Slide Note

Every art form has its ancient masters. In the case of websites, it is the newspapers.

When you dig into the basic principles of news design, overlap with the web are frequent and sometimes indistinguishable.

Above The Fold - A newspaper term that dates back to centuries. It is the content that you see when you land on a webpage.

The Gutenberg Principle - The Gutenberg Principle states that when faced with homogenous content, we start at the top left-hand corner and finish at the bottom right-hand corner, flicking from right to left as we go. It stems from an idea called reading gravity. Newspaper design tends to imitate that flow.

Nameplates - Every newspaper has a nameplate. It is the only thing you can guarantee that doesn't change from edition to edition. In both news and web design, the underlying purpose of the nameplate is the same, which is to get the brand front and center, guiding users to something they care.

Grid Systems - The grid system is foundational to newspaper design. As water shapes itself to a bowl, news content shapes itself to grid systems. It is full of information that needs to be well-organized and well-presented.

Facts: According to a survey, there are 831 million millennials representing one-quarter of the population. These millennials spend around 7.2 hrs of each day using different websites and mobile apps. On average, these millennials spend around 12 seconds on each page. Google shares that a new user spends only 3 seconds on a site to find what he wants, then hops on to another.

What Can Newspapers Teach Us About Web Design?

Above the Fold in Newspaper

Grid System

Slide Note

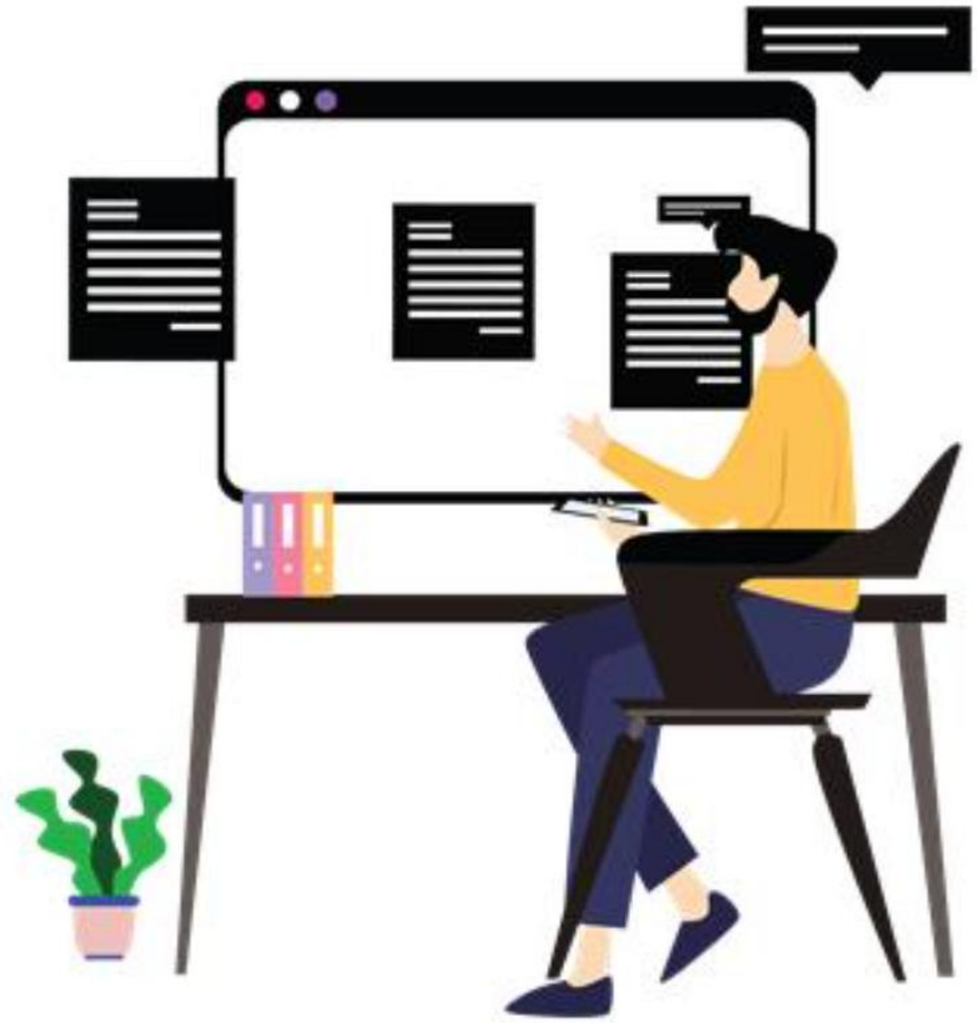
What was the headline that you saw in the newspaper?

How many columns did you see?

What Headline Do You See on the Newspaper?

Slide Note

Structure a Web Page Using CSS Floats



Slide Note

Learning Objectives

- Explain normal document flow
- Create multiple column layouts using floats
- Describe the relative and absolute positioning of CSS
- Use z-index and stack order to change the rendering order of HTML elements
- Make use of flexbox to create multiple column layouts

Slide Note

Normal Flow, or Flow Layout, is the way Block and Inline elements are displayed on a page before any changes are made to their layout. The flow is essentially a set of things that are all working together and know about each other in your layout. Once something is taken *out of flow* it works independently.

In a normal flow, **inline** elements display in the inline direction, that is in the direction words are displayed in a sentence according to the Writing Mode of the document. **Block** elements display one after the other, as paragraphs do in the Writing Mode of that document.

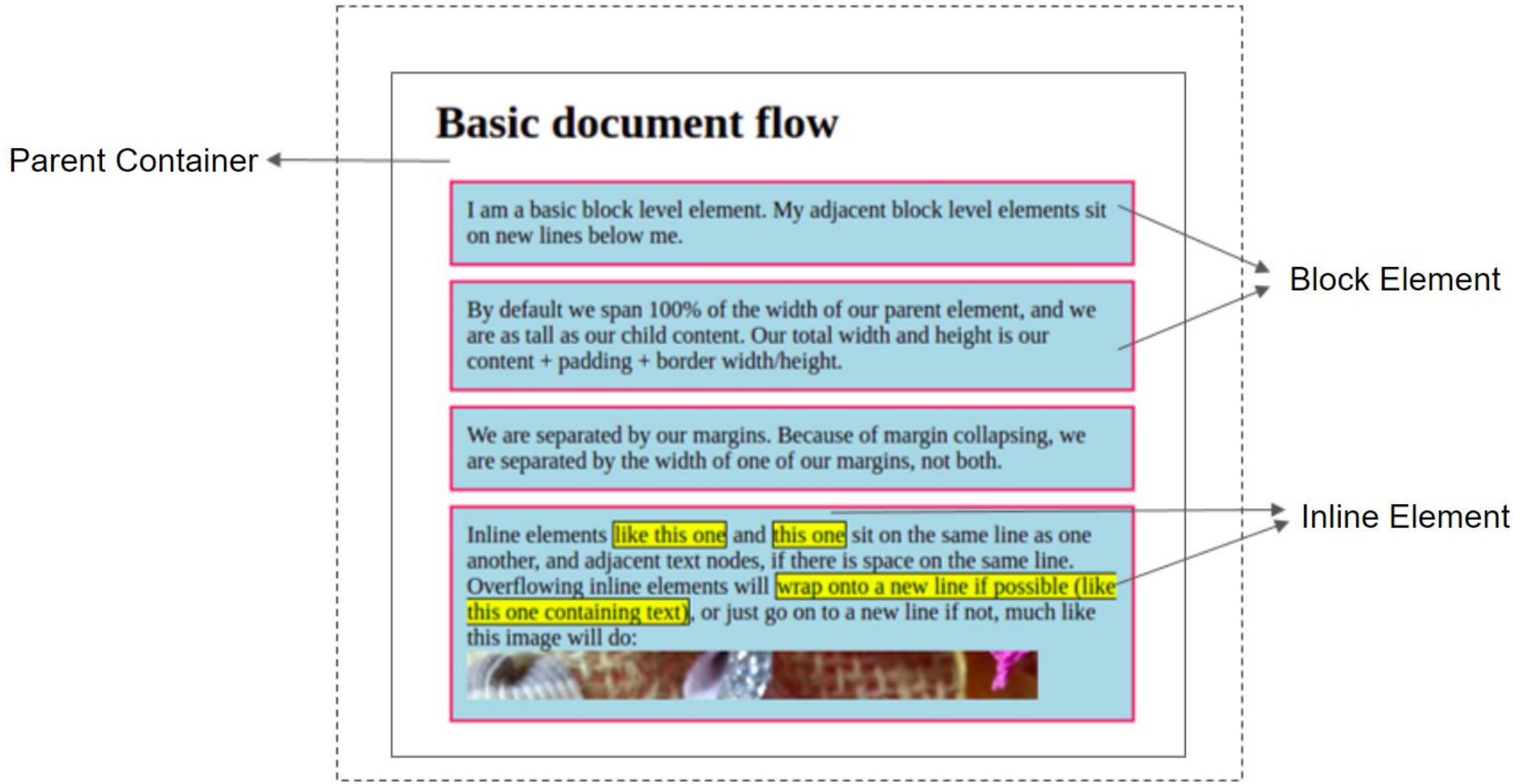
Here in the example, we have used

and , where

are block elements

are inline elements

Normal Document Flow



Slide Note

HTML and CSS Code

```
<h1>Basic document flow</h1>
<p>I am a basic block ….</p>
<p>By default we span ....</p>
<p>We are separated ...</p>
<p>
   Inline elements <span>like this one
   </span> and <span>this one</span> ….
   <span>wrap onto a new line if
   possible (like this one containing
   text)</span>, or just go on to a new
   line if not, much like this image
   will do:
    <img src="long.jpg">
</p>
```

```
body {
    width: 500px;
    margin: 0 auto;
}
p {
    background: lightblue;
    border: 2px solid rgb(255, 84, 104);
    padding: 10px;
    margin: 10px;
}
span {
    background: yellow;
    border: 1px solid black;
}
```

Slide Note

Normal Document flow: Uncovering how are elements laid out on a web page?

1. The elements' behavior can be changed either by adjusting their position in that normal flow or by removing them altogether.

2. By default:

   a. A block-level element's content is 100% of the width of its parent element.

   b. Inline elements are as tall and as wide as their content.

   c. The width or height of the inline element is adjustable.

   d. Block-level elements are laid out in the *block flow direction* (initial: horizontal-tb).

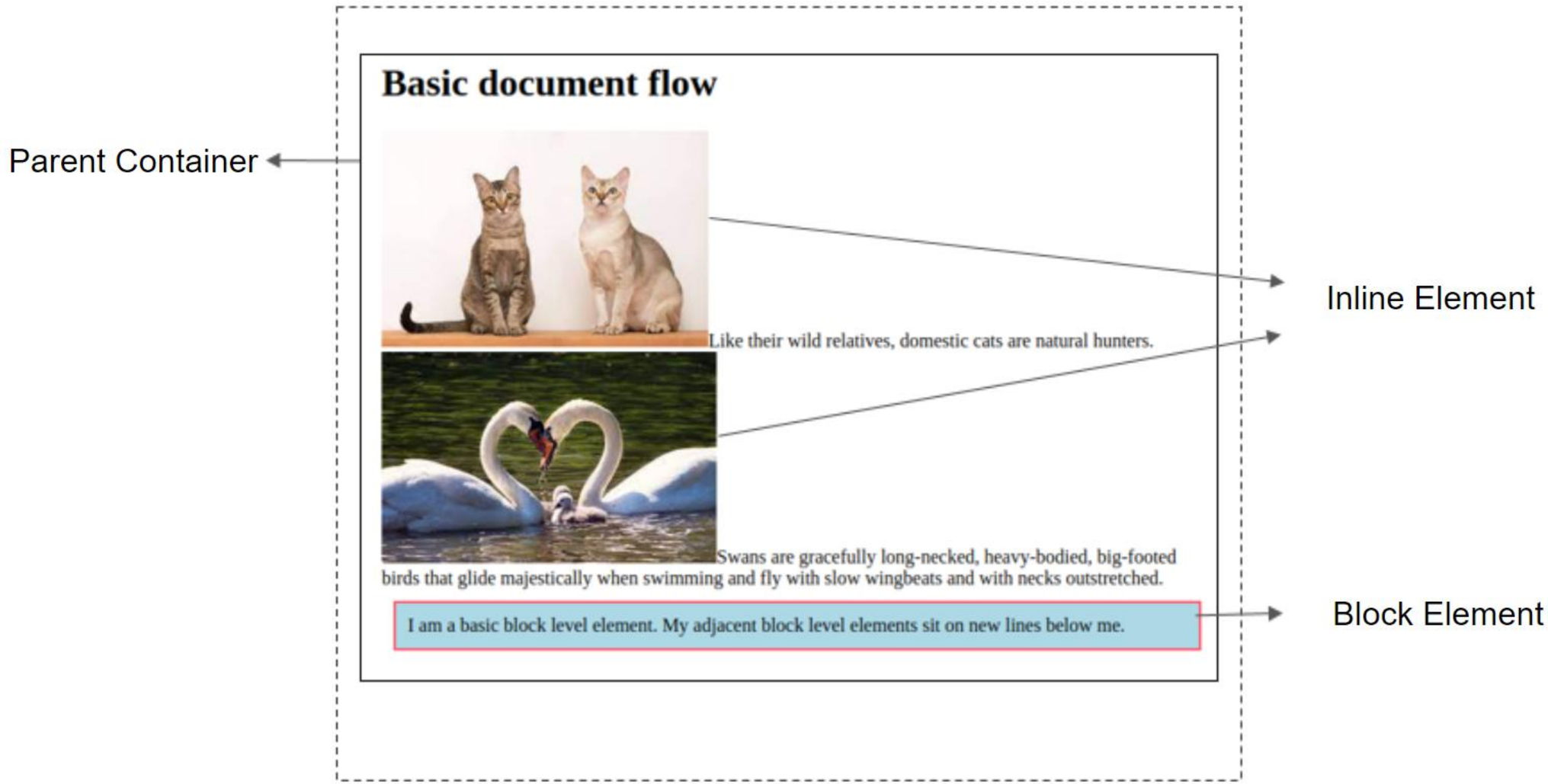
   e. Each block-level element will appear on a new line below the last one.

   f. For horizontal, top to bottom writing mode (for example, English), block-level elements are laid out vertically.

3. When there is space, the Inline elements sit on the same line.

4. When there is no space the overflowing elements or text will move down to a new line.

Normal Document Flow: Another Example



Slide Note

HTML and CSS Code

```
<h1>Basic document flow</h1>
<img src="cat.jpeg" alt="cats"> Like their
wild relatives, domestic cats are
natural hunters.
<img src="swan.jpeg" alt="cats"> Swans
are gracefully long-necked, heavy-
bodied, big-footed birds that glide
majestically when swimming and fly with
slow wingbeats and with necks
outstretched.
<p>I am a basic block level element.
 My adjacent block level elements sit
on new lines below me.
</p>
```

```
body {
    width: 700px;
    margin: 0 auto;
}
p {
    background: lightblue;
    border: 2px solid rgb(255, 84, 104);
    padding: 10px;
    margin: 10px;
}
```

Slide Note

Demonstrate the way block and inline elements are displayed on a page before any changes are made to their layout.

1. Normal document flow with paragraph, span and img elements.

Observe the normal flow of document for block elements like `

` which displays one after another and inline elements like `` which is just the size of their content. The `` element appears in the next line due to its bigger size.

2. Normal Document flow with multiple images of smaller size.

Observe the flow of document when inline elements like `` and `` with smaller size are added. They all sit on the same line along with any adjacent (or wrapped) text content as long as there is space for them.

Normal Document Flow

Demonstrate how the block and the inline elements are displayed on a page before any changes are made to their layout.

Click the link Normal Document flow demo code to check the demo code.

DEMO

Slide Note

CSS Floats

Slide Note

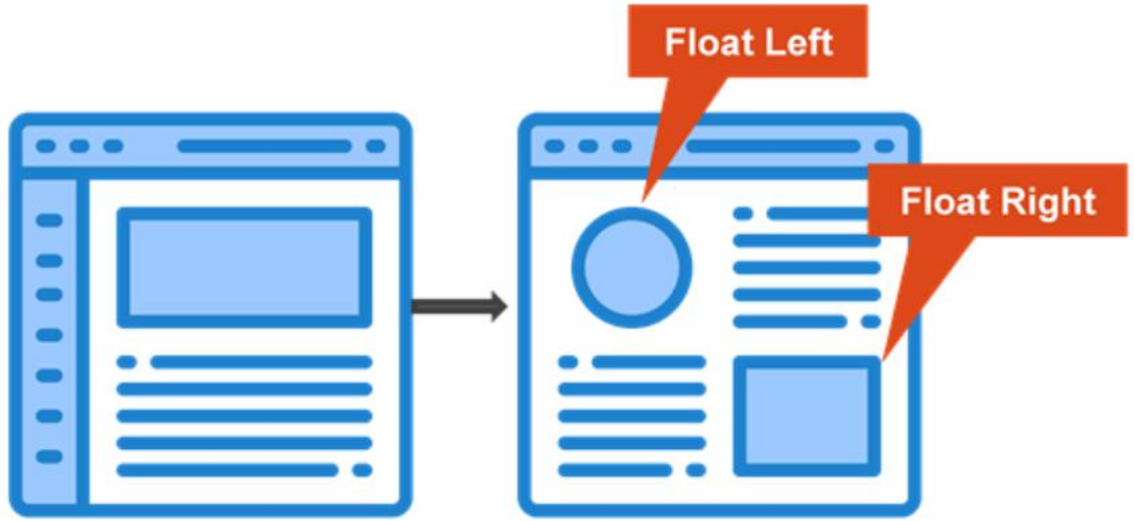
The float property was introduced to allow web developers to implement simple layouts involving an image

floating inside a column of a text, with the text wrapping around the left or right of it. This is similar to what you get in a newspaper layout.

But web developers quickly realized that you can float anything, not just images, so they use float broadened, for example to fun layout effects such as drop-caps.

Why Was CSS Floats Introduced?

The float property was introduced to implement simple layouts by having an image floating inside the column of a text, along with the text wrapping around its right or left. This is similar to what you might get in a newspaper layout.



The use of float is extended to create layout effects such as drop-caps.

Lorem ipsum dolor sit amet, consectetuer adipiscing elit, sed diam nonummy nibh euismod tincidunt ut laoreet dolore magna aliquam erat volutpat. Ut wisi enim ad minim veniam, quis nostrud exerci tation ullam sus-consequ cipit lobortis nisl ut aliquip ex ea commodo conse-quat. Duis autem vel eum iriure dolor in hendrerit in vulputate velit esse molestie consequat, vel illum dolore eu feugiat nulla facilisis at vero eros et accumsan et iusto odio dignissim qui blandit praesent luptatum zzril delenit augue duis dolore te feugait nulla facilisi.
Lorem ipsum dolor sit amet, cons ectetuer adipiscing elit, sed diam nonummy nibh euismod tincidunt ut laoreet dolore magna aliquam erat volutpat. Ut wisi enim ad minim veniam, quis nostrud exerci tation ullamcorper suscipit lobortis nisl ut aliquip ex ea commodo consequat.

Slide Note

Using Floats

How can we take out HTML elements from a normal document flow to create floats?

Slide Note

1. The Float element is taken out of the normal layout flow of the document and stuck to the left-hand side of its parent container (when CSS value is given as **left** for the float property)

.box{

float: left;

}

• Any content that comes below the floated element in the normal layout flow will now wrap around it.

• This will fill up space to the right-hand side of it till the floated element's top.

• Then, it will stop.

How Do Floats Work?

HTML Code:

```
<body>
  <h1>Simple float example</h1>
  <div class="box" >
     <img src="cats.jpg">
  </div>
  <p>Lorem ipsum ….</p>
  <p>Nam vulputate ....</p>
  <p>Sed auctor cursus ...</p>
</body>
```

CSS Code:

```
.box {
    float: left;
    margin-right: 15px;
   }
```

Slide Note

Floating the content to the right has the same effect but in reverse.

Float Position Changed

HTML Code:

```
<body>
  <h1>Simple float example</h1>
  <div class="box" >
     <img src="cats.jpg">
  </div>
  <p>Lorem ipsum ...</p>
  <p>Nam vulputate ....</p>
  <p>Sed auctor cursus ...</p>
</body>
```

CSS Code:

```
.box {
    float: right;
    margin-left: 15px;
   }
```

Slide Note

Demonstrate floats by using the CSS float property. Check how a floating element is removed from the normal flow of a document.

1. Floating elements

An `` is added to a container and made to float left to the following `

` text content.

2. Floating elements removed from normal flow

The `

` following the image is styled with properties like padding, background-color and color. To make the effect easier to see, change the "margin-left" on your float to "margin" so you get space all around the float. You can see the background on the paragraph running right underneath the floated image box.

CSS Floats

Demonstrate floats by using the CSS float property. Check how a floating element is removed from the normal flow of a document. Use the following link to execute the demo code.

Simple Float demo code

DEMO

Slide Note

Compare the image shown in this slide and in the "**How Does the Float Work?"** slide.

1. Find the difference between the current layout and the previous layout.

2. You can see that there are additional spaces around the blue box.

3. Now the question that comes up is, how can this be done? There are two ways.

   1. Add a margin to the box.

   2. Add a margin or padding to the text.

4. The second way is not possible in this case.

5. While a margin can be added to the Float to push the text away, a margin cannot be added to the text to move it away from the Float.

6. This is because a floated element is taken out of the normal flow and the boxes of the following items run behind the Float.

What do You See in the Preview Image?

HTML Code:

```
<body>
  <h1>Simple float example</h1>
  <div class="box" >
     <img src="cats.jpg">
  </div>
  <p class="special">Lorem ipsum ...</p>
  <p>Sed auctor cursus ...</p>
  <p>Nam vulputate ....</p>
</body>
```

CSS Code:

```
.box {
    float: left;
    margin-right: 15px;
   }
.special {
    background-color: blue;
    padding: 10px;
    color: #fff;
   }
```

Slide Note

You have seen that float is removed from the normal flow and the other element is displayed beside it. So, if you want to stop the following element from moving up, you need to clear them; This can be achieved through clear property.

Clearing floats – Clear property can be used to stop the following element from moving up. In our case, the second paragraph is added with a new class called cleared and is also given the following property:

.cleared {

clear:left;

}

You should see that the following paragraph clears the floated element and that it no longer comes up alongside.

The clear property accepts the following values:

• left: Clear items floated to the left.

• right: Clear items floated to the right.

• both: Clear any floated items, left or right.

What do You Notice in This Preview Image?

HTML Code:

```
<body>
  <h1>Simple float example</h1>
  <div class="box" >
     <img src="cats.jpg">
  </div>
  <p class="special">Lorem ipsum ...</p>
  <p class="cleared">Sed auctor cursus..</p>
  <p>Nam vulputate ....</p>
</body>
```

CSS Code:

```
.box {
    float: left;
    margin-right: 15px;
   }
.special {
    background-color: blue;
    padding: 10px;
    color: #fff;
   }
.cleared {
    clear: left;
   }
```

Slide Note

Demonstrate what elements can float beside the cleared element, and on which side using the CSS clear property.

1. Clear floats

Clear property can be used to stop the following element from moving up. In our case, the second paragraph is added with a new class called cleared and is also given the following property:

```css
.cleared {
clear:left;
}
```

You should see that the following paragraph clears the floated element and that it no longer comes up alongside.

2. Clear-fix Hack: Clearing boxes wrapped around a float

Change the document so that the first paragraph and the floated image are wrapped with a `
` with a class of wrapper.
Insert some generated content after the box that contains both the float and the content wrapping around it, then setting that generated content to clear both.

Clearing Floats

Demonstrate what elements can float beside the cleared element, and on which side using the CSS clear property. Click on Floats with clear demo code to execute the demo codes.

DEMO

Slide Note

Let's see what happens if you have a tall float and a short paragraph, with a box wrapped around both elements. Change the document so that the first paragraph and the floated image are wrapped with a with a class of wrapper.

If you want the bottom of the box to wrap both the floated item and the wrapping content, even if the content is shorter, one of the best solutions is to use the clearfix hack.

Clearfix Hack: This involves inserting some generated content after the box, which contains both the float and wrapping content that is set to clear both.

Using Overflow:

An alternative method is to set the overflow property of the wrapper to a value other than visible.

Remove the clearfix CSS you added in the last section; instead add overflow: auto to the rules for wrapper. Once again, the box should clear.

Clear Fix Hack

HTML Code:

```
<body>
  <h1>Simple float example</h1>
  <div class="wrapper" >
  <div class="box" >
     <img src="cats.jpg">
  </div>
  <p>Lorem ipsum ...</p>
  <p class="cleared">Nam vulputate ....</p>
  <p>Sed auctor cursus..</p>
</body>
```

CSS Code:

```
.box {
    float: left;
    margin-right: 15px;
   }
.wrapper {
    background-color: blue;
    padding: 10px;
    color: #fff;
   }
.wrapper::after {
     content: "";
     clear: both;
     display: block;
   }
```

Slide Note

Quick Check

Select the most appropriate CSS declaration to position an element so that other elements that follow it may occupy space to the right of it.

a. float : none;

b. float : right;

c. float : left;

d. clear : right;

Slide Note

Quick Check: Solution

Select the most appropriate CSS declaration to position an element so that other elements that follow it may occupy space to the right of it.

a. float : none;

b. float : right;

c. **float : left;**

d. clear : right;

Slide Note

CSS Positioning

Slide Note

The **position** CSS property sets how an element is positioned in a document. The top, right, bottom, and left properties determine the final location of the positioned elements.

Types of CSS Positioning

- **Static**: This is the default position that every element gets. It means, the element is positioned according to the normal flow of the page.

- **Absolute**: The element is positioned relative to the nearest positioned ancestor. If it has no positioned ancestor, it uses the document body and moves along with page scrolling. They are removed from the normal flow and can overlap elements.

- **Relative**: This is similar to Static positioning, but once the positioned element has taken its place, the final position, including the overlap feature, can be modified.

- **Fixed**: Usually fixes an element in place relative to the visible portion of the viewport, so it always stays in the same place even while scrolling. The element is removed from the normal document flow, and no space is created for the element in the page layout.

Slide Note

HTML elements are positioned static by default.

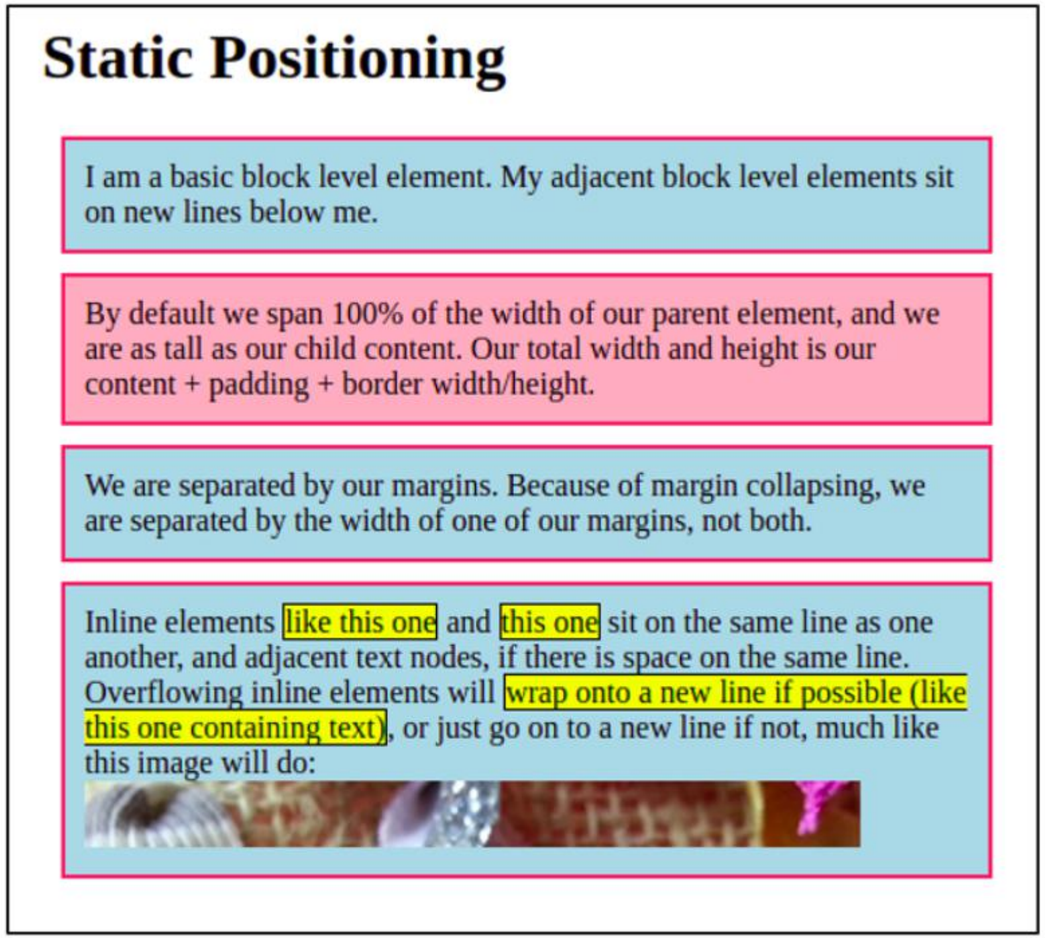
Static positioned elements are not affected by the top, bottom, left, and right properties.

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

Static Positioning

Static: The element is always positioned according to the normal flow of the page. HTML elements are positioned static by default.

```
.positioned {
    position: static;
    background: lightpink;
}
```



Slide Note

Use the HTML code of the normal document flow example and add class "positioned" to the second paragraph.

Later, apply the styles as specified in the slide.

HTML and CSS Code: Static Positioning

```
<h1>Static Positioning</h1>
<p>I am a basic block ….</p>
<p class="positioned">By default we spa
n...</p>
<p>We are separated ...</p>
<p>
   Inline elements <span>like this one
   </span> and <span>this one</span> ….
   <span>wrap onto a new line if
   possible (like this one containing
   text)</span>, or just go on to a new
   line if not, much like this image
   will do:
    <img src="long.jpg">
</p>
```

```
body {
    width: 500px;
    margin: 0 auto;
}
p {
    background: lightblue;
    border: 2px solid rgb(255, 84, 104);
    padding: 10px;
    margin: 10px;
}
span {
    background: yellow;
    border: 1px solid black;
}
.positioned {
    position: static;
    background: lightpink;
}
```

Slide Note

Demonstrate the various types of CSS position property values.

1. Static positioning is the default positoning. Providing values as `position:static` to `

` element does not change the document flow.

2. Relative positioning : Provide `position:relative` and positive values for top and left css properties to `

` and observe the change in the final position of the `

` after taking its initial position in the normal document flow.

3. Relative positioning: Provide `position:relative` and negative values for top and left css properties to `

` and observe the change in the final position of the `

` aftre taking its initial position in the normal document flow.

4. Absolute positioning: Provide `position:absolute` and top, left properties to the second `

` element and observe that the positioned element is no longer exists in the normal document flow. It is positioned relative to the document body and hence overlap with other `

` elemnents.

5. Fixed positioning: Provide `position:fixed` and `top:0` to the heading and observe that the heading sticks to the top of the screen (staying fixed) and the contents appears to scroll up and disappear underneath it.

CSS Position Property

Demonstrate the various types of CSS position property values. Click on various positioning types to execute demo codes.

DEMO

Slide Note

Use the HTML code of the normal document flow example and add class "positioned" to the second paragraph.

Later, apply the styles as specified in the slide.

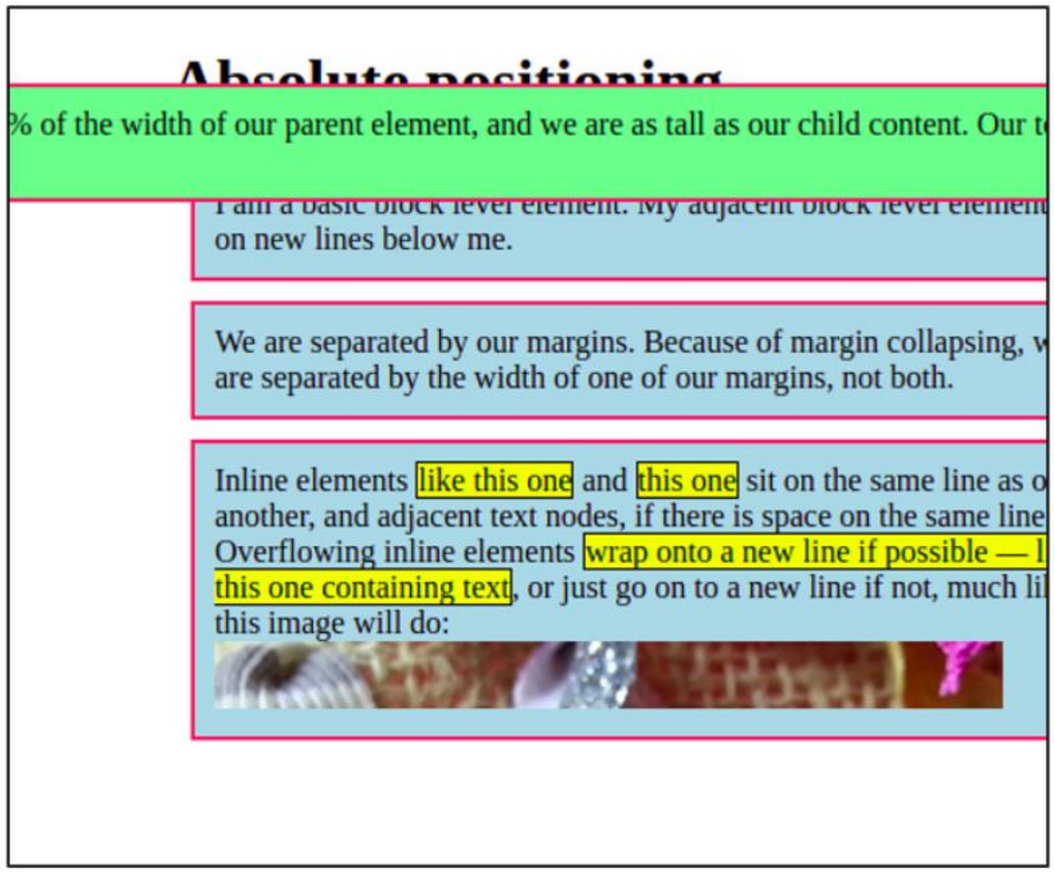
Absolute positioning is the trickiest positioning value, as it behaves like a fixed value except relative to the nearest positioned ancestor rather than being relative to the viewport.

Setting the position property values for top and left of an absolutely-adjusted element moves the element away from the normal position to the top-left position of the screen. Other content elements adjust to fit into that gap, which is left behind by moving the element.

Absolute Positioning

Absolute: The element is positioned relative to the nearest positioned ancestor.

```
.positioned {
    position: absolute;
    background: lightgreen;
    top: 30px;
    left: 140px;
}
```



Slide Note

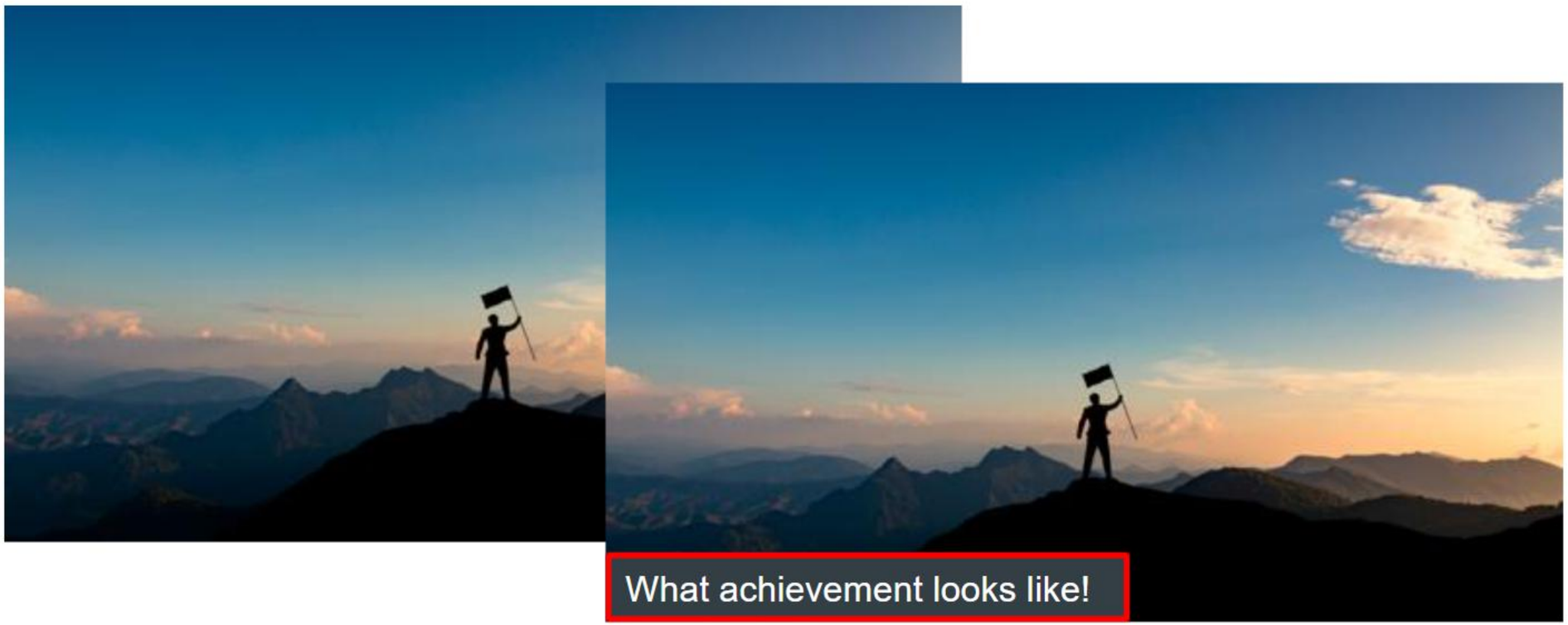
One of the use cases of absolute positioning is to add a caption to an image.

Any content added after the tag will be in normal flow, so it will appear in the next line following the image.

To write a caption over the image, one has to use the absolute positioning to fix this problem.

Depending on the position where the caption should be placed over the image, absolute positioning property will be used.

Adding Captions to an Image Using Absolute Positioning



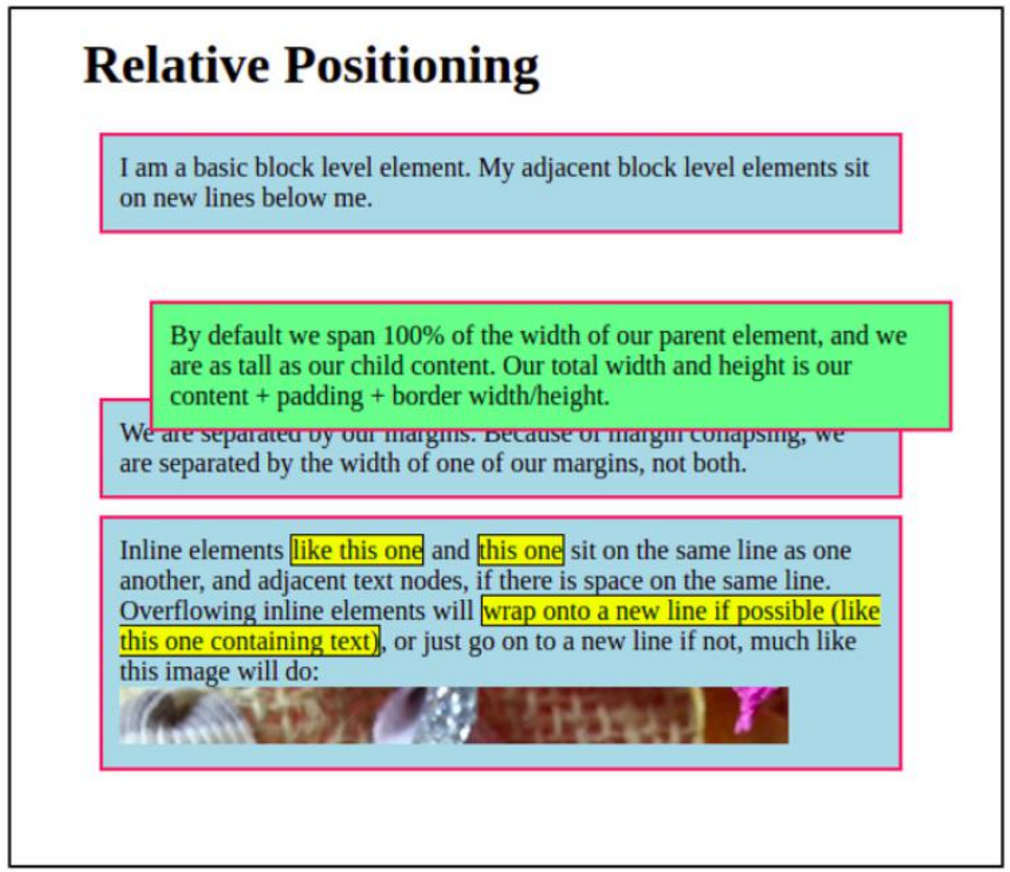
Slide Note

Setting the values of top and left properties of a relatively-positioned element moves the element away from its normal position as it does in absolute positioning, but over here, the other content elements do not adjust their positions to fill in the gap left by the recently moved element.

Relative Positioning

Relative: An element is positioned relative to its normal position.

```
.positioned {
    position: relative;
    background: lightgreen;
    top: 30px;
    left: 30px;
}
```



Slide Note

Use the HTML code of the normal document flow example and add class "positioned" to the second paragraph.

Later, apply the styles as specified in the slide.

HTML and CSS Code

```
<h1>Relative Positioning</h1>
<p>I am a basic block ….</p>
<img src="swan.jpeg" class="positioned"
>
<p>We are separated ...</p>
<p>
   Inline elements <span>like this one
   </span> and <span>this one</span> ….
   <span>wrap onto a new line if
   possible (like this one containing
   text)</span>, or just go on to a new
   line if not, much like this image
   will do:
    <img src="long.jpg">
</p>
```

```
body {
    width: 500px;
    margin: 0 auto;
}
p {
    background: lightblue;
    border: 2px solid rgb(255, 84, 104);
    padding: 10px;
    margin: 10px;
}
span {
    background: yellow;
    border: 1px solid black;
}
.positioned {
    position: relative;
    background: lightgreen;
    top: 30px;
    left: 30px;
}
```

Slide Note

Use the HTML code of the normal document flow example and add class "positioned" to the newly added image element which has replaced the second paragraph.

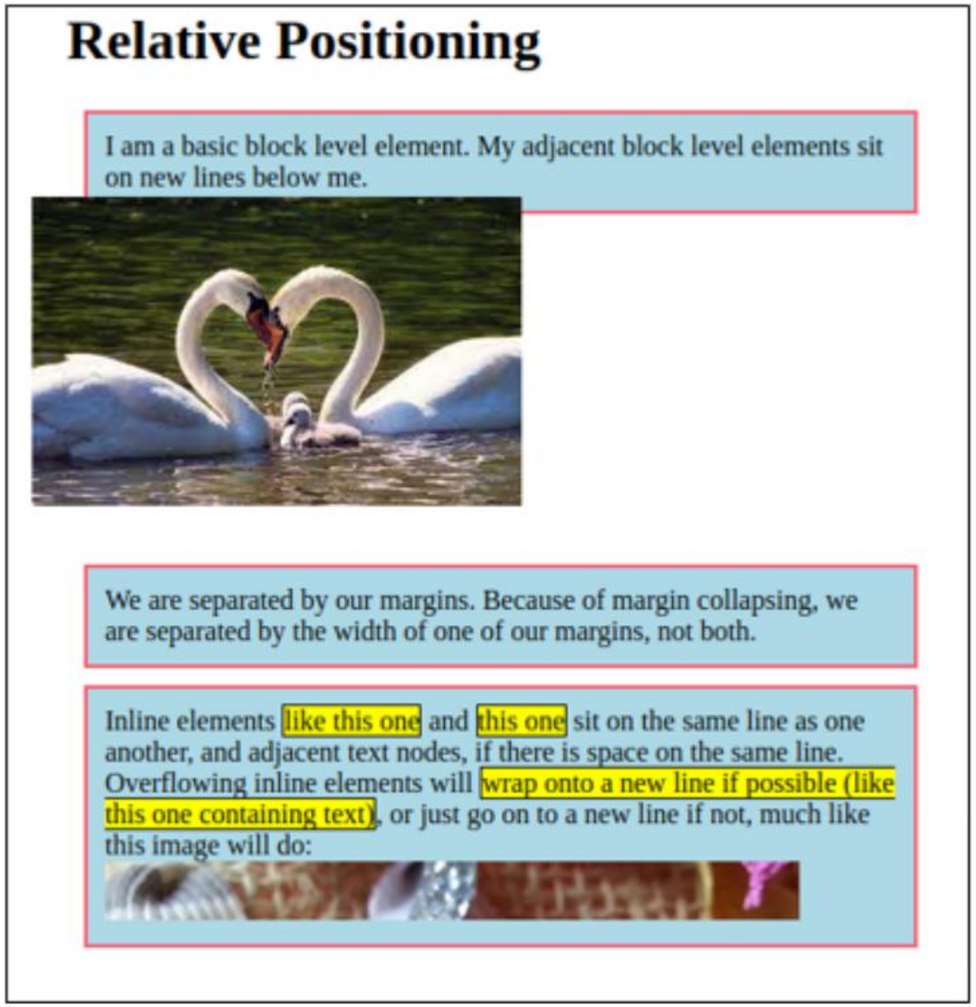
Later, apply the styles as specified in the slide.

Setting the values of top and left properties of a relatively-positioned element moves the element away from its normal position as it does in absolute positioning, but over here, the other content elements do not adjust their positions to fill in the gap left by the recently moved element.

Relative Positioning: Another Example

Relative: Setting the top, right, bottom, and left properties of a relatively-positioned element will cause it to be adjusted away from its normal position. It will not adjust other content to fit into any gap left by the element.

```
.positioned {
    position: relative;
    top: -20px;
    left: -20px;
}
```

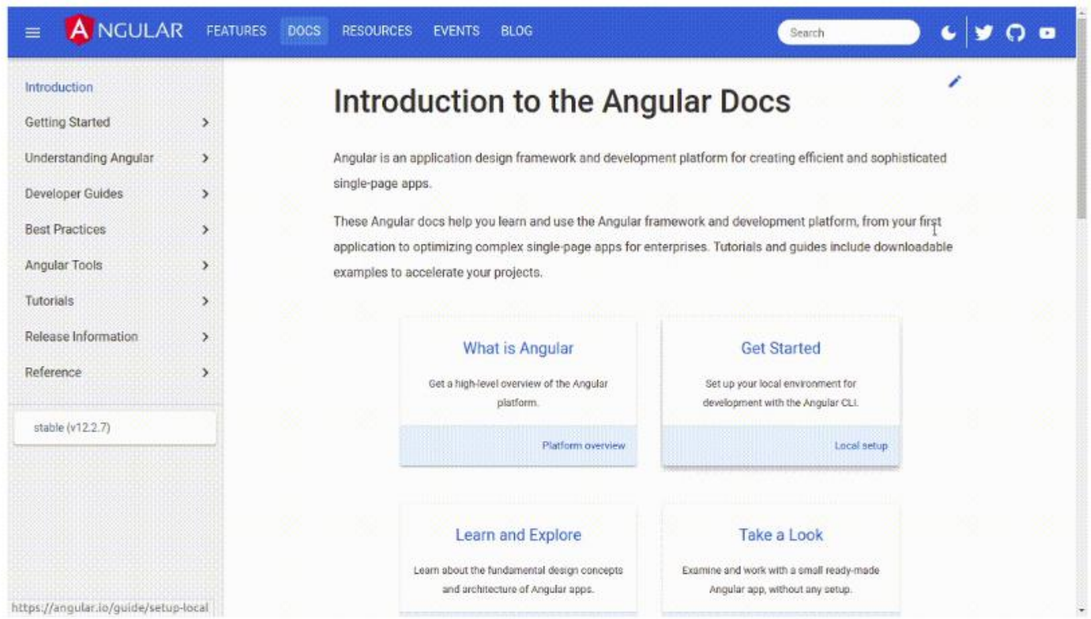


Slide Note

The above video is an implementation of the fixed position used for navbars in an angular website. See how the top navbar with the search box is fixed, and the contents below are displayed while scrolling vertically.

Fixed Navigation Bar: Fixed Position Implementation

Fixed: The element is positioned relative to the viewport, which means it always stays in the same place even on scrolling the page.




```
h1 {
    position: fixed;
    width: 500px;
    top: 0;
    margin-top: 0;
   padding: 10px;
    background: white;
}
p:nth-of-type(1) {
    margin-top: 60px;
}
```

Credit: webfx.com

Slide Note

Quick Check

_____ positioning makes an element appear in the same place relative to the browser window even while scrolling.

a. Relative

b. Absolute

c. Fixed

d. Static

Slide Note

Quick Check: Solution

_____ positioning makes an element appear in the same place relative to the browser window even while scrolling.

a. Relative

b. Absolute

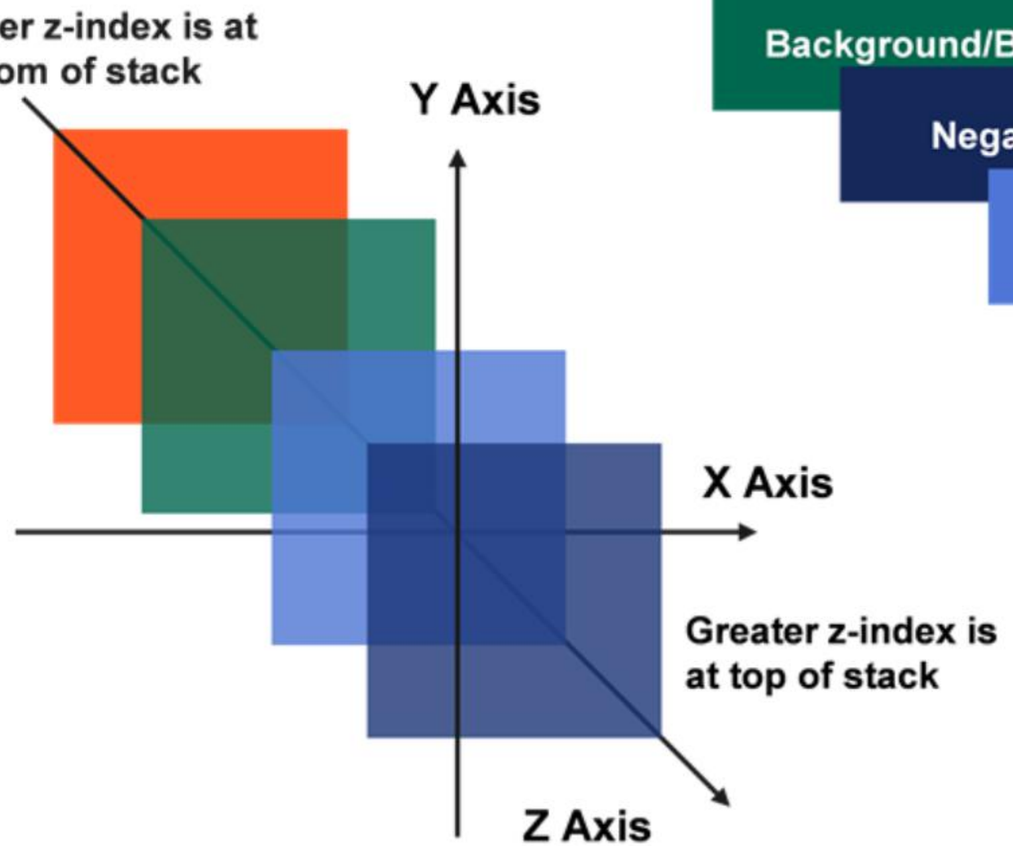
c. **Fixed**

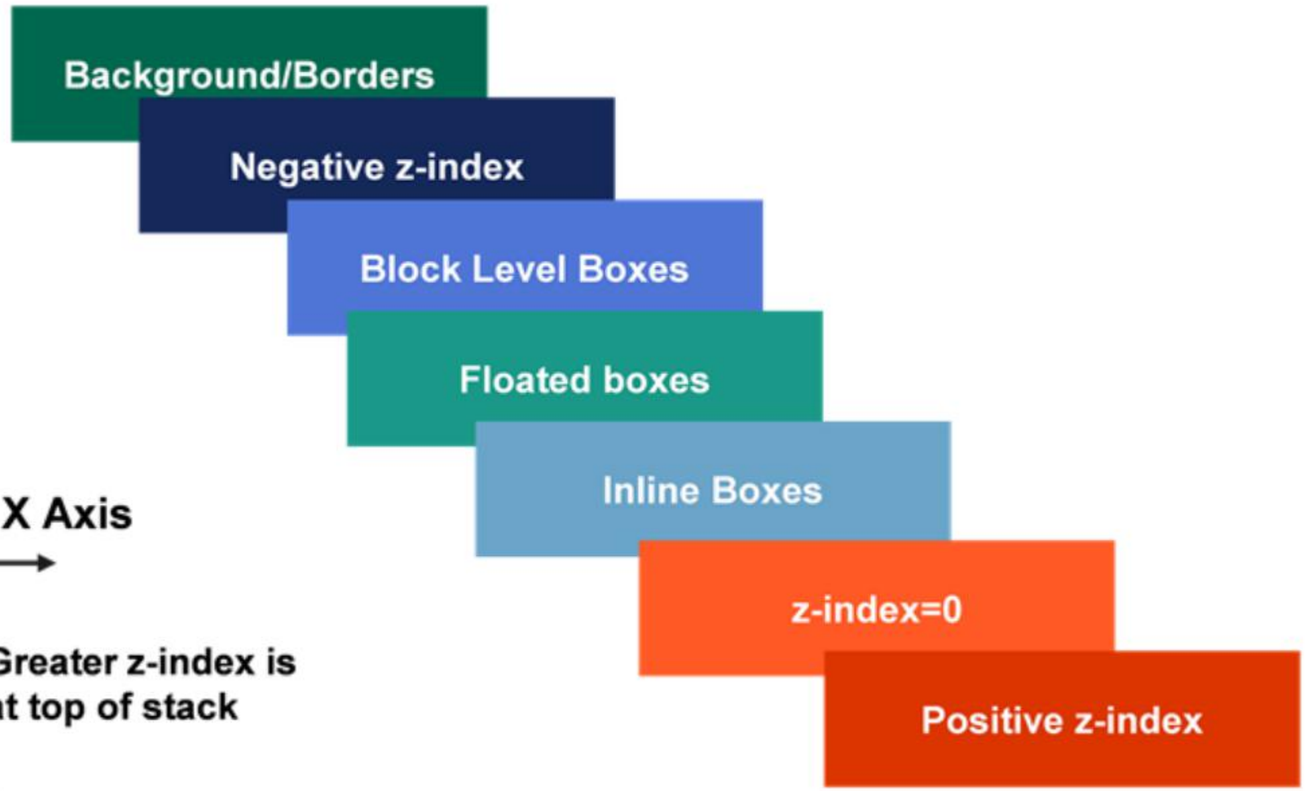
d. Static

Slide Note

z-index and Stack Order

We can change the stack order of an element using the z-index property.





Slide Note

Change the order of the HTML elements displayed in the browser using the z-index property.

Both lime and yellow paragraphs are absolutely positioned.

The lime paragraph is given an index value as 1 which is greater than the yellow paragraph(default z-index value 0).

Hence lime paragraph overlaps the yellow paragraph.

z-index and Stack Order

Let us change the order of the HTML elements displayed in the browser using the z-index property. Click the following link to know more.

z-index stacking order demo code

DEMO

Slide Note

z-index values affect where positioned elements sit on that axis. Positive values move them higher up the stack, and negative values move them lower down the stack.

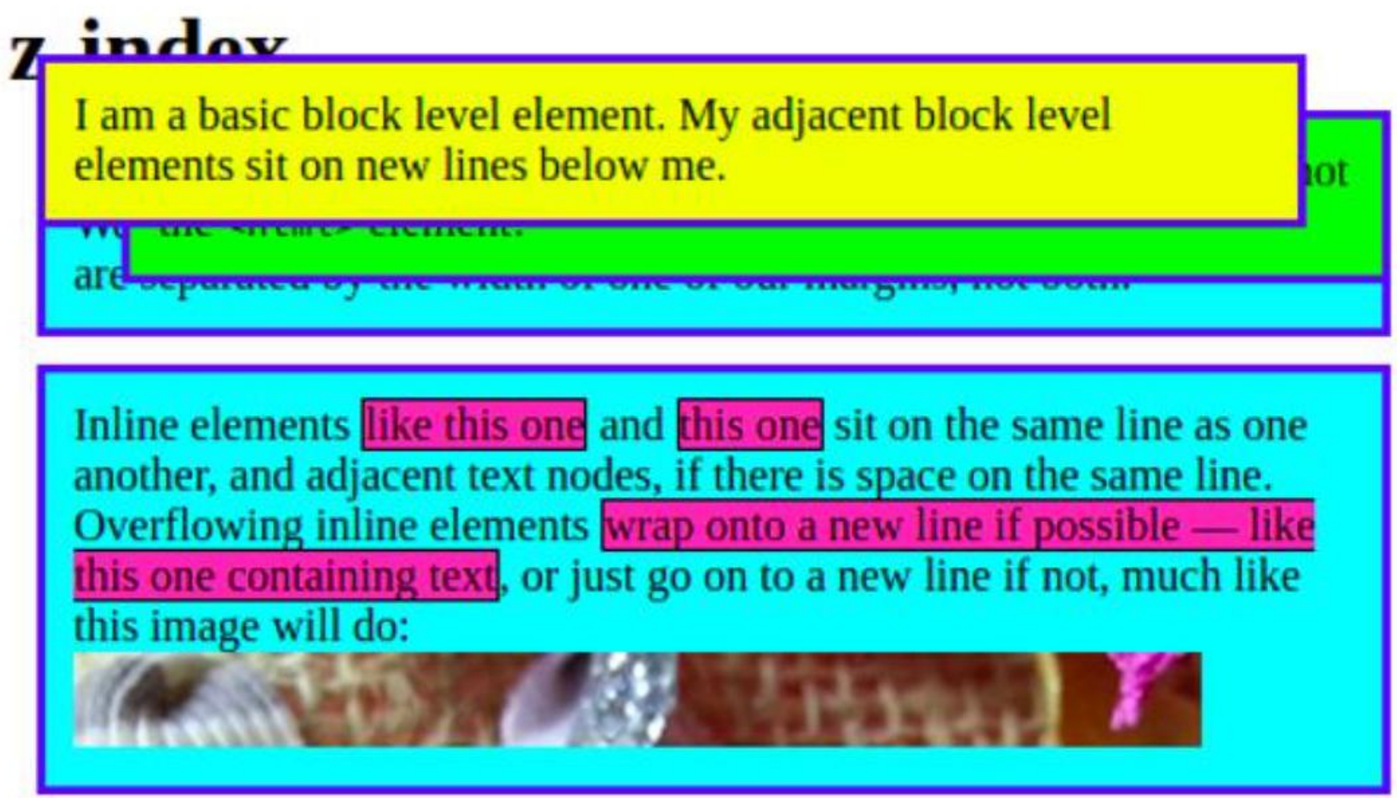
Changing the Stack Order of Elements

- The z-index property in CSS controls the vertical stacking order of elements that overlap.
- "z-index" is a reference to the z-axis.
- Web pages also have a z-axis; an imaginary line that runs from the surface of your screen towards your face.
- z-index values affect where positioned elements sit on that axis; positive values move them higher up the stack, and negative values move them lower down the stack.
- By default, all positioned elements have a z-index of auto, which is effectively 0.

Slide Note

The slide shows the output image for explaining the z-index property.

z-index and Stack Order



Slide Note

Use the HTML code of the normal document flow example.

Later, apply the styles as specified in the slide for the second and the first paragraph.

The images shown use absolute positioning for the first two paragraphs. As they're not in the normal flow, they get overlapped. Since the first paragraph (yellow color) is given the z-index value as 1, it's above the "positioned" paragraph (lime color).

HTML and CSS Code

```
<h1>z-index</h1>
<p>I am a basic block...</p>
<p class="positioned">
  Now I'm absolutely positioned ...
</p>
<p>We are separated....</p>
<p>
  Inline elements <span>like this one
  </span> and <span>this one</span> …
  .<span>wrap onto a new line if
  possible (like this one containing
  text)</span>,
  or just go on to a new line if not,
  much like this image will do:
  <img src="long.jpg">
</p>
```

```
.positioned {
    position: absolute;
    background: lime;
    top: 30px;
    left: 30px;
}
p:nth-of-type(1) {
    position: absolute;
    background: yellow;
    top: 10px;
    right: 30px;
    z-index: 1;
}
```

Slide Note

Think and Tell

- In some ways, creating CSS layouts using floats and positioning can be limiting and frustrating.
- Some of the following simple layout designs are either difficult or impossible to achieve using these tools.
  - Vertically centering a block of content inside its parent
  - Making all the children of a container take up an equal amount of the available width/height, regardless of how much width/height available.
  - Make all columns in a multiple-column layout adopt the same height even if they contain a different amount of content.
- How to makes the layout tasks easier?

Flexbox

Slide Note

The slide displays the output image used for explaining the flexbox layout.

Flexbox

Sample flexbox example

Flexbox

Flexbox is a one-dimensional layout method for arranging items in rows or columns. Items flex (expand) to fill additional space or shrink to fit into smaller spaces..

Why Flexbox?

For a long time, the only reliable cross-browser compatible tools available for creating CSS layouts were features like floats and positioning. These work, but in some ways they're also limiting and frustrating. Some of the simple layout designs are either difficult or impossible to achieve with such tools. Flexbox makes a lot of layout tasks much easier.

How Flexbox Works?

To start with, we need to select which elements are to be laid out as flexible boxes. To do this, we set a special value of display on the parent element of the elements you want to affect.

The element we've given a display value of flex to is acting like a block-level element in terms of how it interacts with the rest of the page, but its children are laid out as flex items. The next section will explain in more detail what this means. Note also that you can use a display value of inline-flex if you wish to lay out an element's children as flex items, but have that element behave like an inline element.

Slide Note

Flexbox

Create an easy multi-column layout using the flex display property. Check this demo code to understand it in detail.

DEMO

Slide Note

A header element with a top-level heading is inside it and a section element containing three articles. We have given display:flex for the parent element called
to create a standard three-column layout for the
elements.

You will learn about Flexbox layout in detail in the next sprint.

HTML and CSS Code

```
<header>
   <h1>Sample flexbox example</h1>
</header>
<section>
  <article>
     <h2>Flexbox</p>
     <p>Flexbox is...</p>
  </article>
  <article>
     <h2>Why Flexbox?</p>
     <p>For a long time...</p>
  </article>
  <article>
     <h2>How Flexbox works?</p>
     <p>To start with ...</p>
  </article>
</section>
```

```
header{
    background: purple;
    height: 100px;
}
h1{
    text-align: center;
    color: white;
    line-height: 100px;
    margin: 0;
}
section{
    display: flex;
}
article{
    padding: 10px;
    margin: 10px;
    background: white;
}
```

Slide Note

We have three different HTML elements, all of which have an outer display type of block. The first is a paragraph, which has a border added in CSS. The browser renders this as a block box, so the paragraph begins on a new line and expands to the full width available to it.

The second is a list laid out using the display: flex. This establishes a flex layout for the items inside the container. However, the list itself is a block box and — like the paragraph — expands to the full container width and breaks onto a new line.

Below this, we have a block-level paragraph inside which are two elements. These elements would generally be inline however one of the elements has a class of blocks, and we have set it to display: block.

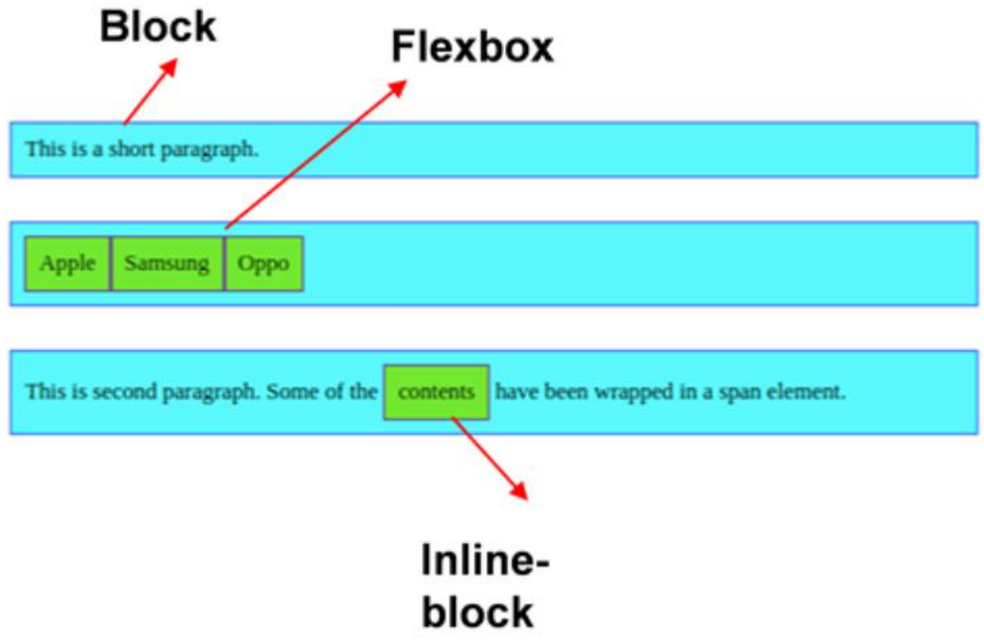
HTML Code:

```
<p>This is a short paragraph.</p>
<ul>
      <li>Apple</li>
      <li>Samsung</li>
      <li>Oppo</li>
</ul>
<p>
  This is second paragraph. Some of the
  <span class="block">contents</span> have been
  wrapped in a <span>span element</span>.
</p>
```

CSS Code:

```
p,ul {
    border: 2px solid blue;
    padding: .5em;
    background-color: aqua;
}
.block,li {
    border: 2px solid rebeccapurple;
    padding: .5em;
    background-color: lime;
}
ul {
    display: flex;
    list-style: none;
    background-color: aqua;
}
.block {
    display: inline-block;
    background-color: lime;
```

Examples of Different Display Types



Slide Note

Quick Check

Elements with a _____ z-index will appear above other elements.

a. higher

b. lower

c. identical

d. negative

Slide Note

Quick Check: Solution

Elements with a _____ z-index will appear above other elements.

a. **higher**

b. lower

c. identical

d. negative