

# Chapter 1

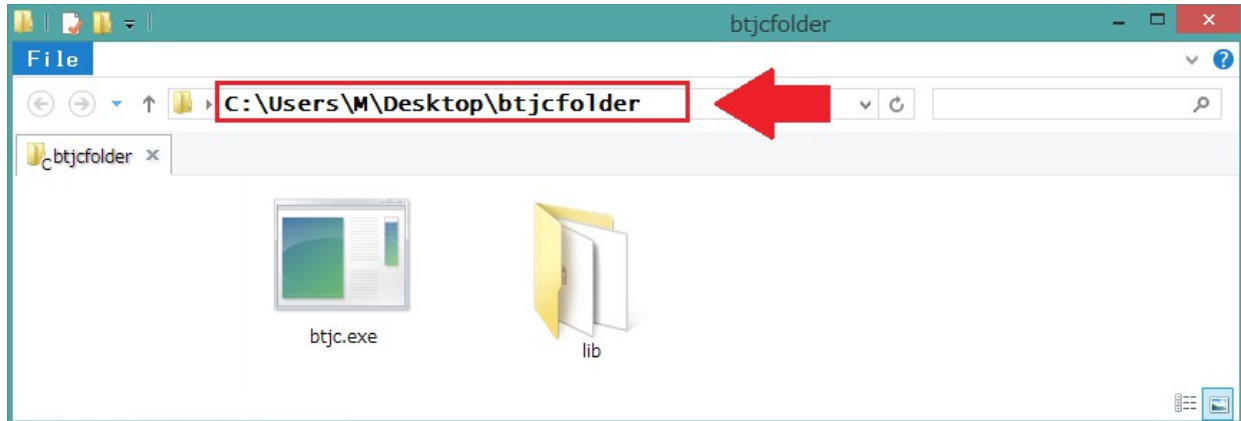
## *Converting Basic To Javascript (Using BTJC)*

**This chapter will provide step by step instructions on how to ...**

1. Convert and run your first program
2. Using Options

## Chapter 1 - 1. Convert and Run Your First Program

1. If you haven't already done so extract "btjc.exe" file and the "lib" folder from btjc.zip to a folder
2. Copy or remember the path of where btjc.exe was extracted.



e.g. C:\Users\M\Desktop\btjcfolder

3. Launch command prompt and cd to the folder where btjc was extracted  
(Type "cd C:\Users\M\Desktop\btjcfolder" into the command prompt)

```
C:\>cd C:\Users\M\Desktop\btjcfolder

C:\Users\M\Desktop\btjcfolder>dir
Volume in drive C is BOOTCAMP
Volume Serial Number is 6849-3C97

Directory of C:\Users\M\Desktop\btjcfolder

2013/07/29  06:21    <DIR>          .
2013/07/29  06:21    <DIR>          ..
2013/06/27  17:41                61,440 btjc.exe
               1 File(s)                61,440 bytes
               2 Dir(s)  76,527,792,128 bytes free

C:\Users\M\Desktop\btjcfolder>
```

4. Open notepad or any plain text editor. Type in the following program:

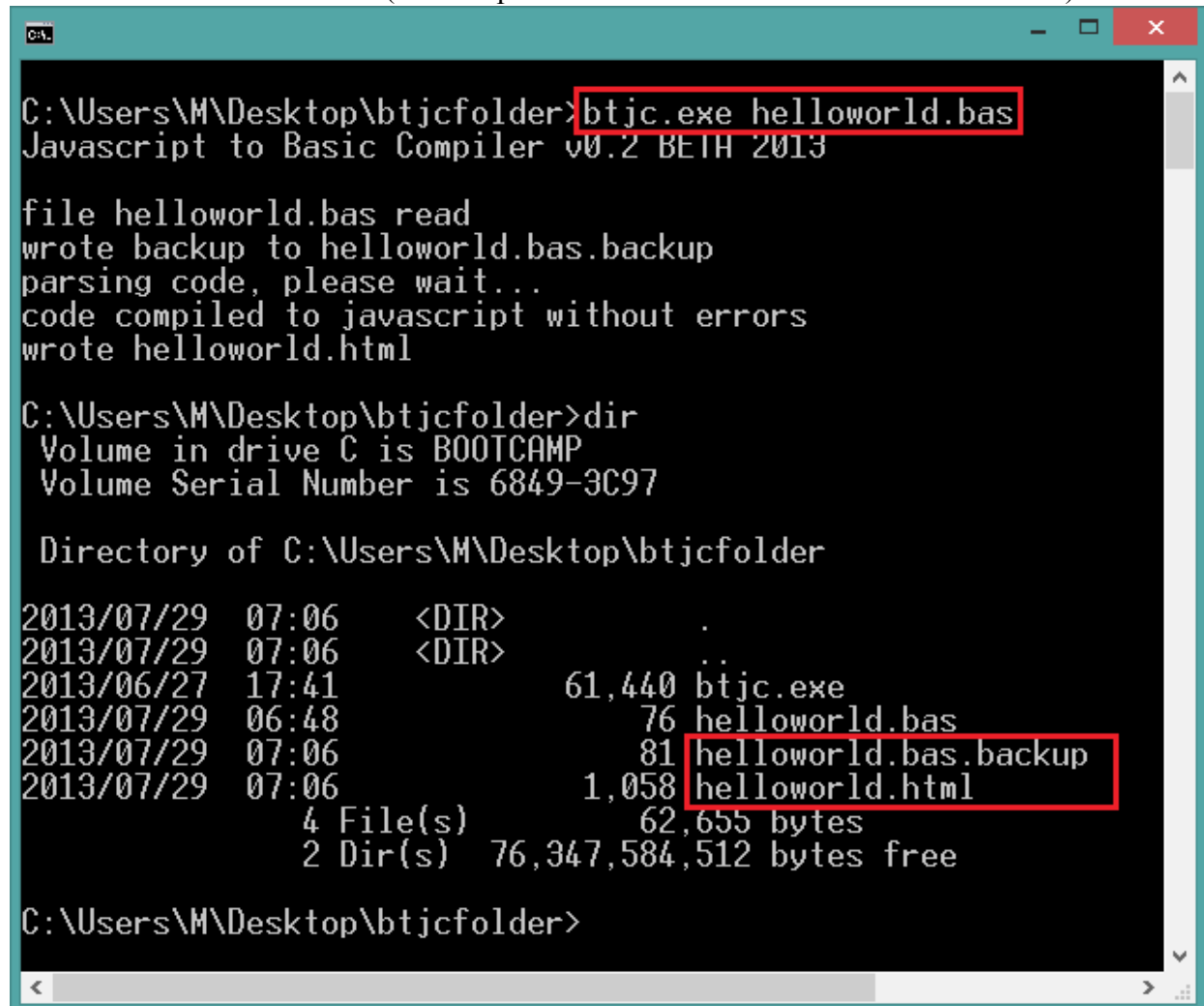
```
INCLUDE HELPER
```

```
NEWSCREEN "output"
```

```
SCREEN "output"
```

```
PRINT "Hello World"
```

5. Save the source code in the btjc folder as "helloworld.bas". Do this by clicking "File" then "Save As" inside notepad.
6. Go back to command prompt and type "btjc.exe helloworld.bas" to convert the bas file to a html javascript page. If the conversion went well, a message that says "code compiled to javascript without errors" should display and a file called "helloworld.html" should have been created in the folder. (A backup of the source should also have been created )



```
C:\Users\M\Desktop\btjcfolder>btjc.exe helloworld.bas
Javascript to Basic Compiler v0.2 BETH 2013

file helloworld.bas read
wrote backup to helloworld.bas.backup
parsing code, please wait...
code compiled to javascript without errors
wrote helloworld.html

C:\Users\M\Desktop\btjcfolder>dir
Volume in drive C is BOOTCAMP
Volume Serial Number is 6849-3C97

Directory of C:\Users\M\Desktop\btjcfolder

2013/07/29  07:06    <DIR>          .
2013/07/29  07:06    <DIR>          ..
2013/06/27  17:41         61,440 btjc.exe
2013/07/29  06:48           76 helloworld.bas
2013/07/29  07:06           81 helloworld.bas.backup
2013/07/29  07:06        1,058 helloworld.html
               4 File(s)          62,655 bytes
               2 Dir(s)  76,347,584,512 bytes free

C:\Users\M\Desktop\btjcfolder>
```

7. Open the "helloworld.html" file with the default browser. You should see a page that displays "Hello World".
8. Let's try something more interesting. Rewrite helloworld.bas so it looks like the following program (program on next page):

```
INCLUDE HELPER
```

```
NEWCANVAS "view", 800, 700
```

```
CALL SETCANVAS("view")
```

```
red = RGB(255,0,0)
```

```
green = RGB(0,255,0)
```

```
blue = RGB(0,0,255)
```

```
CALL LINE(200, 50, 150, 100)
```

```
CALL COLOR( blue )
```

```
CALL FILLRECT(30, 30, 80, 80)
```

```
CALL COLOR( green )
```

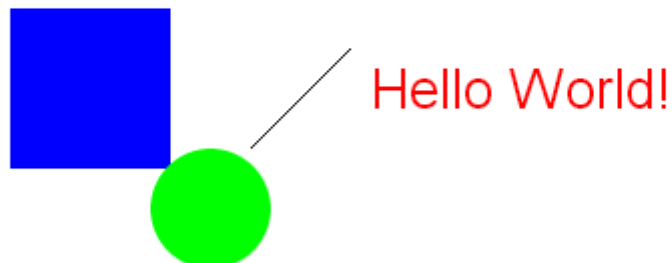
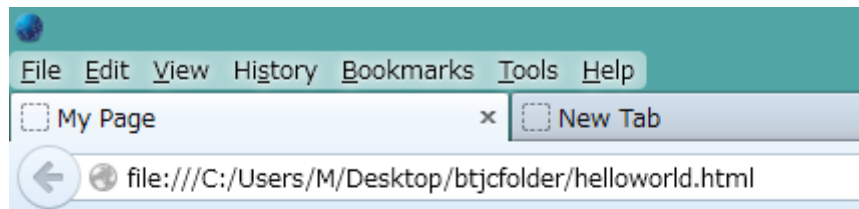
```
CALL FILLCIRCLE( 130, 130, 30)
```

```
CALL COLOR( red )
```

```
CALL SETFONT(20, "Arial")
```

```
CALL TEXT(210, 80, "Hello World!")
```

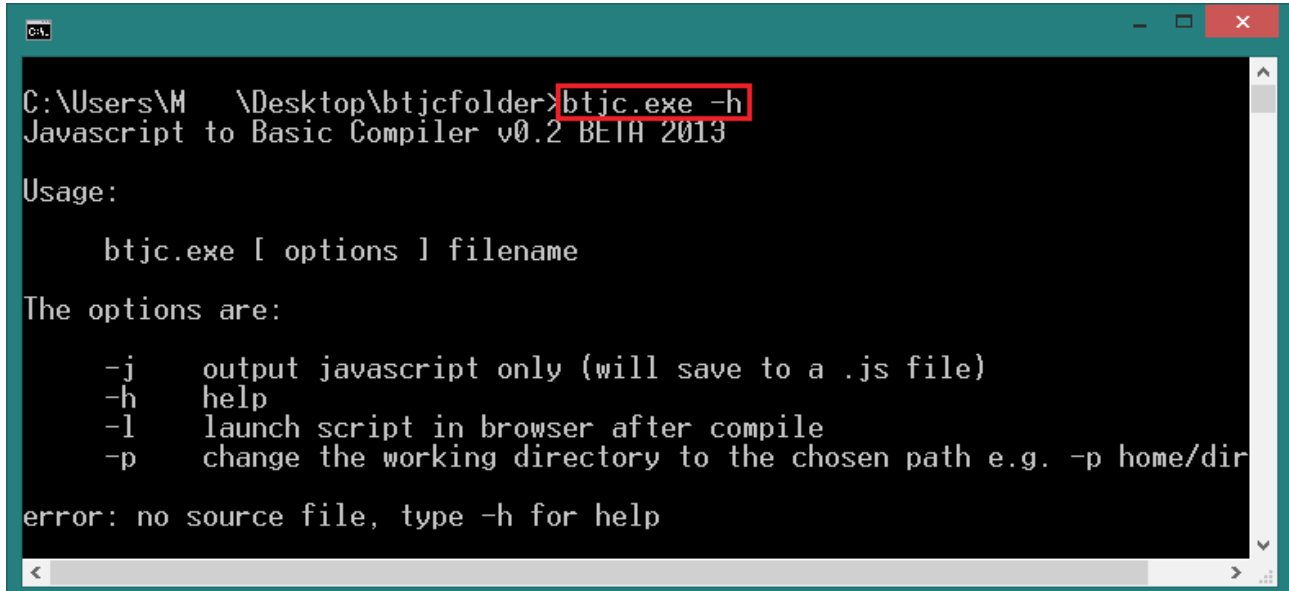
9. Repeat steps 5~7. If the program was typed in correctly the following image should display in the browser after converting the program:



## Chapter 1 - 2. Using Options

There are various options that can be selected when converting programs with BTJC.

Type “btjc.exe -h” into the command prompt to show all the options available:



```
C:\Users\M  \Desktop\btjcfolder>btjc.exe -h
Javascript to Basic Compiler v0.2 BETA 2013

Usage:
    btjc.exe [ options ] filename

The options are:
    -j    output javascript only (will save to a .js file)
    -h    help
    -l    launch script in browser after compile
    -p    change the working directory to the chosen path e.g. -p home/dir

error: no source file, type -h for help
```

The following is a detailed explanation of each option available in the latest version of BTJC:

### “-j” Option : Output Javascript File

Sometimes you may only want the Javascript output for the program. By using the “-j” option, instead of a .html file a .js will be created containing the Javascript output of the program.

E.g. `btjc.exe -j helloworld.bas`

This will output Javascript of the BASIC program in a file called helloworld.js.

### “-l” Option : Launch Browser after Conversion

Instead of having to open the .html file to run the program each time the program has been converted, you can use the -l option to automatically launch the default browser and run the html after the conversion finishes.

E.g. `btjc.exe -l helloworld.bas`

This will tell BTJC to open helloworld.html with the default browser after the conversion is finished.

## **“-p” Option : Change Working Directory**

When the “-p” option is selected the user can choose a path to a folder where the output files of the conversion will go. The path must be typed after the “-p” with a space.

E.g. `btjc.exe -p test/outputfolder helloworld.bas`

This will output the “helloworld.html” file to the folder at  
`C:\Users\M\Desktop\btjcfolder\test\outputfolder`

The “-p” option can be combined with the “-j” option to output a javascript file at the selected output folder.

E.g. `btjc.exe -p test/outputfolder -j helloworld.bas`

This will output the “helloworld.js” file to the folder at  
`C:\Users\M\Desktop\btjcfolder\test\outputfolder`

## Chapter 2

### *Single threaded and Synchronous Programming (Change title to about loops?)*

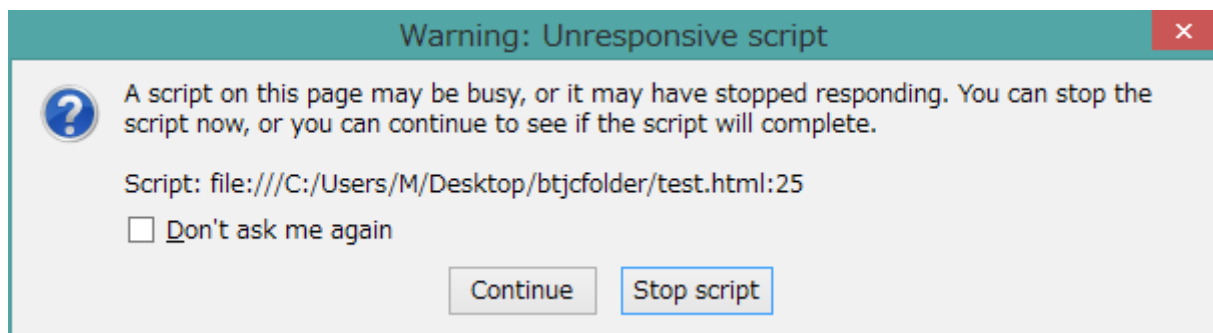
Javascript is single threaded and a synchronous programming language. Since the code written in BTJC will be converted to Javascript and run as Javascript the programmer must know how single threaded and synchronous programs are executed. The following is a list of some common pitfalls the programmer might face when programming with a single threaded synchronous language for the first time.

#### 1) Loops will freeze the browser until the code breaks out of the loop

The code below displays an image at 0,0 on the canvas. It will then go into an infinite loop and increment `x` and `y`. If `x` reaches its limit the image will be set back to 0,0. Normally a programmer familiar with a multi-threaded asynchronous language would expect the image to slide diagonally downwards across the screen over and over again creating a small animation.

```
INCLUDE HELPER                                : REM IMPORT THE HELPER LIBRARIES
NEWCANVAS "VIEW", 800, 600                    : REM CREATE A NEW "VIEW" CANVAS
CALL SETCANVAS("VIEW")                       : REM SET THE CURRENT CANVAS AS "VIEW"
MYIMAGE = LOADIMAGE("image.png")              : REM LOAD AN IMAGE AND STORE IT IN VARIABLE MYIMAGE
X = 0
Y = 0
DO
    CALL CLEARRECT(0,0,8,6)                   : REM CLEAR THE CANVAS
    CALL IMAGE(MYIMAGE, X, Y)                 : REM DISPLAY THE IMAGE AT X,Y ON THE CURRENT CANVAS
    X = X + 1
    Y = Y + 1
    IF X >= 800 : X = 0 : Y = 0 : ENDIF
LOOP
```

However, compiling and running the program will result in the browser freezing and an error message might appear similar to the following image:



How can one avoid the browser from freezing when using infinite or time consuming loops?

In JavaScript the answer is to use the *setInterval* function to call a function containing the loop code repeatedly. In BTJC there is the easy to use REPEAT command:

*Usage:*

**REPEAT** *function\_name* **EVERY** *n* **WITH** *variable*

*function\_name*: the function containing the code to repeat

*n*: amount of milliseconds to delay after each call

*variable*: variable to store the ID of the repeating instance. Useful for managing multiple repeat instances, or when you want to stop the function from repeating by calling the STOPREPEAT(*variable*) function.

Using the REPEAT command the first program can be rewritten as the following:

```
INCLUDE HELPER                : REM IMPORT THE HELPER LIBRARIES
NEWCANVAS "VIEW", 800, 600    : REM CREATE A NEW "VIEW" CANVAS
CALL SETCANVAS("VIEW")       : REM SET THE CURRENT CANVAS AS "VIEW"
GLOBAL MYIMAGE = LOADIMAGE("image.png") : REM LOAD AN IMAGE AND STORE IT IN VARIABLE MYIMAGE
GLOBAL X = 0
GLOBAL Y = 0
GLOBAL ID_MYLOOP = null
REPEAT MYLOOP EVERY 16 WITH ID_MYLOOP : REM REPEAT MYLOOP() EVERY 16 MILLISECONDS

FUNCTION MYLOOP()
    CALL CLEARRECT(0,0,8,6)    : REM CLEAR THE CANVAS
    CALL IMAGE(MYIMAGE, X, Y)  : REM DISPLAY THE IMAGE AT X,Y ON THE CURRENT CANVAS
    GLOBAL X = X + 1
    GLOBAL Y = Y + 1
    IF X >= 800
        GLOBAL X = 0
        GLOBAL Y = 0
    ENDIF
ENDFUNCTION
```

When the program is compiled and run the image animates correctly without freezing the browser.