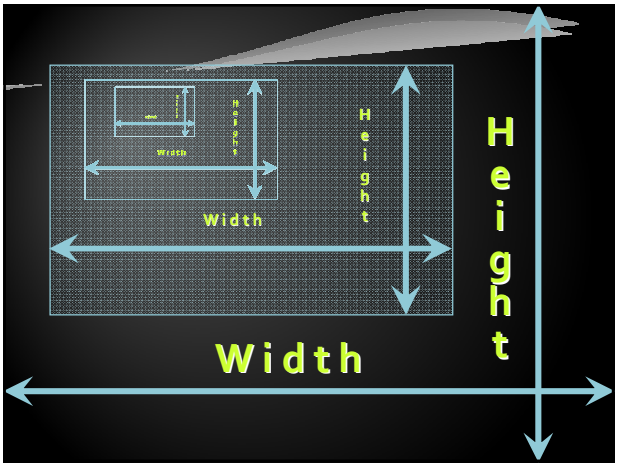
CSS Layout

Control the arrangement of the HTML elements

Table of Contents



Width

- `width` – defines numerical value for the width of element, e.g. `200px`
- `width` applies only for block elements
  - Their with is 100% by default
  - The width of inline elements is always the width of their content, by concept
- `min-width` - defines the minimal width
  - `min-width` overrides width if (`width<min-width`)
- `max-width` - defines the maximal width
  - `max-width` overrides width if (`width>max-width`)

Width Values

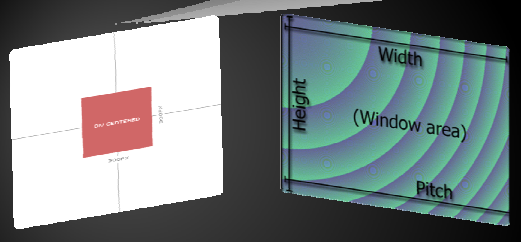
- The values of the width property are numerical:
  - Pixels (px)
  - Centimeters (cm)
  - Or percentages
    - A percent of the available width

Width

Live Demo

Height

- `height` – defines numerical value for the height of element, e.g. `100px`
- `height` applies only on block elements
  - The `height` of inline elements is always the height of their content
- `min-height` - defines the minimal height
  - `min-height` overrides height
- `max-height` - defines the maximal height
  - `max-height` overrides height



Height

Live Demo


Overflow

Overflow

- `overflow`: defines the behavior of element when content needs more space than the available
- `overflow` values:
  - `visible` (default) – content spills out of the element
  - `auto` - show scrollbars if needed
  - `scroll` – always show scrollbars
  - `hidden` – any content that cannot fit is clipped

Overflow

Live Demo


Display

Display

- `display`: controls the display of the element and the way it is rendered and if breaks should be placed before and after the element
  - `inline`: no breaks are placed before and after (`<span>` is an inline element)
    - The element's height and width depend on the size of the content
  - `block`: breaks are placed before AND after the element (`<div>` is a block element)
    - Height and width may not depend on the size of the content

Display (2)

- `display`: controls the display of the element and the way it is rendered and if breaks should be placed before and after the element
  - `none`: element is hidden and its dimensions are not used to calculate the surrounding elements rendering (differs from `visibility: hidden`!)
  - `inline-block`: : no breaks are placed before and after (like inline)
    - Height and width can be applied (like block)

Display (3)

- `display`: controls the display of the element and the way it is rendered and if breaks should be placed before and after the element
  - `table`, `table-row`, `table-cell` : the elements are arranged in a table-like layout

Display

Live Demo

Visibility

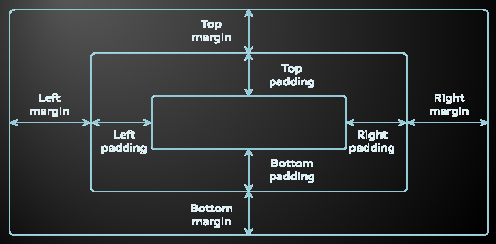
Visibility

- `visibility`
  - Determines whether the element is visible
  - `hidden`: element is not rendered, but still occupies place on the page (similar to `opacity:0`)
  - `visible`: element is rendered normally

Visibility

Live Demo

Margins and Paddings



Margin and Padding

- margin and padding define the spacing around the element
  - Numerical value, e.g. 10px or -5px
  - Can be defined for each of the four sides separately - margin-top, padding-left, ...
  - margin is the spacing outside of the border
  - padding is the spacing between the border and the content
  - What are collapsing margins?

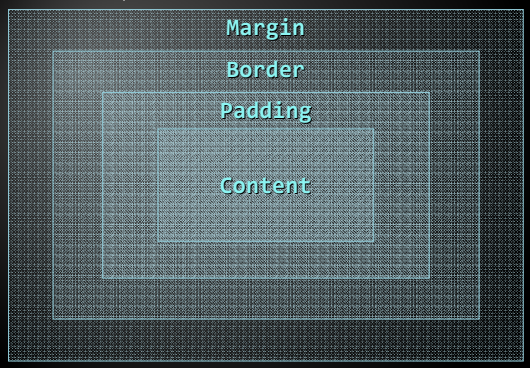
Margin and Padding: Short Rules

- margin: 5px;
  - Sets all four sides to have margin of 5 px;
- margin: 10px 20px;
  - top and bottom to 10px, left and right to 20px;
- margin: 5px 3px 8px;
  - top 5px, left/right 3px, bottom 8px
- margin: 1px 3px 5px 7px;
  - top, right, bottom, left (clockwise from top)
- Same for padding

Margins and Paddings

Live Demo

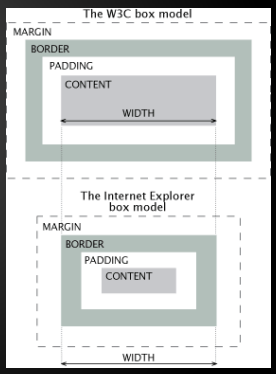
The Box Model



IE Quirks Mode

- When using quirks mode (pages with no DOCTYPE or with a HTML 4 Transitional DOCTYPE)

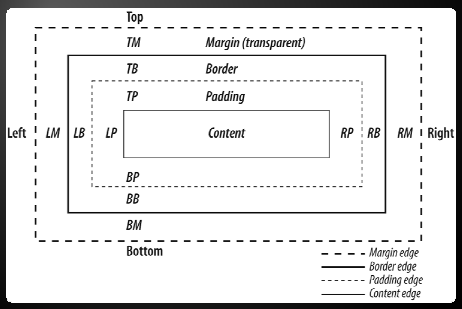
Internet Explorer violates the box model standard!



IE Quirks Mode

Live Demo

Box Model



CSS3 box-sizing

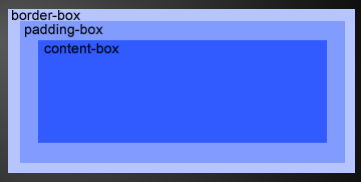
- Determine whether you want an element to render it's borders and padding within its specified width, or outside of it.
- Possible values:
  - `box-sizing: content-box` (default)
    box width: 288 pixels + 10 pixels padding and 1 pixel border on each side = 300 pixels
  - `box-sizing: border-box`
    box width: 300 pixels, including padding and borders

CSS3 box-sizing (Example)

- Example: Box with total width of 300 px (including paddings and borders)

```
width: 300px;
border: 1px solid black;
padding: 5px;

/* Firefox */
-moz-box-sizing: border-box;
/* WebKit */
-webkit-box-sizing: border-box;
/* Opera 9.5+, Google Chrome */
box-sizing: border-box;
```

Box Model

Live Demo

Positioning

Positioning

- `position`: defines the positioning of the element in the page content flow
- The value is one of:
  - `static` (default)
  - `relative` – relative position according to where the element would appear with static position
  - `absolute` – position according to the innermost positioned parent element
  - `fixed` – same as absolute, but ignores page scrolling

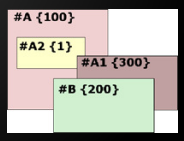
Positioning (2)

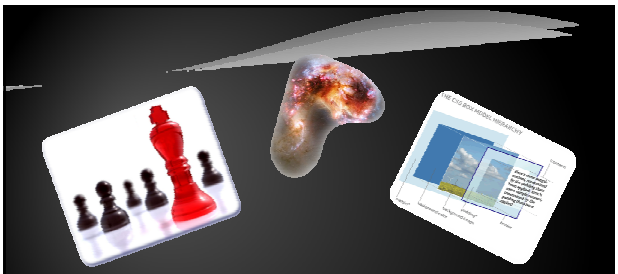
- Margin VS relative positioning
- Fixed and absolutely positioned elements do not influence the page normal flow and usually stay on top of other elements
  - Their position and size is ignored when calculating the size of parent element or position of surrounding elements
  - Overlaid according to their z-index
  - Inline fixed or absolutely positioned elements can apply height like block-level elements

Positioning (3)

- `top`, `left`, `bottom`, `right`: specifies offset of absolute/fixed/relative positioned element as numerical values
- `z-index` : specifies the stack level of positioned elements
  - Understanding stacking context

Each positioned element creates a stacking context.
Elements in different stacking contexts are overlapped according to the stacking order of their containers. For example, there is no way for #A1 and #A2 (children of #A) to be placed over #B without increasing the z-index of #A.



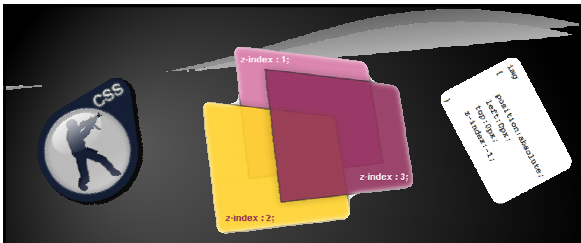


Positioning

Live Demo

Inline element positioning

- `vertical-align`: sets the vertical-alignment of an inline element, according to the line height
  - Values: `baseline`, `sub`, `super`, `top`, `text-top`, `middle`, `bottom`, `text-bottom` or numeric
  - Also used for content of table cells (which apply `middle` alignment by default)



Alignment and Z-Index

Live Demo

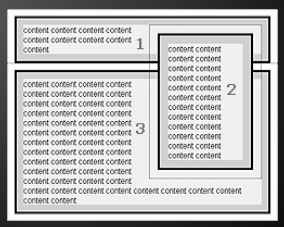
Floating

Float

- `float`: the element "floats" to one side
  - `left`: places the element on the left and following content on the right
  - `right`: places the element on the right and following content on the left
  - floated elements should come before the content that will wrap around them in the code
  - margins of floated elements do not collapse
  - floated inline elements can apply height

Float (2)

- How floated elements are positioned



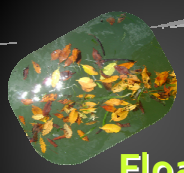
Clear

- `clear`
  - Sets the sides of the element where other floating elements are NOT allowed
  - Used to "drop" elements below floated ones or expand a container, which contains only floated children
  - Possible values: `left`, `right`, `both`
- Clearing floats
  - Clear using pseudo-class :after
  - Additional element (`<div>`) with a clear style
    - Deprecated - semantically unused div

Clear (2)

- Clearing floats (continued)
  - `:after { content: ""; display: block; clear: both; height: 0; }`
  - Triggering hasLayout in IE expands a container of floated elements
    - `zoom: 1`

```
.clearfix {
   zoom:1;}
.clearfix: after {
   content: "";
   display: block;
   clear: both;
   height: 0; }
```

Floating Elements

Live Demo