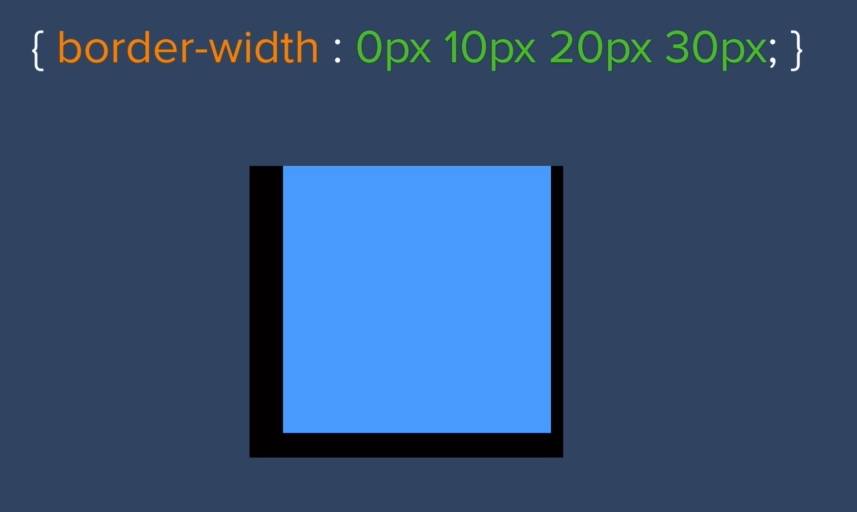
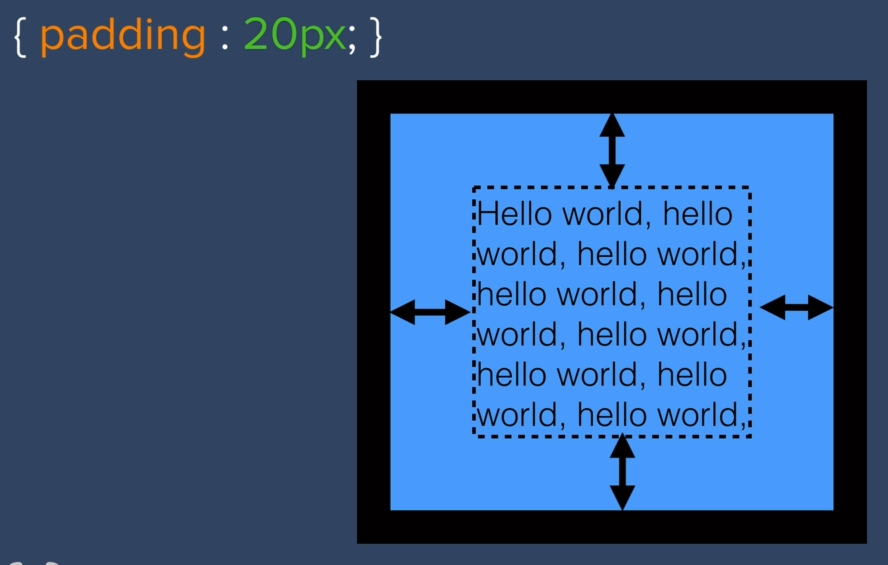
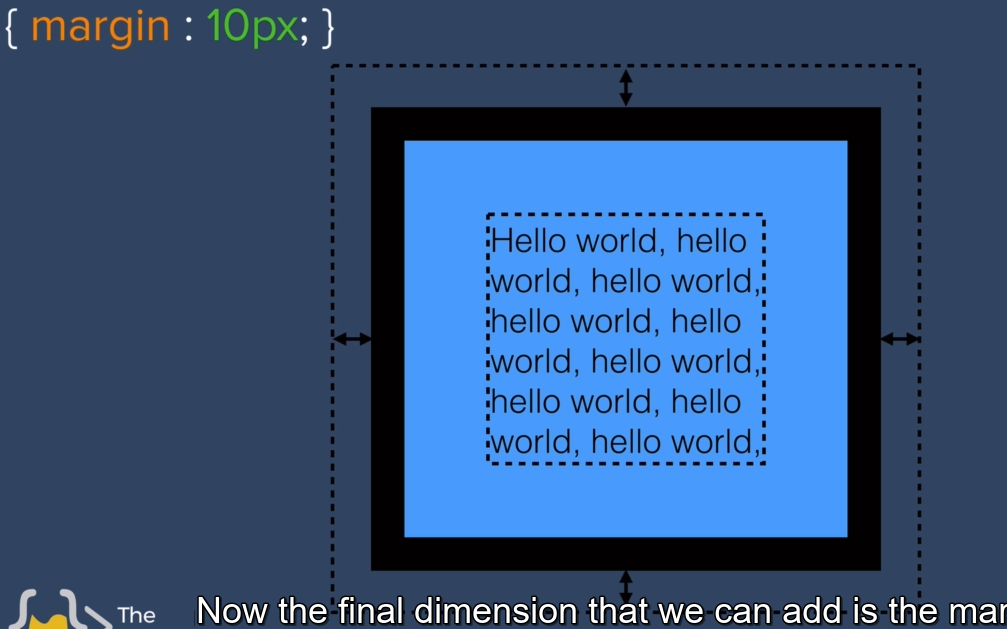
**Lecture 01**

**Lecture 02**

**Lecture 03**

* Div element can have height is you specify it using style, otherwise it’s height is zero
* It also has a height if it has content in it.

**Lecture 04 (CSS box model)**

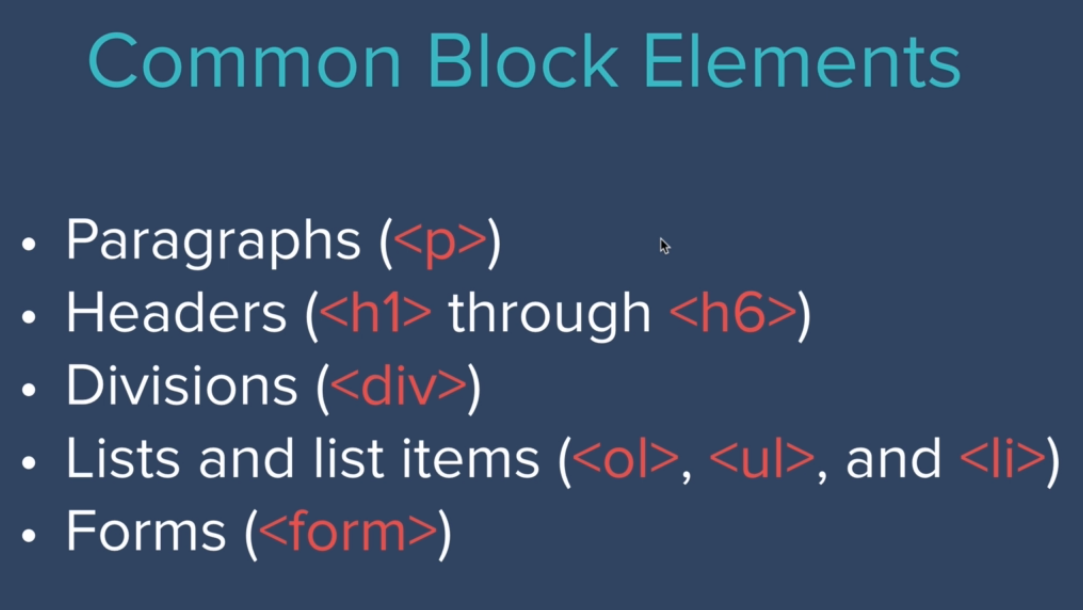
* Width/height can be specified in pixels/pecentages.
* Viewport is the current screen on which you are viewing the website
* A solid border has a width of 3 px and looks like this
* 
* You can specify border width using border width property
* The border is applied outside the div, so if your div has 100px width and a border of 10px on all sides then the combined width of your object will be 120px.
* You can specify the width of each side of the border using this property 
* Padding increases the size of your div 
* Even if you don’t have padding your div will always stretch itself to fit enough content inside it.
* You can also add the margin but the margin is not included in the size of the object it acts like a buffer around your object 
* This border, padding and margin is collectively known as ‘HTML box model’

**Lecture 05**

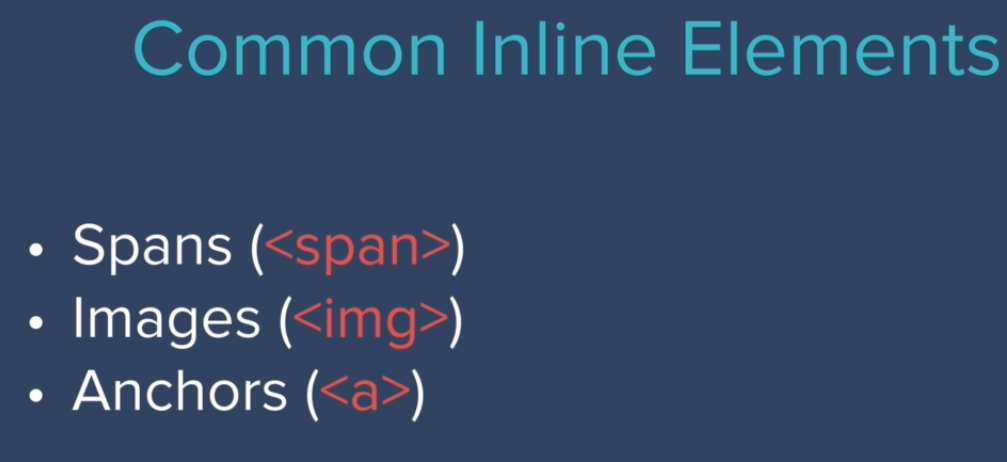
* Display property has 4 different values



* Block display elements take up whole width of the screen like H1 and p tags



* Span element is inline element and allows us to target by making inline elements and highlighting it.
* Inline elements only take as much space as they need to, in terms of both height and space



* You can’t change width of default inline elements like span, thus they are a bit restrictive
* However height and width of block elements can be changed, and they can also be converted into inline elements by changing their display property to inline, but then you lose the ability to change its width.
* There is also a type of display property called inline-block that allows you to get the best of both worlds i.e. you get the inline properties of inline display and you can change the height and width of the images too.
* The last display option is display:none which removes the element from the website like it never existed.
* There is another property called visibility which makes it the element hidden but keeps other elements position around it the way they were {visibility:hidden;}



**Lecture 06**

**Lecture 07 (Relative Positioning)**

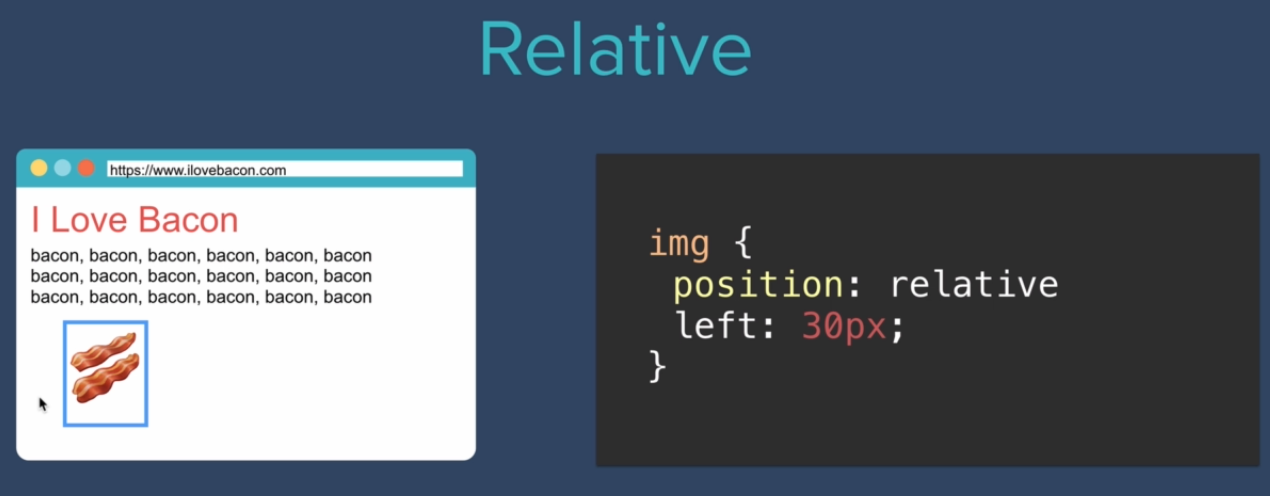
* For inline content determines the height and weight
* For block elements, width is max but height is determined by the content
* The order of elements comes from the HTML code file
* Children elements sit on parent elements, kindof think on the z-axis.



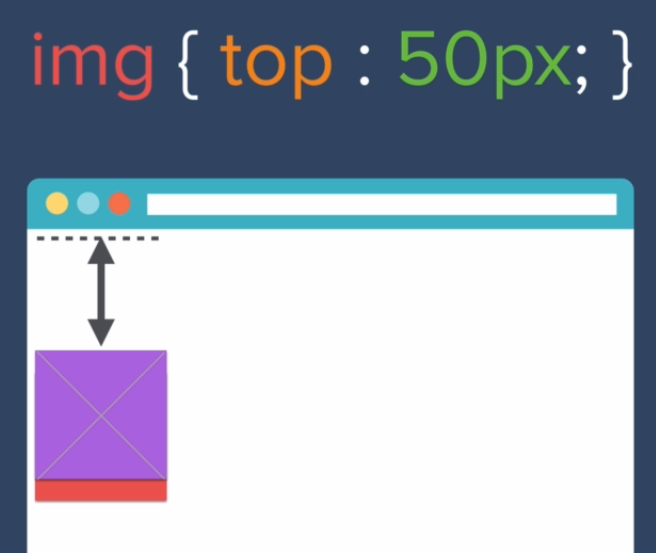
* Here span will be on top, h1 will be below it, and below it will be div
* These are the 3 basic rules of elements rendering using HTML
* Following are the different positions allowed in HTML



* Static is the default setting i.e. without any change in settings
* Relative positioning is used to change the position of an element wrt to where it would’ve been if its position wasn’t changed.



* You can use coordinate property to specify where you want to place your elements.
* Relative positioning doesn’t affect other elements around that element

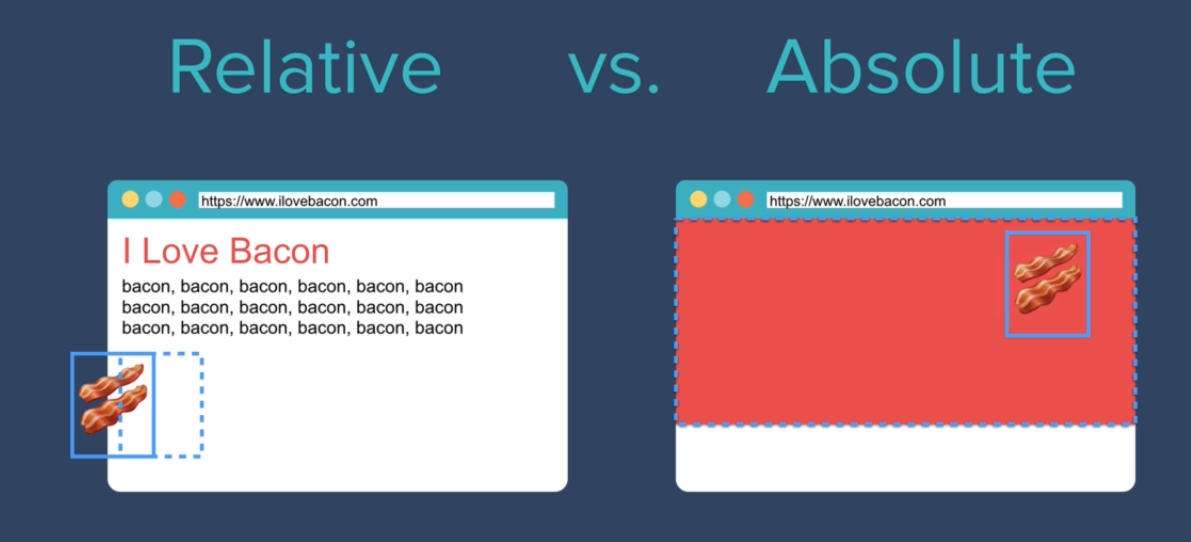


* You can think of it giving it as a top margin of 50px

**Lecture 08 (Absolute positioning)**



* In absolute positing the element is placed wrt to its parent for eg in the above example the image of bacon is pushed towards the right of 30px of the parent div.



* Relative positioning means adding a margin wrt where the element should’ve been had it not moved, absolute positioning means adding the margin wrt its parent element.
* Absolute positioning takes the element out of the natural flow of the document unlike relative positioning.
* Fixed positioning allows the element to stay stick to the webpage during scrolling, it doesn’t move along with other elements of the webpage.

**Lecture 09 (The dark art of centering elements with CSS**

* You can align text/elements in the center of the page by using text-align:center property but you have to use this property in the parent element.
* But this works only till you have only inline or block elements, if you change the width of a block element (Let’s say you changed the width of the element to 10%) then that element won’t be centered on the page anymore.
* If you encounter any of this scenario then you can use the margin property in the following manner to center the element. Margin: 0 auto 0 auto it translates to <top>,<right>,<bottom>,<left> start from the top and then go counter-clockwise
* In order to set absolute condition for an element, you should make one of its parent’s position relative otherwise it would position itself relative to the body.

**Lecture 10 (Font styling)**

* There are basically 2 types of fonts



* Monospace fonts are used for showing code since in such fonts each letter takes up the same amount of space.
* You can specify broadly the type of font in your website by using the “font-family” css property there are 3 major options you are ever gonna use that are serif, sans serif and monospace
* By default you have sans serif family
* By default, serif font is Times now roman and sans-serif is Arial in most websites.
* You can specify the specific font family using the same css property like this



* Here the browser first searches “verdana” font and renders the website if its found, but if the font does not exit on the users pc then the browser would render the website in sans serif font.
* There are some fonts which are considered “websafe” since there is a relatively large userbase that has them on their computers but no font is 100% web safe.
* In this case fallbacks are rather useful to make the website look almost the same.



* In this code the first choice, would be Helvetica Neue, after that browser will try to find Helvetica, after that arial will be tried and finally sans-serif will be tried.
* In this strategy we go from most specific to less specific.
* You can find fallbacks on [www.cssfontstack.com](http://www.cssfontstack.com)
* But to make sure that everyone sees the exact same font we can use something known as “font embedding” it looks something like this



* Here we have selected 3 fonts (from google fonts), you can select this embedding link at the top of the webpage, you can get this link from google fonts.
* Now if the user doesn’t have these fonts already on this computer, the browser would actually source it from fonts.google.com
* If your font has more than one styles i.e. simple/bold/italic you want to choose all of them in your embedding link otherwise when you choose a heading your browser will automatically use a bold font but if it doesn’t find a bold font it will try to make it bold by itself and it may appear differently on every user’s computer.
* Only include those fonts that you are using, because if you have included a font and you are not using it then you’d be increasing you network’s bandwidth and your website’s loadtime for no reason.

**Lecture 11**

**Lecture 12**

**Lecture 13 (CSS Sizing)**

* You can give your text specific size using “font size” property in CSS but then your font won’t be dynamic, to overcome this issue we can give our font size in terms of percentage.
* Now in CSS 100% font size means 16 pixels, suppose you want to give 90px font size to a block of text then the formula for that would be

90/16\*100=562.5 percent

* This is called providing a dynamic font size to the text on your website.
* Another way of specifying font-size is in “em” units.
* The conversion goes something like this

1em=16px=100%

* Thus 90px would be 9.625em
* One thing to keep in mind is that the “em” font size get inherited to child elements, for example if body has font size set to “2em” and the child element has font size set to “3em” then the final size of the element would be “6em”.
* But if we give our font size in pixels then it doesn’t get inherited.
* We tend to use dynamic font size to make our website better for visually impaired user and provide better accessibility.
* In CSS3 we have a new feature called “rem”, its full form is root em. It basically says that ignore the previous ems applied on this element and just apply the current em setting.
* When sizing text it is recommended to use “rem” because it is the least error prone option.
* The default font size in css is 16px or 1em or 100%.

**Lecture 14**

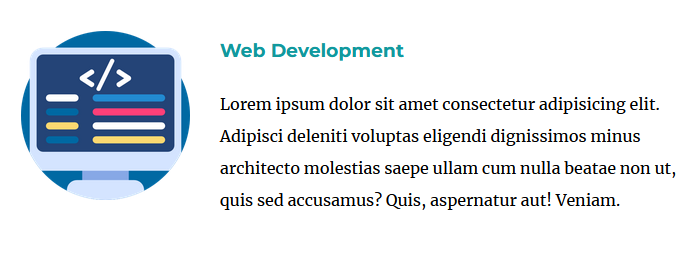
**Lecture 15**

**Lecture 16**

**Lecture 17**

**Lecture 18 (CSS float and clear)**

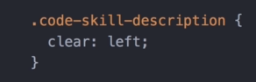
* Float allows you to wrap elements around one another, for example you have an image and you want to display its description by its side like this.



* In this scenario you will have to add the “float:left” property to the image to float it towards the left of the text, in the image above a right margin of 20px has also been added to give it an overall cleaner look.
* But let’s say that you only want the heading to wrap around the image and not the description in this case you will have to add the description in another class and use the “clear:left” property to clear anything that is on its left and allow it to appear separately like this.



* The css code for clearing the left side of the element would look something like this:



* Float is one of the most abused properties of css, therefore it is recommended to only use float for the purpose its intended i.e. wrapping text around images. Don’t use it for positioning purposes.
* For positioning use relative and absolute positioning etc. because they are more straightforward.

**Lecture 19**

**Lecture 20**