

Xway Flexbox Tutorial

Contents

- Contents 3**
- Exploring Flexbox 5**
 - Introduction 5
 - Creating a flex container 5
 - Adding flex items 6
 - Configuring flex layout 7
 - Wrapping, sizing, and alignment 7
 - Proportional sizing 10
 - Secondary alignment 11
 - Column layouts 13
 - Vertical centering 15
 - Pinning a footer 15
 - Images and flex layout 16
 - Margins 18
- Adding flex layout to Ambient 21**
 - Pinning the Ambient footer 21
 - Responsive layout (Ambient About page) 22
- Further information 27**
 - Xway tutorials 27
 - Xway user guide 27
 - Xway discussion forum 27
 - Xway resources 27
 - Other 27

Exploring Flexbox

Introduction

Flexbox (CSS Flexible Box Layout) is a CSS layout model that can be used to position boxes flexibly on a page. A parent container box, known as a *flex container*, contains child items which are known as *flex items*. Flex items are laid out within their container on a horizontal or vertical axis—in rows or columns.

This is the third in a series of Xway tutorials, following the *Xway Tutorial*, which provides a general introduction to Xway, and the *Xway Menu Tutorial*, which introduces navigation menus. In the first chapter of this tutorial, we explore some of the options that Flexbox provides. The next chapter shows how we can use Flexbox in the *Ambient 2* and *Ambient Light 2* documents that are created by the *Xway Tutorial* and *Xway Menu Tutorial*.

*Note: This tutorial has been updated for Xway 1.1. If you are using an earlier version of Xway, please update to the latest version. Click on **About Xway** (in the Xway menu) if you are not sure which version of Xway you are using.*

Creating a flex container

Our first step is to create a flex container. Flex containers are a special kind of container box. They are similar to standard (text) container boxes, except that they cannot contain text at the top level (in Xway). You cannot enter text directly within a flex container, but you can enter text within a flex container's child boxes.

Creating a flex container is simple. Start by opening a new document to use as a scratchpad. Insert an ordinary container box by choosing **Box** from the **Insert** menu, or by clicking on the **Insert Box** button in the **Toolbar**. Now look at the **Style** section in the **Box Inspector**. Near the bottom of this section there is a **Flex Container** checkbox. Click on this to convert the box into a flex container. After you do this, you will see that Xway has added a **Flex Container** section to the **Box Inspector**, immediately below the **Style** section. This contains controls that

determine the layout of boxes within the flex container. To see how these work, we need to add some boxes. First, however, let's rename the flex container to make it easier to identify within the Site panel: do this by typing `flex-container` in the Name/ID field at the top of the Box Inspector.

Adding flex items

Now let's add some flex items. Click within the flex container (or press Command-Return) to get into editing mode. This looks and behaves pretty much like editing mode in standard text containers, except that there isn't a text insertion point. You can see that you're in editing mode from the fact that the box has a solid outline (black against a light background) and the mouse cursor changes to an insertion symbol when it's over the box. You can insert boxes within a flex container in the same way that you would insert them within a text container. Insert five boxes by clicking on the Insert Box button in the toolbar.

The first thing you will notice is that the boxes you insert are not stretched across the width of the container (as happens with boxes in a text container), but sit on a line next to each other. Let's make it easier to see the boxes by giving them contrasting background colors—to set a background color, select a box and choose a color from the Background Color popup in the Background section of the Box Inspector.



The boxes that we have inserted are *flex items*. A flex item is simply a child of a flex container. It doesn't have to be a container box. You can also insert images, videos, audio boxes, iframes—essentially, any box except for markup items (these can be inserted within a parent flex item if needed). You might wonder what is setting the width of these items. If you look at the Box Inspector, you will see that their width is undefined, but they have a minimum width of 100px. If you delete this minimum width, you will see that the width shrinks to nothing; do this for one of the items, then choose Undo Minimum Width from the Edit menu (or use the shortcut ⌘Z). This illustrates another characteristic of flex items: if their width is undefined (and not constrained by a minimum

width), it is flexible and grows or shrinks to fit their contents. This is similar to the way that undefined height normally behaves (and to the way that undefined width behaves with absolute-positioned items).

Configuring flex layout

Flexbox can be used to create horizontal or vertical layouts, with boxes distributed horizontally (in rows) or vertically (in columns). Let's start by looking at horizontal (row) layouts. This is the default direction for flex containers, but most of the same principles apply equally to vertical layouts, which we'll look at later. Layout properties can be set on flex containers, using controls in the **Flex Container** section of the **Box Inspector**, and on flex items, using controls in the **Flex Item** section of the **Box Inspector**. Properties that are set on flex containers affect the general layout of flex items, while properties that are set on flex items apply to those particular items.

Wrapping, sizing, and alignment

Flex-container properties are used to control how flex items are distributed in rows and columns. These are similar to properties that control how text is laid out: boxes can wrap (as text does), they can grow and shrink (as text sometimes does in justified layouts), and there are alignment controls that are similar to text-alignment controls.

To explore these behaviors, let's give the flex items a defined width: select them (click on one and choose **Select All** from the **Edit** menu or use its shortcut, ⌘A) and type 200px in the **Width** field in the **Dimensions** section of the **Box Inspector**. No surprises here: the boxes are now 200 pixels wide. Now set their width to 400px, and you will see that they are a bit wider, but they're not twice as wide: their width has been compressed so that they fit into their parent container.



Look at the **Flex** popup in the **Flex Item** section of the **Box Inspector**, and you will see that it is set to **Initial**. The default (initial) behavior of flex items allows them to shrink so that they fit within a line. Click on the popup and choose

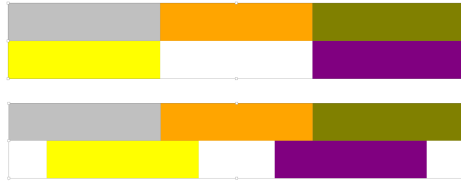
None instead of Initial, and you will see that the boxes no longer shrink (they are inflexible). Instead, they extend beyond the width of their container. Undo this change, so that the Flex popup says Initial again.

Now select the parent container: you can do this by clicking on it in the Site panel, or you can choose **Select Parent** from the Edit menu or use its shortcut `⌘+⬆` (Command-Option-Uparrow). Look at the Flex Container section in the Box Inspector, and you will see that **Wrap** is set to **Don't Wrap**. If you change that to **Wrap**, you will see that instead of shrinking to fit within a line, the boxes wrap onto successive lines. This is a very useful behavior, because it allows us to create layouts that will work well on different devices. Try shrinking the page width by dragging the divider between the Page view and the Inspector panel, and you will see that the boxes wrap when they no longer fit within the available width. You can also switch to Xway's Web view (and resize that in the same way) to confirm that this is what will happen in a browser. Set the Page view back to its normal width when you've done this.



We might also want to control the way that flex items are aligned within each row. The default behavior (seen here) is that they are aligned from the start of each line, which is similar to left alignment for text. You can change this by selecting a different alignment from the **Justify Content** popup in the Flex Container section of the Box Inspector. Try different alignments to see how they work: **Start**, **Center**, and **End** are pretty self-explanatory (and similar to left, center, and right text alignment), while **Space Between** and **Space Around** add space between or around each item (similar to the way that text is justified). The following screenshots show the flex container with **Justify Content** set to **Center**, **End**, **Space Between**, and **Space Around**.





Instead of adding space between or around items, perhaps we'd like the items to grow and use up the available space. Revert to the default horizontal alignment by choosing **Undefined** from the **Justify Content** popup. Then select the flex items within the container—a quick way to do this is to use the shortcuts for **Select First Child** (⌘⌘▼) followed by **Select All** (⌘A) in the **Edit** menu. Now choose **Auto** instead of **Initial** from the **Flex** popup in the **Flex Item** section of the **Box Inspector**. What this does is to change the behavior of the flex items, so that they can grow and shrink to fit the available space. You can also control this behavior by using the **Grow** and **Shrink** controls immediately below the **Flex** popup—the **Flex** popup provides commonly used combinations of grow, shrink, and basis (more on that later). In contrast to **Initial**, **Auto** allows boxes to grow as well as shrink: the boxes in Xway's **Page** view have now expanded to take up all the available width in each line.



Perhaps we would like to add a bit of space between each item, while still allowing the items to take up most of the available space. Select the parent flex container again (try using the **Select Parent** shortcut, ⌘⌘▲). Type **10px** in the **Column Gap** field to add some space between items in a row, then tab to the **Row Gap** field and type **10px** to add space between rows.



Since we now have a layout that wraps instead of shrinking items, you might wonder if we still need shrink behavior. The answer is yes. If you shrink the

Page view to see what happens on narrow devices (e.g. phones), you will see that the items start shrinking when we are down to a single item on a line and it is no longer possible to shorten the line by wrapping. This avoids horizontal scrolling.



Proportional sizing

Let's explore some of the other options that are available. Set Xway's Page view to its normal size, by dragging the divider between it and the Inspector panel. Then select one of the flex items on the second line, and set its grow value to 2 instead of 1. You should see that it grows slightly. If the Page view is wide enough, there should be two items on this line, and any unused space has been distributed between these items in a ratio of 2 to 1.



But perhaps we would like this item to be twice the width of the other item: instead of distributing *unused* space, so that one item is slightly wider than the other, can we distribute *all* the available space, so that one item is twice as wide as the other? There is a simple way to do this. Select all the flex items (§A) and choose 1 from the Flex popup. Then select a single item, and choose 2 from the Flex popup. What you should see is that the item that has a flex value of 2 is twice as wide as the other items, but all the items are now fitted within a single line. To understand this behavior, you need to understand what is meant by the **basis** property. Essentially, **basis** represents the basic width (or height in vertical layouts) of a flex item. This is the item's initial size, before it is grown or shrunk. If **basis** is set to **auto**, the basic width is taken

from the item's width value (set in the Width field). But if basis is set to a different value, that value is used as its basic width. If you choose a number from the Flex popup, basis has a default value of zero. If all the items in a flex container have a flex number value (as in our case), their basic width is zero and all the space that is in a line is divided proportionally between them.



If we want items to be sized proportionally, but we don't want them to shrink beyond a certain point, we can achieve this by setting a minimum width. Be careful when doing this: if you set a minimum width that is greater than (say) 320px, it will not be able to shrink beyond this point and users will need to scroll horizontally on small devices. 320px is reasonable, however. This is the width of an iPod Touch, and there aren't many web-page devices that are smaller than this. Select all the flex items (⌘A), and type 320px in the Minimum Width field of the Box Inspector to see how this works.



Note that if the default or actual value of Basis is not auto, any value that is in the Width field (for row layouts) or Height field (for column layouts) is irrelevant. If you delete the current Width value (400px) it will make no difference to the layout.

Basis has a default value of auto if a grow value is not specified. Otherwise it has a default value of 0.

Secondary alignment

Flex items can be aligned on both their primary axis (horizontally in the case of row layouts) and on their secondary axis (vertically in the case of row layouts). Primary alignment is controlled by Justify Content (as we've seen above), while secondary alignment is controlled by Align Items and Align Content (immediately after Justify Content in the Flex Container section of the Box Inspector). For horizontal layouts, Align Items controls the way that items are vertically aligned within a row. To see how this works, select one of the

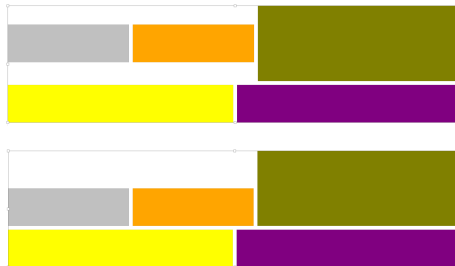
flex items (e.g. the olive-colored box) and give it a minimum height of 200px. You should see that all the items that are on the same line are now stretched to be 200 pixels high.



Select the parent flex container again, and you will see that **Align Items** is set to **Stretch** (this is the default value). If you change this to **Start**, you will see that instead of being stretched, the items that have a smaller minimum height are aligned with the top of the line.



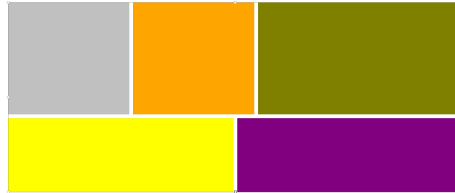
Center and **End** also do exactly what you would expect.



Baseline is less straightforward, and we'll skip it for now (see the *Xway User Guide* for more information). If you want to change the secondary alignment of a single flex item, you can do this by selecting that item and choosing a value from the **Align** popup in the **Flex Items** section of the **Box Inspector** (the default value, **Auto**, causes the item to use the alignment set for the parent container).

Align Content is similar to **Align Items**, but instead of aligning items within a line, it aligns lines within a parent container. Currently, the parent container

has flexible (undefined) height, but if you give it a defined height of (say) 500px, you will see that the default (Stretch) behavior causes lines to be stretched vertically within their parent container. This might be easier to see if you also set **Align Items** back to its default (Stretch) value, so that items are vertically stretched within the stretched lines. To set a popup back to its default value, choose **Undefined** from the bottom of the popup menu.



The following screenshot shows what happens if **Align Content** is set to **Center**: content is vertically centered within its container (in horizontal layouts).



Column layouts

Let's turn to vertical layouts. Start by duplicating the current page: select it by clicking on it in the **Site** panel and choose **Duplicate Selection** from the **Edit** menu. Close the previous page in the **Site** panel (by clicking on its disclosure arrow) and open the new page and its contents by option-clicking on its disclosure arrow.

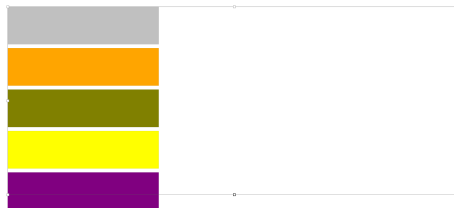
Previously, we gave one of the flex items a minimum height of 200px in order to see how the **Align** controls work. Change that back to be the same as the other items. The simplest way to do this (if you don't remember which item you changed) is to select all the items within the flex container (on the new page) and set their minimum height to 100px.

Now select the flex container and look at the **Direction** popup in the **Flex Container** section of the **Box Inspector**. This is currently set to **Row**, meaning that items are distributed horizontally in rows. Change that to **Column**, so the flex items are distributed vertically in columns. Also, set **Align Items** and **Align Content** back to their default (**Stretch**) value, if they are not already set to this, by choosing **Undefined** from their popup menus. You should see that the items are now arranged vertically in two columns.



Since we previously set one of the items to have a flex value of 2, and since the parent container has a defined height of 500px, the item that has a flex value of 2 is grown to almost twice the height of the other items within its column. (There isn't enough space for it to grow to exactly twice the height, since each of the items has a minimum height of 100px, and there is a 10px row gap between items.)

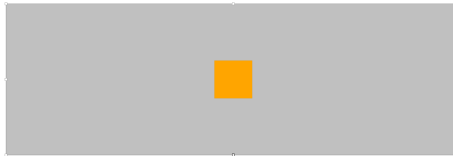
If we turn off **Wrap** for the container, the items are arranged in a single column, as you would expect. There is no extra space to be distributed, so all the items are the same height. Their minimum height is set to 100px, and there is a 10px row gap between each item, so they actually extend beyond the height of the parent container.



If you remove the defined height of the container (delete it from the **Height** field in the **Dimensions** section of the **Box Inspector**), the container expands to contain its content.

Vertical centering

Column layouts are particularly useful in two cases: vertically centering an item within its parent, and pinning a footer to the bottom of a page. Vertical centering is something that is otherwise difficult to do in CSS layout. Insert a new page in the test document (choose **Page** from the **Insert** menu), and close the previous page in the **Site** panel by clicking on its disclosure arrow. Now insert a box on this page. Give it a height of 400px, and set a background color on it. Then set it to be a flex container, and choose **Column** from the **Direction** popup in the **Flex Container** section of the **Box Inspector**. Insert a second box within the parent container, and give it a different background color. Now select the parent container again, and choose **Center** from the **Justify Content** popup. This centers the box vertically. If you also want to center the box horizontally (which is the secondary axis for a column layout), choose **Center** from the **Align Items** popup. Note that the box also shrinks to its minimum width. Previously it stretched across the width of the flex container, because the previous (default) **Align Items** value was **Stretch**.



Pinning a footer

Footers normally appear at the bottom of a page, and column layouts give us an easy way to achieve this. Insert a new page in your test document, and close the previous page in the **Site** panel. Now add three boxes, and give them contrasting background colors. Set the first box to be a header (choose **Header** from the **Type** popup in the **Box Inspector**) and set the third box to be a footer (choose **Footer** from the **Type** popup). There isn't any content to grow the height of the first two boxes, so the footer is positioned somewhere in the

middle of the page. We can change that by setting the page div to be a flex container with a column layout. Select the page div by clicking on it, turn on the Flex Container checkbox in the Style section of the Box Inspector, and select Column from the Direction popup in the Flex Container section. Now select the middle box on the page, and choose Auto from the Flex popup in the Flex Item section of the Box Inspector. This causes it to grow so that it takes up any available vertical space and pushes the footer box to the bottom of the page.



Images and flex layout

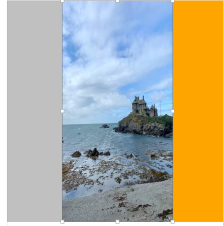
Up to now, we have used container boxes as flex items. Let's see what happens if we insert an image within a flex container. Add a new page to your test document, and close the previous page in the Site panel. Insert a box and set it to be a flex container. Then insert a couple of container boxes within the flex container and give them contrasting background colors. Now select the first of these child boxes and use the Insert Image control in the toolbar to insert an image at this point. Choose a fairly large image, such as the `anglesey.jpg` image which accompanies the main Xway tutorial. Delete the maximum width of 50% which Xway automatically adds to the image, and you should find that the image grows to its natural size (1024 x 768 in the case of the Anglesey image), and may also extend beyond the bounds of its parent container.



If you make the **Page** view narrower, to see what will happen when it is viewed on a small device, the image stays at its original size. In contrast to container boxes, images don't automatically shrink to fit the available space. But we can make that happen. The key to doing this is to give the image a minimum width. Do that by selecting the image and typing 200px in the **Minimum Width** field of the **Box Inspector**. You should see that the image shrinks to fit within the width of its parent box, and if you shrink the width of the **Page** view, the image continues shrinking until it reaches its minimum width. You will also see that the image's height shrinks along with its width so that the image preserves its original aspect ratio.



Preserving an image's aspect ratio is generally a good thing, but there is a situation in which the aspect ratio may become distorted. Select the flex item that precedes the image, and give it a minimum height of 400px. Now if you reduce the width of the **Page** view, you will see that none the three boxes, including the image, shrink to less than 400px high.



This is because the default value of the `Align Items` property is `Stretch`, so that items stretch to the height of their row, which is the height of the tallest item within that row. We can stop that happening for the image. Select it, and change the value that is set in the `Align` popup in the `Flex Item` section of the `Box Inspector`. The default value, `Auto`, causes a flex item to behave according to the way that `Align Items` is set for the parent flex container. If you change this to `Start`, or `Center`, or `End`, the image keeps its natural height and is no longer stretched.



Margins

You can use margins with flex items in the same way that you would use them in non-flex layouts. If you want to add extra space around a particular item, instead of adding the same amount of space around all items (using `Column Gap` and/or `Row Gap`), margins are the way to do this.

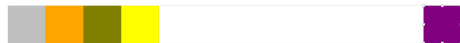
You might also recall that there is a special `auto` value for horizontal margins that causes a box to be horizontally centered, or left/right-aligned within its parent container. Xway provides an easy way to do this for non-flex items: you can choose `Left`, `Center`, or `Right` from the `Align` popup in the `Margins` and

Alignment section of the Box Inspector, and Xway adds the appropriate auto margins. Auto margins can also be used with flex items.

To see how this works, add a new page to the test document and close the previous page in the Site panel. Set the Page view back to its normal width, by dragging the divider between it and the Inspector panel. Insert a container box on the new page, set it to be a flex container, and insert five boxes within that container (give them contrasting background colors: Silver, Orange, Olive, Yellow, Purple). Now select the middle box and choose Center from the Align popup in the Margins and Alignment section of the Box Inspector. This causes the middle box to be centered while the other boxes are aligned left and right.



Alternatively, you could set the final box to be right-aligned while the first four boxes are left-aligned: choose Undo Margin Alignment from the Edit menu (so the five boxes are next to each other once more), then select the final box and choose Right from the Align popup in the Margins and Alignment section.



Save the test document, in case you want to refer back to it at a later date: call it Flexbox Tests.

Adding flex layout to Ambient

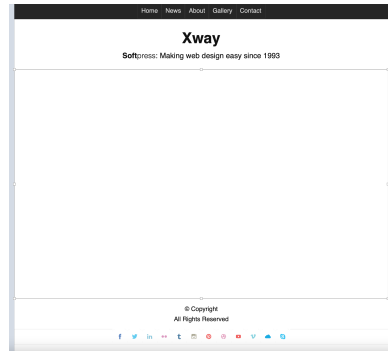
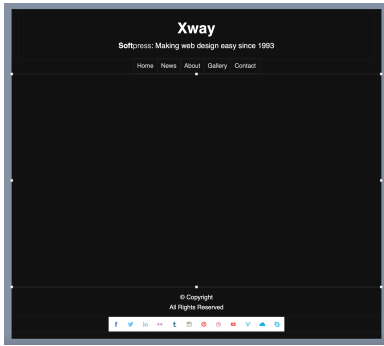
The previous chapter of this tutorial explored flex layout using a test document. Now let's apply it to a real document. If you previously worked through the *Xway Tutorial* and *Xway Menu Tutorial*, open the *Ambient 2* or *Ambient Light 2* document that you created with those tutorials. Alternatively, download *Ambient 2* or *Ambient Light 2* from our website (choose *Xway Resources* from Xway's Help menu, then click on *Tutorials* followed by *Download the tutorial documents*).

When you've opened the document, duplicate it by choosing *Duplicate* from the *File* menu. Rename the duplicate document *Ambient 3* or *Ambient Light 3*. Close the original document.

Pinning the Ambient footer

The first thing we can do is to pin the footer so that it appears at the bottom of the page, regardless of how much content there is before it. We've already done this in our scratch document. Go to the master page (*Master 1*), and open it (if it is not already open) by option-clicking on the disclosure arrow next to it in the *Site* panel. If another page is open, close it by clicking on its disclosure arrow. You should see that the page div (*page-wrapper*) contains three child boxes: *header*, *main*, and *footer*. Despite its name, the footer is somewhere in the middle of the page.

Click on *page-wrapper* in the *Site* panel and convert it into a flex container, using the *Flex Container* checkbox in the *Style* section of the *Box Inspector*. Don't worry about the transitional layout that you see in the *Page* view—we're not going to leave it like that. Go to the *Flex Container* section of the *Box Inspector*, and choose *Column* from the *Direction* popup. We now have a vertical layout that looks pretty much the same as it did originally. The next thing we need to do is to select the *main* item and set its *Flex* property to *Auto*, using the *Flex* popup in the *Flex Item* section of the *Box Inspector*. This causes the *main* item's *Grow* value to be set to 1, so that it grows to take up the available vertical space. The footer is now pinned to the bottom of its page.

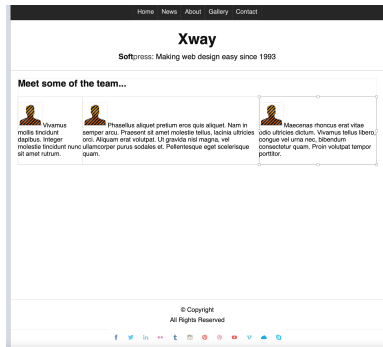
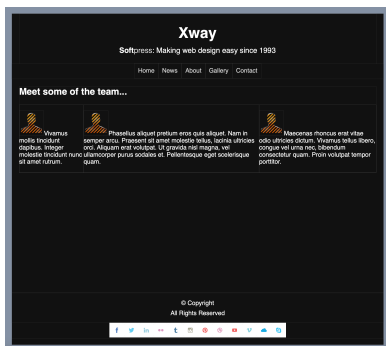


Responsive layout (Ambient About page)

Now let's create a responsive layout for the **About** page. Close the **Master 1** page by clicking on its disclosure arrow in the **Site** panel, then switch to the **About** page, and open it by option-clicking on its disclosure arrow. This page is already flexible, but apart from growing and shrinking in width, the layout is essentially the same on all devices. This can result in long lines of text on wide devices. To prevent this from happening, let's redesign the page so that it uses boxes that are laid out horizontally on large screens, but wrap vertically on smaller devices.

Select **section1** in the **Site** panel, and click somewhere inside—but not at the start of—the first line of text (“Meet some of the team...””) to get a text insertion point. Then insert a box by clicking on the **Insert Box** tool in the toolbar. This should appear immediately below the first line of text (“Meet some of the team...”). Now convert this box (**item1**) into a flex container, and rename it **flex-container**. Insert three boxes within the flex container. Now select the first paragraph that is below the flex container: you can do this by triple-clicking on it, or by selecting the user image at the start of the paragraph and choosing **Select Parent** (shortcut: `⌘+⌥+⬆`) from the **Edit** menu. You should see that the entire paragraph, including the user image, is highlighted. We want to move this paragraph into the first of our flex items, so choose **Cut** from the **Edit** menu (shortcut: `⌘+X`) to cut it from the **Page** view into the clipboard. Then paste it within the first flex item (**item1**): click on that item,

then click a second time to put it into editing mode, and choose **Paste** from the **Edit** menu (shortcut: ⌘V). Do the same with the second and third paragraphs: cut and paste them into the second and third flex items.

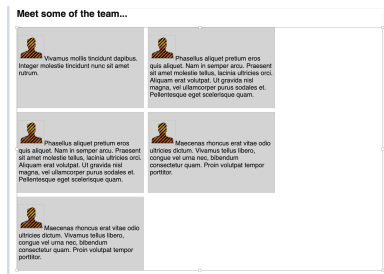
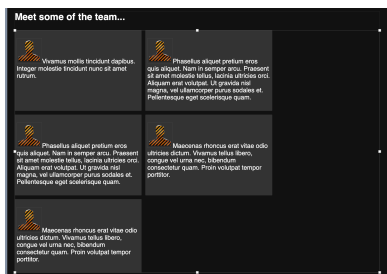


The layout looks a little messy at this point. The first thing to notice is that the three boxes have different widths. Why is this? If you click on any (or all) of these boxes, you can see from the **Box Inspector** that they don't have a defined width. In normal layouts, this causes them to expand to fit within their parent, but in flex layouts it causes their width to be determined by their content. So the box that contains the most text is also the widest. This uses the available space very efficiently, but it looks untidy. We can change that by giving the boxes a defined width. Select all three flex items (**item1**, **item2**, and **item3**) and set their width to **320px**, using the **Width** field in the **Box Inspector**. We also need to add some space so that text in one box doesn't run directly up against text in an adjacent box. To do this, select the parent flex-container and type **10px** in the **Column Gap** field in the **Flex Container** section of the **Box Inspector**.

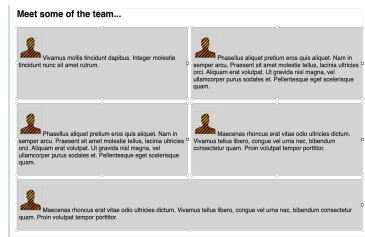
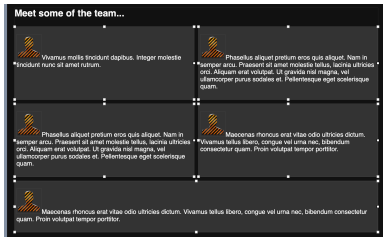
To create more visual separation between the boxes, let's give them a distinct background color. Select the three boxes, and click on the **Background Color** popup in the **Background** section of the **Box Inspector**. If you are working with a dark background (*Ambient* rather than *Ambient Light*), choose **Other** from the bottom of this popup. Create a color called **Light Charcoal** and give it a hex value of **333333**. If you are working with a light background (*Ambient Light*), choose **Light Gray** from the **Background Color** menu—or choose **Other** and create it if it doesn't already exist (you can create it by typing **Light Gray** in the

Name field and accepting the hex value that Xway adds automatically). Now create some space between the text and the edge of each box by setting 5 pixels of left, right, top, and bottom padding on each box (using the padding fields in the Padding section of the Box Inspector).

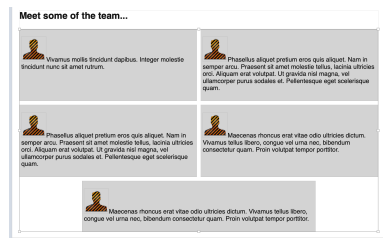
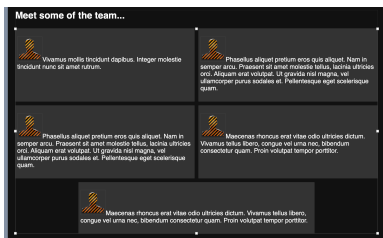
For the sake of this tutorial, let's suppose that we have five people on our team. Duplicate one of the boxes a couple of times, or duplicate two boxes (select it/them and choose **Duplicate Selection** from the **Edit** menu). We now have five boxes that are fitted into the available space, and you can see by shrinking the width of the page that the boxes get pretty narrow on small devices. To fix that, select the parent flex container and choose **Wrap** from the **Wrap** popup in the **Flex Container** section of the **Box Inspector**. Now the boxes wrap onto successive lines, instead of shrinking. We need some vertical space between each line, so enter 10px in the **Row Gap** field.



There are a few things we can do to improve this layout. The first is to choose a setting that allows the flex items to grow as well as shrink. Click on one of the items, and then select the others by choosing **Select All** from the **Edit** menu (shortcut: ⌘A). Now choose **Auto** from the **Flex** popup in the **Flex Item** section of the **Box Inspector**. This sets a **Grow** value of 1, allowing the boxes to grow so that they take up any unused space. If your **Page** view is set to its maximum width (the divider between it and the **Inspector** panel is in its normal position), you should see that the boxes are arranged in three rows, with two boxes in the first two rows and a single box in the final row.

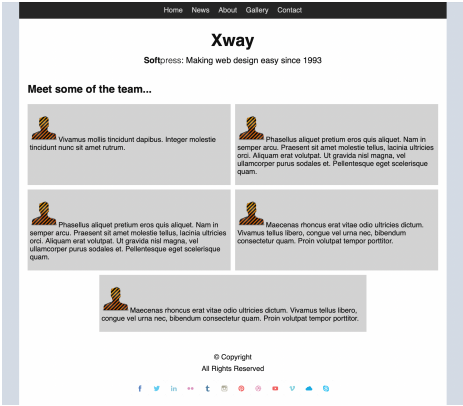
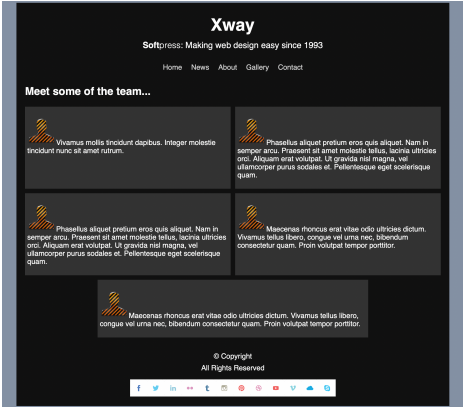


We are now back in a situation where the final box contains long lines of text. We can prevent that from happening by setting a maximum width for each box. While the boxes are still selected, enter 600px in the Maximum Width field. Then select the parent flex container, and choose Center from the Justify Content popup.



Then switch to Xway's Web view and adjust its width to see how the page will look on different devices. Save the document: name it Ambient 3 or Ambient Light 3 if you haven't already done so.

This is what the *About* pages look like in a desktop browser and on a phone:



Further information

Xway tutorials

This is the third in a series of Xway tutorials, following the *Xway Tutorial*, which provides a general introduction to Xway, and the *Xway Menu Tutorial*, which introduces navigation menus. All these tutorials, and the documents they create, are available from our website.

There are also short tutorials (tagged “tutorial”) in the Xway discussion forum (see below).

Xway user guide

Xway comes with a detailed reference manual, which you can access by selecting Xway User Guide from Xway’s Help menu. This contains information on a wide range of topics, including a chapter on Flexbox.

Xway discussion forum

Choose Support and Community from Xway’s Help menu to visit our Xway discussion forum. This contains Xway-related news and discussions, and is a good place to ask questions and get help from us and from other Xway users.

Xway resources

Choose Xway Resources from the Help menu to visit the Xway Resources section of our website. This contains links to the latest versions of this tutorial, along with other tutorials and completed tutorial documents. There are also other resources such as templates.

Other

For a general introduction to Flexbox, with helpful illustrations, see *A Complete Guide to Flexbox* (<https://css-tricks.com/snippets/css/a-guide-to-flexbox/>).