CSS II

Programming with Web Technologies

THE UNIVERSITY OF AUCKLAND
NEW ZEALAND
THE UNIVERSITY OF WAIKATO
Te Whare Wānanga o Waikato
Auckland ICT Graduate School

Last time

- Including CSS in your websites
- Colors
- Borders
- Fonts
- CSS selectors

Today

- The DOM
- More CSS selectors
  - Attribute selectors
  - DOM selectors
  - Pseudoclass selectors
- CSS box model, margins & padding
- CSS positioning

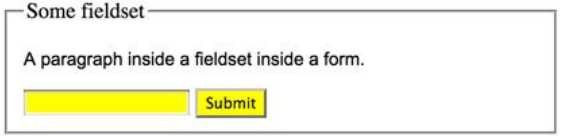
More CSS Selectors

Attribute selectors

We know that HTML elements can have attributes, and since this information is present in our markup it makes sense that we can use these values as part of selectors

An example of where this is needed would be when styling text input boxes. Recall that text boxes are <input> tags, but so are checkboxes, radio buttons etc. This means that the following rule would change all <input> elements

```
input { background-color: yellow; }
```



Attribute selectors

Attribute selectors allow us to add information to type selectors so that only tags that match that type **and** have particular attributes will be affected

Attribute selectors take the form

```
tag[attribute="value"] {}
```

So for the previous example, we could write the following


```
input[type="text"] { background-color: yellow; }
```

Grouping selectors

When programming, we seek to reduce duplication wherever possible - including in CSS. To this end, CSS allows us to specify multiple comma-separated selectors for one block of CSS

```
p, input[type="text"], .abc, #unique {
    border: 1px inset orangered;
}
```

All paragraphs, text inputs, elements with class `abc` and the element with id `unique` will have a 1 pixel wide, inset orangered border

Document Object Model

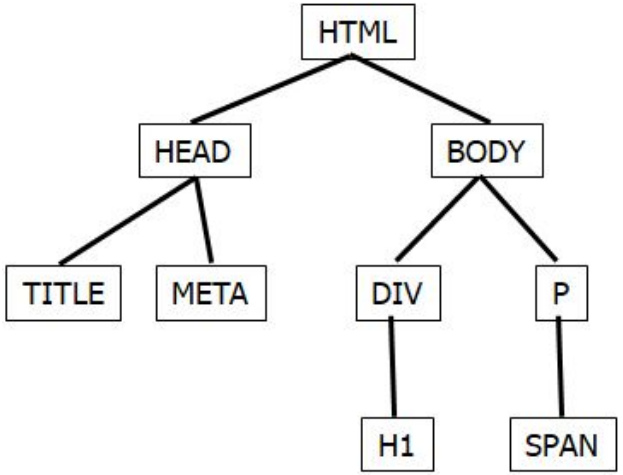
The Document Object Model (DOM) is a tree like representation of a web page. When a web browser loads a web page, it parses the HTML that makes up a page, and constructs a corresponding representation in the form of an internal tree data structure

There is a little more to it than this, but we will discuss it in more detail in the future

This important thing to take away from this, is that this DOM tree can be used by CSS and JavaScript code to identify and manipulate elements

DOM - Example tree

```
<!doctype html>
<html>
  <head>
    <meta charset="UTF-8">
    <title>
      Inline CSS
    </title>
  </head>
  <body>
    <div>
      <h1>heading</h1>
    </div>
    <p>A <span>new</span> paragraph.</p>
  </body>
</html>
```



DIV is a **descendant** of BODY and HTML
DIV is a **child** of BODY
DIV is the **first-child** of BODY
DIV is a **parent** of H1
DIV and P are **siblings**

Child selector

Using what we know about the DOM and the relationships that elements have within the tree, we can come up with selectors that match elements based on their positions in the tree relative to another

The first of these is the child selector, which specifies a parent and child element to match, separated by a right-tag

```
thead > tr { color: orange; }
```

Will match any `tr` elements that are direct children (ie, one level below) of a `thead` element.

Descendant selector

The descendant selector is similar to the child selector, but it matches elements that are anywhere below the matched element in the DOM tree, rather than just first level children. The syntax is similar too, only a space is used rather than a right tag to separate the ascendant and descendant

```
.article p { background-color: red; }
```

Will match any `p` element that is below elements matched by the class name `article` in the DOM tree

Pseudo class selectors

In CSS1 we looked at a number of selectors, but all of them had one thing in common - they all related to information that was given in the HTML structure, such as the element type, id, class and attributes

But what about characteristics that aren't represented directly in the markup?

- A link that hasn't been visited
- A link that has been visited
- The mouse cursor is hovering over an element
- Whether an input has focus

Pseudo class selectors can be used to select elements in these special states

Pseudo class selectors

There are a large number of pseudo classes available, far too many to cover here, but a complete list can be found on MDN

Pseudo class selectors take the form

```
selector:pseudo-class { property: value; }
```

Where `selector` is any valid CSS selector, and `pseudo-class` is a pseudo class appropriate for the element selected

Pseudo class selectors

```
a:visited { }
```
Matches an anchor element that has been visited

```
div.important:hover { }
```
Matches a `div` element with the class important, that is being hovered over

```
input[type="text"]:focus { }
```
Matches a text `input` field that the client has clicked into

```
p:nth-child(5) { }
```
Matches all `p` elements that are the 5th child of their parent

CSS Box Model

CSS box model

In HTML, every element is created inside of a box. That box may be contained inside another box, or may contain boxes within itself. The browser places these boxes appropriately on the page

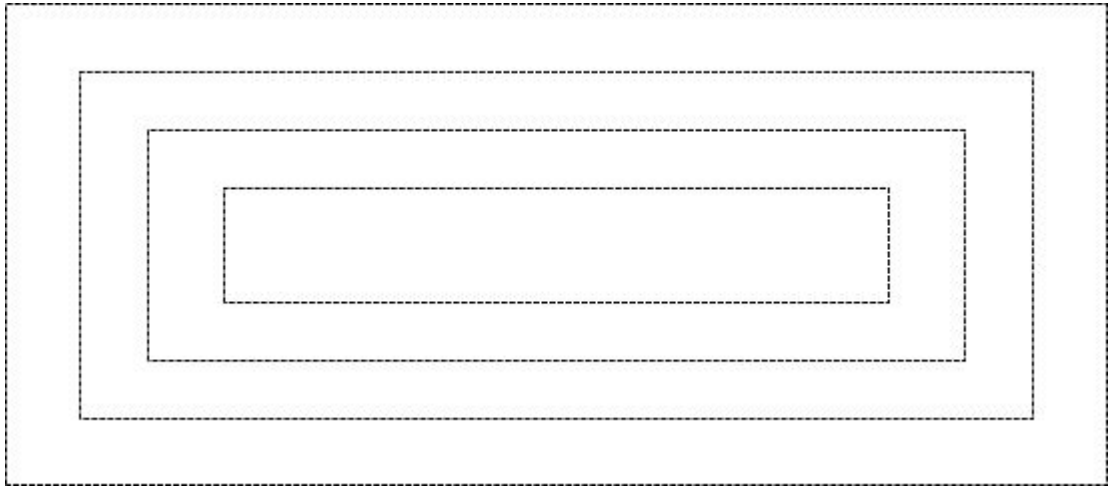
This can be tricky to visualize, so you can force a web page to reveal its boxes by inserting a new CSS rule into a page using the inspection tool

```
* { border: 1px solid black; }
```

This will temporarily force the page to show borders around all elements

CSS box model

The box for each individual element has this structure

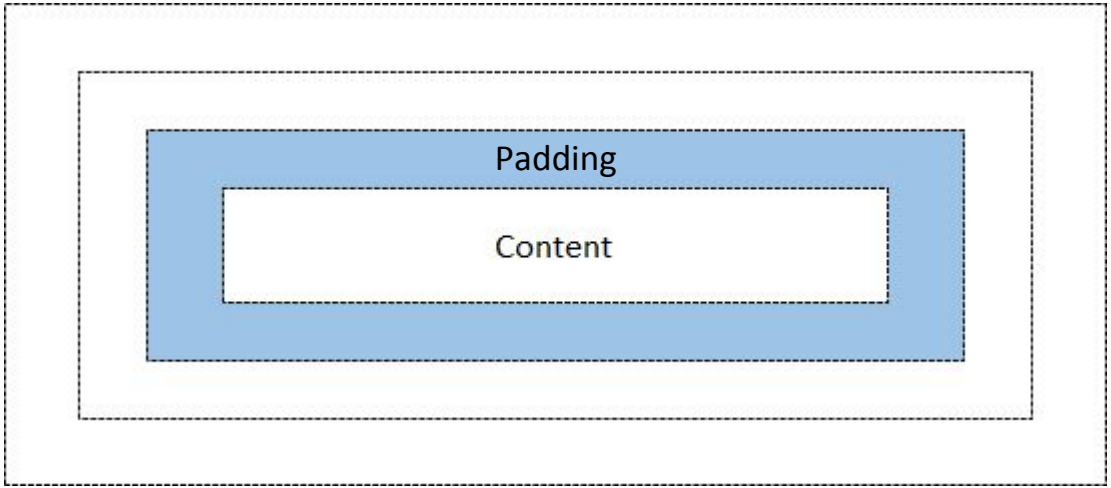


CSS box model

In the center is content. This will be the text, image, input control, etc that makes up this element

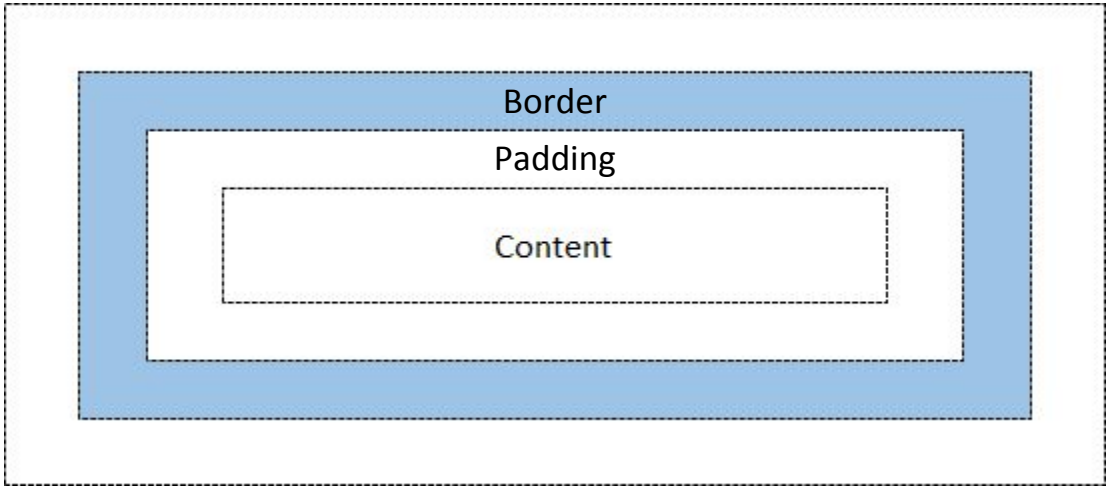
CSS box model

Around the content is the padding. This is space inside the element, between the content and the border. This will be the same color as the `background-color` of the element



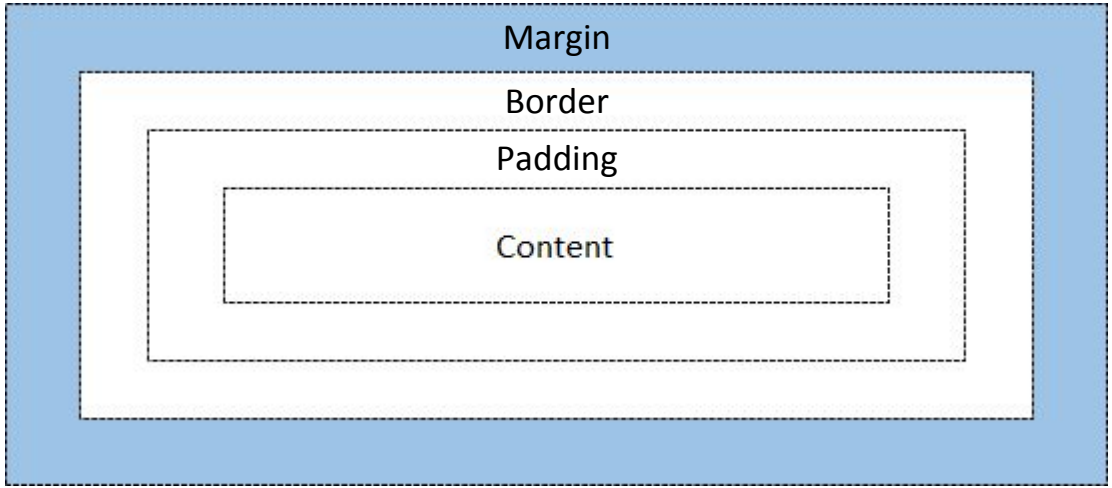
CSS box model

Next is the border. This surrounds the content and the padding. The color of the border is defined by the `border-color` property



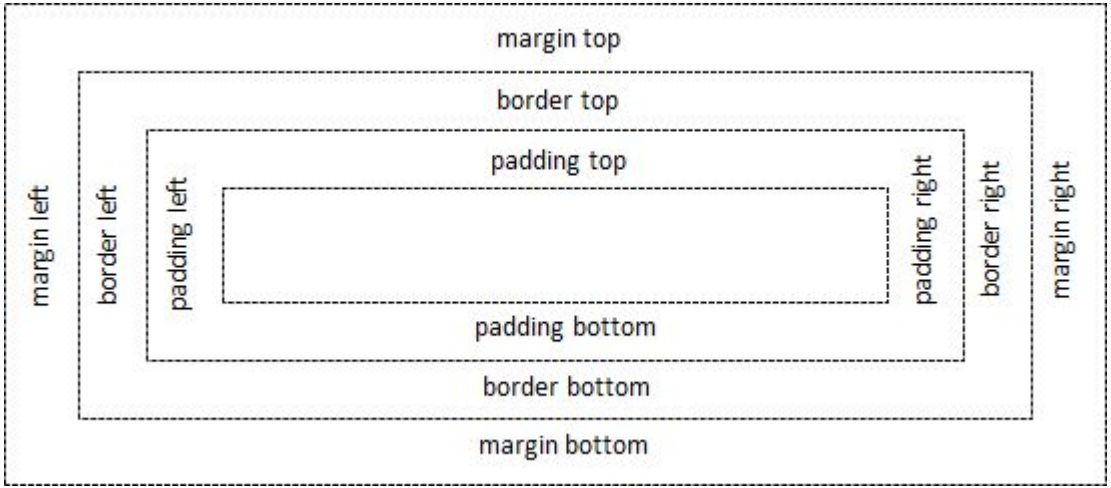
CSS box model

Finally, the margin is outside the border. The margin is always transparent



CSS box model

Just like with the border properties, the width of the margins and padding can be set either for all sides at once, or once side at a time.



Visualizing the box model

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Ut tristique, ex nec interdum sollicitudin, nunc eros lobortis leo, eu scelerisque diam metus eget ante. Quisque at nulla tempus, eleifend tortor ac, laoreet purus. Nam venenatis feugiat nisi, ut euismod quam lobortis ut. Proin quis maximus odio. Duis condimentum ultricies tristique. Sed tempus arcu eget sem porta, venenatis imperdiet ex pellentesque. Pellentesque suscipit rhoncus ante, sed dignissim urna dapibus sit amet. Interdum et malesuada fames ac ante ipsum primis in faucibus. Phasellus scelerisque dui at nulla

```
#img1 {
  background-color: green;
  padding         : 20px;
  border-color    : black;
  border-width    : 20px;
  border-style    : solid;
  margin          : 20px;
}
```

auctor, ac tempus sapien dignissim. Curabitur tincidunt metus in augue accumsan ullamcorper. Class aptent taciti sociosqu ad litora torquent per conubia nostra, per inceptos himenaeos. Curabitur vehicula tincidunt euismod. Cras feugiat mi at nisi semper, eu mollis quam porttitor. Aliquam erat volutpat. Suspendisse vitae leo in justo fermentum egestas. Sed ut urna maximus, maximus urna at, ultrices lectus. Maecenas ullamcorper, nunc sit amet pharetra aliquet, augue nunc hendrerit diam, vel tempor nisl turpis dapibus erat. Nam bibendum, nulla eleifend ultricies volutpat, mauris augue aliquet nulla, sit amet bibendum sem tellus sed erat. Donec euismod nisi vitae sem euismod, at tempor ipsum tincidunt. Sed scelerisque, elit ut ornare venenatis, nisi metus egestas arcu, a mattis risus erat ut augue. Nunc placerat convallis mauris eu convallis. Sed aliquam sapien quis lectus facilisis, eu porttitor est semper.

Visualizing the box model

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Ut tristique, ex nec interdum sollicitudin, nunc eros lobortis leo, eu scelerisque diam metus eget ante. Quisque at nulla tempus, eleifend tortor ac, laoreet purus. Nam venenatis feugiat nisi, ut euismod quam lobortis ut. Proin quis maximus odio. Duis condimentum ultricies tristique. Sed tempus arcu eget sem porta, venenatis imperdiet ex pellentesque. Pellentesque suscipit rhoncus ante, sed dignissim urna dapibus sit amet. Interdum et malesuada fames ac ante ipsum primis in faucibus. Phasellus scelerisque dui at nulla

```
#img1 {
  background-color: green;
  padding      : 20px 40px 20px 70px;
  border-color: black;
  border-width: 20px 50px 10px 0px;
  border-style: solid;
  margin       : 10px 80px 60px 30px;
}
```

auctor, ac tempus sapien dignissim. Curabitur tincidunt metus in augue accumsan ullamcorper. Class aptent taciti sociosqu ad litora torquent per conubia nostra, per inceptos himenaeos. Curabitur vehicula tincidunt euismod. Cras feugiat mi at nisi semper, eu mollis quam porttitor. Aliquam erat volutpat. Suspendisse vitae leo in justo fermentum egestas. Sed ut urna maximus, maximus urna at, ultrices lectus. Maecenas ullamcorper, nunc sit amet pharetra aliquet, augue nunc hendrerit diam, vel tempor nisl turpis dapibus erat. Nam bibendum, nulla eleifend ultricies volutpat, mauris augue aliquet nulla, sit amet bibendum sem tellus sed erat. Donec euismod nisi vitae sem euismod, at tempor ipsum tincidunt. Sed scelerisque, elit ut ornare venenatis, nisi metus egestas arcu, a mattis risus erat ut augue. Nunc placerat convallis mauris eu convallis. Sed aliquam sapien quis lectus facilisis, eu porttitor est semper.

CSS Positioning

CSS positioning

Up until now, all of our CSS has been positioned in **normal flow**

What this means for **block** elements is that they have been laid out vertically as boxes placed one after the other, top-to-bottom. Distance between other box is based on the margin properties of each, and the left edge of each box touches the left edge of its parent

For our **inline** elements, they have been laid out horizontally, left-to-right, separated by their margins, padding and borders. Vertical alignment can be by the element tops, bottoms or text baseline

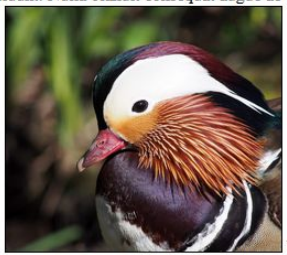
CSS positioning - Normal flow

```
<h1>Heading 1</h1>
  <p>
    ...
  </p>
  <p>
    ...
    <img src="duck.png">
    ...
  </p>
<h2>Heading 2</h2>
```

Heading 1

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Praesent fringilla urna vel posuere pellentesque. Proin eget dictum arcu. Nulla molestie ut ante at porttitor. Aliquam ut varius nulla, nec luctus velit. Sed sed lorem risus. Morbi a dui imperdiet, molestie urna sit amet, tempor mi. Aenean hendrerit fringilla porta. Donec ultrices nisi ligula, vel semper tellus tristique ut. Morbi aliquam malesuada ex. Quisque rhoncus posuere leo, ut aliquet orci faucibus eget. Sed facilisis tellus auctor, laoreet ligula non, feugiat elit. Integer consequat, arcu id commodo tempus, augue libero eleifend eros, a pretium sapien quam in sem. Sed in dolor elit. Morbi auctor tincidunt nulla sit amet rutrum. Donec ac orci sapien. Ut molestie nisl nec odio consectetur, eget luctus sapien mattis.

Sed sollicitudin pulvinar nulla sed tincidunt. Nulla blandit consequat augue ac convallis. Quisque hendrerit purus id



nisi dignissim, a suscipit nunc dictum. Vivamus dui nulla, tristique non nisl at, suscipit mattis eros. Sed non nibh ut enim facilisis eleifend ac at leo. Praesent molestie molestie dapibus. Aliquam consectetur imperdiet lacus et sodales. Phasellus tempor mi nulla, quis iaculis orci tincidunt et. Integer vestibulum eros at ex molestie consequat.

Heading 2

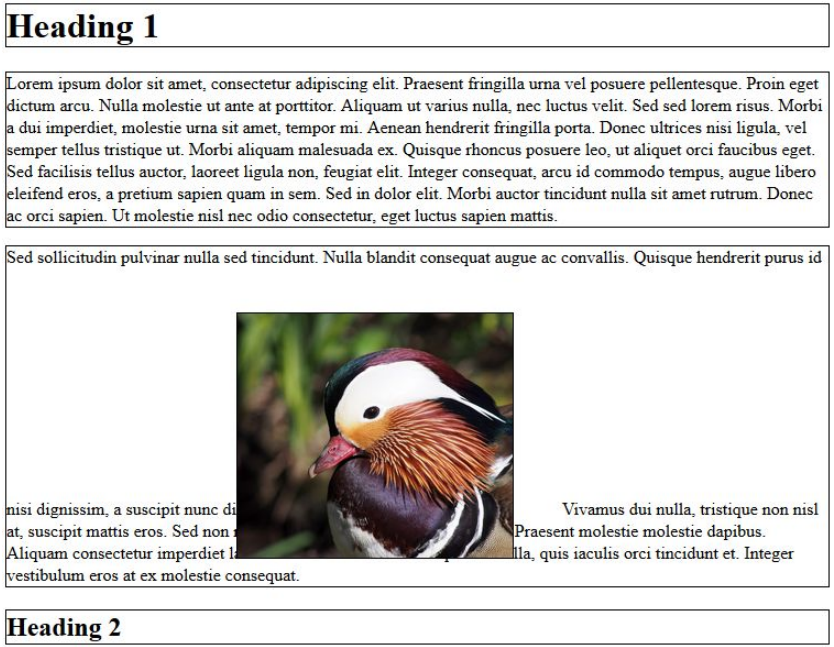
CSS positioning

The way an element is positioned is controlled by the `position` property, and this can be set to one of 5 values

`static`	(default) Positioned using normal flow
`relative`	Positioned relative to where it would be using normal flow by providing an offset
`absolute`	Removed from normal flow and positioned by coordinates relative to the containing block Note: you may need to add position relative to the containing/parent element for this to work
`fixed`	A constant position relative (normally) to the window i.e. not affected by scrolling
`sticky`	Toggles between relative and fixed depending on scroll position

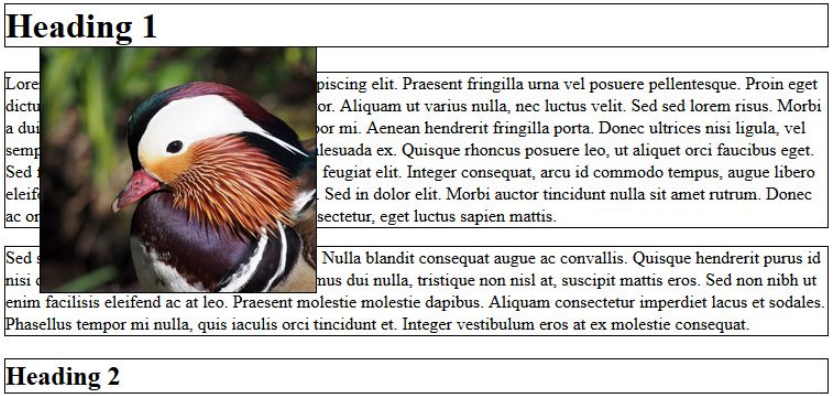
CSS positioning - Relative

```
img {
  position: relative;
  left: -40px;
  top: 40px;
  /* Could use right or bottom */
}
```



CSS positioning - Absolute

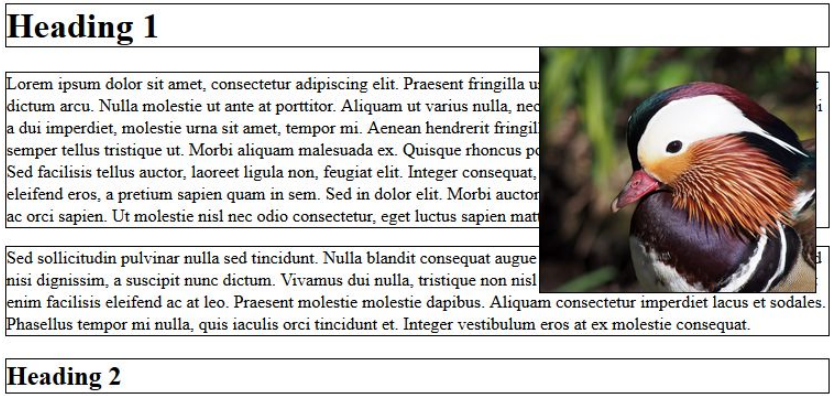
```
img {
  position: absolute;
  left: 40px;
  top: 60px;
  /* Could use right or bottom */
}
```



Absolute positioning positions within the first containing element that is not in **normal flow**. In this case, the paragraph where the image is placed is in the normal flow. The closest container that would make sense would be the body element

CSS positioning - Fixed

```css
img {
  position: fixed;
  right: 20px;
  top: 60px;
}
```

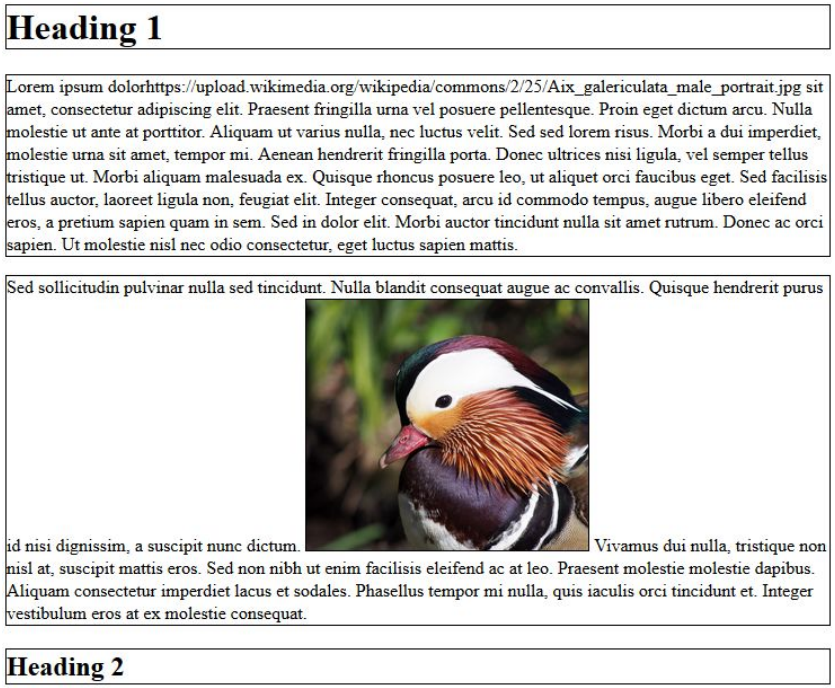


Fixed positioning is used when you want something to always stay at the same position on the screen. A fixed position element will not scroll with the page

CSS positioning - Sticky

```
img {
  position: -webkit-sticky;
  position: sticky;
  top: 0px;
}
```

Sticky is difficult to visualize with a static image, but when the image reaches the top of the page due to scrolling, it will 'stick' to it, acting like a fixed positioned element



CSS positioning - Float

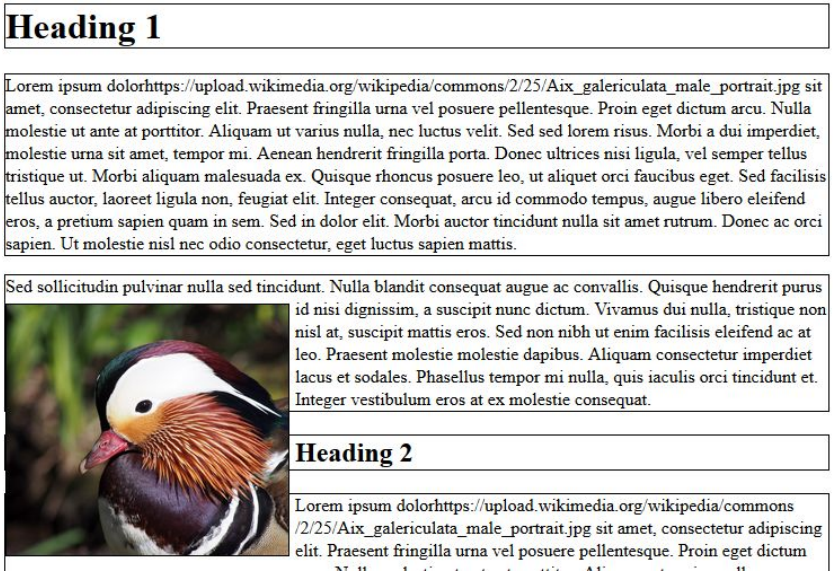
Sometimes, we want to be able to remove elements from the normal flow. This is known as **floating** an element. When an element is floated, it moves to the left or right side of its container, and all other content will flow around it. The `float` property controls the floating state of an element

After an element has been floated, all other content in the normal flow will flow around the element. If this is not what is wanted, you can correct this by clearing the float with the `clear` property

CSS positioning - Float

```
img {
  float: left;
  margin: 5px 5px 5px 0px;
}
```

Even though the image has been floated, we can still use box-model properties such as margin to affect how they interact with other elements. In this case a margin has been added to keep the wrapped text a short distance from the image



CSS units

Absolute units are useful assuming you know the characteristics of the device you are displaying on

in	Inches
cm	Centimetres
mm	Millimetres
pt	Points (1/72nd of an inch)
pc	Picas (1/6th of an inch, 12 points)

CSS units

Relative units are useful if you want your elements to be sized relative to some other length property in the document. Relative units scale better than absolute units when displaying across different devices

px	Pixels. Depends upon the resolution of the viewing device
em	Relative to the font-size of the element (2em means 2 times the size of the current font)
ex	x-height, relative to the size of a lower-case "x" in the font being used

CSS units

These relative units units are proportional to the size of other elements - either the whole page, or a parent element.

%	Percent. Proportional to the size of the parent (containing) element.
vw, vh	1% of the viewport's (browser window's) width or height. E.g. 23vh = 23% of the browser window's height.
vmin, vmax	1% of the viewport's smaller or larger dimension. E.g. 23vmax is 23% of the browser window's width or height - whichever is larger.

References

- MDN CSS Selectors
- MDN Box Model
- W3Schools CSS Positioning
- MDN Units