**Trienz**

**What is Trienz?**

Trienz is a way of organizing a collection of things [CoT]. The system helps to organize a CoT in such a way that the relationships among the items in the CoT can be easily known. In fact, the ability to make it easy to know the relationships among the items in a CoT, is the reason why Trienz was created. In other words, if you feel a strong need to not only organize a CoT, but to also do it in a way that shows the relationships among the items in it, then you should really consider Trienz.

However, when I say Trienz is good at helping show the relationships among the items in a CoT, I’m not saying the system is good at showing all kinds of relationships. Trienz is rather meant for situations where the items needed to be organized are: *stans*, *dooms*, and *goons*.

**Stans:** A stan is a normal item in a collection. A stan is an item that isn’t a doom or a goon.

**Dooms:** When a stan incorporates another item, the item it incorporates is called a doom. Take for instance, if a webpage [HTML file] contains a picture, the JPG file of the picture can be called a doom.

It is however worthwhile stating that while the example given is about computer files, Trienz is not limited to organizing only things related to computers. While the need to organize software source codes, in a better way, is what gave birth to this system, its use isn’t limited to computers only. You should rather feel free to use the system wherever you deem it fit.

Also, not only stans can have dooms. Dooms can also have dooms, and goons can also have dooms.

**Goons:** A goon is like a doom. But unlike a doom which is incorporated by a single item, a goon is an item incorporated by multiple items. For instance, if multiple webpages [HTML files], make use of a common CSS file, the CSS file should be called a goon, not a doom.

And just like a doom, a goon can also be incorporated by a stan, a doom, and a goon.

**Trienz in action**

To help deepen your understanding of Trienz, we’d be taking a look at how an unorganized CoT can be organized, using Trienz.

Imagine we have a project that involves creating a website, using HTML, CSS, and PHP. Imagine the website is for a Nigerian university. And the website would make certain information publicly available to students of the school, including those from other countries. Also, let’s assume that after creating the site, the following files happen to be what power the site:

**File 01 [ index.php ]:** The file of the webpage serving as homepage of the website.

**File 02 [ index-some-css.css ]:** The CSS file containing styling instructions that would be used for “File 01”.

**File 03 [ index-some-image.jpg ]:** An image was used on the homepage, and this is the file of the image.

**File 04 [ tuition.php ]:** The file of the webpage providing info about the tuition fee of the school.

**File 05 [ tuition-some-css.css ]:** The CSS file containing styling instructions that would be used for “File 04”.

**File 06 [ calendar.php ]:** The file of the webpage providing info about the current academic session’s calendar.

**File 07 [ calendar-some-css.css ]:** A CSS file containing styling instructions that would be used for “File 06”.

**File 08 [ calendar-some-css-2.css ]:** Another file like “File 07”.

**File 09 [ calendar-some-css-from-developer-x.css ]:** Another file like “File 07”, but unlike “File 07” and “File 08”, this file was created by some other developer.

**File 10 [ calendar-some-css-from-developer-x-2.css ]:** Another file like “File 07”, but unlike “File 07” and “File 08”, this file was created by some other developer.

**File 11 [ international-student-additional-tuition.php ]:** The file of the webpage providing info about additional fees international students would be required to pay.

**File 12 [ international-student-additional-tuition-css.css ]:** The CSS file containing styling instructions that would be used for “File 11”.

**File 13 [ international-student-additional-requirement.php ]:** The file of the webpage providing info about additional requirements needed by international students, to graduate.

**File 14 [ international-student-additional-requirement-css.css ]:** The CSS file containing styling instructions that would be used for “File 13”.

**File 15 [ international-student-from-europe-additional-tuition.php ]:** The school not only charges international students more money, it further charges European students even more. And this is the file of the webpage providing info about the additional fees.

**File 16 [ international-student-from-europe-additional-tuition-some-css.css ]:** The CSS file containing styling instructions that would be used for “File 15”.

**File 17 [ international-student-from-europe-additional-requirement.php ]:** The school not only demands more requirements from international students, to graduate, it further demands even more requirements from European students. And this is the file of the webpage providing more info about the additional requirements.

**File 18 [ international-student-from-europe-additional-requirement-some-css.css ]:** The CSS file containing styling instructions that would be used for “File 17”.

**File 19 [ common-css.css ]:** The file containing styling instruction used by all files of the webpages.

**File 20 [ common-php.php ]:** The file containing PHP functions used by all files of the webpages.

**File 21 [ global-php.php ]:** A file containing PHP functions used by: all files of the webpages as well as “File 20”.

**File 22 [ global-php-from-developer-x.php ]:** Another file like “File 21”. But unlike “File 21” which was created by the developer of the site, this file was created by some other developer.

**File 23 [ global-php-from-developer-x-2.php ]:** Another file like “File 21”. But unlike “File 21” which was created by the developer of the site, this file was created by some other developer.

**Organizing using Trienz**

Now, if we are to organize this unorganized CoT, using Trienz, here’s what we’re going to do:

***Note:*** *In Trienz, we think of the items we’re trying to organize as physical items, even thought they might not be physical. We also think as if the place we’re organizing the items in, is a physical room. So when you start to see things like “divide the room into two”, “use a string to attach item X to item Y”, etc, do not get confused.*

**Step 01**

Bring files 01, 04, 06, 11, 15 and 17 into the room. These files would be stans in the room.

**Step 02**

Get a rope, and tie one of its end to file 01. A rope tied to a stan is called a *pro rope*, so take note of that term, as you might come across it again.

Afterwards, bring in file 02. Get another rope, and tie one end of the second rope to file 02. File 02 is of course a doom, if you look at it’s definition. And a rope tied to a doom, is called a *sec rope;* you should also take note of this term.

At this point, proceed with tieing the loose ends of the two ropes together. In other words, join File 01 and 02 together, by tieing together the loose ends of their ropes.

That’s still not all for step two.

Pick up file 03, and get a rope. Tie one end of the rope to the file. And proceeding with tieing the loose end of File 03’s rope to the junction where the ropes of files 01 and 02 meet.

**Step 03**

Get a rope, and tie one end of the rope to File 06.

Bring in File 07, get a rope, and tie one end of the second rope to it. Then tie the loose ends of the first and second ropes together, attaching file 07 to file 06.

Bring in File 08, get a rope, and tie one end of the third rope to it. Then tie the loose end of the third rope to the point where the first and second ropes meet, attaching files 07 and 08 to file 06.

Bring in File 09, get a rope, and tie one end of the fourth rope to it. Then tie the loose ends of the fourth rope to the point where the first, second, and third ropes meet, attaching files 07, 08 and 09 to file 06.

Bring in File 10, get a rope, and tie one end of the fifth rope to it. Then tie the loose ends of the fifth rope to the point where the first, second, third, and fourth ropes meet, attaching files 07, 08, 09, and 10 to file 06.

Now all dooms of file/stan 06 have been attached to it. But if you think about what we’ve done, you might notice files 09 and 10 shouldn’t be allowed to mix freely with the other CSS files, since files 09 and 10 were developed by some other developer [at least this is the way I feel.]. I feel it would make sense to put them in a separate container, then tie that container to file 06.

So if we’re to do something about this feel, I’d have to introduce you to a new term called *toom*.

A toom is just like the room we’re currently organizing things in. You can think of it as a smaller and lighter room we can magically create. However, note that I may call a toom a room. And when I do so, just understand I’m using that term *room* because I’m not concerned about the fact that it’s inside or smaller than the original room. I’m rather using the term because I’m currently seeing the toom as also a place where I can organize things, and nothing else.

So once you create the toom, untie files 09 and 10 from file 06. Also remove the ropes attached to the files, and throw them away. Then put files 09 and 10 in the toom. Afterwards, tie the toom to file 06, using a new rope.

Right now, I hope you can see how this seems to make more sense.

**Step 04**

Get a new rope, and tie one end of the rope to file 11. Bring in file 12. Pick up a second rope, and tie one end of the second rope to file 11. Then tie together the loose ends of the first rope and the second rope, attaching file 12 to file 11.

**Step 05**

Get a new rope, and tie one end of the rope to file 13. Bring in file 14. Pick up a second rope, and tie one end of the second rope to file 13. Then tie together the loose ends of the first rope and second rope, attaching file 14 to file 13.

**Step 06**

Get a new rope, and tie one end of the rope to file 15. Bring in file 16. Pick up a second rope, and tie one end of the second rope to file 15. Then tie together the loose ends of the first rope and second rope, attaching file 16 to file 15.

**Step 07**

Get a new rope, and tie one end of the rope to file 17. Bring in file 18. Pick up a second rope, and tie one end of the second rope to file 5. Then tie together the loose ends of the first rope and second rope, attaching file 18 to file 17.

**Wait a min**

I don’t know if you’ve noticed what has been happening so far. We’ve been attaching dooms to their stans. We’ve handled files 01 to 18, and it’s time to move on to the other files.

**Step 08**

So far, you can see that all the files we’ve dealt with are either stans or dooms. So how are we going to deal with dooms [since file 19 is a doom]? Are we also going to tie to them to the stans they’re related to? Oh wait, dooms are items related to more than one item, so which item are going to tie them to? Or are we going to connect them with multiple ropes, with the items they’re related to? Well, that we’d do is quite different.

Since a doom would be related to more than one item, it would be unfair to tie it to a specific item. So instead of doing that, what we’d do is to create a corner in the room. In Trienz, these corners are called salts. So when you see an item in a salt, you should automatically know that item is related to more than one other items in the room or in the tooms of the room.

So at this point, you should bring in file 19 and 20, and put them in the salt.

**Step 09**

I understand you might have already started to wonder, why didn’t we also bring in files 21, 22, and 23, and also put them in the salt. Well, if you look at the defintion of these files, you’d realize that file 19 depends on them. And as a result, it wouldn’t make much sense to put them in the same place as file 19. So instead of putting them in the same salt as file 19, we’d create a new salt, behind the first salt. Then we’d put files 21, 22, and 23 in the new salt.

So in short, whenever you see a doom behind another doom, it means the doom behind is a dependency of a doom directly ahead of it.

**Step 10**