Topic/Title: CSS Static and Relative Positioning

Keywords/Questions:

3 HTML Display rules

Static Positioning

Relative positioning

postion:relative

```
img {position:relative;
    left:50px;
}
```

Coordinates

Notes:

HTML element Rules of Display:

1.Content is everything

-the size of the content determines the size of the box. For

-an inline element: both height and width are based on it.

-a block element: the height is based on it.

2. Order comes from your code.

For example, if we change the position of the image tag, the location of the

image on our browser will change.

Summary: The 3 HTML rules of display:

1. Content is everything.

2. Order comes from your code.

3. Children sit on parents.

Positioning: -Static -Relative -Absolute -Fixed

Static positioning: original positioning set by default

Relative positioning: positioning relative to the default margins.

Coordinates:-left -top -right -bottom

More Notes

HTML element Rules of Display continued:

3. Children Sit On Parents.

-the first parent here is div then h1 then span. So, the span lies over the h1 tag and the h1 tag lies over the div tag.

CSS Positioning:

Static Positioning:

-All HTML elements are static in their position by default, and static just means go along with the HTML rules and keep to the default HTML flow, and that is what we see when we just have HTML without any CSS, or if we don't change this position property at all.

Relative Positioning:

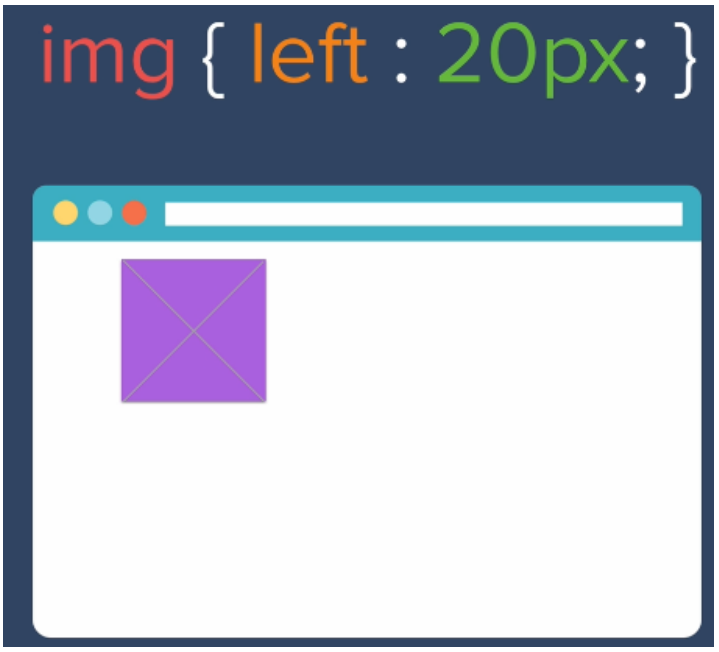
-this allows us to position the element that we select relative to how it would have been positioned had it been static. Don't forget to write position:relative

For example now we're saying give that image 30 pixels of space between its left edge and where the left edge used to be or give it a 30 pixels margin from the previous left edge of the img element.

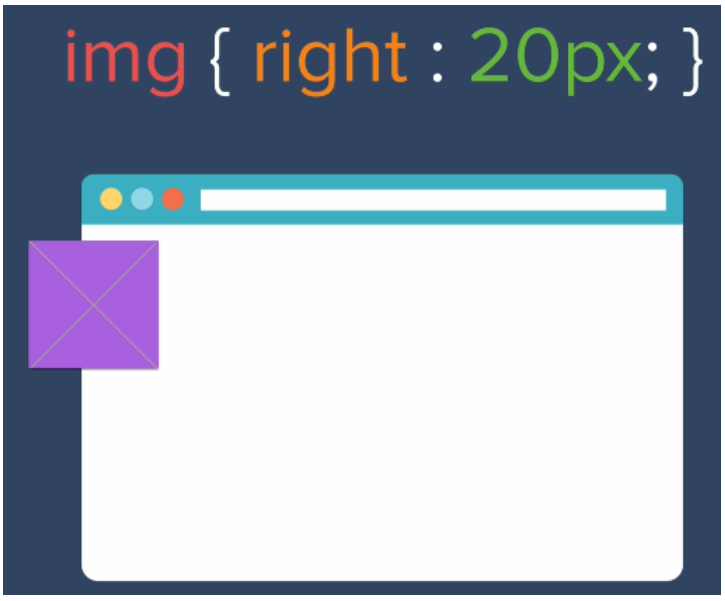
-We can set values for them in-order to determine how we wanna move our elements.

-This means it'll get moved down by 20 pixels.

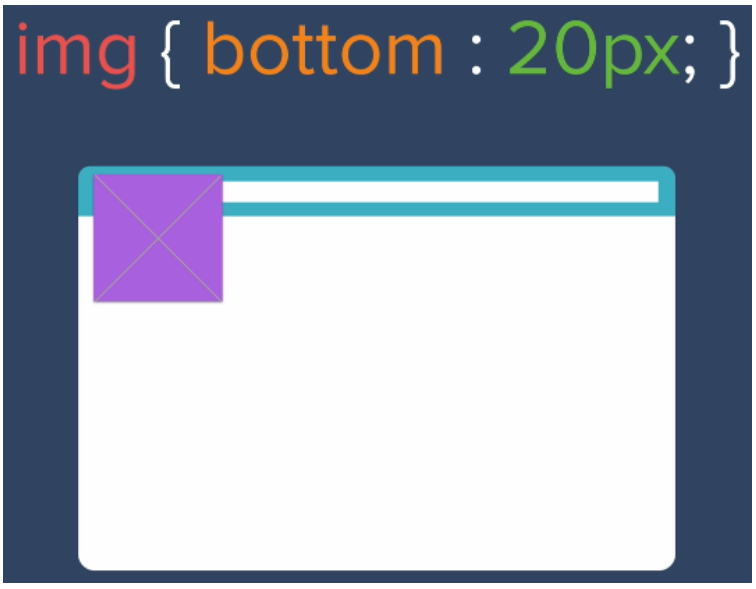
Relative Positioning Continued:



–it will get pushed to the right 20 pixels



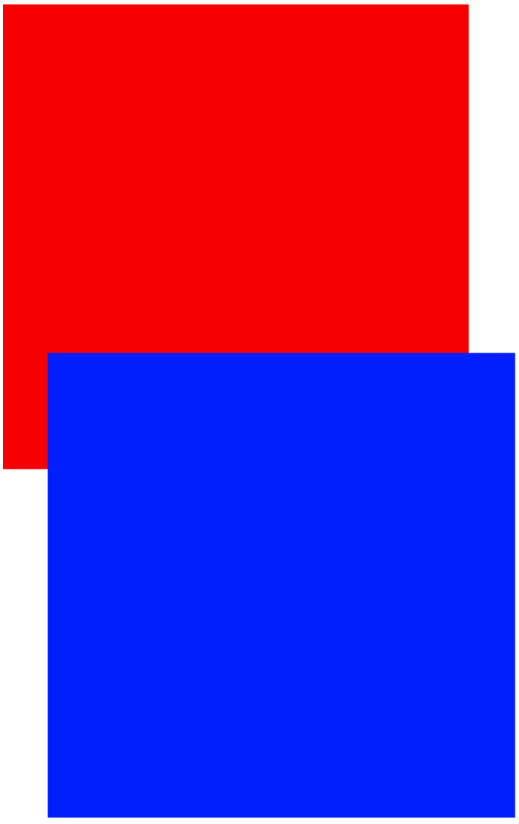
–it will get pushed to the left 20 pixels



–it will get pushed to the top 20 pixels

img {top:50px;}

—this means that we're taking the top of where that image used to be and we're giving it a 50 pixel margin from the top of our current image.



```
.middle-container{
  height: 200px;
  width: 200px;
  background-color: red;
}
.bottom-container{
  height: 200px;
  width: 200px;
  background-color: blue;
  position:relative;
  left:20px;
  bottom:50px;
}
```

—it just goes over the other box when we change its position.

HTML

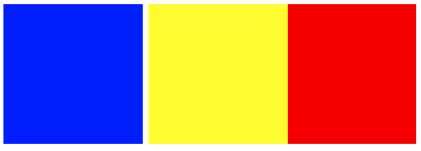
```
<body>
  <div class="red">
  </div>

  <div class="blue">
  </div>

  <div class="yellow">
  </div>

</body>
```

```
.red{
  height:100px;
  width:100px;
  background-color:red;
  display:inline-block;
  position:relative;
  left:200px;
}
.blue{
  height:100px;
  width:100px;
  background-color:blue;
  display:inline-block;
  position:relative;
  right:100px;
}
.yellow{
  height:100px;
  width:100px;
  background-color:yellow;
  display:inline-block;
  position:relative;
  right:100px;
}
```



—the reason why the spaces between our squares are inconsistent is because by making it an inline—block there's actually a little space that gets added in by the browser.