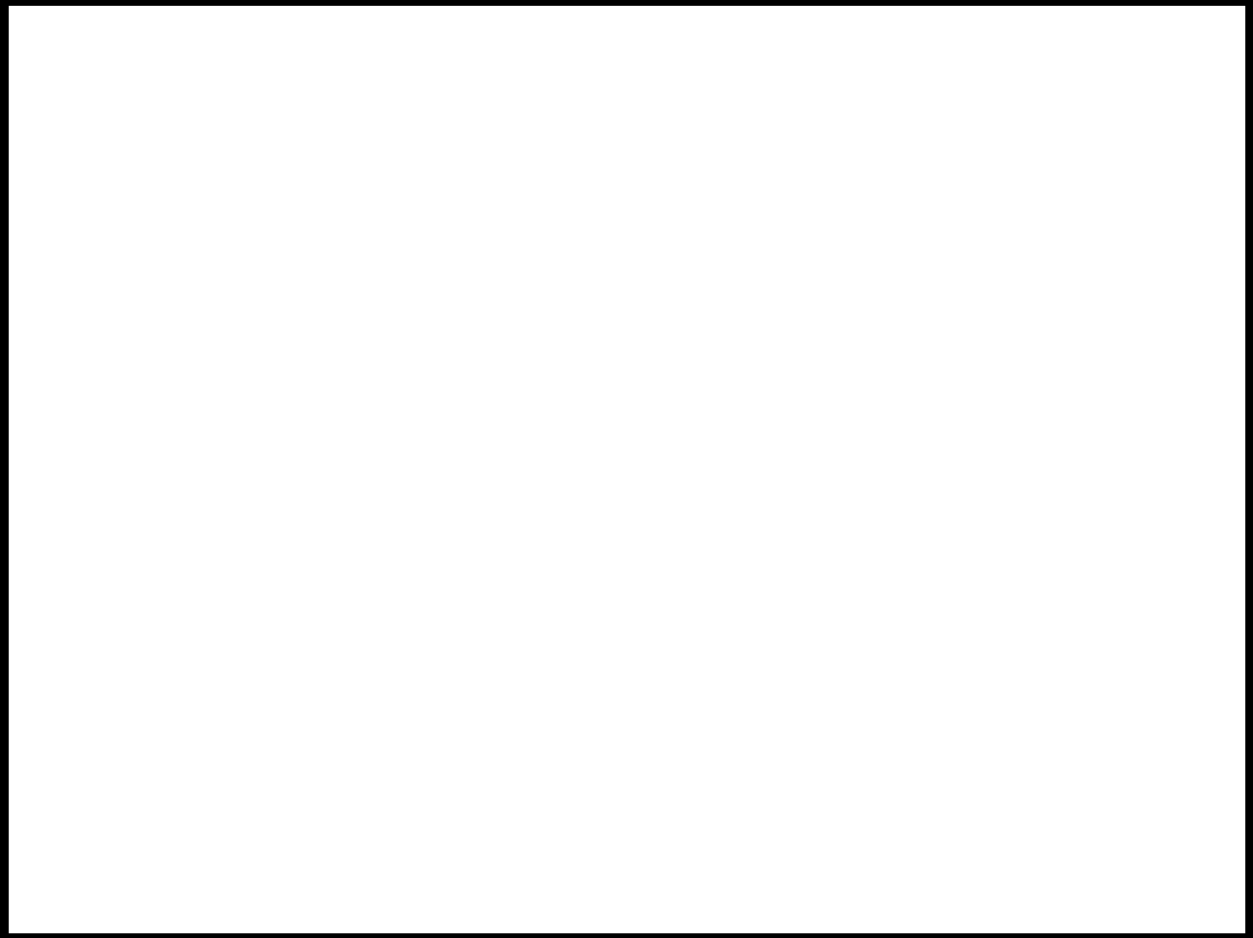
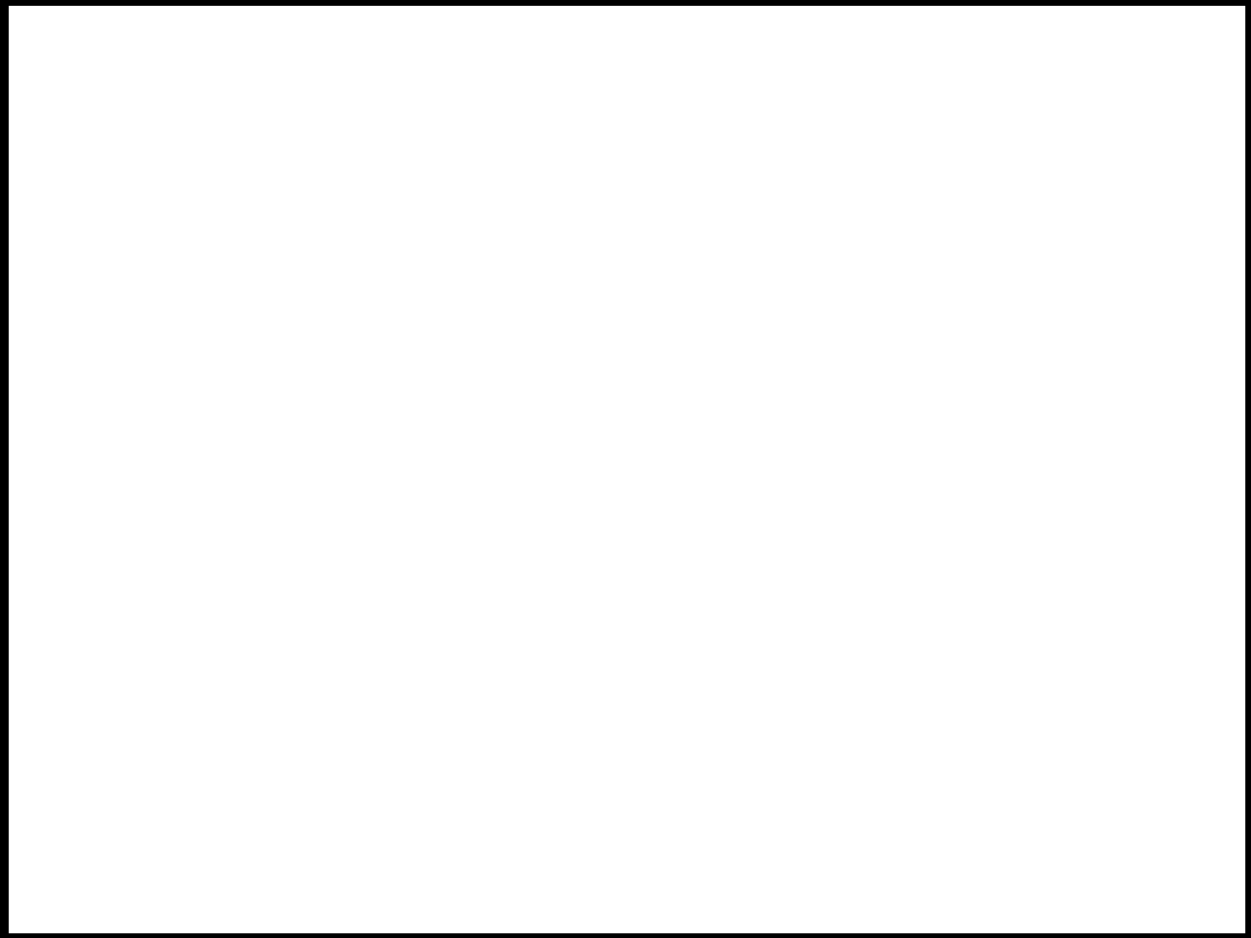
Design Guidelines

by Hai Tran

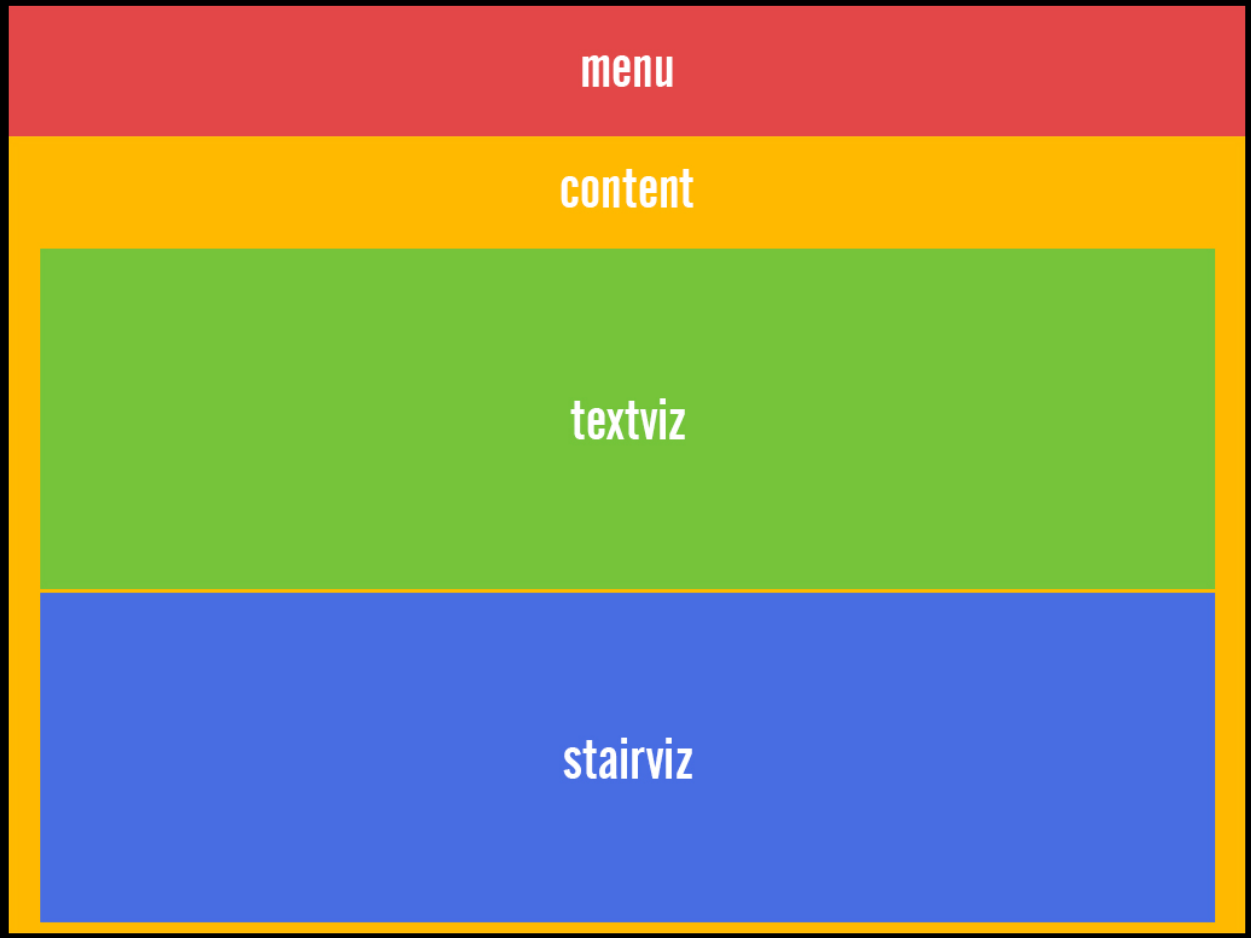


This is our screen.

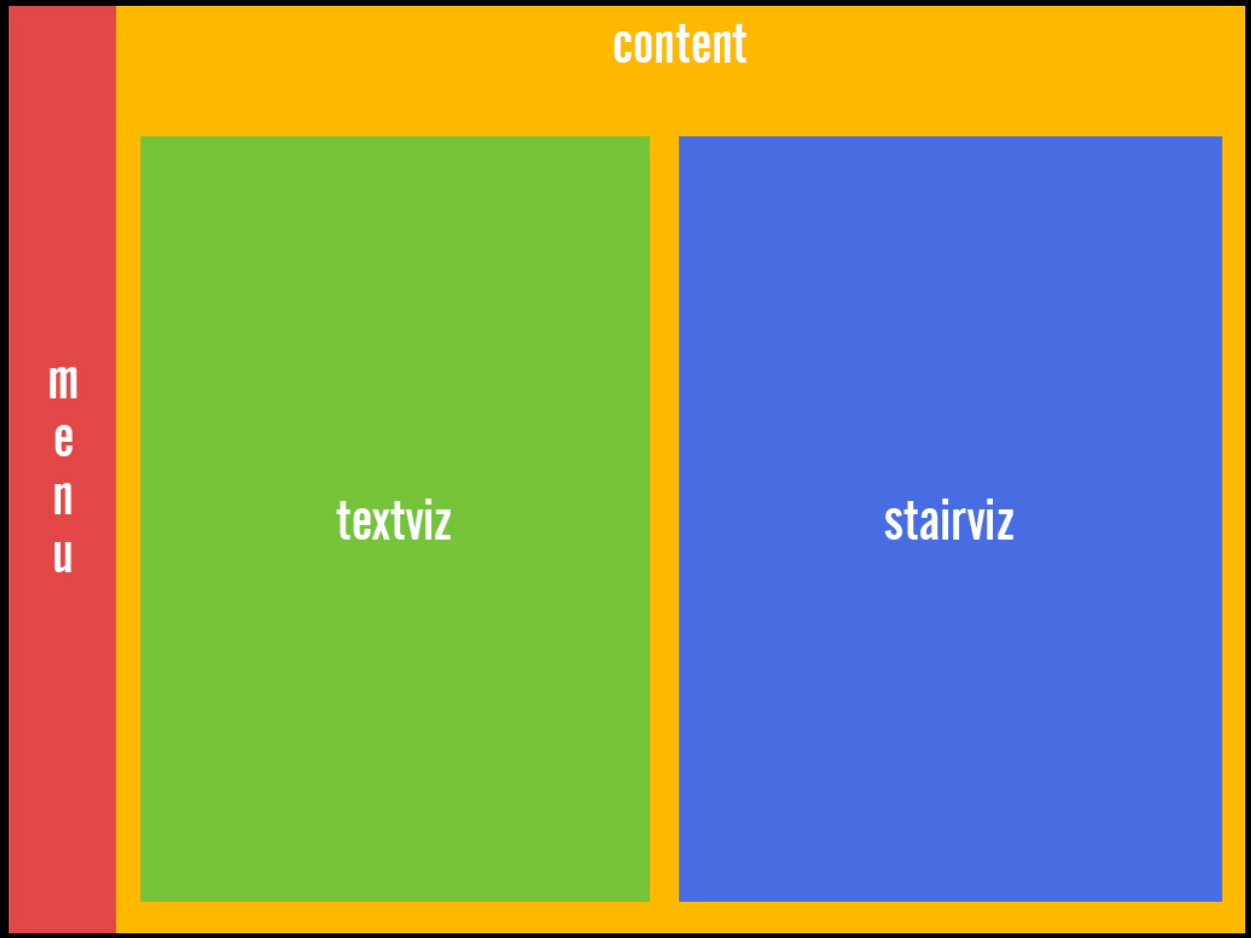
I'm drawing it at 1024 x 768, but the size shouldn't matter much.



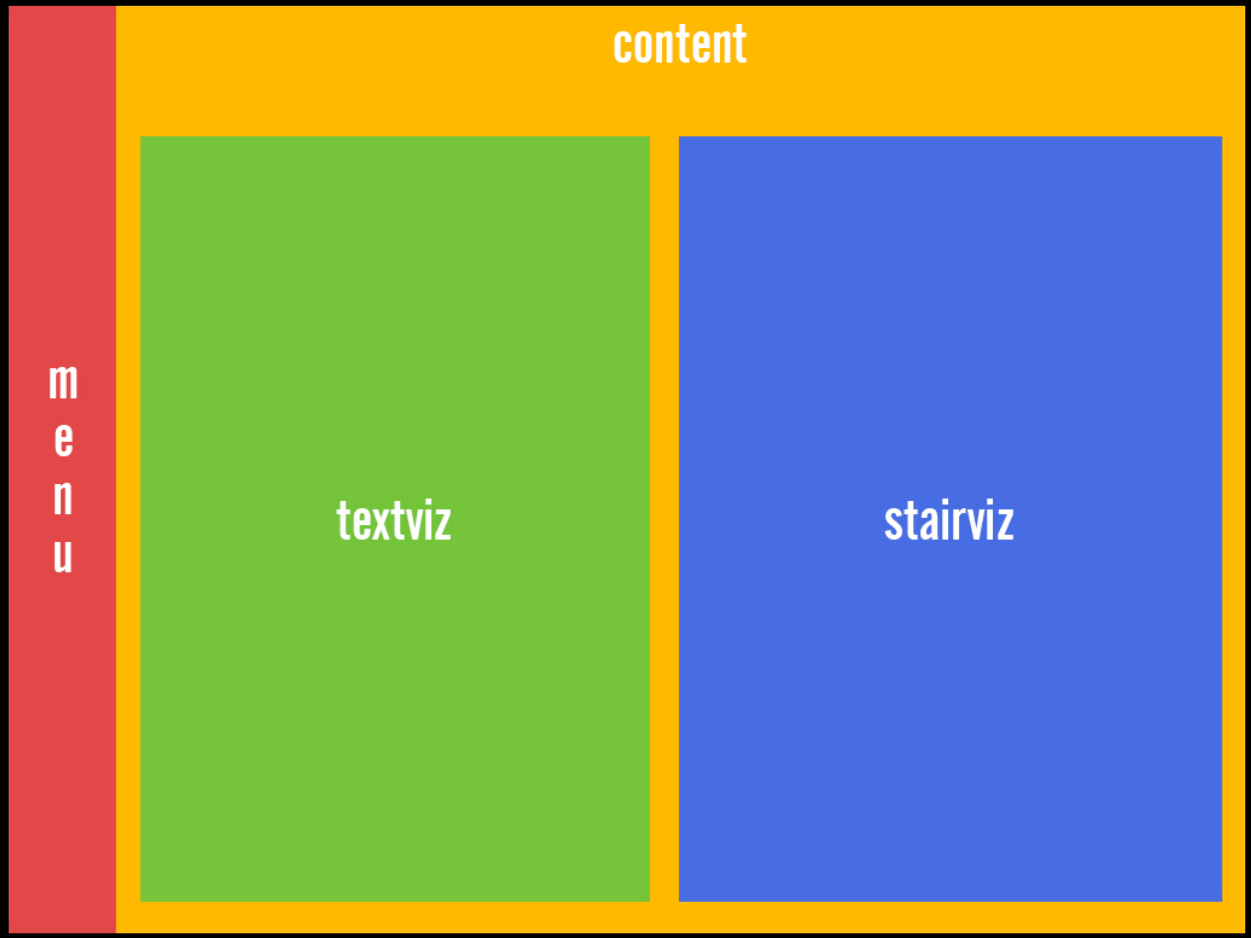
Our page will be divided into 2 main components: menu and content.
content is again divided into 2 smaller components: textviz and stairviz.
Size and position of the components should be defined using CSS. And these values should be responsive, not fixed.
If we follow this rule, we can organize them like this...



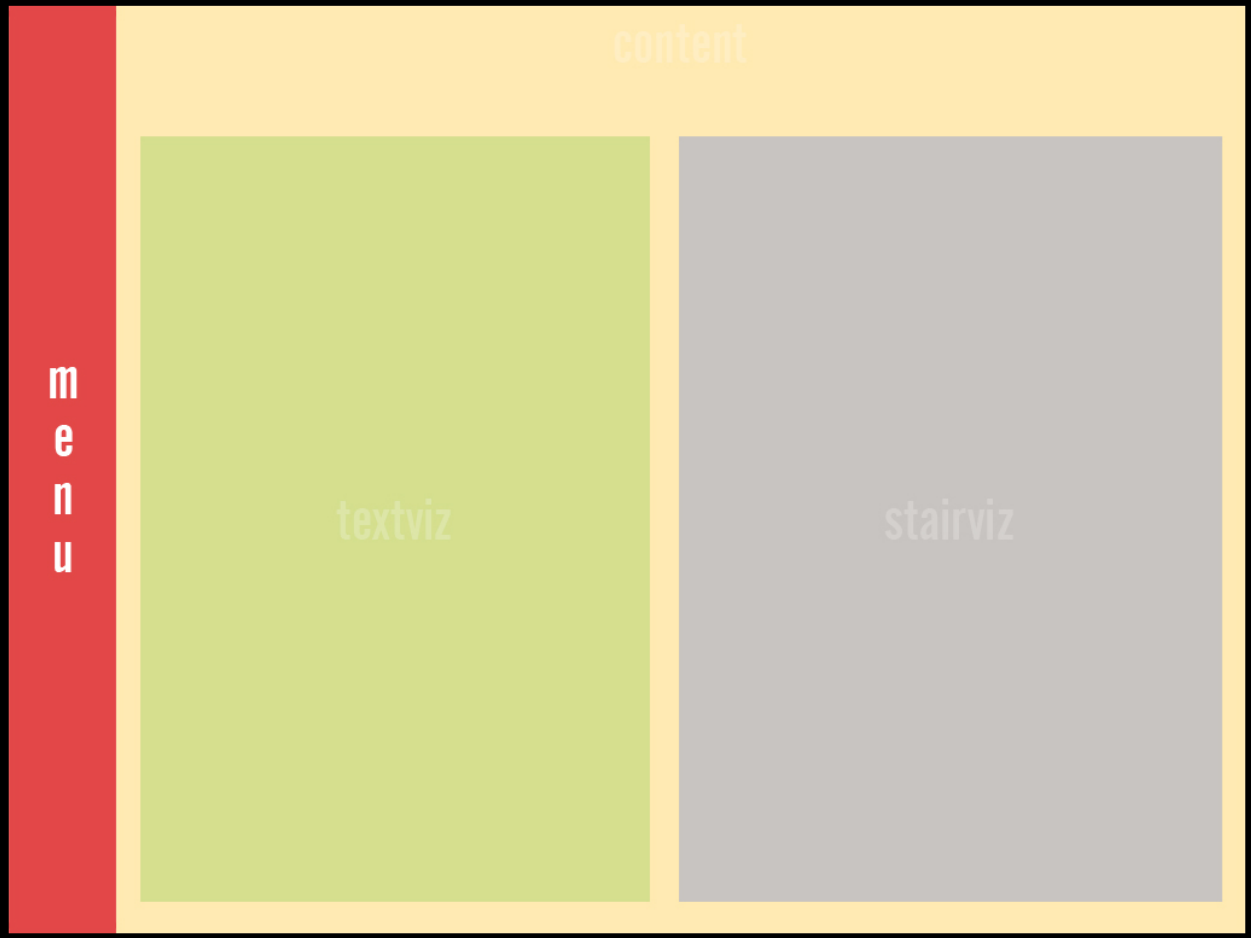
... or like this.
CSS helps us make this immediate switch by editing only a few lines of code.
Since I don't like looking at paragraphs that have width much larger than height, I'll stick with this second layout in this guidelines. If I have time, I'll add drawings for the first layout.



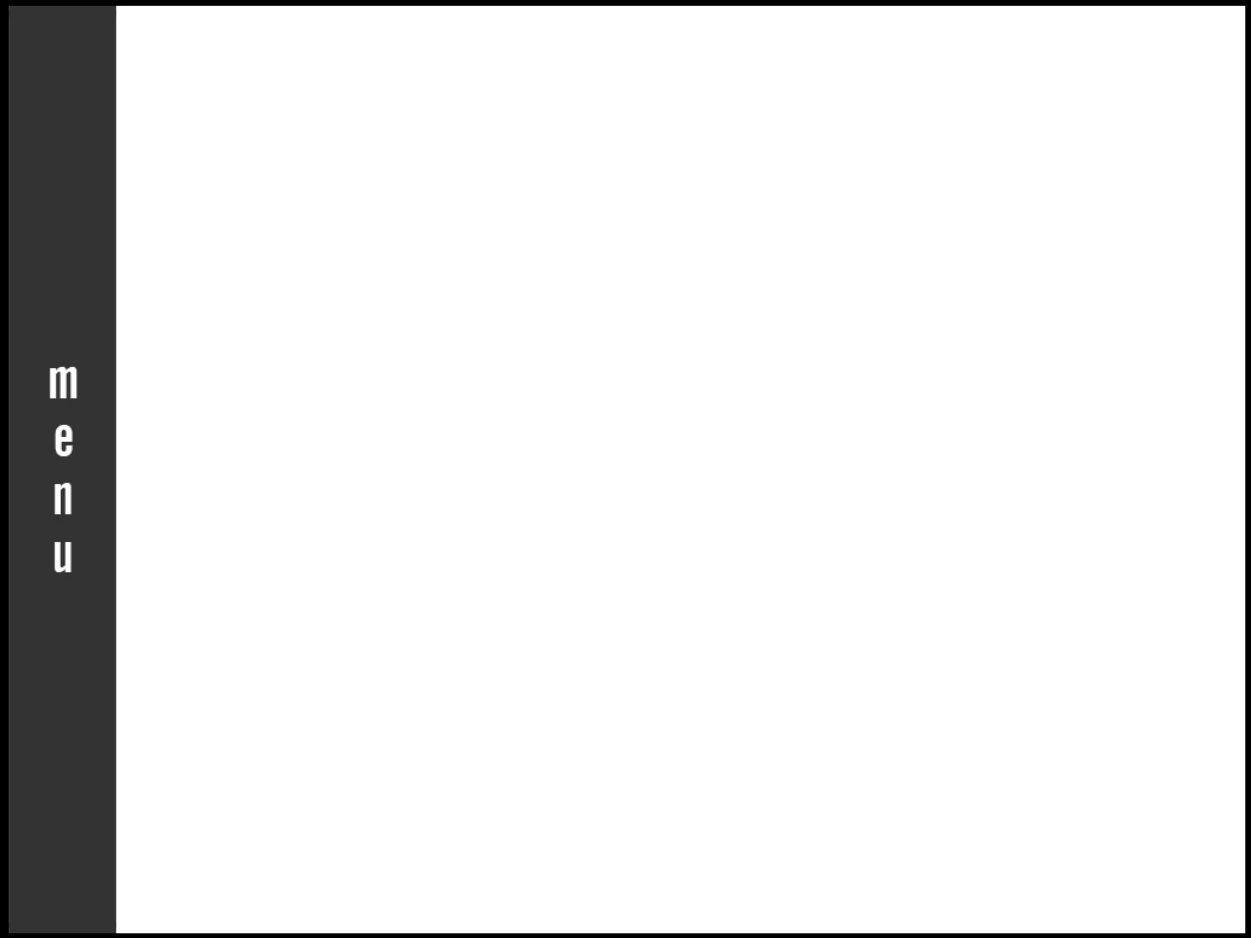
Now let's go into details and see what each container has.



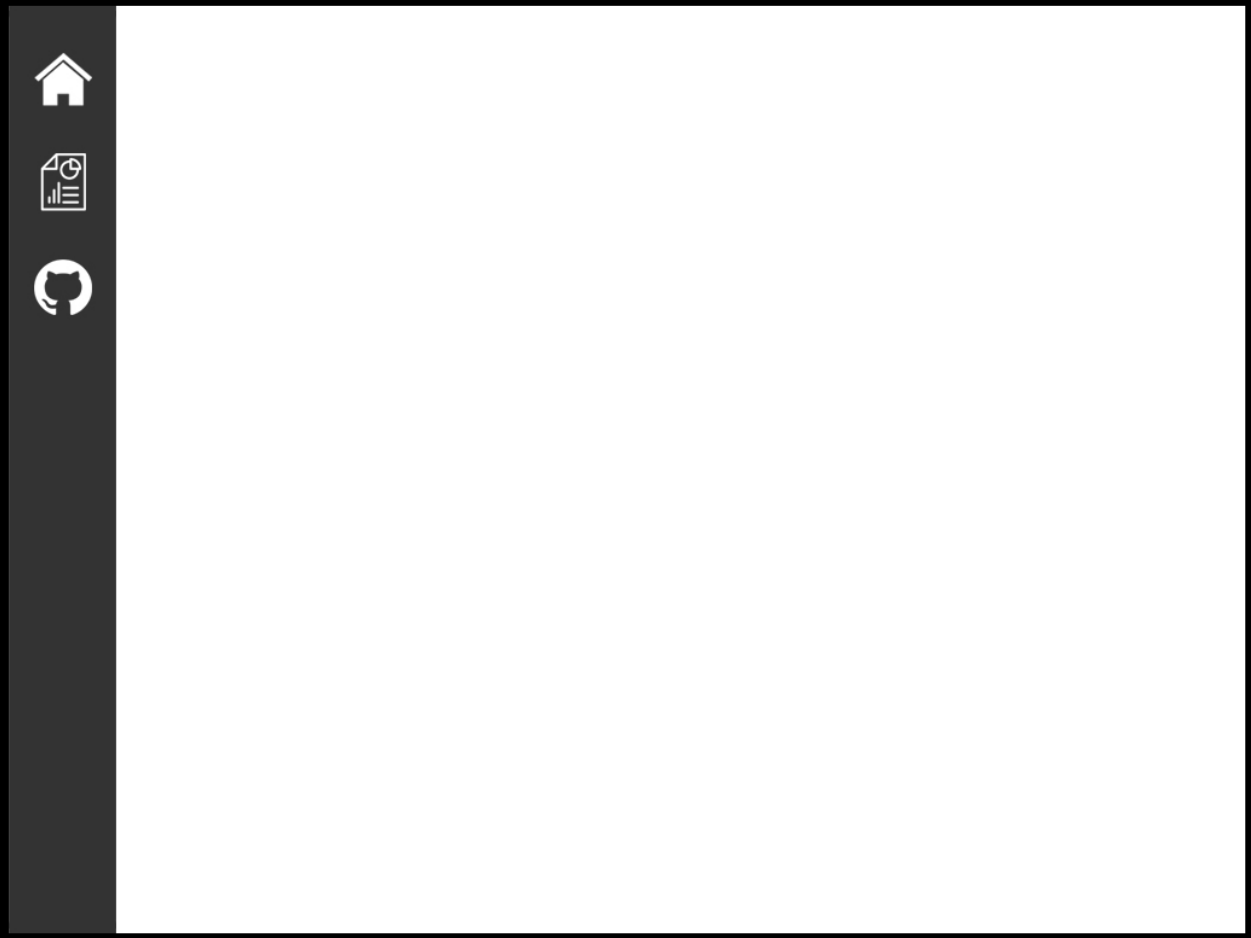
1. Menu



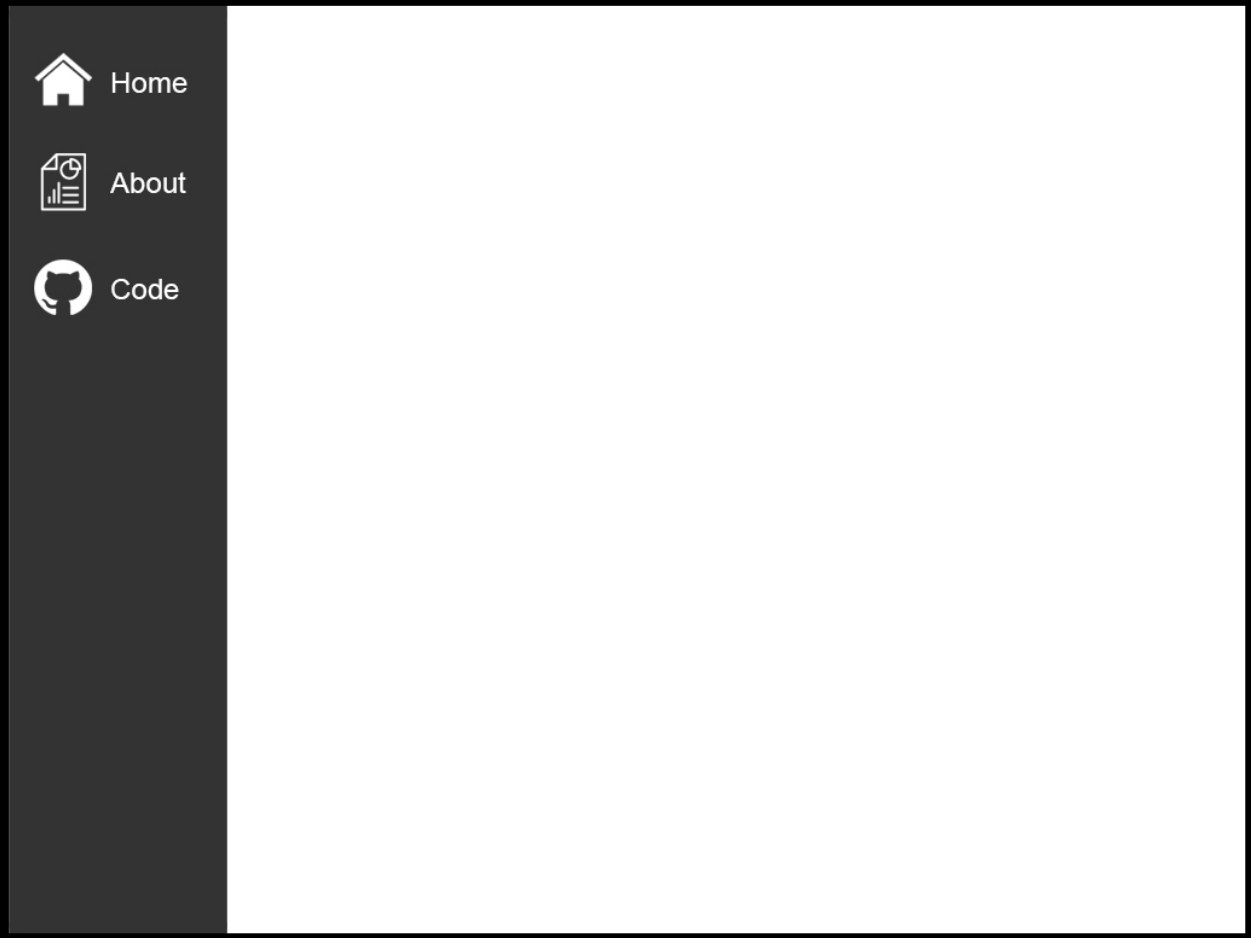
Our menu will have background color of #333333.
Size: height 100%, width 10%.
Note: these are height and width of the entire webpage.



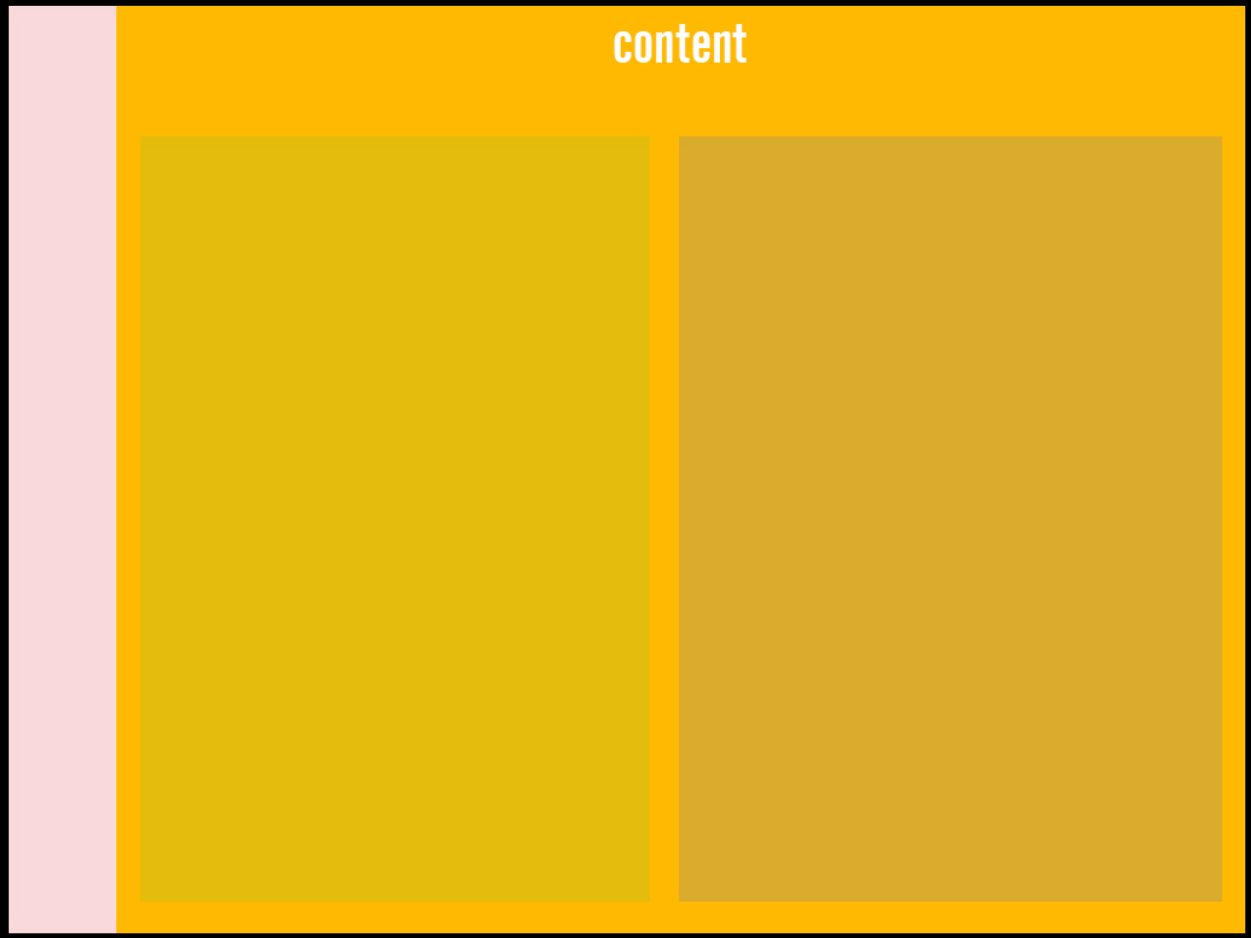
menu doesn’t contain many things except icons. Each icon links to a webpage. Unless the icons are hand-drawn using d3, their maximum size should be 48px * 48px. Each has a top-margin of 40px and is centered horizontally within its container.



On mouseover, menu will expand to display texts that go with the icons. On mouseleave, menu will return to its original state. Whoever does this will be responsible to choose an appropriate width for the expanded menu.

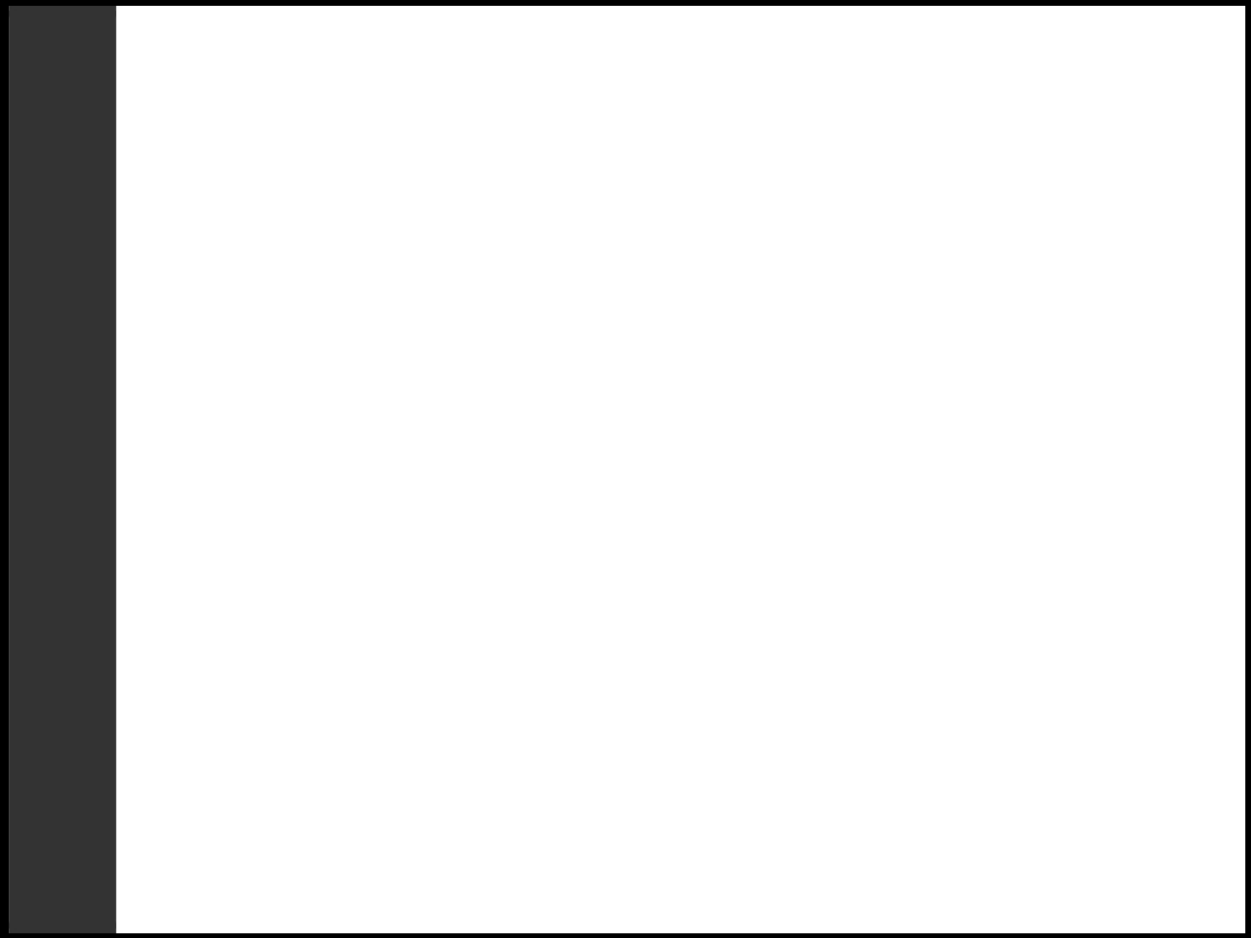


2. Content



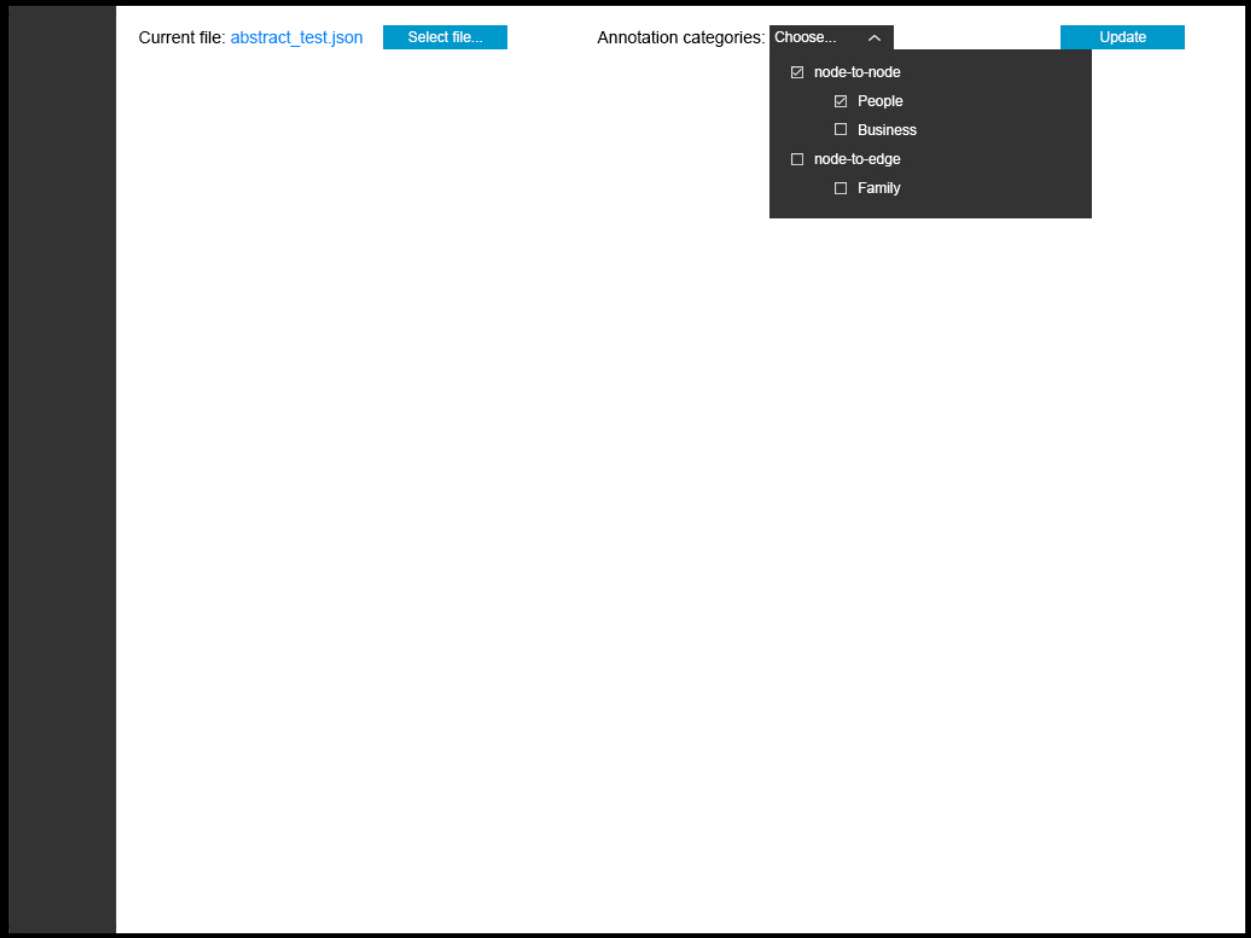
CSS properties of content:

- background color: white
- width: 90%
- height: 100%
- padding (all sides): 10%

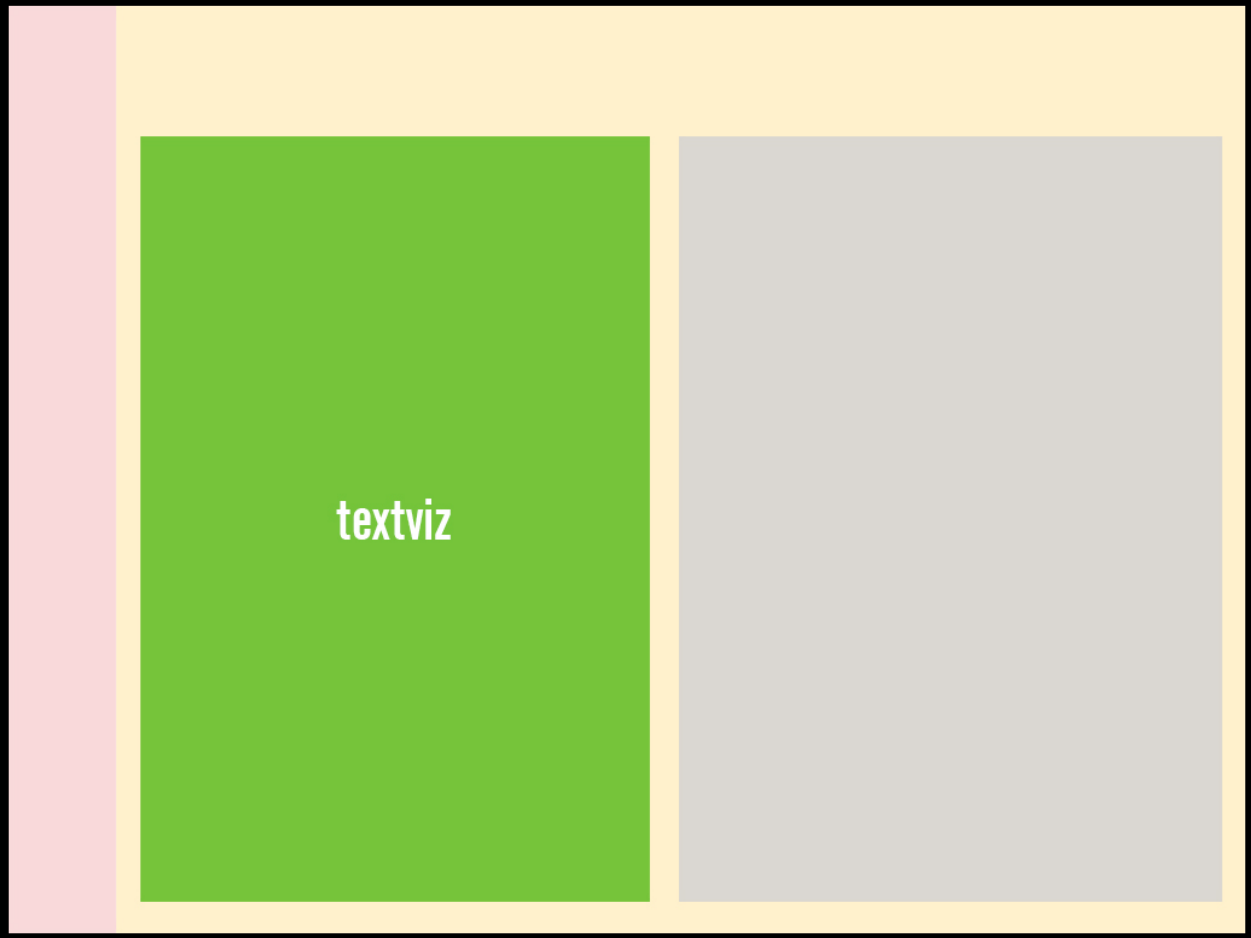


The top of content looks like this.

- Color of buttons is #0099cc
- The dropdown list may not be the default dropdown provided by HTML. This can be a hidden div that only appears on mouseclick.
- The list should be populated directly from data file (annotation file).

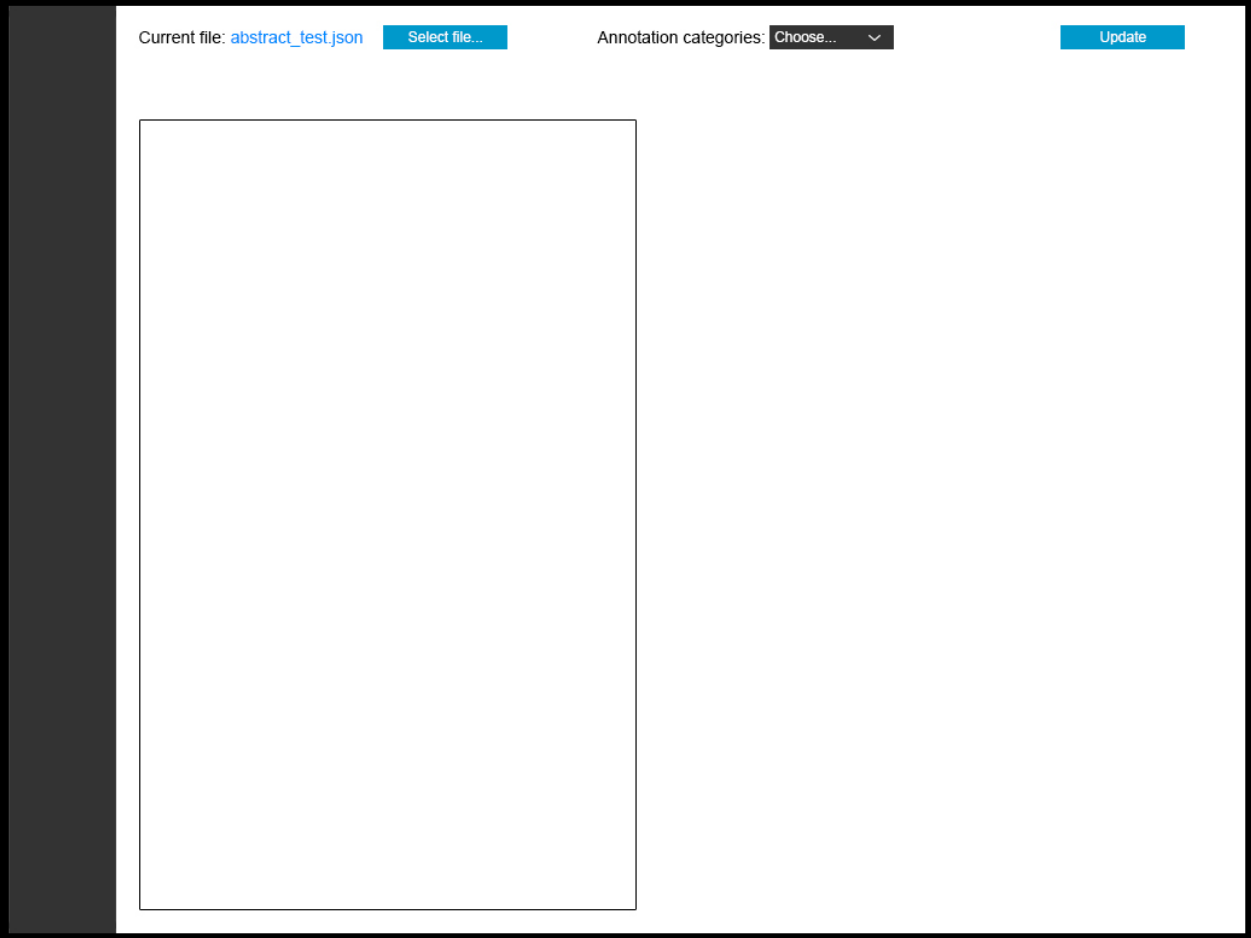


3. Textviz



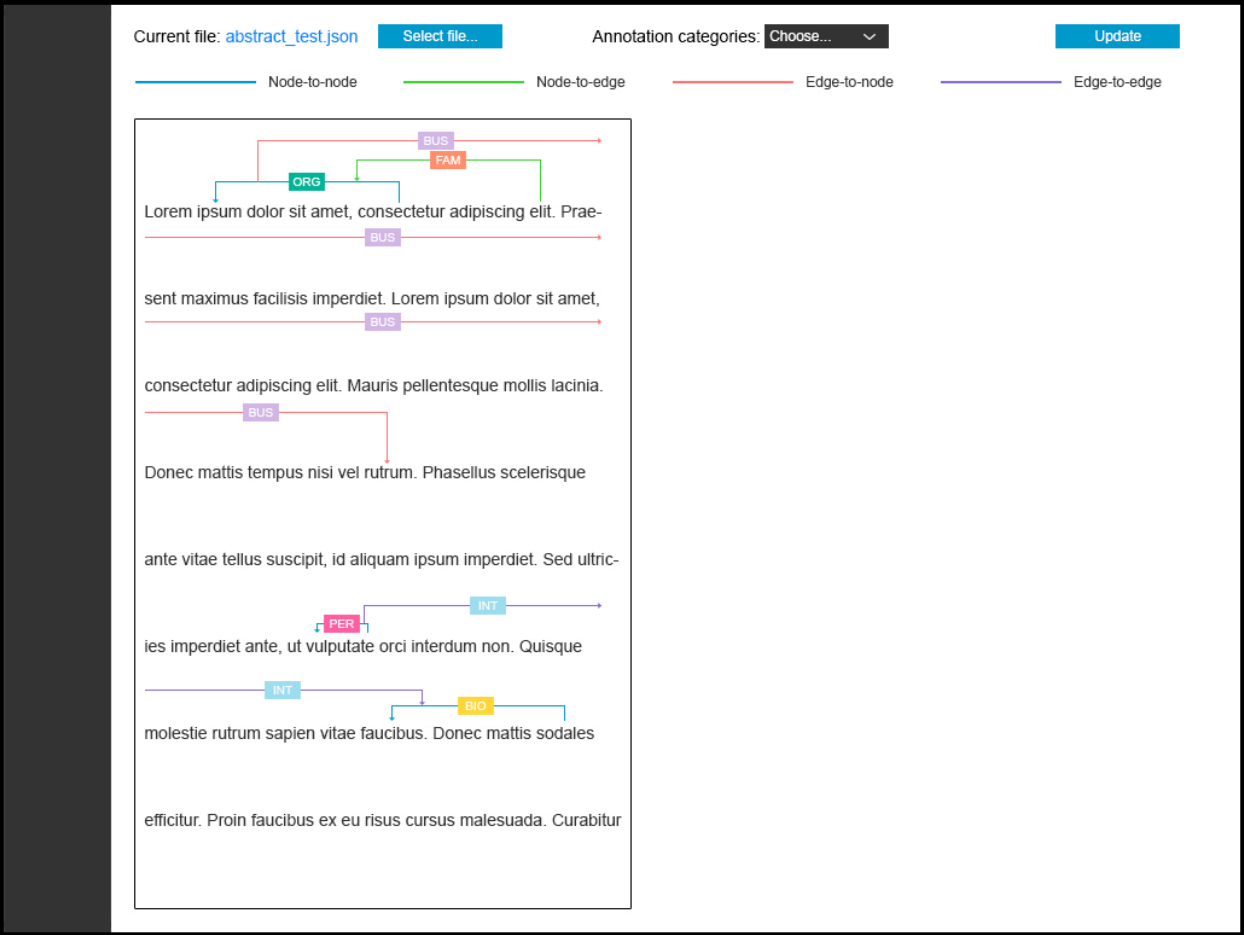
CSS properties:

- height: 85%
- width: 40%
- background-color: white
- border: black
- margin (all sides): 2%



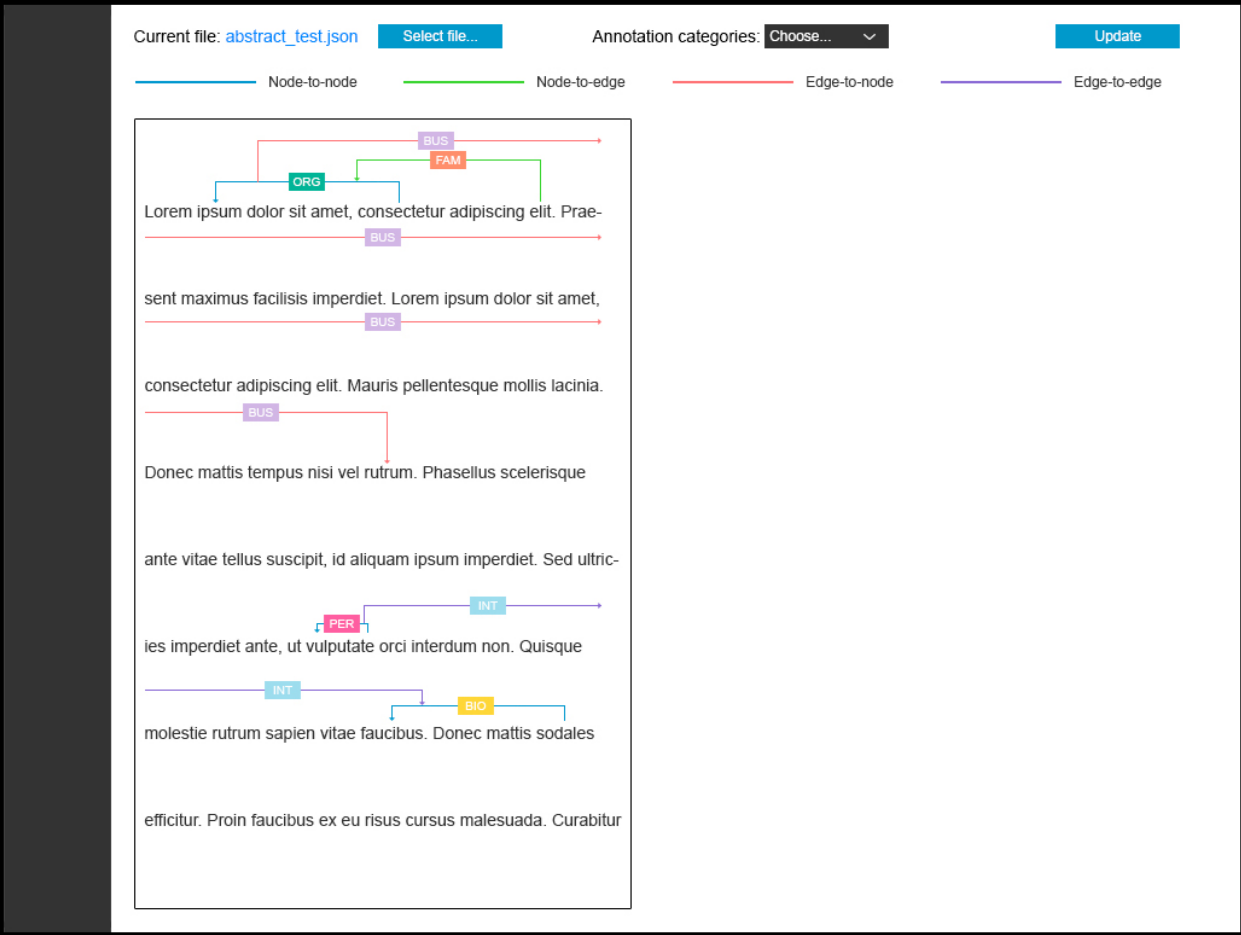
This textviz is used for displaying document in form of paragraphs, similar to how BRAT and ODIN do.

Connections (edge/link/arrows) will be color-coded based on their type:
node2node = #0098ce, node2edge = #3ad531,
edge2node = #ff7376, edge2edge = #8a6ad4
(Please check Readme.txt in folder data for details)

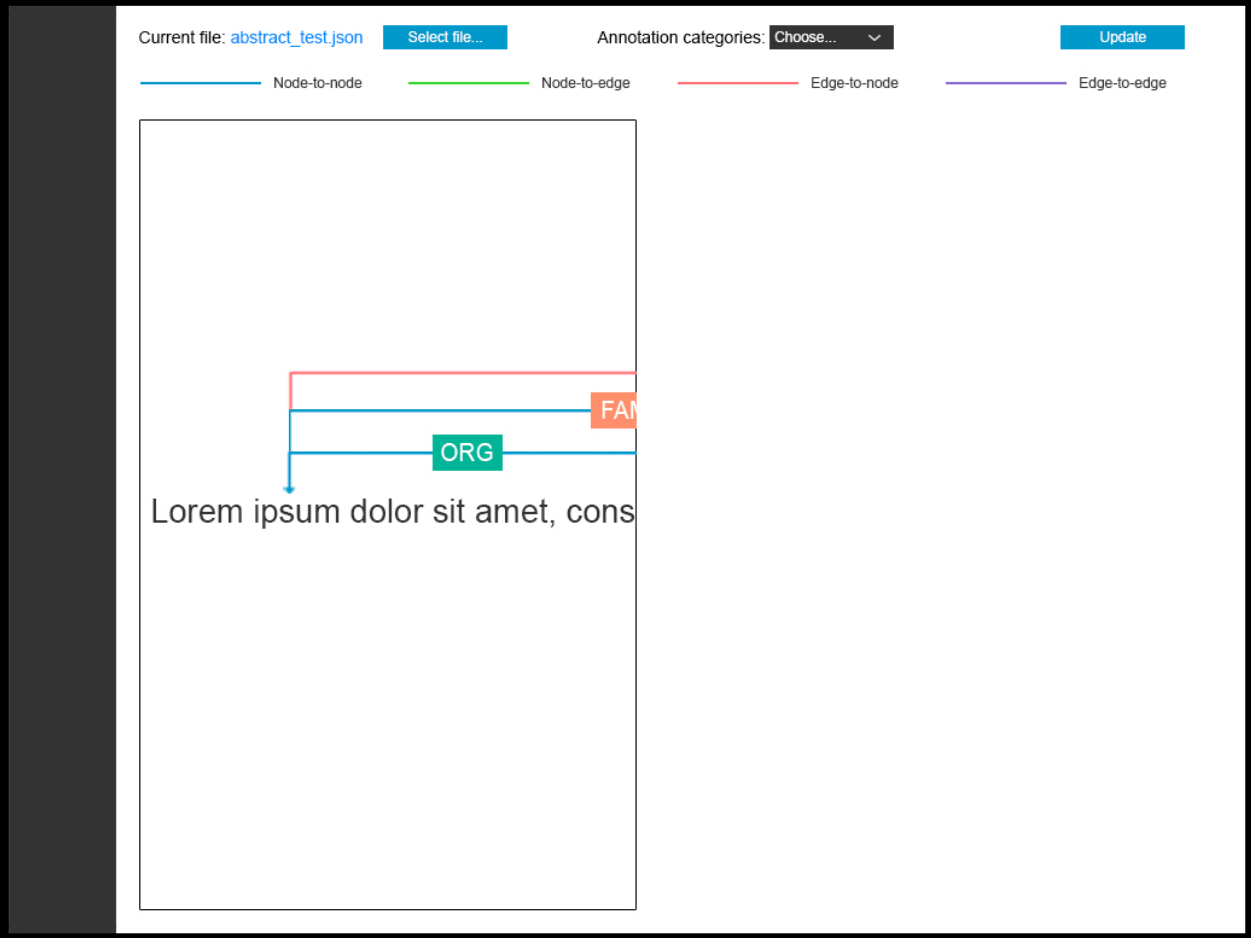


The labels are colored randomly, but still need to check for similarity (search for “Detect similar colours from hex values” on StackOverflow).

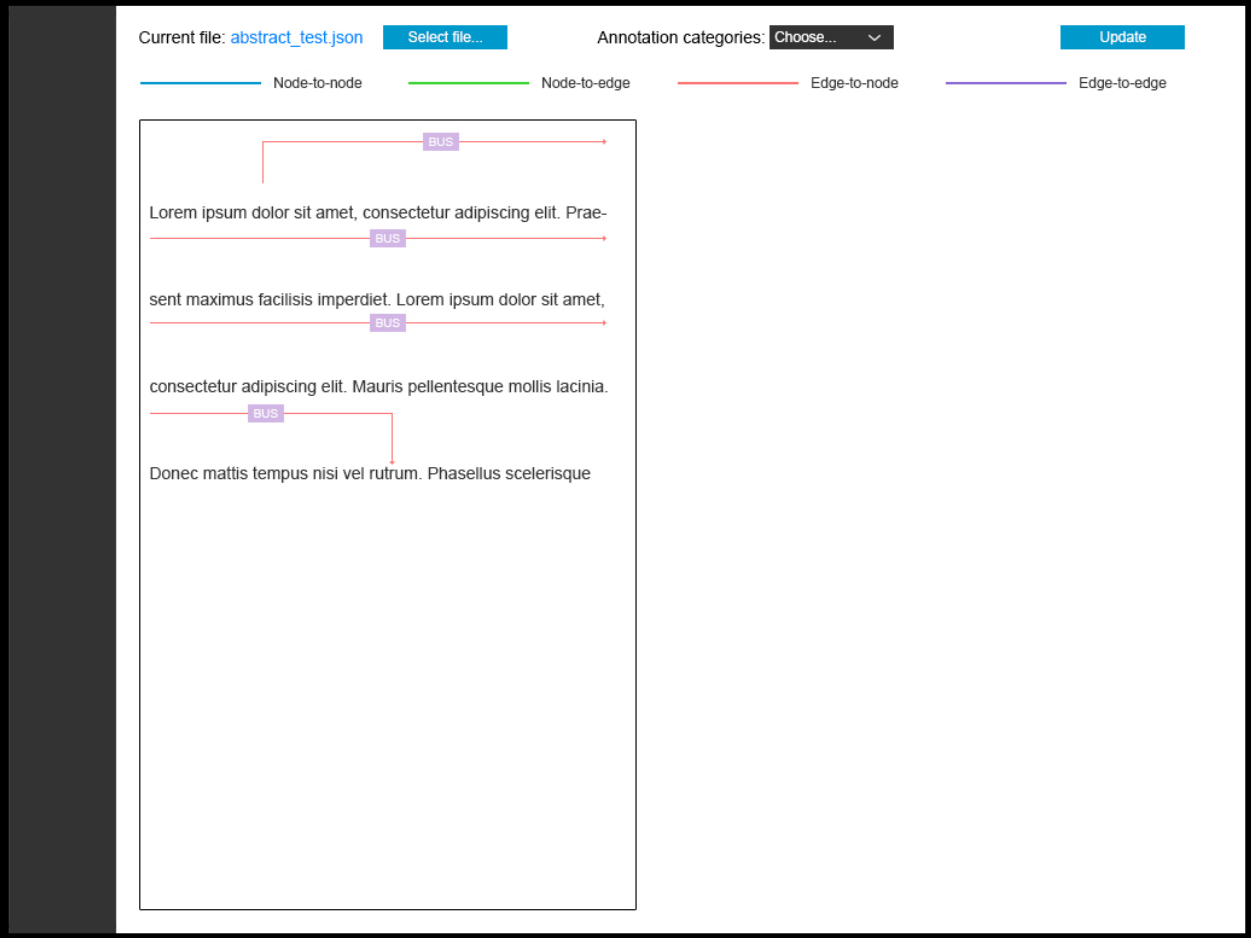
We need an algorithm to keep labels from overlapping each other. On next page, I’ll list a few rules that I’ve come up with.



1. There maybe more than 1 arrows point to the same node/word. If it happens, stack them up



2. If an arrow spans over multiple lines, replicate its label on all lines. On each line, put the label at center of the arrow segment.

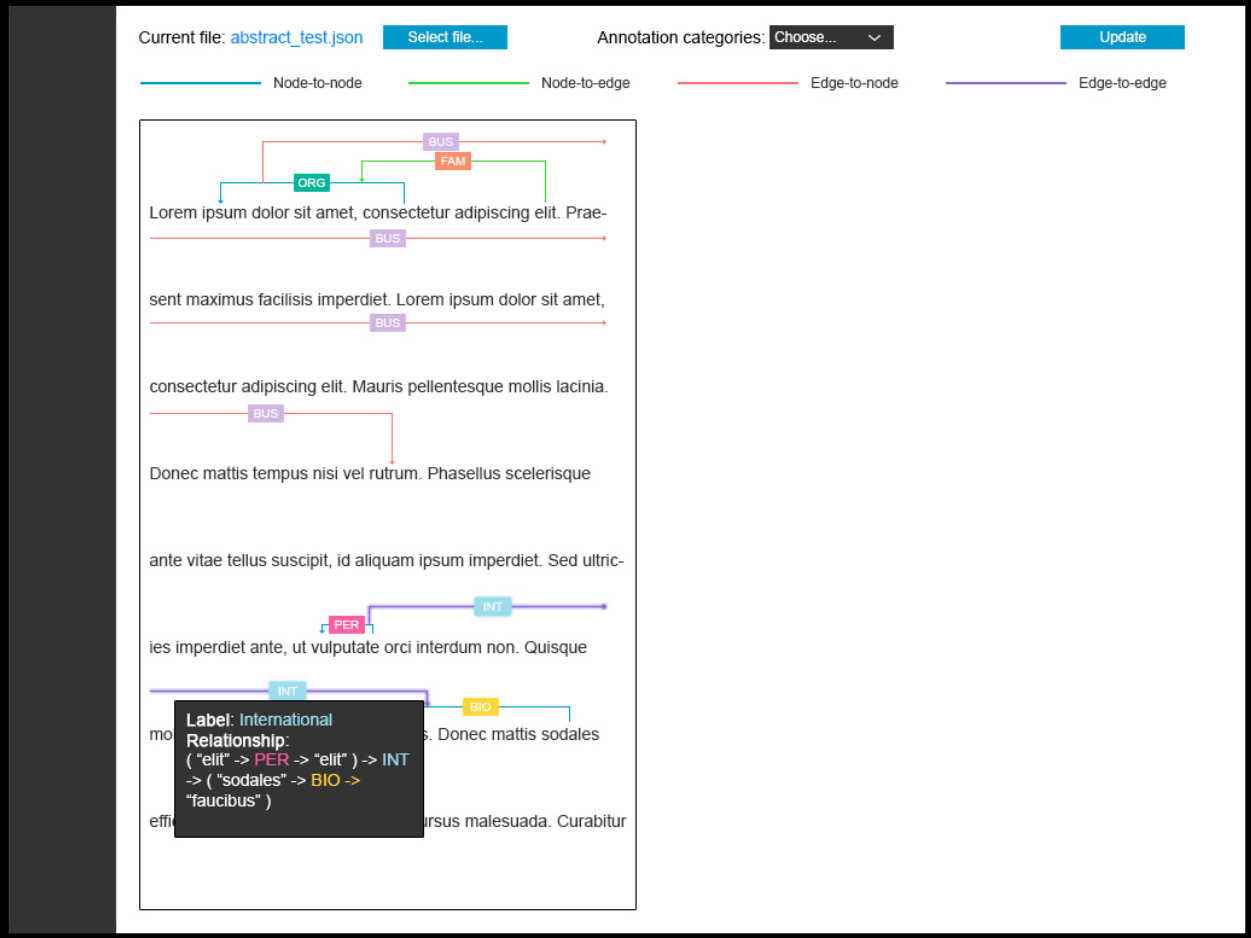


3. For each word/node, we need to manage where the arrows come in/out so they don't tangle up. My idea is to order by distance:

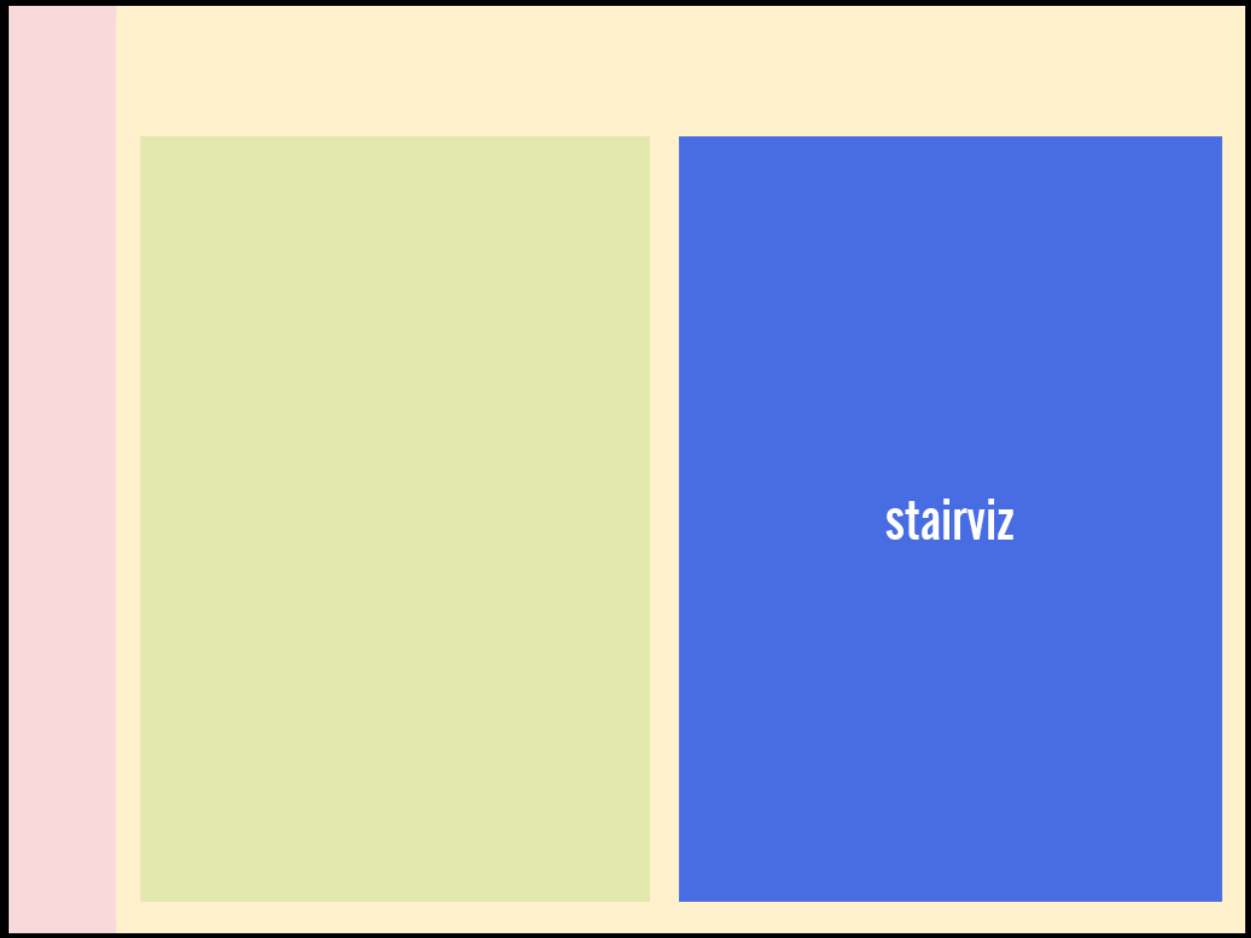
- For each node, gather how many arrows going in + out of it. Divide the length of the node so the gaps between 2 adjacent arrows are equal.
- Arrows that travel to the left are put on the left, arrows that travel to the right are put on the right.
- Starting from middle, put the arrow that travel the farthest left + farthest right first. Keep doing this until all arrows are placed.
- I'll adjust heights later.

When user hovers on an arrow, it will be hightlighted.
When user hovers on a label, details of that label will be displayed.

That's our first viz. Now I'll describe the second viz.



4. Stairviz



CSS properties: exactly the same as textviz