Build beautiful interfaces

Basic Sizing

- **Width** and **Height**
- selector {width:100px;height:100px}
- Like many other CSS rules, they do not always influence the object behavior

Basic Sizing

- Size can be measured Relatively or Absolutely
- Px for pixles
- Rem - the default as user have set it
- Em – the current default (if modified by parents)

See here

Basic Sizing – Min/Max

- **Min-width / min-height**
- **Max-width / max-height**

Play with with [min/max] width/height here

Colors

- Colors styling appears in many places:
  - color:red;
  - background-color:yellow;
  - border:1px red solid;
- And more ...

Colors

- Colors can be set by RGB values: Red, Green and Blue
- Using the rgb function in CSS: {color:**rgb(22,57,99);**}
- The values goes from 0 to 255
- A shorter way to write a color is using Hex numbers
- {color:**#7a66ff**;}
- Or even shorter (with less colors variety)
- {color:**#fbg**;}

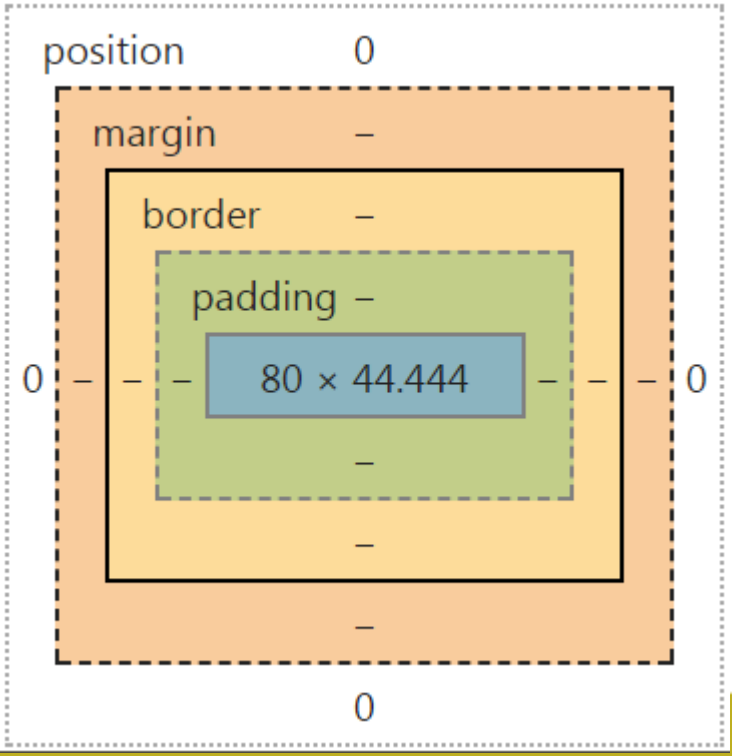
Colors

- Colors can also be transparent and semi transparent using the alpha value
- {color: **rgba(67,78,89,0.5);** }

Box Model

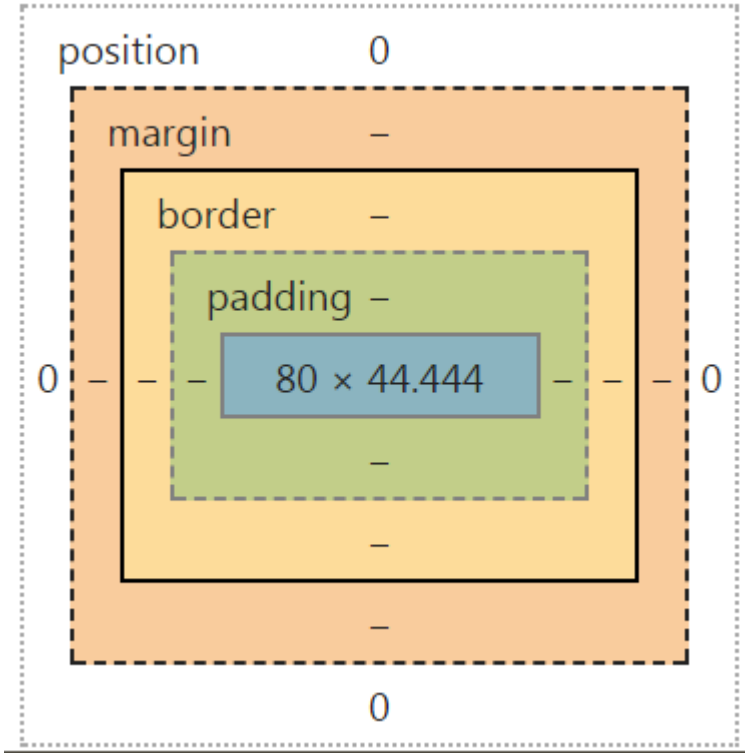
- Some elements in HTML
  - like ,
  - Unlike <a>

  are treated as boxes.
- The browser uses the box model to display them:



Box Model

- The box model is consistent of:
- The content size
- The padding
- The border
- And the margin



Box Model

- We can set each padding / border / margin values using:
- **{padding: 5px}** – all padding will be set to 5px
- **{margin-left:10px}**
- **{border-width: 5px 8px}**
- **{margin: 1px 2px 3px 4px}**
- And any combination of them: -top -right -bottom -left

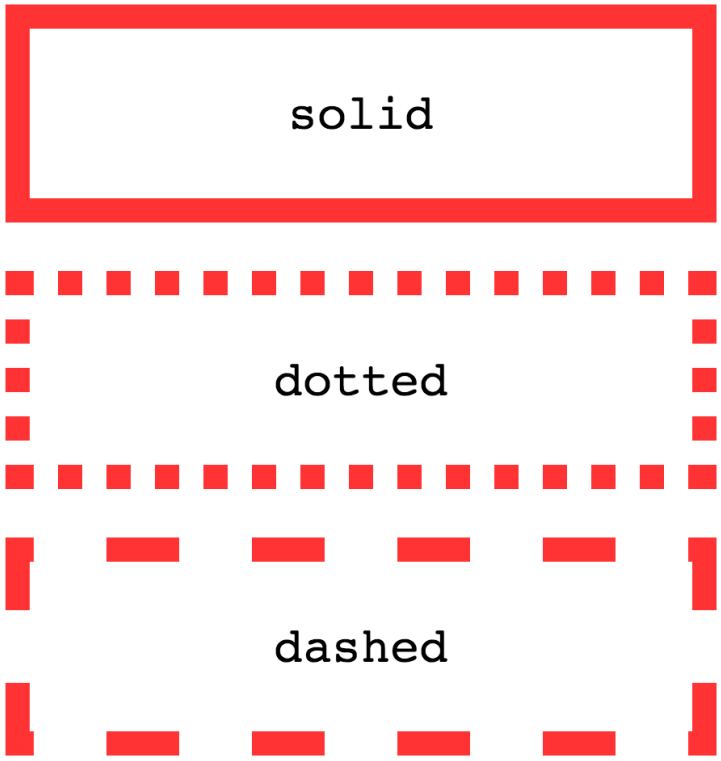
Box Model

- On it's default behavior box model:
- Top and bottom margin are collapsed (merged together)
- And the **box-sizing** is set to **content-box**
- TIP: It is common to change the box-sizing to **border-box** when the your size definition should include padding and borders.

Box Model - Borders

- A border have also a visual style
- **Border-type: none / dotted / dashed / solid / double / groove / ridge / inset / outset / hidden**
- **Border-width**: (size)
- **Border-color**: (color / transparent)
- Short hand: **{border-bottom: 1px solid black;}**

Box Model - Borders



Inline

- When the browser display the elements it does it from top left to right.
- That's an **inline** flow
- Those are all **inline** elements:
  span, a, img, button…

Box Model

- Setting different box model style is done using the **display** style
- ***None*** – the element will not be displayed and will not occupy any space at the view
- ***Block*** – an element that takes the maximum width it can, and can have height / width styles
- ***Inline*** – an element with in the line of the rendering
- ***Inline-block*** – part of the rendering but can also get width and height
- And few more that we will see later on

Background Images

- We can set each element it's background image
- **{background-image: url(*image path relative to the css*);}**
- **Example:**
- **{background-image: url(../img/logo.png);}**

Background Images

- If the image is smaller than the object size, the image will repeat itself
- Controlling image repetition can be made using:
- **background-repeat:**
  - **Repeat** (default)
  - **No-repeat**
  - **Repeat-x or repeat-y**

Background Images

- When scrolling – the background will scroll with the content
- We can change it using the:
- **{background-attachment:fixed;}**
- By default it is:
- **{background-attachment:scroll;}**

Background Images

- We can set the background-position directly
- **{Background-position: 90px 90px}**
- Or even in words or percentage
- **{Background-position: center top}**
- **{Background-position: 20% 50%}**

Background Images

- There is a shorthand syntax to set all background styling:

```
selector {
    background: url(..) no-repeat yellow fixed top center;
}
```

Background Images

- Using sprites (or tiles) it is possible to load less images per page and display part of the image using background positioning
- See here and here

Table Styling

- The old <table border=”1”>
- Table border – is only around the table itself
- Bordering <td> - is done using:
- {border-collapse:collapse;}
- {border-spacing: 5px;} instead of cellspacing=0
- {caption-side: bottom}

List Items

- We have <ul> or
- List items can be used in many cases. They are even used for creating menus
- The <li> have display type list-item, that can be changed

List Items

- List items have property for:
- **List-style-type: disc / circle / square / decimal / lower-roman / lower-alpha / lower-latin / upper - ..etc / decimal-leading-zero / lower-greek**
- We can even set our own: **list-style-image: url(..)**
- **List-style-position: inside / outside**
- Shorthand: **{list-style: disc inside}**

Pseudo Classes

- Pseudo Classes and Pseudo Elements are elements that does not really exist within the page
- The are logical elements or created on-the-fly during the web page run-time

Pseudo Classes - Example

- We can set each link state a different CSS rule
- **a:link** {} - element which is part of a link
- **a:visited** {} - a link that we already viewed
- **a:active** {} - the current link we are “standing” on
- **a:hover** {} - when the mouse is hovering above it – on IE hover is supported only for links
- **a:focus** {} - focused element

Pseudo Classes

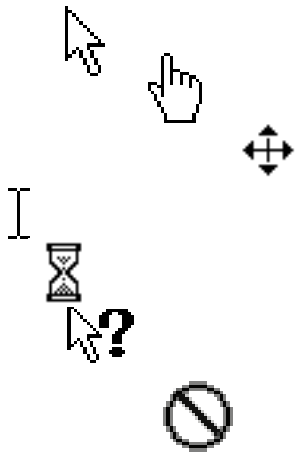
- More examples:
- **:first-child**
- **:first-letter**
- **:first-line**
- **:nth-child(n)**

- **:checked**

Cursor changing

- A few cursor samples: (And there are more ...)

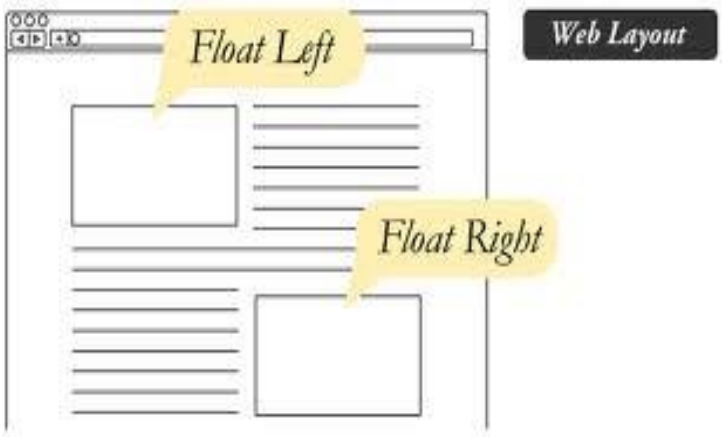
default
pointer
move
text
wait
help
not-allowed



- We can also set our own Cursor Image
- **A {cursor: url(imagename.gif);}**

Floating

- When we let an element "**float**" we take it out from the usual rendering location
- Float can be set: **left**, **right** or **none** (default)
- Allowing the text to wrap around the floating element
- Multiple floating elements are stacked one after the other



Rounded Corners

- Rounded Corners where added in CSS3 and now supported by all major browsers
- **{border-radius:10px;}** - border-radius-top-left ..

Floating Clearing

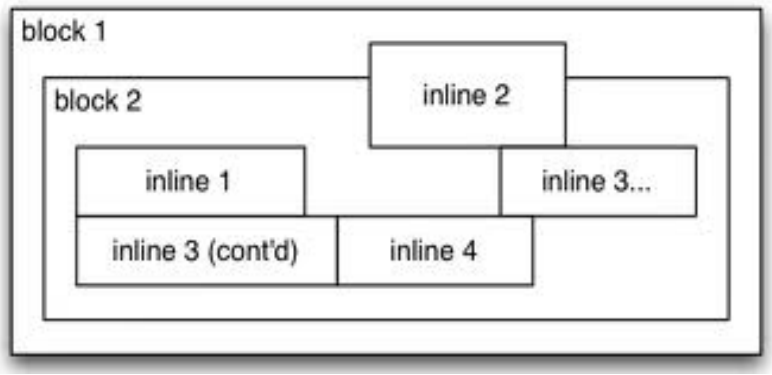
- Every floating element must have a width style (or the behavior is unknown and different between browsers)
- We can set element to appear solo on the line and preventing other elements from floating around it
- We use the clear style
- {**clear: right / left / both or none (default)**}

Layers

- Layers allow us to set elements to appear on top of other elements with in a page
- We can enable layers using the position style
- **{position: static (default) / relative / fixed / absolute}**

Position - relative

- The relative position, positions the element relatively to the place the element should have been
- **#square2 {position:relative;left:20px;top-20px}**
- We use the left / right / top / bottom styles to position the element



Position - fixed

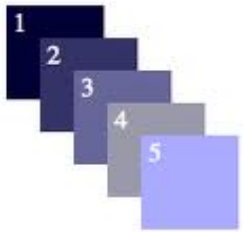
- The fixed position acts towards the browser window
- No matter the window size, and no matter scrolling
- **#feedback{position:fixed;bottom:0;right:0}** – will always be on the bottom right corner of the screen

Position - absolute

- The absolute position put the element relatively to the first parent element that is not in position static
- That's allow a position fixed like within other elements

Position - z-index

- Now when we have layers and elements can overlapped other elements, we need a way to determine which element will be in front
- We do that by setting the:
- **{z-index: value}**



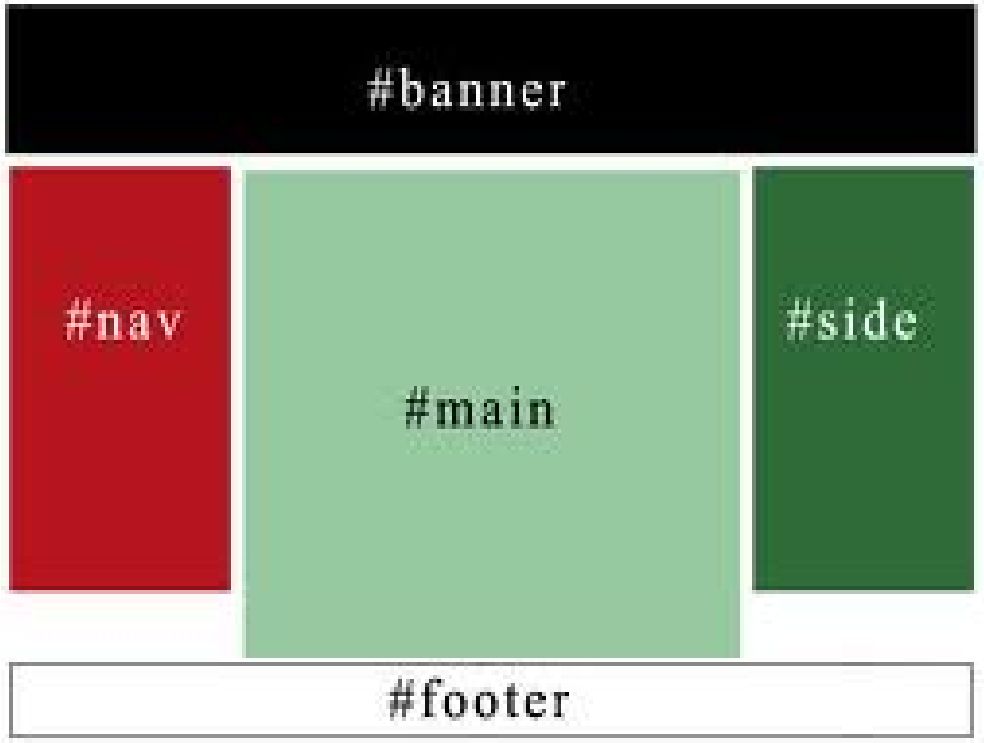
- The higher the value is, the top most the element will be.
- Auto or 0 is the default, means whom ever drawn last will be on top

Tables Layout

- Now days we usually don't use table to create layout
  - One of the reasons is that it is not accessible and negatively affect SEO
- Although it does simplify things a bit it is still have many cons
- CSS3 Solves many of the difficulties that forced us to use tables for layout

CSS Layout

Build this layout



Victorious!

CSS is Power

Now lets build something great