# Wohola X

Release: 1.0 internal beta

Author: Pankaj Goel

Date: 2nd February 2024

Document version: 1.0 (date: 8th February 2024)

# Creating layouts with WXML

## Introduction

In general with WXML most layouts can be created directly using an xten. There is no need for complex CSS for creating these layouts.

We recommend you go through quick start guide in docs folder before going through this guide.

It is expected that developers learning WXML would have prior knowledge of HTML/CSS.

## Rows and Columns

The layout can be broken down into rows and columns.

    <row>

        <column>

            <fitwidth>

                <image name="site-logo" object-name="all" alt="Site Logo" />

            </fitwidth>

        </column>

        <column ratio="8">

            <align horizontal="flex-end">

                <navbar transform="uppercase" css-margin="0.5rem" item-gap="15px" navitem-hover-style="color: blue">

                    <navitem href="/">Home</navitem>

                    <navitem href="blog">Blog</navitem>

                    <navitem href="contact">Contact</navitem>

                    <navitem>

                        <show role="public"><a href="login">Login</a></show>

                        <show role="logged\_in"><logout>Logout</logout></show>

                    </navitem>

                </navbar>

            </align>

        </column>

    </row>

In example above row xten has 2 columns the second column has argument ratio=”8”. This divides the row into 2 columns with ratio 1:8.

By default on phones the rows are automatically switched to a single column. To control responsiveness row takes tablet-direction and phone-direction arguments where you can specify row or column as value.

Row also takes item-gap as argument to specify gap between columns. height and width arguments for row default to fit-content and 100%. These can be overridden.

Unlike some of the popular libraries for row and column layout wxml row and column can handle any number of columns.

## Example 2:

<row item-gap="15px" tablet-direction="column" height="200px" width="80%">

    <column>

        column 1 content

    </column>

    <column>

        column 2 content

    </column>

</row>

The quick start guide contains an example of creating a complex layout using row and column xtens.

## Aligning Items

To align items with row and column the center and align xtens are used. These xtens can have only 1 child. If there are more than 1 child to center or align they must be wrapped in a container xten.

center positions its child in the middle of parent row or column.

align when used without arguments achieves the same results as center.

align has further arguments height, width, horizontal and vertical. The arguments take values which are standard css options for flex-direction.

**horizontal options**

* flex-start: left align content in relation to parent
* flex-end: right align content in relation to parent
* center: center the content in relation to parent
* space-between: Distributes items evenly along the main axis, with the first item aligned to the start and the last item aligned to the end.
* space-around: Distributes items evenly along the main axis, with equal space around them.
* space-evenly: Distributes items evenly along the main axis, with equal space between them.
* stretch: Stretch items to fill the flex container along the main axis. This is the default behavior when the align-items property is set to stretch.
* if omitted this parameter defaults to center

**vertical options**

* flex-start: align item to the top of the parent
* flex-end: align item to the bottom of the parent
* center: align item vertically in the center of the parent
* baseline: Aligns items such that their baselines align
* stretch: Stretch items to fill the flex container along the cross axis. This is the default value.
* if omitted this parameter defaults to center

**Notes**

* The align and center work only when used inside rows and columns.
* The should have only 1 child element. If there are more than 1 children then they need to be wrapped in a container.

## Example 1

<align>

<span>This is placed in the middle of the parent</span>

</align>

## Example 2

<align vertical="flex-end" horizontal="flex-end">

<span>This is placed in the bottom right of the parent</span>

</align>

## textalign

align and center use flex container so they only work when inside a parent with display: flex or rows and columns

To align a single textitem where veritical alignment is not needed you can use textalign xten.

It takes an argument align. valid values are left, right, center, justify. default value when omitted is center.

<textalign>

    center aligned text

</textalign>

<textalign align="right">

    <span>right aligned text</span>

</textalign>

## Absolute positioning

If you want to position elements with absolute position then following xtens can be used.

* pos-left: argument left, default 0
* pos-right: argument right, default 0
* pos-top, argument top, default 0
* pos-bottom, argument bottom, default 0
* pos-top-left, arguments top and left, default 0
* pos-top-right, arguments top and right, default 0
* pos-bottom-left, arguments bottom and left, default 0
* pos-bottom-right, arguments bottom and right, default 0

These would work on any parent with position: relative. Generally you would use these inside container xten.

## Example

<container height="200px"> <!-- hard coded height for demo -->

    <pos-top-left>this would be on top left</pos-top-left>

    <pos-left left="10px">

this would be 10px from left vertically in middle

</pos-left>

    <pos-left left="10px">

this would be on right and vertically in middle

</pos-left>

    <pos-bottom bottom="30px">

this would be 30 px from bottom

</pos-bottom>

</container>

## Which one to choose?

* Align and center work if the parent is display: flex. row and column are flex so these would work.
* pos-\* work if the parent is a normal display: block or display: inline-block element. container is a display: block element so these would work inside it.
* Limitations of CSS are applicable as is on these xtens.
* As is the limitation with CSS in terms of height the parent container must have some height of its own either through a content or height attribute for pos-\* to work.
* textalign is to be used just for text elements where vertical position does not matter.

## cover

cover is an absolute positioned xten which covers the parent. If there are any clickable elements on parent these cannot be clicked when covered.

cover takes left, right, top, bottom, horizontal and vertical as argument. default value for position arguments is 0 and for horizontal and vertical is center.

if you want cover to cover the whole parent and children to be in the middle then you simply say

<container height="200px"> <!-- hard coded height for demo -->

    <cover>

        ... some children

    </cover>

<container>

So the container would be wholly covered and the children of cover would be in center horizontally and vertically.

If you want to leave 10px space on each side and align children to top left, you would say

<container height="200px"> <!-- hard coded height for demo -->

    <cover left="10px" right="10px" top="10px" bottom="10px" horizontal="flex-start" vertical="flex-start">

        ... some children

    </cover>

<container>

## screencover

screencover xten covers the whole browser screen. It does not take any arguments.

So for example if you want to show 'hello' in the middle of a div that that covers the whole browser screen then you would say

<screencover>

    <align>

        hellow

    </align>

</screencover>

## Another example

You can ignore the other xtens being used and only focus on alignment related xtens covered so far in this code.

* It is not important about the content but you can focus on layout and alignment related xtens only in this example for now.
* We have an element which is covering the while browser - screencover.
* Within it we have an icon to close this with name cross. It is placed in top right corner 10px from top and 10px from right.
* The we have a div on top with some text.
* Exactly in the center (horizontally and vertically through center xten) we have a container.
* Within this container first we have 2 rows with 2 columns each with some text and another element called inline.
* The first column is left aligned horizontally and is centered vertically through <align horizontal="flex-start"> xten.
* Then we have an element called textedit.

<screencover class="edit-email-text">

    <pos-top-right top="20px" right="20px">

        <icon name="cross">

            <click>

                const element = event.target.closest('.wx-xten-email-form');

                wxfns.removeClass(element, 'show-email');

            </click>

        </icon>

    </pos-top-right>

    <pos-top>

        <div style="padding: 10px; font-weight: bold; background: #efefef; text-align: center; width: 600px;">Email Body email-text-[$name]</div>

    </pos-top>

    <center>

        <container>

            <row>

                <column><align horizontal="flex-start">To Email</align></column>

                <column ratio="3">

                    <inline class="email-settings" name="email-recipient" object-name="[$name]" collection="constants">

                        To Email Address

                    </inline>

                </column>

            </row>

            <row>

                <column><align horizontal="flex-start">Email Subject</align></column>

                <column ratio="3">

                    <inline class="email-settings" name="email-subject" object-name="[$name]" collection="constants">

                        Email subject

                    </inline>

                </column>

            </row>

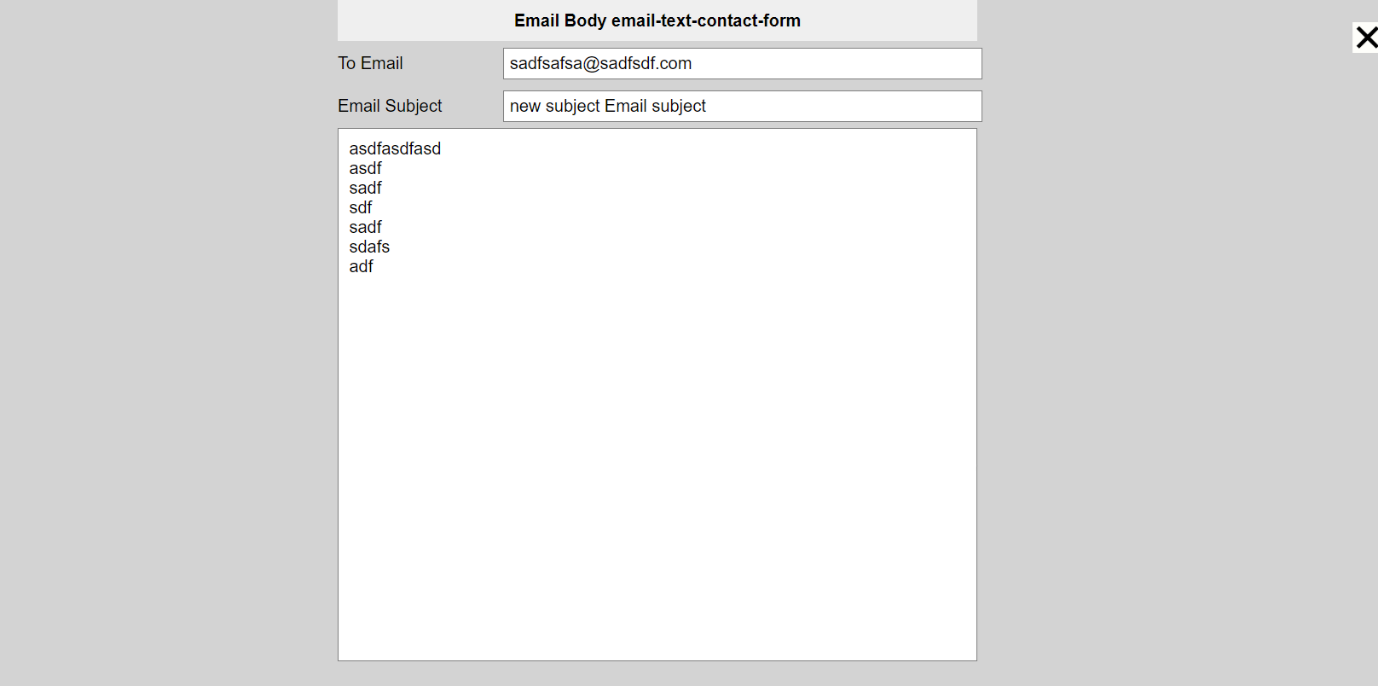
            <textedit name="email-text" object-name="[$name]" collection="constants"/>

        </container>

    </center>

</screencover>

And this is what the output looks like



## Some simpler xtens

1. <gap height="10px /> can give a gap of 10px between 2 different elements.

2. <space n="10" /> can add 10 white spaces.

3. columns

<columns cols="3" cols-tablet="2" cols-phone="1">

... some big text block

</columns>

This xten can break the big text block into a number of columns like newspaper columns. The arguments specify how many columns to break the text into on computer tablet or phone through the arguments this xten takes.

4. fitwidth and fitheight

<fitwidth>

<image name="xyz" />

</fitwidth>

and similarly fitheight fit the child image to width or height of the parent element.

5. flex container

This is a wrapper to create a flex container. In example below without flex wrapper the text in the 2 spans would have some gap due to how CSS works. If you want the output to be without any gaps then you can wrap them in flex container like so.

<flex>

<span>text1</span>

<span>text2</span>

</flex>

6. repeat

This xten takes an argument n and repeats its children n times.

<repeat n="5">

... some wxml content

</repeat>

## banner

Banner uses the following syntax.

<banner name="site-banner" aspect-ratio="1.5">

wxml content for a single slide here ...

</banner>

The below example has a banner with each slide containing an image and then some heading and text positioned on top through pos-middle.

## Example

<banner name="banner" aspect-ratio="2" interval="5">

    <image name="slide-image" list-child="true" />

    <pos-middle>

        <inline name="banner-label-heading" list-child="true">banner label</inline>

        <gap height="10px"/>

        <inline name="banner-label-text" list-child="true">banner text</inline>

    </pos-middle>

</banner>

## Carousel

Carousel works in similar manner. It animates the slides automatically continuously. The interval defines how many seconds to wait before changing the slide.

        <carousel name="banner" aspect-ratio="2" interval="5">

            <image name="slide-image" list-child="true" />

            <pos-middle>

                <inline name="banner-label-heading" list-child="true">banner label</inline>

                <gap height="10px"/>

                <inline name="banner-label-text" list-child="true">banner text</inline>

            </pos-middle>

        </carousel>

## Grid

Below is an example of a grid layout with 2 columns. Again within grid you only need to specify what a single grid item looks like.

        <grid name="features-list" columns="2" item-gap="15px">

            <quickplay name="feature-video" class="list-video" list-child="true" />

            <pos-middle>

                <inline class="video-heading" name="feature-heading" list-child="true" css-display="block">Feature Description</inline>

            </pos-middle>

            <container>

            <inline name="feature-description" list-child="true">Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore</inline>

            </container>

        </grid>

## List

List is similar except it has orientation of row or column. If orientation is row all items are placed side by side in a single row.

Example below shows a list with a row with 2 columns, where a video is placed in first column through quickplay xten and in second column there is textedit element.

        <list name="benefits-list" item-gap="15px">

            <row item-gap="30px" class="benefits-row">

                <column>

                    <quickplay name="benefits-image" list-child="true" aspect-ratio="calc(16/9)" />

                </column>

                <column ratio="1.2">

                    <align>

                        <textedit name="benefits-text" list-child="true">

                            Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut. Lorem ipsum dolor sit amet.

                        </textedit>

                    </align>

                </column>

            </row>

        </list>

Here is an example of list with row orientation. In this list as many items you add they would be placed in a single row, while in grid layout above more rows would be added.

<list name="featured" item-gap="32px" orientation="row" css-min-height="200px">

    <align>

        <image name="feature-image" list-child="true" aspect-ratio="1"/>

    </align>

    <align height="fit-content">

        <inline name="feature-label" list-child="true">

feature label

</inline>

    </align>

</list>

## Data management in banner, carousel, grid, list

In WXML you only specify what a single list item or grid item or slide looks like in that element. Adding and deleting additional slides or grid items or list items can be done after logging in as admin.

To edit data in individual list items simply use editable xtens like image, inline or textedit which allow admin to change image, change single line text or edit a formatted text block respectively. Simply add list-child="true" and wx knows how to manage the data related to it.

You only need to specify field names while coding and need not worry about actual text or images that would go there. Simply create the layout first and then log in as admin and then put the actual text or images.

## Layers

In situations where you need to create layers like in image editing software you can use absolute positioning. Another xten exists for the same which is layers. You can create overlapping layers using this xten.

<layers>

    <image name="abc" />

    <layer>

        <center>

            layer 1 - middle positioning

        </center>

    </layer>

    <layer index="-2"> <!-- index specifies z-index -->

        <align horizontal="flex-start" vertical="flex-start">

            layer 2 - top right positioning

- not visible as z-index takes it below image

        </align>

    </layer>

    <layer index="1">

        <align horizontal="flex-end" vertical="flex-end">

            layer 3 - bottom right positioning

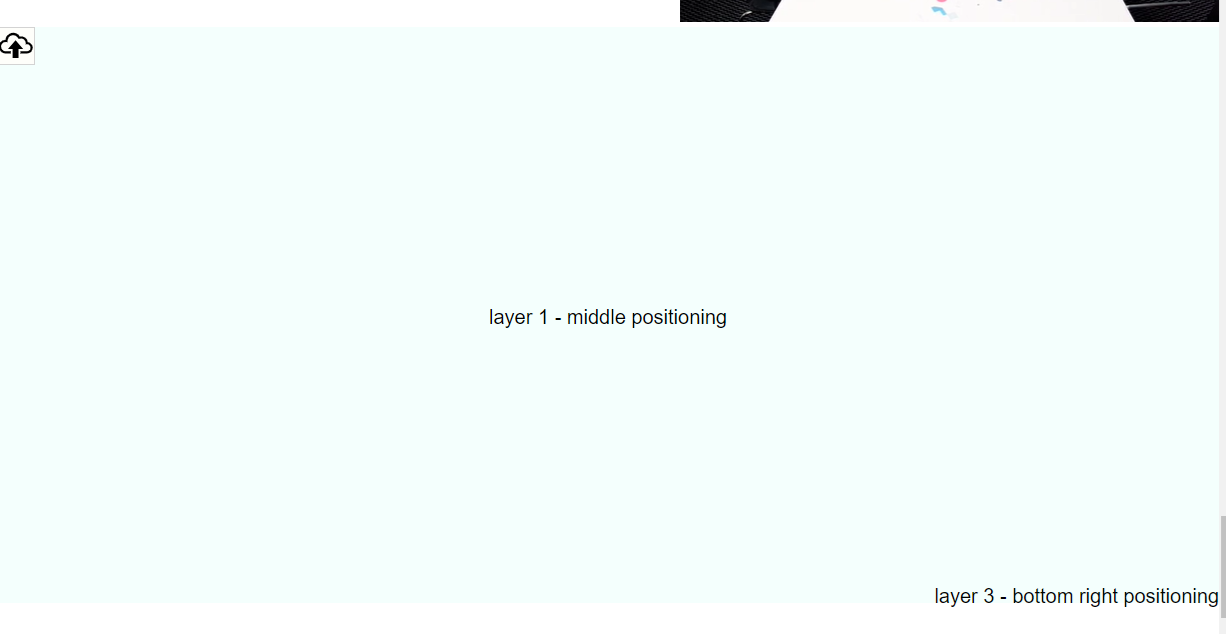
        </align>

    </layer>

</layers>

In layers xten there must be one layer that is the base layer that defines the height. Rest of the elements are put into layers and z-index can be controlled through index argument.

Output of above code is shown below



# Conditional Layouts

There are situations where the layout needs to be changed based on certain conditions. There are many ways to achieve that in wxml.

Let us say you want to create a layout which aligns a video and text left and right based on odd or even numbering.

Here is one way of doing it

<list name="benefits-list" item-gap="15px">

    <row item-gap="30px" class="benefits-row">

        <css name="benefits-row">

            .wx-xten-listitem:nth-child(even) .benefits-row {

flex-direction: row-reverse !important;

}

        </css>

        <column>

            <quickplay name="benefits-image" list-child="true"

aspect-ratio="calc(16/9)" />

        </column>

        <column ratio="1.2">

            <align>

                <textedit name="benefits-text" list-child="true">

                    Lorem ipsum dolor sit amet, consectetur adipiscing elit,

                </textedit>

            </align>

        </column>

    </row>

</list>

Here the listitem is checked to see if it is nth-child(even) and if it is then in its descendant .benefits-row the flex-direction is set as reverse. This achieves the desired result where video would be on left in one row and on right in next row.

There is another way to do the same thing or to get even more complex conditional layouts created.

<list name="benefits-list" item-gap="15px">

    <row item-gap="30px" class="benefits-row">

        <if condition="(node.closest('.wx-xten-listitem') ? parseInt(node.closest('.wx-xten-listitem').attribs['list-idx']) : 0) % 2 == 1">

            <column>

                <quickplay name="benefits-image" list-child="true"

aspect-ratio="calc(16/9)" />

            </column>

        </if>

        <column ratio="1.2">

            <align>

                <textedit name="benefits-text" list-child="true">

                    Lorem ipsum dolor sit amet, consectetur adipiscing elit,

                </textedit>

            </align>

        </column>

        <if condition="(node.closest('.wx-xten-listitem') ? parseInt(node.closest('.wx-xten-listitem').attribs['list-idx']) : 0) % 2 != 1">

            <column>

                <quickplay name="benefits-image" list-child="true"

aspect-ratio="calc(16/9)" />

            </column>

        </if>

    </row>

</list>

Here we put video column at the start and end of the row and used if condition to switch them on or off alternatively. If condition uses a javascript expression. It is a bit more complex but it can be used to create very complex conditional layouts.

# Code and overriding

The code for all the xtens described exists in \_system/xtens in main wohola x folder. You can review that code. All of these xtens can be overridden in your project by simply creating a folder called \_system/xtens in your project directory and create a file with same name as the xten you want to override. You can then customize or change that xten as per your needs.

You can also create new xtens in your project for any custom requirements or reusable components in your project.

# Summary

There are a number of xtens that allow creating virtually any type of layout that already exist. More xtens are being created. Each xten enhances wxml language and adds a new feature to that language. You can either override the core xten or create new ones for your custom requirements.

Using these xtens very complex layouts can be created easily and even responsiveness can be taken care of easily. If you use these xtens there is minimal CSS required as all layout related CSS is taken care of by the xtens themselves, so you only need to specify CSS for colours, fonts, borders, margin, padding etc in most cases.

This means that issues encountered during CSS development or maintenance can be minimised through use of wxml. Also code is very small and compact, there is no need to go looking around in different files to see what is happening where, so all in all it is easier and quicker to code and easier to maintain.

The code also looks clean and there are no external interdependencies as even the backend functions and data management are taken care of in this code.

The only word of caution is that eventually all CSS restrictions still apply to output of wxml so you need to be aware of how CSS works while working with xtens, even though you may not be actually writing most of that CSS.