreur existe dans la sortie du procédé : un tel régulateur ne peut modifier que de petites erreurs. La fonction de transfert du régulateur proportionnel-intégral est égale à :

$$G_r(s) = K_r(1 + \frac{1}{T_I S}) \quad (4.5)$$

Par rapport au régulateur proportionnel, le régulateur proportionnel-intégral présente l'avantage d'éliminer l'écart entre la sortie et la consigne grâce à l'action intégral. Ce régulateur peut produire des réponses oscillatoires et diminue la stabilité du système en boucle fermée. D'autre part, l'action intégrale peut devenir indésirable lorsqu'il y a emballement : le régulateur agit à son niveau maximal et pourtant l'erreur persiste :

dans ce cas, le terme intégral augmente sans arrêt et il est nécessaire de stopper l'action intégrale [15].

4.2.3 Régulateur proportionnel-dérivé idéal

La sortie agissante du régulateur proportionnel-dérivé idéal est liée à l'erreur en entrée par :

$$r(t) = K_r(e(t) + T_D \frac{de(t)}{dt}) + r_b \quad (4.6)$$

L'action dérivée a pour but d'anticiper sur les erreurs futures. La fonction de transfert du régulateur proportionnel-dérivé idéal est égale à :

$$G_r(s) = K_r(1 + T_D s) \quad (4.7)$$

Ce régulateur est théorique car le degré du numérateur de la fonction $G_r(s)$ est supérieure au degré du dénominateur : il est donc irréalisable physiquement. Dans la pratique, l'action dérivée précédente est remplacée par le rapport de deux polynômes de premier ordre présentant des caractéristiques voisines pour les fréquences basses et moyennes. De plus, on lui rajoute toujours une action intégrale. L'action dérivée a une influence stabilisatrice sur le procédé commandé et prend en compte la prédiction [15].

4.3 Régulateur proportionnel-intégral-dérivé idéal

Ce type de régulateur est le plus souvent employé toutefois sous une forme légèrement différente de la forme idéale. La sortie agissante du régulateur proportionnel-intégral-dérivé idéal est liée à l'erreur en entrée par :

$$r(t) = K_r(e(t) + T_D \frac{de(t)}{dt} + \frac{1}{T_I} \int_0^t e(x) dx) + r_b \quad (4.8)$$

La fonction de transfert du régulateur proportionnel-intégral-dérivé est égale à :

$$G_r(s) = K_r(1 + T_D s + \frac{1}{T_I s}) \quad (4.9)$$

La remarque précédente sur la non-réalisabilité physique de l'action dérivée est à nouveau valable.

Grâce à l'action dérivée, le PID prend en compte (anticipe sur) le futur, et grâce à l'action intégrale, le PID prend en compte (intègre) le passé.

4.4 Rôle du régulateur

Le rôle du régulateur pour le système en boucle fermée est d'assurer que la réponse présente des caractéristiques dynamiques et stationnaires convenables.

On peut citer les critères suivants [15] :

- Le régulateur doit être capable de maintenir la variable commandée à sa consigne.
- Le système en boucle fermée doit être stable asymptotiquement et présenter une performance satisfaisante dans une large gamme de fréquences.
- L'influence des perturbations doit être minimale.
- Les réponses à des variations de consigne doivent être rapides et douces.
- Une action de commande excessive doit être évitée (la variable de commande u ne doit pas être trop sollicitée).
- Le système de commande doit être robuste : il doit être insensible aux variations du procédé et aux erreurs du modèle du procédé.

Dans les applications développées, nous allons procéder à l'intégration des équations différentielles ordinaires, en utilisant la méthode de Runge Kutta. Dans cette section, on fait un rappel sur cette méthode numérique.

4.5 L'algorithme de Runge-Kutta

De nombreux problèmes scientifiques peuvent être formulés en termes de système d'équations différentielles (ODE) :

$$y'(x) = f(x, y) \quad (4.10)$$

Avec une condition initiale :

$$y(x_0) = y_0$$

Les méthodes de Runge-Kutta sont des méthodes en une seule étape pour l'intégration numérique d'ODE (4.10).

La solution y est avancée du point x_0 à $x_1 = x_0 + h$ en utilisant une formule en une étape :

$$y_1 = y_0 + hk$$

Où y_1 est l'approximation de $y(x_1)$, et k est une constante astucieusement choisie (vecteur). Les méthodes de Runge-Kutta se distinguent par leur ordre : une méthode est

d'ordre p s'il peut intégrer exactement une ODE où la solution est un polynôme d'ordre p . En d'autres termes, si l'erreur de la méthode est $O(h^{p+1})$ pour le petit h . La méthode de Runge-Kutta du premier ordre est la méthode d'Euler :

$$k = f(x_0, y_0)$$

Les méthodes Runge-Kutta de second ordre font avancer la solution par une évaluation auxiliaire du dérivé, par exemple. Par méthode du point médian :

$$k_0 = f(x_0, y_0),$$

$$k_{\frac{1}{2}} = f(x_0 + \frac{1}{2}h, y_0 + \frac{1}{2}hk_0),$$

$$k = k_{\frac{1}{2}}. \quad (4.11)$$

Ou la méthode à deux points :

$$k_0 = f(x_0, y_0),$$

$$k_1 = f(x_0 + h, y_0 + hk_0),$$

$$k = \frac{1}{2}(k_0 + k_1). \quad (4.12)$$

Ces deux méthodes peuvent être combinées en une méthode de troisième ordre :

$$k = \frac{1}{6}k_0 + \frac{4}{6}k_{\frac{1}{2}} + \frac{1}{6}k_1. \quad (4.13)$$

La plus courante est la méthode du quatrième ordre appelée RK4 ou simplement la méthode Runge-Kutta d'ordre 4. Son algorithme est comme suit :

$$k_0 = f(x_0, y_0),$$

$$k_1 = f(x_0 + \frac{1}{2}h, y_0 + \frac{1}{2}hk_0),$$

$$k_2 = f(x_0 + \frac{1}{2}h, y_0 + \frac{1}{2}hk_1),$$

$$k_3 = f(x_0 + h, y_0 + hk_2),$$

$$k = \frac{1}{6}(k_0 + 2k_1 + 2k_2 + k_3). \quad (4.14)$$

L'implémentation en JavaScript est :

```
function rkstep(f,x,y,h){//Runge-Kutta étape intermédiaire
    avec erreur estimation
    var k0 = f(x,y)//dérivés à x0
    var y12 = [y[i] + k0[i]*h/2 ]//demi-étape
    var k12 = f(x+h/2 ,y12)//dérivés à demi-étape
    var y1 = [y [i] + k12[i]*h ]//étape complète
    var deltax =[(k12[i]-k0[i])*h/2 ]// estimation d'erreur
    return [y1,deltax]
}
```

4.6 Commande d'un système de premier ordre avec un PI

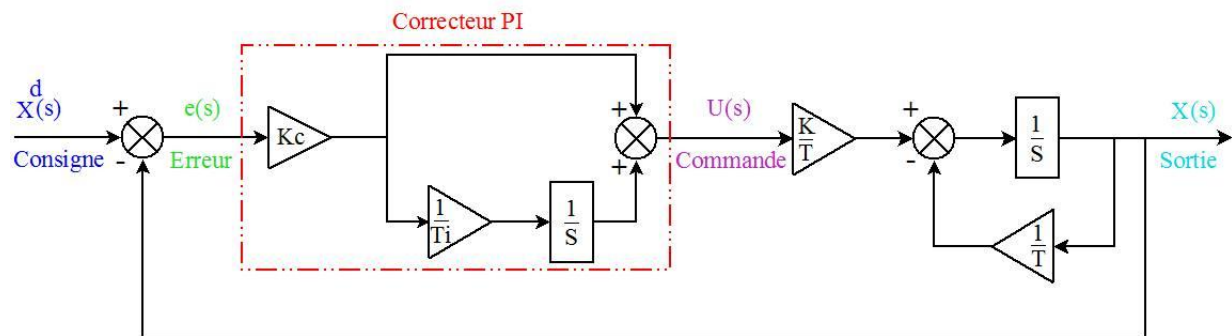


FIGURE 4.1 – Commande d'un système de premier ordre avec un PI.

On considère un système de premier ordre commandé par un PI. Grâce à la méthode de Runge-Kutta nous allons programmer ce system en JavaScript.

Les équations à programmer sont :

- Equation du modèle.
- Equation du régulateur.

Pour le système, on a :

$$\frac{x}{u} = \frac{k}{Ts + 1}$$

Alors :

$$T\dot{x} + x = ku$$

Donc en boucle ouverte :

$$\dot{x} = \frac{k}{T}u - \frac{1}{T}x$$

Ensuite, pour le régulateur on a :

$$\dot{x}_c = \frac{k_c}{Ti}(x_d - x) \quad (4.15)$$

$$u = x_c + k_c e$$

$$u = x_c + k_c(x_d - x)$$

Enfin les équations d'état à programmer sont :

$$\dot{x} = -\frac{1}{T}x + \frac{k}{T}[x_c + k_c(x_d - x)] \quad (4.16)$$

$$\dot{x}_c = \frac{k_c}{Ti}(x_d - x) \quad (4.17)$$

Résultats :

En boucle ouverte :

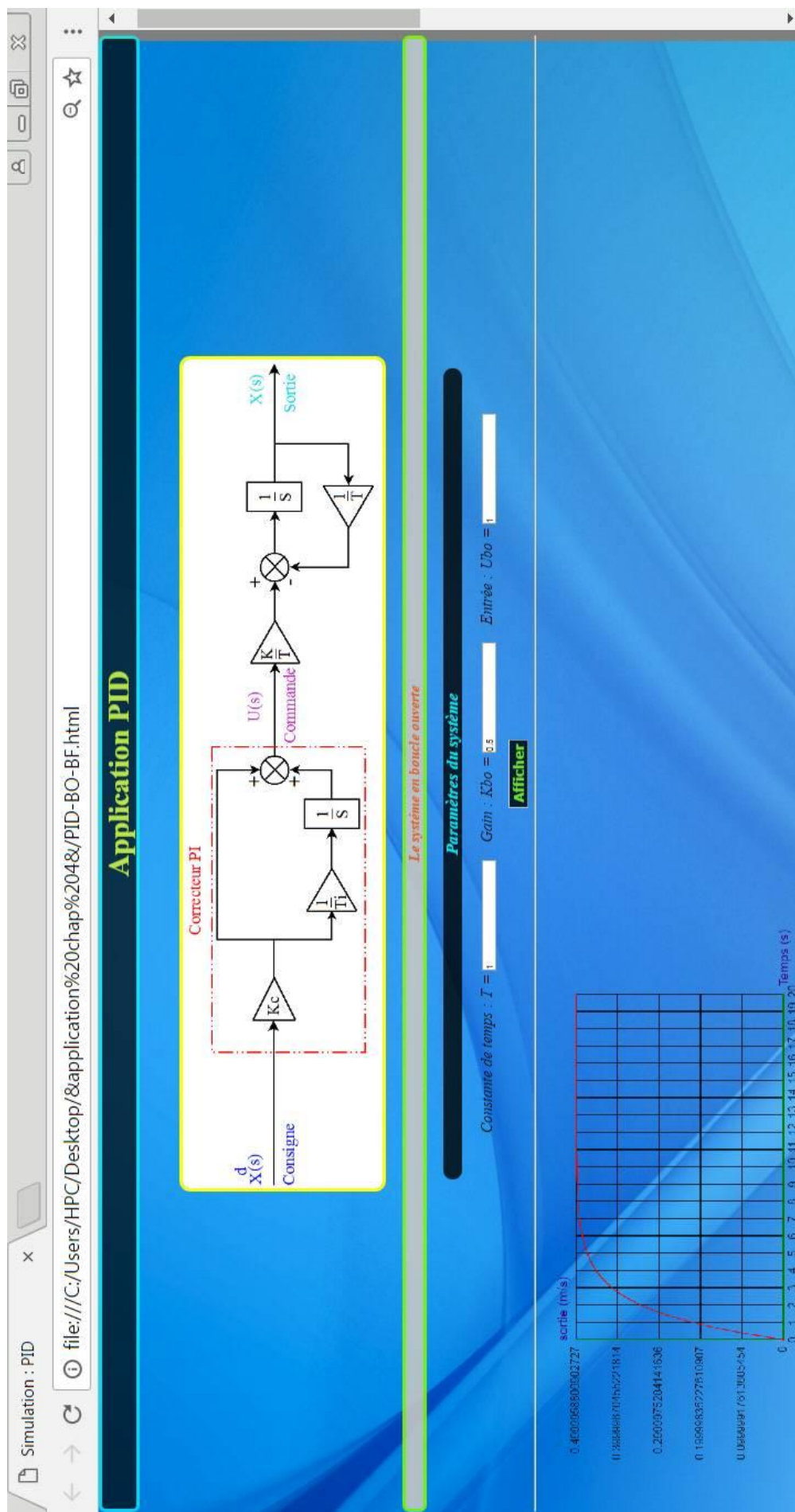


FIGURE 4.2 – Simulation d'un système de premier ordre en boucle ouverte avec un PI.

En boucle fermée :

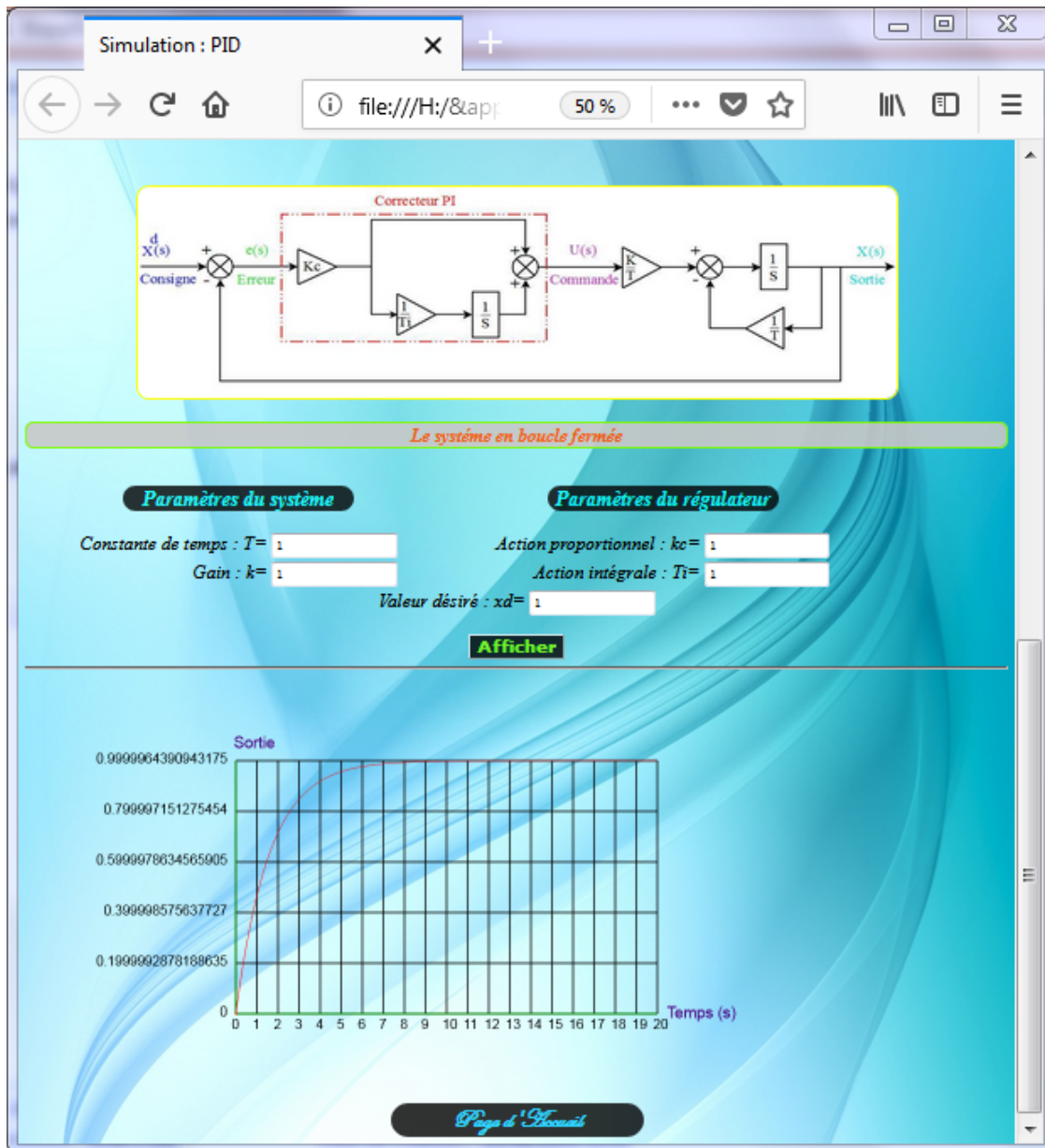


FIGURE 4.3 – Simulation d'un système de premier ordre en boucle fermée avec un PI.

On a programmé les équations différentielles de notre système en boucle ouverte et fermée grâce à la méthode de Runge Kutta implémenter sous forme d'une fonction javascript. Pour afficher le résultat de l'évolution de notre système on a utilisé la balise `<canvas>` associé à une bibliothèque graphique externe javascript qui permet l'affichage de données numériques sous une forme graphique et dynamiques.

4.7 Simulation d'un système masse-ressort-amortisseur

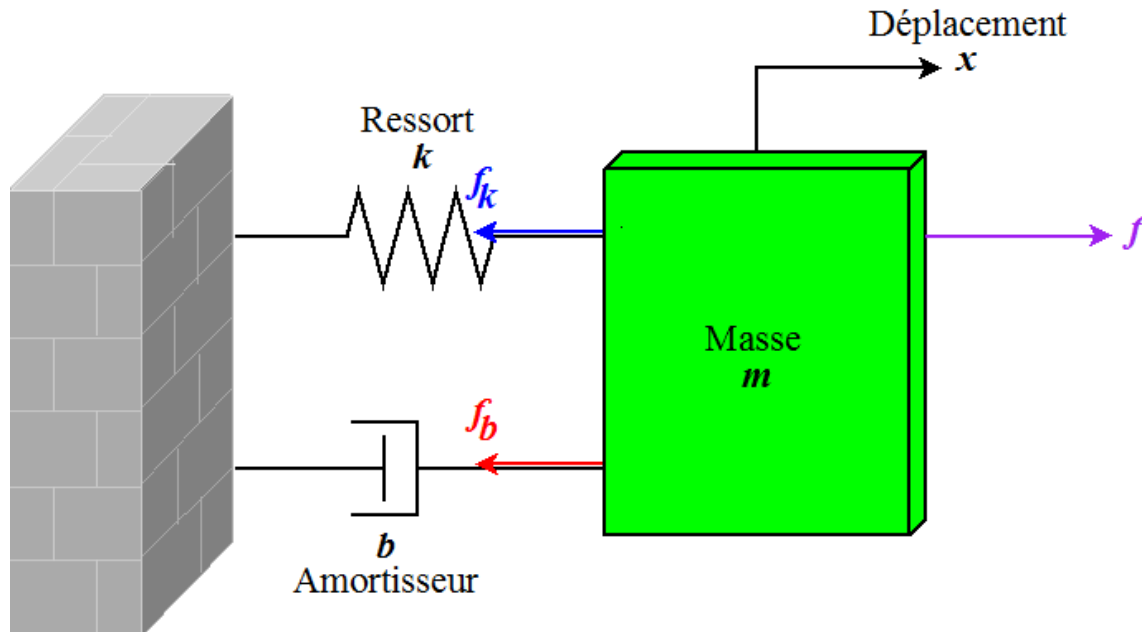


FIGURE 4.4 – Système masse-ressort-amortisseur.

Nous allons programmer la méthode Runge-Kutta pour la simulation d'un système masse-ressort-amortisseur en JavaScript.

L'équation différentielle régissant la dynamique du système est :

$$m\ddot{x} = f - kx - b\dot{x}$$

$$\ddot{x} = \frac{f}{m} - \frac{k}{m}x - \frac{b}{m}\dot{x} \quad (4.18)$$

En effectuant un changement de variables, on obtient les équations d'état suivante :

$$\dot{x}_1 = x_2 \quad (4.19)$$

$$\dot{x}_2 = \frac{f}{m} - \frac{k}{m}x_1 - \frac{b}{m}x_2 \quad (4.20)$$

Avec x_1 c'est la position et x_2 c'est la vitesse.

Résultat :

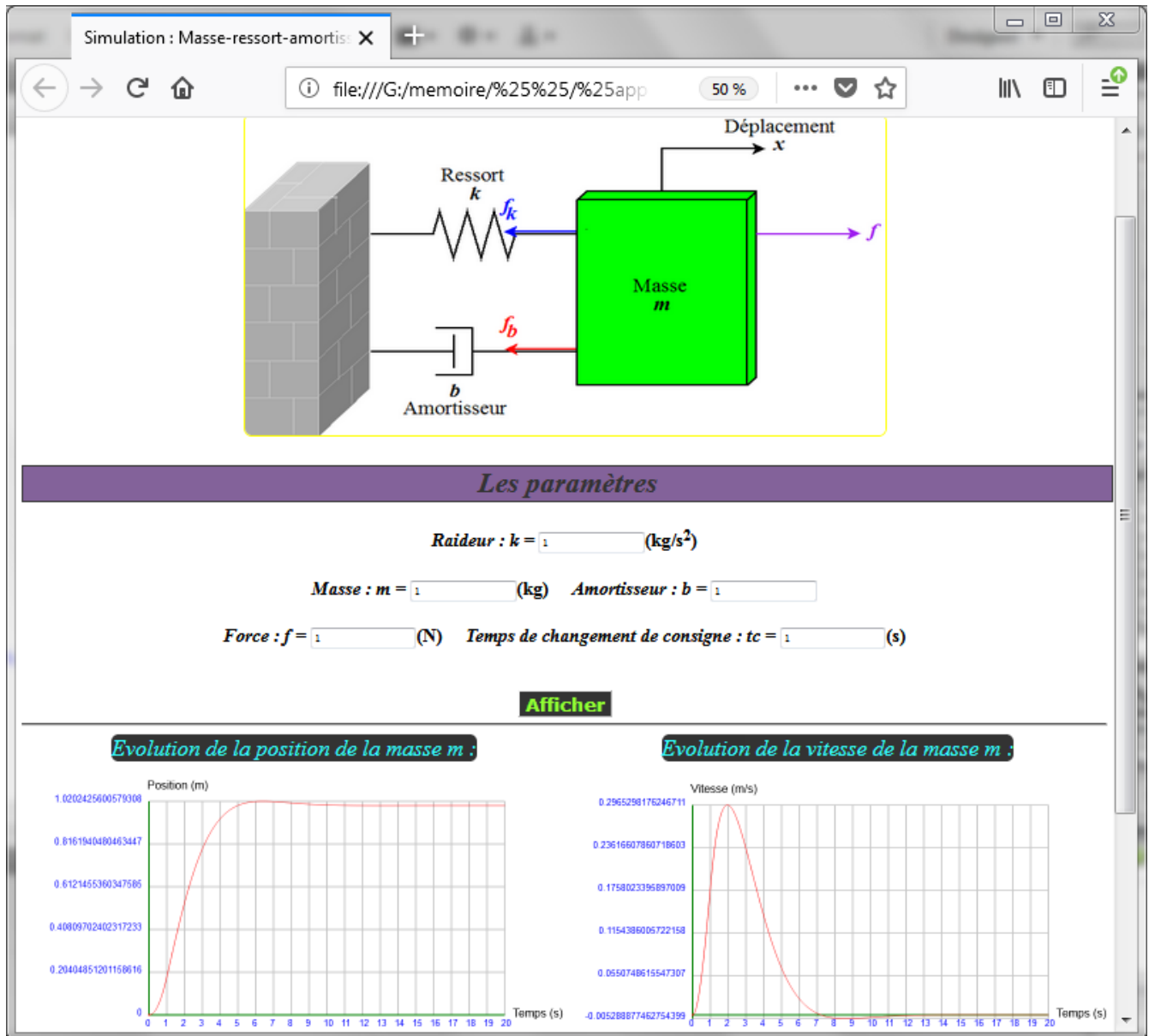


FIGURE 4.5 – Application masse-ressort amortisseur.

4.8 Conclusion

Dans ce chapitre, nous avons présenté un laboratoire virtuel composé de deux applications :

Correction PI et Simulation d'un système masse ressort-amortisseur.

Le problème de développement revient à programmer la méthode de Runge-Kutta par la simulation des équations différentielles ordinaires en JavaScript. Ce programme est intégré avec des pages web et utilisée par les deux applications développée. Des tests ont été effectués pour vérifier les deux applications réalisées.

Conclusion générale

Notre travail s'inscrit dans les recherches qui s'intéressent à l'enseignement à distance où nous avons essayé d'apporter une solution qui est l'élaboration d'une plateforme de type laboratoire virtuel. Dans une situation où le manque de moyens (capacité d'accueil limitée, matériels pédagogique inaccessible), le manque de temps et l'augmentation du nombre des étudiants, les enseignants et les étudiants ont besoin d'un environnement qui leur permet de faire leurs travaux pratiques.

Le premier chapitre du mémoire a traité la simulation et les laboratoires virtuels, leur mise en œuvre et leur déploiement, leurs interactivités, les avantages qu'ils apportent par leur utilisation et les objectifs qu'ils permettent d'atteindre.

La second chapitre a été consacrée à l'étude du langage de balisage HTML qui permet de structurer et de présenter un document pour le web. Son utilisation intense par les sites internet et la conception de son esthétique par les feuilles de style CSS. Nous avons donc créé un site complet en HTML et CSS qui pourra servir d'exemple par la suite pour les laboratoires pédagogique.

Dans le troisième chapitre, nous nous sommes focalisés sur le JavaScript, un langage de programmation qui se distingue par son universalité, historiquement associé au monde web qui permet de développer des applications et d'ajouter du dynamisme et de l'interactivité aux sites web. Nous avons étudié les aspects majeurs du JavaScript qui rendent son utilisation très intéressante et très puissante. Cela nous a permis de développer une application web dynamique qui permet de tracer des courbes de fonctions mathématiques.

En dernier lieu, les connaissances et le savoir acquis en HTML et JavaScript nous ont permis de développée deux applications web. L'une sur un PI et l'autre sur la simulation d'un système masse-ressort-amortisseur qui permettent une fois hébergé d'effectuer des tests et de manipuler à distance des paramètres de n'importe quel endroit.

Les coûts de mise en œuvre, d'opération et d'équipements d'un laboratoire virtuel sont réduits radicalement par l'élimination des risques matériels et réduction des coûts, donc

ils sont une meilleure alternative que les laboratoires traditionnels en termes d'expérimentations dangereuses ou coûteuses.

De ce fait les laboratoires virtuels sont accessibles au plus grand nombre, c'est-à-dire des étudiants qui sont dispersés géographiquement et qui sont soumis à différents fuseaux horaires et à ce titre constituent déjà une petite révolution.

La répétition des centaines de fois différentes expériences dans un laboratoire réelle engendre des dégradations sur le matérielle qui nécessite constamment des maintenances ou bien quasiment l'achat de nouveaux matériaux qui revient à des couts élevés. Donc un laboratoire virtuel sera un choix beaucoup plus intelligent et la valeur pédagogique de l'expérience n'en sera pas diminuée.

Les laboratoires virtuels offrent une meilleure flexibilité avec des coûts réduits, c'est-à-dire qu'on peut facilement améliorer une application, de lui apporter des modifications ou carrément la changé, ce qui est très difficile dans un laboratoire réelle ou on doit déplacer, acheter de nouveau matériel plus sophistiqué afin de répondre aux besoins de la théorie et de la recherche qui ne cesse de se développer.

Le travail présenté dans ce mémoire présente deux applications simple d'automatique sous forme d'un laboratoire virtuel. Ce laboratoire peut être amélioré par ajout de nouvelles applications.

Annexes

EjsS (Easy Java/Javascript Simulation)

Ejs a été créé par Francisco Esquembre et fait partie du projet open source physics. C'est un outil de programmation gratuit écrit en Java qui aide les non programmeurs à créer des simulations interactives en Java ou en JavaScript.

Applets Java

Une applet Java est fournie aux utilisateurs sous la forme de bytecode Java. Elle peut fonctionner dans un navigateur Web grâce à une machine virtuelle Java (JVM). Les applets sont utilisées pour fournir au sein d'application Web des fonctionnalités interactives qui ne peuvent pas être fournies par le langage HTML.

bytecode

Code binaire destiné à regrouper des instructions exécutables pour une machine virtuelle Java.

Lab-View (Laboratory-Virtual Instrument Engineering Workbench)

Le Lab-View est le cœur d'une plate-forme de conception de systèmes de mesure et de contrôle, fondée sur un environnement de développement graphique de National Instrument. Le langage graphique utilisé dans cette plate-forme est appelé G. Le Lab-View est utilisé principalement pour la mesure par acquisition de données pour le contrôle d'instruments et pour l'automatisme industriel.

WebGL

Le WebGL est une spécification d'interface de programmation de 3D dynamique pour les pages et applications HTML5. WebGL permet d'afficher, de créer et de gérer dynamiquement des éléments graphiques complexes en 3D dans la fenêtre du navigateur Web d'un client.

Robocode

Jeu vidéo de programmation éducatif libre crée et distribué gratuitement par IBM. Il est destiné à l'apprentissage du langage de programmation Java.

Node.js

Environnement bas niveau permettant l'exécution de JavaScript coté serveur. Il nous permet de programmer du JavaScript en dehors du navigateur.

Raspberry Pi

Est un nano ordinateur mono carte à processeur ARM conçu par le créateur de jeu vidéo David Braben, dans le cadre de sa fondation Raspberry Pi. Cet ordinateur qui a la taille d'une carte de credit est destiné à encourager l'apprentissage de la programmation informatique.

Gamification (Ludification)

Est l'utilisation des mécanismes du jeu dans d'autres domaines. Son but est d'augmenter l'acceptabilité et l'usage d'une application en s'appuyant sur la prédisposition humaine au jeu.

W3C (World Wide Web Consortium)

Est un organisme de standardisation à but non lucratif fonctionnant comme un consortium international, fondé en octobre 1994 chargé de promouvoir la compatibilité des technologies du World Wide Web telles que le HTML5, HTML, XHTML, XML, RDF, SPARQL, CSS, XSL, PNG, SVG et SOAP en développant des protocoles et des directives.

Blender

Est un logiciel libre et gratuit de modélisation, d'animation et de rendu en 3D, crée en 1995. Il est actuellement développé par la Fondation Blender. Il dispose de fonctions avancées de modélisation, de sculpture 3D.

Haptiques

L'haptique qui étymologiquement veut dire le toucher, permet une interaction sensorielle entre le réel et le virtuel, entre l'écran et le bout du doigt. Les technologies haptiques ont surtout été utilisées pour ressentir le contact physique lors de la manipulation d'objets virtuels en 3D.

XML

C'est un langage de balisage extensible, il est extensible car il permet de définir différents espaces de noms (recommandation du W3C qui permet d'employer des éléments et des attributs nommés dans une instance XML)

Bibliographie

- [1] N. A. Kheir, K. J. Aström, D. Auslander, K. C. Cheok, G. F. Franklin, M. Masten, & M. Rabins. Control systems engineering education. *Automatica*, Vol.32, No 2, pp 147–166.1996
- [2] J. Saenz, F. Esquembre, F. J. Garcia, L. de la Torre, & S. Dormido. *An architecture to use Easy Java-Javascript Simulations in new devices*. Ifac workshop on internet based control education, Brescia, Italie, 2015.
- [3] L. de la Torre. New generation virtual and remote laboratories : Integration into web environments 2.0 with learning management systems. National University of Distance Education (UNED), 2013.
- [4] J. Sanchez Moreno. A new methodological approach for distance learning of experimental subjects : analysis, design and development of a virtual and remote laboratory for the study of automation through the Internet. National University of Distance Education, 2001.
- [5] J. Bermudez-Ortega, E. Besada-Portas, J. Lopez-Orozco, J. Bonache-Seco, & J. de la Cruz. Remote web-based control laboratory for mobile devices based on ejss, raspberry pi and node.js. *In 3rd IFAC workshop on internet based control education, IFAC-PapersOnLine*, Vol. 48, pp.158–163, 2015.
- [6] V. Harward, J. del Alamo, S. Lerman, P. Bailey, J. Carpenter, K. DeLong, D. Zych. The ilab shared architecture : A web services infrastructure to build communities of internet accessible laboratories. *Proceedings of the IEEE*, vol. 96, No.6, pp.931–950, 2008.
- [7] L. Santos, & C. Carvalho. Improving experiential learning with haptic experimentation. *International Journal of Online Engineering*, vol. 9, No.8, 2013.
- [8] M. R. Quintas, M. T. Restivo & P. Ubaldo. Let's use haptics!. *International Journal of Online Engineering*, vol. 9, pp.65–67, 2013.
- [9] J. Chacon , M. Guinaldo , J. Sanchez & S. Dormido. A new generation of online laboratories for teaching automatic control. *In 3rd IFAC workshop on internet based control education, IFAC-PapersOnline*, Vol. 48, pp.140–145, 2015.

- [10] D. Galan, R. Heradio, L. de la Torre, S. Dormido & F. Esquembre. Automated experiments on EjsS laboratories. *In International conference on remote engineering and virtual instrumentation*, pp.78–85, 2016.
- [11] www.ee.ucl.ac.uk/~mflanaga/java/index.html (Cette page, et les liens qu'elle contient, décrivent les classes plus générales d'une bibliothèque scientifique et numérique Java écrite par Michael Thomas Flanagan pour soutenir à la fois ses propres recherches et ses cours et projets de programmation). Date de la dernière consultation : 7 juin 2018.
- [12] www.books.goalkicker.com (Livres de programmation gratuits sur le développement Android, C, C #, CSS, HTML5, développement iOS, Java, JavaScript, PowerShell, PHP, Python, SQL Server). Date de la dernière consultation : 20 mai 2018.
- [13] D. Charnay, P. Chaléat. *HTML et JavaScript*, Editions Eyrolles, France, 1998.
- [14] www.OpenClassrooms.com (OpenClassrooms est une école en ligne. Initialement orientée autour de la programmation informatique). Date de la dernière consultation : 30 avril 2018.
- [15] J.P .Corriou. *Commande des procédés 3^e édition*, Editions Lavoisier, Paris, 2012.
- [16] J.P. Vary. Rapport de la réunion d'experts sur les laboratoires virtuels organisée par l'Institut international de physique théorique et appliquée (IITAP), Ames, Iowa, 10-12 mai 1999.