# Wohola X

Release: 1.0 internal beta

Date: 2nd February 2024

# Chapter 1: Getting Started

## Introduction

Wohola X is an enterprise grade solution for developing large scale web based solutions quickly and easily. It can be configured to handle websites of any size, scale or traffic.

Wohola X scales automatically based on server capability. There is no need for external software or any coding for the same. It has a very small footprint, very small code base and fast page processing times so even without any additional configuration it can handle large loads on relatively smaller servers.

Wohola X is the tenth version of Wohola, a long running platform. WX was written from scratch and has been developed with the mindset that one solution should fit all and should take care of potentially ALL problems faced in their web based solutions by developers, managers, testers, server or database administrators, site administrators, users and companies.

The development in WX is done with WXML a programming language designed specifically for Wohola X coding. It is an enhanced version of HTML so HTML/CSS developers can build most of the solution without knowledge of other programming languages, technical platforms, databases or software architecture.

WX manages most of the technical complexity automatically under the covers. WX also handles most database and back-end functions automatically so WXML developers only need HTML/CSS skills in general. Anything that is not already implemented can be implemented using javascript so there is no limitation on what can be done.

WX can handle any software development requirements. Since the underlying platform is normal HTML/CSS and Javascript which means anything that can be done on any technical platform can be done on WX. There are no limitations other that what is possible to be done with any other language or technology.

Since it is designed to be a one solution fits all model for web based solutions, we are targeting Wohola X as an alternative for not only development platforms like React, Angular, Laravel, Symphony, Django, Bootstrap, jQuery and other such platforms, but also as an alternative to CMS systems like WordPress, Joomla, Drupal etc. Going one step further we plan to take Wohola X to a stage where it could be seen as viable alternative to ERP systems like SAP or MS Dynamics.

The UI in Wohola X is WYSIWYG enabled, so it is easy to develop as there is no need to build additional administrative screens and solutions built on Wohola X are easy to administer as the site administrators can simply click on the item they see on the page and change it instead of having to sift through menu items and pages in admin panels.

## Installation

1. Install Mongodb and start it: <https://www.mongodb.com/download-center/community/releases>
2. Download Wohola X
3. Configure PATH in include the wx folder

## Test the installation

To test the installation go to the folder where you installed Wohola X and run the following command from windows command prompt.

wx --file=sites/check/check.xml --port=8000

Then in your browser go to the URL : <http://localhost:8000>

You should see the welcome message if the installation is successful.

This process must be running in order for that site to be live.

To kill the process use ctrl+c on command prompt.

## Observations

* If your computer has more than one CPU you would see can see that WX starts as many processes as the number of CPUs. This is the best configuration for live environment for maximising performance.
* So WX is automatically scalable. You simply increase server capacity and the WX scales automatically. If your application is very large which requires a server farm then WX can be configured for that too. WX can also handle database scaling for large applications.
* WX is enterprise grade solution that can handle any size application. For high bandwidth enterprise solutions you can contact one of the Wohola Partners for your project requirements and they can help.

## Running in single process mode

While developing and debugging WX applications you may need to run it in single process mode. To do so add –mode=debug to your command like this:

wx --file=sites/check/check.xml --port=8000 --mode=debug

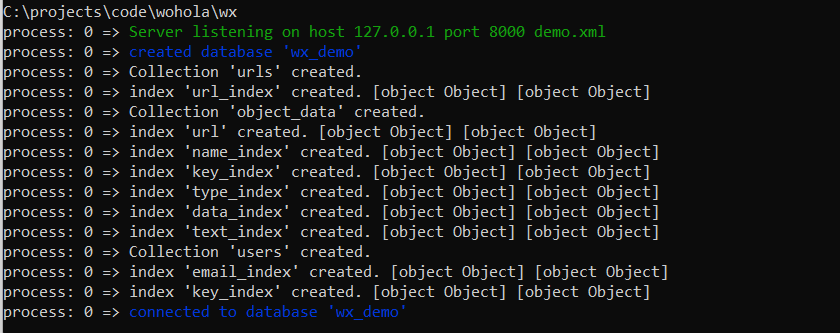
Now WX would run in a single process mode to make it easier during development phase.

## Running the demo site

To run the demo site run the following command:

wx --file=sites/demo/demo.xml --port=8000 --mode=debug

You would see following output on command prompt.



First time WX runs it creates the database, collections and the indexes needed for YOUR application.

WX manages the database internally so you don’t have to. It comes with a default DB which is sufficient for most needs of most applications. In the event you have some unlikely requirement that cannot be handled by default WX db there are mechanisms to customize the database or use a different database altogether. Those topics would be covered in advanced configuration techniques later on.

Now go to localhost:8000 to see your output. You only see some layouts with blank elements or dummy text. This is not a problem because in WX admin can manage the content after logging in. Let us try that next.

## Admin Login

To log in you need to create the first admin user. For that go back to the WX folder and run first-admin.

Enter the database name, in this case database name given in sites/demo/demo.xml file, email, password and user type as admin.



Now your have your first admin user ready.

Go to login page on the demo site and log in with these credentials. You would now see many icons and buttons appear. Hover over these icons to find out about them.

Admin can edit content, upload images and manage various elements through these buttons.

## Editing WX sites

Once logged in you can click on text to edit it. In demo these sections clearly state that you can click to edit those text elements.

On top left of image elements you see an upload icon.

Click on these icons and upload images wherever you see them.

To add items to banner, list or grid layouts on the page click on + icon at top right of these elements and items would be added.

Reduce the browser width to see that these items are mostly responsive. Further responsiveness can be configured easily.

## Observations

While developing the site developers do not need to worry about actual content. They can create the pages, page layouts and fully working functionality and the site admin can add the actual content later on after completion. This makes the development process simpler.

In general WX sites are inherently responsive.

The editing functions are all on screen, so reducing admin complexity for having to navigate through complex admin panel for large sites or for developers to have to spend time building additional screens for administrators.

# Chapter 2: Creating a new website

## Create a new website

Copy the demo folder to any folder outside of core WX system folder.

Change the name of the folder to any name you would like to give to your site. In this document let us call it ‘test-site’.

Go to test-site folder and change the name of the demo.xml file to test-site.xml.

Open the test-site.xml file in a code editor of your choice and change the name of the database to any name you would like to give. Let us give it a name wxdb\_test\_site.

Save this file. Your first website is ready.

You can run it with any of the following command.

wx --file=sites/demo/demo.xml --port=8000

wx --file=sites/demo/demo.xml --port=8000 --mode=debug

The port number can be of your choice. We would learn how to make the site live later on. During development you can choose a port which is not used by any system services and run it from localhost:port.

The database is created automatically and as you go along you would see most of the database functions are also enabled automatically.

**Notes:**

1. Name of the site can contain dashes (-) and underscores (\_) but rest of the characters must be alphanumeric only. Its best to give the name in lowercase only. Mixed case names can cause issues in browsers when used as URLs.
2. Name of the database can contain underscores (\_). Rest of the name must be alphanumeric.

## Introduction to server file

Each site in WX has a starting point which is the server file. The demo.xml or test-site.xml files we talked about are server files for those websites. These are starting points for WX and are given as --file argument when you start your WX server from command line.

For demo.xml it looks like this:

#include \_system/system.xml

#define max-width   1000px

<db name="wx\_demo" />

#include pages/home.xml

#include pages/blog.xml

#include pages/contact.xml

#include pages/login.xml

You would notice #include and #define directives here which are similar to C/C++ programming languages.

## #include directive

#include is meant to include code written in other files. Technically you could have code for your whole WX site in the server file and you would not need any other files but that is not maintainable. You need to break the code into different files to make it easier to manage. You can then include different parts of code in different files wherever needed using #include directive.

Here we are including \_system/system.xml

This file contains all the system features called XTENS (short for extensions – and X as roman numeral for ten as well as X for Wohola X). As the name suggests xtens allow enhancing the features of Wohola X and WXML. Any feature that does not already exist in WXML can be added by creating a new xten. There is no limit of what type of functionality needs to be implemented using an xten. Anything that can be done with HTML/CSS and javascript can be implemented as an xten.

We are also including various page files in the server file.

## Path resolution

The paths to these files are relative to the folder where the server file is located.

However, there is no \_system/system.xml in your site folder. WX looks for a path in the site folder first if its not found there then it looks for the same file in the WX installation folder.

## Overriding system files

The mechanism for path resolution in WX is a very powerful feature as it allows you to override system features in your site.

**Examples for this:**

1. There are icons and other files being included by default from system folder. If you want to make one or more of the icons look different then you just need to put that icon file in your site folder with same relative path and file name. Then your file would be used.
2. There are xtens which implement default features. If you need one of the xtens to do something different than default processing then you create the file for that xten in your site folder with same name and same relative path. Then for that xten your file with your custom code would be used.

In both cases only the files you override would be used from your site folder and rest would be picked up from system folder. So you can selectively override certain files or features if the defaults do not meet your project requirements. This means that you are not tied to some particular function working in a particular way. You can make any system feature work differently if you need to.

This also allows you to not have to change system files risking the core system being corrupted for your custom project needs. You can only override small parts of features that are not sufficient for your project and let WX use the remaining features as default.

## #define directive

#define is to define any variables. Once defined these variables can be used anywhere in the code like this [#var-name].

Example variable used here is max-width. It is being used like this in page files:

<page url="\_home" max-width="[#max-width]">

So you defined the value for max-width variable in one place and then can use it everywhere. In future if you want to change this value you only need to change it at one place and that change would be affected in all files.

## Adding pages to the website

Create a new xml file in your site pages folder.

Give it a name which would be the URL for that page. For example if you want that page URL to be ‘blog’ then the file name should be blog.xml, if that URL is meant to be ‘search’ then the file name should be ‘search.xml’.

Now add the following code to this file.

<page url="my-url">

    my new page

</page>

You would see #includes at the end of your server file where all the pages are included.

#include pages/home.xml

#include pages/blog.xml

#include pages/contact.xml

#include pages/login.xml

Include the file in there in similar manner.

Now you can restart the WX process and then go to localhost:port/my-url to see your page.

**Notes:**

1. The URL names must all be in lowercase and should not contain any special characters. Underscores and dashes are fine. You may have dots also.
2. The folder and naming conventions are recommendations only. In WX there is no restriction on folder structure, database structure or file names. You can give any names or structures you want which you would learn as you use WX. The recommendations are for purposes of clarity while getting started or in some cases for allowing best use of core WX features.

## Header and footer

First let us look at the code for blog page to understand page code structure.

<page url="blog" max-width="[#max-width]">

    #include includes/header.xml

    <container name="page-body">

        blog page

    </container>

    #include includes/footer.xml

</page>

Here you see max-width variable being used. It would set the maximum page width to the value defined using #define. The page would be centred. If browser width is more than this width then the rest of the space would show background for the body. If its less than the max-width then page would span the whole width of the browser.

Also you see header.xml and footer.xml being included. If you have some code which would be used in more than one place then you should separate out that code into a separate file and include it wherever needed.

You also have container for page body. Let us understand what these page and container tags mean and where they come from.

## WXML and Xten Introduction

In this code example you see page and container tags. These are not standard HTML tags. These are xtens. There is a library of xtens that exists which can make things easier or automate certain tasks and you can develop custom xtens too.

To understand this think of WXML as nothing but standard HTML where all standard HTML tags work as it is. WX has a mechanism through which you can define new tags which can be used in normal HTML just like any other tag, with one difference, HTML tags do not do anything whatsoever, but WX tags take care of functionality including display, responsiveness, database management and anything else that needs to be done.

These new tags which can be defined in WX are called xtens and the new enhanced version of HTML that the result becomes after adding all these new tags is like a brand new programming language called WXML.

WXML is not limited to existing xtens (tags). WX has an architecture where new tags can be defined at will to take care of any features as needed or override existing xtens (tags).

**Notes:**

1. Technically you could have plain simple HTML/CSS and Javascript based application and it would work perfectly fine with WX being used as a server. But in that case you could use any other webserver like apache, IIS or nginx and you do not need WX. So while the capability to run a normal HTML/CSS, Javascript site exists in WX it is not what its meant for.
2. On the other hand since WXML is an enhanced version of HTML, CSS and Javascript (fron end and node.js backend javascript) combined into one language so all the features of HTML, CSS and Javascript are available and can be used within WXML. The xtens use HTML, CSS and Javascript to implement the required features anyway.

## Adding page to navbar

Now that you have a new page it can only be accessed from URL. To add it to navbar open includes/header.xml

You would find navbar code like this:

<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>

Add your page to this navbar in the similar manner.

**Note**

If you notice while defining the page the home page was defined like this:

<page url="\_home">

While accessing it we have given the href=”/” in the navbar. It is the same concept that is used by all other webservers where they name the home page as index.html, index.asp or index.php. The reason is that you cant have a filename with a blank name or name = ‘/’. So blank or / URL has to have a different file name. Other web servers use index while in WXML we use \_home for the same.

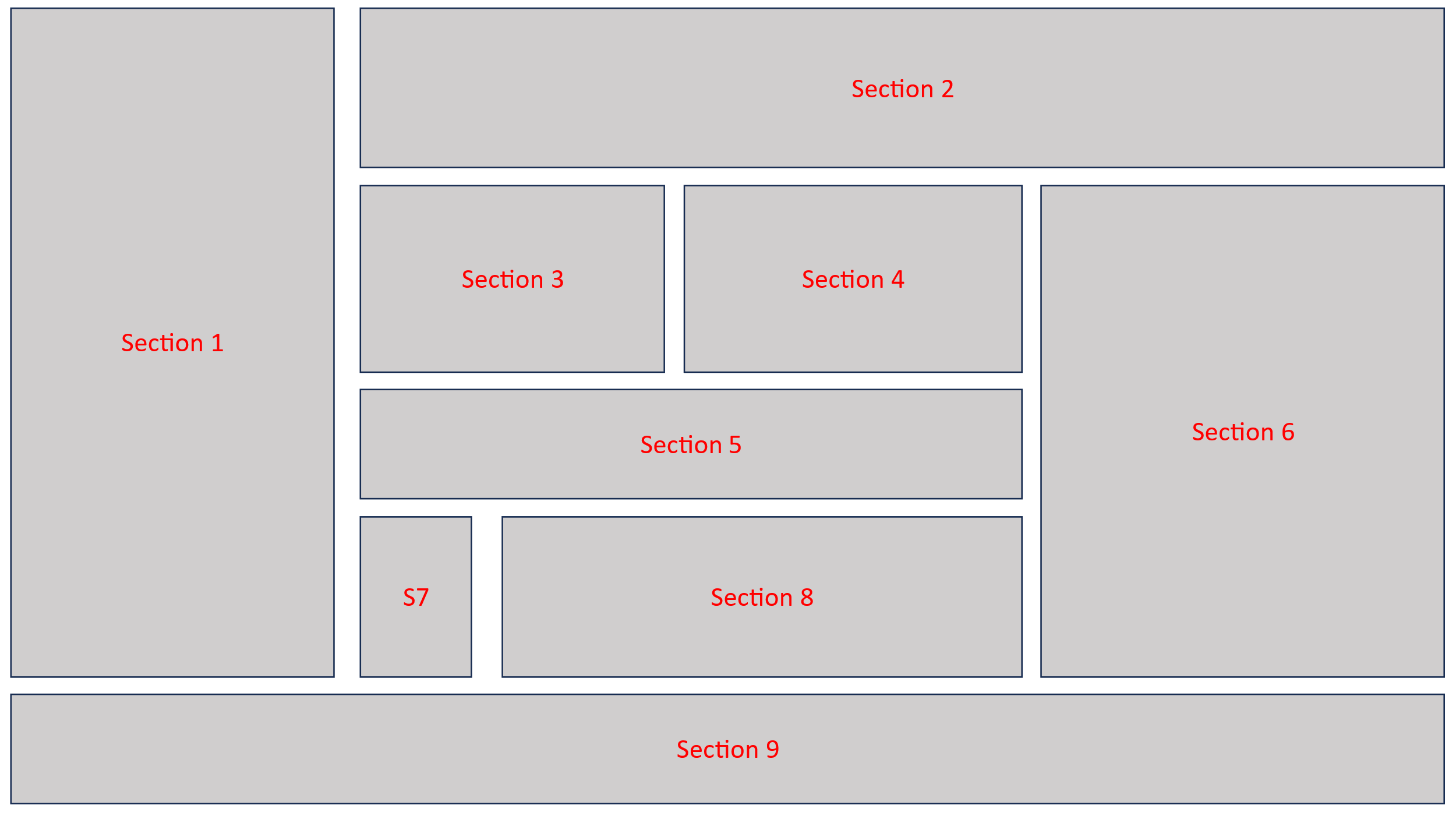
Now you have learnt how to create a new website and how to add a page to this website. Let us look at adding code to this newly created page.

# Chapter 3: Creating page layout

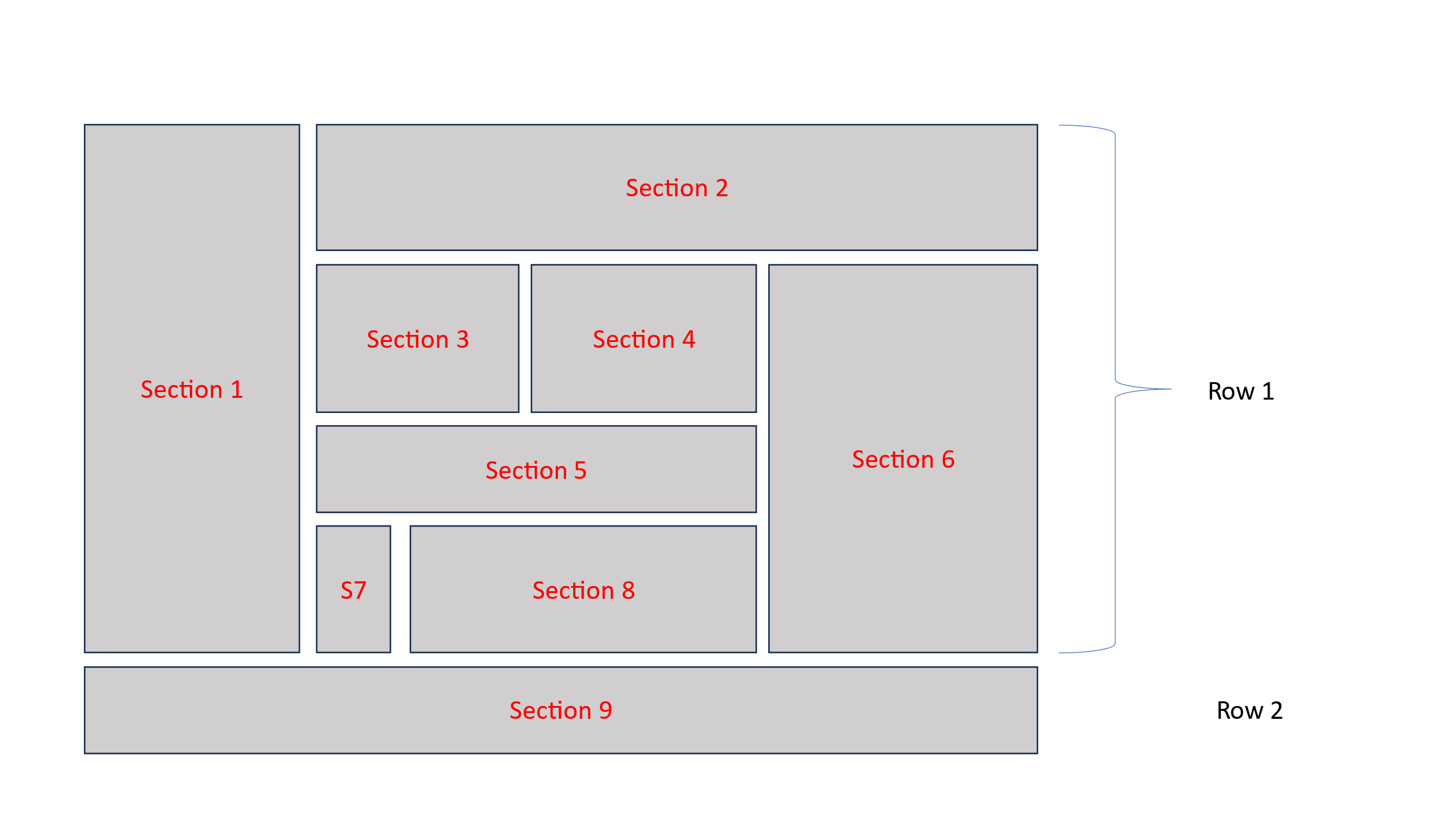
A page layout can be logically constructed by a series of nested rows and columns. For this purpose WXML has row and column xtens.

Let us create a complex page layout. For purpose of visualization and clarity we would give each element some height and a different background color.

This is the layout we would create.



Here we break down the rows and columns one by one



So we write:

<page url="layout-example" max-width="[$max-width]">

    #include includes/header.xml

    <container name="page-body">

<!-- section 1 to 8 -->

        <row>

            To be done

        </row>

<!-- section 9 -->

        <row css-background-color="yellow" css-height="100px">

            <center>

                Section 9

            </center>

        </row>

    </container>

    #include includes/footer.xml

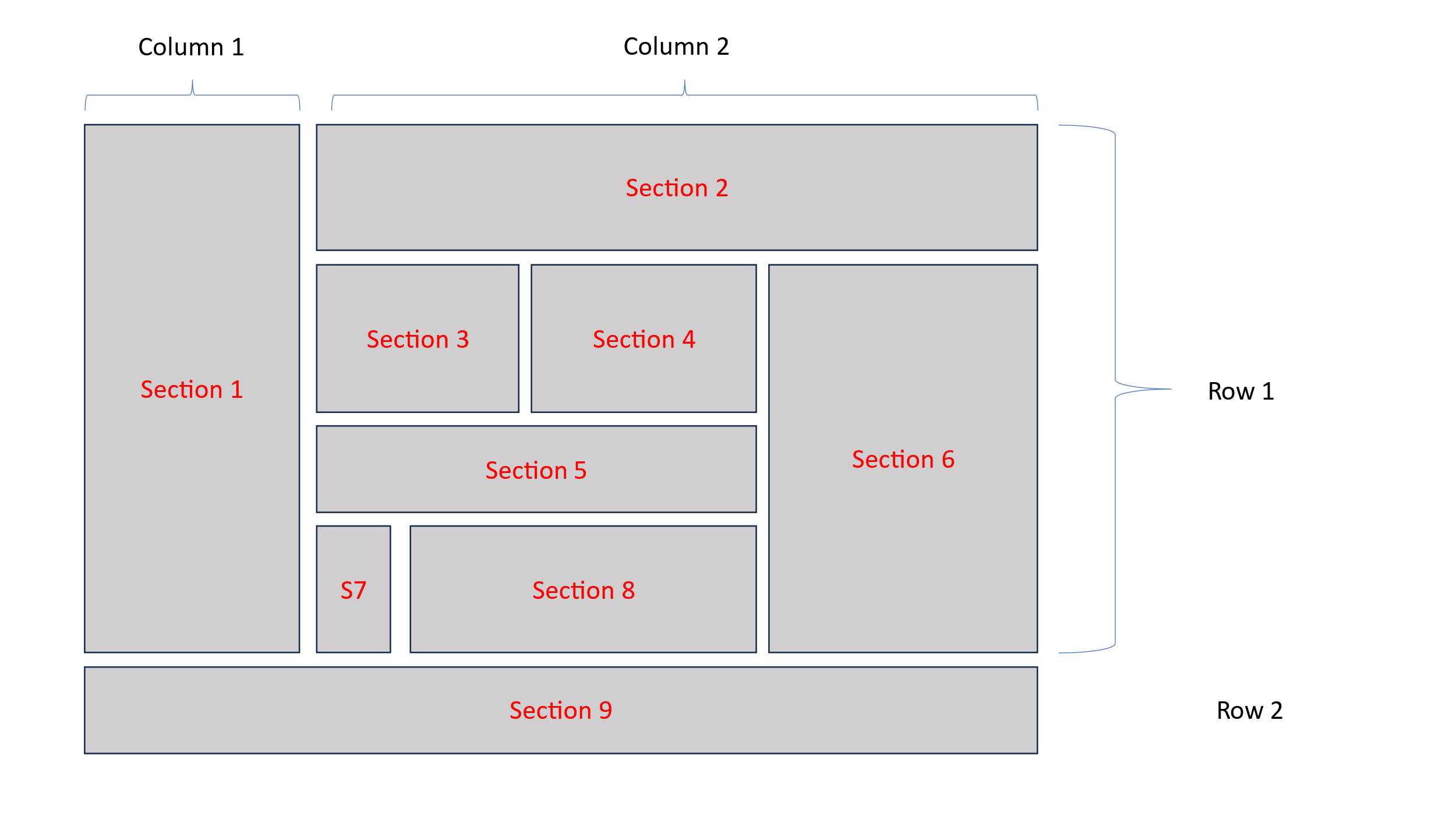
</page>

The color and height would help us see the layout.

**Notes:**

1. We have used center xten which would align the text into the middle of the row vertically and horizontally.
2. We have used css-background-color and css-height attributes to give some height and background color. Any css attributes can be used in this manner by prefixing them with css and giving them as attributes.

We break down this layout further. In Row 1 we have 2 columns:



So we write

<page url="layout-example" max-width="[$max-width]">

    #include includes/header.xml

    <container name="page-body">

        <row css-min-height="300px" css-height="fit-content">

<!-- section 1 to 8 -->

            <column css-background-color="lightgrey">

                <center>

                    Section 1

                </center>

            </column>

            <column ratio="3">

                to be done

            </column>

        </row>

        <row css-background-color="yellow" css-height="100px">

<!-- section 9 -->

            <center>

                Section 9

            </center>

        </row>

    </container>

    #include includes/footer.xml

</page>

We have given a ratio=”3” to second column because visually we can see that there is 1:3 ratio between these columns. If we omit this then all columns end up with same width. By giving ratio we can specify the relative ratio between columns with default being 1.

In this manner we keep breaking this down and end up with the final code like this.

    <container name="page-body">

        <row >

            <column css-background-color="lightgrey">

                <center>

                    Section 1

                </center>

            </column>

            <column ratio="3">

                <row css-background-color="yellow">

                    <center>

                        Section 2

                    </center>

                </row>

                <row>

                    <column>

                        <row>

                            <column css-background-color="brown" color="white" css-height="100px">

                                <center>

                                    Section 3

                                </center>

                            </column>

                            <column>

                                <center css-background-color="green">

                                    Section 4

                                </center>

                            </column>

                        </row>

                        <row css-background-color="red" css-color="white" css-height="200px">

                            <center>

                                Section 5

                            </center>

                        </row>

                        <row>

                            <column css-background-color="brown" css-color="white" css-height="200px">

                                <center>

                                    Section 7

                                </center>

                            </column>

                            <column ratio="5">

                                <center css-background-color="green">

                                    Section 8

                                </center>

                            </column>

                        </row>

                    </column>

                    <column css-background-color="blue">

                        <center>

                            Section 6

                        </center>

                    </column>

                </row>

            </column>

        </row>

        <row css-background-color="yellow" css-height="100px">

            <center>

                Section 9

            </center>

        </row>

    </container>

Notice how we removed other heights and only gave heights to section 3, 5 and 7 and rest of the columns automatically resized to fill the layout. In real project you should not give hard coded heights. With row and column xtens the column heights would automatically be adjusted as per the content height so there is no need to specify hardcoded heights.

Hardcoding is to be avoided in general except in very rare cases.

Also try to reduce the browser width to see that this is a responsive layout. Further responsiveness can be improved which we would look into when delve into more advanced topics.

The code for this layout example can be found in demo/pages/layout-example.xml

The output looks like this



To be updated further shortly.