# WX Xten List

This is a work in progress document

# Internal xtens

\_form, \_list, configuration

These xtens are not meant for direct use by WXML developers

# HTML tags

All HTML tags are allowed to be used within WXML and can be used as is. The difference in using an HTML tag vs an xten is that an xten tag have features and functionalities, while HTML tags are just tags that do nothing.

**All tags must be closed**

# add-libs

to add custom js or css files

args:

libs: comma separated paths of js or css files

Example:

<add-libs libs="js/mycustom.js, css/mycustom.css"/>

It would create the link or script tags in the <head> tag for each of these libraries. The paths are relative to site public folder. It would first look for libraries in the site public folder if not found then it would look for these libraries in wx public folder.

# align

to align a child element in relation to the parent

args:

width: width of the element - default 100%

height: height of the element - default 100%

horizontal:

* flex-start: Aligns items to the start of the flex container.
* flex-end: Aligns items to the end of the flex container.
* center: Centres the items along the main axis of the flex container.
* 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.

default is center

vertical:

* flex-start: Aligns items to the start of the cross axis of the flex container.
* flex-end: Aligns items to the end of the cross axis of the flex container.
* center: Centres the items along the cross axis of the flex container.
* baseline: Aligns items such that their baselines align.
* stretch: Stretch items to fill the flex container along the cross axis.

default is center

align can have only child. If more than one item as children then those children must be wrapped in another element.

# banner

to create a banner with multiple slides

inherits: \_list

args:

aspect-ratio: mandatory - the width to height ratio of the banner

**animation: to be done - it would take the name of the animation and switch slides using this animation**

children must be contents of a single slide

example:

<banner name="banner" aspect-ratio="3.5">

     <image name="slide-image" list-child="true" translate-y="-50%" />

     <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>

for any data which is linked to the banner list item add attribute list-child="true" to indicate that data is linked to the banner.

# captcha - to be done

captcha for detecting bots

# carousel

inherits: banner

args:

aspect-ratio: mandatory - width to height ratio

interval: the time duration between slide change in seconds

**animation: to be done - it would take the name of the animation and switch slides using this animation**

It is used in the same manner as banner. The only difference is it allows for automatic transitions between slides after 'interval' seconds.

The editable elements like inline, image etc should have list-child="true" to indicate to WX that that data item needs to be linked to the list context.

# center

no arguments

it centers the child element vertically and horizontally in the middle of the parent. It can have only one child and if more than one child they should be wrapped in another element.

# circle

args:

radius: CSS border-radius - default 50%

It creates a circle or oval div within which you can have some other content.

# click

It adds the click event handler to the parent element. The text can be any client side javascript code. The event is received in a predefined 'event' variable to this code. From this any normal javascript that is required can be written. To run a transaction on server wxfns.transaction method is used. Only xtens are allowed to run any transactions.

# column

to be used within a row to define a column

args

ratio: the relative ratio of the column - default 1

This xten allows having as many columns with any random width ratios. This is a more powerful feature compared to most libraries dividing the display in units of 12 and hence failing if you wanted to create 5, 7, 8, 9 or > 12 columns. Also this xten is much more efficient because it does not download large CSS files. It only adds a few lines of CSS code to the current page.

**It is important to note that not just column xten WX uses similar strategies for each aspect of its architecture to minimize issues faced, remove any limitations, and not have performance impact. Every little thing within WX is architected keeping such things in mind.**

# columns

this feature has been added to textedit and this xten has been removed

container

a block container to be used as replacement for a div. advantage of using this over div is that this can be enhanced to have additional features like being able to change the style. so when those features are added all sites using this container would get that new feature automatically without any code change.

Only major sections of the layout should be made a container, sometimes visually small items require a wrapper to handle technical nuances but are not major screen elements then they should not be made a container.

# cover

a wrapper to cover the parent element

args:

top: margin-top - default 0

left: margin-left - default 0

bottom: margin-bottom - default 0

right: margin-right - default 0

horizontal: justify-content - horizontal alignment for child element

vertical: align-items - vertical alignment for child element

This would cover the parent and content can be placed inside it. It can have only one child.

# css

args:

name - any unique name for the css element

the text contains any normal CSS that you may normally write in a .css file

The reason for this tag is to ensure code encapsulation of css in or around the tag it is being applied to, to reduce the persistent problems faced with css files of trying to find out what is affecting and where. Using this tag properly would ensure there are less chances of problems and when there is a problem that problem can be investigated easily.

Another reason is that contents of this tag can be customized through WX variables while an external css file is fixed.

A WXML developer should avoid using a fixed external CSS.

# currency-icon

To show currency symbol as an icon

args:

name: 2 digit ISO country code in lowercase. Lowercase country name can also be used but ISO code is preferred.

height: default 30px

border: default 1px solid grey

border-color: default #efefef

# currency

To show currency symbol in a span element

args:

name: 2 digit ISO country code in lowercase. Lowercase country name can also be used but ISO code is preferred.

# datetime

to format the date and time output

args:

name: name of the field - mandatory

format: the format of the datetime output

currently supports DD, MM, YY, YYYY, hh, mm, ss, ms, month, mon, day, weekday

collection: name of the collection where data is stored

this argument is optional and not to be used without knowledge of underlying architecture

object-name: the name of the object - optional

when not given it picks up data from database within the context of the page

if provided - this is normally the URL of the page if its a static page and name of the object if its an object defined using object-config

# draggable

to create a wrapper which the user can drag and resize

args

overflow: default value hidden

resize: whether to resize or which axis to resize - default both, valid values: none, both, horizontal, vertical

cursor: default value move

# dynamic-grid - to be done

a layout where its possible to have a grid layout where each item can be dragged or resized and new items can be added or existing items can be deleted.

It would work like this: https://dsmorse.github.io/gridster.js/

and it would be fully responsive.

# email-form

a form that emails the data to an admin email account when the form is filled. useful for things like contact form for smaller websites which do not need a CRM.

args:

name: some unique name for the element

object-name: the name of the object where the smtp details would be saved

Example

<email-form name="contact-form" object-name="\_all">

    <heading class="contact-form-heading">

        <textalign>

            <inline name="contact-form-heading" object-name="all">Contact Form Heading</inline>

        </textalign>

    </heading>

    <xinput name="contact-name" type="text" label="Name" required="true" />

    <xinput name="contact-email" type="email" label="Email" required="true" />

    <xinput name="contact-phone" type="phone" label="Phone" />

    <xinput name="message" type="textarea" label="Message" />

    <captcha />

    <textalign>

        <xbutton type="submit" style="[#button-style]">Schedule A Demo</xbutton>

    </textalign>

</email-form>

This form has 2 buttons, one for smtp settings and one for email details.

The smtp settings button specifies the server details for smtp server and from email address. The other button allows changing the email text, subject and to email address.

The email would be sent from 'from' email address using smtp settings provided to 'to' email address and subject and text would be as specified by admin. The form data would be automatically placed where [$form-data] variable is placed in email body.

# event

adds an event to the parent element

args:

name: name of event (click, blur, keyup, keydown etc)

text contains any javascript code that would be run when that event occurs.

The event is received in a predefined 'event' variable to this code. From this any normal javascript that is required can be written. To run a transaction on server wxfns.transaction method is used. Only xtens are allowed to run any transactions.

# fitheight

fits the underlying element to the height of the parent and the width is auto calculated. This can have only one child.

# fitwidth

fits the underlying element to the width of the parent and height is auto calculated. This can have only one child.

# flag

for country flag in svg format

args:

name: 2 letter lowercase ISO country code, country name can also be provided in lowercase but for consistency ISO code should be preferred

ratio: either 4x3 or 1x1 - default is 4x3

If some flag is missing or needs to be overridden then it can be added in <site>/public/flags/4x3 or 1x1 folder in svg format and used as is.

# fit - to be done

fits the underlying element to the width and height of the parent.

# flex

creates a flex container. a typical example is that span elements have gaps in between, if you want the span elements to have no gap they can be wrapped in a flex container. also in cases where child element needs parent to be a flex container this can be used.

# font

to add any google font to the project

args:

name: name of the font

weight: weight of the font

Example

<font name="Poppins" weight="400" />

After this the font can be used like normal font in font-family in CSS or style

font-family: Poppins, Arial, Helvetica;

It currently works only with google fonts. https://fonts.google.com/

# fullscreen - incomplete - to be done

If you want to have some element in a full screen mode then fullscreen can be used.

# gap

to add a gap between two block elements which are above and below each other.

args: height

grid

to create a layout with x columns and y rows

inherits: \_list

args:

orientation: row or column

columns: if orientation is column then how many columns

item-gap: gap between each item

child contains an individual item of that grid

This allows creation of row column based layout.

# icon-list - to be done

this xten would show images of all icons available and their name, it would allow adding new icons to the system or to the site which can then be used by icon xten.

# icon

to create an icon

args:

name: the name of icon

There are existing icons in \_system/public/icons folder. these are all svg files. these icons can be used by giving name of the file without .svg extension. so if the file is \_system/public/icons/add.svg this icon can be used like this

<icon name="add" />

The xten first looks for icon in the <site folder>/public/icons and if it finds it there it uses that file, if it is not found in site folder then it picks up that icon from \_system/public/icons folder. So any or all icons can be overridden by project basis if needed by creating new files in site folder and using them.

First check which icons are available in \_system/public/icons and use them if already available. if not available then download the svg icons from google icons: https://fonts.google.com/icons

Any icons that can be reused in future projects and have meaningful names can be moved to wx folder so that they can be available to future projects.

The xten does not have performance issues as it simply places the svg code inline within the html. This is the reason for all icons to use only svg, otherwise either large amount of code has to be written to manage icons properly through a single font file or there is a big performance impact due to a lot of icon files being downloaded. There are also advantages over common strategy of using font files for icons, as large font file has to be downloaded even if one icon is needed and then there are situations where some icon may not be there in that font file.

The WX strategy for managing icons has no performance bottlenecks and no such issues as faced using other icon libraries. Any and all icons can be used without any limitation and no extra files are downloaded.

**Icon strategy is simple to explain and hence difference is explained here how WX strategy is much more effective, has either same or better performance than other such libraries and has no issues or limitations that are encountered while using those libraries.**

**It is important to note that not just icons WX uses similar strategies for each aspect of its architecture to minimize issues faced, remove any limitations, and not have performance impact. Every little thing within WX is architected keeping such things in mind.**

# if

There are some situations where the layout needs to be designed conditionally. If xten allows conditional layouts.

args

condition: any javascript expression that returns true or false

defined: name of any variable

not-defined: name of any variable

var-value: name of any variable

children are elements that are to be show if the condition is true.

if defined attribute is given then the if would show content within it if a variable has been defined.

if not-defined attribute is given then if would show content within it if a variable has not been defined.

if var-value is given then if would show content within it if the value of variable is not null

If condition is given then the condition is evaluated and if its true then the content is shown.

<if condition="node.parent.attribs['show-banner']">

<banner name="my-banner" />

</if>

In above example if the if element's parent has attribute 'show-banner' then the banner would be shown. This xten works at server level. To do the same thing in browser use <show> xten.

# image

replacement for img tag for using with images

args:

name: some meaningful unique name for that image

src: optional - same as src attribute of img tag

alt: optional - same as alt attribute of img tag

aspect-ratio: width to height ratio of the video - default is the original aspect ratio of the image

toolbar-top: optional - top margin for toolbar position - default is 0

toolbar-left: optional - left margin for toolbar position - default is 0

toolbar-right: optional - right margin for toolbar position - default is auto

toolbar-bottom: optional - bottom margin for toolbar position - default is auto

object-name: the name of the object - optional

when not given it picks up data from database within the context of the page

if provided - this is normally the URL of the page if its a static page and name of the object if its an object defined using object-config

collection: name of the collection where data is stored

this argument is optional and not to be used without knowledge of underlying architecture

href: when provided clicking on images changes location to href value

**scale: to be done - it should take a value and when this argument is provided hovering on image should scale it**

This xten allows automatic selection of image by admin and saving to database. If there is no src and no value in database then \_system/public/images/blank.jpg is picked up.

blank.jpg can be overridden for a project by creating <site folder>/public/images/blank.jpg

This xten first uploads the image in \_tmp folder in site folder and then moves it to public/uploads/images folder and deletes it from \_tmp. Both these folders are automatically created if they do not exist.

So all uploaded images can be seen in <site folder>/public/uploads/images folder

If aspect ratio is used the image is fit into that size in center and rest of it is clipped.

# inline

to edit single line or unformatted text elements

args:

name: some meaningful unique name for that text

object-name: the name of the object - optional

when not given it picks up data from database within the context of the page

if provided - this is normally the URL of the page if its a static page and name of the object if its an object defined using object-config

collection: name of the collection where data is stored

this argument is optional and not to be used without knowledge of underlying architecture

# layers

this xten is meant for creating layers and stacking elements on top of each other.

children have one main layer which defines the height of the whole block and then instances of layer which are above or below it.

Each child <layer> xten can have any WXML content.

args to <layer>

layer-top: optional - top margin of that layer - default 0

layer-bottom: optional - bottom margin of that layer - default 0

layer-left: optional - left margin of that layer - default 0

layer-right: optional - right margin of that layer - default 0

# line

to create a horizontal line

args:

line-style: optional - style of the line drawn - default 1px solid #000000

before-style: optional - if you want multiple lines then style of line before main line

after-style: if you want multiple lines then style of line after main line

# list

to create list of items

inherits: \_list

args:

orientation: row or column - default is column

item-gap: gap between each item

child would be what a single list item looks like

in row orientation the all elements of the list would be in the same row. in row orientation they would be one after the other. all items would be of equal width.

The admin can add and delete items in the list. The editable elements like inline, image etc should have list-child="true" to indicate to WX that that data item needs to be linked to the list context.

# navbar

This allows for creation of a navbar

args:

orientation: row or column - default is row

item-gap: gap between each item

navitem-style: style of each navitem

navitem-hover-style: hover style of each navitem

transform: CSS transform to apply to navitems

nav-cursor: cursor for navitems - default pointer

the navbar can have <navitem> or <navgroup> as children

navgroup is used for creating nested navbars. if parent navbar is horizontal the navgroup automatically becomes vertical.

Example

<navbar css-width="100%" css-margin="0.5rem"

navitem-style="[#navitem-style]"

navitem-hover-style="[#navitem-hover-style] [#navitem-transition]"

>

    <navitem href="/">Home</navitem>

    <navgroup>

        <navitem>Product</navitem>

        <navbar orientation="column" item-gap="0px" style="[#childnav-style]" navitem-hover-style="[$childnav-hover-item-style]" navitem-style="[#childnav-navbar-item-style]">

            <navitem href="introduction">Introduction</navitem>

            <navitem href="features">Features</navitem>

            <navitem href="benefits">Benefits</navitem>

            <navitem href="audience">Audience</navitem>

        </navbar>

    </navgroup>

    <navgroup>

        <navitem>Community</navitem>

        <navbar orientation="column" item-gap="0px" style="[#childnav-style]" navitem-hover-style="[$childnav-hover-item-style]" navitem-style="[#childnav-navbar-item-style]">

            <navitem href="institutes">Institutes</navitem>

            <navitem href="faculties">Faculties</navitem>

            <navitem href="leaderboard">Leaderboard</navitem>

        </navbar>

    </navgroup>

    <navgroup>

        <navitem>Resources</navitem>

        <navbar orientation="column" item-gap="0px" style="[#childnav-style]" navitem-hover-style="[$childnav-hover-item-style]" navitem-style="[#childnav-navbar-item-style]">

            <navitem href="news">News</navitem>

            <navitem href="faqs">FAQs</navitem>

        </navbar>

    </navgroup>

    <navgroup>

        <navitem>About Us</navitem>

        <navbar orientation="column" item-gap="0px" style="[#childnav-style]" navitem-hover-style="[$childnav-hover-item-style]" navitem-style="[#childnav-navbar-item-style]">

            <navitem href="about-us">About Us</navitem>

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

        </navbar>

    </navgroup>

    <navitem href="[#DEMO-DOMAIN]">Demo</navitem>

</navbar>

**todo list**

**navbar should automatically collapse on phones and expand button should come up. it should work like standard navbar in such conditions.**

**navbar items should be added via a UI instead of hardcoding in the code**

**navitem should accept any WXML instead of just text or icons**

# pos-\*

pos-left, pos-right, pos-top, pos-bottom, pos-top-left, pos-top-right, pos-bottom-left, pos-bottom-right, pos-middle

to align child elements in relation to the parent block container. It uses absolute positioning instead of flex positioning like align or center.

Each takes arguments based on name, e.g. pos-left takes left as argument which is left margin with default value of 0 while pos-top-right takes top and right as arguments as right margin and top margin with default values being 0

# quickplay

to show videos

args:

name: some meaningful unique name for that video

src: optional - same as src attribute of video tag

aspect-ratio: width to height ratio of the video - default is the original aspect ratio of the video

object-name: the name of the object - optional

when not given it picks up data from database within the context of the page

if provided - this is normally the URL of the page if its a static page and name of the object if its an object defined using object-config

collection: name of the collection where data is stored

this argument is optional and not to be used without knowledge of underlying architecture

this can be used as replacement for images for showing small few second videos. It autostarts in mute mode. Only mute videos are allowed to autostart in a browser.

This player does not have any controls. If video controls are required video-player xten should be used.

# repeat

to repeat some elements n times

args:

n - number of times to repeat something

child can be a single element that needs to be repeated

<repeat n="5">

<img src="something.png" />

</repeat>

# row

Parent of column element. Check out column description above.

# screencover

To have an element cover the whole screen. Typical example would be background for a popup.

# search-results

To show results of search type="pages"

**todo - improve the look and feel of the output**

**todo - pagination if search type is pages**

# search

to search the site pages or for a particular object

args:

type: mandatory - either "pages" or any object-type defined using object-config

pagesize: mandatory - pagesize of output

if type is not pages then argument search-view is also needed

search-view must be defined in object-config for that object-type

It creates an input element with text and redirects to search-results.xml page. This page has a default version in \_system/pages/search-results.xml. If this needs to be overridden on project basis create <site folder>/pages/search-results.xml file.

The search-results.xml page uses an xten with same name xten name="search-results". Notice the page file and xten file have same name but are meant for different purposes.

**todo: add wildcard search**

# show

to show or hide elements based on role and login status

args:

role: the role can be public, user, admin, logged\_in or other roles defined in system xml file using <roles> attribute

if the role is matched with the role of the user that element is shown otherwise it is hidden

# space

If you want to add a blank space and control how many spaces there are you need this as in HTML it automatically only shows 1 space even if in your html you may have many

args:

n: how many spaces you need

# sudo

To add style to CSS pseudo selectors

args:

name/sudo: which pseudo selector to use e.g. :link, :nth-child(even), :visited etc

selector: optional - the selector of the element on which to apply the CSS, if not specified the parent of sudo element is picked up

switch

equivalent of switch case statement in javascript

Example

<switch value="node.closest('.wx-xten-xinput').attribs.type">

    <case match="textarea">

        <textarea name="[$name]" required="[$required||false]" class="xinput" />

    </case>

    <case match="check-human">

        <check-human />

    </case>

    <default>

        <input name="[$name]" type="[$type]" required="[$required||false]" class="xinput" />

    </default>

</switch>

The value takes an expression which results in an output value. This value is matched with the match element of case. If there is a match then the children of case are displayed. If no match then default children are displayed. Default is optional, so if there is no match then nothing is shown.

# textalign

To align an inline or inline-block element horizontally in situations where vertical alignment does not matter

args:

align: left, right or center - center is default

# textedit

to allow editing of a large text block and allow formatting of that text

args:

name: some meaningful unique name for that text

object-name: the name of the object - optional

when not given it picks up data from database within the context of the page

if provided - this is normally the URL of the page if its a static page and name of the object if its an object defined using object-config

collection: name of the collection where data is stored

this argument is optional and not to be used without knowledge of underlying architecture

In the text block you can have bullet points, bold italics, add images etc

**todo: add a columns feature to allow large text to be split into newspaper type columns - columns xten was removed and that feature would be added to this xten**

# toolbar

xten to create a toolbar

args:

left, top, right, bottom: margins for placement, default is top left

Example

<toolbar name="object-list-toolbar" class="object-list-toolbar" top="[$toolbar-top||0]" right="[$toolbar-right||0]">

    <icon name="add">

        <tooltip>add [@object-label]</tooltip>

        <click name="\_eventObjectAdd">

            <transaction name="add-object" />

            <reload/>

        </click>

    </icon>

</toolbar>

# tooltip

to create a tooltip on hover of an element

args:

left: top placement in relation to the element - default 0

right: right placement - default auto

index: z-index - default 1

background: background-color - default white

padding: default is 5px

border: default is 1px solid grey

The placement of tooltip is adjusted automatically in case it would go out of the screen.

# value

any value that is edited using inline or text edit, if this value needs to be shown somewhere in read only mode then value is used

it takes same arguments as inline

it also takes an additional argument summary where you can specify how many characters to show as summary value instead of showing the whole value. This can be used to show summary and have a read more button below it which takes user to a different page where full text is available.

# video-player

A normal video player with controls.

args:

name: name of video

# xbutton

replacement for HTML button

args:

name: name of the button

type: type of button [link, submit]

href: hyperlink of the page where to take user if button is clicked - applicable if type is link

padding: default 5px

**todo: need to add more use cases where on click something else needs to be done instead of href**

# xinput

replacement for HTML input

args:

name: name of the data item

type: type of input (text, hidden, number, email, phone, textarea, password etc)

padding: default 5px

This creates an input with label. The label is editable.

**todo: auto save value in database if autosave flag is specified**

**todo: handle radio buttons and checkbox input**

**todo: handle drop down select box and multiple select inputs**

# xstyle

to apply style to an element

args:

selector: optional - css selector - if not specified parent of xstyle is used

The format of CSS you can put in the text body is same as what you would put in style tag of the element. It is different from css xten. In css xten you put css in same format as in css file with full selector and then its values like selector { ...values } while in style or xstyle you only put something like 'background-color: red'