QuickVU

ICOM 4036-036
Second Semester 2015

Members:

- Tarik Calderon
- Benjamin Acevedo
- Michael Mercado
- Andros Rosa

Introduction

QuickVU is a programming language designed to facilitate user interface development for web projects by merging the most common UI web technologies into one product. The current UI development method for the web requires knowledge of various web technologies such as HTML, CSS, JavaScript. Hence, developers end up working with different languages each with different syntax, procedures and learning curves. Consequently, our motivation for creating QuickVU comes from the idea that the process of developing web UI's can be simplified by unifying the most used web technologies and standards into one programming language. QuickVU provides a tool for developers where the code they write is converted into a fully functional static website containing all the required HTML, Javascript and CSS code in combination with standard web development libraries and dependencies.

Motivation

Our motivation for developing QuickVU was to make easies HTML scripts. Coding in HTML can be troublesome and frustrating. With our language we can optimize time in making a script with basic elements. We made it so that HTML veterans and beginners can use QuickVU to their advantage. For example a beginner with little knowledge can understand what the language does for them. Also, a advanced user can modify the script generated by QuickVU and continue from that point, adding or subtracting content as they please. Making it a useful tool for developing web pages.

Language Features

- Do more with less. Create a working UI quickly with only a few lines of code.
- Choose from a variety of templates for your web page.
- Integrated support for well known dependencies such as JQuery and Bootstrap.
- Dedicated modules for handling basic Bootstrap and JQuery functions.
- Basic form handling support.
- Mobile device responsive UI Support.
- Views are generated in HTML 5.
- Resulting HTML packaged in ready to use folder structure.
- Test your web page instantly

Approach

We first downloaded an example provided by ply. The example was a calculator, from there we began to understand how a programming language works. Also, we downloaded Gardensnake, which basically is a small *Python* game. This one was more complicated, but it complemented the knowledge gained from studying the calculator example. We used the examples mentioned above to understand the proper way to define grammar and rules inside PLY. Afterwards it was just a matter of implementing the grammar and rules we designed for QuickVu and merging them with the intermediate code.

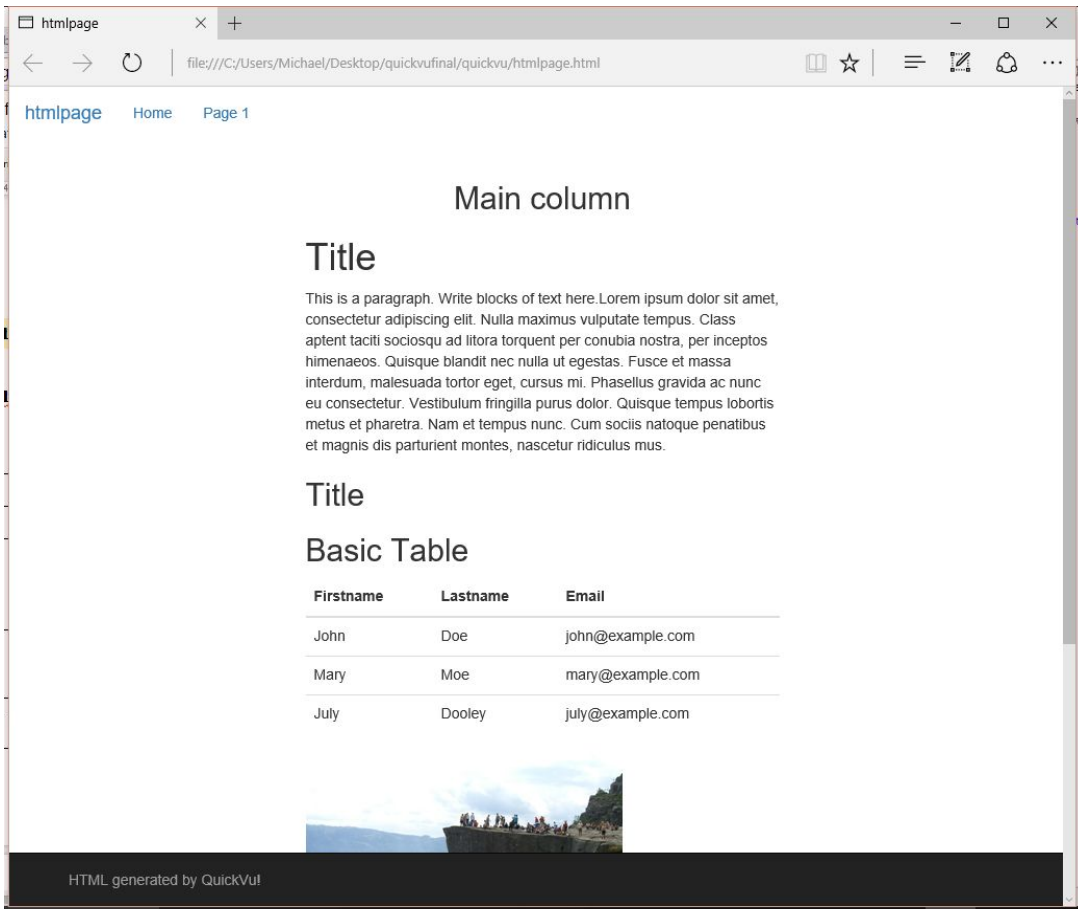
Software Specifications and Requirements

- Python 2.7
- PLY
- HTML5 lib
- CSS
- Bootstrap
- Pyquery
- Git
- Lxml
- Bs4 (Beautifulsoup)

Language Tutorial

1) QuickVu > <vucreate htmlpage>
2) QuickVu > <vumenu 1>
3) QuickVu > <vuelement heading1>
4) QuickVu > <vuelement paragraph>
5) QuickVu > <vuelement heading2>
6) QuickVu > <vuelement table>
7) QuickVu > <vuelement image>
8) QuickVu > <vuform textarea>
9) QuickVu > <vuform dropdown>
10) QuickVu > <vuform password>
11) QuickVu > <vuform submit>
12) QuickVu > <vufinish>

After <vufinish>, the web page will open automatically.



Language Reference Manual

Example: <vucreate parameter>

Command	**Parameter**	**Description**
<vucreate page_title>	Any name	The title of the web page
<vumenu x>	1, 2, 3	1) White navigation template 2) Black navigation template 3) Black navigation and responsive template
<vuelement headingX>	heading1, heading2, heading3, heading4, heading5, heading6	Creates a dummy heading
<vuelement paragraph>	paragraph	Adds a dummy paragraph
<vuelement table>	table	Creates a dummy table
<vuelement image>	image	Displays a dummy image
<vuelement list>	list	Creates a dummy list
<vuelement button>	button	Creates a default button
<vuform textarea>	textarea	Creates a form with a text area
<vuform dropdown>	dropdown	Creates a dropdown on the form

<vuform radio>	radio	Creates a radio element for the form
<vuform checkbox>	checkbox	Creates a simple checkbox element
<vuform text>	text	Creates a simple are to input text
<vuform password>	password	Creates area for password input
<vuform number>	number	Creates area to input numbers
<vuform submit>	submit	Creates a submit button
<vufinish>	N/A	Finishes the web page and opens it. Also adds default footer for web page.

Language Development

Translator Architecture

The architecture of the translator is based on three components. First, our lexical analyzer (Lex in our case) which is where the language rules and the tokens that used for the syntaxis of our language are defined. Second, the parser (Yacc in our case) which validates that our statements conform with the rules defined for our language and contains the logic for mapping statements to the corresponding actions in the intermediate code. And our third component is the intermediate code which contains the functions that will create the HTML file based on the code entered by the user.

Interfaces between the modules

With the lexer we can scan for rules for regular expressions, to make this possible we must write a collection of regular expressions. The yacc module will let us create a program that takes tokens and establish a language from them. It uses BNF grammar that ensembles tokens. To define a function for each of the rules, it's necessary to make the parameter a iterable object, something like a list. This list will contain each symbol for the rule.

Software for creating the Translator

Sublime Text is a cross-platform source code editor with a Python application programming interface (API). It natively supports many programming languages and markup languages, and its functionality can be extended by users with plugins, typically community-built and maintained under free-software licenses.

Test methodology used during development

The test methodology we used was System Testing combined with a Test Early Test Often approach. Since the start of development, we performed tests on a system level for each new iteration of the development. This way we made sure that all modules were working together accordingly and providing the expected output.

Programs used to test your translator

The program used to test the translator were Python 2.7 and the most recent versions of Firefox, Chrome and Internet Explorer web browsing software. Whenever a new implementation to the translator was done, we executed a code containing the expression to test the translator similar to the one below:

<vucreate tarik>

<vumenu 2>

<vuelement image>

<vufinish>

Then we evaluated the result in the web browsers mentioned above to verify that the output matched our expectations.

Conclusion

With the creation of QuickVu, we were able to get hands on experience on how a programming language works and get a better understanding of its components. We also learned how to put theory into practice by applying concepts learned in class into a fully working programming language. With this knowledge, we can better understand the implementation of complex languages such as Java and Python. In general, this project helped us view programming languages as powerful problem solving tool that can help us simplify and automate tasks. In the future, more HTML elements and commands can be added in order to enable the language to generate more complex web pages.