# Display and Position

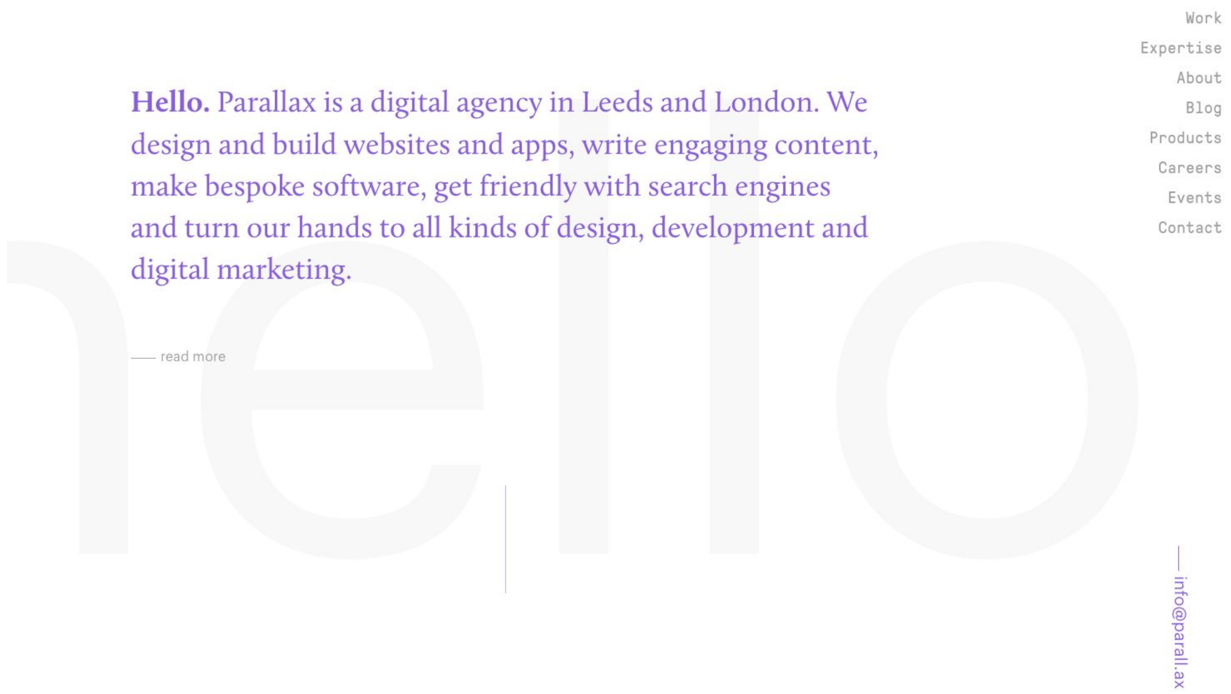
Nic Aguirre   j363 Fall 2018

## Today

|  |  |
|---|---|
| Inspiration | Parall.ax |
| Lecture | Display and Position |
| Practice | Display and Position |

## Inspiration

[https://parall.ax/](https://parall.ax/)



**Hello.** Parallax is a digital agency in Leeds and London. We design and build websites and apps, write engaging content, make bespoke software, get friendly with search engines and turn our hands to all kinds of design, development and digital marketing.

## CSS Display and Position

## Properties of interest

Many CSS properties are used for layouts, but in this lecture we’re focusing on a few:

- `display`
- `position`
- `overflow`
- `float` and `clear`
- `box-sizing`

## display

The `display` property specifies how an element is displayed.

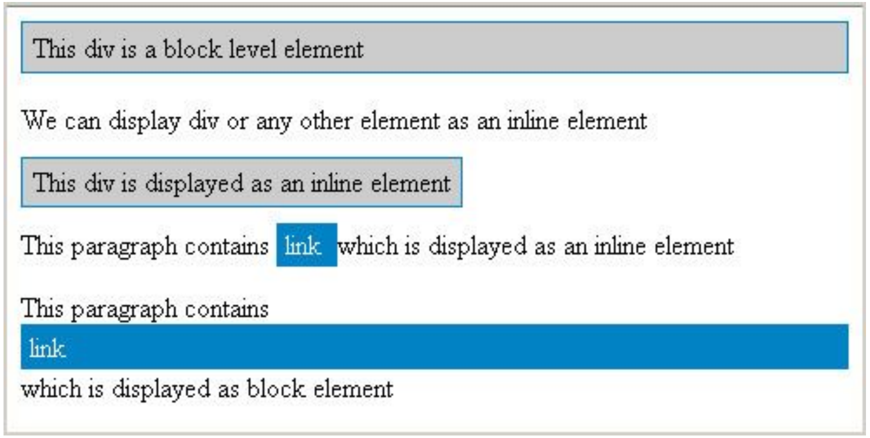
Every HTML element has a default display value depending on what type of element it is. The default display value for most elements is `block` or `inline`.

# display - block level elements

A block-level element always starts on a new line and takes up the full width available (stretches out to the left and right as far as it can).

Examples of block-level elements:

- <div>
- <h1> - <h6>
- <p>
- <header>
- <footer>
- <section>



This div is a block level element

We can display div or any other element as an inline element

This div is displayed as an inline element

This paragraph contains link which is displayed as an inline element

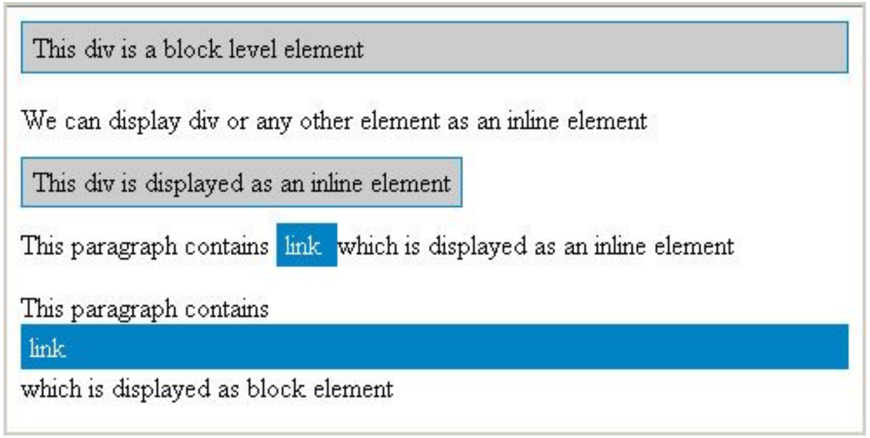
This paragraph contains
link
which is displayed as block element

## display - inline level elements

An inline element does not start on a new line and only takes up as much width as necessary.

Examples of inline elements:

- <span>
- <a>
- <img>



This div is a block level element

We can display div or any other element as an inline element

This div is displayed as an inline element

This paragraph contains link which is displayed as an inline element

This paragraph contains

link

which is displayed as block element

## display - none

`display: none;` is commonly used with JavaScript to hide and show elements without deleting and recreating them.

## display - inline-block

It has been possible for a long time to create a grid of boxes that fills the browser width and wraps nicely (when the browser is resized), by using the `float` property.

However, the `inline-block` value of the `display` property makes this even easier.

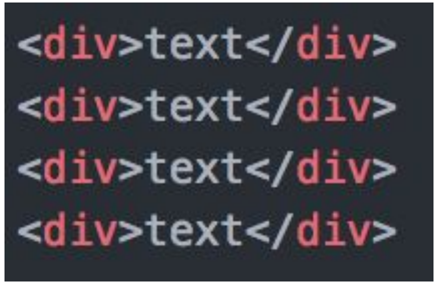
inline-block elements are like inline elements but they can have a width and a height.

## example

## Block-level elements:

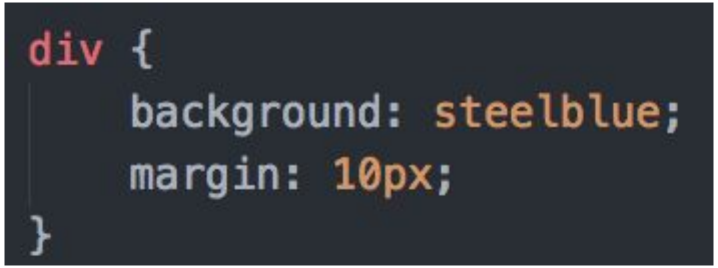
1. Always starts on a new line
2. Take up the full width available (stretches out to the left and right as far as it can)

HTML:

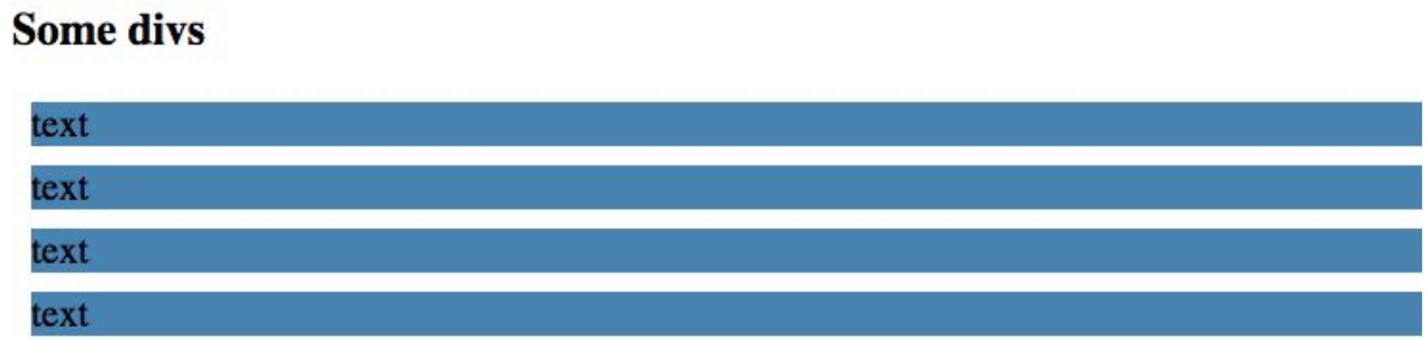


```html
<div>text</div>
<div>text</div>
<div>text</div>
<div>text</div>
```

CSS:



```css
div {
    background: steelblue;
    margin: 10px;
}
```



**Some divs**

text

text

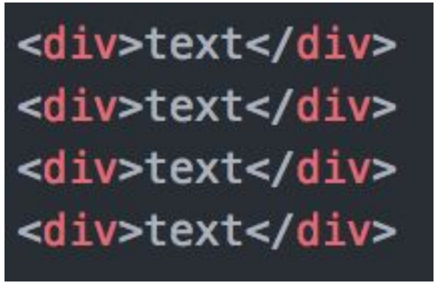
text

text

## Inline elements:

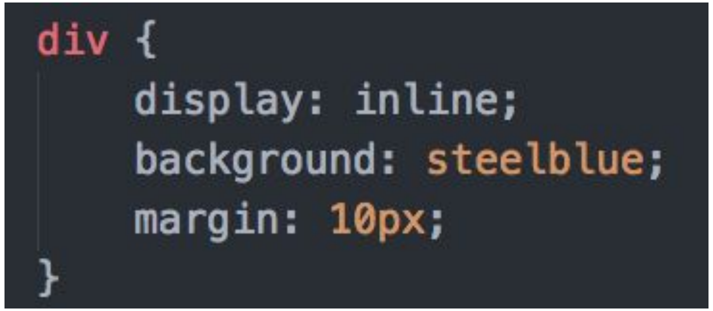
1. Don't start on a new line; they start on the same line as the previous element
2. Only take up as much width as necessary

HTML:

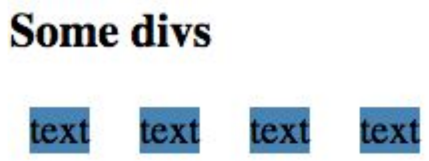


```html
<div>text</div>
<div>text</div>
<div>text</div>
<div>text</div>
```

CSS:



```css
div {
    display: inline;
    background: steelblue;
    margin: 10px;
}
```



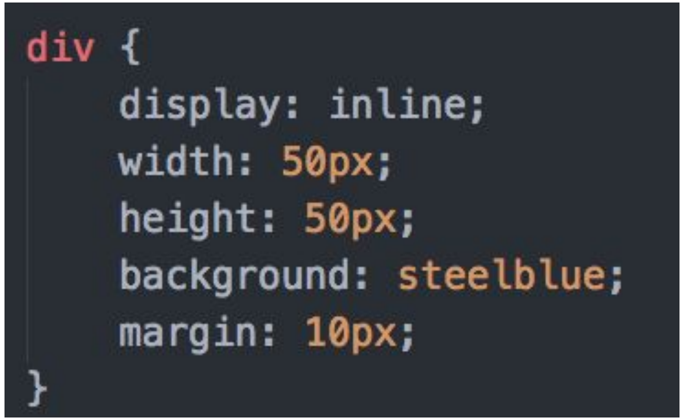
Some divs

text text text text

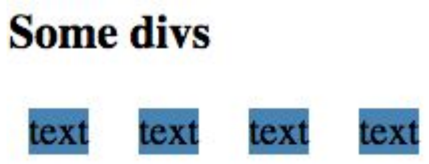
## Inline-block elements:

Compared to `display: inline`, the major difference is that `display: inline-block` allows to set a width and height on the element.

### with inline:



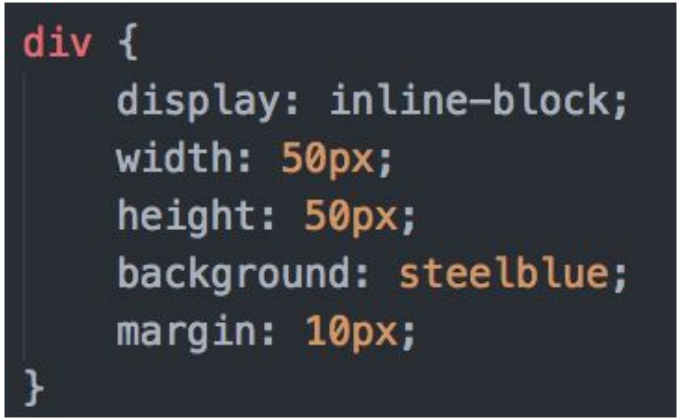
```
div {
    display: inline;
    width: 50px;
    height: 50px;
    background: steelblue;
    margin: 10px;
}
```



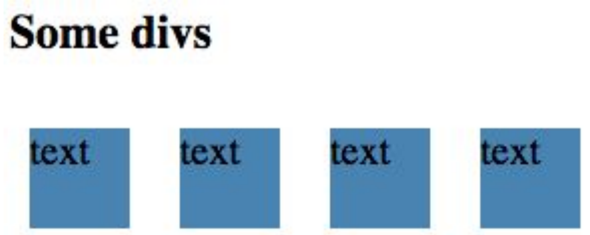
**Some divs**

text text text text

### with inline-block:



```
div {
    display: inline-block;
    width: 50px;
    height: 50px;
    background: steelblue;
    margin: 10px;
}
```



**Some divs**

text text text text

# position

The `position` property specifies the type of positioning method used for an element.

There are five different position values:

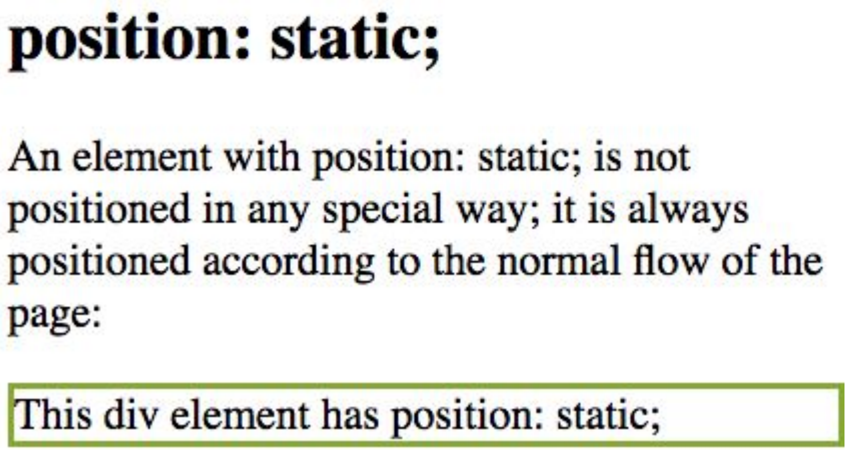
- `static`
- `relative`
- `fixed`
- `absolute`
- `sticky`

In this class, we're mainly interested in `static` and `relative`

Elements are then positioned using the top, bottom, left, and right properties.

However, these properties will not work unless the `position` property is set first.

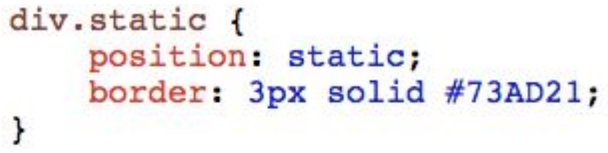
They also work differently depending on the position value.



## position: static;

An element with position: static; is not positioned in any special way; it is always positioned according to the normal flow of the page:

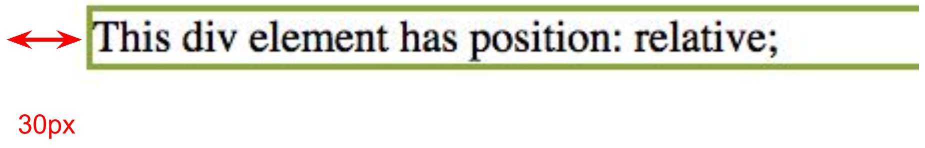
This div element has position: static;



```
div.static {
    position: static;
    border: 3px solid #73AD21;
}
```

## position: relative;

An element with position: relative; is positioned relative to its normal position:


This div element has position: relative;

```
div.relative {
    position: relative;
    left: 30px;
    border: 3px solid #73AD21;
}
```

## overflow

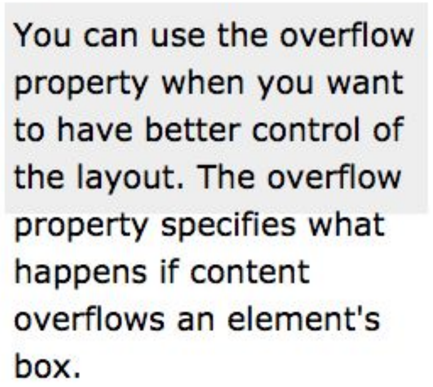
The CSS `overflow` property specifies whether to clip content or to add scrollbars when the content of an element is too big to fit in a specified area.

The `overflow` property has the following values:

- `visible` - Default. The overflow is not clipped. It renders outside the element's box
- `hidden` - The overflow is clipped, and the rest of the content will be invisible
- `scroll` - The overflow is clipped, but a scrollbar is added to see the rest of the content
- `auto` - If overflow is clipped, a scrollbar should be added to see the rest of the content

## overflow: visible

By default, the overflow is `visible`, meaning that it is not clipped and it renders outside the element's box:

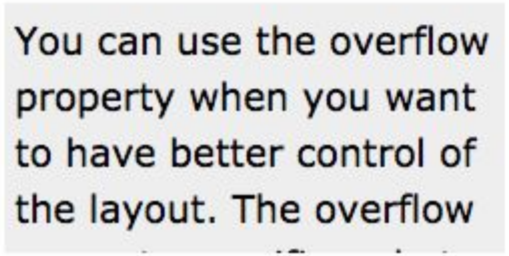


You can use the overflow property when you want to have better control of the layout. The overflow property specifies what happens if content overflows an element's box.

```
div {
    width: 200px;
    height: 50px;
    background-color: #eee;
    overflow: visible;
}
```

## overflow: hidden

With the `hidden` value, the overflow is clipped, and the rest of the content is hidden:

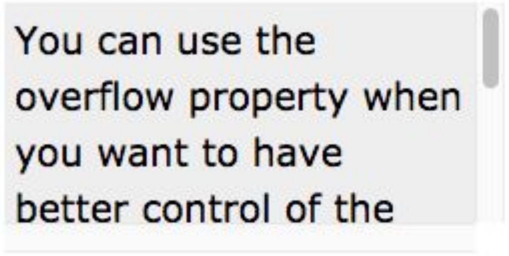


You can use the overflow property when you want to have better control of the layout. The overflow

```
div {
    overflow: hidden;
}
```

## overflow: scroll

Setting the value to `scroll`, the overflow is clipped and a scrollbar is added to scroll inside the box. Note that this will add a scrollbar both horizontally and vertically (even if you do not need it):


You can use the overflow property when you want to have better control of the

```
div {
    overflow: scroll;
}
```

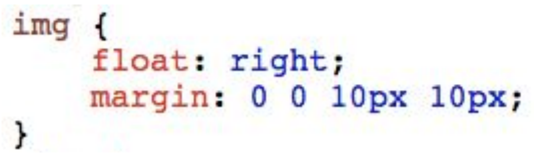
## float and clear

The `float` property specifies whether or not an element should float.

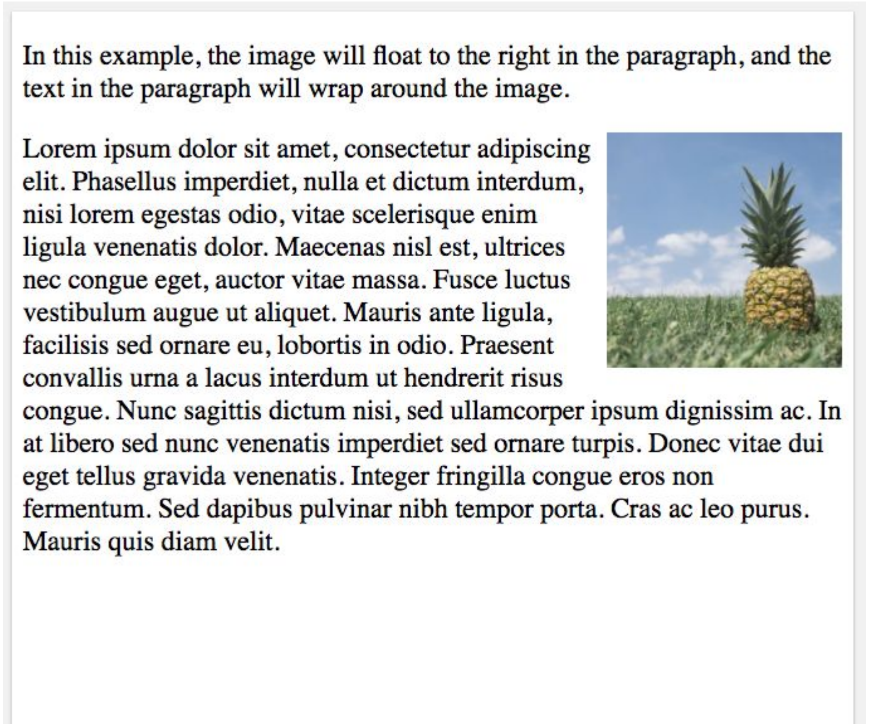
The `clear` property is used to control the behavior of floating elements.

## float and clear

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



```
img {
    float: right;
    margin: 0 0 10px 10px;
}
```



In this example, the image will float to the right in the paragraph, and the text in the paragraph will wrap around the image.

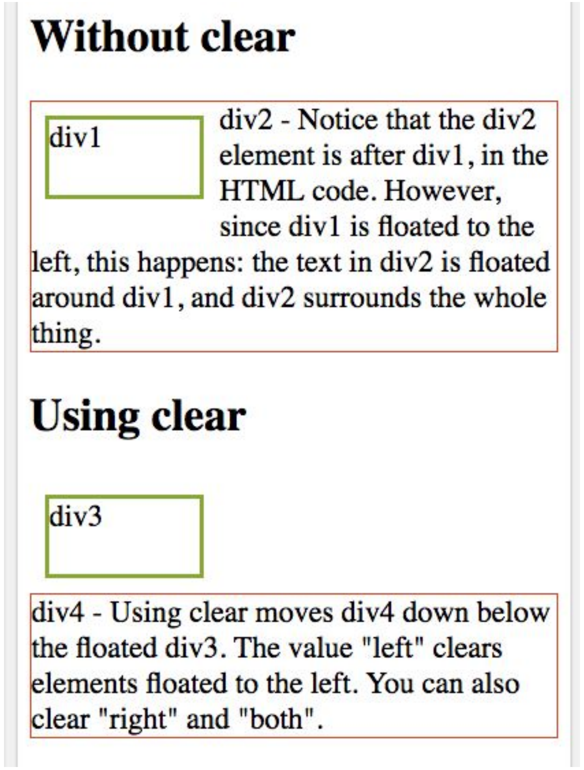
Lorem ipsum dolor sit amet, consectetur adipiscing elit. Phasellus imperdiet, nulla et dictum interdum, nisi lorem egestas odio, vitae scelerisque enim ligula venenatis dolor. Maecenas nisl est, ultrices nec congue eget, auctor vitae massa. Fusce luctus vestibulum augue ut aliquet. Mauris ante ligula, facilisis sed ornare eu, lobortis in odio. Praesent convallis urna a lacus interdum ut hendrerit risus congue. Nunc sagittis dictum nisi, sed ullamcorper ipsum dignissim ac. In at libero sed nunc venenatis imperdiet sed ornare turpis. Donec vitae dui eget tellus gravida venenatis. Integer fringilla congue eros non fermentum. Sed dapibus pulvinar nibh tempor porta. Cras ac leo purus. Mauris quis diam velit.

## float and clear

The `clear` property is used to control the behavior of floating elements.

Elements after a floating element will flow around it. To avoid this, use the `clear` property.

The `clear` property specifies on which sides of an element floating elements are not allowed to float



**Without clear**

div1

div2 - Notice that the div2 element is after div1, in the HTML code. However, since div1 is floated to the left, this happens: the text in div2 is floated around div1, and div2 surrounds the whole thing.

**Using clear**

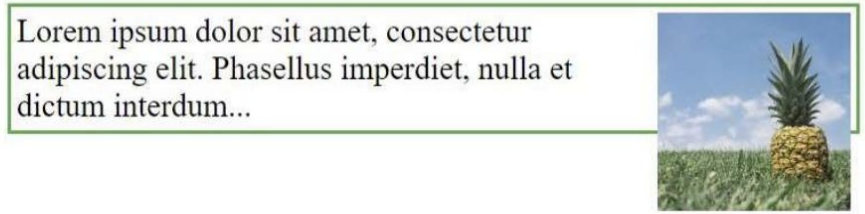
div3

div4 - Using clear moves div4 down below the floated div3. The value "left" clears elements floated to the left. You can also clear "right" and "both".

## the clearfix hack

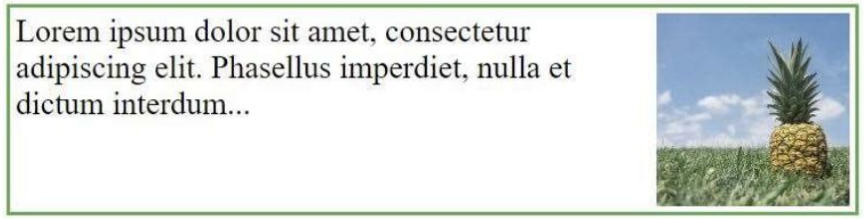
If an element is taller than the element containing it, and it is floated, it will overflow outside of its container.

Without Clearfix



Lorem ipsum dolor sit amet, consectetur adipiscing elit. Phasellus imperdiet, nulla et dictum interdum...

With Clearfix



Lorem ipsum dolor sit amet, consectetur adipiscing elit. Phasellus imperdiet, nulla et dictum interdum...

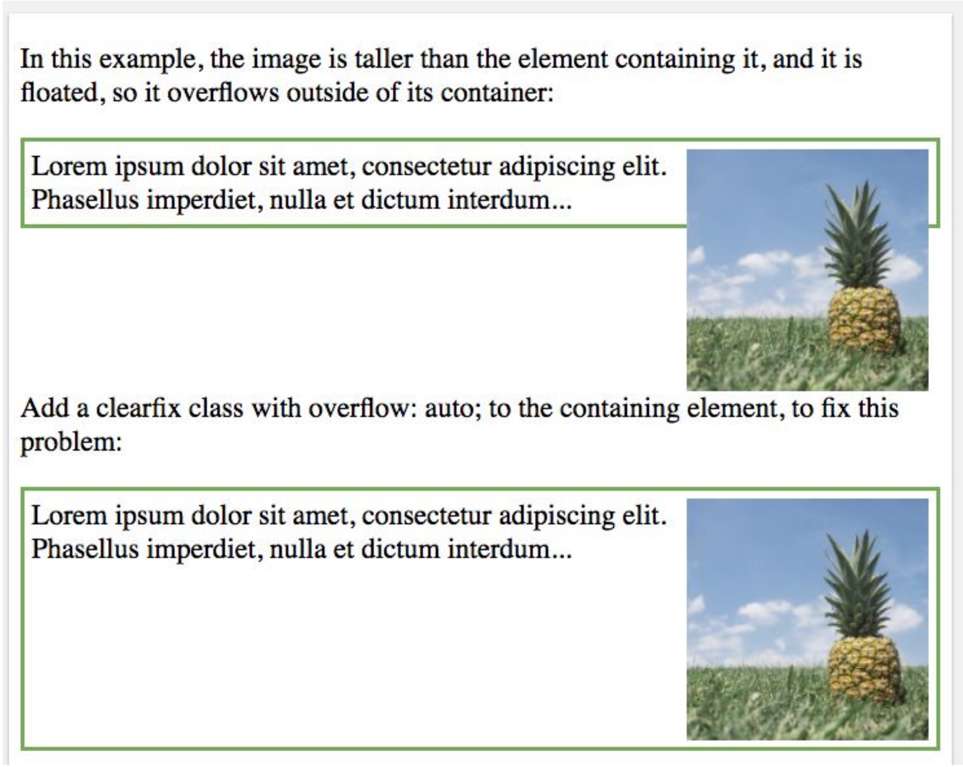
## the clearfix hack

```
div {
     border: 3px solid green;
     padding: 5px;
}

.img1 {
     float: right;
}

.clearfix {
     overflow: auto;
}

.img2 {
     float: right;
}
```



In this example, the image is taller than the element containing it, and it is floated, so it overflows outside of its container:

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Phasellus imperdiet, nulla et dictum interdum...

Add a clearfix class with overflow: auto; to the containing element, to fix this problem:

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Phasellus imperdiet, nulla et dictum interdum...

## box-sizing

There are two ways of sizing elements:

content-box and border-box

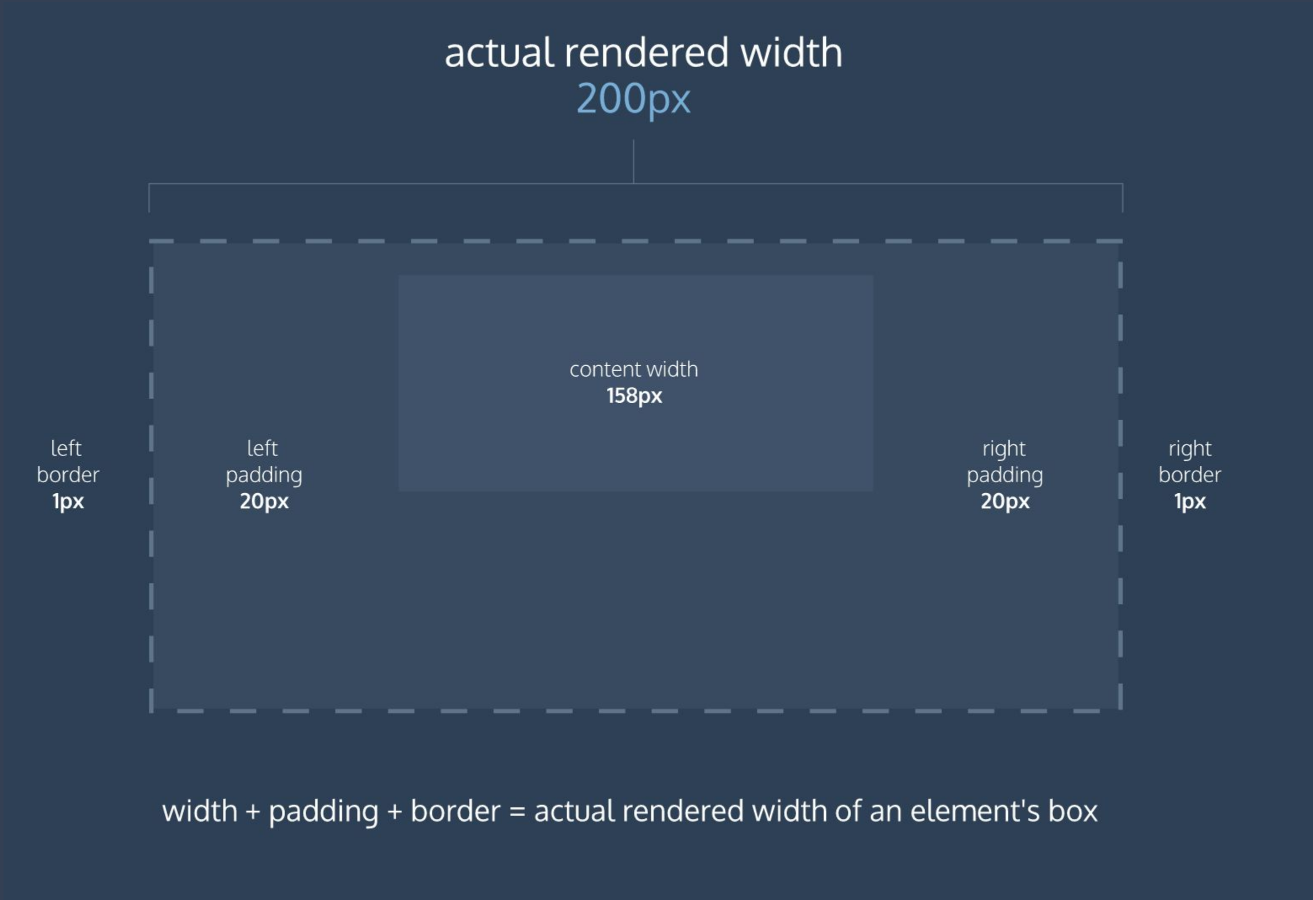
# box-sizing

content-box is the default box sizing model.

With content-box,

Actual rendered width is:
Width + padding + border

## content-box



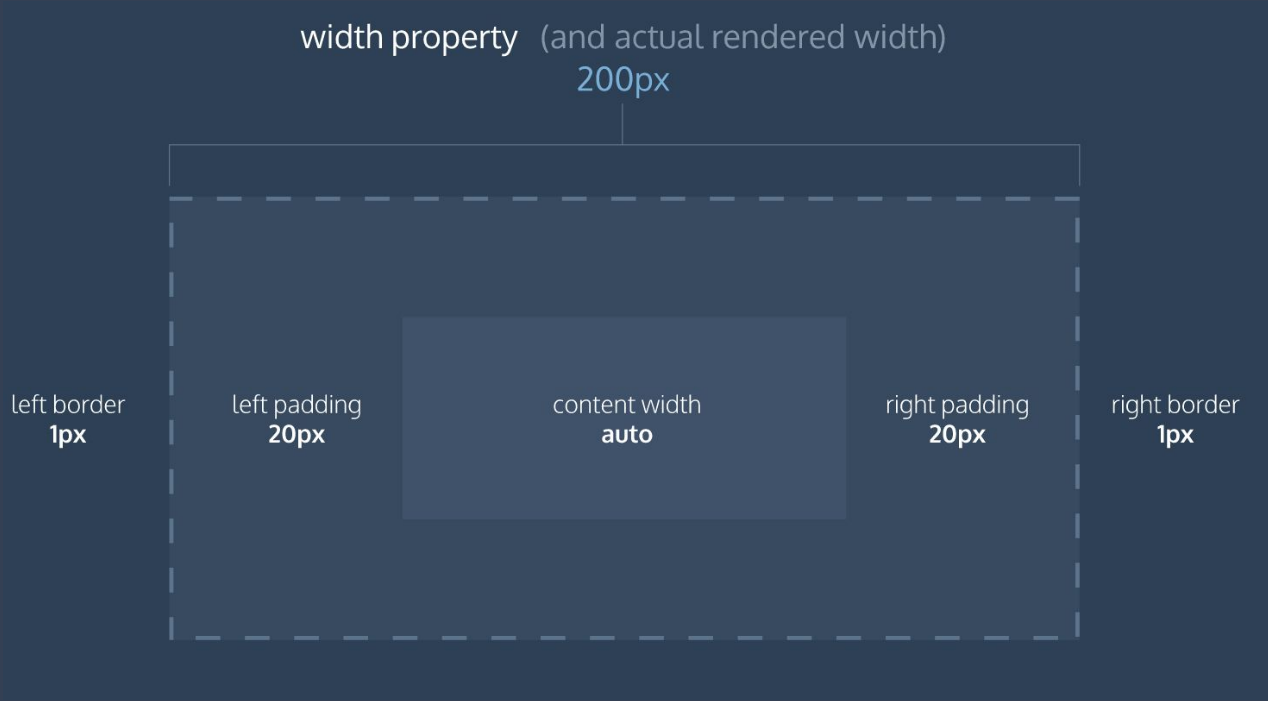
## box-sizing

With border-box,

Actual rendered width is
the same as the width property

Content area is automatically sized based on the remaining width after border and padding have been subtracted

### border-box



## Practice

Download 6.2-display-and-position