**Navigation problem nr 1: anchor elements don’t fit in a row**

When you are using soft tabs (two space characters) you have to be mindful of the space characters being rendered in the event there are inline and inline-block elements. Since this HTML uses soft tabs, a space character is rendered between each list item, making the total width of the row greater than 930px. To fix this, we can set a **font-size on the ul to 0** and set the **font-size on the lis to 1rem**. Rem measurements are relative ems based on the outer most font size declaration. So in this case, they will be set to the body font size of 16px.

**Navigation problem nr 2: anchor elements padding and text-align don’t work**

The anchors are still inline. For them to observe the top and bottom padding, they need to be a display type other than inline. Since they should also take up the entire li dimensions, setting them to **display: block will fix both issues**.

**Example of clearfix nr 1 (**put the ::after pseudo-element **on the containing parent element** of the floated contents**):**

main:after {

display: block;

clear: both;

content: "";

}

**Example of clearfix nr 2 (#columns is the div containing the floated divs):**

#columns {

width: 840px;

padding: 20px;

margin: 0 auto;

**overflow: hidden;**

}

**Example of clearfix nr 3:**

.cf::before, .cf::after {

display: block;

content: "";

}

.cf::after {

clear: both;

}

**Example of clearfix nr 4 (add clear: both to the element following the floated divs):**

.flbx-3 {

clear: both;

}

**Setting a margin between floated columns option 1:**

.row { /\* parent div \*/

clear: both;

**margin-left: -2%;**

}

.fourth {

float: left;

width: 23%;

**margin-left: 2%;**

}

**Setting a margin between floated columns option 2:**

.row {

clear: both;

}

.fourth {

float: left;

width: 23.5%;

margin-left: 2%;

}

.row .fourth:first-child {

margin-left: 0;

}

**Setting a margin between floated columns option 3:**

.row {

clear: both;

}

.fourth {

float: left;

width: 23.5%;

}

.fourth + .fourth { /\* **this sibling selector will add the left margin to each element beyond the first one** \*/

margin-left: 2%;

}

**Direct child unordered list of the header element:**

header > ul

**Sibling selectors 1 (all the list items in main will have border-top except the first one):**

main li ~ li {

border-top: 1px solid #0066cc;

}

**Sibling selectors 2 (setting the top padding of paragraphs that come immediately after an h2):**

h2 + p {

padding-top: 20px;

}

**Styling inputs for when they are being edited:**

input[type="text"]:focus {

background: #fff9b5;

}

**Using a combination of a pseudo selector and adjacent sibling selector:**

input[type="checkbox"]:checked + span {

font-weight: bold;

}

**Using an attribute selector:**

input[required] {

border: 1px solid #004f75;

}

**First-child:**

main li:first-child {

text-indent: 15px;

}

**Style all table cells within odd numbered table rows:**

tr:nth-of-type(odd) td {

background: #d4f7fb;

}

**Style the table cells in the third column:**

td:nth-of-type(3) {

text-align: center;

}

**Specify that <div> elements should have padding and border included in the element's total width and height, i.e. the padding and border of element no longer increase its width (see http://www.w3schools.com/cssref/tryit.asp?filename=trycss3\_box-sizing):**

\* { /\* setting box-sizing for all the elements \*/

box-sizing: border-box;

}

**or**

.simple {

width: 500px;

margin: 20px auto;

**box-sizing: border-box;**

}

.fancy {

width: 500px;

margin: 20px auto;

padding: 50px;

border: solid blue 10px;

**box-sizing: border-box;**

}

**Difficult float problem, how to offer a fixed width on one column and a variable width on the other:** <https://launchschool.com/lessons/72c8ad7b/assignments/1abad513>

If our primary column were the one with a fixed width, this would be very simple. We could replace its float and width with an overflow value of anything other than visible. This causes the container to adjust its width to fit whatever space is left over within that row.

body { background: #e0e0e0; }

#primary, #secondary { padding: 25px; background: yellow; box-sizing: border-box; }

#primary { float: left; width: 800px; margin: 0 20px 0 0; }

#secondary { overflow: hidden; }

This will allow the sidebar to take up whatever space is left. The overflow: hidden property is added to the secondary, otherwise the primary's right margin won't have any effect.

To pull off what our designer is asking for, while keeping the HTML structure, we will first need a parent container for these two columns. Wrap the two columns in another div, then add 220px of left margin to it. This will serve as the 200px of width for our fixed-width secondary column and the 20px margin between it and the primary column.

<div id="columns">

<div id="primary">

<h1>Main Content</h1>

</div>

<div id="secondary">

<h3>Sidebar Content</h3>

</div>

</div>

#columns { margin: 0 0 0 220px; }

Now comes the tricky part. Since our primary column is the one that will receive the remaining width of the parent container, we set it to float right and give it a width of 100%.

#primary { float: right; width: 100%; }

Now you'll see that we have the space available for the secondary column, but it's still not sitting within that area. We would have to escape the dimensions of the parent container to do this, and we can use negative margin to do so. On the secondary container, set a negative left margin equal to the columns container's left margin to overcome it.

#secondary {

float: left;

width: 200px;

margin: 0 0 0 -220px;

}

Now it works! The sidebar now maintains a fixed width and the main column adjusts to the parent's width. If you need to do the same with the fixed width column on the right side, you just reverse the margin and the float on the fixed width column.

**Centering horizontally:** margin: 0 auto;

**Modifying right column to take up the remainder of the space without floating the column or setting a width on it:**

section {

**overflow: hidden;**

max-width: 1200px;

margin: 0 auto;

background: #ffffff;

}

article {

float: left;

width: 500px;

padding-right: 30px;

background: #ffffcc;

box-sizing: border-box;

}

aside {

**overflow: hidden;**

background: #ccffcc;

}

**Use the box-sizing property to ensure that article takes up 70% of the row width, rather than 70% plus the 30px of right padding:**

article {

float: left;

width: 70%;

padding-right: 30px;

background: #ffffcc;

box-sizing: border-box;

}

**Position absolute**

Absolutely positioned elements are **moved in relation to their closest relatively positioned parent element**. Should a relatively positioned parent element not exist, the absolutely positioned element will be positioned in relation to the <body> element.

**Centering a position absolute div:**

To horizontally center a fixed width element that is positioned, you can move the element to the center of the window with position and move it back in the opposite direction with negative margin. In our example, we're moving the element from the left edge to the center of the page, then moving it back towards the left edge by half its width using negative margin.

#positioned {

position: absolute;

top: 0;

**left: 50%;** **// moving the element from the left edge to center of the page**

width: 400px;

padding: 30px 15px;

**margin-left: -200px; // moving it back towards left edge by half its width**

background: #ffffcc;

box-sizing: border-box;

}

**Taking up the entirety of the viewport:**

width: 100vw;

height: 100vh;

**Adding an element to absolutely positioned div so that it sits partly outside of parent element's boundaries:**

#positioned {

position: absolute;

top: 20px;

left: 50%;

/\* Add a z-index greater than the elements you want to appear on top of \*/

z-index: 2;

padding: 30px 15px;

margin-left: -200px;

background: #ffffcc;

width: 400px;

box-sizing: border-box;

}

#**positioned span** {

**position: absolute**;

**top: -10px;**

**left: -10px;**

width: 20px;

height: 20px;

background: #fcfc99;

}

**Inline elements (e.g. anchors) & margins:** they take only left and right margin values

**Inline elements (e.g. anchors) & padding:** they take padding values for all sides

**Floating inline elements take padding and margins on all sides**

**Centering div + improving the browser's handling of small windows:**

#main {

**max-width: 600px**;

margin: 0 auto;

}

**Inline-block layout:** there will be a gap between the columns if there is any whitespace between them in the HTML

**CSS frameworks:** [**http://learnlayout.com/frameworks.html**](http://learnlayout.com/frameworks.html)

**A simple CSS reset:**

/\*

----------------------------------------

Tantek Celik's Whitepsace Reset

Author: Tantek Celik, Shane Riley

Version: (CC) 2010 Some Rights Reserved - http://creativecommons.org/licenses/by/2.0

Description: Resets default styling of common browsers to a common base

----------------------------------------

\*/

ul,ol { list-style: none; }

h1,h2,h3,h4,h5,h6,pre,code { font-size: 1em; }

ul,ol,li,h1,h2,h3,h4,h5,h6,pre,form,body,html,p,blockquote,fieldset,input,dl,dt,dd, figure, figcaption {

margin: 0;

padding: 0; }

a img,:link img,:visited img, fieldset { border: none; }

address { font-style: normal; }

header, section, article, nav, footer, hgroup, details, summary, figure, main { display: block; }

mark {

color: inherit;

background: transparent; }

abbr { border: none; }

summary::-webkit-details-marker { display: none; }

**Other CSS resets:**

<http://necolas.github.io/normalize.css/>

<http://meyerweb.com/eric/tools/css/reset/>

http://cssreset.com/scripts/undohtml-css-tantek-celik/