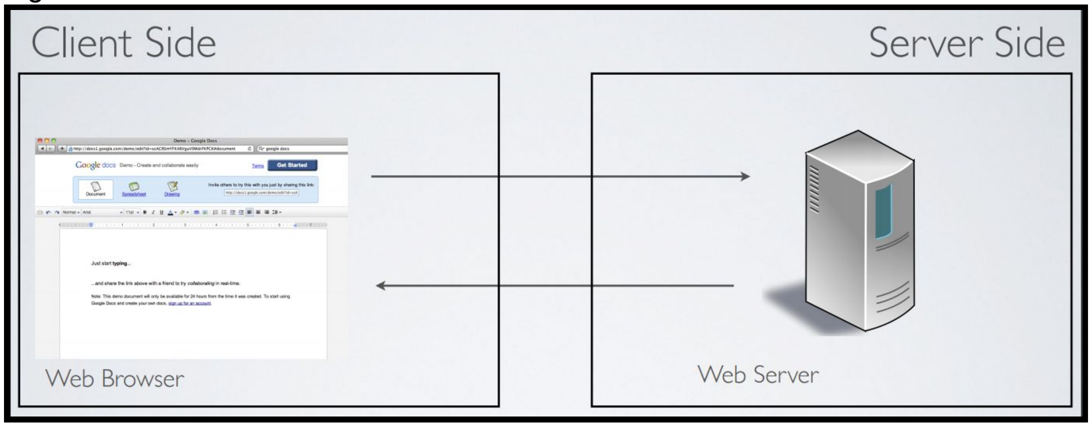
Introduction:

- A **web application** is a computer program that utilizes web browsers and web technology to perform tasks over the Internet.
- There's 2 parts to a web application:
  1. **Client side:**
     - The front end.
     - Runs on your browser on your machine (laptop, PC, phone, tablet, etc).
  2. **Server side:**
     - The back end.
     - Runs on a server.

  E.g.



  The frontend and backend communicates with each other and sends data back and forth.
- Examples of frontend languages/markup language:

HTML	(Markup language)
CSS	(Markup language)
Javascript	(Programming language)

- Examples of backend languages:
  - Python
  - Java
  - Go
  - NodeJS
- Web development has changed and evolved tremendously in the past few decades.
- One consequence of this evolution is that the application is moving from the server to the client. This makes for richer content. The traditional web platform was browsers, but now, modern web platforms are smartphones and tablets.
- A second consequence is cloud computing. Data storage and data processing are moving from the desktop to the cloud.
- HTML deals with content, while CSS deals with presentation and style and Javascript deals with processing.

HTML:

- Stands for Hyper-Text Markup Language.
- HTML is the standard markup language for creating Web pages.
- HTML describes the structure of a Web page.
- HTML consists of a series of elements.
- HTML elements tell the browser how to display the content.
- HTML is a markup language. It is not a programming language.
- It describes a web page's content and structure.

- **History of HTML:**

HTML 1	Invented by Tom Berners-Lee (1991)
HTML 2	First standard from W3C (1995)
HTML 4	Separation between content and presentation CSS (1997)
HTML 5	Multimedia (2008)

- **Terminology:**
- **Markup:** Tags starting and ending with angle brackets.
  E.g. **<p> </p>**
  Note: Some tags have a start and end tag, while others only have the starting tag.
  E.g.
  <p> </p> ← Has both start and end tags. All end tags have a / in them.
  <img> ← Only has start tag.
- **Content:** Everything else.
- **Element:** A start and end tag and the content in between.
  I.e. An HTML element is defined by a start tag, some content, and an end tag.
  E.g. **<p> Some content </p>** ← This entire thing is an element.
  Note: Some HTML elements have no content. They are called **empty elements**.
  E.g. The **<br>** tag defines a line break, and is an empty element without a closing tag.
- **Attribute:** The name/value pairs specified in a start tag.
  All HTML elements can have attributes.
  Attributes provide additional information about elements.
  Attributes are always specified in the start tag.
  Attributes usually come in name/value pairs like: name="value".
  E.g. Consider the **<a>** tag. The href attribute specifies the URL of the page the link goes to.
  I.e. **<a href="..."> … </a>** ← Notice how the href attribute is in the starting tag.
- **Comments:** Tags that will be ignored at rendering. In HTML, comments are denoted using **<!-- -->**.
  E.g. **<!-- This is a comment. -->**
- **Skeleton of a HTML document:**

```
<!DOCTYPE HTML>
<html>
        <head>
                This is where page metadata and invisible content goes.
                Anything you want to show on the website should not go here.
        </head>

        <body>
                This is where visible page content goes.
                I.e. Anything you want to show on the website should go here.
</body>
</html>
```

- **Differences between XHTML and HTML:**
- At first, browsers were quite forgiving about missing/mismatched tags in HTML, however, different browsers rendered differently.
- Then, HTML became XHTML (aka HTML 4 and 5).

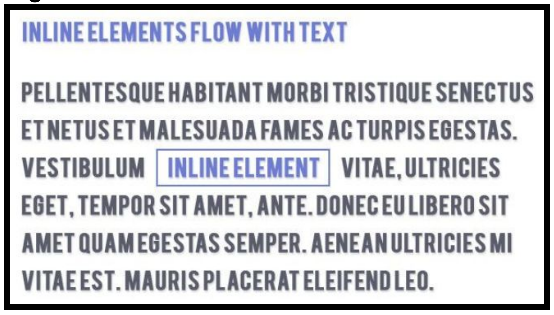
- Now, there is homogeneity across browsers for the most part and it is easier to parse in Javascript. Furthermore, the styling disappeared in HTML 4 to become CSS and elements related to styling became deprecated. Some of these elements include **<b>**, **<i>**, **<font>**, etc.
  Note: The browser will still render them and not tell you that they're deprecated, but you should not use them.
- **Note:** If there are errors or missing elements/tags, the browser will not give an error. The browser will still try to load the HTML page.
- You can check if HTML is valid using **https://validator.w3.org/**.
- **List of HTML Tags:**
- **<!DOCTYPE html>:** All HTML documents must start with a <!DOCTYPE html> declaration. It is used to tell the browser about what document type to expect.
- **<a>:** The <a> tag defines a hyperlink, which is used to link from one page to another. The most important attribute of the <a> element is the href attribute, which indicates the link's destination.
  By default, links will appear as follows in all browsers:
  An unvisited link is underlined and blue.
  A visited link is underlined and purple.
  An active link is underlined and red.
  Note: If the <a> tag has no href attribute, it is only a placeholder for a hyperlink.
- **<body>:** The <body> tag defines the document's body. It contains all the contents of an HTML document, such as headings, paragraphs, images, hyperlinks, tables, lists, etc.
  Note: There can only be one <body> element in an HTML document.
- **<div>:** The <div> tag defines a division, block or a section in an HTML document. The <div> tag is used as a container for HTML elements, which is then styled with CSS or manipulated with JavaScript.
  The <div> tag is easily styled by using the class or id attribute.
  Any sort of content can be put inside the <div> tag.
  Note: By default, browsers always place a line break before and after the <div> element.
- **<h1> - <h6>:** The <h1> to <h6> tags are used to define HTML headings.
  <h1> defines the most important heading. <h6> defines the least important heading.
  Text in <h1> tags are larger than text in <h2> tags, and so on.
  Text in <h6> tags are the smallest.
- **<head>:** The <head> element is a container for metadata and is placed between the <html> tag and the <body> tag. Metadata is data about the HTML document. It is not displayed. Metadata typically defines the document title, character set, styles, scripts, and other meta information. You would put things like links to external CSS or Javascript files, the title tag among other stuff in the head tag.
- **<html>:** The <html> tag represents the root of an HTML document. The <html> tag is the container for all other HTML elements except for the <!DOCTYPE> tag.
  Note: You should always include the lang attribute inside the <html> tag, to declare the language of the webpage. This is meant to assist search engines and browsers.
- **<link>:** The <link> tag defines the relationship between the current document and an external resource. It is most often used to either contain internal CSS or link to external CSS file(s).
  Note: The <link> element is an empty element. It contains attributes only.
- **<p>:** The <p> tag defines a paragraph. Browsers automatically add a single blank line before and after each <p> element.
- **<script>:** The <script> tag is used to embed JavaScript. It either contains internal Javascript, or it points to an external script file through the src attribute.

- **<span>:** The <span> tag is an inline container used to mark up a part of a text, or a part of a document. It is easily styled by CSS or manipulated with JavaScript using the class or id attribute. The <span> tag is much like the <div> element, but <div> is a block-level element and <span> is an inline element.
- **<title>:** The <title> tag defines the title of the document. The title must be text-only, and it is shown in the browser's title bar or in the page's tab. It is required in HTML documents. The contents of a page title is very important for search engine optimization. The page title is used by search engine algorithms to decide the order when listing pages in search results.
  The <title> element:
    - Defines a title in the browser toolbar.
    - Provides a title for the page when it is added to favorites.
    - Displays a title for the page in search-engine results.

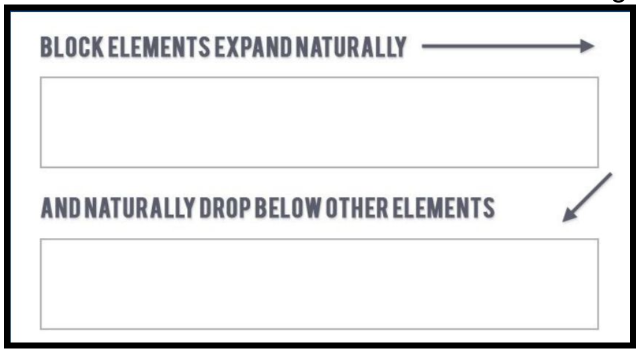
  Note: You can not have more than one <title> element in an HTML document.
- **Semantic vs Non-Semantic Tags:**
- **Semantic tags** are HTML tags that indicate their expected use to both the user and the browser. Examples include:
  <form>
  <h1>, <h2>, <h3>, <h4>, <h5>, <h6>
  <table>
- Some advantages of semantic tags are:
    1. **Design:** Meaning of page is always the same regardless of style.
       E.g. It doesn't matter if your heading is blue or red or green, it's still an <h1> heading.
       E.g. It doesn't matter what the size of the table is, it's still a <table> tag.
    2. **Accessibility:** Screen readers can change voice tone on a tag.
       E.g. If a word is in a <strong> tag, the screen reader would read that word louder.
    3. **Search Engine Optimization:** Density of keywords is higher when more semantic tags are used.
- **Non-semantic tags** tell nothing about its content. They are generic and have no specific purpose. Examples of non-semantic tags are <span> and <div>. <span> is a generic inline element while <div> is a generic block element. They are used more for creating natural divisions throughout your page.
- **Class vs Id:**
- Class:
    - The HTML class attribute is used to specify a class for an HTML element.
    - Multiple HTML elements can share the same class name.
    - The class attribute is often used to point to a class name in a style sheet. It can also be used by JavaScript to access and manipulate elements with the specific class name.
- Id:
    - The HTML id attribute is used to specify a unique id for an HTML element.
    - You cannot have more than one element with the same id in an HTML document.
    - The id attribute specifies a unique id for an HTML element. The value of the id attribute must be unique within the HTML document.
    - The id attribute is used to point to a specific style declaration in a style sheet. It is also used by JavaScript to access and manipulate the element with the specific id.

- **Block vs Inline vs Inline-Block:**
- Elements on a web page can be displayed in different ways.

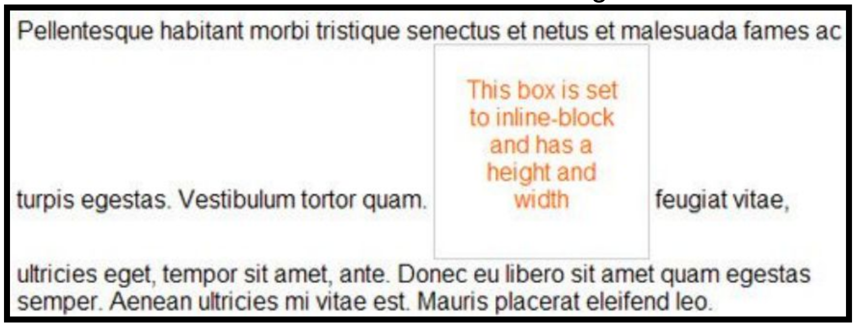
1. **Inline:**
    - Examples include: <strong>, <a>, <br>, <span>
    - It doesn't have defined width/height and can't have block elements inside it. It also doesn't create newlines.
    - This is often used for changing part of a text inside a paragraph.
    - An inline element does not start on a new line and it only takes up as much width as necessary.
    - E.g.


INLINE ELEMENTS FLOW WITH TEXT
PELLENTESQUE HABITANT MORBI TRISTIQUE SENECTUS ET NETUS ET MALESUADA FAMES AC TURPIS EGESTAS. VESTIBULUM
INLINE ELEMENT
VITAE, ULTRICIES EGET, TEMPOR SIT AMET, ANTE. DONEC EU LIBERO SIT AMET QUAM EGESTAS SEMPER. AENEAN ULTRICIES MI VITAE EST. MAURIS PLACERAT ELEIFEND LEO.

2. **Block:**
    - Examples include: <p>, <h1> - <h6>, <div>
    - The height and width can be specified and changed. But by default, the width is the full width of the parent element and the height is enough to fit the content.
    - It forces creation of newlines.
    - A block-level element always starts on a new line and takes up the full width available. It stretches out to the left and right as far as it can.


BLOCK ELEMENTS EXPAND NATURALLY
AND NATURALLY DROP BELOW OTHER ELEMENTS

3. **Inline-block:**
    - Examples include: <img>
    - These are inline elements that can have a height/width.


Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac
turpis egestas. Vestibulum tortor quam.
This box is set to inline-block and has a height and width
feugiat vitae,
ultricies eget, tempor sit amet, ante. Donec eu libero sit amet quam egestas semper. Aenean ultricies mi vitae est. Mauris placerat eleifend leo.

- **Div vs Span:**
- Both <div> and <span> are used to define parts of a web page.
- <span> shows the inline portion of a document while <div> shows a block-level portion of a document.
  I.e. A div is a block element while a span is an inline element.
- The div should be used to wrap sections of a document, while spans are used to wrap small portions of text, images, etc.

CSS:

- Stands for Cascading Style Sheets.
- Previously, CSS was a part of HTML and we could style the webpage using HTML. However, that changed. One of the reasons it changed is because we could want the same style over multiple pages. Another reason is that across multiple platforms, we could want different layouts. For example, a website is not going to look the same for mobile and desktop. Lastly, we want to have custom layout for people with conditions. For example, some people are colour blind or they need to have large font and we want to let users be able to superimpose their own css to satisfy their needs.
- Comments in CSS are denoted with: **/* */**. Anything placed between /* and */ will be commented out.
- **CSS Format:**

```
selector{
        property: value;
        property: value;
        property: value;
}
```

- The selector points to the HTML element you want to style.
- The property: value pairs are usually fixed. This means that you have to use what is available and cannot create your own property: value pairs.
- E.g.
  Here, p is the selector, color and text-align are properties while red and center are values.

```
p {
  color: red;
  text-align: center;
}
```

- There are a few ways we can write CSS selectors:
  1. **CSS element Selector:**
     - The element selector selects HTML elements based on the element name.
     - E.g.
       Here, all <p> elements on the page will be center-aligned, with a red text color.

```
p {
  color: red;
  text-align: center;
}
```

  2. **CSS id Selector:**
     - The id selector uses the id attribute of an HTML element to select a specific element.
     - The id of an element is unique within a page, so the id selector is used to select one unique element.
       Note: If you use the same id for multiple elements, you won't get an error message.

- To select an element with a specific id, write a hash character, #, followed by the id of the element.
- E.g.
  The CSS rule below will be applied to the HTML element with id="para1".
```
#para1 {
  text-align: center;
  color: red;
}
```

3. **CSS class Selector:**

- The class selector selects HTML elements with a specific class attribute.
- To select elements with a specific class, write a period, ., followed by the class name.
- Unlike id's, you can use the same class name for multiple elements.
- E.g.
  In this example all HTML elements with class="center" will be red and center-aligned.
```
.center {
  text-align: center;
  color: red;
}
```
- **Note:** You can also specify that only specific HTML elements should be affected by a class.
- E.g.
  In this example only <p> elements with class="center" will be red and center-aligned.
```
p.center {
  text-align: center;
  color: red;
}
```

4. **CSS Universal Selector:**

- The universal selector, *, selects all HTML elements on the page.
- E.g.
  The CSS rule below will affect every HTML element on the page.
```
* {
  text-align: center;
  color: blue;
}
```

5. **CSS Descendent Selector:**

- The descendant selector matches all elements that are descendants of a specified element. The first simple selector within this selector represents the parent element and the second simple selector represents the descendant element we're trying to match.
  E.g.
```
p strong{
        background-color: yellow;
}
```
  This applies to all <strong> elements that are inside a <p> element.

6. **CSS Multiple Element Selector:**
   - The multiple element selector is used to select multiple, different elements.
     E.g.
     ```
     p, strong, h1 {
             background-color: yellow;
     }
     ```
     This applies to all <p>, <strong> and <h1> elements.

- **Conflicting Selectors:**
- When two selectors appear to conflict, the more specific selector takes precedence.
- E.g. Consider the code below:
  HTML:
  ```
  <p>
          <em> Green or red? </em>
  </p>
  ```

  CSS:
  ```
  p em {
          color: green;
  }

  em {
          color: red;
  }
  ```

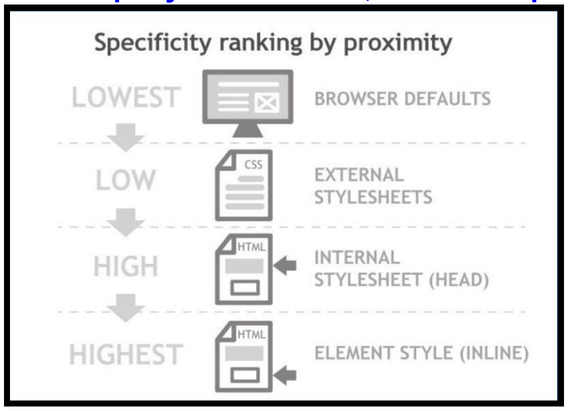
  Since "p em" is more specific than "em", it will be used.
- **Specificity Precedence:**
  1. Ids (Highest specificity)
  2. Classes
  3. Elements (Lowest specificity)

  Note: Parent elements are more specific than children elements.
- Furthermore, if you have multiple rules with the same specificity, then the rule further down the style sheet wins.
  E.g. If you have 2 rules about the <h1> tag, the rule that's further down wins.
- **CSS Inheritance:**
- Children elements inherit parent styles in most cases.
- If the styles of a child HTML element are not specified, the element inherits the styles of its parent.
- **CSS - Inline, embedded or separate file:**
- There are three ways of inserting a style sheet:
  1. **External CSS:**
     - Here, you write the CSS in a separate file and save it as a .css file.
     - Each HTML page must include a reference to the external style sheet file inside the <link> element, inside the head section.
     - E.g.
       ```
       <head>
               <link rel="stylesheet" href="mystyle.css">
       </head>
       ```
  2. **Internal CSS:**
     - You can include internal CSS inside the <style> element, inside the head section.
     - This way is not preferred.

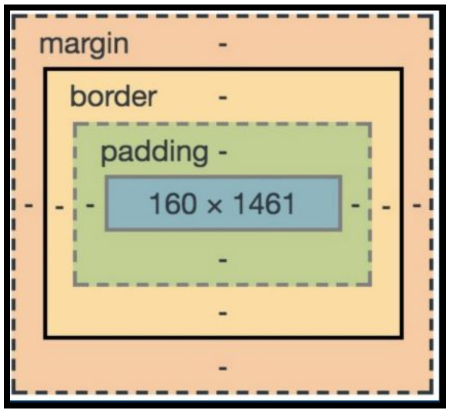
- E.g.
  <head>
  <style>
  … CSS stuff here ...
  </style>
  </head>

3. **Inline CSS:**

- An inline style may be used to apply a unique style for a single element.
- To use inline styles, add the style attribute to the relevant element. The style attribute can contain any CSS property.
- This way is also not preferred.
- E.g.
  <p style="color:red;">This is a paragraph.</p>

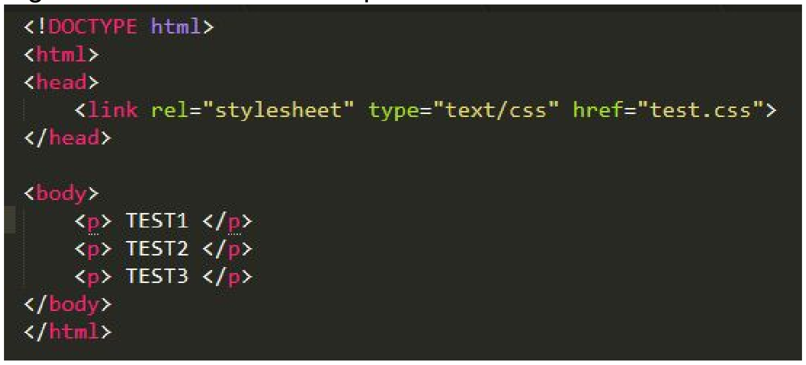


- **CSS Box Model:**
- Think of each element as a box or rectangle.

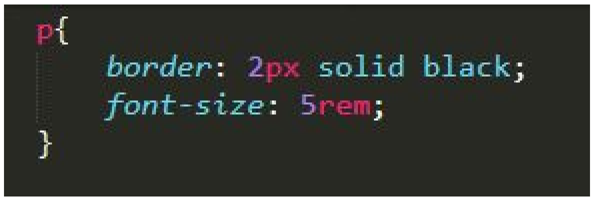


- The size of an element in the box model is determined by the blue rectangle (shown in the picture above) and the 3 rectangular rings around it (padding, border and margin).
- The blue rectangle in the center is the size of the content of the element.
  Note: The numbers are in pixels.
  We can modify the size of the content using the height and width properties in CSS.
- Just outside the content is the **padding**, which is the space between the content and the border of the element. It's used to give some space between the content and the border so that the content isn't touching the border.
- The **border** comes after the padding and it can be any size you want it to be.

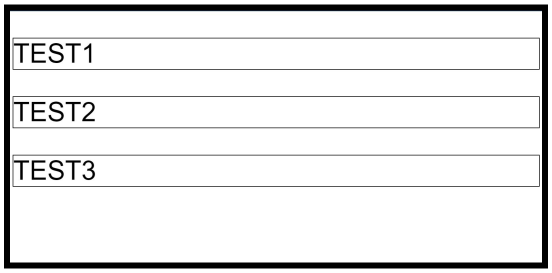
- Lastly, we have the **margin**, which is the area around the border (clearing space). It's used so that the elements are not touching each other.
  I.e. The margin is used to give some space between elements.
- All size elements are properties in CSS.
- If the content is block-displayed, we can change its size using height and width. We can specify the content size using either the number of pixels or with a percentage of the parent element it is held in.
- E.g. Consider the code and picture below.



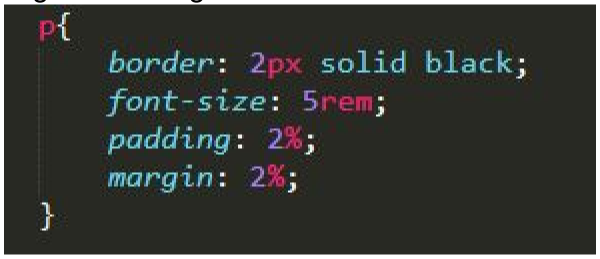
```
<!DOCTYPE html>
<html>
<head>
    <link rel="stylesheet" type="text/css" href="test.css">
</head>

<body>
    <p> TEST1 </p>
    <p> TEST2 </p>
    <p> TEST3 </p>
</body>
</html>
```



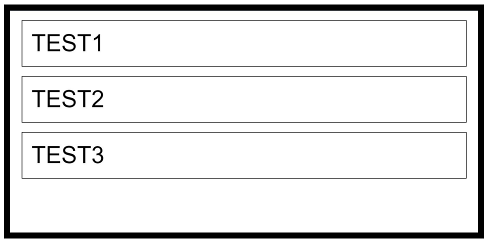
```
p{
    border: 2px solid black;
    font-size: 5rem;
}
```



Now, if I add margin and padding, like shown below, notice how the area within each box and text increased, and the area between boxes and the area between a box and the edges all changed.

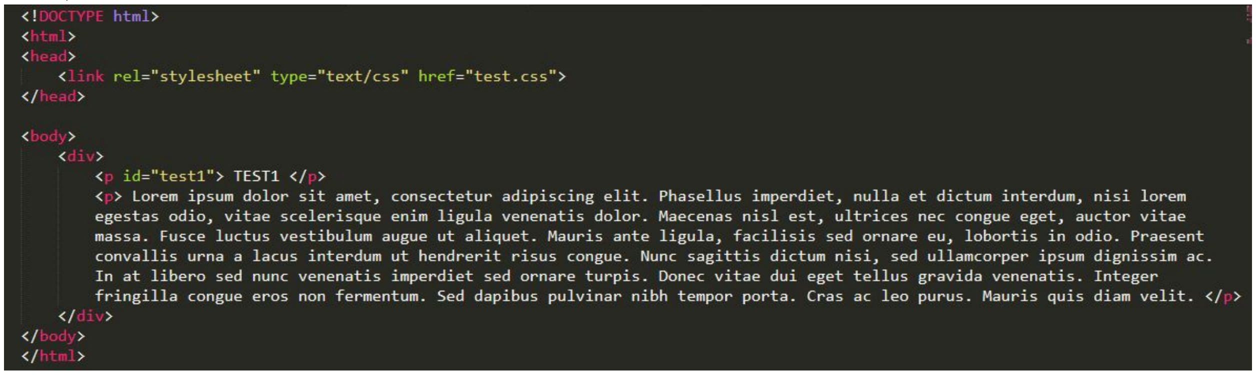


```
p{
    border: 2px solid black;
    font-size: 5rem;
    padding: 2%;
    margin: 2%;
}
```

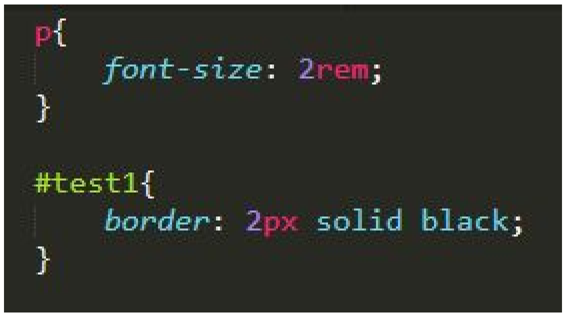


- **CSS Positioning:**
- **Positioning of elements** refers to a deviation from their **natural flow**, which is where the elements are placed by default.
- Here are a few ways you can position an element using CSS:
    1. **Static:** The default position of the element.
       I.e. Where the element is in their natural flow.
       When you put an element in HTML without changing the position type, you are by default putting it into a static position.
    2. **Fixed:** It fixes the element in one position in the viewport of the browser.
       Elements in a fixed position do not move even with scrolling.
    3. **Relative:** Allows us to change the position of an element relative to its natural location.
       E.g. We can say that we want to move an element x pixels to the right of where it is naturally located.
    4. **Absolute:** An absolutely positioned element is positioned relative to the first positioned parent. However, if an absolute positioned element has no positioned parent, it uses the document body, and moves along with page scrolling.
    5. **Sticky:** The element is treated like a relative value until the scroll location of the viewport reaches a specified threshold, at which point the element takes a fixed position where it is told to stick.

  Note: If you don't specify a position, it will have, by default, the static position.
- **Note:** Except for position: absolute, if both top and bottom are specified and are not auto, top wins. If both left and right are specified and are not auto, left wins when direction is left to right (ltr) (English, horizontal Japanese, etc) and right wins when direction is right to left (rtl) (Persian, Arabic, Hebrew, etc). In the case of position: absolute, if you have both top and bottom, it will span the page. Same with left and right. Top, bottom, left, and right are css properties.
- **CSS Float:**
- The CSS float property places an element on the left or right side of its container, allowing text and inline elements to wrap around it.
- The float property is used for positioning and formatting content.
- The float property can have one of the following values:
    - **Left:** The element floats to the left of its container.
    - **Right:** The element floats to the right of its container.
    - **None:** The element does not float. It will be displayed just where it occurs in the text. This is default.
    - **Inherit:** The element inherits the float value of its parent.
- E.g. Consider the code and picture below.
  Here, I did not use float: left on test1.

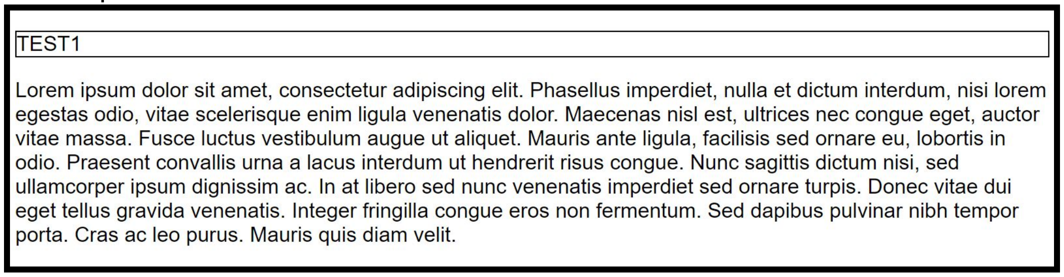


```html
<!DOCTYPE html>
<html>
<head>
    <link rel="stylesheet" type="text/css" href="test.css">
</head>

<body>
    <div>
        <p id="test1"> TEST1 </p>
        <p> Lorem ipsum dolor sit amet, consectetur adipiscing elit. Phasellus imperdiet, nulla et dictum interdum, nisi lorem
        egestas odio, vitae scelerisque enim ligula venenatis dolor. Maecenas nisl est, ultrices nec congue eget, auctor vitae
        massa. Fusce luctus vestibulum augue ut aliquet. Mauris ante ligula, facilisis sed ornare eu, lobortis in odio. Praesent
        convallis urna a lacus interdum ut hendrerit risus congue. Nunc sagittis dictum nisi, sed ullamcorper ipsum dignissim ac.
        In at libero sed nunc venenatis imperdiet sed ornare turpis. Donec vitae dui eget tellus gravida venenatis. Integer
        fringilla congue eros non fermentum. Sed dapibus pulvinar nibh tempor porta. Cras ac leo purus. Mauris quis diam velit. </p>
    </div>
</body>
</html>
```



```
p{
    font-size: 2rem;
}

#test1{
    border: 2px solid black;
}
```

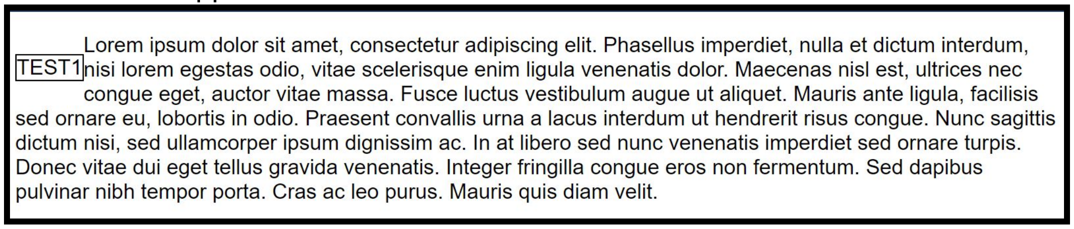
The output looks like this:



TEST1

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Phasellus imperdiet, nulla et dictum interdum, nisi lorem egestas odio, vitae scelerisque enim ligula venenatis dolor. Maecenas nisl est, ultrices nec congue eget, auctor vitae massa. Fusce luctus vestibulum augue ut aliquet. Mauris ante ligula, facilisis sed ornare eu, lobortis in odio. Praesent convallis urna a lacus interdum ut hendrerit risus congue. Nunc sagittis dictum nisi, sed ullamcorper ipsum dignissim ac. In at libero sed nunc venenatis imperdiet sed ornare turpis. Donec vitae dui eget tellus gravida venenatis. Integer fringilla congue eros non fermentum. Sed dapibus pulvinar nibh tempor porta. Cras ac leo purus. Mauris quis diam velit.

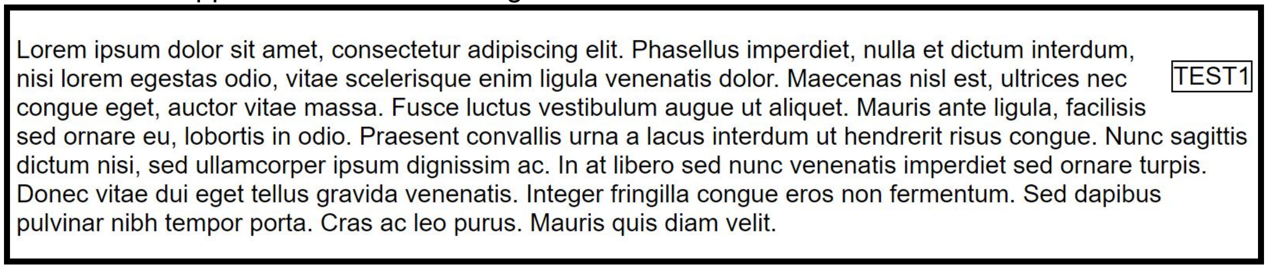
This is what happens when I do float: left:



TEST1

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Phasellus imperdiet, nulla et dictum interdum, nisi lorem egestas odio, vitae scelerisque enim ligula venenatis dolor. Maecenas nisl est, ultrices nec congue eget, auctor vitae massa. Fusce luctus vestibulum augue ut aliquet. Mauris ante ligula, facilisis sed ornare eu, lobortis in odio. Praesent convallis urna a lacus interdum ut hendrerit risus congue. Nunc sagittis dictum nisi, sed ullamcorper ipsum dignissim ac. In at libero sed nunc venenatis imperdiet sed ornare turpis. Donec vitae dui eget tellus gravida venenatis. Integer fringilla congue eros non fermentum. Sed dapibus pulvinar nibh tempor porta. Cras ac leo purus. Mauris quis diam velit.

This is what happens when I do float: right:



Lorem ipsum dolor sit amet, consectetur adipiscing elit. Phasellus imperdiet, nulla et dictum interdum, nisi lorem egestas odio, vitae scelerisque enim ligula venenatis dolor. Maecenas nisl est, ultrices nec congue eget, auctor vitae massa. Fusce luctus vestibulum augue ut aliquet. Mauris ante ligula, facilisis sed ornare eu, lobortis in odio. Praesent convallis urna a lacus interdum ut hendrerit risus congue. Nunc sagittis dictum nisi, sed ullamcorper ipsum dignissim ac. In at libero sed nunc venenatis imperdiet sed ornare turpis. Donec vitae dui eget tellus gravida venenatis. Integer fringilla congue eros non fermentum. Sed dapibus pulvinar nibh tempor porta. Cras ac leo purus. Mauris quis diam velit.

TEST1

Notice when I do float: left or float: right, the “Lorem ipsum …” text wraps around TEST1.