EXTRA CSS3.

#4

There are 3 places you can write CSS.

- The best option is to write CSS is in a **separate .css file**. You then link that file to your HTML file in the head of your document:
  <link rel="stylesheet" href="css/screen.css">

- You can also write CSS in the **<head> of each HTML** page.

```
<style>
  p{
    color:red;
  }
</style>
```

- Or the last resort is to write CSS **inline within a HTML element**. To do that you add an attribute to the opening tag:
  This will be red

- You can use all **3 variations simultaneously** if you wish. You can also load **mulitple external CSS files**.

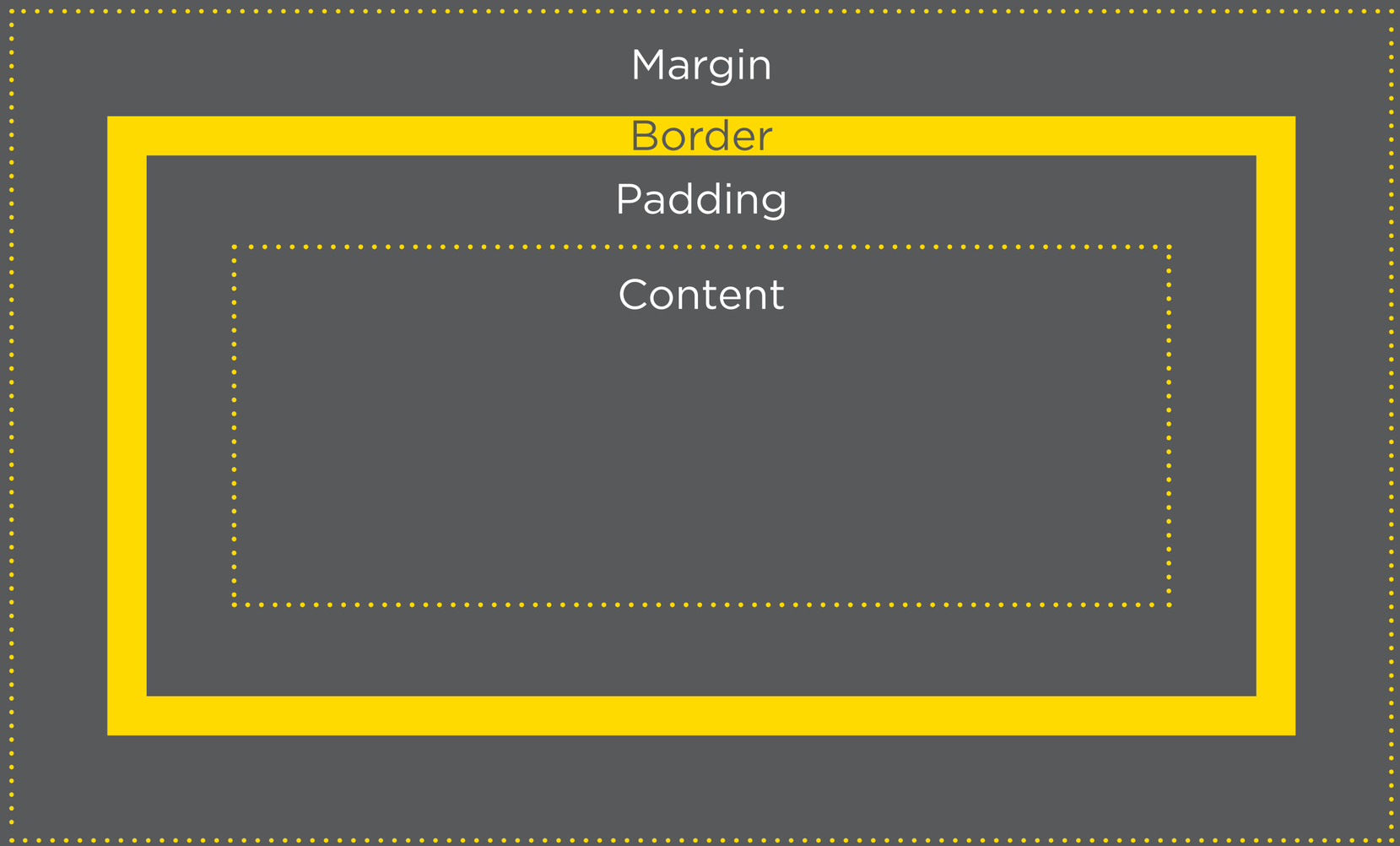
There are 3 parts to a rule, the **selector**, the **property** and the **value**.

selector

```
p {
        property    value
color:red;
}
```

- The punctuation is very important.
  - Each selector is followed by a set of curly brackets.
  - Inside this a property is followed by a colon
  - and the value is followed by a semi-colon.

A block HTML element can consist of margins, borders, padding and the content.

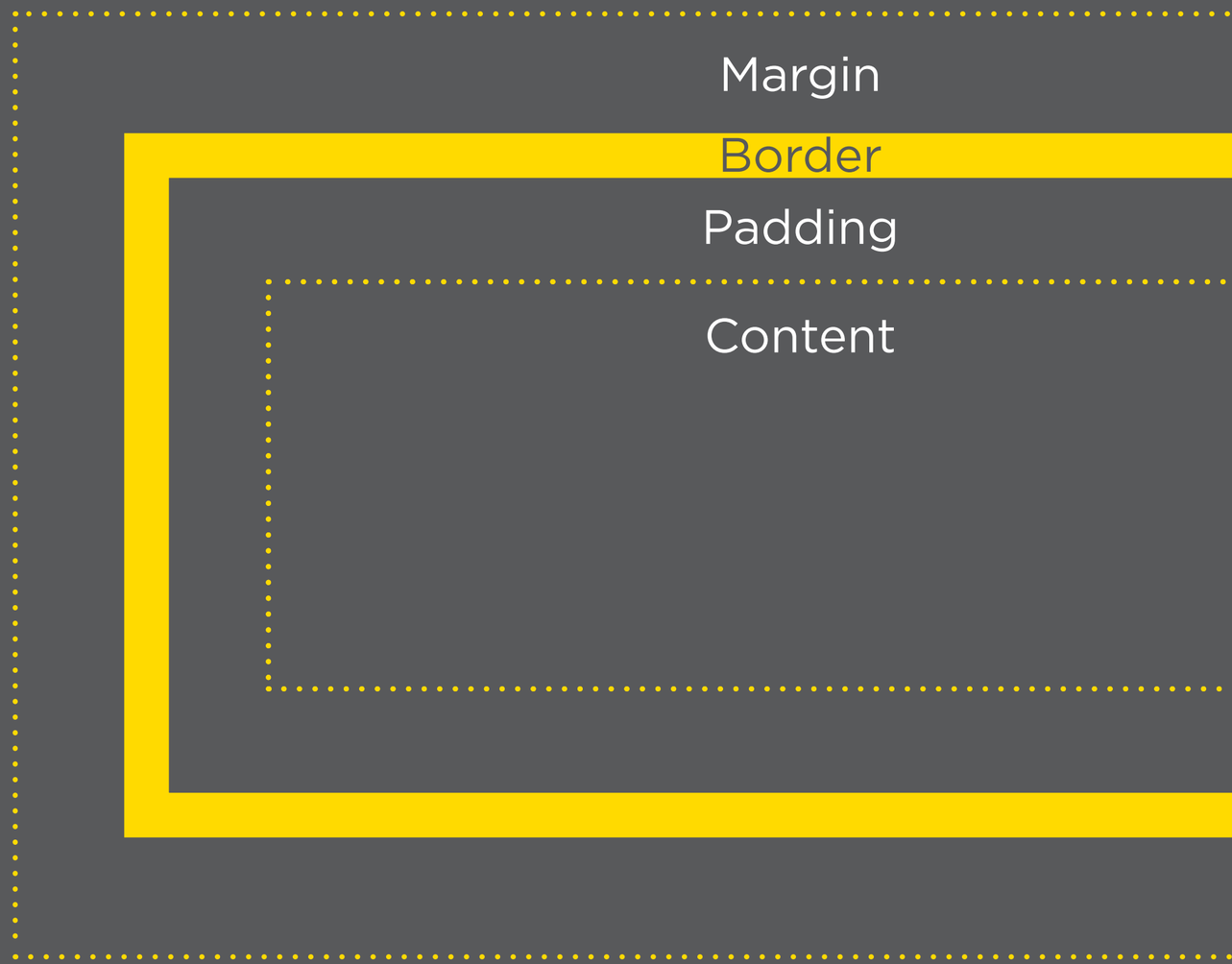


Content
The content of the HTML element, where text and images appear.

Padding
Clears an area around the content but inside the border. It will have the background style of the element.

Border
The border goes outside the content and padding.

Margin
Clears an area outside the border. It is transparent.



Margin makes an empty area or spacing around an HTML element. The margin area is transparent. You can use px or % as the unit.

```
h1{
  margin:20px;
}
```

There are 2 shorthand versions of margin. The first allows you to write the 4 values in one line. The order is top, right bottom, left.

```
h1{
  margin:20px 15px 10px 5px;
}
```

or

```
margin:20px 15px;
```

You can use the value 'auto' for the left and right margins that allows you to center an element horizontally. You also need to set a width.

```
h1{
  margin:20px auto;
  width:400px;
}
```

You can also define one margin side at a time.

```
h1{
  margin-left:20px;
}
```

Padding makes an empty area or spacing around an HTML content but inside the border. The padding area will use the background style of the HTML element. You can use px or % as the unit.

```
h1{
  padding:20px;
}
```

Like margin, there are the same 2 shorthand versions of padding. The first allows you to write the 4 values in one line. Again the order is top, right bottom, left.

```
h1{
  padding:20px 15px 10px 5px;
}
```

or

```
padding:20px 15px;
```

The default width of a block element is 100%. You can set a smaller width in CSS.

```
h1{
  width:250px;
}
```

You can also set a max-width. This is useful for responsive websites and different screen sizes.

The max-width rule below will allow the element to reach a maximum width of 450px. At smaller screen sizes, it will be 100% wide.

```
p {
  max-width:450px;
}
```

You can also set a height for an element. Generally setting a height is not advisable as you can have problems if the content doesn't fit in the space defined.

```
p {
  height:450px;
}
```

A compromise is to use min-height. The rule below will set the element to be at least 450px high but allow it to expand if the content inside is taller.

```
p {
  min-height:450px;
}
```

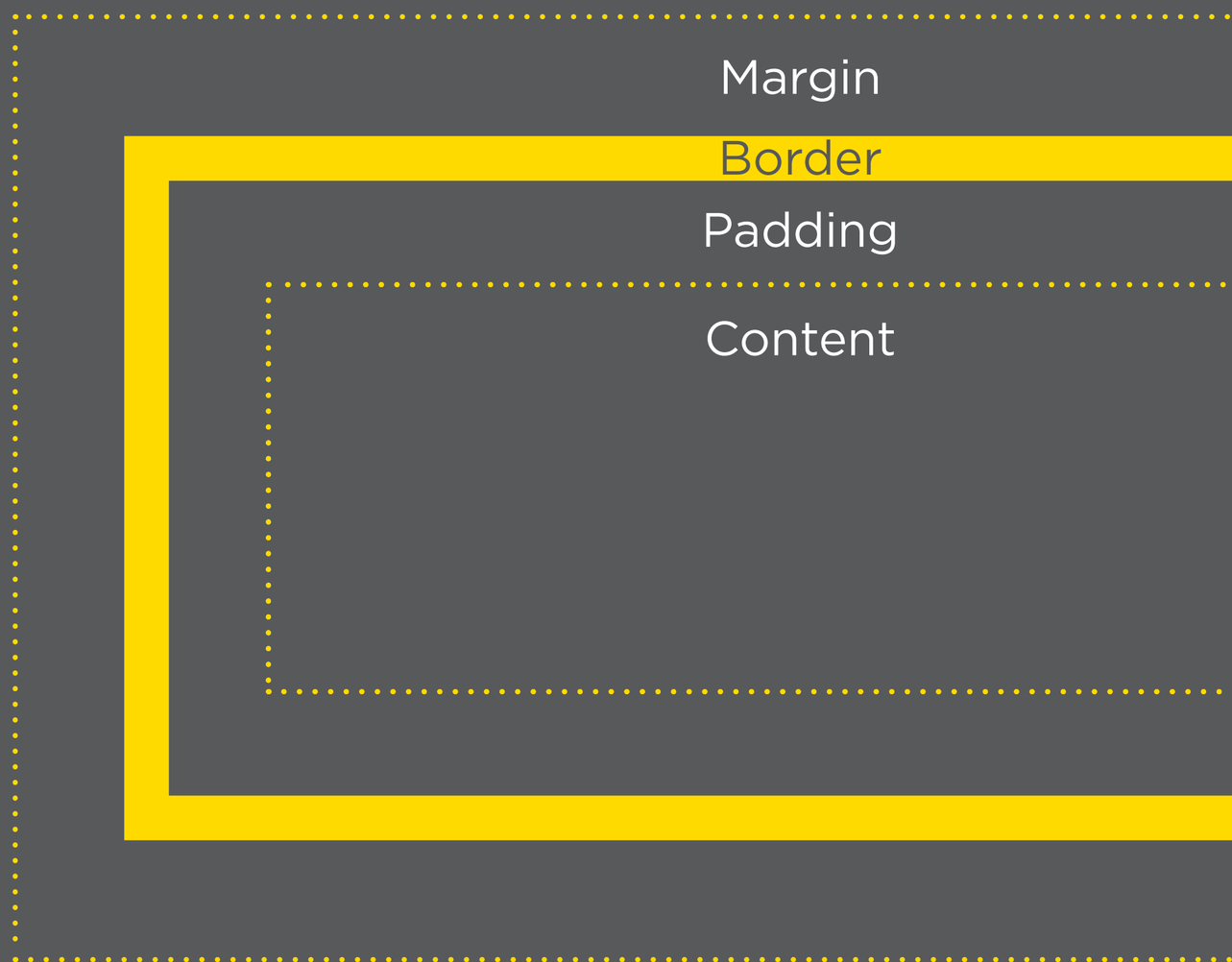
Working out Size

There are 2 ways to calculate the size of an element. You can change this by using the CSS rule

box-sizing:border-box;

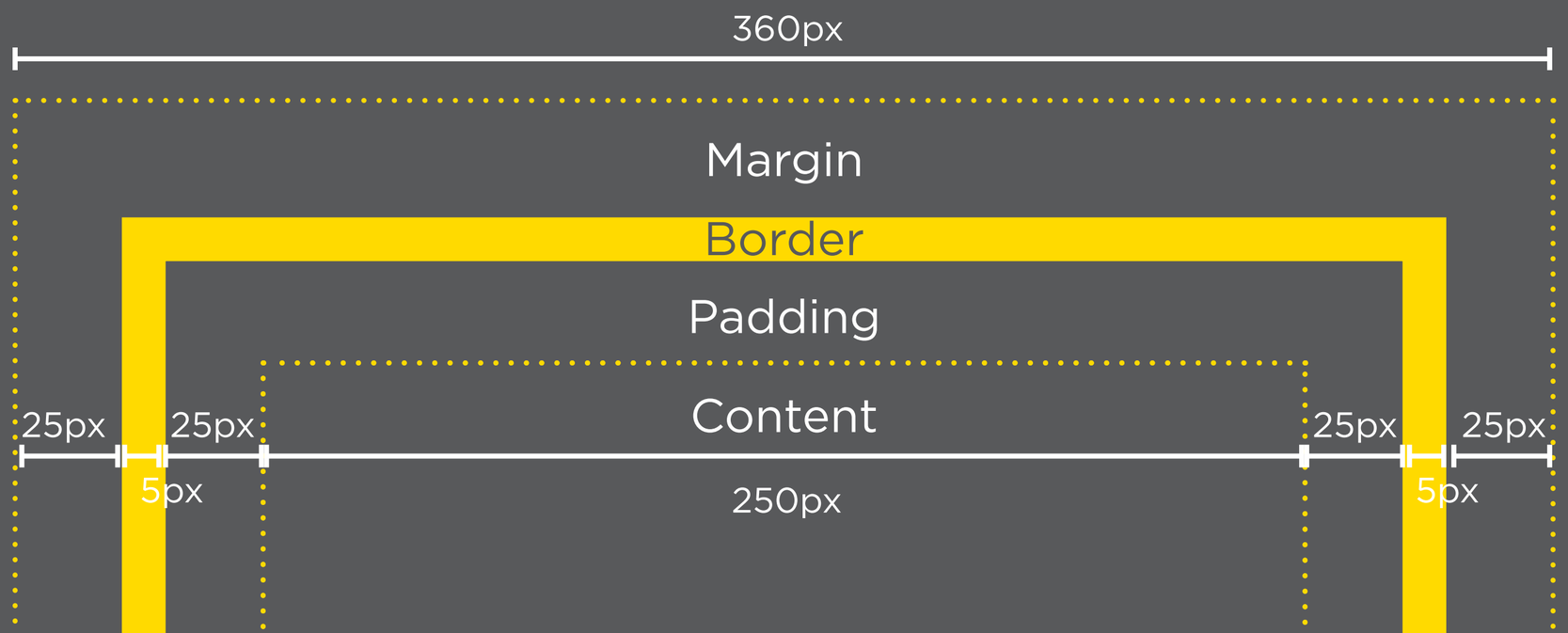
or the default is

box-sizing:content-box;



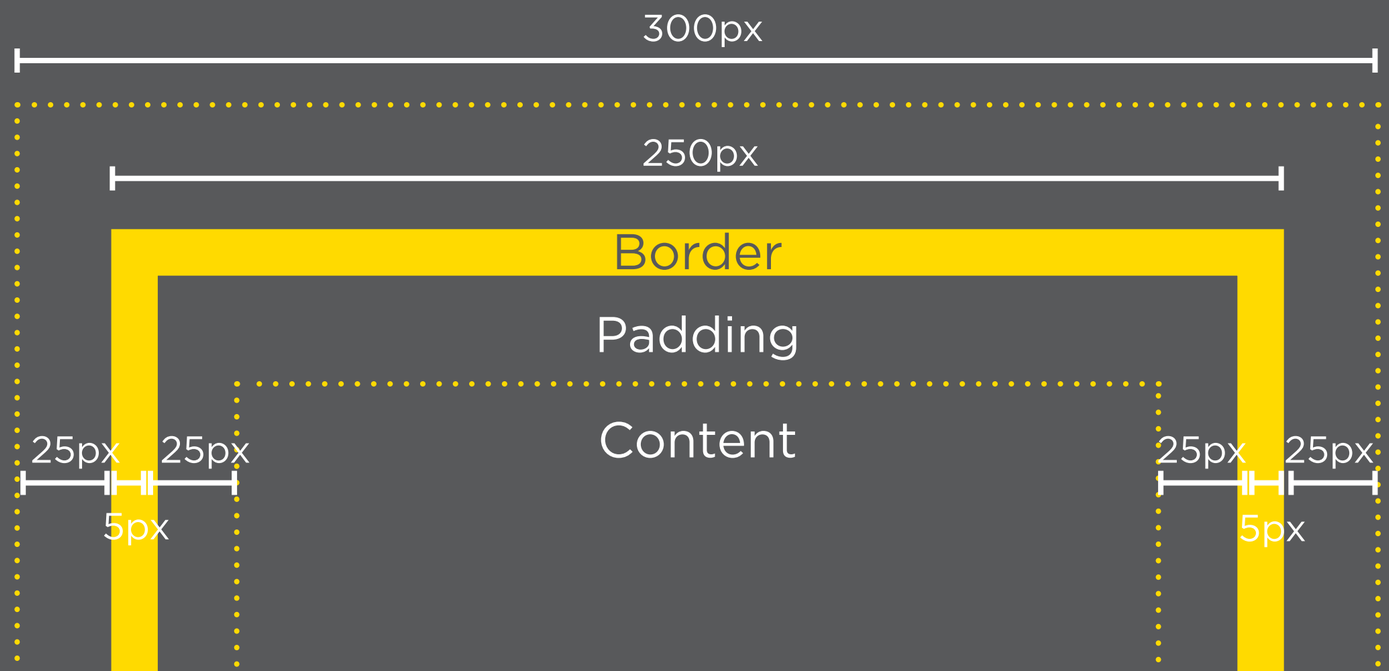
With content-box, you set the width and height of the content area. To calculate the full size that element takes up, you must also add padding, borders and margins.

```
.box{
    width:250px;
    padding:25px;
    margin:25px;
    border:5px solid #FFDA00;
}
```



With border-box, the width and height you set include any padding,borders and the content area. To calculate the full size that element takes up you only add margins.

```
.box{
   box-sizing:border-box;
   width:250px;
   padding:25px;
   margin:25px;
   border:5px solid #FFDA00;
}
```



CSS can be quite an unpredictable language to use. Browsers can treat CSS rules differently. One way to help is to use a CSS browser reset file.

A good one is normalize.css , you can copy the CSS code from **https://necolas.github.io/normalize.css/**

You can then save it as a separate CSS file or place it at the top of an existing CSS file. If you save it separately, then you need to link it from your HTML pages.

Normalize.css makes browsers render all elements more consistently and in line with modern standards.

Writing CSS layouts almost always involves trial and error and adjustments.

There can be multiple ways to achieve a similar effect but one may work better than another in a certain scenario.

Don't be afraid to try a method, copy your code and try another.

Always remember to ensure your browser is loading the newest version of the stylesheet. You can use cmd+r on a mac or f5 on a pc.

The display property allows you to define if and how an element is displayed. All HTML elements have a default display mode but you can overwrite them.

The 4 main options are block, inline, inline-block and none. There is also a new mode called flex

h1 = block (100% wide)

p = block (100% wide)

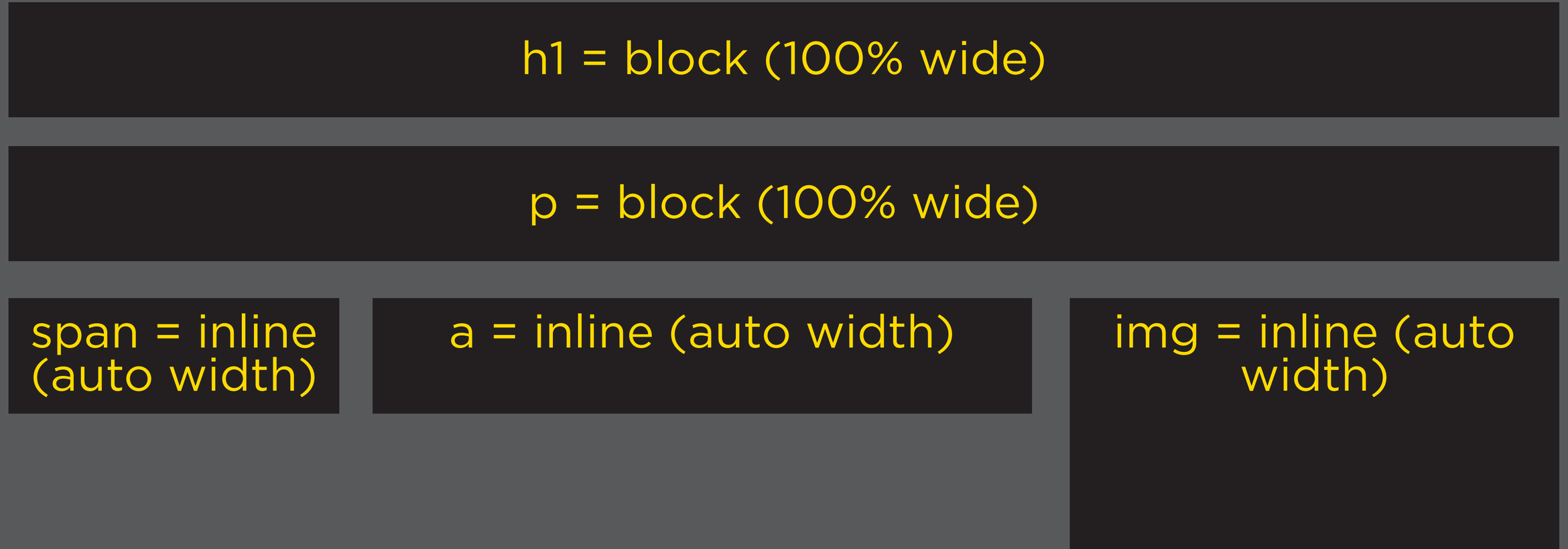
span = inline (auto width)

a = inline (auto width)

img = inline (auto width)

A block-level element starts on a new line and takes up the full width available.

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



One example where you may change display mode is with an image. Adding margin, padding, widths or heights to inline elements can be unpredictable.

```
img{
  display:block;
  margin:20px;
}
```

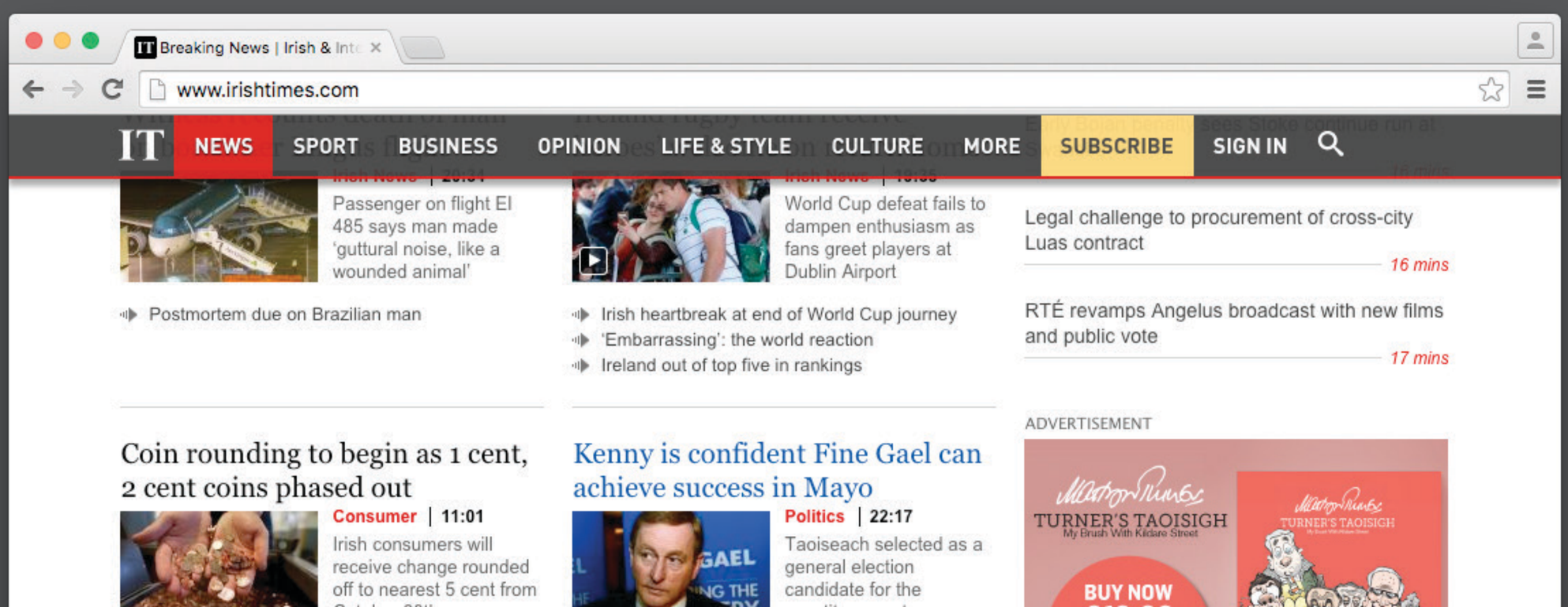
The position property allows you to take an element out of the flow of the page. The default position value is **static**. The other options are **relative**, **absolute** and **fixed**.

Setting an absolute or fixed position takes that element out of the flow of the page and elements that follows will fill in the gap left behind.

One problem is that you can be left with overlapping content so care is needed to keep an accessible, usable design.

position:fixed; is used to set a position relative to the browser window. A fixed element will not move if a user scrolls their window.

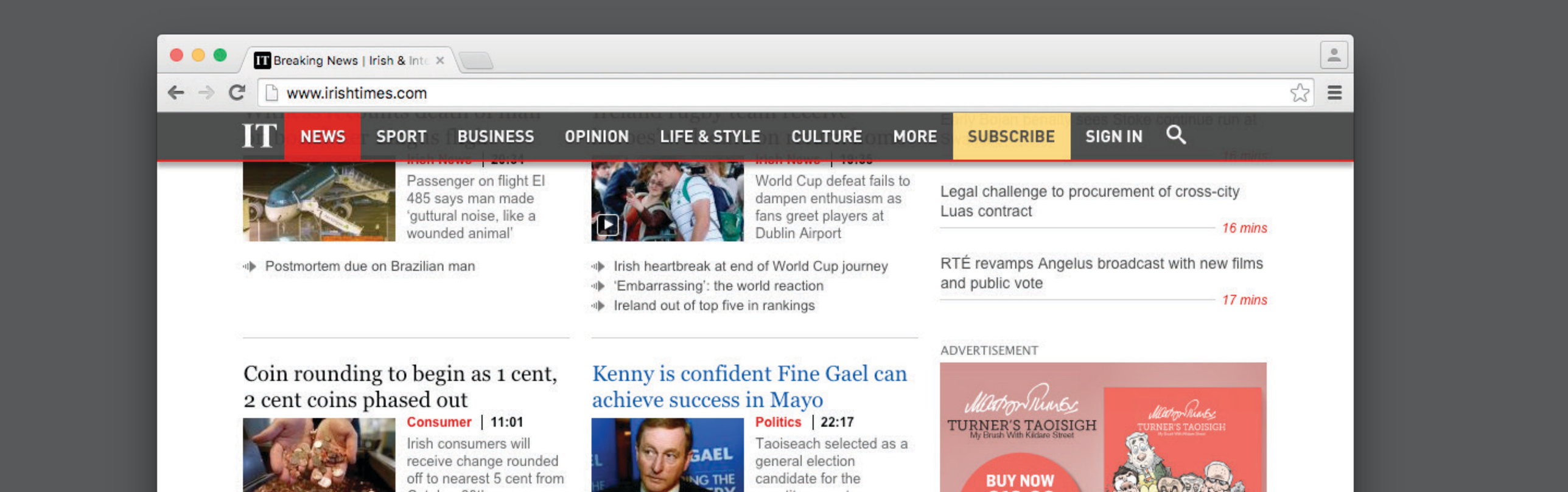
A common implementation of position:fixed; is for a sticky navigation or footer.



```
nav{

   position:fixed;
   width:100%;
   top:0;
   left:0;

}
```



position:absolute; allows you to position an HTML element relative to another element. Again it is taken out of the flow and other content fills the space.

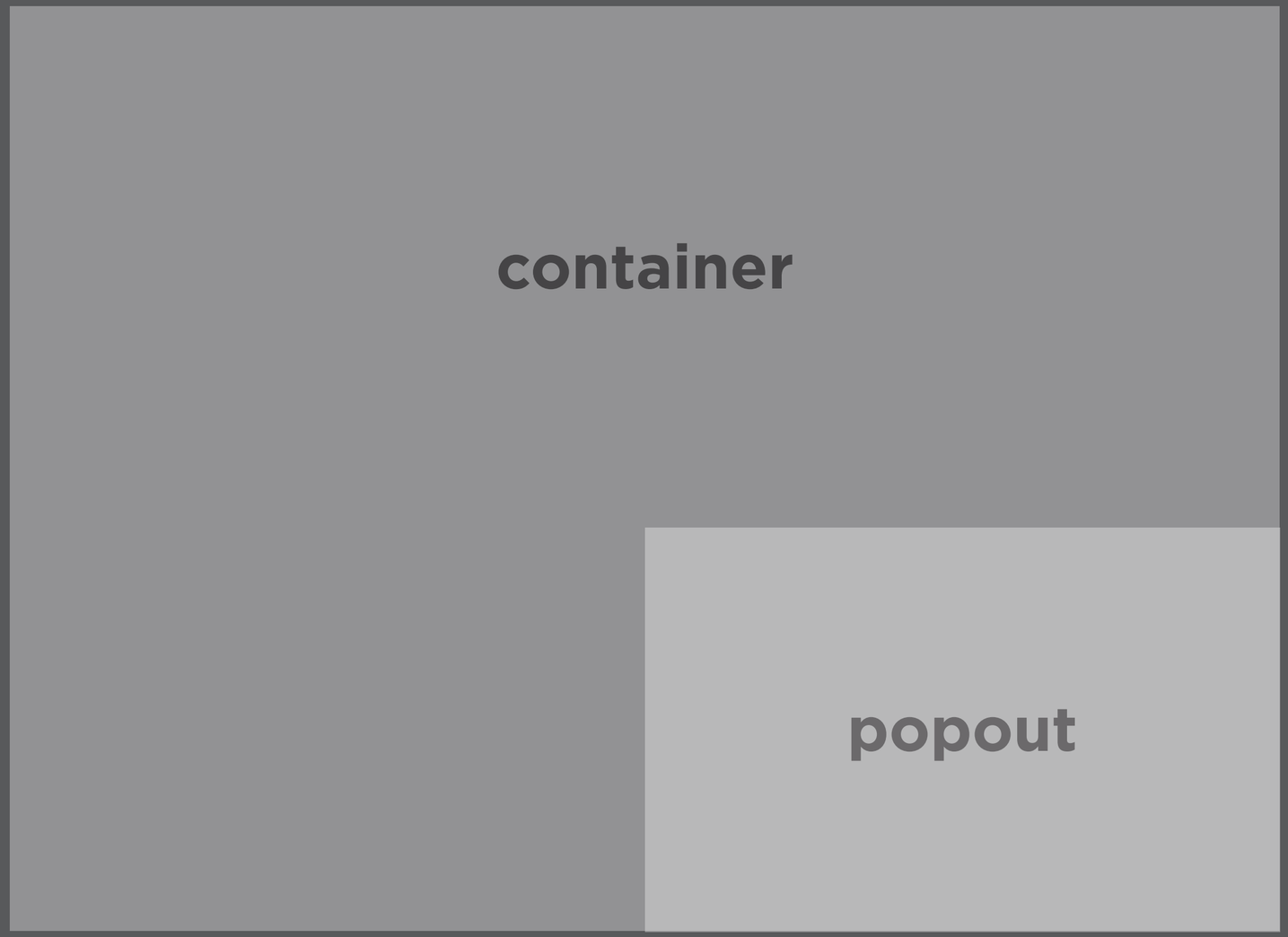
Instead of being positioned relative to the viewport, an element is positioned **relative to the nearest positioned parent**. A positioned parent is an element with position:relative;

If no parent is positioned, the element will be positioned relative to the browser window.

Unlike a fixed element, an absolute positioned element will move as the user scrolls.

```
.container{
    width:600px;
    position:relative;
}

.popout{
    position:absolute;
    background:orange;
    bottom:0;
    right:0;
    width:300px;
}
```

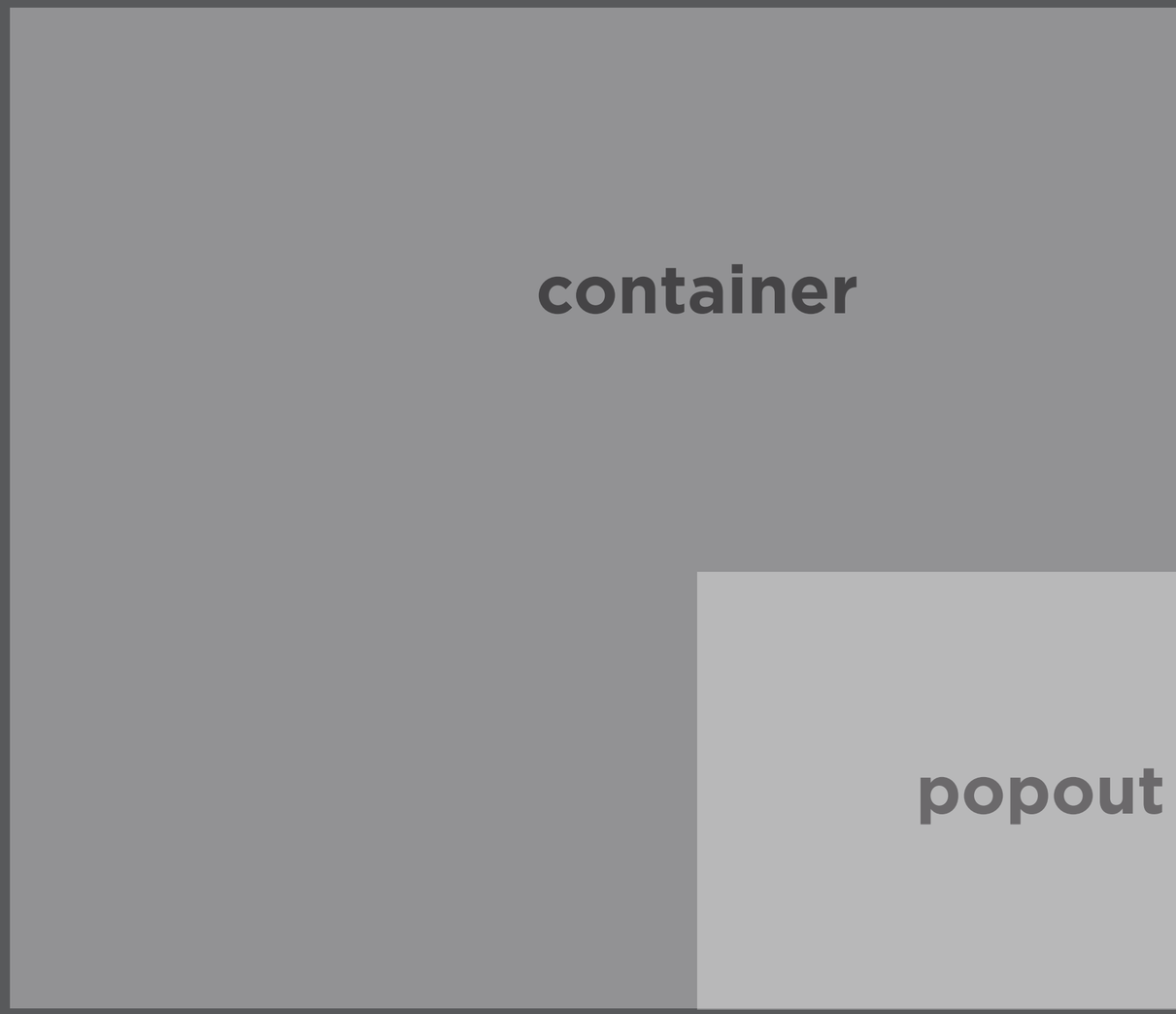


```
<div class="container">

    <p>Pellentesque habitant
morbi tristique senectus et netus
et malesuada fames ac turpis
egestas. Vestibulum tortor quam,
feugiat vitae, ultricies eget, tempor
sit amet, ante.</p>

    <div class="popout">
        <p>Pellentesque habitant
morbi tristique senectus et netus
et malesuada fames ac turpis
egestas. Vestibulum tortor quam,
feugiat vitae, ultricies eget, tempor
sit amet, ante.</p>
    </div>


</div>
```



When elements are positioned absolute or fixed, they can overlap other elements.

The z-index property is similar to the order of layers in Photoshop or Illustrator. It specifies which element should be placed in front or behind the others.

The higher the z-index the higher that element will display.

```
nav{
   z-index:10;
}
```

```
.popout{
   z-index:15;
}
```

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

```
img {
   float: right;
}
```

You can float an element to the left or to the right. Any content that follows will wrap around the element if possible.

Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Vestibulum tortor quam, feugiat vitae, ultricies eget, tempor sit amet, ante.

Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Vestibulum tortor quam, feugiat vitae, ultricies eget, tempor sit amet, ante.

Floating also takes an element out of the flow of the page.

By default the parent element of a floated element will disregard the height of the floated element. This can lead to problems.

If the content is not longer than the floated element, the height can be incorrect.

There is a fix for this, generally called clearfix.

A good fix is available at **nicolasgallagher.com/micro-clearfix-hack/**

Pellentesque habitant morbi tristique senectus.

Here is a simplfied version

```
.clearfix:before, .clearfix:after {
    content: " ";
    display: table;
}
.clearfix:after {
    clear: both;
}
.clearfix {
    *zoom: 1;
}
```

You can apply this clearfix in 2 ways.

The more efficient way is to apply it to the parent element that contains the floated element.

```
<div class="clearfix">
   <img src="images/cat.jpg">
   <p> Pellentesque habitant morbi
tristique senectus.</p>

</div>
```

Pellentesque habitant morbi tristique senectus.

You can also create a HTML element with a class clearfix just to fix the float.

The disadvantage with this method is that you are adding HTML elements that are solely required for layout.

Pellentesque habitant morbi tristique senectus.

```
<div>

   <img src="images/cat.jpg">
   <p> Pellentesque habitant morbi
tristique senectus.</p>
   <div class="clearfix"></div>

</div>
```

Using the same principles and the clearfix, you can **float divs** to make layouts.

```
.col{
   width:33%;
   float:left;
}
```

Pellentesque habitant morbi tristique senectus.

Pellentesque habitant morbi tristique senectus.

Pellentesque habitant morbi tristique senectus.

```
<div class="clearfix">
  <div class="col">
    <p> Pellentesque habitant morbi tristique senectus.</p>
  </div>
<div class="col">
    <p> Pellentesque habitant morbi tristique senectus.</p>
  </div>
<div class="col">
    <p> Pellentesque habitant morbi tristique senectus.</p>
  </div>
</div>
```

Pellentesque habitant morbi tristique senectus.

Pellentesque habitant morbi tristique senectus.

Pellentesque habitant morbi tristique senectus.

There is a new display mode called flex designed for layouts. Read more at css-tricks.com/snippets/css/a-guide-to-flexbox/

```
.container{
   display:flex;
}

.container div{
   flex-grow:1;
}

<div class="container">
   <div>...</div>
   <div>...</div>
   <div>...</div>
</div>
```

Pellentesque habitant morbi tristique senectus.

Pellentesque habitant morbi tristique senectus.

Pellentesque habitant morbi tristique senectus.

You can write CSS rules for certain screen sizes. These are generally split by breakpoints that suit certain screen types, e.g. tablet, smartphone, laptop/desktop and bigger monitors.

You write CSS rules as before. This difference is you wrap the Screen size rules with another set of curly brackets so the rules become nested.

```
@media screen and (max-width: 480px) {
    p{
        font-size:11px;
    }
    .container{
        padding:15px;
    }
}
```

CSS Assignment

Using the same HTML content, create 3 visually different versions an About page using just CSS. http://www.csszengarden.com demonstrates this principle.

The HTML content is at ixdncad.com under class #4

The only HTML changes you can make are:
- Path to the CSS files in the head
- Path and alt attributes for the 2 images
- The text in the <h1></h1> and <title></title> Tags

The 3 pages should represent:
- An architectural/design led consultancy
- A governmental body/department
- A new 'branded' drink/food company

Guidelines

- You **cannot** use Bootstrap or other CSS frameworks.
- You **can** use background images on any element.
- The intention is to explore the possibilities of CSS layouts, you will be graded on **how different** the 3 layouts are to each other.
- This is a demonstration of CSS so you can go be conceptual with your layouts and styles.
- Remember to use child parent selectors .main p{ You can't add new classes and ids to your HTML