The following guide provides an outline of the steps required to add additional components to the **PEBBLE** breadboard layout application.

I stress that you MUST always make a back-up of the entire PEBBLE folders before starting to edit the files used in the PEBBLE application.

Relatively simple things such as a space inserted or a missing brace character (i.e. { or }) may cause the new component not to be displayed, some sub-function to stop for ALL components (e.g. the component edit windows will not appear) or worse, sometimes may cause the entire program to stop working (e.g. even resistors and wires will not show on the main menu).
It can be quite hard in the long program files to spot exactly what the problem is.

Please note that I am not a 'html' or Javascript 'guru' myself and have learnt just enough through working on PEBBLE albeit with a lot or "Googling" on java website to work out what to do. Sometimes I even achieved functions through experimentation that the 'gurus on websites stated was not possible/easy.

The caveat here is that this guide is intended to help folks add their own new components, but I will not be responsible if you corrupt the program and lose your work for new components.

1. You will need to create a number of new images for the new component.
   Keep in mind the 27 pixel grid scheme used for the breadboards and strip/protoboards.
   You may find it easier to copy an existing image with some pins as a starting point if unsure.

   The number of images depends upon the component and how may orientations it is to be displayed in. As a guide:

   (a) An IC and has two possible orientations and therefore needs two images 180 rotated to each other.
   (b) Capacitors, diodes, transistors and switches have 4 possible orientations and thus need 4 images each rotated through 90 degrees.
   (c) Terminal strips and related parts, and the "floating" (i.e. non grid locked) components have two orientations and thus typically need 2 images but, in some cases (such as simple headers and corresponding sockets) where the rotated images is identical and it is possible to direct the program to use the same image for both views.
   (d) Miscellaneous items (under the LDR icon) only have one orientation available.

   The wire and resistor groups are special ad are drawn "on-the-fly" for each image shown. Do not try to add new component types into these groups. The 1-Span wire links are also a special set although using images and it is recommended that you do not try to add anything other than 1-span wire links into that group.

   The images need to be prepared in a drawing/painting program that will allow you to define some parts of the image area as transparent. I use the old JASC Paintshop Pro V9 (which will still work with Windows10). If you look at any of the existing component images in such a program I have used the colour pink consistently for transparent areas (since you do not see too many real pink coloured components).

   The images need to be saved as .gif type files to work in PEBBLE.
   See also section 3 regarding numbers as a suffix to the image name.

2. Having decided in which group you will be using the new component (e.g. IC's, terminals, capacitors, miscellaneous, or one of the other groups) you need to edit the html files in the main PEBBLE folder.
   The same changes will need to be made in the FF_for_FF and the P_for_IE.htlm files as follows:
   (a) Open the file in Notepad or a similar text file editor, but not MSWord or similar which will insert hidden characters and prevent the program working).
   (b) Scroll down to the section of the html file for the component/device group under which you wish to add the new items.
   The first group is roughly 25% of the way down the document. In order the sequence is:

Component Type In order defined	Commencing at line Starting with	Approx. distance down the document
IC	<div id="ic_dip_dialog"	25%
LED	<div id="led_dialog"	30%
Transistor	<div id="transistor_dialog"	33%
Diodes	<div id="diode_dialog"	38%
Capacitors	<div id="capacitor_dialog"	43%
Terminals	<div id="terminal_dialog"	47%
Wires – do not add to this component group		
Switches	<div id="switch_dialog"	57%
DIP Switches	<div id="dipsw_dialog"	63%
Miscellaneous (Under LDR icon)	<div id="miscell_dialog"	66%
The FourD modules – recommend only add new FourD devices here		
Strip & Proto Board Cuts, edits, etc	<div id="protobrd_dialog"	75%
1-Span wire links	<div id="wlinks_dialog"	78%
Notes (floating)	<div id="note_dialog"	82%
Floating Parts (floating bolt icon)	<div id="slider_dialog"	85%
Resistors– do not add to this component group		

(c) Within the desired group scroll down to the device selection part of the program. This will look similar to the following (example from the miscellaneous group)

```
<option value="1">
    <script language="javascript">document.writeln(_("LDR"));</script>
    </option>
<option value="2">
    <script language="javascript">document.writeln(_("Thermistor"));</script>
    </option>
<option value="3">
    <script language="javascript">document.writeln(_("Resonator"));</script>
    </option>
<option value="4">
    <script language="javascript">document.writeln(_("Clock Crystal"));</script>
    </option>
```

There may already be many more options in the list. Insert a new 3-line entry for your new options. You can insert the new device definition anywhere in the list, but the option value (i.e. the “1”, “2”, “3” . . . part), must be unique – typically the next higher number.

If, for example, you wish to add a new colour of LED or a new jumbo sized range of capacitor, the same applies.
For capacitors the size options are in the form:

```
<option value="3">
    <script language="javascript">document.writeln(_("Large"));</script>
    </option>    // Span 4
```

For IC’s a more descriptive name has been used such as:

```
    <option value="DIP4">4 Pin DIP</option>
```

or

```
    <option value="PICAXE18M2">PICAXE 18M2</option>
```

or

```
    <option value="ULN2803">ULN2803</option>
```

In the examples above the **reddish** colour represents the option "name" used within the program and the **blue** colour represents the text shown in the PEBBLE component drop-down menus.

(d) Now save the file keeping it in the .html file type/extension.

3. Next go into the "javascript" folder/directory within the main PEBBLE folder.

   Open the file named "imagedata.js" using Notepad or other text editor (you may not see the ".js" part if your OS hides the file extensions).

   Scroll down to the section for the component types you are adding. Most entries are clear as to which is the right area. You will need to insert two new lines for each new image that you have created.

   For a new capacitor these two lines will be in the format:

```
var cap_534 = new Image();
cap_534.src = "images/cap_534.gif";
```

   For a new IC with 2 images the lines will be in the format:

```
var DIP8_1  = new Image();
var DIP8_2  = new Image();
DIP8_1.src  = "images/dip8_1.gif";
DIP8_2.src  = "images/dip8_2.gif";
```

   For the Miscellaneous Group a device has 1 image and the 2 lines will be in the format:

```
var misc_46 = new Image();
misc_46.src = "images/misc_46.gif";
```

   The first line is in effect registering the new image for the PEBBLE application so the program knows it can exist.

   The second line type (i.e. the one starting xxx.**src** = . . .
   informs the program where to find the corresponding image. These are all saved in the "images" folder so the source definition commences with "images/" followed by our new image name.

   As highlighted above by the green and orange colours, the name used for registration and the name for the .src part must be the same.

   It is not compulsory that the actual .gif file name is the same as the registered name but I have typically (but not always done so) for each of keeping track on the images.

   Now for the complex part – those numbers after the underscore character (_).

   For the Miscellaneous, fourD and Protoboard groups they are simply the next sequential device and the number part is the same as the option number used in the .html file as discussed in part 2 above.

   But for other devices they relate to the size, colour, and/or orientation of the components.

   The following table endeavours to show how these numbers are created/structured;

DEVICE	**PREFIX**	**1st DIGIT FIELD**	**2nd DIGIT FIELD**	**3rd DIGIT FIELD**
IC's	<xxx>_ From html file menu See section 2(c)	Orientation (1 digit)	none	none
Capacitor	cap_	Package type (1 digit)	Size (1 digit)	Orientation (1 digit)
Diode	diode_	Package type (1 digit)	Size (1 digit)	Orientation (1 digit)
LEDs	led_	Colour (1 digit)	Orientation (1 digit)	none
Pushbutton & Toggle Switches	switch_	Package type (1 digit)	Colour (1 digit)	Orientation (1 digit)
DIP & Hex rotary Switches	Dipsw_	Size (1 digit)	Orientation (1 digit)	none
Terminal Devices	terminal_	Package type (1 or 2 digits)	Size (1 digit)	Orientation (1 digit)
Transistors	transistor_	Package type (1 digit)	Orientation (1 digit)	none
Miscellaneous (LRD icon)	misc_	Package type (1-2 digit sequential)	none	none
FourD devices	fourd_	Package type (1 digit sequential)	none	none
Strip & Proto Brd Cuts and Edits	protobrd_	Edit/Item type (1-2 digit sequential)	none	none
1-Span Wire Link	protowire_	Orientation (1 digit)	Package type (1 digit sequential)	
'Floating' Parts (bolt icon)	floating_	Orientation (1 digit)	Package type (1 digit sequential)	
"Floating" Notes (yellow)	NOTEPAD_	Size (1 digit)	Orientation (1 digit)	none

4. Next you may need to provide some javascript program lines to align your new component/device with the board grid. This may be a bit of trial and error to have the PEBBLE program display the component exactly where you want it relative to the breadboard or strip/proto board holes.

   For a standard IC no edits should be required but for specials such as mounted on a break-out/adaptor board then some offset adjustment is required. Most other components other than floating components/notes will nearly always need some definition of offset values.

   The offset values are relative to the top left corner of the image, whether that point is transparent or part of the device.

   Horizontal offset adjustments use the 'x' variable and vertical offset adjustments use the 'y' variable. Positive values for x and y will move the image right or down respectively and negative values move the image left or up respectively.

   The steps are:

   (a) Open the file "application.js" (located in the PEBBLE/javascript folder) using Notepad or a similar text editor.

   (b) At the top of the file is the function:

   ```
   function getDropOffset(compObject)
   ```

scroll down to the CASE for the new component type you are adding. For example:

case "Miscell" :

or

case "ProtoBrd" :

Under these sections there may be further CASE statements for the package type or the orientation, etc. These will be a number in quotes (e.g. “1”, “2”, “3”, . . etc.)

(c) Create a new entry under the package type or under each orientation section for the new component.

At each location in the file, this will be in the format:

```
 case "1" :
          x = -9;
           y = 10;
          break;
```

If you have several identical entries or your one new entry is identical for both x and y values to an existing entry, these can be groups such as:

```
case "9" :
case "10" :
case "11" :
          x = -2;
           y = -1;
          break;
```

Once you have made these entries save the javascript file (application.js).

As mentioned earlier, will likely need to make the initial entry save the file and run a new instance of PEBBLE and check that the positioning is correct.
If not, determine which direction the adjustment is required and edit the x and/or y values you previously entered into the application.js file.

5. Finally, if the component/device type allows text entry in the component propertied edit windows and especially if you expect text to be added as a label you will likely need to position the text relative to the component image and maybe set a different colour so it will stand out clearly.

   To do this:

   (a) In the PEBBLE/javascript folder, open the “utils.j” file using Notepad or a similar text editor.

   Scroll (or search) about two-thrids of the way down to the function:

   function applyComponentValuesToView(componentRef, componentDIV)

   scroll further down to the CASE statement for the component type you are adding. For example:

   case "Miscell" :

   or

   case "ProtoBrd" :

(b) Then you can add a new entry to provide the positioning, text size and text colour. This will be in the format:

```
case "8" :
    innerDIV.innerHTML = "<span style='position:absolute;top:60px;left:50px;
    color:white;'>" + componentRef.text + "</span>";
    break;
```

or

```
case "37" :
case "38" :
    innerDIV.innerHTML = "<span style='position:absolute;top:43px;left:205px;
    color:white;font-weight:bold;'>" + componentRef.text + "</span>";
    break;
```

The top: value is the number of pixels down the top of the image (your .gif file). That includes any transparent area above the component proper.

The left: value is the number of pixels from the left edge of the image.

Note that there should be no spaces as you will find in the actual util.js file.

Again you may need separate entries for components with multiple orientations.

Finally, I reiterate that you MUST always make a back-up before starting to edit the files used in the PEBBLE application.