**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 a 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 elements 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:** A doom is an item that is a part of another item, although physically separate from the item it is a part of. Take for instance, a JPG file which contains data of a picture displayed in a webpage [HTML file], can be considered a part of the HTML file, even though the HTML and the JPG file are two separate files. And in this case, the HTML file can be considered the stan, while the JPG can be considered the doom.

It is also worthwhile stating that while the example given above 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.

However, 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 would be a part of a single thing, a goon is an item which is part of 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 related to a doom. It can furthermore be related to 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. The website would make certain information publicly available to students of the school, including those from other countries. And 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 for 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 07”.

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

**File 09 [ calendar-some-css-from-developer-x.css ]:** A CSS file containing styling instructions that would be used for “File 06”, 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 ]:** A CSS file containing styling instructions that would be used for “File 06”, but unlike “File 07”, “File 08” and “File 09”, 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 12”.

**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 that providing more 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 containing PHP functions used by: all file of the webpage as well as “File 20”. 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 22”.

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 as the items we’re trying to organize as physical, even thought they might not be physical. We also think as if they 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**

Put file 1, 4, 6, 11, 15 and 17 in the room. These files would be stans in the room.

**Step 02**

Get a rope, and tie one of its end to file 1. A rope tied to a stan is called a *pro rope*.

Afterwards, bring in file 2. Get another rope, and tie one end of the second rope to file 2. File 2 would be a doom, so a rope tied to a doom, is called a *sec rope*.

At this point, proceed with tying the loose ends of the two ropes together. In other words, join File 1 and 2 together, by tying together the loose ends of their ropes.

That’s still not all for step two.

Pick up file 3, and get a rope. Tie one end of the rope to the file. In this case, the rope tied to file 3 is also a sec rope.

Proceeding with tying the loose end of File 3’s rope to the junction where the ropes of file 1 and file 2 meet.

**Step 03**

Get a rope, and tie one end of the rope to File 6.

Bring in File 7, 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, connecting file 7 with file 6.

Bring in File 8, get a rope, and tie one end of the third rope to it. Then tie the loose ends of the third rope to the point where the first and second rope meet, connecting files 7 and 8 with file 6.

Bring in File 9, 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, connecting files 7, 8 and 9 with file 6.

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, and third ropes meet, connecting files 7, 8, 9, and 10 with file 6.

Now all dooms of stan 6 have been attached to it, but if you think about what we’ve done, you might feel like file 9 and 10 shouldn’t be allowed to mix freely with the other CSS files, since file 9 and 10 were developed by some other developer [at least this is the way I would feel.]. I feel it would rather make sense to put them in a separate container, then tie that container to file 6.

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 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 room as also a place where I can organize things.

So once you create the toom, untie file 9 and 10 from file 6. Also remove the ropes attached to the files, and throw them away. Then put file 9 and 10 in the toom. Afterwards, tie the toom to file 6, using a new rope.

Right hope, 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 second rope, joining file 11 and 12 together.

**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, joining file 13 and 14 together.

**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, joining file 15 and 16 together.

**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, joining file 17 and 18 together.

**Step 08**

At this point, we’ve organized files 01 to 18. Now let’s move on to file 19.

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**