Display

The display property specifies if/how an element is displayed.

Every HTML element has a default display value depending on what type of element it is. The default display value for most elements is block or inline.

Block-level Elements

A block-level element always starts on a new line and takes up the full width available (stretches out to the left and right as far as it can).

Here are the block-level elements in HTML:

<address>	<article>	<aside>	<blockquote>	<canvas>	<dd>	
<dl>	<dt>	<fieldset>	<figcaption>	<figure>	<footer>	<form>
<h1>-<h6>	<header>	<hr>		<main>	<nav>	<noscript>
	<p>	<pre>	<section>	<table>	<tfoot>	
<video>						

Inline Elements

An inline element does not start on a new line and only takes up as much width as necessary.

Here are the inline elements in HTML:

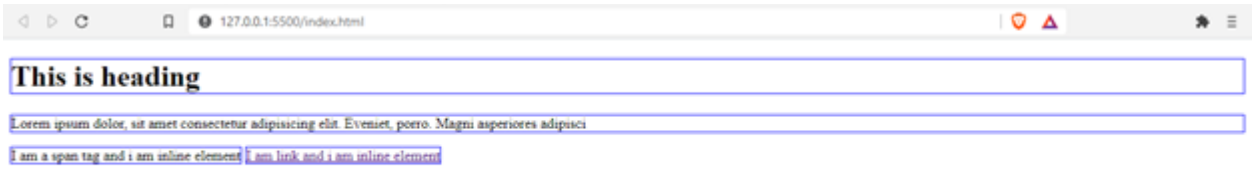
<a>	<abbr>	<acronym>	<b>	<bdo>	<big>	
<button>	<cite>	<code>	<dfn>		<i>	<img>
<input>	<kbd>	<label>	<map>	<object>	<output>	<q>
<samp>	<script>	<select>	<small>			<sub>
<sup>	<textarea>	<time>	<tt>	<var>		

- Block – New line, Takes Full Width, Height and Width can be applied.
- Inline – No new line, no full width, height and width cannot be applied.
- Inline-block – No new line, no full width, height and width can be applied.

```html
<body>
  <div class="main-div">
    <h1>This is heading</h1>
    <p>Lorem ipsum dolor, sit amet consectetur adipisicing elit.
      Eveniet, porro. Magni asperiores adipisci</p>
    <span>I am a span tag and i am inline element</span>
    <a href="">I am link and i am inline element</a>
  </div>
</body>
```

```css
h1,
p,
span,
a {
  border: 1px solid blue;
}
```

Output:



As mentioned, every element has a default display value. However, you can override this.

Changing an inline element to a block element, or vice versa, can be useful for making the page look a specific way, and still follow the web standards.

A common example is making inline <li> elements for horizontal menus.

```css
.navbar li {
  display: inline;
}

a {
  display: block;
  margin-top: 20px;
}


a {
  display: inline-block;
  margin-top: 20px;
  width: 1200px;
  height: 200px;
}
```

display:none VS visibility:hidden

Hiding an element can be done by setting the display property to none. The element will be hidden, and the page will be displayed as if the element is not there.

visibility:hidden; also hides an element. However, the element will still take up the same space as before. The element will be hidden, but still affect the layout.

Example,

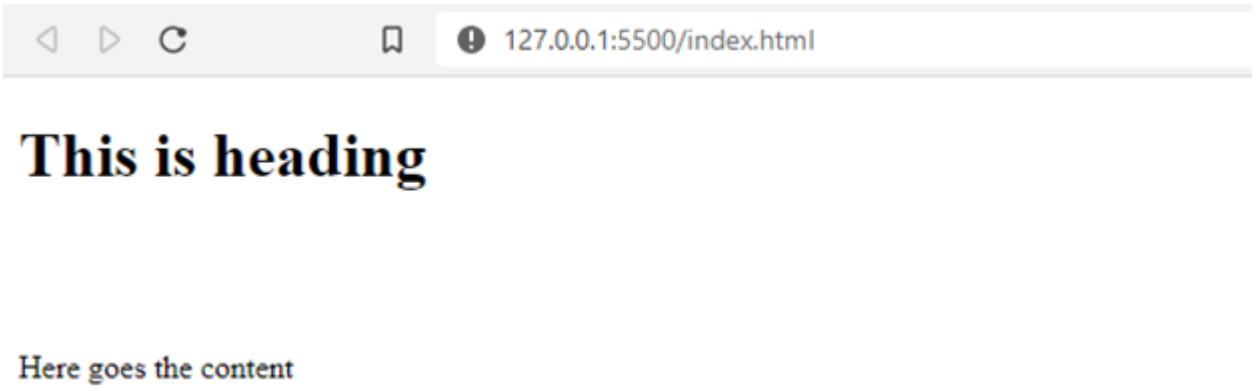
```html
<body>
  <div class="main-div">
    <h1>This is heading</h1>
    <h1 class="demo">This is second heading</h1>
    <p>Here goes the content</p>
  </div>
</body>
```

```css
.demo {
  display: none;
}
```

Output:


```css
.demo {
  visibility: hidden;
}
```

Output:



Position

The position property specifies the type of positioning method used for an element.

There are five different position values:

- static
- relative
- fixed
- absolute
- sticky

Elements are then positioned using the top, bottom, left, and right properties. However, these properties will not work unless the position property is set first. They also work differently depending on the position value.

position: static;

HTML elements are positioned static by default.

Static positioned elements are not affected by the top, bottom, left, and right properties.

An element with position: static; is not positioned in any special way; it is always positioned according to the normal flow of the page.

position: relative;

An element with position: relative; is positioned relative to its normal position.

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.

```
.main-div {
  width: 600px;
  height: 200px;
  position: relative;
  right: 0;
  bottom: 0;
}
```

position: fixed;

An element with position: fixed; is positioned relative to the viewport, which means it always stays in the same place even if the page is scrolled. The top, right, bottom, and left properties are used to position the element.

A fixed element does not leave a gap in the page where it would normally have been located.

```
.fixed {
  position: fixed;
  right: 0;
  bottom: 0;
  height: 50px;
  width: 100px;
}
```

position: absolute;

An element with position: absolute; is positioned relative to the nearest positioned ancestor (instead of positioned relative to the viewport, like fixed).

However; if an absolute positioned element has no positioned ancestors, it uses the document body, and moves along with page scrolling.

Note: A "positioned" element is one whose position is anything except static.

```css
.main-div {
  width: 600px;
  height: 200px;
  position: relative;
}

a {
  position: absolute;
  right: 0;
  height: 50px;
  width: 100px;
}
```

Output:

position: sticky;

An element with position: sticky; is positioned based on the user's scroll position.

A sticky element toggles between relative and fixed, depending on the scroll position. It is positioned relative until a given offset position is met in the viewport - then it "sticks" in place (like position:fixed).

```html
<body>
    <div class="main-div">
        <h1>This is heading</h1>
        <p>Lorem ipsum dolor, sit amet consectetur adipisicing elit.
            Eveniet, porro. Magni asperiores adipisci</p>
    <span>I am a span tag and i am inline element</span>
    <a href="">I am link and i am inline element</a>
    <p>Lorem ipsum dolor sit amet, consectetur adipisicing elit. Perspiciatis ve
    Inventore omnis magni amet officia quidem corrupti saepe! Tempora a molestia
    Odio blanditiis aspernatur quasi vitae corrupti quae, dignissimos consequunt
    Nobis necessitatibus ipsa fuga dicta quae laboriosam esse assumenda deleniti
    Voluptatum repellendus dignissimos magni ex error commodi illo nostrum, quis
```

```css
h1 {
  background-color: black;
  color: white;
  position: -webkit-sticky;
  /* safari and browser that doesnot support sticky position */
  position: sticky;
  top: 0;
}

.main-div {
  width: 900px;
  height: auto;
}
```

Output:

Media Queries

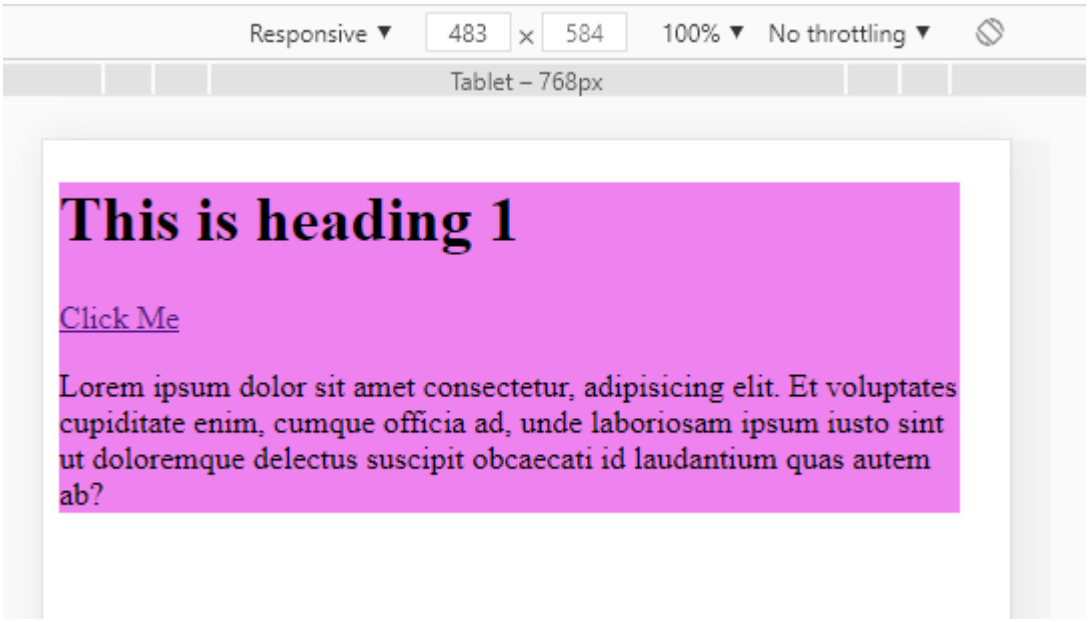
Media queries are used to design responsive web pages. Using media queries are a popular technique for delivering a tailored style sheet to desktops, laptops, tablets, and mobile phones (such as iPhone and Android phones).

Example,

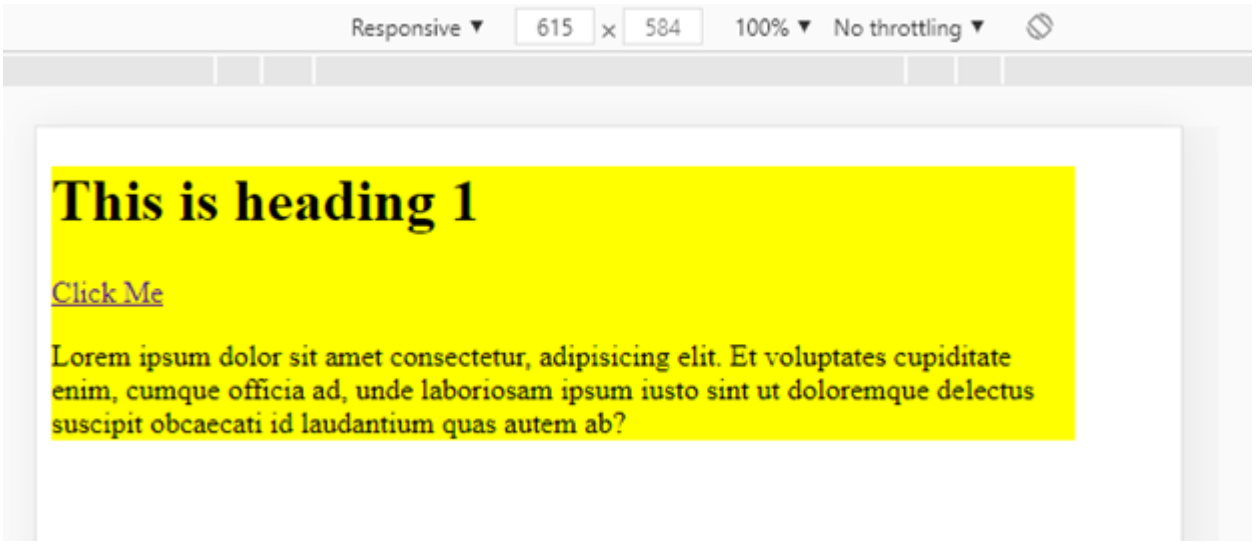
```
<body>
  <div class="main-div">
    <h1>This is heading 1</h1>
    <a href="">Click Me</a>
    <p>Lorem ipsum dolor sit amet consectetur, adipisicing elit. Et volupta
      laboriosam ipsum iusto sint ut doloremque delectus suscipit obcaecati
  </div>
</body>
```

```css
@media screen and (max-width: 1024px) {
  .main-div {
    width: 750px;
    background-color: yellowgreen;
  }
}

@media screen and (max-width: 800px) {
  .main-div {
    width: 550px;
    background-color: yellow;
  }
}

@media screen and (max-width: 600px) {
  .main-div {
    width: 450px;
    background-color: violet;
  }
}
```

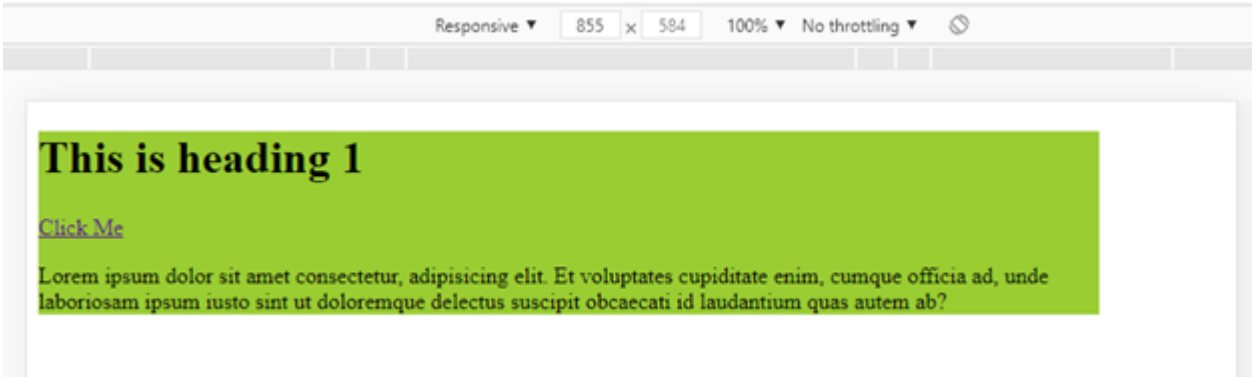
Output 1:



Output 2:



Output 3:


Responsive
855
584
100%
No throttling
This is heading 1
Click Me
Lorem ipsum dolor sit amet consectetur, adipisicing elit. Et voluptates cupiditate enim, cumque officia ad, unde laboriosam ipsum iusto sint ut doloremque delectus suscipit obcaecati id laudantium quas autem ab?