Rapport

Un auteur que nous citons de mémoire avait dit qu’il n’existe pas de pratique sans théorie et que toute théorie sans la pratique ne sert à rien. C’est dans cette philosophie qu’il nous a été demandé, en tant qu’étudiants de deuxième graduat en Génie Logiciel de l’Ecole Supérieure d’Informatique Salama (E.S.I.S), par groupe de deux, de réaliser un des quatre projets proposés à l’issu de la fin du cours de JavaScript, afin de trouver un terrain d’entente entre notre théorie acquise et son application dans un projet concret. Parmi ces projets nous trouvons entre autres : Un script qui imite le logiciel Paint de Windows, un script de dactylogiciel, un script de puzzle et enfin un script de génération de questions à choix multiples (Q.C.M).

Etant donné que le travail demandé se doit d’être effectué en duo d’étudiants dont l’un est de la sous-filière Systèmes informatiques et l’autre de la sous-filière de gestion, après débat et délibération, nous avons porté notre choix sur le puzzle, un défi de taille que nous nous réjouissons de relever pour évaluer notre aptitude acquise après les 30 heures du cours de JavaScript.

Certes la réalisation dudit projet n’a pas été une partie de plaisir, elle s’est accompagnée de difficultés et obstacles que nous avons pu remonter contre vents et marées grâce à notre détermination et notre passion.

**Avantages et inconvénients**

Etant donné qu’il nous était pratiquement impossible de voir toutes les facettes du langage de script de prédilection du développement web qu’est le JavaScript, durant les 30 heures confiées à son apprentissage, nous étions contraint de pouvoir approfondir les notions acquises et celles non vues, mais indispensables à la matérialisation de notre projet. Parmi ces notions, nous pouvons citer : l’élément canvas, l’API File, le Drag & Drop, le débogage, le responsive web, etc.

Comme avantages tirés de ce projet, nous avons acquis une bonne base en JavaScript et avons découvert des nouveaux horizons.

**Difficultés rencontrées**

Voici quelques difficultés que nous avons rencontrées dans la réalisation du projet :

* Nous avons eu du mal à choisir au premier abord, la manière de présenter la base de notre puzzle : il nous fallait choisir de le faire soit à l’aide une balise <img />, soit en utilisant les propriétés CSS backgroundImage et backgroundPosition, soit encore le recours à l’élément canvas et sa méthode de dessin ‘drawImage’.
* Nous avons aussi eu l’embarras du choix entre un tableau et une division simple pour accueillir les différentes pièces du puzzle.
* Nous avons beaucoup sué pour mettre en place le module permettant de parcourir l’ordinateur du client pour choisir une image qui servira de base au puzzle.
* Une autre difficulté était la gestion de la taille de l’image étant donné que l’utilisateur a la possibilité de choisir n’importe quel fichier image.

**Solutions adoptées**

* Etant donné qu’avec l’usage des balises <img /> ou des propriétés CSS backgroundImage et backgroundPosition, la gestion de niveaux est une tâche laborieuse, qui demande dans ce cas d’implémenter une balise <table> qui doit contenir des lignes (balise <tr>) et ces dernières doivent à leur tour contenir des cellules (balise <td>), la difficulté réside au niveau de calculer à chaque fois le nombre de lignes et de colonnes qui correspond au nombre de balises <td> et <tr> à placer dans notre balise <table>. Nous avons trouvé judicieux d’utiliser l’élément canvas où nous pouvons dessiner à notre guise les parties de l’image de base pour en faire des pièces du puzzle.
* En ce qui concerne le module utilisant l’explorateur du système d’exploitation du client à la recherche du fichier image à charger comme base du puzzle, notre premier réflexe était d’utiliser l’API file. Cette API, utilisée seule, ne nous permettais d’accéder cependant qu’au nom de l’image choisie sans toutefois nous donner accès à son adresse dans notre script. Par conséquent, le client ne pouvait choisir que les images se trouvant dans le dossier ‘images’ de notre projet sinon le chargement ne serait pas possible faute d’adresse. Mais étant donné les contraintes du projet, nous devrions continuer notre quête, et avons finalement pu combiner l’API File et l’objet FileReader et sa méthode ‘readAsDataUrl ()’ qui se charge de convertir un fichier dans un format DataURL. Ce format consiste à convertir toutes les données binaires d'un fichier en base64 pour ensuite stocker le résultat dans une chaîne de caractères. Les DataURL permettent donc de stocker un fichier sous forme d'une URL lisible par les navigateurs récents.
* En ce qui concerne la taille de l’image qui devant constituer la base du puzzle, nous avons voulus rogner toutes les images qui dépassait le cadre réservé, pensant ainsi rendre notre puzzle adaptable à tout format d’image. Cependant, sur certaines photos de grande taille, on arrivait à récupérer une partie constituée juste du fond de la photo sans image dessus ; ce qui pouvait décevoir le joueur désirant utiliser sa photo et ne voyant qu’un fond presque vide. Nous avons trouvé convenable de limiter la largeur de l’image à 800 pixels et la hauteur à 550 pixels pour un bon affichage, car ainsi grâce au message d’alerte, l’utilisateur pourra alors redimensionner sa photo pour la rendre utilisable dans notre puzzle.