CS472 WAP

Inline and Block

Page Layout

The Whole is Greater than the Sum of the Parts

Maharishi International University Fairfield, Iowa © 2020

Main Point Preview

- 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).
- An inline element does not start on a new line and only takes up as much width as necessary. These are the fundamental display types for almost all HTML elements. Understanding these fundamental types is critical to creating effective layouts.
- *Familiarity with fundamental levels of awareness is critical to successful action.*

Details about **`block`** boxes

- By default **`block`** elements take the entire width space of the page unless we specify.
- To align a **`block`** element at the **`center`** of a horizontal space you must set a **`width`** first, and **`margin: auto;`**
- **`text-align`** does not align **`block`** elements within the page.

Centering a block element: auto margins

```
<p> Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut
 la-bore et dolore magna aliqua.
</p>

p {
 border: 2px solid black;
 width: 33%;
 margin-left: auto;
 margin-right: auto;

}
```

- Set the width of a block-level element to prevent it from stretching out to the edges of its container.
  - Set left and right margins to auto to horizontally center that element.
  - Remaining space will be split evenly between the two margins.
- to center inline elements within a block element, use text-align: center;

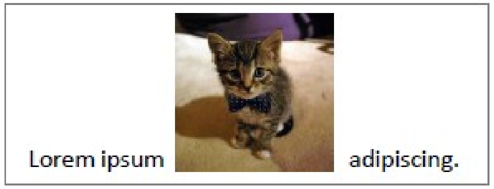
Details about `inline` boxes

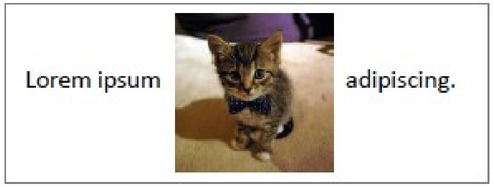
- size properties (width, height, min-width, etc.) are ignored for inline
- margin-top and margin-bottom are ignored,
    - but margin-left and margin-right are not
    - Padding top and bottom ignored
- each inline box's vertical-align property aligns it vertically within its block box (see next slide)
- **text-align** describes how inline content is aligned in its parent block element.
    - does not control the alignment of block elements, only their inline content
    - See lesson3_examples/textalign.html

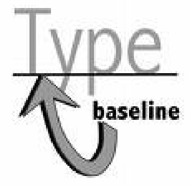
The `vertical-align` property

- Specifies where an **inline element** should be aligned vertically, with respect to other content on the same line within its block element's box
- Can be top, middle, bottom, baseline (default), sub, super, text-top, text-bottom, or a length value or %. baseline means aligned with bottom of non-hanging letters

```
img {
 vertical-align: baseline;
}
img {
 vertical-align: middle;
}
```







- See common error example: lesson3_examples/verticalaligin.html
- image is vertically aligned to the baseline of the paragraph, which isn't the same as the bottom

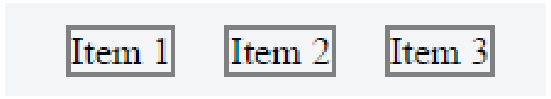
(review) ***display*** *property*: block vs inline

- The default value of display **property** for most elements is usually **block** or **inline**.
- **Block:** **div** is standard block-level element. A block-level element starts on a new line and stretches out to the left and right as far as it can. Other common block-level elements are **p** and **form**, and new in HTML5 are **header**, **footer**, **section**, and more.
- **Inline:** **span** is standard inline element. An inline element can wrap some text inside a paragraph **<span>** like this **</span>** without disrupting the flow of that paragraph. The **a** element is the most common inline element, since you use them for links.
- **None**: Another common display value is **none**. Some specialized elements such as **script** use this as their default. It is commonly used with JavaScript to hide and show elements without really deleting and recreating them.
- **inline-block** elements are like **inline** elements but they can have a width and height.
- Flex and grid: new options for responsive layouts

Displaying `block` elements as `inline`

- Lists and other **block** elements can be displayed **inline**, flow left-to-right on same line.
  Note: Width will be determined by content (while block elements are 100% of page width)

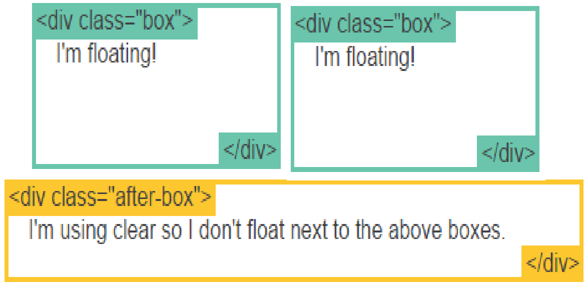
```
<ul id="topmenu">
 <li>Item 1</li>
 <li>Item 2</li>
 <li>Item 3</li>
 </ul>

#topmenu li {
 display: inline;
 border: 2px solid gray;
 margin-right: 1em;
 list-style-type: none;
}
```



Item 1 Item 2 Item 3

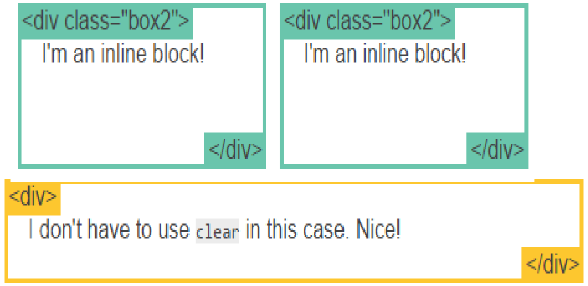
Boxes (Photo Gallery)

The Hard Way (using `float`)



```
.box {
 float: left;
 width: 200px;
 height: 100px;
 margin: 1em;
}
.after-box {
 clear: left;
}
```

The Easy Way (using `inline-block`)



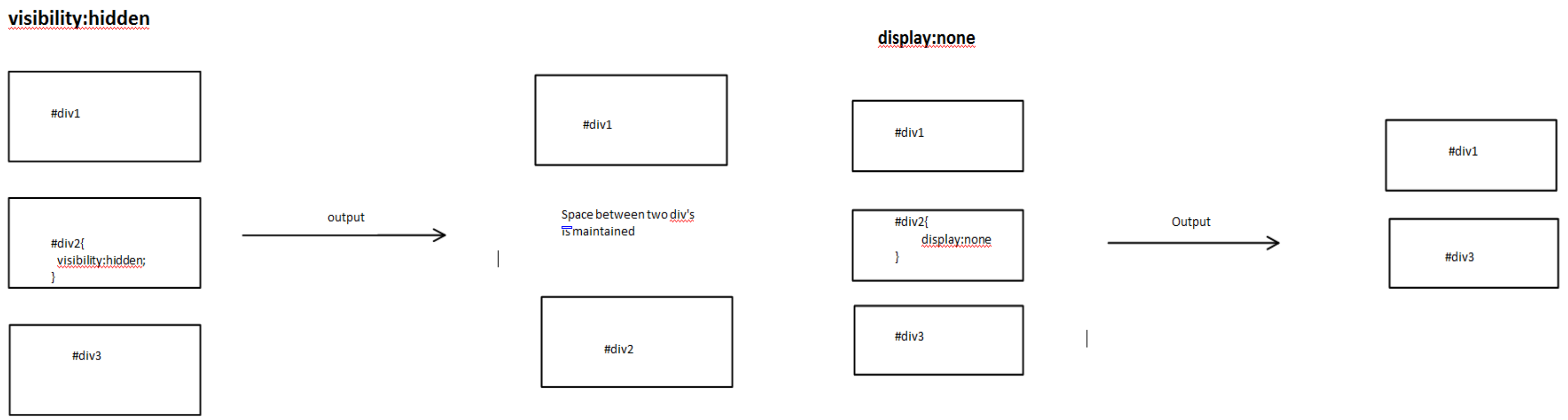
```
.box2 {
 display: inline-block;
 width: 200px;
 height: 100px;
 margin: 1em;
}
```

display: **inline-block** , the top and bottom margins/paddings are respected, but with display: **inline** they are not

nav links another possible use case for inline-block display

Visibility vs Display

- Setting **`display`** to **`none`** will render the page as though the element does not exist.
- **`visibility`**: **`hidden`**; will hide the element, but the element will still take up the space it would if it was fully visible.

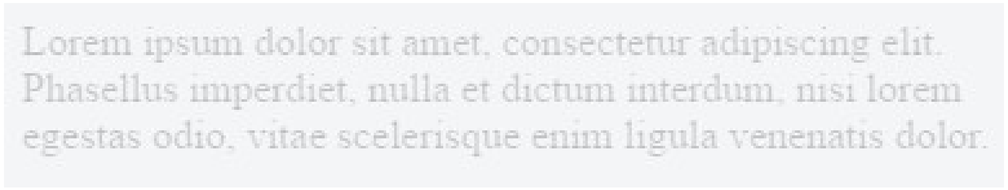


display: none vs visibility: hidden

Opacity

- The **`opacity`** property sets the opacity level for an element.
- Value ranges from 1.0 (opaque) to 0.0 (transparent)
- Example usage with :hover

```
div {
 opacity: 0.5;
}
```



Lorem ipsum dolor sit amet, consectetur adipiscing elit. Phasellus imperdiet, nulla et dictum interdum, nisi lorem egestas odio, vitae scelerisque enim ligula venenatis dolor.

Main Point

- 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).
- An inline element does not start on a new line and only takes up as much width as necessary. These are the fundamental display types for almost all HTML elements. Understanding these fundamental types is critical to creating effective layouts.

- *Familiarity with fundamental levels of awareness is critical to successful action.*

MAHARISHI INTERNATIONAL UNIVERSITY

HIGHER EDUCATION FOR HIGHER CONSCIOUSNESS