Webpages

**HTML- tags**

The ***<a>* tag** defines a hyperlink, which is used to link from one page to another. The most important attribute of the <a> element is the href attribute, which indicates the link's destination.

The **<div> tag** defines a division or a **section** in an HTML document. The <div> element is often used as a container for other HTML elements to style them with **CSS** or to perform certain tasks with JavaScript.

The **<span> tag** is used to group inline-elements in a document.

------------------------------------------------------------------------------

**Tips & Tricks & Info**

Margins work like a clock. Top, right, bottom and left.

(Ctrl shift /): block text with Visual Code.

Block and inline block are function for controlling the size of elements.

Ul>li\*3: shortcut meerdere links. Handig gap!

Enlarge pictures and then shrink them back! For some reason… Border-radius 50% to make a picture round, a circle.

------------------------------------------------------------------------------

FlexBox; parent is flex container and children are flex items. Display defines a container, properties either inline or block.

Elements with display:inline-block are like display:inline elements, but they can have a **width** and a **height**. That means that you can use an inline-block element as a block while flowing it within text or other elements.

**Flex-direction** establishes the **main- axis**.

By default, flex items will all try to fit onto one line. You can change that and allow the items to wrap as needed with this property.

.container{

flex-wrap: nowrap | wrap | wrap-reverse;

}

**Flex-flow** is is a shorthand for the flex-direction and flex-wrap properties, which together define the flex container's main and cross axes. The default value is row nowrap.

**Justify-content** defines the alignment along the main axis. It helps distribute extra free space leftover when either all the flex items on a line are inflexible, or are flexible but have reached their maximum size. It also exerts some control over the alignment of items when they overflow the line.

**Align-items** defines the default behavior for how flex items are laid out along the **cross axis** on the current line. Think of it as the justify-content version for the cross axis.

Parameters: flex-start | flex-end | center | space-between | space-around | space-evenly | stretch | start | end | baseline | first baseline | last baseline + ... safe | unsafe;

**Align-content** aligns a flex container's lines within when there is extra space in the cross-axis, similar to how justify-content aligns individual items within the main-axis.

**Note:** this property has no effect when there is only one line of flex items.

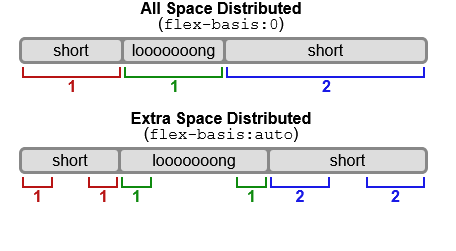
**Properties for the children**

This will be done now. Lets start! You can give the items a number for their **order**. By default, flex-items are laid out in source order. Order: <integer>. Default is 0.

**Flex-grow** defines the ability for items to grow if necessary. Flex-grow: <number>. Default is 0. It dictates what amount of the available space in the container the item should take up. If all items have flex-grow set to 1, the remaining space will be distributed equally to all children. If one of the children has a value set to 2, the remaining space would take up twice as much space as the others.

**Flex-shrink** defines the ability for a flex-item to shrink if necessary. Syntax == flex-grow.

**Flex-basis** defines the default size of an element before the remaining space is distributed. It can be a length (e.g. 20%, 5rem, etc.) or a keyword. Rem is relative to the font sixe of the element. The “auto” keyword means “look at my width or height property”. The “content” keyword means “size it based on the item’s content. Content function not well supported. If set to 0, the extra space around content isn’t factored in. If set to auto, the extra space is distributed based on its flex-grow value. Default is auto.



**Flex** shorthand for flex-grow, flex-shrink and flex-basis combines. Second and third parameters are optional. Default is 0 1 auto, but if you set it with a single number value, it’s like <number> 1 0. **Recommended to use the shorthand flex function, because it sets the other values intelligently.**

**Align-self** allows the default alignment (or the one specified by align-items) to be overridden for individual flex items. Align-self: auto | flex-start | flex-end | center | baseline | stretch. Float, clear and vertical-align have no effect on a flex-item.

----------------------------------------------------------------------------------

## The float Property is used for positioning and formatting content e.g. let an image float left to the text in a container.

The float property can have one of the following values:

* left - The element floats to the left of its container
* right - The element floats to the right of its container
* none - The element does not float (will be displayed just where it occurs in the text). This is default
* inherit - The element inherits the float value of its parent

In its simplest use, the float property can be used to wrap text around images.

## The clear Property specifies what elements can float beside the cleared element and on which side.

The clear property can have one of the following values:

* none - Allows floating elements on both sides. This is default
* left - No floating elements allowed on the left side
* right- No floating elements allowed on the right side
* both - No floating elements allowed on either the left or the right side
* inherit - The element inherits the clear value of its parent

The most common way to use the clear property is after you have used a float property on an element.

When clearing floats, you should match the clear to the float: If an element is floated to the left, then you should clear to the left. Your floated element will continue to float, but the cleared element will appear below it on the web page.

The following example clears the float to the left. Means that no floating elements are allowed on the left side (of the div):

Div {

Clear: left;

}

----------------------------------------------------------------------------