

1 Rue de la Chebarde, Complexe des Cézeaux, BP 125 – 63178 Aubière

Rapport de projet de première année :

|  |
| --- |
| **Formulaire « de style » avec HTML(5)**  **et CSS(3)** |

Elèves :

**Nahel CHAZOT**

**Juliette NAUDIN**

Durée : 20 heures

Remerciements

Nous souhaitons remercier M. David HILL, notre encadrant, pour son soutient, ses conseils et son suivi.

Table des figures

[Figure II.1 : Mockup du formulaire 10](#_Toc518139325)

[Figure III.1 : Diagramme de Gantt du projet 11](#_Toc518139326)

[Figure III.2 : Fonctionnement d’un logiciel de gestion de version 12](#_Toc518139327)

[Figure IV.1 : Fichier d’extension .html 14](#_Toc518139328)

[Figure IV.2 : Syntaxe d’une balise de type paire 14](#_Toc518139329)

[Figure IV.3 : Insertion d’une image 16](#_Toc518139330)

[Figure IV.4 : Tableau Noms, Prénoms, Professions 17](#_Toc518139331)

[Figure V. 1 : syntaxe d’une feuille de style 19](#_Toc518139332)

[Figure V. 2 : code source HTML où sera appliqué style.css 19](#_Toc518139333)

[Figure V. 3 : rendu sans CSS 19](#_Toc518139334)

[Figure V. 4 : couleur rouge appliqué à tous les titres h1 20](#_Toc518139335)

[Figure V. 5 : rendu après application de la couleur rouge pour les titres h1 20](#_Toc518139336)

[Figure V. 6 : sélection d’une classe avec CSS 20](#_Toc518139337)

[Figure V. 7 : code source HTML où sera appliqué style.css 21](#_Toc518139338)

[Figure V. 8 : rendu avec CSS 21](#_Toc518139339)

[Figure V. 9 : Sélection d’une balise contenue dans une autre 22](#_Toc518139340)

[Figure V. 10 : Sélection d’un balise possédant un attribut où la valeur est ou n’est pas précisée 22](#_Toc518139341)

[Figure V. 11 : Bordures en CSS 23](#_Toc518139342)

[Figure V. 12 : Modification du style d’un élément lors d’un survol (CSS) 24](#_Toc518139343)

[Figure V.13 : Modification du style d’un élément lors d’un survol (Rendu) 24](#_Toc518139344)

[Figure V. 14 : « You are the CSS to my HTML » 25](#_Toc518139345)

[Figure VIII. 1 : code des fonctions JS pour masquer et afficher un élément 27](#_Toc518139346)

[Figure VIII. 2 : fonction JS de désactivation des messages de correction 28](#_Toc518139347)

[Figure VIII. 3 : code HTML relatif au champ Nom du formulaire 28](#_Toc518139348)

[Figure VIII. 4 : fonction JS de vérification du champ Nom du formulaire 29](#_Toc518139349)

[Figure VIII. 5 : fonction JS de …………… ?????????? 29](#_Toc518139350)

[Figure VIII. 6 : code HTML relatif au champ Mail du formulaire 29](#_Toc518139351)

[Figure VIII. 7 : fonction JS de vérification du champ Mail du formulaire 30](#_Toc518139352)

[Figure VIII. 8 : fonction JS de …………… ?????????? 30](#_Toc518139353)

[Figure VIII. 9 : code HTML relatif au champ Téléphone du formulaire 31](#_Toc518139354)

[Figure VIII. 10 : fonction JS de vérification du champ Téléphone du formulaire 31](#_Toc518139355)

[Figure VIII. 11 : fonction JS de …………… ?????????? 32](#_Toc518139356)

[Figure VIII. 12 : code HTML relatif au champ Message du formulaire 32](#_Toc518139357)

[Figure VIII. 13 : fonction JS de vérification du champ Message du formulaire 32](#_Toc518139358)

[Figure VIII. 14 : code HTML relatif au champ Catégorie du formulaire 33](#_Toc518139359)

[Figure VIII. 15 : fonction JS de vérification du champ Catégorie du formulaire 34](#_Toc518139360)

[Figure VIII. 16 : code HTML relatif à l’envoi du formulaire 34](#_Toc518139361)

[Figure VIII. 17 : fonction JS de vérification du formulaire avant envoi 34](#_Toc518139362)

[Figure VIII. 18 : rendu en cas d’erreur sur chaque champ, avant envoi 35](#_Toc518139363)

[Figure IX. 1 : attributs de <form> relatifs au PHP 36](#_Toc518139364)

[Figure IX. 2 : vérification de variables existantes et non nulles 37](#_Toc518139365)

[Figure IX. 3 : récupération des données dans des variables par php 38](#_Toc518139366)

[Figure IX. 4 : rendu si l’utilisateur ne passe pas par le formulaire (erreur) 39](#_Toc518139367)

[Figure IX. 5 : rendu lorsque l’utilisateur envoie des données correctes via le formulaire 39](#_Toc518139368)

Table des matières

[Remerciements 2](#_Toc518139369)

[Table des figures 3](#_Toc518139370)

[Table des matières 5](#_Toc518139371)

[I. Introduction 8](#_Toc518139372)

[I. 1) Préambule 8](#_Toc518139373)

[I. 2) Motivation 8](#_Toc518139374)

[II. Présentation du projet 9](#_Toc518139375)

[II. 1) Objectifs 9](#_Toc518139376)

[II. 2) Démarches 9](#_Toc518139377)

[II. 3) Environnement 10](#_Toc518139378)

[III. Méthodes de travail 11](#_Toc518139379)

[III. 1) Word et le suivi de modifications 11](#_Toc518139380)

[III. 2) Diagramme de Gantt 11](#_Toc518139381)

[III. 3) Git 12](#_Toc518139382)

[III. 3) a. Fonctionnement 12](#_Toc518139383)

[III. 3) b. Quelques avantages 12](#_Toc518139384)

[IV. Tutoriel HTML 14](#_Toc518139385)

[IV. 1) Démarrage 14](#_Toc518139386)

[IV. 2) Balises 14](#_Toc518139387)

[IV. 2) a. Type 14](#_Toc518139388)

[IV. 2) b. Syntaxe 14](#_Toc518139389)

[IV. 2) c. Liste des balises HTML ? 15](#_Toc518139390)

[IV. 3) Structure de base d’une page HTML 15](#_Toc518139391)

[IV. 3) a. La balise <html> 15](#_Toc518139392)

[IV. 3) b. L’en-tête <head> 15](#_Toc518139393)

[IV. 3) c. Corps de la page <body> 15](#_Toc518139394)

[IV. 4) Un site web sans CSS ? Sans… Nahel ? 17](#_Toc518139395)

[IV. 5) Sources du tutoriel 17](#_Toc518139396)

[V. Tutoriel CSS 18](#_Toc518139397)

[V. 1) Emplacement 18](#_Toc518139398)

[V. 2) Appliquer un style 18](#_Toc518139399)

[V. 2) a. Syntaxe 18](#_Toc518139400)

[V. 2) b. Sélecteurs de balises 19](#_Toc518139401)

[V. 2) c. Class et id 20](#_Toc518139402)

[V. 2) d. Sélecteurs spéciaux 21](#_Toc518139403)

[V. 3) Propriétés utiles aux formulaires 22](#_Toc518139404)

[V. 3) a. Le texte 22](#_Toc518139405)

[V. 3) b. Couleur et fond 22](#_Toc518139406)

[V. 3) c. Bordures 23](#_Toc518139407)

[V. 3) d. Les apparences dynamiques 23](#_Toc518139408)

[V. 4) Suite du travail 24](#_Toc518139409)

[V. 5) Sources du tutoriel 25](#_Toc518139410)

[VIII. Le JavaScript (JS) 26](#_Toc518139411)

[VIII. 1) Pourquoi avoir fait du JavaScript ? 26](#_Toc518139412)

[VII. 1) a. Avantages 26](#_Toc518139413)

[VIII. 1) b. … Mais le JavaScript n’est pas fiable 26](#_Toc518139414)

[VIII. 2) Masquer/afficher les messages de correction 27](#_Toc518139415)

[VIII. 2) a. Fonctions pour masquer et afficher un élément 27](#_Toc518139416)

[VIII. 2) b. Cacher les messages de correction 27](#_Toc518139417)

[VIII. 3) Vérification du nom 28](#_Toc518139418)

[VIII. 3) a. Code HTML 28](#_Toc518139419)

[VIII. 3) b. Code JS : checkNom() 29](#_Toc518139420)

[VIII. 3) c. Code JS : editionNom() 29](#_Toc518139421)

[VIII. 4) Vérification du mail 29](#_Toc518139422)

[VIII. 4) a. Code HTML 29](#_Toc518139423)

[VIII. 4) b. Code JS : checkMail() 30](#_Toc518139424)

[VIII. 4) c. Code JS : editionMail() 30](#_Toc518139425)

[VIII. 5) Vérification du téléphone 31](#_Toc518139426)

[VIII. 5) a. Code HTML 31](#_Toc518139427)

[VIII. 5) b. Code JS : checkTel() 31](#_Toc518139428)

[VIII. 5) c. Code JS : editionTel() 32](#_Toc518139429)

[VIII. 6) Vérification du message 32](#_Toc518139430)

[VIII. 6) a. Code HTML 32](#_Toc518139431)

[VIII. 6) b. Code JS : checkMessage() 32](#_Toc518139432)

[VIII. 7) Vérification de la catégorie 33](#_Toc518139433)

[VIII. 7) a. Code HTML 33](#_Toc518139434)

[VIII. 7) b. Code JS : checkCat() 34](#_Toc518139435)

[VIII. 8) Vérification avant envoi 34](#_Toc518139436)

[VIII. 8) a. Code HTML 34](#_Toc518139437)

[VIII. 8) b. Code JS : Envoi() 34](#_Toc518139438)

[VIII. 9) Rendu avant envoi 35](#_Toc518139439)

[IX. Le PHP 36](#_Toc518139440)

[IX. 1) Pourquoi avoir fait du PHP ? 36](#_Toc518139441)

[IX. 1) a. Définition 36](#_Toc518139442)

[IX. 2) a. Rôle 36](#_Toc518139443)

[IX. 3) Prérequis 36](#_Toc518139444)

[IX. 2) Comment envoyer les données saisies par l’utilisateur ? 36](#_Toc518139445)

[IX. 3) Une fois envoyées, comment traiter les données ? 37](#_Toc518139446)

[IX. 4) Page de traitement des données : envoi.php 37](#_Toc518139447)

[IX. 4) a. Vérification de l’existence et du caractère non nul des données 37](#_Toc518139448)

[IX. 4) b. Récupération des données 38](#_Toc518139449)

[IX. 5) Rendu après envoi 38](#_Toc518139450)

[IX. 5) a. Si l’utilisateur se rend sur envoi.php sans passer par le formulaire 38](#_Toc518139451)

[IX. 5) b. Si l’utilisateur envoie des données correctes via le formulaire 39](#_Toc518139452)

[ANNEXES 40](#_Toc518139453)

[Annexe 1 : supports pour le tutoriel de HTML 41](#_Toc518139454)

[Annexe 2 : supports pour le tutoriel de CSS 43](#_Toc518139455)

[Annexe 3 : arborescence du projet 44](#_Toc518139456)

[Annexe 4 : code HTML 45](#_Toc518139457)

[Annexe 5 : code CSS 46](#_Toc518139458)

[Annexe 6 : code JavaScript 47](#_Toc518139459)

[Annexe 7 : code PHP 50](#_Toc518139460)

I. Introduction

I. 1) Préambule

Dans le cadre de notre second semestre à l’Institut Supérieur d’Informatique, de Modélisation et de leurs Applications (ISIMA), nous avons travaillé sur un projet de développement. Ce projet a été d’une durée de 20 heures par personne, et a été encadré par M. David Hill.

I. 2) Motivation

Issus de filières différentes (IUT informatique et classe préparatoire), nous avions à cœur de choisir un projet qui ne nécessitait pas de connaissances pré-requises, et où nous pourrions tous les deux apprendre. L’un de nous deux, ayant un bagage plus solide en informatique, possédait déjà des notions sur le sujet, mais ceci n’a pas été dérangeant. En effet, nous pouvons toujours acquérir de nouvelles compétences en développement web, tant la complexité et l’étendu du travail que peut nécessiter un site sont grands. Curieux de savoir comment une page web était construite, et d’en créer à notre tour, nous nous sommes rapidement positionnés sur ce projet qui était notre premier choix.

II. Présentation du projet

II. 1) Objectifs

Pour créer un site web, on doit donner des instructions à l'ordinateur. Il ne suffit pas simplement de taper le texte qui devra figurer dans le site (comme on le ferait dans un traitement de texte Word, par exemple), il faut aussi indiquer où placer ce texte, insérer des images, ou bien encore créer des formulaires.

Pour ce faire, il faut utiliser au minimum deux langages de programmation qui se complètent puisqu’ils ont des rôles différents :

* **HTML** (HyperText Markup Language) : son rôle est de gérer et organiser le contenu : paragraphes, titre, texte, des liens, des images… Il sert à dire au navigateur comment structurer les pages web que l’on visite. Le HTML 5 en est la dernière version, elle rajoute des fonctionnalités telles que la possibilité d'inclure facilement des vidéos.
* **CSS :** (Cascading Style Sheets, aussi appelées Feuilles de style) : son rôle est de gérer l'apparence de la page web (agencement, positionnement, décoration, couleurs, image de fond, taille du texte…). Le CSS 3 en est la dernière version, elle apporte des fonctionnalités telles que l’ajout de bordures arrondies et les ombres.

Notre projet consistait à **créer une page web avec un formulaire en utilisant, donc, HTML 5 et CSS 3. Ce formulaire devait être fait de manière ergonomique** (« de style »).

II. 2) Démarches

Dans un premier temps, nous devions nous initier à la pratique du HTML et du CSS afin de découvrir ou redécouvrir ce qu’était le HTML et le rôle du CSS. Pour cela, nous avons fait un tutoriel chacun : l’un expliquait à l’autre ce qu’était le HTML et l’autre expliquait à l’un comment utiliser le CSS. Chacun devait donc se confronter à des documentations en français ou en anglais pour faire un travail de synthèse et expliquer au mieux ce dont il était nécessaire de maîtriser pour la suite du projet. Puis, nous avons effectué ensemble la mise en forme toute simple d’une page web, qui est devenue par la suite notre page de formulaire.

Une fois ces bases posées, nous sommes rentrés dans le vif du sujet en insérant notre formulaire au sein de notre pageweb. Nous avons donc utilisé le HTML propre aux formulaires et nous l’avons personnalisé grâce au CSS : design des champs, du bouton… Nous avons réalisé un mockup (= une maquette) du site afin d’avoir un modèle à suivre.

Une image contenant capture d’écran

Description générée avec un niveau de confiance très élevé

Figure II.1 : Mockup du formulaire

Ce travail accompli, nous avons profité du temps qu’il nous restait pour approfondir : comment à la fois vérifier si les données recueillies par le formulaire sont celles attendues et rendre la page interactive ? Pour répondre à cette question, nous avons inséré du langage javascript dans notre page. Notre formulaire est alors devenu dynamique.

Enfin, nous nous sommes intéressés au langage PHP afin de pouvoir récupérer et utiliser les données envoyées via le formulaire.

II. 3) Environnement

Les pages web du projet ont été testées sous les navigateurs Mozilla Firefox et Google Chrome. Elles ont été éditées sous Sublime Text, et nous avons utilisés un serveur local Wamp pour utiliser le PHP.

III. Méthodes de travail

III. 1) Word et le suivi de modifications

Nous avons rédigé nos tutoriels sous Word, et nous avons effectué une relecture croisée de nos tutoriels en utilisant le suivi de modification que propose Word. Le suivi de modification permet de connaitre qui a apporté des modifications au document, de choisir les modifications à accepter ou refuser, et aussi d’ajouter des commentaires.

III. 2) Diagramme de Gantt

Le diagramme de Gantt est un outil d’ordonnancement de projet. Il nous a permis de visualiser la liste des tâches à effectuer et d’évaluer le temps à accorder à chacune.

A la fin de notre projet, nous avons obtenu le diagramme de Gantt suivant :

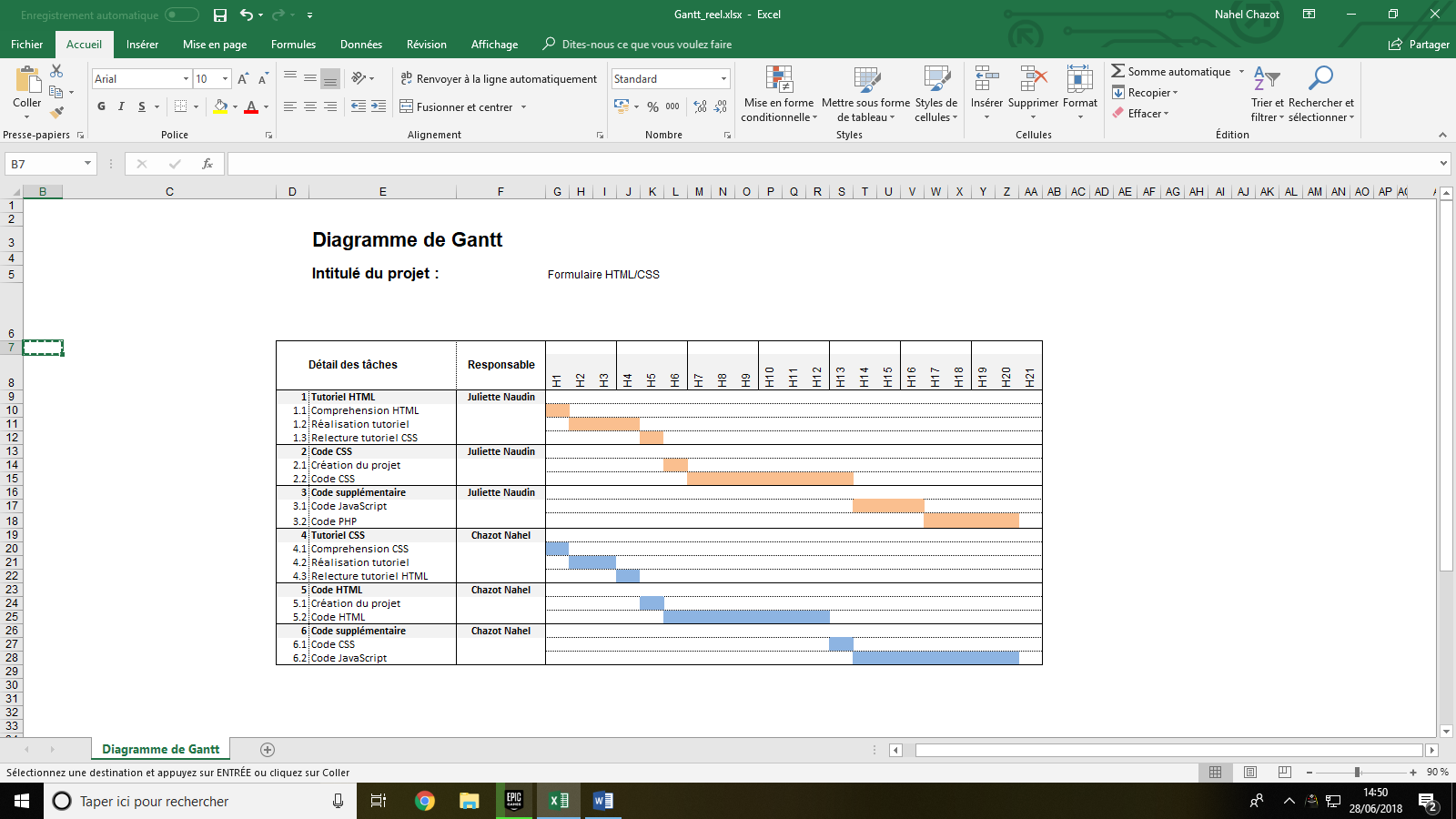


Figure III.1 : Diagramme de Gantt du projet

Les tâches n°1, n°2, n°4 et n°5 sont l’objectif de base du projet. Après les avoir terminées, il nous restait une douzaine d’heures au total. Nous avons alors décidé, comme expliqué précédemment, de rajouter du code supplémentaire, à savoir du JavaScript et du PHP (tâches n°3 et n°6).

III. 3) Git

III. 3) a. Fonctionnement

Git est un logiciel de gestion de version. Le principe est simple, on créer un dépôt distant sur un serveur sur lequel sera stocké notre projet. Dès que l’un des contributeurs veut ajouter ou modifier un élément du projet, il récupère le projet sur sa machine et peut travailler dessus en local.

Quand il a fini, il valide ses modifications en indiquant les fichiers ajoutés/modifiés accompagné d’un commentaire expliquant ce qui a été fait.

Le contributeur renvoie ensuite le projet au serveur qui s’occupe de faire une mise-à-jour.

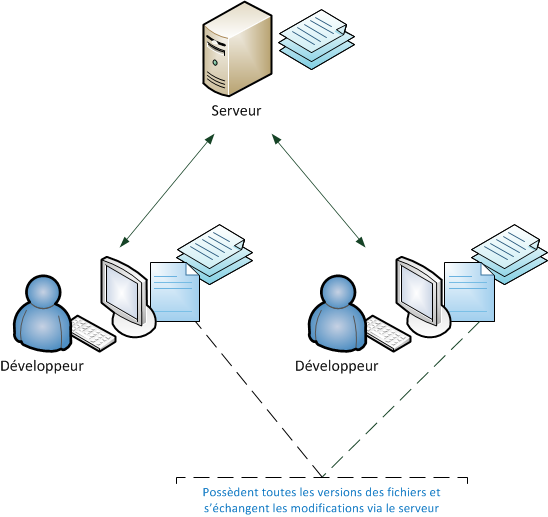


Figure III.2 : Fonctionnement d’un logiciel de gestion de version

III. 3) b. Quelques avantages

* Git enregistre la liste de chaque modification, chacune appelée « commit ». Nous pouvons alors revenir à toute ancienne version du projet. Par exemple si un bug apparait, il est simple de trouver d’où il provient en s’aidant des précédentes modifications.
* Git enregistre l’émetteur de chaque commit. Si un contributeur ne comprend pas une ligne de code, il peut savoir qui l’a ajouté et lui demander de l’aide.
* Un message est attaché à chaque commit. En regardant la liste des commit, nous pouvons suivre le déroulement du projet.

Dans le cadre de notre projet, nous avons utilisé GitHub comme service d’hébergement.

L’adresse du dépôt est la suivante : <https://github.com/nalchaz/formulaire_ZZ1>.

IV. Tutoriel HTML

réalisé par Juliette Naudin pour Nahel Chazot

Ce tutoriel fait référence à l’annexe 1.

IV. 1) Démarrage

On utilise un éditeur de texte (ici Sublime Text) pour créer un fichier ayant l'extension.html (ici : accueil.html). Ce sera notre première page web. Ce fichier peut être ouvert dans le navigateur web.



Figure IV.1 : Fichier d’extension .html

IV. 2) Balises

À l'intérieur du fichier, nous écrirons le contenu de notre page, accompagné de balises HTML. Elles servent à décrire correctement la structure du document. Le navigateur n'affiche pas les balises telles quelles : lorsqu'un utilisateur visite une page web, son navigateur analyse le document et l'interprète afin d'afficher la page web correctement.

IV. 2) a. Type

Les balises peuvent être de deux types :

* <balise> </balise> : celles qui fonctionnent par paire (s’ouvrent et se ferment)
* <balise /> : les orphelines

IV. 2) b. Syntaxe

Balise ouvrante Zone d’application Balise fermante

<p class="ZZ1"> Hello world ! </p>

Attribut(s) (optionnel) Valeur de l’attribut Nom de la balise

Figure IV.2 : Syntaxe d’une balise de type paire

|  |
| --- |
| Avertissement Les balises doivent être fermées dans le sens inverse de leur ouverture :  Coche  <html><body></html></body>  Fermer <html><body></body></html> |

IV. 2) c. Liste des balises HTML ?

Comme il existe de nombreuses balises HTML, les liens ci-dessous ne répertorient que les balises les plus utilisées en pratique :

* <https://openclassrooms.com/courses/apprenez-a-creer-votre-site-web-avec-html5-et-css3/memento-des-balises-html>
* <https://jaetheme.com/balises-html5/>

Notons cependant que pour écrire un commentaire il faut utiliser :

**<!--** Ceci est un commentaire **-->**

IV. 3) Structure de base d’une page HTML

IV. 3) a. La balise <html>

Englobe tout le contenu de la page. Indique au navigateur que le contenu est le code HTML.

IV. 3) b. L’en-tête <head>

Donne des informations générales importantes sur la page comme son titre, l'encodage (pour la gestion des caractères spéciaux), etc.

* **L’encodage (« charset »)**

<meta charset="utf-8" /> : indique l'encodage utilisé dans accueil.html. Détermine comment les caractères spéciaux vont s'afficher (accents, idéogrammes chinois et japonais, caractères arabes, etc.). L’encodage UTF-8 précisé ici permet d'afficher sans aucun problème pratiquement tous les symboles de toutes les langues existantes.

* **Titre de la page**

<title> Prems </title> : donne un titre en haut de la page, à l’onglet du navigateur.

Nb : c’est le titre qui apparaît aussi dans les résultats des moteurs de recherche.

J’ai appelé notre première page « Prems » … c’est la première !

IV. 3) c. Corps de la page <body>

C’est cette balise qui contient le contenu de la page. C’est donc cette balise qui contient la majeure partie du code.

* Une image contenant capture d’écran

  Description générée avec un niveau de confiance très élevé**Image** <img />

src : (attribut obligatoire) chemin (relatif ou absolu) de l’image à insérer

alt : (attribut obligatoire) texte alternatif à afficher si l’image ne peut être chargée

title : rajoute une infobulle

Figure IV.3 : Insertion d’une image

En haut du body, j’ai mis l’image projetZZ.png ci-dessus, ayant pour adresse relative à accueil.html : images/projetZZ.png. Au survol de l’image, on a l’infobulle « un super projet ! » qui apparaît (comme si on ne le savait pas…). Si l’image n’est pas chargée, le texte « Logo\_Projet » est alors affiché.

* **Un peu de mise en page !**

<br/> : retour à la ligne.

<p> : créé un paragraphe. Les paragraphes sont automatiquement séparés par des sauts de lignes.

* **Lien** <a href="#" ></a>

href : chemin (relatif ou absolu) pouvant être externe (ie vers un autre site, on parle alors d’URL et non de chemin) du lien.

J’ai placé un lien sur "ici" dirigeant vers un autre fichier html : formulaire.html, où sera notre formulaire (étape 2 du travail !,... restons encore 2 minutes à l’étape 1).

Nb 1 : D’autres attributs sont bien sûr possibles comme title pour rajouter une infobulle, ou pour avoir un lien qui ouvre une nouvelle fenêtre du navigateur, télécharger un fichier, etc.

Nb 2 : Le hashtag # permet de ne pas bouger de la page sur laquelle on est, même en cliquant sur le lien (🡺 lien fictif).

* **Le pied de page** <footer>

À l'inverse de l'en-tête, le pied de page se trouve en général tout en bas du document.

* **Tableau** <table>

<table> </table> : indique le début et la fin d'un tableau

<tr> </tr> : indique le début et la fin d'une ligne du tableau

<td> </td> : indique le début et la fin du contenu d'une cellule

Une image contenant capture d’écran

Description générée avec un niveau de confiance très élevé

Figure IV.4 : Tableau Noms, Prénoms, Professions

J’ai donc fait un pied de page avec nos coordonnées : une ligne chacun avec nom, prénom et mail ISIMA.

IV. 4) Un site web sans CSS ? Sans… Nahel ?

Résultat de recherche d'images pour "emoji diable"

On peut très bien créer un site web uniquement en HTML (et sans Nahel Chazot ), mais celui-ci ne sera pas très beau : l'information apparaîtra « brute ». On peut tout de même styliser le contenu brut par des balises HTML : comme dans cette première page par exemple, j’ai utilisé <i> pour mettre en italique, <u> pour mettre en gras. Néanmoins l’aspect de cette page laisse toujours à désirer... C'est pour cela que le langage CSS vient toujours le compléter : à toi Nahel !

IV. 5) Sources du tutoriel

Les balises :

<https://developer.mozilla.org/fr/Apprendre/HTML/Balises_HTML>

<http://www.commentcamarche.com/contents/488-balise-html>

Structure :

<https://developer.mozilla.org/fr/Apprendre/HTML/Comment/Cr%C3%A9er_un_document_HTML_simple>

Tableaux :

<https://openclassrooms.com/courses/apprenez-a-creer-votre-site-web-avec-html5-et-css3/les-tableaux-1>

Complément de structure (ex : pied de page) : <https://openclassrooms.com/courses/apprenez-a-creer-votre-site-web-avec-html5-et-css3/structurer-sa-page>

V. Tutoriel CSS

réalisé par Nahel Chazot pour Juliette Naudin

Cette partie fait référence à l’annexe 2.

V. 1) Emplacement

Le CSS peut s’écrire à différents endroits :

* Directement dans le fichier HTML au sein de balise (attribut « style »)
* Dans l’en-tête <head> du fichier HTML
* Dans un fichier .css

La solution la plus propre est d’utiliser un fichier .css qui permet de séparer le code CSS du code HTML pour plus de lisibilité.

Pour ajouter une feuille de style à un fichier HTML il faut lui indiquer où la trouver grâce à la balise <link>. Il faut indiquer l’option « rel » avec la valeur « stylesheet » ainsi que l’option « href » avec la localisation du fichier de style.

Par exemple :

<link rel="stylesheet" href="style.css" />

V. 2) Appliquer un style

V. 2) a. Syntaxe

Pour donner un style on a besoin de trois informations :

* Le **nom** de la balise ou du type de balise
* Les **propriétés** CSS à appliquer
* Les **valeurs** attribuées à ces propriétés

Ainsi, une feuille de CSS ressemble à ça :

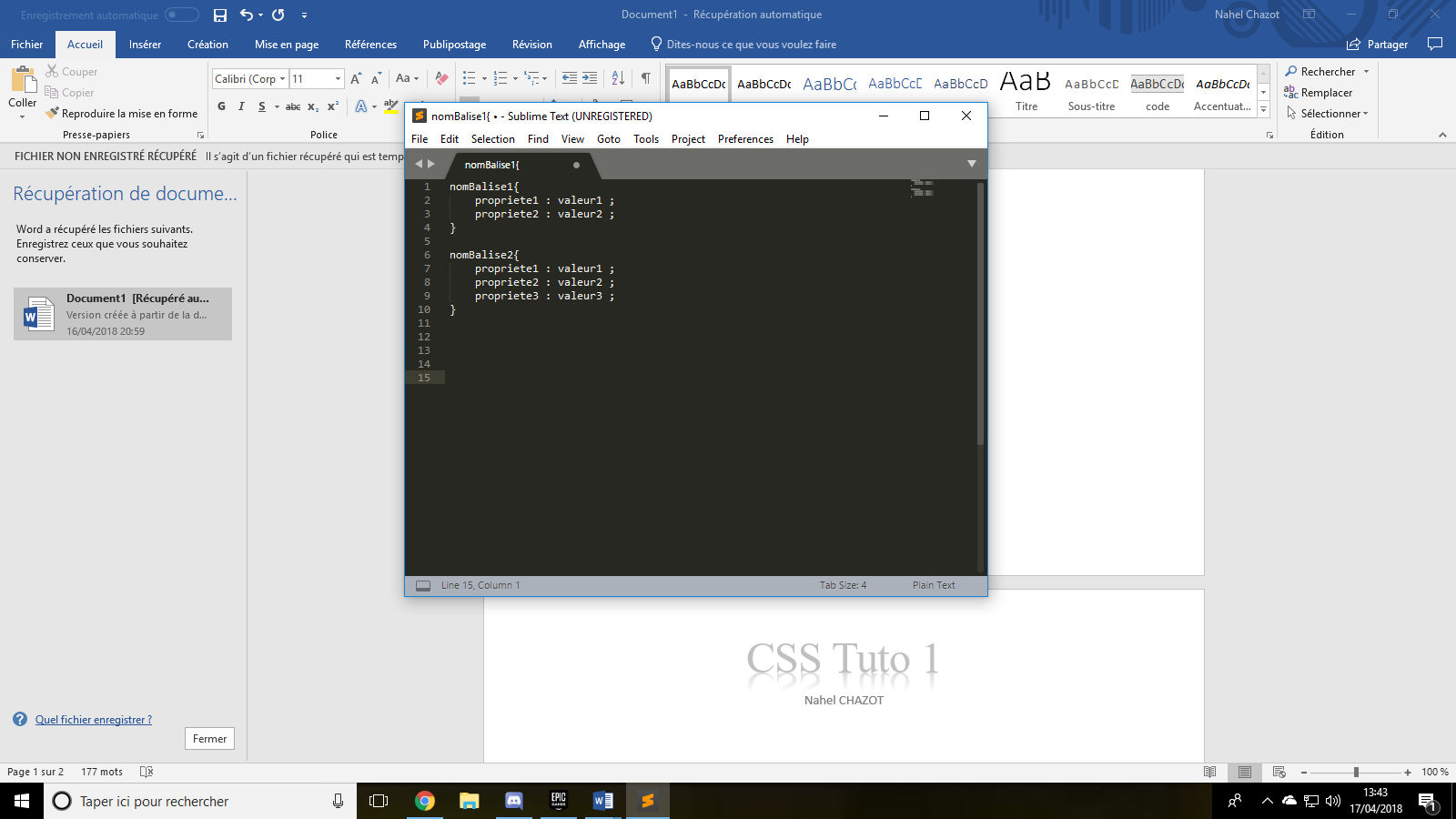


Figure V. 1 : syntaxe d’une feuille de style

*Note : Pour ajouter un commentaire dans un fichiers CSS il faut l’inserer entre* /\* et \*/

V. 2) b. Sélecteurs de balises

La feuille de style « style.css » sera appliqué sur le code HTML suivant :

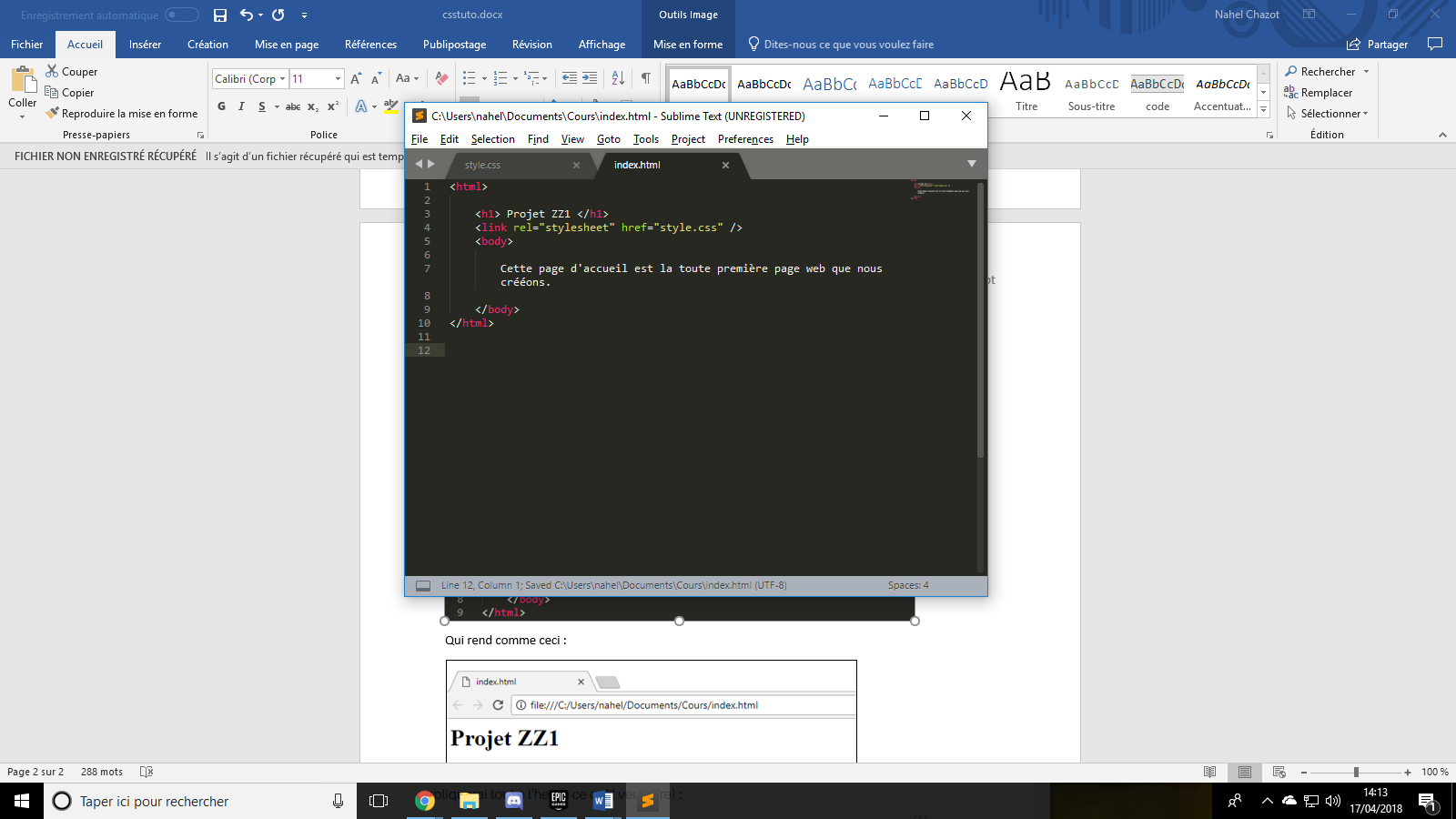


Figure V. 2 : code source HTML où sera appliqué style.css

Qui rend comme ceci :

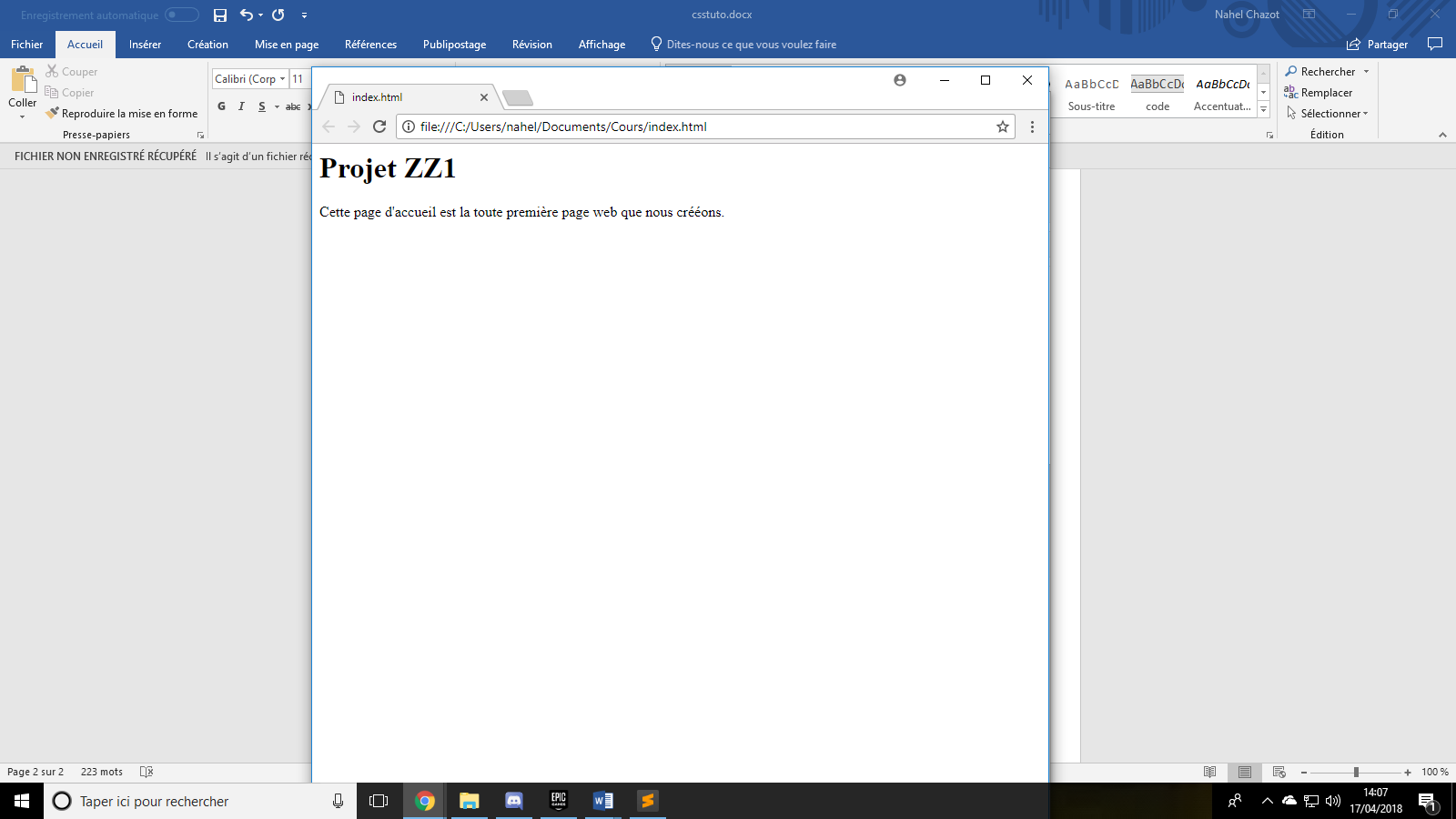


Figure V. 3 : rendu sans CSS

Pour appliquer un style à un type de balise il faut indiquer le nom du type. Par exemple pour appliquer une couleur à tous les grands titres on indique le nom h1 :

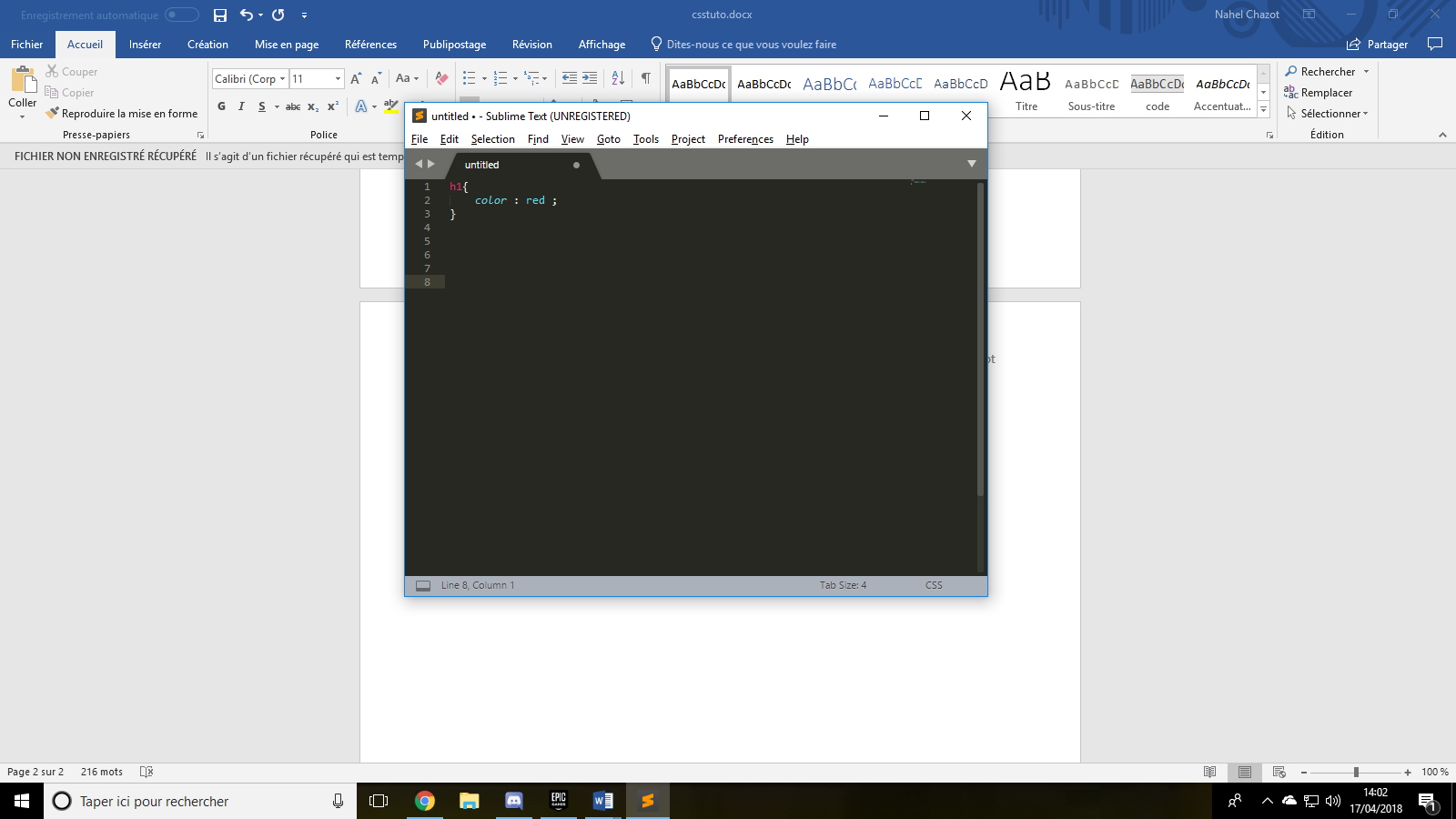


Figure V. 4 : couleur rouge appliqué à tous les titres h1

On obtient alors la page suivante :

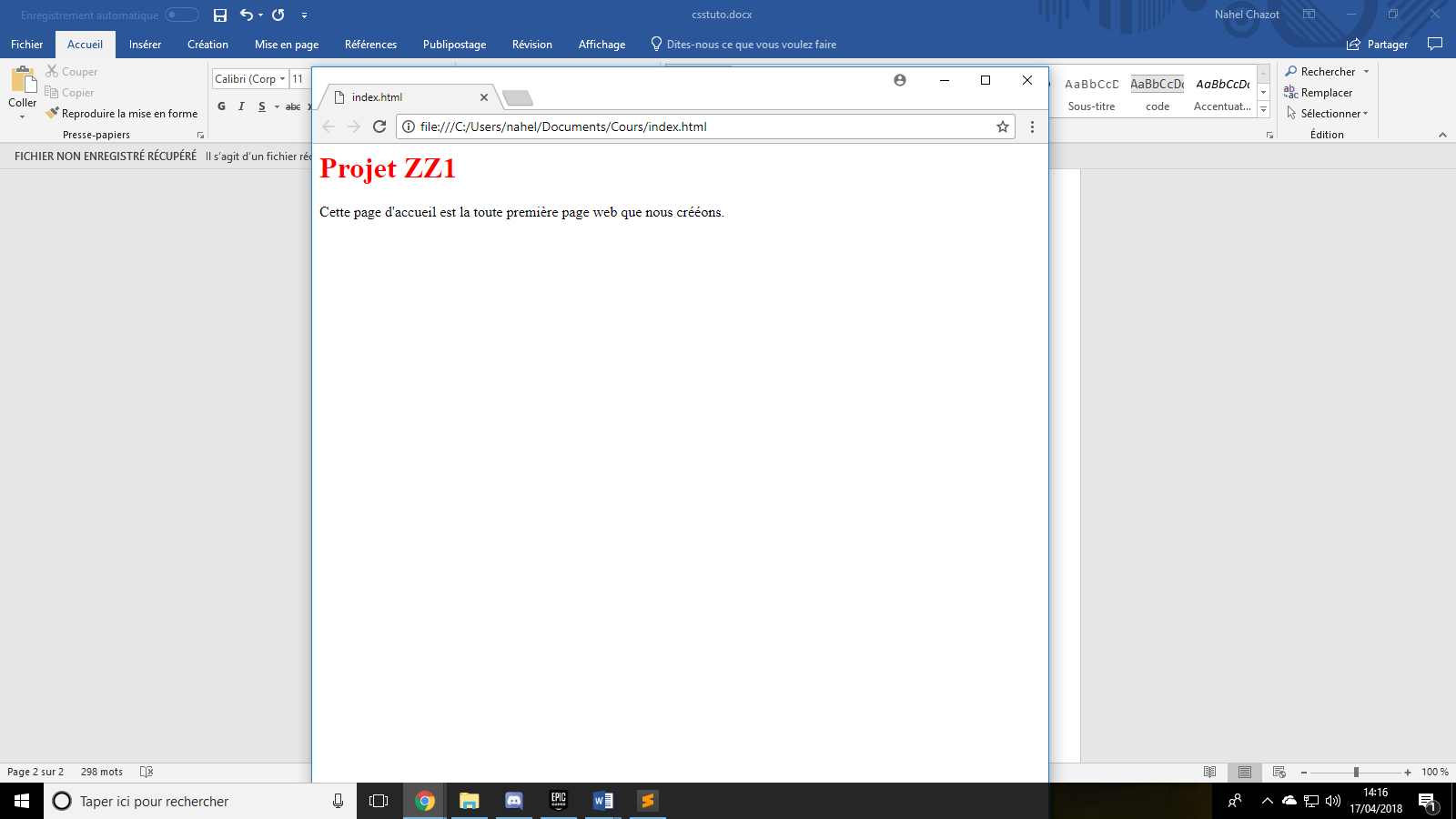


Figure V. 5 : rendu après application de la couleur rouge pour les titres h1

Cependant ceci implique que toute les balises h1 soit écrite en rouge. Comment éviter cela et être plus sélectif ?

V. 2) c. Class et id

Dans les balises HTML, on peut utiliser deux attributs afin de différencier les balises de même type :

* L’attribut class
* L’attribut id

Ces deux attributs sont quasiment identiques, seulement l’attribut class peut être attribué à **plusieurs** balises tandis que l’attribut id ne peut être utilisé qu’**une** **seule fois** dans le code.

Pour sélectionner une classe dans le code CSS, il faut indiquer le nom de la classe en commençant par un point comme ceci :

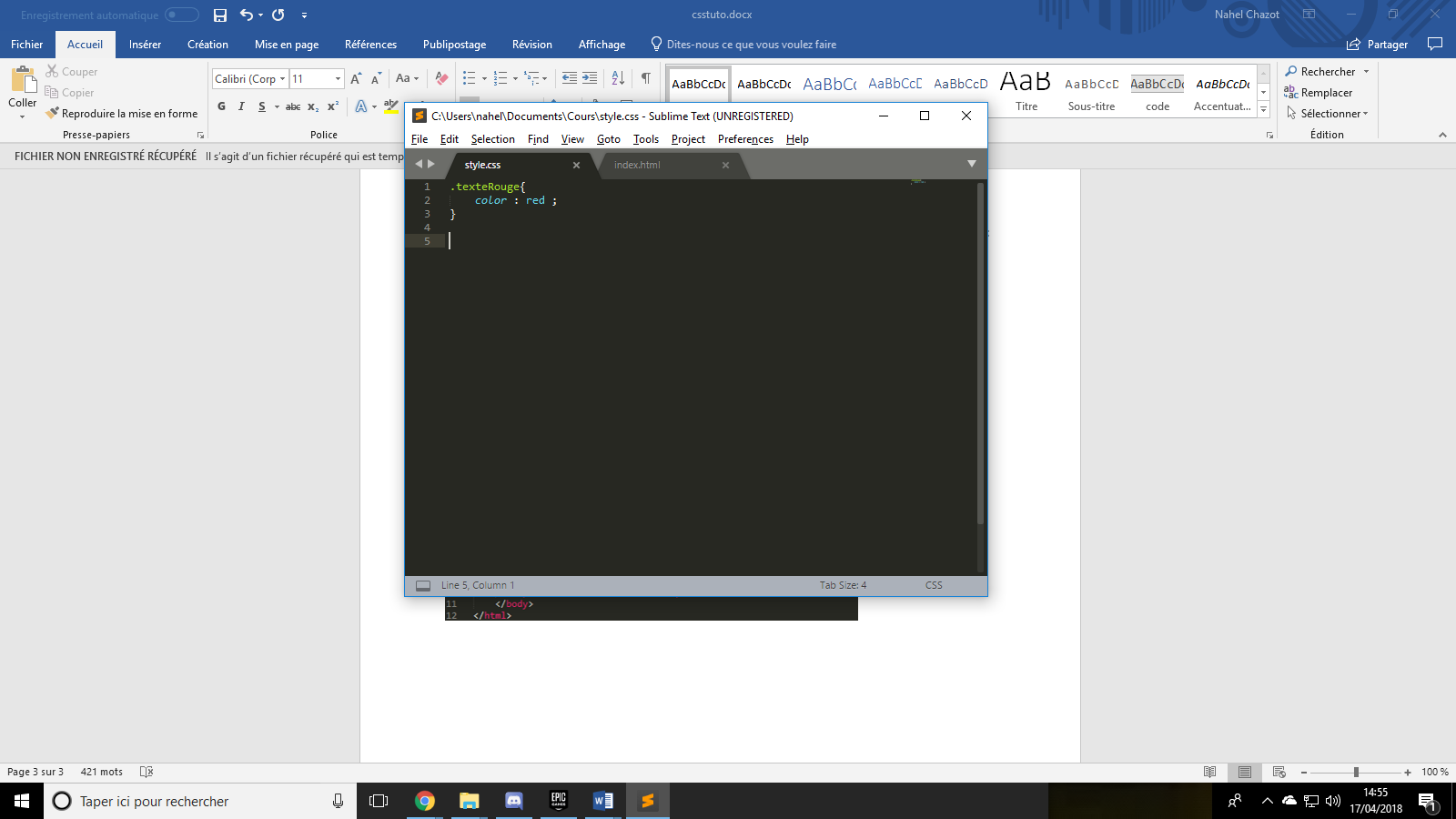


Figure V. 6 : sélection d’une classe avec CSS

Ainsi pour le code HTML suivant :



Figure V. 7 : code source HTML où sera appliqué style.css

On obtient la page web :

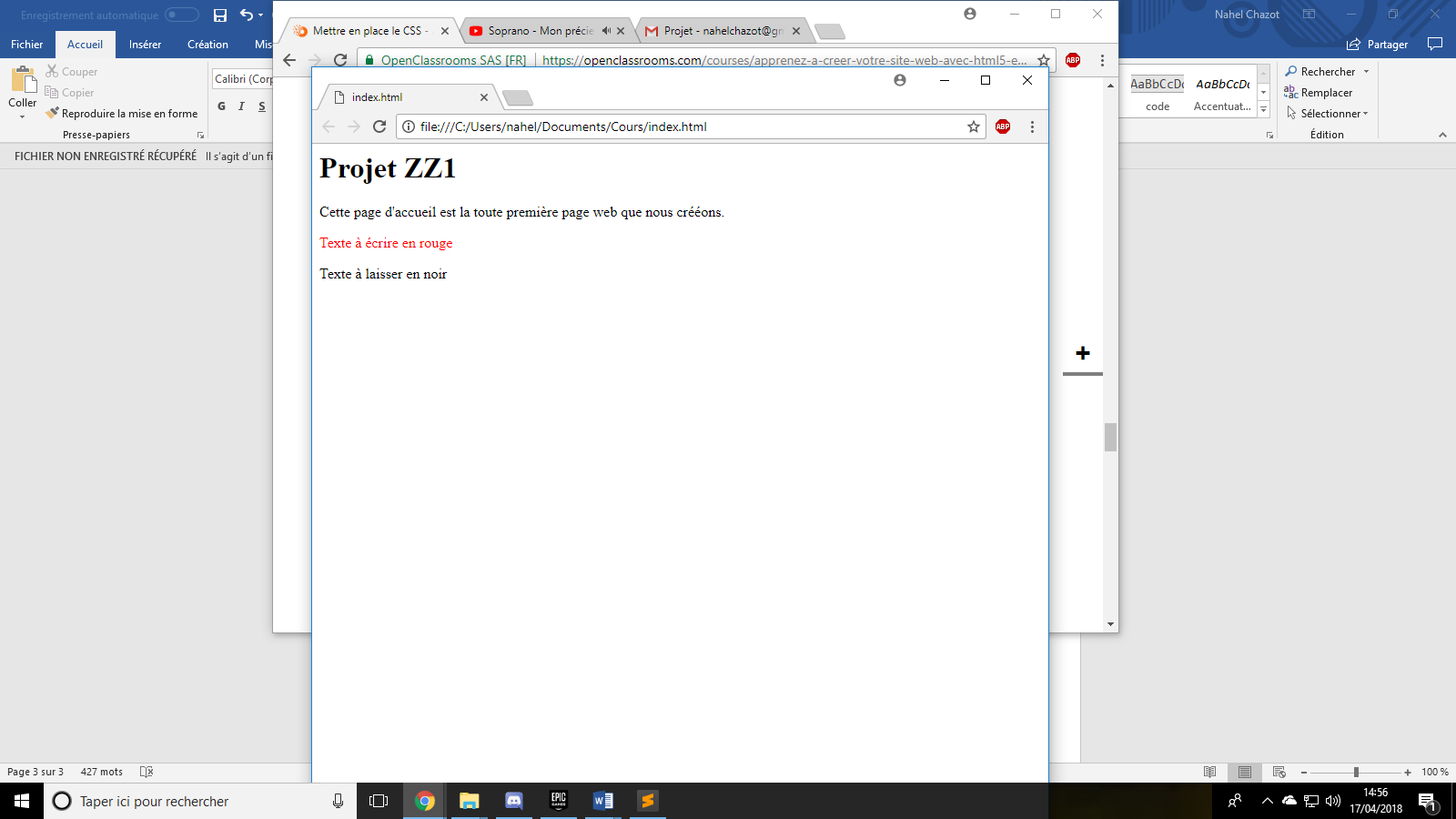


Figure V. 8 : rendu avec CSS

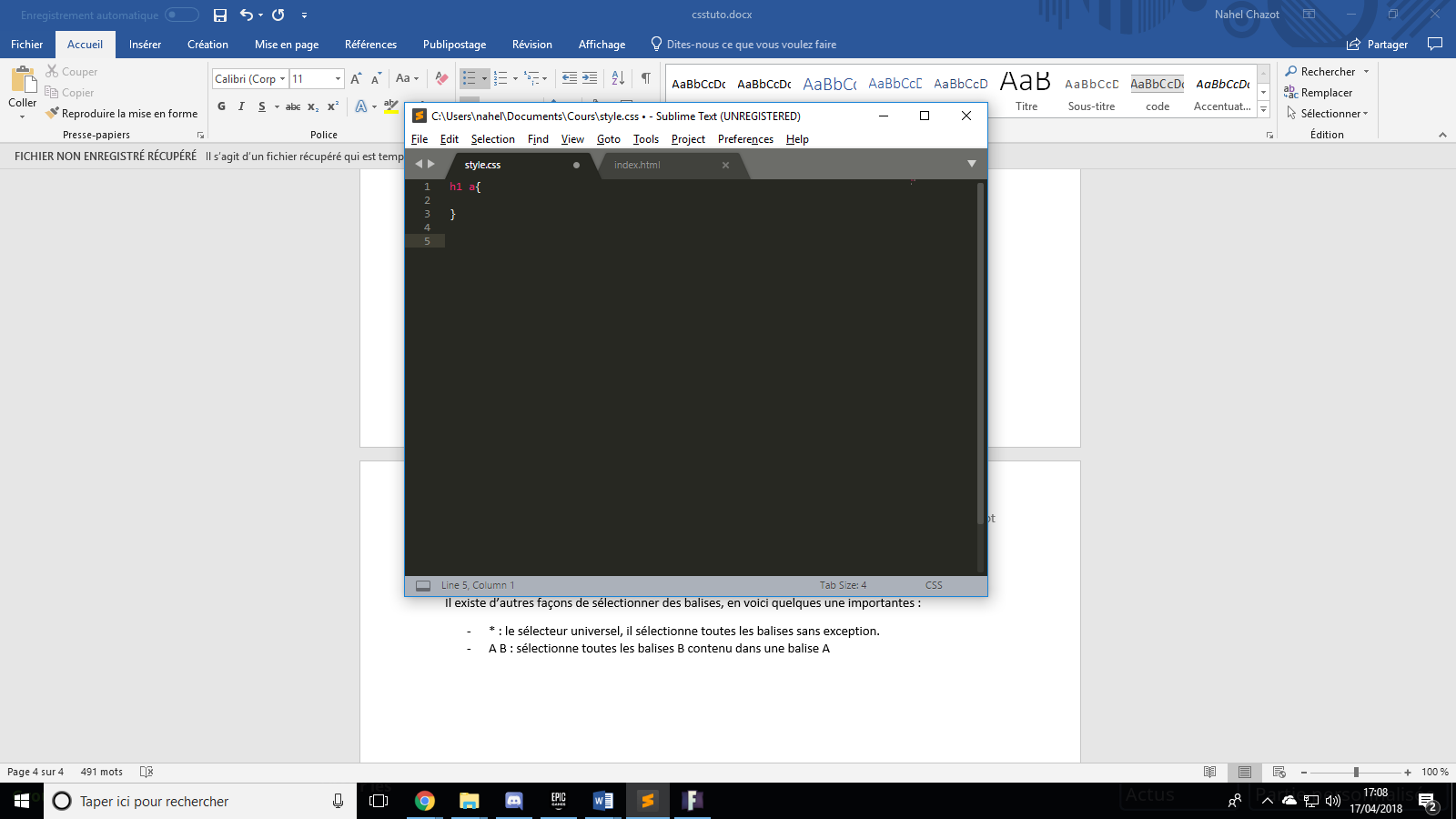
Pour sélectionner un id il faut indiquer le nom de l’id en commençant par un « # ». Le fonctionnement est identique.

V. 2) d. Sélecteurs spéciaux

Il existe d’autres façons de sélectionner des balises, en voici quelques-unes importantes :

* \* : le sélecteur universel, il sélectionne toutes les balises sans exception.
* A B : sélectionne toutes les balises B contenu dans une balise A.

Exemple :



Sélectionne toutes les balises a

contenu dans une balise h1.

Figure V. 9 : Sélection d’une balise contenue dans une autre

A[attribut] ou A[attribut= "Valeur" ] : sélectionne toutes les balises A possédant l’attribut entre crochet ou plus précisément l’attribut et sa valeur.

Exemple :



La première ligne selectionne toutes les balises a possedant un titre.

La cinquième ligne selectionne toutes les balises a possèdant le titre « titre1 ».

Figure V. 10 : Sélection d’un balise possédant un attribut où la valeur est ou n’est pas précisée

Beaucoup d’autres selecteurs existe, ils sont tous listé à la source, le [site du W3C](https://www.w3.org/Style/css3-selectors-updates/WD-css3-selectors-20010126.fr.html#selectors).

V. 3) Propriétés utiles aux formulaires

L’objectif étant de créer un formulaire, intéressons-nous aux propriétés intéressantes afin de le mettre en forme.

V. 3) a. Le texte

Il y a énormément de possibilités concernant la mise en forme du texte. Voici simplement les plus importants :

* La police : font-family : nomPolice;
* La taille de la police : font-size : 10px;
* Le style de la police : font-style : italic; /\* Police en italique \*/

font-weight : bold; /\* Police en gras \*/

- L’alignement : text-align : left/center/right;

V. 3) b. Couleur et fond

* Couleur du texte : color : blue/#FFFFFF ;
* Fond : Background-color : red/#EEEEEE;

V. 3) c. Bordures

Pour les bordures il existe une super-propriété dans laquelle on peut indiquer trois valeurs :

* La largeur (en pixel px)
* La couleur
* Le type de bordure :
  + solid : un trait simple
  + dotted : pointillés
  + dashed : tirets
  + double : bordure double
  + groove : en relief
  + ridge : autre effet relief
  + inset : effet 3D global enfoncé
  + outset : effet 3D global surélevé

Elle s’utilise de la manière suivante :

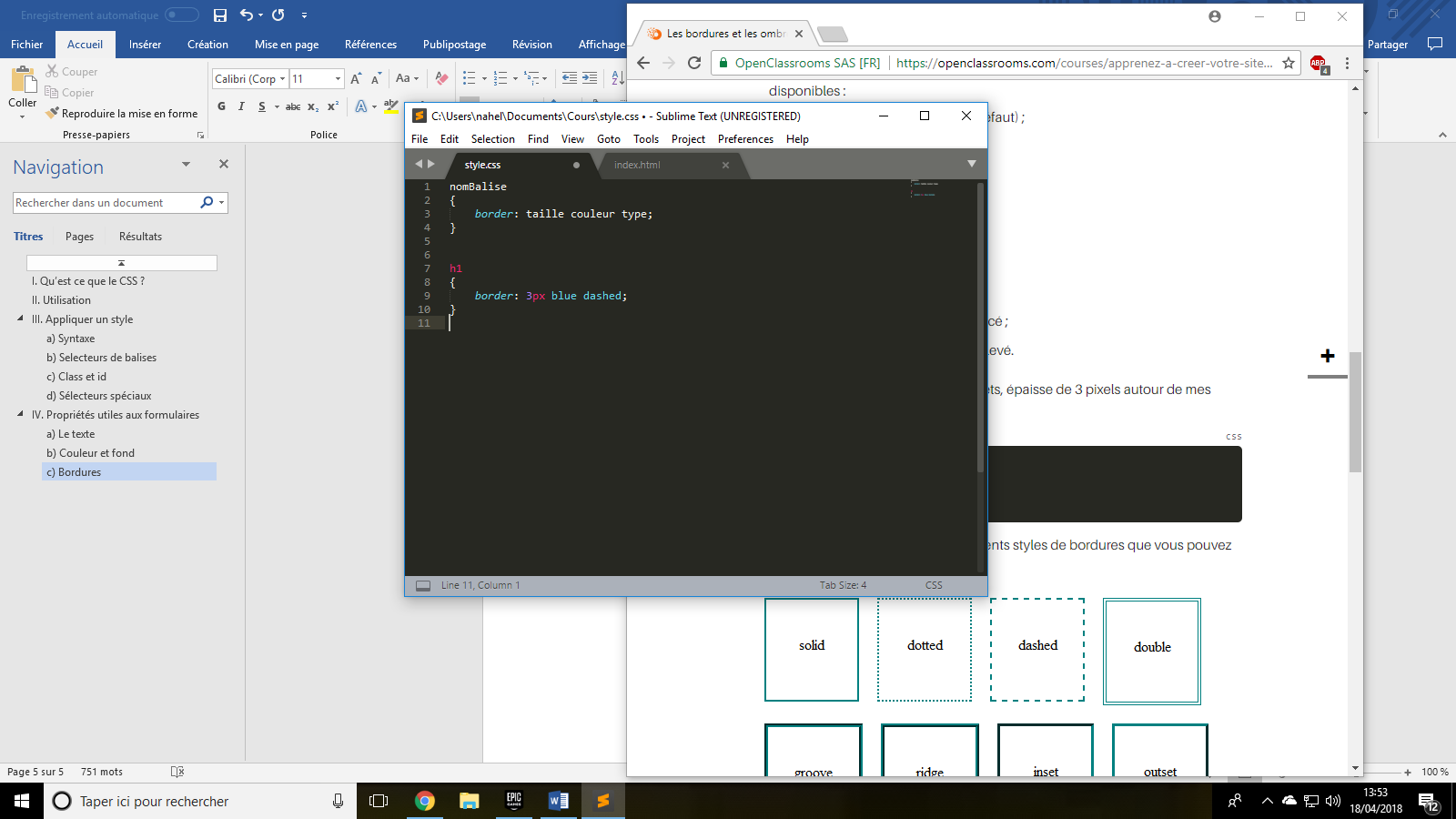


Figure V. 11 : Bordures en CSS

Il est aussi possible d’arrondir les angles des bordures (heureusement) grâce à la propriété border-radius.

Il suffit d’indiquer la taille (« l’importance ») de cet arrondi en pixel :

border-radius : 8px ;

V. 3) d. Les apparences dynamiques

Il est possible de modifier l’apparence d’un élément lors de certains évènements, après que la page ait été chargée.

L’évènement le plus connu étant le « survol » c’est-à-dire quand l’utilisateur passe la souris au-dessus de l’élément.

Pour modifier l’apparence d’un élément lors d’un survol, il faut rajouter l’information :hover après le nom de la balise puis indiquer les propriétés à changer.

Exemple :

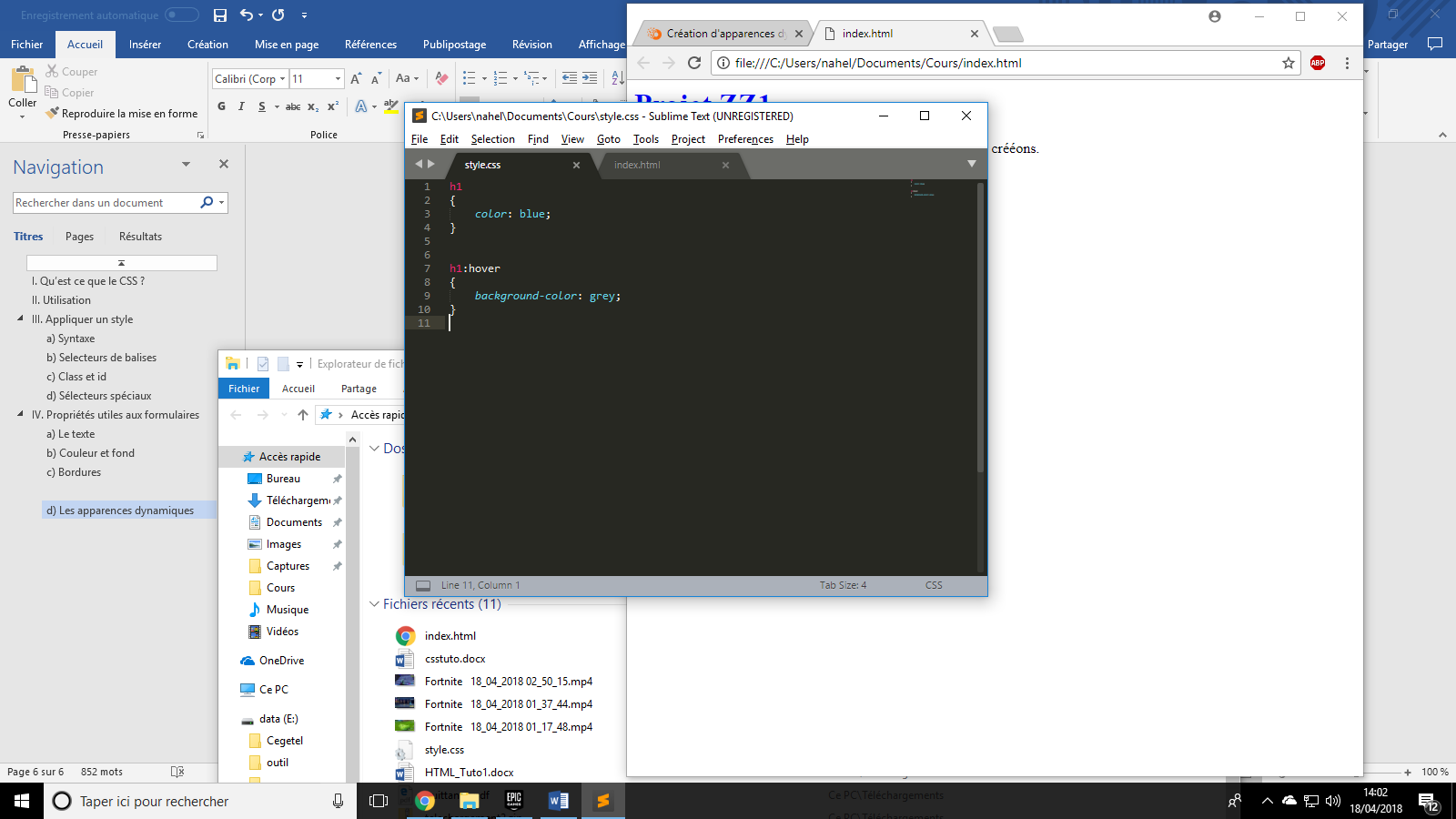
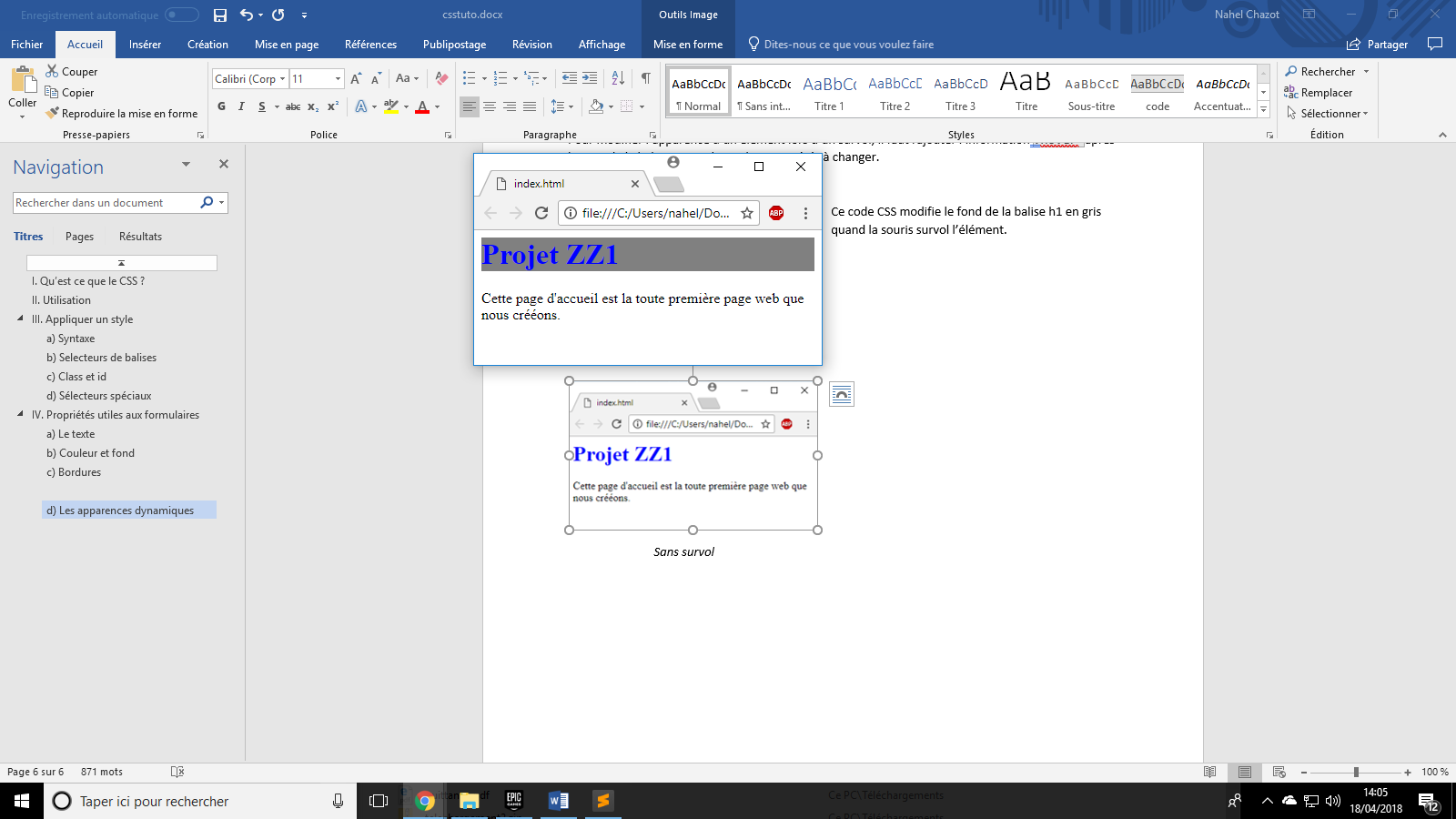


Figure V. 12 : Modification du style d’un élément lors d’un survol (CSS)

Ce code CSS modifie le fond de la balise h1 en gris quand la souris survol l’élément.



*Sans survol Avec survol*

Figure V.13 : Modification du style d’un élément lors d’un survol (Rendu)

Il existe d’autres évènements, la liste de toutes ces pseudo-classes est disponible sur [ce lien](https://developer.mozilla.org/fr/docs/Web/CSS/Pseudo-classes).

V. 4) Suite du travail

Pour la suite, je prépare une page formulaire en HTML et Juliette l’embellira grâce à sa maitrise du CSS.



Figure V. 14 : « You are the CSS to my HTML »

V. 5) Sources du tutoriel

Documentation :

<https://developer.mozilla.org/fr/docs/Web/CSS>

Utilisation du CSS :

<https://openclassrooms.com/courses/apprenez-a-creer-votre-site-web-avec-html5-et-css3/creation-d-apparences-dynamiques>

Sélecteurs d’éléments :

[https://www.w3.org/Style/css3-selectors-updates/WD-css3-selectors-20010126.fr.html#selectors](https://www.w3.org/Style/css3-selectors-updates/WD-css3-selectors-20010126.fr.html%23selectors)

Pseudo-classes :

<https://developer.mozilla.org/fr/docs/Web/CSS/Pseudo-classes>

VIII. Le JavaScript (JS)

Cette partie fait référence à l’annexe 6.

VIII. 1) Pourquoi avoir fait du JavaScript ?

VII. 1) a. Avantages

Grâce à JavaScript, nous pouvons manipuler le formulaire et ses données directement côté navigateur client, avant éventuellement d'envoyer ces données vers un serveur externe. Ainsi, on peut avertir immédiatement l'utilisateur en cas de saisie erronée. Si c’est le cas, le formulaire n’est pas envoyé et on reste sur la page du formulaire, ce qui permet donc :

* d’éviter les redirections, et donc les temps de chargements qui vont avec (qui peuvent être longs si le serveur est saturé) ;
* de ne pas être redirigé sur un formulaire vierge, et donc ne pas avoir à tout retaper

Par ailleurs, le JS permet de prévenir l’utilisateur qu’un champ est mal rempli dès que qu’il a fini de le remplir, et non pas lorsqu’il clique sur le bouton « Envoyer » du formulaire.

Ainsi, le JS a l’avantage d’être très agréable pour l’utilisateur, et c’est pourquoi nous avons décidé de l’incorporer au sein de notre formulaire.

VIII. 1) b. … Mais le JavaScript n’est pas fiable

Une vérification en JS n'est **pas fiable**, pour la simple et bonne raison qu'il est très facile de désactiver le JS. En effet, le JavaScript est un langage qui s'exécute côté client, or le client peut très bien modifier son comportement ou bien même le désactiver, ce qui annulerait les vérifications.

Par conséquent, le test du contenu d'un formulaire en Javascript doit être seulement considéré comme un plus pour le confort de l'utilisation de la page, afin d'éviter un envoi et un rechargement du formulaire si son contenu est invalide. Il ne remplace en aucun cas une vérification faite côté serveur, qui est indispensable et doit être systématique afin de palier le cas où le Javascript n'est pas activé, donc au cas où le formulaire n'est pas vérifié côté client.

VIII. 2) Masquer/afficher les messages de correction

VIII. 2) a. Fonctions pour masquer et afficher un élément

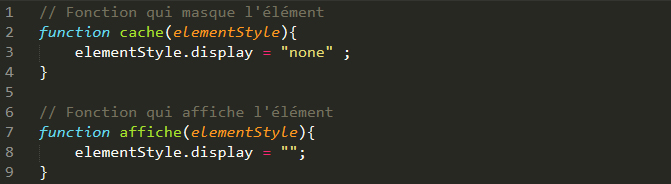


Figure VIII. 1 : code des fonctions JS pour masquer et afficher un élément

Principe : La fonction cache(), resp. affiche(), prend en argument elementStyle qui est le .style d’un élément. Cette fonction modifie la propriété display à « none », resp. « », afin de masquer, resp. afficher, cet élément.

Notons qu’une autre option est possible pour masquer des éléments ; c’est visibility: hidden;. Néanmoins, display: none; masque totalement l'élément et annule des propriétés telles que margin, padding, width, height... tandis que visibility: hidden; masque seulement l'élément, ce qui peut laisser des espaces vides. Nous avons donc choisit display: none; afin de ne pas avoir d’espaces vides.

VIII. 2) b. Cacher les messages de correction

Dans notre code HTML nous avons créé des balises <div> avec la classe .hiddenMessage. Comme le nom de la classe l’indique, ce sont des messages de correction cachés qui seront dévoilés seulement pour informer l’utilisateur du format attendu de chaque champ du formulaire, si jamais ce qu’il rentre n’est pas correct. Seulement, ces balises sont affichées par défaut et il nous faut donc les cacher par le biais du JavaScript.

Nous avons décidé d’afficher par défaut et masquer grâce au JavaScript et pas l’inverse : nous n’avons pas caché par défaut ces balises pour les afficher via le Javascript. En effet, si nous avions procédé ainsi, nous prenions le risque qu'un utilisateur ayant désactivé le JavaScript ne puisse pas voir les messages de correction.

Voici le code de la fonction desactivHidden() contenu dans main.js :

Une image contenant capture d’écran

Description générée avec un niveau de confiance très élevé

Figure VIII. 2 : fonction JS de désactivation des messages de correction

Principe : Une variable hiddenMessage est un tableau contenant tous les objets de classe .hiddenMessage contenu dans le document HTML. Pour chaque objet de ce tableau, on modifie la propriété display à « none » grâce à la fonction cache() définie précédemment.

Cette fonction est appelée après le chargement de la page, afin de cacher les messages avant que l’utilisateur ne commence à remplir le formulaire, d’où son appellation directement dans le main.js, qui lui-même est appelé à la fin du <body>. En effet, une page Web est lue par le navigateur de façon linéaire, c'est-à-dire qu'il lit d'abord le <head>, puis les éléments du <body> les uns à la suite des autres. Ainsi, les éléments <script> seront appelés après le chargement des éléments de la page.

VIII. 3) Vérification du nom

VIII. 3) a. Code HTML

Une image contenant capture d’écran

Description générée avec un niveau de confiance élevé

Figure VIII. 3 : code HTML relatif au champ Nom du formulaire

L'évènement onblur fait appel à la fonction JS checkNom() lorsque le champ relatif au nom sur lequel il est appliqué perd le focus, i.e. quand l’utilisateur quitte le champ.

L’évènement onkeydown fait appel à la fonction JS editionNom() dès que l’utilisateur frappe une touche du clavier dans le champ relatif au nom.

VIII. 3) b. Code JS : checkNom()

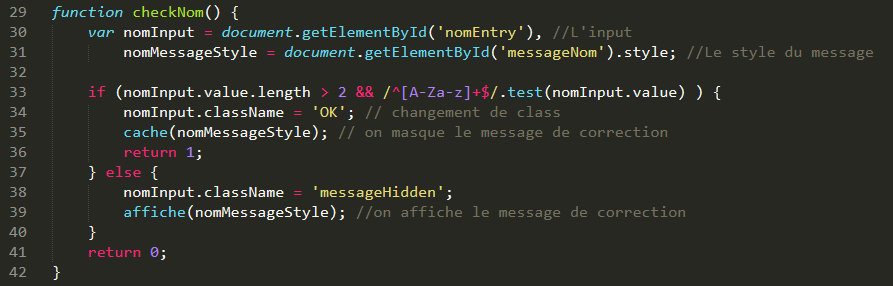


Figure VIII. 4 : fonction JS de vérification du champ Nom du formulaire

Principe : La variable nomInput récupère l’input relatif au champ Nom du formulaire et la variable nomMessageStyle récupère le style associé. On vérifie si le nom fait plus de 2 caractères et grâce à une regex on vérifie que ces caractères ne soient que des lettres majuscules ou minuscules. Si c’est le cas, i.e. si le nom rentré vérifie ces critères, alors la classe de nomImput change et l’on masque le message de correction relatif au champ Non grâce à la fonction cache(). Si ce n’est pas le cas, la classe de nomInput change et on affiche le message de correction relatif au champ Nom grâce à la fonction affiche().

VIII. 3) c. Code JS : editionNom()

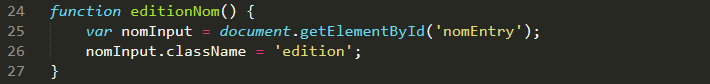


Figure VIII. 5 : fonction JS de …………… ??????????

VIII. 4) Vérification du mail

VIII. 4) a. Code HTML

Une image contenant capture d’écran

Description générée avec un niveau de confiance très élevé

Figure VIII. 6 : code HTML relatif au champ Mail du formulaire

Le code HTML est similaire à la section précédente, avec l’événement onblur qui fait appel à la fonction JS checkMail() et l’événement onkeydown qui fait appel à la fonction editionMail().

VIII. 4) b. Code JS : checkMail()

Une image contenant capture d’écran

Description générée avec un niveau de confiance très élevé

Figure VIII. 7 : fonction JS de vérification du champ Mail du formulaire

Le principe est le même que la fonction de vérification précédente, hormis le fait que cette fois on vérifie si le contenu correspond à la forme d’un mail. On vérifie donc par une regex qu’il y ait bien :

* Tout d’abord un pseudonyme (au minimum une lettre) constitué uniquement de lettres minuscules, chiffres, points, tirets et des underscores.
* Ensuite, un arobase.
* Puis, un nom de domaine, ayant les mêmes règles que le pseudonyme hormis qu’il y a cette fois au minimum deux caractères.
* Enfin, une extension (comme « .fr »), qui comporte un point, suivi de 2 à 4 lettres minuscules. (Il existe « .es », « .de », mais aussi « .com », « .net », « .info », etc.)

VIII. 4) c. Code JS : editionMail()

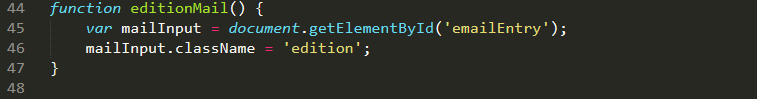


Figure VIII. 8 : fonction JS de …………… ??????????

VIII. 5) Vérification du téléphone

VIII. 5) a. Code HTML

Une image contenant capture d’écran

Description générée avec un niveau de confiance très élevé

Figure VIII. 9 : code HTML relatif au champ Téléphone du formulaire

Le code HTML est similaire à la section précédente, avec l’événement onblur qui fait appel à la fonction JS checkTel() et l’événement onkeydown qui fait appel à la fonction editionTel().

VIII. 5) b. Code JS : checkTel()

Une image contenant capture d’écran

Description générée avec un niveau de confiance très élevé

Figure VIII. 10 : fonction JS de vérification du champ Téléphone du formulaire

Le principe est le même que la fonction de vérification précédente, hormis le fait que cette fois on vérifie si le contenu correspond à la forme d’un numéro de téléphone français. On vérifie donc par une regex qu’il y ait bien :

* le premier chiffre qui est toujours 0 ;
* le second chiffre va de 1 à 8 : 1 à 5 pour les fixes de différentes régions, 6 et 7 pour les téléphones portables, mais il y a aussi le 8 pour les numéros spéciaux ;
* ensuite viennent les 8 chiffres restants allant de 0 à 9.
* Il y a possibilité d’avoir un tiret, un point, un espace ou rien entre chaque paire de chiffres.

VIII. 5) c. Code JS : editionTel()

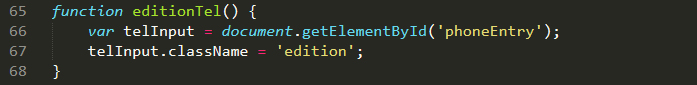


Figure VIII. 11 : fonction JS de …………… ??????????

VIII. 6) Vérification du message

VIII. 6) a. Code HTML

Une image contenant capture d’écran

Description générée avec un niveau de confiance très élevé

Figure VIII. 12 : code HTML relatif au champ Message du formulaire

Le code HTML est similaire à la section précédente, avec l’événement onblur qui fait appel à la fonction JS checkMessage().

**POURQUOI ICI ON FAIT PAS APPEL A UN editionMessage() ???**

VIII. 6) b. Code JS : checkMessage()

Une image contenant capture d’écran

Description générée avec un niveau de confiance très élevé

Figure VIII. 13 : fonction JS de vérification du champ Message du formulaire

Le principe est le même que la fonction de vérification précédente, hormis le fait que cette fois on vérifie si le message fait au minimum 20 caractères et au maximum 1200.

VIII. 7) Vérification de la catégorie

VIII. 7) a. Code HTML

Une image contenant capture d’écran, texte

Description générée avec un niveau de confiance très élevé

Figure VIII. 14 : code HTML relatif au champ Catégorie du formulaire

L'évènement onchange fait appel à la fonction JS checkCat() lorsque l’utilisateur fait un choix de catégorie.

**POURQUOI ICI ON FAIT PAS APPEL A UN editionMessage() ???**

VIII. 7) b. Code JS : checkCat()

Une image contenant capture d’écran

Description générée avec un niveau de confiance très élevé

Figure VIII. 15 : fonction JS de vérification du champ Catégorie du formulaire

Le principe est le même que la fonction de vérification précédente, hormis le fait que cette fois on vérifie si la catégorie est une autre que « Sélectionnez le groupe auquel vous appartenez », i.e. que l’utilisateur a bien choisit une catégorie.

VIII. 8) Vérification avant envoi

VIII. 8) a. Code HTML

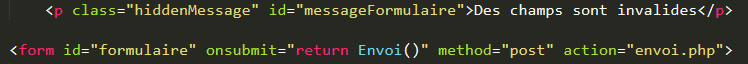


Figure VIII. 16 : code HTML relatif à l’envoi du formulaire

L’évènement onsubmit fait appel à la fonction Envoi() lorsque le formulaire est soumis et avant l'exécution de l'action, donc avant que l’on ne charge envoi.php. Il peut permettre de déterminer si l'action doit être exécutée (si Envoi() retourne True) ou non (si Envoi() retourne False).

VIII. 8) b. Code JS : Envoi()

Une image contenant capture d’écran

Description générée avec un niveau de confiance très élevé

Figure VIII. 17 : fonction JS de vérification du formulaire avant envoi

Principe : Si toutes les vérifications précédentes sont bonnes, alors le message d’erreur du formulaire est masqué grâce à la fonction cache(), une fenêtre averti l’utilisateur que le formulaire est bien rempli, et Envoi() retourne True. Sinon, on affiche le message d’erreur grâce à affiche() et Envoi() retourne False.

VIII. 9) Rendu avant envoi

Si la vérification n’est pas bonne à chaque champ du formulaire, voici un aperçu de ce qu’affiche la page, et ce avant l’envoi :

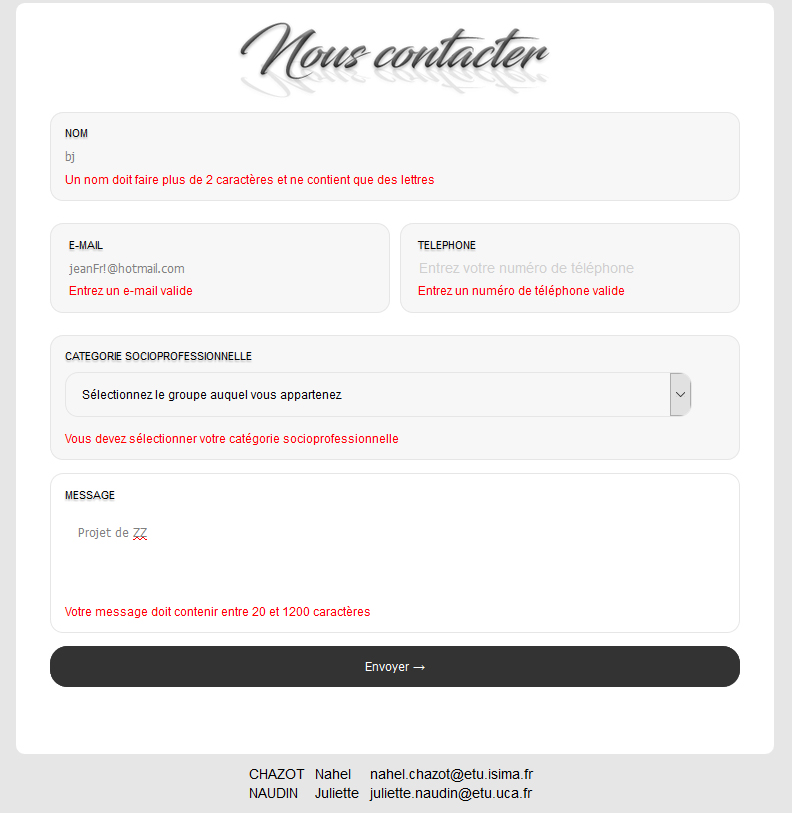


Figure VIII. 18 : rendu en cas d’erreur sur chaque champ, avant envoi

IX. Le PHP

Cette partie fait référence à l’annexe 7.

IX. 1) Pourquoi avoir fait du PHP ?

IX. 1) a. Définition

PHP est un autre langage informatique utilisé sur l’internet. C’est l’abréviation, en anglais, de « PHP Hypertext Preprocessor ». A la différence des langages HTML, CSS et JavaScript qui s’exécutaient côté client, c’est à dire dans le navigateur même des visiteurs, le PHP va s’exécuter côté serveur.

IX. 2) a. Rôle

Le PHP permet de créer des sites dynamiques et interactifs. Par exemple, nous pouvons charger une page différente pour chaque visiteur demandant l’accès à une URL.

Dans ce projet, le PHP va nous permettre de récupérer les données (si elles ont été saisies correctement) rentrées par l’utilisateur dans le formulaire, et d’afficher ses données sur une autre page.

IX. 3) Prérequis

Le PHP étant interprété par un serveur, un serveur est nécessaire pour développer en PHP. Pour ce projet, nous avons installé un serveur WAMP local, que l’on accède en tapant l’adresse http://localhost/ ou http://127.0.0.1/.

La page du formulaire utilisée, formulaire.php, est la même que formulaire.html précédente, hormis le fait que son extension est donc maintenant .php.

IX. 2) Comment envoyer les données saisies par l’utilisateur ?

Une image contenant montrant

Description générée avec un niveau de confiance élevé

Figure IX. 1 : attributs de <form> relatifs au PHP

L’attribut method de <form> indique par quel moyen les données vont être envoyées. Il peut prendre deux valeurs : "get" ou "post". La différence entre ces deux méthodes est que GET fait transiter les données saisies dans le formulaire par la barre d’adresse, contrairement à POST. Les données du formulaire pouvant être considérées comme sensibles, nous avons naturellement choisi la méthode POST.

IX. 3) Une fois envoyées, comment traiter les données ?

L’attribut action de <form> sert à définir la page appelée par le formulaire, ici envoi.php. C'est cette page qui recevra les données du formulaire et qui sera chargée de les traiter. Comme expliqué précédemment, avant l’envoi, c’est onsubmit qui fait appel à la fonction Envoi() et qui permet à l’attribut action de rediriger vers envoi.php si la fonction renvoie True.

IX. 4) Page de traitement des données : envoi.php

IX. 4) a. Vérification de l’existence et du caractère non nul des données

Une image contenant texte, capture d’écran

Description générée avec un niveau de confiance élevé

Figure IX. 2 : vérification de variables existantes et non nulles

Premièrement, il faut vérifier que l’utilisateur arrive sur cette page envoi.php suite à l’envoi du formulaire par formulaire.php, et non pas qu’il ait tapé lui-même l’adresse dans la barre d’adresse du navigateur "…/envoi.php". Pour cela, vérifie s’il y a bien eu un clic sur le bouton d’envoi du formulaire (d’attribut name="submit") grâce à la variable $\_POST[‘submit’] et la fonction empty(). Empty() permet de vérifier si la variable est vide, nulle, ou non définie. Si ce n’est pas le cas, la page web affiche un message d’erreur.

Une fois passé ce test, toujours avec empty(), on vérifie que les champs du formulaires ont été remplis. Cette vérification est nécessaire dans le cas ou l’utilisateur n’utiliserait pas JavaScript. Si les champs du formulaire ne sont pas remplis intégralement, la page web affiche un message d’erreur.

**NB** : Nous nous sommes contentés ici de ne vérifier que l’existence et le caractère non nul des données, mais nous avons conscience que cela n’est pas suffisant pour un vrai site web. Il faudrait également vérifier la conformité des données, i.e. faire le même travail qu’en JavaScript pour chaque champ du formulaire, mais cette fois-ci côté serveur, en PHP.

IX. 4) b. Récupération des données

Une image contenant texte, capture d’écran

Description générée avec un niveau de confiance élevé

Figure IX. 3 : récupération des données dans des variables par php

Après avoir vérifié que des données conformes avaient bien été envoyées, nous les récupérons en les stockant dans différentes variables : $nom, […], $message.

Nous utilisons la fonction htmlentities() pour se prémunir des attaques qui consistent à injecter du code générant des comportements non prévus sur notre site web. On appelle ce type d’attaque « Attaque XSS » ou « Cross-Site Scripting ». La fonction htmlentities() décode les caractères qui ont des significations spéciales en HTML. Par exemple, htmlentities(‘Un apostrophe en <strong>gras</strong>’) affichera « Un apostrophe en &lt;strong&gt;gras&lt;/strong&gt ».

Nous affichons ensuite sur la page les données que l’utilisateur a rentré dans le formulaire.

IX. 5) Rendu après envoi

IX. 5) a. Si l’utilisateur se rend sur envoi.php sans passer par le formulaire

Une image contenant capture d’écran

Description générée avec un niveau de confiance très élevé

Figure IX. 4 : rendu si l’utilisateur ne passe pas par le formulaire (erreur)

IX. 5) b. Si l’utilisateur envoie des données correctes via le formulaire

Une image contenant capture d’écran

Description générée avec un niveau de confiance très élevé

Figure IX. 5 : rendu lorsque l’utilisateur envoie des données correctes via le formulaire

ANNEXES

[Annexe 1 : supports pour le tutoriel de HTML 32](#_Toc518135768)

[Annexe 2 : supports pour le tutoriel de CSS 34](#_Toc518135769)

[Annexe 3 : arborescence du projet 35](#_Toc518135770)

[Annexe 4 : code HTML 36](#_Toc518135771)

[Annexe 5 : code CSS 37](#_Toc518135772)

[Annexe 6 : code JavaScript 38](#_Toc518135773)

[Annexe 7 : code PHP 41](#_Toc518135774)

Annexe 1 : supports pour le tutoriel de HTML

accueil.html :

Une image contenant capture d’écran

Description générée avec un niveau de confiance très élevé

Rendu sur navigateur de accueil.html :

Une image contenant capture d’écran

Description générée avec un niveau de confiance très élevé

Annexe 2 : supports pour le tutoriel de CSS

Annexe 3 : arborescence du projet

Annexe 4 : code HTML

Annexe 5 : code CSS

Annexe 6 : code JavaScript

main.js :

Une image contenant capture d’écran, texte

Description générée avec un niveau de confiance très élevé

Une image contenant capture d’écran, texte

Description générée avec un niveau de confiance très élevé

Une image contenant capture d’écran, texte

Description générée avec un niveau de confiance élevé

Annexe 7 : code PHP

* formulaire.php utilisé est exactement le même que formulaire.html précédemment utilisé, hormis son extension.
* envoi.php :

Une image contenant texte, capture d’écran

Description générée avec un niveau de confiance élevé