Intro to Flexbox

Intro to Flexbox

Flexbox (Flexible Layout) is a relatively new module in the CSS specification, which offers designers a way to design, align, and even center content in web pages even when we don’t know the context of those pages or the screen sizes they’ll be displayed on.

Intro to Flexbox

Flexbox (Flexible Layout) is a relatively new module in the CSS specification, which offers designers a way to design, align, and even center content in web pages even when we don’t know the context of those pages or the screen sizes they’ll be displayed on.

The key concept is that a flexible container has the ability to alter its child items’ size, shape and position as the context they’re displayed in changes.

Intro to Flexbox

Flexbox (Flexible Layout) is a relatively new module in the CSS specification, which offers designers a way to design, align, and even center content in web pages even when we don’t know the context of those pages or the screen sizes they’ll be displayed on.

The key concept is that a flexible container has the ability to alter its child items’ size, shape and position as the context they’re displayed in changes.

Flexbox is also direction-agnostic; that is, it doesn’t work only in the horizontal, like floats do.

Intro to Flexbox

Flexbox (Flexible Layout) is a relatively new module in the CSS specification, which offers designers a way to design, align, and even center content in web pages even when we don’t know the context of those pages or the screen sizes they’ll be displayed on.

The key concept is that a flexible container has the ability to alter its child items’ size, shape and position as the context they’re displayed in changes.

Flexbox is also direction-agnostic; that is, it doesn’t work only in the horizontal, like floats do.

Flexbox is being supplemented by the Grid layout specification, which is another extension of CSS. Grid has wide browser support, and it’s what we’ll learn next.

Flexbox vs. Grid layout

Flexbox is meant mainly for one-dimensional layout, where you're solving a problem of items needing to move, flex or wrap in a predictable way, but in one direction only.

Flexbox vs. Grid layout

Flexbox is meant mainly for one-dimensional layout, where you're solving a problem of items needing to move, flex or wrap in a predictable way, but in one direction only.

Grid layout is a full two-dimensional grid, in which items can be placed exactly into spots where we need them to be.

Flexbox vs. Grid layout

Flexbox is meant mainly for one-dimensional layout, where you're solving a problem of items needing to move, flex or wrap in a predictable way, but in one direction only.

Grid layout is a full two-dimensional grid, in which items can be placed exactly into spots where we need them to be.

Combining flexbox and grid layouts, and even working in some floats when necessary, allows us to take full advantage of the tools we have for layout.

Flexbox basics: terminology

Flexbox is a module in CSS, which means it has an entire set of properties meant to be set on different parts of a flex container and its flex items.

Flexbox basics: terminology

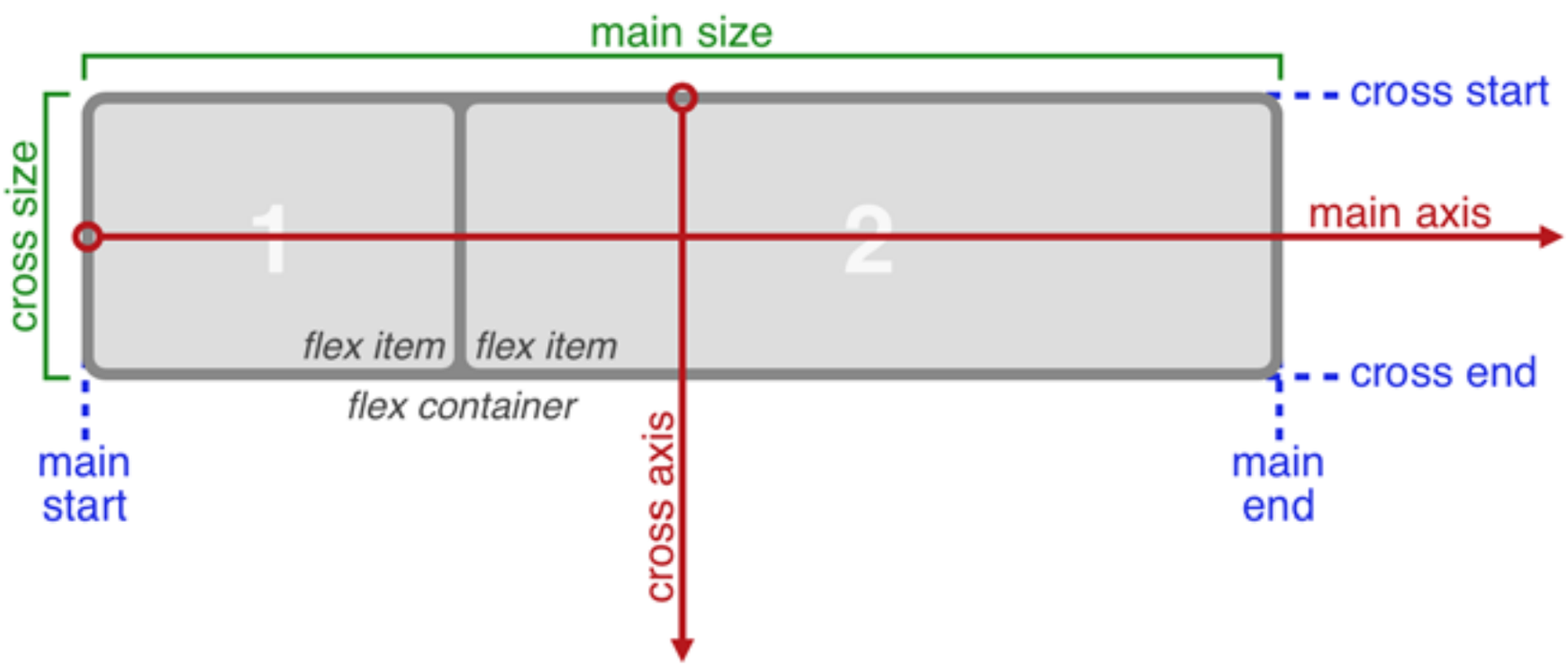
Flexbox is a module in CSS, which means it has an entire set of properties meant to be set on different parts of a flex container and its flex items.

The flex container is defined not by left-right or top-bottom properties, but by its **flex-flow** direction.

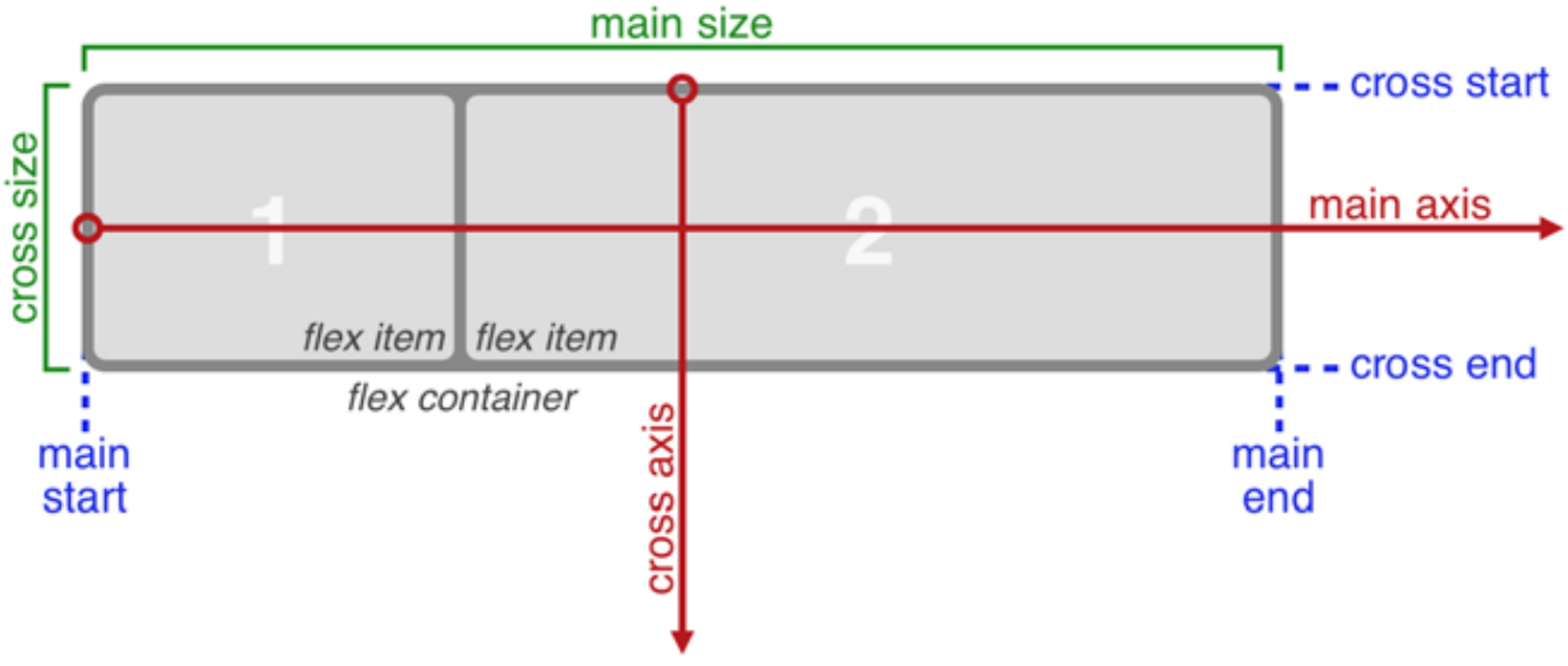
Flexbox basics: terminology

Flexbox is a module in CSS, which means it has an entire set of properties meant to be set on different parts of a flex container and its flex items.

The flex container is defined not by left-right or top-bottom properties, but by its **flex-flow** direction.



Flexbox basics: terminology



Basically, items will be laid out following either the main axis or the cross axis. Although the cross axis is always perpendicular to the main axis, the main axis is not always horizontal; you control that with the flex-direction property.

Flexbox basics: terminology

* main-axis: The main axis along which flex items are laid out. Direction is controlled by the flex-direction property.

Flexbox basics: terminology

* main-axis: The main axis along which flex items are laid out. Direction is controlled by the flex-direction property.

* main-start | main-end: Flex items' default ordering in the container goes from main-start to main-end

Flexbox basics: terminology

* main-axis: The main axis along which flex items are laid out. Direction is controlled by the flex-direction property.

* main-start | main-end: Flex items' default ordering in the container goes from main-start to main-end, although you can edit this

Flexbox basics: terminology

* main-axis: The main axis along which flex items are laid out. Direction is controlled by the flex-direction property.

* main-start | main-end: Flex items' default ordering in the container goes from main-start to main-end, although you can edit this

* main-size: The width or height of a flex item is the item's main size, depending on whether width or height is the main dimension.

Flexbox basics: terminology

* main-axis: The main axis along which flex items are laid out. Direction is controlled by the flex-direction property.

* main-start | main-end: Flex items' default ordering in the container goes from main-start to main-end, although you can edit this

* main-size: The width or height of a flex item is the item's main size, depending on whether width or height is the main dimension.

* cross-axis: The axis that's perpendicular to the main axis.

Flexbox basics: terminology

* main-axis: The main axis along which flex items are laid out. Direction is controlled by the flex-direction property.

* main-start | main-end: Flex items' default ordering in the container goes from main-start to main-end, although you can edit this

* main-size: The width or height of a flex item is the item's main size, depending on whether width or height is the main dimension.

* cross-axis: The axis that's perpendicular to the main axis.

* cross-start | cross-end: Flex items are ordered on the cross axis starting at cross-start and going to cross-end

Flexbox basics: terminology

* main-axis: The main axis along which flex items are laid out. Direction is controlled by the flex-direction property.

* main-start | main-end: Flex items' default ordering in the container goes from main-start to main-end

* main-size: The width or height of a flex item is the item's main size, depending on whether width or height is the main dimension.

* cross-axis: The axis that's perpendicular to the main axis.

* cross-start | cross-end: Flex items are ordered on the cross axis starting at cross-start and going to cross-end

* cross-size: The width or height of a flex item is the item's cross size, depending on whether width or height is the main dimension

Flex container properties

Parent container: This is the flex container that contains flex items.
Flex items become items by being child elements of the container.

Flex container properties

Parent container: This is the flex container that contains flex items. Flex items become items by being child elements of the container.

To make a container a flex container, just add a class to it and give it a value of display-flex:

Flex container properties

Parent container: This is the flex container that contains flex items. Flex items become items by being child elements of the container.

To make a container a flex container, just add a class to it and give it a value of display: flex;

```
.container {
    display: flex;
}
```

(do this in course example)

Flex container properties

Flex-direction will define the main axis.

Flex container properties

Flex-direction will define the main axis.

```
.container {
    flex-direction: row;
}
```

Flex container properties

Flex-direction will define the main axis.

```
.container {
    flex-direction: row;
}
```

Other values: row-reverse, column, column-reverse

(show code example)

Flex container properties

Flex-wrap will allow flex items to wrap onto a new line

Flex container properties

Flex-wrap will allow flex items to wrap onto a new line

```
.container {
    flex-wrap: nowrap;
}
```

Flex container properties

Flex-wrap will allow flex items to wrap onto a new line

```
.container {
    flex-wrap: nowrap;
}
```

Other values: wrap, wrap-reverse

(show code example)

Flex container properties

Flex-flow is a shorthand for the flex-direction and flex-wrap properties

Flex container properties

Flex-flow is a shorthand for the flex-direction and flex-wrap properties

```
.container {
    flex-flow: <flex-direction> || <flex-wrap>
}
```

Flex container properties

Flex-flow is a shorthand for the flex-direction and flex-wrap properties

```
.container {
    flex-flow: row nowrap;
}
```

(show example)

Flex container properties

Flex-justify will define alignment along the main axis. It’s an easy way to distribute extra space among items on the main axis, and control somewhat how they wrap in extra space.

Flex container properties

Flex-justify will define alignment along the main axis. It's an easy way to distribute extra space among items on the main axis, and control somewhat how they wrap in extra space.

```
.container {
    justify-content: flex-start | flex-end | center | space-between | space-around
}
```

(show example)

Flex container properties

```
.container {
    justify-content: flex-start;/*default value*/
    justify-content: flex-end;/*moves items to end of
            main axis*/
    justify-content: center;/*centers all the items
            along the main axis*/
    justify-content: space-around;/*puts equal amounts
            of space around each item*/
    justify-content: space-between;/*puts equal amounts
            of space between each item, moving first and
            last item to outer edges*/
}
```

Flex container properties

align-items will define alignment along the cross axis. Think of it as justify-content for the cross axis.

Flex container properties

align-items will define alignment along the cross axis. Think of it as justify-content for the cross axis.

```
.container {
    align-items: flex-start | flex-end | center | baseline | stretch;
}
```

(show example)

Flex container properties

align-content will align a flex container's lines when there's extra space in the cross-axis. It won't work when there's only one line of content, so you'll have to set flex-wrap: wrap; for it to work.

```
.container {
    flex-wrap: wrap;
    align-content: flex-start | flex-end | center | space-between | space-
around | stretch;
}
```

(show example)

Flex item properties

* Flex **item** properties refer to flex items, which are children of flex containers.

Flex item properties

* Flex **item** properties refer to flex items, which are children of flex containers.

To make an item a flex item, just put it in an element that's already a flex container. (Flex container: any element that has display:flex; applied to it.)

Flex item properties

* Flex **item** properties refer to flex items, which are children of flex containers.

To make an item a flex item, just put it in an element that's already a flex container. (Flex container: any element that has display:flex; applied to it.)

Properties include order, flex-grow, flex-shrink, flex-basis, flex(shorthand) and align-self.

Flex item properties

The order property allows you to change the order of a flex item. The order is changed by using an integer, where the default order value is 0. (Integers are numbers without fractions, including negative numbers.)

Flex item properties

The order property allows you to change the order of a flex item. The order is changed by using an integer, where the default order value is 0. (Integers are numbers without fractions, including negative numbers.)

```
.item {
  order: <integer>;
}
```

(show code example)

Flex item properties

The flex-grow property allows for a flex item to grow if necessary. It will accept a value that serves as a proprotion; the default value is 0 and negative numbers are invalid. The flex-grow property will accept fractional (decimal) values.

Flex item properties

The flex-grow property allows for a flex item to grow if necessary. It will accept a value that serves as a proprotion; the default value is 0 and negative numbers are invalid. The flex-grow property will accept fractional (decimal) values.

```
.item {
  flex-grow: <number>;
}
```

(show code example)

Flex item properties

The flex-shrink property is rarely used, but you may run across it in code examples. It allows for a flex item to shrink if necessary. The default value is 1, and negative numbers are invalid.

Flex item properties

The flex-shrink property is rarely used, but you may run across it in code examples. It allows for a flex item to shrink if necessary. The default value is 1, and negative numbers are invalid.

```
.item {
  flex-shrink: <number>;
}
```

(show code example)

Flex item properties

The flex-basis property will set the basic width for items before remaining content sizes are distributed. The basis is the amount of space an item should take up; if it becomes **narrower** than that basis size, remaining space will be redistributed.

Flex item properties

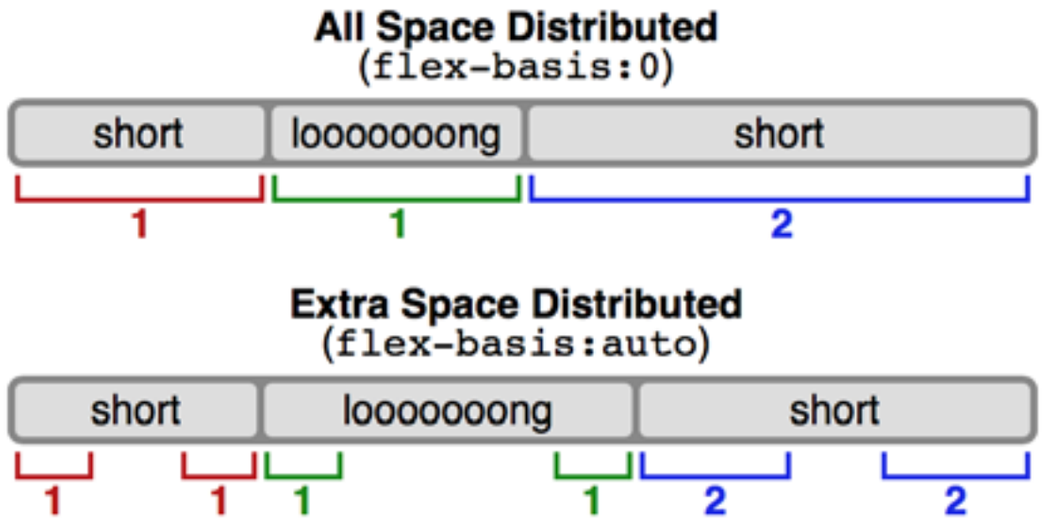
The flex-basis property will set the basic width for items before remaining content sizes are distributed. The basis is the amount of space an item should take up; if it becomes **narrower** than that basis size, remaining space will be redistributed.

Flex basis can take a length (20em; 5%, 100px) or a keyword (auto, content, max-content, min-content). Auto is the only widely supported keyword right now.

Flex item properties

Flex-basis: 0; will ignore extra space around content.

Flex-basis: auto; will redistribute extra space based on the flex-grow value.



Flex item properties

The flex shorthand property is the recommended way to set flex-grow, flex-shrink and flex-basis values. The shorthand helps the flex properties set intelligently.

Flex item properties

The flex shorthand property is the recommended way to set flex-grow, flex-shrink and flex-basis values. The shorthand helps the flex properties set intelligently.

```
.item {
  flex: none | [ <'flex-grow'> <'flex-shrink'>? || <'flex-basis'> ]
}
```

Flex item properties

The flex shorthand property is the recommended way to set flex-grow, flex-shrink and flex-basis values. The shorthand helps the flex properties set intelligently.

```
.item {
    flex: 1;
}
```

(show example)

Flex item properties

The align-self property will allow a flex item to align itself in the cross-axis separately from other flex items. It takes the same values as the align-items property, but applies to one item.

Flex item properties

The align-self property will allow a flex item to align itself in the cross-axis separately from other flex items. It takes the same values as the align-items property, but applies to one item.

```
.item {
    align-self:  auto | flex-start | flex-end | center | baseline | stretch;
}
```

(show example)

Practical example: centering

Centering in vanilla CSS relies heavily on the margin: 0 auto; property, which sets the left and right margins of an item to automatic centering, and which can be circumvented by floats, clears, etc.

Centerting vertically in vanilla CSS is nearly impossible.

Enter Flexbox.

Practical example: centering

```
.container {
    display: flex;
    height: 300px;
}

.item {
    margin: auto;
}
```

(show example)

Practical example: responsive menu

Many of your class groups wrote navigation menus that were basically vertical lists, which would get hamburgered up when they were eventually set for the mobile devices.

Flexbox makes it easy to transition from a vertical to a horizontal menu layout with just a couple of commands.

(show example)