
Développement d'une application web de création de quiz

Version 0.1

Benoît Léo Maillard

14 January 2015

1	Présentation de RapydScript	3
1.1	Pourquoi utiliser Python plutôt que Javascript ?	3
1.2	Différentes manières d’aborder le problème : CoffeeScript, Brython et RapydScript	3
1.3	Introduction à RapydScript	4
1.4	Comparaison du code source et du code généré	7
2	Documentation de l’application	9
2.1	Description des fonctionnalités	9
2.2	Guide du programmeur	11
2.3	Diagrammes de classes et organisation générale du code	11
2.4	Explication des parties clés du code	11
3	Tables des matières	13

Contenu :

Présentation de RapydScript

1.1 Pourquoi utiliser Python plutôt que Javascript ?

Les sites web d'aujourd'hui utilisent de plus en plus le javascript afin de rendre la navigation plus confortable pour le visiteur. Les fonctionnalités offertes par ce langage permettent de concevoir une grande variété d'applications utilisables directement dans le navigateur, qui s'exécutent côté client et qui sont donc par conséquent très accessibles au grand public. Le Javascript est donc un langage quasiment incontournable pour toute personne qui s'intéresse au développement web.

Malgré cela, ce langage n'est pas forcément facile à aborder pour un programmeur habitué à coder en Java, en Python ou dans un autre langage de programmation moderne, car son fonctionnement diffère dans un certain nombre d'aspects fondamentaux. Par exemple, le Javascript est un langage orienté objet à prototype, c'est à dire que les objets utilisés en Javascript sont des copies d'objets prototype, et non des instances de classes comme en Python et Java. On peut également mentionner d'autres différences importantes, comme la portée des variables par défaut. Afin de palier aux difficultés que peut rencontrer un développeur qui débute dans ce langage, divers outils sont apparus pour tenter de remplacer Python par Javascript, avec pour objectif de combiner les possibilités offertes par le Javascript avec la simplicité et la clarté de Python. Ce travail a pour but de présenter et expliquer les fonctionnalités de RapydScript, un outil permettant d'écrire du code très semblable à du code Python (dans sa syntaxe et sa philosophie) et de générer du Javascript à partir de ce code.

1.2 Différentes manières d'aborder le problème : CoffeeScript, Brython et RapydScript

Un certain nombre d'outils open-source tentent de faciliter l'écriture de code Javascript et différentes approches du problème sont possibles. CoffeeScript (<http://coffeescript.org/>) est décrit par ses auteurs comme un langage à part entière, avec sa propre syntaxe (qui ressemble un peu à celle de python, mais qui s'inspire surtout de ruby), qui peut être compilé en Javascript. CoffeeScript permet l'écriture d'un code source plus clair et concis, le code généré est exécuté avec les mêmes performances que du code javascript natif (puisque'il s'agit simplement de javascript), mais son utilisation nécessite l'apprentissage d'un nouveau langage, ce qui peut être un problème pour un débutant.

À l'opposé, Brython (<http://www.brython.info/>) propose une toute autre approche : le développeur peut écrire du code en Python qui sera exécuté par un interpréteur python entièrement intégré dans la page web codé en javascript. C'est certainement la solution la plus fidèle à Python, puisque la syntaxe de Brython est exactement identique à celle de Python. L'accès au DOM et l'utilisation d'Ajax se fait via l'import de modules externes. Brython est donc idéal pour quelqu'un qui ne connaît pas Javascript mais possède de bonnes bases en Python. Cependant, comme la traduction en Javascript se fait "en live", l'impact sur les performances se fait ressentir (<http://pyppet.blogspot.ch/2013/11/brython-vs-pythonjs.html>) et peut poser problèmes pour des scripts complexes. Le script de l'interpréteur (environ 10'000

lignes) doit également être inclus dans chaque page html qui comporte du code Brython. De plus, il n'est pas possible d'utiliser des libraires Javascript externes, telles que jQuery ou AngularJS.

RapydScript a l'avantage de combiner les qualités des deux outils cités plus haut, en permettant d'écrire du code très similaire à du code Python standard et en le compilant en Javascript. Rapydscript est donc facile à prendre en main pour n'importe quelle personne sachant coder en Python et ne limite pas la rapidité d'exécution puisque le code qui sera exécuté est tout simplement du Javascript. Il est aussi possible d'appeler n'importe quelle fonction Javascript standard et par extension d'utiliser n'importe quelle librairie externe. Ce dernier point n'est pas négligeable puisque j'ai opté pour jQuery pour développer mon application de création de quiz en ligne. Le choix de RapydScript pour la réalisation de ce travail est apparu comme évident après avoir considéré les qualités et défauts de ces différents outils. La prise en main ainsi que la compréhension de son fonctionnement a pu se faire aisément, d'autant plus que l'absence de documentation ou de tutoriel complet sur cette technologie en français constitue une motivation supplémentaire et rend ce travail inédit.

1.3 Introduction à RapydScript

1.3.1 Installation

1.3.2 Compilation

1.3.3 Programmer avec RapydScript

Notions de base

Pour un habitué de Python, la prise en main de RapydScript peut se faire très rapidement : il suffit d'écrire son programme comme si on écrivait un programme Python, même si dans certains cas particuliers il n'est pas possible de faire exactement la même chose avec RapydScript. Ces cas particuliers seront abordés plus loin.

Pour commencer avec un exemple simple, voici en python une fonction qui prend nombres en argument et retourne le plus grand. La fonction est ensuite appelée. Ce code est parfaitement valide en Python.

```
def maximum(n1, n2):  
    if n1 >= n2:  
        return n1  
    elif n2 > n1:  
        return n2  
  
maximum(n1, n2) #Appel de la fonction
```

Une fois la compilation effectuée avec RapydScript, on obtient ce résultat :

```
function maximum(n1, n2) {  
    if (n1 >= n2) {  
        return n1;  
    } else if (n2 > n1) {  
        return n2;  
    }  
}  
maximum(5, 18);
```

On peut voir les opérations qu'a fait RapydScript "traduire" le code source en Javascript : Remplacer le mot clé def par function, ajouter les accolades qui englobent la fonction et les if, ajouter des parenthèses autour des conditions, et ajouter un ; à la fin de chaque instruction. On remarque aussi que les commentaires ont été supprimés, puisqu'ils sont seulement utiles dans le code source. Le code ainsi produit peut maintenant être exécuté dans n'importe quel navigateur.

Cet exemple montre cependant un cas assez général puisque le code Javascript correspondant au code source est quasiment identique. Mais RapydScript permet aussi de compiler du code typiquement Python, comme par exemples des boucles for, qui n'ont pas d'équivalent en javascript.

```
names_list = ["Paul", "Marie", "Pierre", "Lucie"]
```

```
for name in names_list:
    print(name)
```

RapydScript produit un code équivalent en Javascript qui s'exécutera comme en Python. Le code généré est cette fois plus complexe et des connaissances en javascript sont nécessaires pour le comprendre. Ce type de manipulations sera étudié dans un prochain chapitre. On peut par exemple aussi implémenter une fonction avec des arguments qui prennent des valeurs par défaut, ce qui n'est pas possible en javascript.

Programmation orientée objet (POO)

Ce qui fait de RapydScript un outil si puissant est principalement les possibilités qu'il offre pour faire de la POO. En Javascript, la POO est basée sur le prototypage, et il est beaucoup plus complexe de créer ses propres objets, avec de l'héritage, etc. En python, cela est beaucoup plus simple, il est donc particulièrement intéressant de pouvoir utiliser la POO Python pour faire de la programmation web front-end.

Encore une fois, il suffit d'écrire une classe comme on le ferait en Python :

```
class MyObject:
    def __init__(self, name):
        self.name = name

    def get_name(self):
        return self.name
```

```
object = MyObject("Object 1") #Instanciation avec un paramètre
object.get_name() #Retourne "Objet 1"
```

Il est également possible de faire de l'héritage :

```
class MyObjectPlus(MyObject): #Classe qui hérite de la classe créée précédemment
    def __init__(self, name, number):
        MyObject.__init__(self, name)
        self.number = number

    def get_number(self):
        return self.number

    def informations(self):
        return "Nom : " + self.name + " /// Nombre : " + str(self.number)
```

```
objet = MyObjectPlus("Objet 2", 5)
objet.get_name() #Retourne "Objet 2"
objet.get_number() #Retourne 5
objet.informations() #Retourne "Nom : Objet 2 /// Nombre : 5"
```

Il n'est par contre pas possible de définir des variables de classes avec RapydScript (si on définit une variable de classe, RapydScript l'ignore simplement). Cela est dû au fait qu'en Javascript, un objet n'est pas une instance d'une classe comme en Python, mais une copie d'un objet prototype. En Python, une variable de classe est en fait un attribut de l'objet Class. Il n'y a donc qu'une seule espace en mémoire pour cette variable. En Javascript, tous les attributs du prototype sont copiés à chaque fois qu'un objet est créé et il n'y a aucun équivalent aux variables de classe.

Utilisation de la bibliothèque standard Python

Avec RapydScript, il est possible d'utiliser dans le code source des fonctions provenant de différentes sources. La première est la bibliothèque standard de Python, c'est à dire les fonctions natives Python, telles que `print()`, `len()` ou `range()`.

Pour utiliser les fonctions de la bibliothèque standard Python, il faut placer l'instruction suivante au début du fichier Python :

```
import stdlib
```

RapydScript définira ainsi ces fonctions dans le fichier généré et leur comportement sera en quelque sorte simulé en Javascript. Ces fonctions peuvent donc être utilisés comme on le ferait en Python, dans le code source.

Séparer son code en plusieurs fichiers

Il est déconseillé d'écrire tout son code dans un seul fichier lorsqu'on travaille sur un gros projet. Pour séparer son code en plusieurs fichiers, RapydScript prévoit un système d'imports qui ressemble à celui de Python. Voici comment procéder pour écrire son code dans plusieurs fichiers.

Premièrement, chaque fichier du code source doit utiliser l'extension `.pyj`, qui est l'extension des fichiers RapydScript. Ensuite, ces fichiers peuvent être utilisés comme des modules et être importés depuis un autre fichier. Par exemple, si on a placé une partie de notre code dans un fichier "moduletest.pyj", on ajoutera l'instruction suivante en début de fichier :

```
import moduletest
```

Lors de la compilation, RapydScript va rassembler tous les fichiers du code source dans un même fichier Javascript, ce qui facilite aussi l'insertion du script dans un fichier HTML. Il est important de noter, cependant, que l'import d'un module ne rend pas ses fonctions disponibles dans un namespace distinct (ce qui est le cas en Python par contre). Par exemple, pour appeler la fonction `test` du module de l'exemple précédent, voici comment procéder :

```
import moduletest
```

```
test() #Correct  
moduletest.test() #Ne fonctionne pas
```

Utilisation de fonctions Javascript natives ou provenant de librairies externes

Il est également possible d'utiliser des fonctions Javascript natives, par exemple :

```
#Ces deux expressions sont équivalentes :  
console.log("Bonjour") #Fonction javascript  
print("Bonjour") #Fonction python
```

L'exemple parle de lui-même et ne nécessite pas d'explications supplémentaires. Il peut parfois être pratique d'utiliser des fonctions qui n'ont pas d'équivalent en Python.

Mais une autre grande force de RapydScript est la possibilité d'utiliser des librairies Javascript externes, telles que jQuery ou AngularJS. Pour cela, rien de plus simple, il suffit d'insérer le script (par exemple jQuery) que l'on veut utiliser dans le code HTML, comme ceci :

```
<html>
<head>
  <script src="jquery.js"></script><!-- jQuery -->
  <script src="myscript.js"></script><!-- Script créé avec RapydScript -->
</head>
<body>
  <div id="mydiv"></div>
</body>
</html>
```

On peut maintenant utiliser les fonctions jQuery dans notre code. Cette fonction sélectionne le <div> et y insère du texte :

```
def add_text(text):
    $("#mydiv").text(text)

add_text("Hello World")
```

On peut procéder de la même manière pour n'importe quelle autre librairie externe Javascript.

Débugger avec RapydScript

Cas problématiques

1.4 Comparaison du code source et du code généré

Documentation de l'application

2.1 Description des fonctionnalités

2.1.1 Professeurs

Création de quiz

Les professeurs peuvent créer un quiz en utilisant un format texte de type markdown pour créer différents types de questions et y attribuer certaines caractéristiques. Cet éditeur de quiz se présente sous la forme d'une zone de texte dans laquelle le professeur peut entrer le code du quiz. Au début de chaque ligne, le prof entre un tag, chaque tag a une signification différente. Chaque tag doit être séparé de la suite de la ligne par un espace

Tag	Signification
##	Question à choix multiple avec plusieurs options cochables
**	Question à choix multiple avec une seule option cochable
^^	Liste déroulante avec une option sélectionnable
??	Question avec un champ de texte à remplir
*	Option invalide dans un QCM
=	Option valide dans un QCM
=	Réponse correcte dans une question à champ de texte
.	Permet de définir le nombre de points sur la question (par défaut, 1)
+	Ajout d'un commentaire d'explication qui sera affiché lors de la correction

Voici un exemple de quiz comprenant 4 questions :

```
## Énoncé de la question à choix multiple (plusieurs cases peuvent être cochées)
* Option 1
= Option 2 (correcte)
* Option 3
= Option 4 (correcte)
+ Commentaire affiché à la correction
. 1.5

** Énoncé de la question à choix multiple (une seule case peut être cochée)
* Option 1
* Option 2
= Option 3 (correcte)
. 2

^^ Liste déroulante
* Option 1
```

```
= Option 2 (correcte)
* Option 3
* Option 4

?? Question à champ de texte
= Réponse correcte
= Autre réponse correcte possible
```

Lorsque le professeur appuie sur le bouton “Aperçu”, il peut voir le rendu du quiz tel que le verront les étudiants. Les éventuelles erreurs dans le code (tag qui n'existe pas, etc.) sont affichés en rouge en dessous de l'aperçu. Le quiz ne peut pas être enregistré tant qu'il y a encore des erreurs dans le code. S'il commence à créer un quiz et désire le terminer plus tard, le professeur peut enregistrer un brouillon. L'outil de création de quiz supporte l'affichage des mathématiques avec MathJax (<http://www.mathjax.org/>).

Le prof doit également donner un titre au quiz et peut le relier à un plusieurs chapitres. Il décide de restreindre l'accès aux membres d'un ou plusieurs groupes ou de le rendre public. Pour créer un quiz, un prof a aussi la possibilité de reprendre un quiz déjà existant et de modifier son code. S'il opte pour cette solution, les statistiques du nouveau quiz ainsi créé repartent à zéro, puisque les questions ne sont plus forcément les mêmes.

Consultation des statistiques avancées

Après que les élèves d'un groupe ont complété un quiz, le professeur peut afficher le résultat moyen des élèves, le % de réussite à chaque question, mais aussi les réponses soumises par chaque élève en particulier. Le prof peut ainsi se faire une idée globale et plus ciblée du niveau de compréhension de son groupe.

2.1.2 Étudiants

Trouver un quiz

Pour trouver un quiz, un étudiant a plusieurs possibilités. Le professeur peut donner l'url exacte du quiz à compléter, ce qui peut être pratique dans un email ou toute communication informatisée. Une fonctionnalité permet aux utilisateurs de générer un code QR correspondant à un quiz, ce qui est idéal pour afficher sur un projecteur en classe ou sur une feuille imprimée. Un étudiant peut aussi accéder à un quiz en mémorisant son id et en l'entrant dans le champ prévu à cet effet sur la page “Trouver un quiz”.

Dans son espace utilisateur, l'étudiant peut aussi consulter les derniers quizzes créés dans son groupe et peut donc y accéder facilement.

Compléter un quiz et correction automatique

Une fois que l'étudiant a accédé au quiz, il peut le compléter très simplement en remplissant les champs de formulaires affichés. Lorsqu'il a fini, il peut soumettre ses réponses et est redirigé vers une page de correction. Les réponses incorrectes sont affichées en rouge avec la solution et une explication pour chaque question. Les points reçus pour chaque question sont affichés avec le total de points sur le quiz. L'étudiant peut aussi comparer son score à la moyenne des autres étudiants du groupe. Un champ de texte est disponible pour envoyer des éventuelles remarques au professeur (signaler une erreur, poser une question). La page pour compléter un quiz ainsi que celle de la correction sont optimisées pour mobile et le design responsive s'adapte parfaitement à tous types de périphériques (desktop, portable, tablette, téléphone mobile).

2.2 Guide du programmeur

2.3 Diagrammes de classes et organisation générale du code

2.4 Explication des parties clés du code

Tables des matières

- *genindex*
- *modindex*
- *search*