**BLOCK AND INLINE / BOX MODEL / POSITIONING / FLEXBOX**

**Block and Inline**

**- Elements with different display property values have subtly different box models.**

**Block vs Inline vs Inline-Block**

**- Block elements, whose default style is display: block, appear on the page stacked atop each other, with each element starting a new line.**

**- Inline elements, whose default style is display: inline, do not start on a new line. Padding and margin do not work well on inline elements.**

**- Inline-block elements, whose style is display: inline-block, behave like inline elements, but they have block-style padding and margin.**

**The Box Model**

**- Everything on a webpage, at its simplest, is a rectangular box.**

**Display: Block and Inline Boxes**

**Block**

**- Elements with a display type of block will:**

**- Break onto a new line**

**- Padding, margin, and border cause other elements to be pushed away**

**- Width and height are respected**

**- Box fills the entire space available in its container**

**- <h1> and <p> are examples.**

**Inline**

**- Elements with an outer display type of inline will:**

**- Not break onto a new line**

**- Width and height properties do not apply**

**- Vertical padding, margins, and borders will apply but will not cause other inline boxes to move away**

**- Horiztonal padding, margins, and borders will apply and will cause other inline boxes to move away**

**- <a>, <span>, and <em> are examples.**

**Inline-Block**

**- Elements with display: inline-block will:**

**- Not break onto a new line**

**- Width and height properties are respected**

**- Padding, margin, and border cause other elements to be pushed away**

**Breaking Down the Box Model**

**<img src='https://developer.mozilla.org/en-US/docs/Learn/CSS/Building\_blocks/The\_box\_model/box-model.png'>**

**- There are four main components of a box:**

**- content: area where content is displayed that can be sized using width and height**

**- padding: increases the space between the border of a box and the content inside of it**

**- border: adds space between the padding and the margin**

**- margin: increases the space between a box and any others that sit next to it**

**- When two elements are next to each other, the largest margin is used because they collapse into each other.**

**- If both elements had a margin of 60px, the total margin between them is only 60px, not 120px.**

**- If the box-sizing property is set to border-box, the total height and width of the entire box will be whatever height and width are set to.**

**Margin Properties**

**Property Description Example**

**--------------------------------------------------------------------------------------------------------------------------------------------------------------------------**

**margin-top sets the top margin margin-top: 5px;**

**margin-right sets the right margin margin-right: 50%;**

**margin-bottom sets the bottom margin margin-bottom: 3em;**

**margin-left sets the left margin margin-left: 1rem;**

**margin set top/bottom and left/right<br>set all margins<br>set top, right, left, and bottom margin: 2em 4em;<br>margin: 5px;<br>margin: 1em 2em 3em 4em;**

**- To horiztonally center an element, set its width and then use the following margin:**

**- margin: 0 auto;**

**Padding Properties**

**Property Description Example**

**--------------------------------------------------------------------------------------------------------------------------------------------------------------------------**

**padding-top sets the top padding padding-top: 5px;**

**padding-right sets the right padding padding-right: 50%;**

**padding-bottom sets the bottom padding padding-bottom: 3em;**

**padding-left sets the left padding padding-left: 1rem;**

**padding set top/bottom and left/right<br>set all paddings<br>set top, right, left, and bottom padding: 2em 4em;<br>padding: 5px;<br>padding: 1em 2em 3em 4em;**

**Border Properties**

**Property Description Example**

**-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------**

**border-width sets the width of all of the borders<br>sets the width of the top, right, bottom, and left borders border-width: 5px;<br>border-width: 10px 5px 10px 5px;**

**border-color color the border border-color: 000000;**

**border-style sets the style of the border (none; dotted; dashed; solid; double) border-style: dotted;**

**border-&lt;side&gt; sets the width, color, and style of the top, right, bottom, or left border border-left: 5px red double;**

**border sets the width, color, and style of all the borders border: 5px red double;**

**border radius rounds the corners of an element border-radius: 5px;**

Positioning Content

Positioning with Floats

- The float property is versatile and can be used in a number of ways.

- It removes an element from the normal flow of a page and positions it to the left or right of its parent element.

- All other elements then flow around the floated element.

- The two most used values are left and right.

- It is best utilized with margin and width to better arrange columns.

Clearing & Containing Floats

- Because floating is not meant for laying out a page, it can sometimes negatively impact the margin and padding of nearby elements.

- To prevent this from occurring, the clear value can be used.

- Its values are left, right, or both, and it signifies which floats to clear.

- An element after floated elements can benefit from having clear: both; to return the page to its normal flow.

Clearfix

- An alternative method is to contain floats by storing them in a parent element that acts as a container, leaving the flow completely normal outside of it.

- The CSS for that parent element is known as clearfix, and it is shown below:

.group::before,

.group::after {

content: "";

display: table;

}

.group::after {

clear: both;

}

.group {

clear: both;

\*zoom: 1;

}

- The pseudo-elements ::before and ::after dynamically generate elements above and below the element and are displayed as table-level elements.

- The clear: both; in the second and third rulesets clears the floats and returns the page back to normal after the container, and it contains some trickery for old browsers.

Position with Inline-Block

- The display: inline-block property and value can be used to layout elements next to each other.

- One concern that must be addressed is that between all inline-block elements, there is a single space between them that needs to be removed for optimal formatting.

- In the HTML document, the space between the </section> of one element and the <section> of the next element can be removed.

<section>

...

</section><section>

...

</section>

- Alternatively, for formatting purposes, a comment can be inserted:

<section>

...

</section><!--

--><section>

...

</section>

Reusable Layouts

- Reusable layouts can be created to establish a grid-like effect on the page with equalivalent columns.

- For instance, to create a layout that enables three columns of equal width or two columns with the width split between them:

.col-1-3 {

width: 33.33%;

}

.col-2-3 {

width: 66.66%;

}

.col-1-3,

.col-2-3 {

display: inline-block;

vertical-align: top;

}

- col-1-3 can be used to create three equal columns.

Uniquely Position Elements

- To uniquely position an element, the position property and box offset properties--top, right, bottom, and left--can be used.

- Every element has a default position of static.

Relative Positioning

- The relative value for the position property allows an element to appear within the normal flow of a page, and it allows it to be modified with the box offset properties.

- The element is moved relative to its initial position.

.offset {

left: 20px;

position: relative;

top: 20px;

}

- For instance, the above code would move the box twenty pixels from the left and top.

- Instead of moving an element beneath it down, it simply overlaps it.

Absolute Positioning

- The absolute value for the position property means that the element does not appear within the normal flow of the document.

- It is positioned in relation to its closest relatively positioned parent element.

- The <body> is the default.

- In the below example, the <div> would be positioned twenty pixels to the left and down twenty pixels from the top right of the <section>.

**FLEXBOX**

[**https://css-tricks.com/snippets/css/a-guide-to-flexbox/**](https://css-tricks.com/snippets/css/a-guide-to-flexbox/)

**Flex Containers and Flex Items**

**- Flexbox is a toolbox of properties that can be used to position elements.**

**- Some properties belong to the flex container and others to the flex items.**

**- A flex container contains the display: flex property and value.**

**- A flex item has the flex property.**

**- Flex items can also be flex containers:**

**Flex Properties**

**- flex is shorthand for three properties that can be set on a flex item.**

**- flex: <flex-grow> <flex-shrink> <flex-basis>**

**- When flex: 1 is set, it is interpreted as flex: 1 1 0.**

**- They affect how flex items size themselves.**

**flex-grow**

**- flex-grow is used as the flex item's "growth factor."**

**- If the value for all the items is 1, they will all be the same.**

**- If one of the values is 2, it will be 2x larger than the other values.**

**flex-shrink**

**- flex-shrink sets the flex item's "shrink factor."**

**- If the value is 1, all flex items will shrink equally.**

**- If an item is 0, it will not shrink.**

**- Higher numbers mean that an item will shrink at a higher rate than normal.**

**flex-basis**

**- flex-basissets the initial size of a flex item.**

**- Any growing or shrinking starts from this baseline.**

**- Using auto tells the items to check for a width declaration and the item will be set to that value.**

**- Using 0% sets the initial value to 0, and all items will proportionately grow to fill in the space.**

**Flex Axes**

**- There are two axes for flex containers:**

**- The Main Axis**

**- The Cross Axis**

**- The direction of the main axisis set by the flex-direction property, which can either be row (left-to-right), column (top-to-bottom), row-reverse (right-to-left) or column-reverse (bottom-to-top). The cross axis is perpendicular in each respective case.**

**- When it is set to row, flex-basis refers to width, but it refers to height when set to column.**

**- It is worth noting that block-level elements default to the full width of their parent, and they default to the height of their content, and where there is no content, the height is 0.**

**- When flex-direction is set to column, the height of the container needs to be set or else the flex shorthand property cannot be used because the basis for the flex-items will be a height of 0, which fills up the container's default of 0.**

**- Alternatively, the height of the flex item can be set and the flex-basis can be set to auto.**

**Alignment**

**- flex: 1 makes each flex item grow to fill available space.**

**- justify-content aligns items across the main axis.**

**- space-between- equal amounts of space between the items; no edge**

**- space-around - equal space on both sides of all items; combines middle**

**- space-evenly - equal space on sides and middle**

**- flex-start - everything to starting side**

**- flex-end - everything to ending side**

**- center - center items**

**- align-items aligns items along the cross axis.**

**- stretch - items stretch to the height of the flex container**

**- flex-start - items align at the start**

**- flex-end - items align at the end**

**- center - items center**

**- gap adds a specified space between flex items**

**- it is similar to adding a margin to the items**

**- align-self aligns a single flex item**

**- same values as align-items plus auto**

**flex-wrap**

**- flex-wrap has a default value of nowrap.**

**- Setting it to wrap means that items, given a width, will be pushed to the next line if they are too large for their container instead of autosizing.**

**flex-flow**

**- flex-flow is a combination of flex-direction and flex-wrap.**

**- flex-flow: <flex-direction> <flex-wrap>**

**Use Cases**

**- A common pattern for navigation is to have a list of items displayed as a horizontal bar.**

**- Another common use is for form controls with inputs and buttons.**