QWEB Programming Language Proposal

Tamanna Hussain, Ramisa Murshed, Kamrul Hossain, Xabier Peralta
{th2704, rm3508}@barnard.edu, {kh2857, xp2159}@columbia.edu

February 3, 2021

1 Overview of QWEB

The QWEB programming language is an imperative and functional pseudocode-style language inspired by block-based visual programming languages such as Scratch and Snap. The purpose of this language is to help both novel and experienced programmers design and develop websites to run in their browser. Although popular web programming languages such as HTML and CSS are generally easy to learn and understand, our goal is to create a language that fuses the two into one that is more intuitive, maximizes human readability, and incorporates familiar programming constructs that are used in traditional high-level programming languages. In essence, QWEB is to HTML/CSS as Processing.js is to JavaScript.

A few key aspects of our language include:

- Python- and Processing-inspired syntax that aligns with the pseudocode standards outlined by Dr. John Dalbey, a retired Professor from California Polytechnic State University.
- Weak static type system that supports modular programming, lazy evaluation, and garbage collection in order to ease the learning curve for programmers who are simply interested in creating web pages.
- Type inference to incorporate conventional pseudocode standards and eliminate the need for users to explicitly declare the types of function parameters and local variables in their code.

2 Language Details

2.1 Data Types

QWEB's primitive data types are a combination of types that are important in both Python and HTML, such as lists, dictionaries, rectangles, circles, etc. Additional types may be added to our language as it evolves.

Data Type	Description	Examples
int	One or more characters in the range 0-9	x = 4
float	One or more characters in the range 0-9 with a decimal point	x = 3.5
bool	A binary variable, having two values: true and false	x = true, y = false
str	A sequence of characters stored in a character array	x = "language"
char	Any alphabetic letter or symbol	x = 'a'

list	An abstract data type that represents a countable number of ordered values	x = ['a', 'b', 'c']
dict	An abstract data type composed of (key, value) pairs with any key appearing only once in the collection	x = 'Languages': 'Python', 'Java'
date	A string consisting of the month, date, and year	x = '2021-02-03'
time	A string consisting of the hour, minutes, and seconds	x = '23:20:50'
size	An unordered set of two valid non-negative integers that do not have a leading '0' character and are separated by a single "x" character	x = "16x16"
rect	A comma-separated list of four integers representing the distance in CSS pixels from the sides of the image to the sides of the rectangle and the height and width	x = [20, 20, 150, 100]
circ	A comma-separated list of three numbers representing the distance in CSS pixels from the edges of the image to the center of the circle and the radius	x = [50, 50, 40]
poly	A comma-separated list of at least six integers that represent coordinates and points of the polygon	x = [200, 10 250, 190 160, 210]

2.2 Keywords

The following are reserved keywords in QWEB:

```
REPEAT until, IF, OTHERWISE, OTHERWISE IF, SET, FOR each, function, length, object, true, false,
    output, display
```

2.3 Syntax

The syntax of QWEB combines pseudocode with Python-like syntax. We used elements of both because pseudocode makes it easy to explain commands like variable initialization and loops, while Python simplifies commands used to reference objects and create functions. A fusion of the two may help when teaching or learning our language, particularly if our users are children or beginner programmers.

2.3.1 Functions

Functions in QWEB are declared by the keyword **function**, similar to how Python declares functions using the keyword **def**. Functions can also have arguments, meaning that modular programming is supported in QWEB. This is an example of a function in our language:

```
function addition(a, b):
    output a + b
```

2.3.2 Loops

QWEB supports several loops, including the **FOR each** loop (corresponding to the Python **for** loop, the **REPEAT until** loop, corresponding to the Python **while** loop, and the **IF...OTHERWISE** loop, corresponding to the Python **IF...ELSE** loop. This is an example of the usage of a **REPEAT until** loop in our language:

```
SET i to 0
REPEAT until i >= list1.length:
       list2.append(x)
       increase i by 1
```

2.3.3 Type Inference

QWEB's compiler will infer the type of each variable used in a program at compile-time, making our language weakly typed and eliminating the need for users to declare each variable's type upon initialization. This will help simplify variables for children and beginner programmers who may use this language. For example, this is the initialization of a variable that is of type **int**.

```
SET x to 14
```

2.4 Object-Oriented Programming

Our language supports object-oriented programming without inheritance. This is an example of an object class in QWEB:

```
object Orange:
    function constructor(length, width):
        SET this.length to length
        SET this.width to width

    function area():
        output this.length * this.width
```

This is an example of how to instantiate an object in a program:

```
// prints the area of an Orange object
SET object Orange(12,4) to clementine
SET areaOfClementine to clementine.area()
display areaOfClementine
```

2.5 Standard Library

Our language's standard library consists of built-in methods for lists and tables, such as append and remove, as well as for other data types. The standard library also contains methods for creating elements

designated by HTML tags, including but not limited to createHeader, createParagraph, createTable, and createUnorderedList.

3 Sample Program

The following program illustrates an example of how to create a table using HTML:

```
<table border = "1">
    <tr>
        <th>Firstname</th>
        <th>Lastname</th>
        <th>Age</th>
    </tr>
    <tr>
        <td>Jill</td>
        <td>Smith</td>
        <td>50</td>
    </tr>
    <tr>
        <td>Eve</td>
        <td>Jackson</td>
        <td>94</td>
    </tr>
</table>
```

QWEB proposes an alternative way to create a table by incorporating features that are common in functional programming languages and traditional high-level languages, such as higher-order functions and loops:

```
// stores the information for the table in a dict
SET data to:
        "Firstname": ["Jill", "Eve"],
        "Lastname": ["Smith", "Jackson"],
        "Age": [50, 94]

// uses the built-in createTable function to return the structure of the table
SET table to createTable(row = 3, col = 3)

// uses a nested for loop to add the data to the cells in the table
FOR each key in data:
    FOR each val in key:
        table.append(val)
```

4 Example of a Web Page

4.1 Program

```
// function declaration that builds the actual webpage
function createPage:
    SET header to "This is a Header"
    createHeader(header)

    SET subheader to "This is a Subheader"
    createSubheader(subheader)
```

```
8
9   SET paragraph to "This is a paragraph"
10  createParagraph(paragraph)
11
12  SET list to ["Apples", "Bananas", "Pears", "Oranges", "Grapes"]
13
14  SET createdunList to createUnorderedList()
15  SET createdorList to createOrderedList()
16
17  SET uList to "UnorderedList"
18  createSubheader(uList)
19
20  FOR each x in list:
21      createdunList.append(x)
22
23  SET oList to "OrderedList"
24  createSubheader(oList)
25
26  REPEAT until i >= list.length:
27      createdorList.createList(x)
28      increase i by 1
29
30  SET tableText to "Table"
31  createSubheader(tableText)
32
33  SET data to:
34          "Firstname": ["Jill", "Eve"],
35          "Lastname": ["Smith", "Jackson"],
36          "Age": [50, 94]
37
38  SET table to createTable(row = 3, col = 3)
39
40  FOR each key in data:
41      FOR each val in key:
42          table.append(val)
43
44  SET shapes to "Rectangle & Circle"
45  createSubheader(shapes)
46
47  createRectangle()
48  createCircle()
```

4.2 Output

The following screenshot is an example of how the web page would look after the program is run:

This is a Header

This is a Subheader

This is a paragraph.

Unordered List

- Apples
- Bananas
- Pears
- Oranges
- Grapes

Ordered List

1. Apples
2. Bananas
3. Pears
4. Oranges
5. Grapes

Table

Firstname	Lastname	Age
Jill	Smith	50
Eve	Jackson	94

Rectangle & Circle