Sujata Batra

CSS UNITS - Sizes

Sujata Batra

- Relative length measurements:
  - px (pixels – size varies depending on screen resolution)
  - em (usually the height of a font's uppercase)
  - ex (usually the height of a font's lowercase)
  - Percentages (of the font's default size)
- Absolute-length measurements (units that do not vary in size):
  - in (inches)
  - cm (centimeters)
  - mm (millimeters)
  - pt (points; 1 pt = 1/72 in)
  - pc (picas; 1 pc = 12 pt)

COMMENTS IN CSS

- CSS uses the "block comment“
  - comment with Start with /*, and end it with */.

```
<style>
/*  p {
font-family: sans-serif;
font-size: 15pt;
  }    */
</style>
```

Style Classes

- Added to the elements of a that type should use a style rule.
  - **In CSS selectors** defines the class name, which is **preceded by a period.**
  - In HTML, the Element has the **class** attribute to specify class name

- **Class name should** be **descriptive** of the **purpose and not Presentation**
  - **nav, news, footer**
  - **redText,smallText,GreenBorder**

Style Classes

- Style classes be “generic,”
  - not tied to a specific element type.

```
.mytag  {
        font-size: 23px;
        background: gray;
}
```

- Style classes can also be tied to specific Element

```
a.anchorStyle  { text-decoration : none }

<a class="anchorStyle" href="somepage.html">Link text</a>
```

Example

```
.general-div {
    width: 300px; height: 300px; border: 1px solid #000;
}

.general-div.special {
    background-color: red;
}
```

- `<div class="general-div">Basic Division</div>`

```
<div class="general-div special">
        Division with Background Color
</div>
```

Box model

Sujata Batra

Box Model

- Introduction - Inline vs. Block level
  - Margin
  - Borders
  - Padding
  - Outlining
  - Visibility & Display

- http://www.css3-tutorial.net/css-box-model/introduction-inline-vs-block-level/

Sujata Batra

Box Model

- HTML element can be in one of the following states:
  - Block,
  - Inline
- *Can Change an inline element to a block element, or vice versa,*

```
li {  display: inline; }
```

```
span {  display: block; }
```

Block-Level and Inline Elements

- **Block level element**
  - Creates a "block" or "box".
  - Element will span the entire available width
  - Displayed with a new line.
  - It may contain inline elements and other block-level elements.

- **Inline Element**
  - Does not break the current flow.
  - Takes up the Just the space it needs to render its content
  - Elements do not begin with new line.
  - Can contain only data and other inline elements.
  - The width and height properties are ignored for inline elements.

HTML Elements

Block-Level Elements

- <article>
- <aside>
-
- <form>
- <h1> to <h6>
- <hr>
- <table>
-

Inline Elements

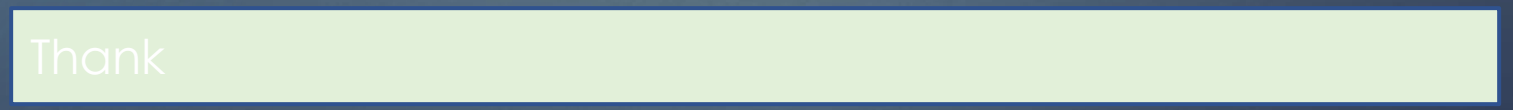
- <a>
- <img>
-
- <button>
- <input>
- <label>
- <select>
- <textarea>

Sujata Batra

Inline and Block Elements


```
<div>
    <span>Hello</span>
    <span>World</span>
</div>

<div>
      <div>Thank</div>
      <div>You</div>
</div>
```



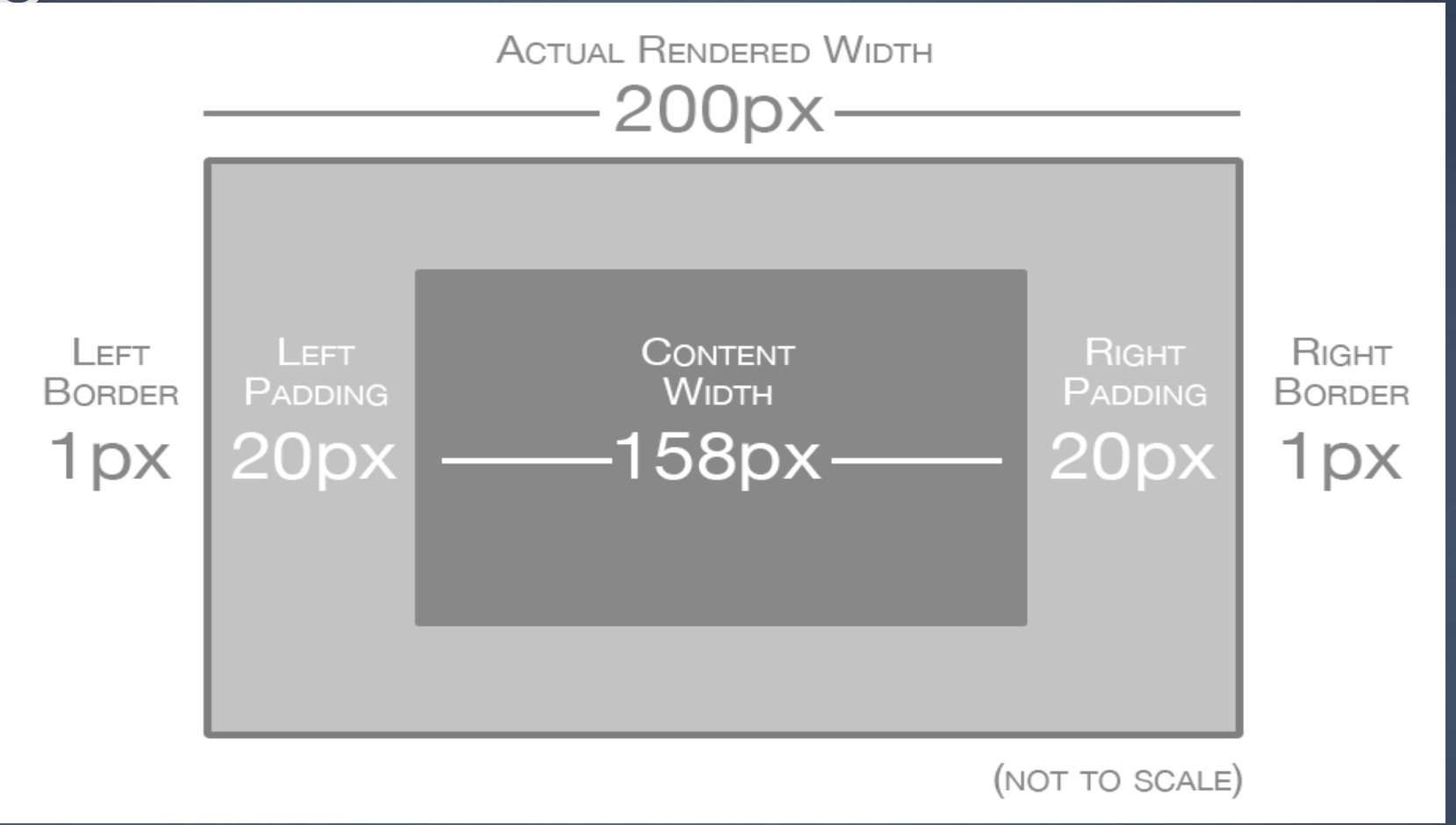
Box Model

- Block elements are inside a box
  - They have margins, padding and borders

- Element's layout is composed of the following:
  - Element's *content area*.
  - *border* around the element.
  - *padding* between the content and the border (inside the border)
  - *margin* between the border and other content (outside the border)

- **Width =** width + padding-left + padding-right + border-left + border-right
- **Height =** **height + padding-top + padding-bottom + border-top + border-bottom**

Demo



- http://codepen.io/vatsank/pen/rjJLqm

Sujata Batra

Outlining

- A kind of an extra border, for visual attention to element.
- The shorthand property includes
  - **outline-width**, **outline-style** and **outline-color**

- ```
  .box {
      background-color: #eee;
      outline: 3px solid LightCoral;
      border: 3px solid LightBlue;
      padding: 5px 10px;
  }
  ```
- **Differences between border and outline**
  - Cannot apply a different outline width, style and color for the four sides of an element
  - Its is not a part of the element's dimensions
  - The browser won't reserve the required space outline
  ```

Visibility & Display

- **Both of them are** used to control visibility:

- **The visibility property**
  - The initial value of the visibility property is visible,
  - Element is visible unless its changed

- Box 2

- ***The element can't be seen, but the browser still reserves the space for it***

display

- To specify the type of rendering box used for an element.
- **display:none**
  - Turns off the display of an element
  - All descendant elements also have their display turned off.
  - The document is rendered as though the element doesn't exist in the document tree.
- Hidden an element and be shown again by setting display to either **inline** or **block**
- *The element can't be seen, and the browser does't reserves the space for it*
- Box 2

Css properties

Sujata Batra

Categories of CSS properties

- Margin and Padding
- Border
- Font and text related
- Color
- Positioning and layout handling related.
- Background related properties.
- Lists related.
- Table related.

Margins

- Define the space around elements outside the border
- It can have **negative values** to deliberately overlap content
  - Can use translate() instead
- May affect the position of background elements (graphics and/or colors) in relation to the edges of the containing block element
- Can be defined independently on top, right, bottom and left
- Can also use CSS shorthand

CSS Margins

- **margin-bottom :** specify the bottom margin of an element.
- **margin-top :** specify the top margin of an element.
- **margin-left :** specify the left margin of an element.
- **margin-right :** specify the right margin of an element.

- **margin** shorthand property for setting margin properties in one declaration.

- Margin from all sides.

- Possible values: auto, length in px, %.

Margin

-
-
-
-
- [top] [right] [bottom] [left]
-
- p {margin: 15px; } : all four margins will be 15px
- padding: 2px 1em 0 1em;

Padding

- Defines the space around elements inside the border;
- Between the **border and the content** itself
- *cannot have negative values*
- Does NOT affect the position of background elements
- Can be defined independently on top, right, bottom and left,
- Can also use CSS shorthand

CSS Padding

- To specify how much space should appear between the content of an element and its border :

  - **padding-bottom** :specify the bottom padding of an element.
  - **padding-top:** specify the top padding of an element.
  - **padding-left:** specify the left padding of an element.
  - **padding-right** : specify the right padding of an element.
  - **padding** : shorthand for the all above properties.

  - [top] [right] [bottom] [left]

  - padding: 2px 1em 1em 2em;

padding

-
  This is a paragraph with a specified bottom padding.

-
  This is a paragraph with a specified top padding.

-
  This is a paragraph with a specified right padding.

CSS Borders

- You can set following **border properties** of an element:

  - **border-color :** set the color of the border.
  - **border-style** :set the style of the border.
  - **border-width** : set the width of the Border

  - **border** : set the **width, style and color** of the border in one declaration.

    - <p style="border: 4px solid red;">

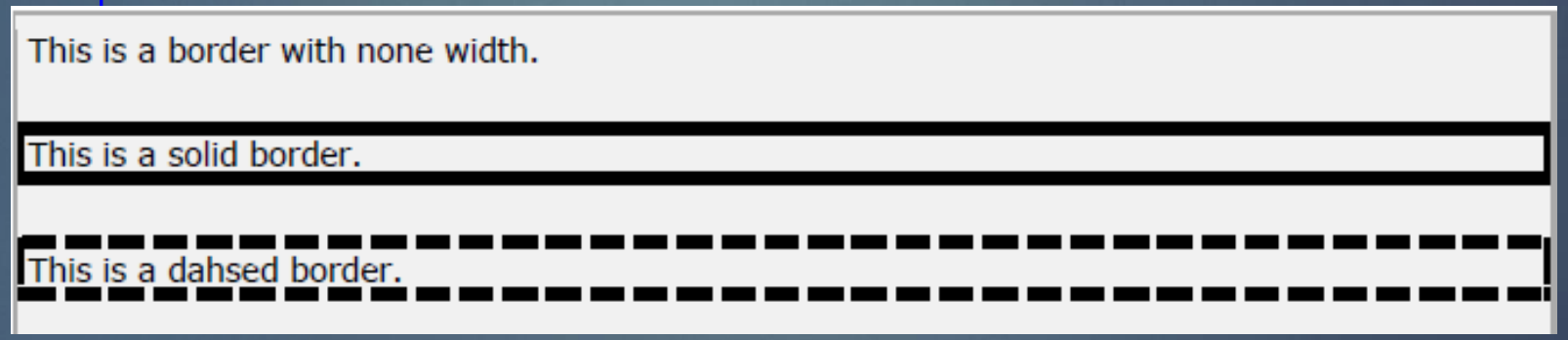
border-style

-
  This is a border with none width.

  This is a solid border.

  This is a dashed border.

- Output:



- Possible values:
  none, solid, dashed, double, groove, ridge, inset, outset

border-color

- <style>

```
 p.example1 {
        border-style: solid;
        border-color: #FF0000;
 }
</style>
<p class="example1">
 This example is showing all borders in same color.
</p>
```

- Output:

- Possible values:

  any color with valid format

Border Width

- Can individually change the width of the bottom, top, left and right borders of an element.
  - **border-bottom-width :** changes the width of bottom border.
  - **border-top-width** :changes the width of top border.
  - **border-left-width** :changes the width of left border.
  - **border-right-width :** changes the width of right border.

-
  This is a solid border whose width is 4px.

- Possible values:
  length in px, pt or cm or it should be thin, medium or thick.

Positioning

Sujata Batra

Positioning

- Introduction
- Relative positioning
- Absolute positioning
- Fixed positioning
- Floating elements
- Static positioning

Sujata Batra

CSS Positioning

- Relies on **CSS properties** rather than **tables** to design a Web page.
  - A **growing trend** is to configure pages **using CSS**
- The **technology** for this is called **"table-less layout"**
- **CSS-P** for **CSS P**ositioning
- **"table-less"** layouts **may include tables** to present info in a tabular manner **to facilitate design** of a **small portion of** the **page**.

CSS Positioning

- **Normal flow** causes the **browser to render** the elements in the **order they appear in** the **HTML/XHTML source code.**
- **Using CSS for page layout the location of elements can be changed.**
- **Cross browser-support positioning** is **more reliable** when the <**div**> element is used **for positioning**.

Absolute positioning

- Element is taken out of the normal flow of the page layout.
- Position is specified using the **left, right, top and bottom** attributes.
- To position in **relation to the nearest ancestor**
- If No positioned ancestors, it uses the document body
- Elements moves along with page scrolling.

Absolute Positioning

- *Position an Element 20px from the top of the browser viewport, and 20px from the left*

```
<div style="position:absolute; left: 20px; top: 20px;"> </div>
```

- To make the **child** element **positioned absolutely** from its **parent** element
  - **Parent** Element is **Positioned relative or Absolute**
  - **Parent Element should NOT be static**

Absolute Positioning

```
.container{
        width: 400px;
        height: 300px;
        border: 1px solid #e52325;
        background-color: #ddbdff;
        position: relative;
        margin: 0 auto;
    }
.content{
        width: 100px;
        height: 100px;
        border: 1px solid #ff748c;
        background-color: #fbdb65;
        position: absolute;
    }
```

Absolute Positioning

```
<div class="container">

    <div class="content" style="left:10px;top:10px;" id="topleft">
    </div>

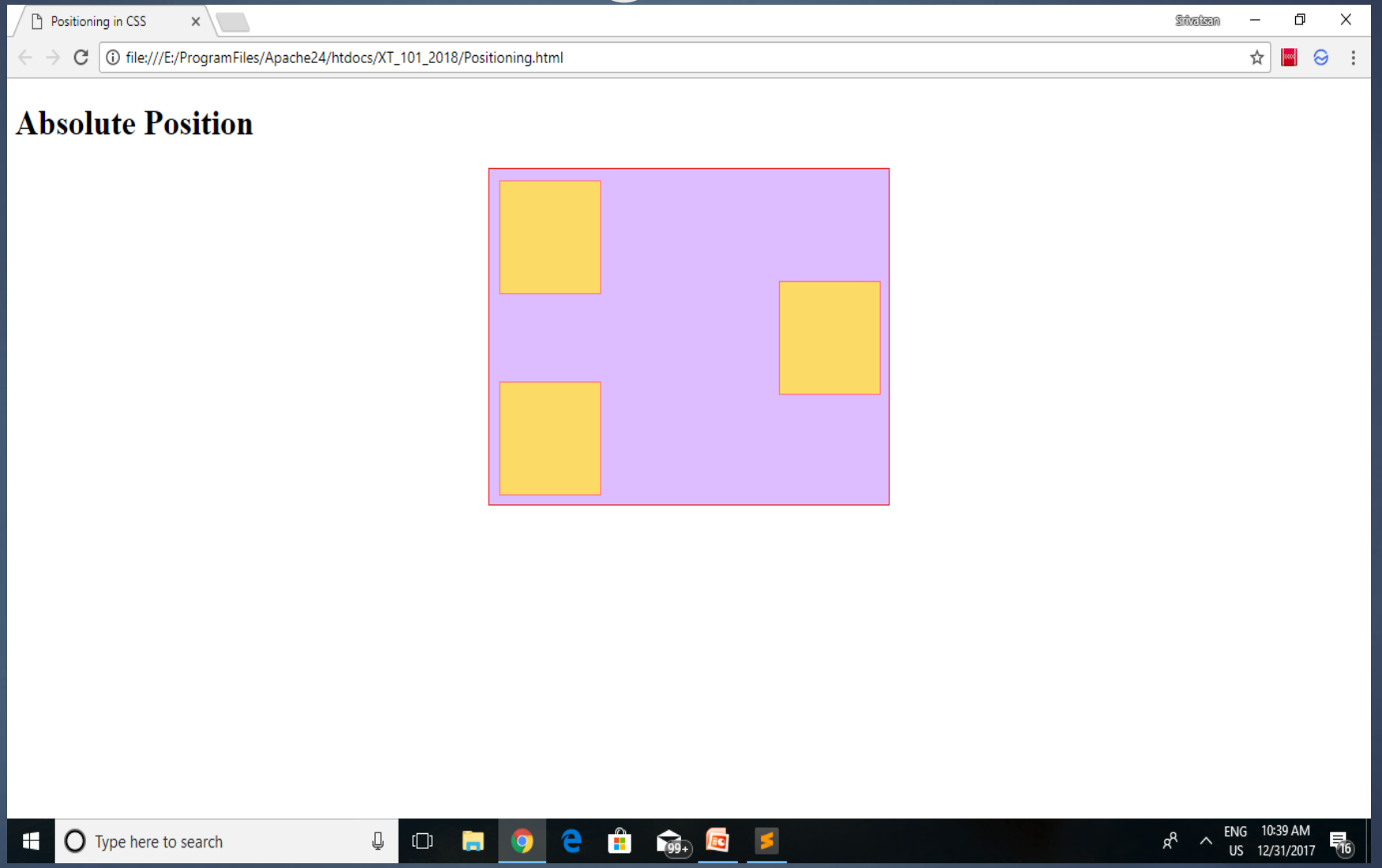
    <div class="content" style="left:10px;top:190px;" id="bottleft">
    </div>

     <div class="content" style="left: 290px;top:100px;" id="rightmiddle">
    </div>

</div>
```

Sujata Batra

Absolute Positioning



Sujata Bhatra

Relative Positioning

- Use to **slightly change** the **location** of an element **in relation** to where it would otherwise appear in normal flow
  - Setting the top, right, bottom, and left properties of a relatively-positioned element will cause it to be adjusted away from its normal position.
  - Other content will not be adjusted to fit into any gap left by the element.
- **Limits the scope of absolutely positioned child elements**.
  - Any element that is a child of the relatively positioned element can be absolutely positioned within that block.

Relative Positioning

```
.container{
        width: 400px;
        height: 300px;
        border: 1px solid #e52325;
        background-color: #ddbdff;
        position: relative;
        margin: 0 auto;
    }
.content{
        width: 100px;
        height: 100px;
        border: 1px solid #ff748c;
        background-color: #fbdb65;
        position: relative;
    }
```

Relative Positioning

Fixed Positioning

- Restricts an element to a specific position in the viewport, which stays in place during scroll:
  - They are not considered to be bound by the viewport:
- Absolutely-positioned elements are still bound by the viewport and will cause scrolling:
- A fixed element does not leave a gap in the page where it would normally have been located.

Fixed Positioning

```
.container{
        width: 400px;
        height: 300px;
        border: 1px solid #e52325;
        background-color: #ddbdff;
        position: relative;
        margin: 0 auto;
    }
.content{
        width: 100px;
        height: 100px;
        border: 1px solid #ff748c;
        background-color: #fbdb65;
        position: absolute;
    }
```

Absolute Positioning

```
<div class="container">

    <div class="content" style="left:10px;top:10px; position: fixed; " id="topleft">
    </div>

    <div class="content" style="left:10px;top:190px;" id="bottleft">
    </div>

     <div class="content" style="left: 290px;top:100px;" id="rightmiddle">
    </div>

</div>
```

Static Positioning

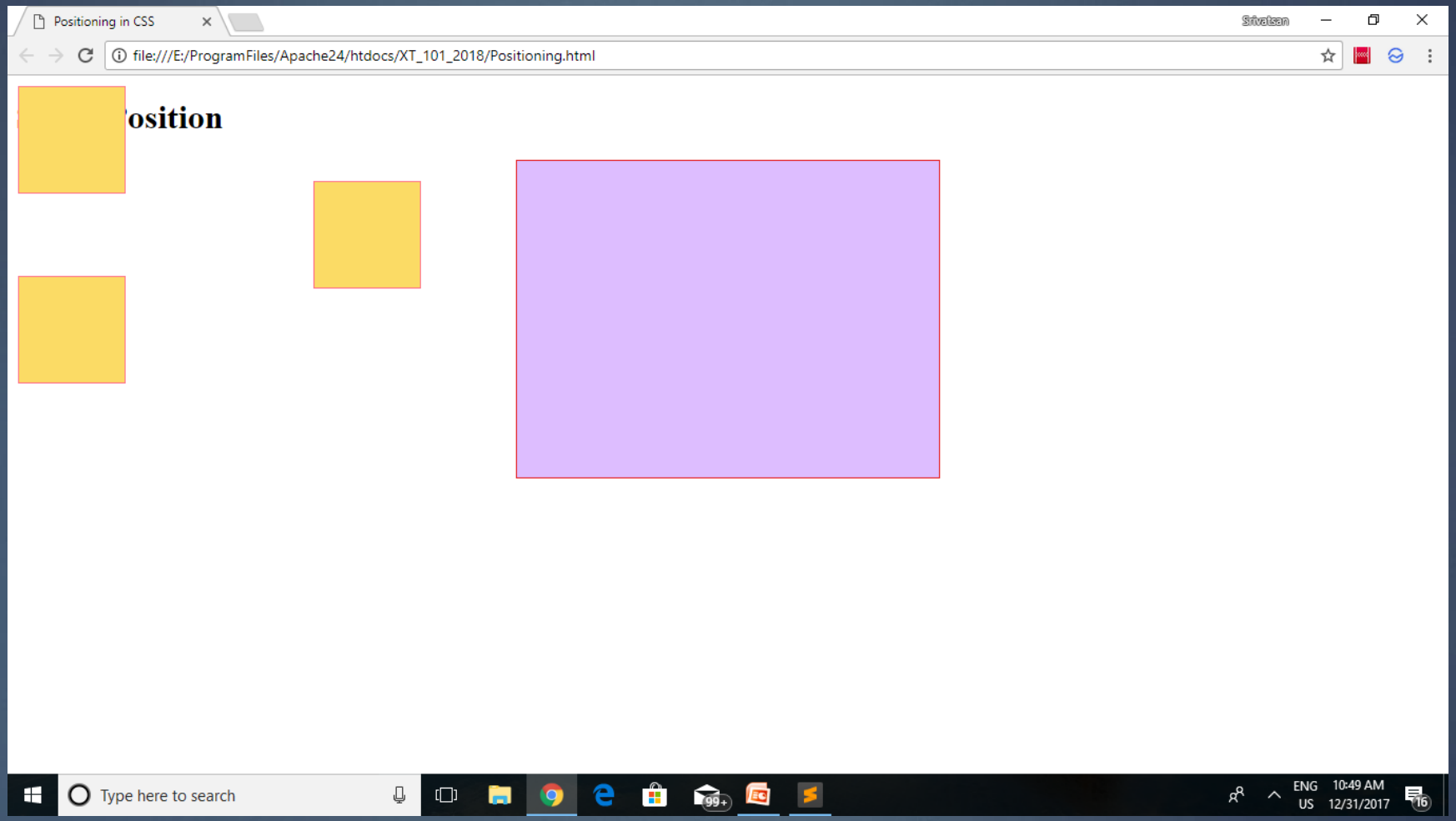
- *A static element is said to be not positioned and an element with its position set to anything else is said to be positioned.*
- Elements are positioned static by default.
  - They are not affected by the top, bottom, left, and right properties.
  - Uses the normal flow of the page:
- Relative positioning makes children absolute positioned be positioned relative to them,
- Static allows absolutely positioned children to ignore their position and be positioned relative to the nearest relative positioned element.

Static and Relative Positioning

```
.container{
        width: 400px;
        height: 300px;
        border: 1px solid #e52325;
        background-color: #ddbdff;
        position: static;
        margin: 0 auto;
}
  .content{
        width: 100px;
        height: 100px;
        border: 1px solid #ff748c;
        background-color: #fbdb65;
        position: absolute;
}
```

Sujata Bata

Static Positioning



Sujata Bhatra

Positioning

-
-
-
-

- inner div would be positioned 20px from the top of the outer div, and 20px from the left edge of same, because the outer div isn't positioned with position:static
- because the outer div isn't positioned with position:static because we've explicitly set it to use position:relative.

Floating elements

- A very powerful positioning technique .
- By default, block level element takes up document's 100% width
- It takes element out of normal order and let other elements float around it.
- **Elements** "**float**" on the **right** or **left** side **of either** the **browser window or another element**
- **Elements** should have an **intrinsic width configured**
  - Images are often **configured** using the **float** property.
- `float: left, right, none;`

Floating

```
.container{

        width: 75%;
        border: 1px solid blue;
}

img.styleImage{

        width: 50px;
        height: 50px;
        margin: 0px 10px 0 10px;
        float: left;
}
```

Sujata Batra

Clear Floating

- **To stop** flow **clear** property
  - "**clear**" or terminate a **float**
- **Clear** values: **left, right**, and **both**

```
.rightfloat {

float:right;
        margin: 5px;
        clear:right;
        border: solid;

}
```

Alignment vs. float vs. position

1. If possible, lay out an element by *aligning* its content
   - horizontal alignment: text-align
     - set this on a block element; it aligns the content within it (not the block element itself)
   - vertical alignment: vertical-align
     - set this on an inline element, and it aligns it vertically within its containing element
2. If alignment won't work, try *floating* the element
3. If floating won't work, try *positioning* the element
   - absolute/fixed positioning are a last resort and should not be overused

- In general, **do not use float when you mean align**,
- The **alignment defines the direction of the flow within an element**, while a **float takes an element out of the current flow**.