Front-End v23.0

Dashboard / My courses / FE23.0 / CSS v23.0 / Day 2 | Pre-work / CSS | Day 2 | Pre-work

Back to 'Day 2 | Pre-work'

CSS | Day 2 | Pre-work

CSS

Day 2

Table of contents:

- Positioning Elements
    - Static Positioning
    - Fixed Positioning
    - Absolute positioning
    - Sticky Positioning
    - Relative positioning
- Difference between <div> and <span> elements
- Box shadow
- Measurements
    - Pixels
    - Rems
    - Ems
    - Percentage
    - Viewports
- Text Formatting
    - Fonts and Typography
        - Font-family
        - Font-style
        - Font-size
        - Font-weight
    - Further Managing Text Styles
        - Alignment
        - Decoration
        - Spacing
        - Shadow
        - Transformation
        - Indenting
    - Web Fonts
- Colors
        - Font-color
        - Background-color

Positioning Elements

Each element on the page is automatically positioned one after another. That doesn’t make the page look all that good in the beginning. In order to position HTML elements in a more appealing way, we can use the position property which can have the values:

1. **Static**
2. **Absolute**
3. **Relative**
4. **Fixed**
5. **Sticky**

Static Positioning

HTML elements are positioned static by default and they are not affected by left, right, top and bottom properties. So it is not positioned in any special way, but it follows the normal flow of the page.

Fixed Positioning

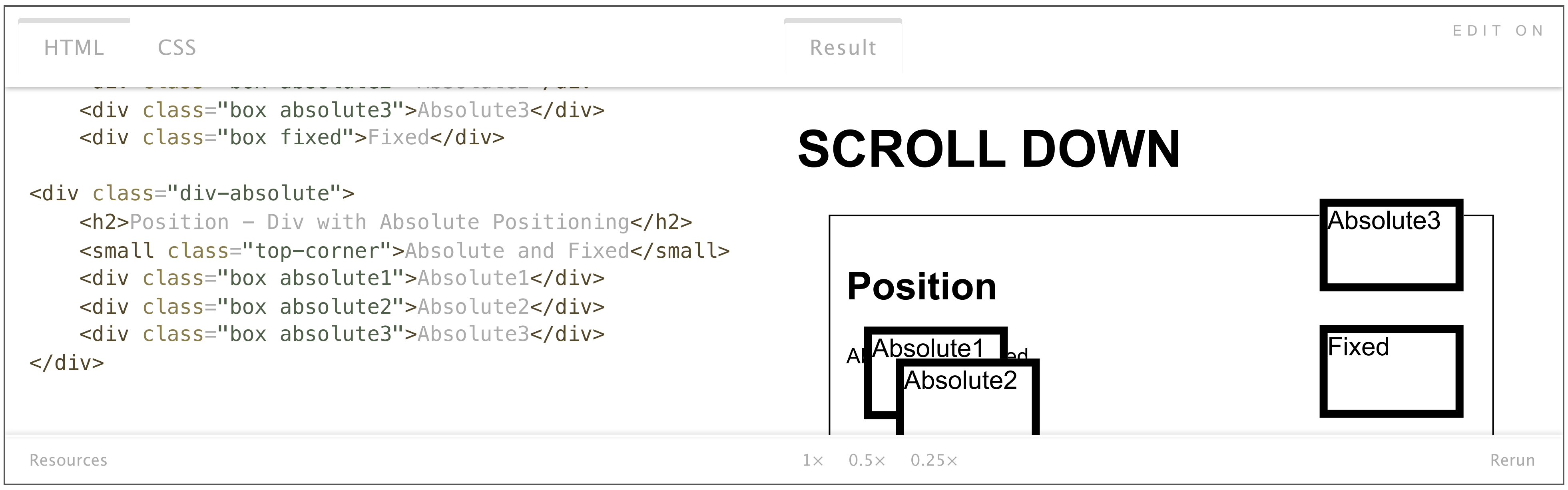
Fixed positioning allows you to move an element to an absolute location but only within the current browser viewport, so when the browser is scrolled the element **remains exactly where** it has been **placed.**

Absolute positioning

An element with absolute positioning is removed from the normal document rendering flow and is capable to “flow” into its released space. You can position such an element using the **top**, **right**, **left** or **bottom** properties. As a result, the element will simply rest on top of all other elements. An element with position: absolute; is **positioned relative to the nearest positioned ancestor** (instead of positioned relative to the viewport, like fixed)

Sticky Positioning

Sticky positioning can be considered a mix of relative and fixed positioning. The element with this property shows the same behavior as with relative position until it reaches a specific position set, then it "sticks" to the screen.



```
    <div class="box absolute2">Absolute2</div>
    <div class="box absolute3">Absolute3</div>
    <div class="box fixed">Fixed</div>

<div class="div-absolute">
    <h2>Position - Div with Absolute Positioning</h2>
    <small class="top-corner">Absolute and Fixed</small>
    <div class="box absolute1">Absolute1</div>
    <div class="box absolute2">Absolute2</div>
    <div class="box absolute3">Absolute3</div>
</div>
```

SCROLL DOWN

Position

Absolute3

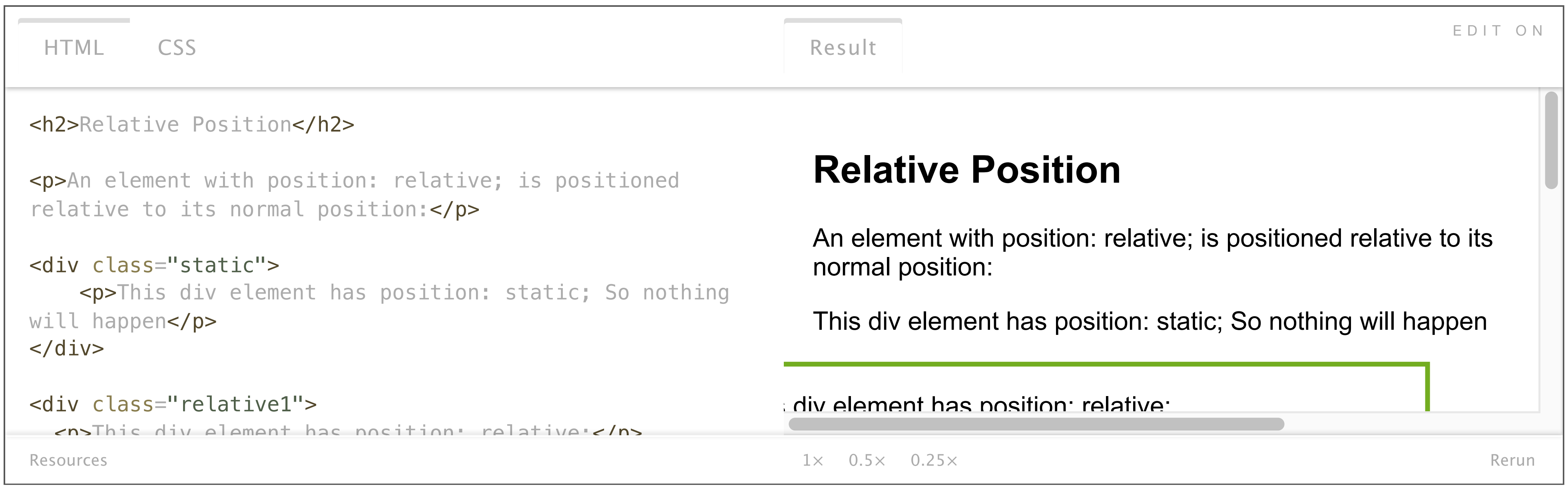
Absolute1

Absolute2

Fixed

Relative positioning

Opposite to the absolute positioning, you can move the element in the normal document flow. An element with position: relative; is positioned relative to its **normal position**.



```
<h2>Relative Position</h2>

<p>An element with position: relative; is positioned
relative to its normal position:</p>

<div class="static">
    <p>This div element has position: static; So nothing
will happen</p>
</div>

<div class="relative1">
  <p>This div element has position: relative;</p>
```

Relative Position

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

This div element has position: static; So nothing will happen

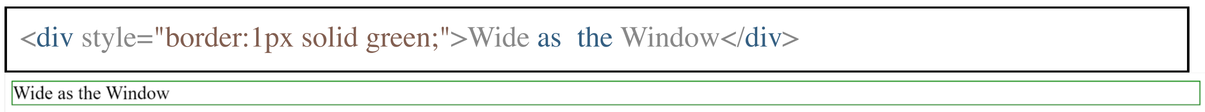
div element has position: relative;

Here is an example covering **all the position values**:

```
<!DOCTYPE html>
<html lang="en" >
 <head>
    <meta charset="UTF-8" />
    <meta http-equiv="X-UA-Compatible"   content="IE=edge" />
    <meta name="viewport"  content ="width=device-width, initial-scale=1.0" />
    <title>Document</title>
    <style>
      * {
       margin: 0;
       padding: 0 ;
       font-family: Arial, Helvetica, sans-serif;
     }

     h1 {
       text-align: center;
       margin: 10px;
     }

     body {
       height: 150vh;
     }

     span {
       color: tomato;
     }

     .positioning-container {
       border-radius: 10%;
       color: white;
        background-image: linear-gradient(rgb(11, 76, 140) 50%, #4d4949 );
       height: 10vh;
       text-align: center;
     }

     #div-static {
       position: static;
     }

      #div-absolute {
       position: absolute;
       bottom: 20%;
       right: 30% ;
       width: 40vw;
     }

     #div-sticky {
       position: sticky;
        top: 0;
       width: 100%;
       z-index: 1 ;
       margin-bottom: 5vh;
     }

     #div-relative {
       position: relative;
        bottom: -50px;
       margin-bottom: 50px;
       width: 100% ;
     }

     #div-fixed {
       position: fixed;
       bottom: 0px;
       width: 100% ;
     }
   </style>
   </head>
 <body>
     <h1>CSS Positioning Elements</ h1 >
    <hr />
    <div class= "positioning-container" id="div-static">
      <h2>This element has </h2>
      <span>position: static;</span>
    </div>
     <p> *An element with position static; will follow the normal flow of the page as the element
above</p>
    <br />< br />
    <div class="positioning-container"  id="div-absolute">
      <h2>Position: absolute</h2>
      This div element has position absolute to the viewport. The normal flow rules do not apply, it is
free of normal flow restrictions<br/>
      <span>position: absolute; bottom: 20%; right: 30%;</ span>
    </div>
    <div class ="positioning-container" id ="div-sticky">
      <h2>Position: sticky</h2>
      This div element has position sticky when it reaches a certain position previously set it will
stick to its position.<br />
      <span> position: sticky; top :0;</span>
     </div>
    <h2> Position: relative</h2>
    <p > Non-relative (aka static) element would appear just below this paragraph:</p>
    <div  class="positioning-container"  id="div-relative" >
      This div element has position relative to the position where it would be in a normal flow;
<br />it can of course go left, right, up and down from its own initial position.<br /> Try
commenting out the command<br  /><span> position:relative; bottom: -50px; </span >
     </div>
    < div  class= "positioning-container"  id ="div-fixed" >
       <h2 > Position: fixed </ h2 >
      Fixed element is glued to the edges of the viewport, regardless of how you change the viewport
or scroll. < br />< span > position: fixed; bottom: 0px; </ span >
     </ div >
   </ body >
</ html >
```

Difference between <div> and <span> elements

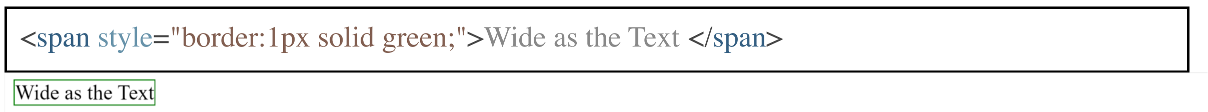
<div> and **< span >** are both types of container elements, wrappers that nest other HTML elements. The main difference between them is that **<div>** is filling all the width that is available.



```
<div style="border:1px solid green;">Wide as  the Window</div>
```

Wide as the Window

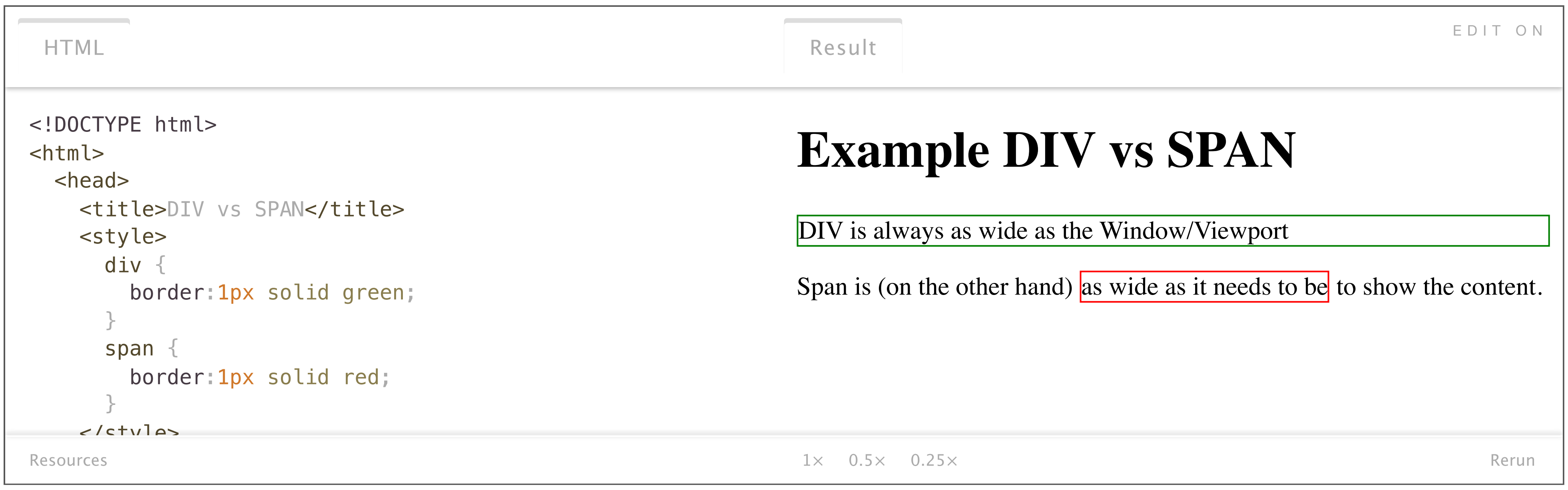
On the other side the **<span>** element is only wide as the text or elements that it contains (wide as necessary to display).



```
<span style="border:1px solid green;">Wide as the Text </span>
```

Wide as the Text

We use **<span>** tags for holding **text** as an inline container or to group elements that share the same purpose or styling using classes or ids. It is normally used within a paragraph, or a div element.
As its name says, a <div> element (division) defines a section in a HTML document. They are better suited for a container or a "wrapper" containing any HTML elements such as images, paragraphs, etc.
Tip: In order to have better perception of where div and span elements flow on the page, you can assign border properties as follows:



```
<!DOCTYPE html>
<html>
  <head>
    <title>DIV vs SPAN</title>
    <style>
      div {
        border:1px solid green;
      }
      span {
        border:1px solid red;
      }
    </style>
```

Example DIV vs SPAN

DIV is always as wide as the Window/Viewport

Span is (on the other hand) as wide as it needs to be to show the content.

```
<!DOCTYPE html>
<html>
 <head >
   <title>DIV vs SPAN</title >
   <style>
     div {
         border:1px solid green;
      }
    span {
         border:1px solid red;
      }
  </style>
 </head >
 <body>
   <h1>Example DIV vs SPAN</h1>
   <div>DIV is always as wide as the Window/Viewport</div >
   <p>Span is (on the other hand) < span> as wide as it needs to be</span> to show the content.
</p >
 </body>
</html>
```

Box shadow

Box-shadow is a CSS property that lets you add shadow effects around an element's frame. This can be especially useful when designing cards, buttons, or other interactive elements that need to stand out from the rest of the page.The box shadow property takes several values including the horizontal offset, vertical offset, blur, spread radius (optional), and color.

The box-shadow property has the following syntax

```
box-shadow: [horizontal offset] [vertical offset] [blur radius] [spread radius] [color];
```

- The horizontal offset sets the horizontal distance of the shadow from the element. A positive value will move the shadow to the right of the element, while a negative value will move it to the left.
- The vertical offset sets the vertical distance of the shadow from the element. A positive value will move the shadow below the element, while a negative value will move it above.
- The blur radius determines the blur level of the shadow. A larger value will make the shadow more blurred and diffuse, while a smaller value will make it sharper.
- The spread radius (optional) determines the size of the shadow. A positive value will expand the shadow, while a negative value will shrink it.
- The color value sets the color of the shadow. This can be specified using a named color, a hexadecimal value, a RGBA value, or a HSLA value

You can apply a simple box shadow to the element like following:

```
.card {
      box-shadow: 2px 2px 4px #888;
}
```

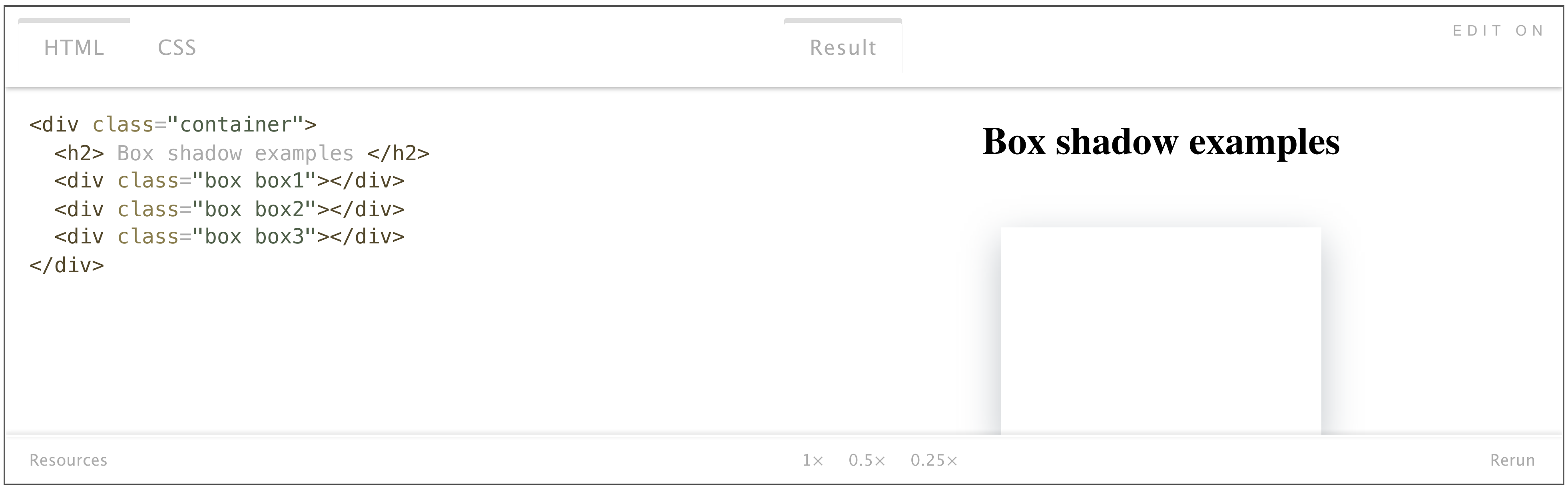
You can also apply multiple box shadows to an element by separating them with commas. Here's an example

```
.card {
      box-shadow: 2px 2px 4px #888, -2px -2px 4px #888;
}
```

You can create an inset box shadow by adding the keyword "inset" before the values, just like this:

```
.card {
      box-shadow: inset 2px 2px 4px #888;
}
```

Here are some examples:



```
<div class="container">
  <h2> Box shadow examples </h2>
  <div class="box box1"></div>
  <div class="box box2"></div>
  <div class="box box3"></div>
</div>
```

Box shadow examples

Measurements

CSS allows us to use different types of measurement, both absolute and relative, to design our websites. Some of the most commonly used measurements are pixels, rems, ems, and percent. You will use them, and maybe some others, that you can find below:

Pixels

The size of a pixel is relative to its physical size on the display and therefore can vary. **One pixel equals the width/height of a single dot on the screen**.

```
.my-class {
      margin: 15px;
}
```

Rems

Rem stands for "root em" and unlike the px unit, which is an absolute measurement, rem is relative to the root element of the document, which is usually the <html> element. By default, most browsers set the size of the root element to 16 pixels, which means that 1 rem equals 16 pixels. However, this base size can be changed. Let’s see an example:

```
html {
 font-size: 18px; /* The base font size is set to 18 pixels */
}

p {
 font-size: 1.25rem; /* 1.25 times the base size, so 1.25rem x 18px = 22.5px */
}
```

Ems

One *em* is equal to the **current font size** and is one of the more useful measurements for CSS. This is because it describes relative dimensions and makes imagining its effects thereby easier.

```
.my-class {
      font-size: 3em;
}
```

Percentage

100% equals 100% of the current font size or, if not relating to a font, is relative to the size of the container of the property being accessed.

```
.myClass {
      width: 120%;
}
```

Viewports

These units are relative to 1% of the width of the viewport(**vw**) or 1% of the height of the viewport(**vh**)

```
.class-name {
      width: 10vw;
      height: 50vh;
}
```

Text Formatting

Fonts and Typography

Primarily, you can style font properties using these four CSS attributes to adapt how your text looks on your website:

1. **family**
2. **size**
3. **style**
4. **weight**

Font-family

You will use the font-family property to define the font used in the text. You can even list multiple fonts to be used in **order of preference from left to right**, so that styling can fall back(default) should your user not have the preferred font installed.
This looks something like this:

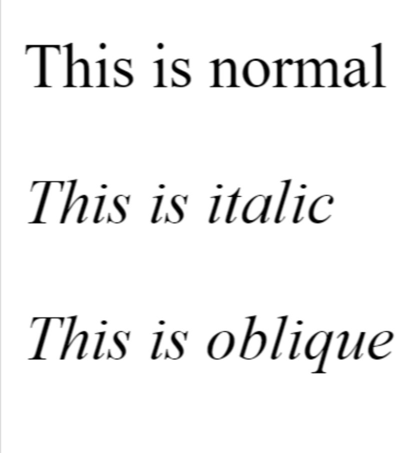
```
h2 {
      font-family: Arial, Verdana, serif;
}
```

Please note that if a font name is made up of two or more words, you must enclose the name in quotation marks.

Font-style

Using the font-style property, you can choose to display a text normally, in italics, or obliquely. The examples below create **three ids (normal, italic, and oblique)** that in turn create these effects.

```
#normal {
      font-style:normal;
}
#italic {
      font-style:italic;
}
#oblique {
      font-style:oblique;
}
```



This is normal

This is italic

This is oblique

Font-size

There are numerous ways you can change a font’s size. But this comes down to two main types: fixed and relative. A **fixed setting** looks like the following rule, which sets the **default paragraph font size to 14 points** :

```
p {
      font-size: 14pt;
}
```

Alternatively, you may wish to work with the current default font size, using it to style various types of text such as headings. In the following rules, relative sizes of some headers are defined, with the **<h4>** tag starting off 20 percent bigger than the default, and with each greater size another 40 percent larger than the previous one:

```
h1 {
      font-size:240%;
}
h2 {
      font-size:200%;
}
h3 {
      font-size:160%;
}
h4 {
      font-size:120%;
}
```

To take a look at more on font-sizes, take a look at https://www.w3schools.com/css/css_font_size.asp

Font-weight

Using the **font-weight** property, you can choose if the text should be displayed as **bold**. It supports a number of values, but the main ones you will use are likely to be normal and bold, like this:

```
.class-name {
      font-weight: bold;
}
```

Further Managing Text Styles

No matter the font you use, you can adapt how the text displays by altering the attributes, setting its **decoration, spacing, and alignment**. The separation between those attributes is not always clear. For example, effects such as italics or bold text are achieved via the font-style and **font-weight** properties, while for example underlining is achieved through the text-decoration property.

Alignment

You can choose between four types of **text alignment** in CSS: **left, right, center, and justify(causes the element to fill the entire width)**.

```
h1 {
      text-align: justify;
}
```

Decoration

Using the **text-decoration** property, different pre-defined attributes can be set. They include effects such as **underline, line-through, overline, and blink.** If you want to apply an underline to a text via its class, it can be done this way:

```
.class-name {
      text-decoration: underline;
}
```

Some more info on text-decoration:
https://www.w3schools.com/css/css_text_decoration.asp

Spacing

There are different ways and units of measurement to **modify line, word, and letter spacing**.

```
p {
     word-spacing :1em;
   letter-spacing:1em;
   line-height :150% ;
}
```

Shadow

You can use the **text-shadow** property to create something called a **drop shadow.** This is akin to a dark version of the word behind and slightly to the side of the text. You can also use it to create an embossed effect by setting the color of the shadow to be slightly lighter than the text. Due to its ability to achieve different effects, the value contains up to three lengths and a color for the drop shadow, making it quite complex.

1. The **first length** indicates **how far** to the **left or right** the shadow should fall.
2. The **second value** indicates the **distance** to the **top or bottom** that the shadow should fall.
3. The **third value** is **optional** and specifies the amount of **blur** that should be applied to the drop shadow.
4. The **fourth value** is the **color of the drop shadow**.

```
p.one {
   background-color: #eeeeee;
   color: #666666 ;
   text-shadow: 1px 1px 0px  #000000;
}
```

Transformation

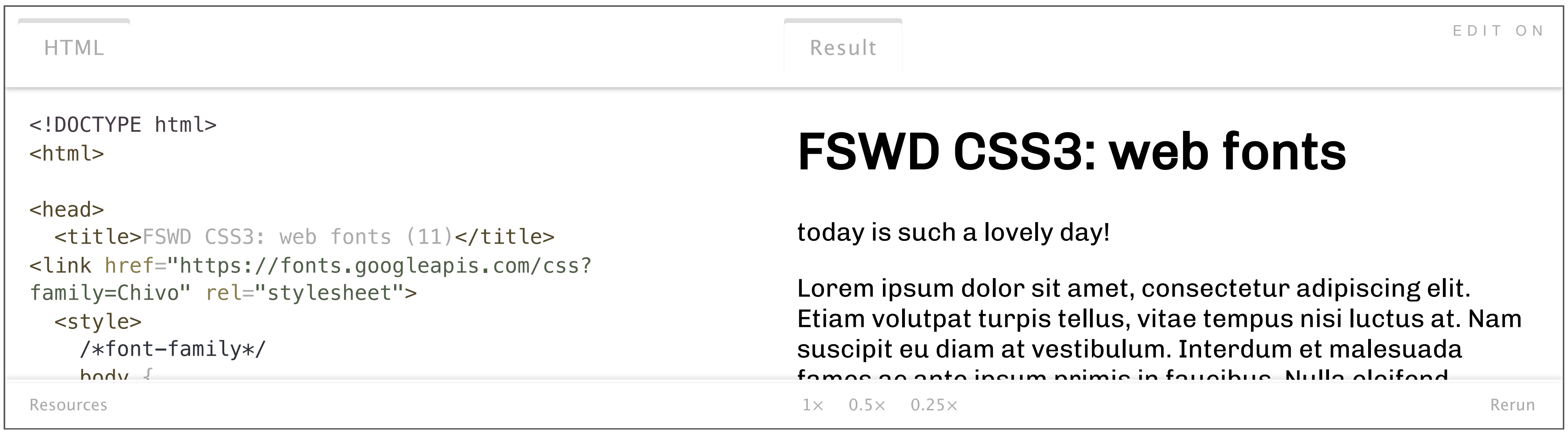
There are four properties available for transforming your text: **none, capitalize, uppercase, and lowercase**. The following rule creates a class called upper that will ensure that all text is displayed in uppercase when it is used:

```
.upper {
      text-transform: uppercase;
}
```

Indenting

Using the text-indent property, you can **indent the first line of a block of text** by a specified amount. The following rule indents the first line of every paragraph by 20 pixels:

```
p {
      text-indent: 20px;
}
```



```
<!DOCTYPE html>
<html>

<head>
  <title>FSWD CSS3: web fonts (11)</title>
<link href="https://fonts.googleapis.com/css?
family=Chivo" rel="stylesheet">
  <style>
    /*font-family*/
    body {
```

FSWD CSS3: web fonts

today is such a lovely day!

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Etiam volutpat turpis tellus, vitae tempus nisi luctus at. Nam suscipit eu diam at vestibulum. Interdum et malesuada fames ac ante ipsum primis in faucibus. Nulla eleifend

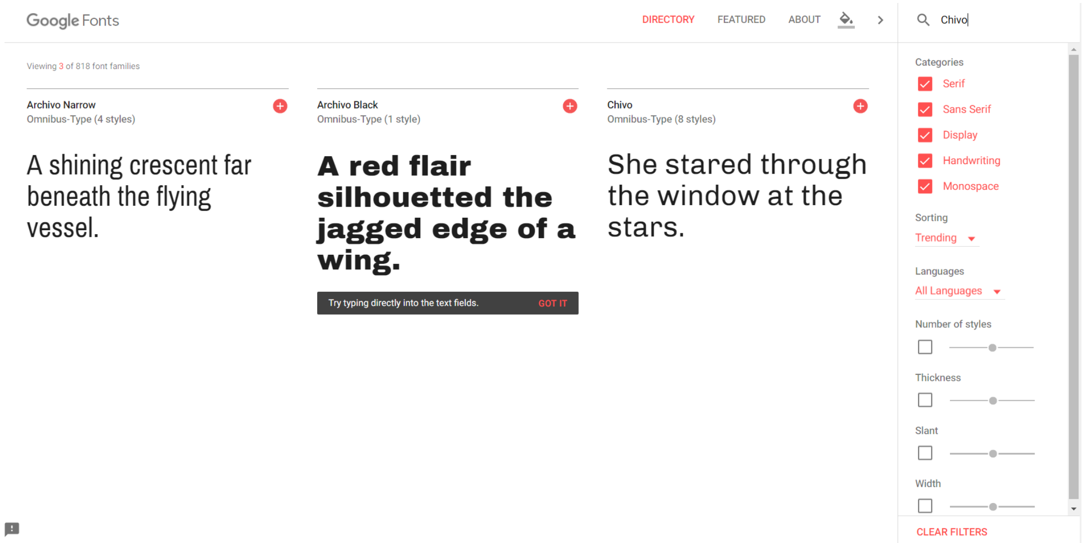
Web Fonts

Luckily, we can do more than just use the default fonts when designing a website. Asides from the obvious issues with copyright, the following ways can help you use more interesting fonts in your projects:
@font-face allows you to use a font by having you **specify a path to a copy of the font**. It will then be downloaded if it is not on the user's machine. Note that this will both require you to have license to use the font, as well as that it may slow down your visitor on their first visit when the font is being downloaded. You add the font to your stylesheet using the @font-face rule, as follows:

```
@font-face {
    font-family: 'CodingApocalypseFont';
    src: url('fonts/codingapocalypse.eot');
}
```

font-family specifies the name of the font and the **src** attribute specifies the path to the font (in this case as a relative path while still requiring you to use the **url()** declaration).

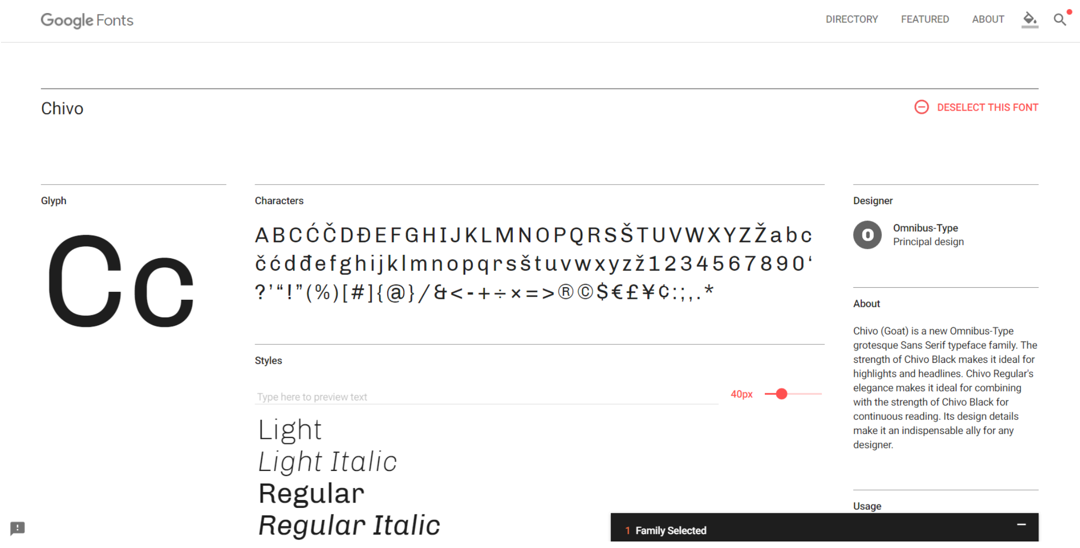
Google also provides open source fonts. Rather than adding the @font-face rule to your own style sheet, you link to a CSS file and font files on their servers: https://fonts.google.com/
Let's see how we can include Google fonts to our web page.



A shining crescent far beneath the flying vessel.

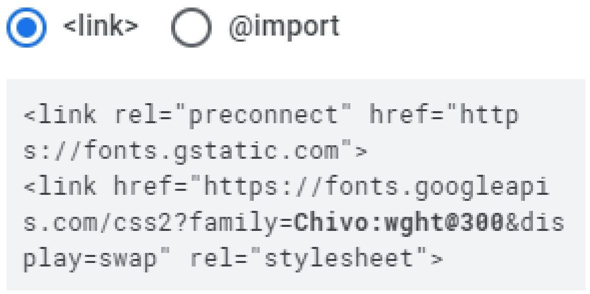
A red flair silhouetted the jagged edge of a wing.

She stared through the window at the stars.



Chivo (Goat) is a new Omnibus-Type grotesque Sans Serif typeface family. The strength of Chivo Black makes it ideal for highlights and headlines. Chivo Regular's elegance makes it ideal for combining with the strength of Chivo Black for continuous reading. Its design details make it an indispensable ally for any designer.

1. Go to https://fonts.google.com/
2. In the top right corner write "Chivo".
3. Click on the Chivo font (the last one on the right side)
4. Click on the + Select this style from one of the styles that you prefer.
5. Next, click on the icon panel on the top-right of your screen
6. Copy the <link>

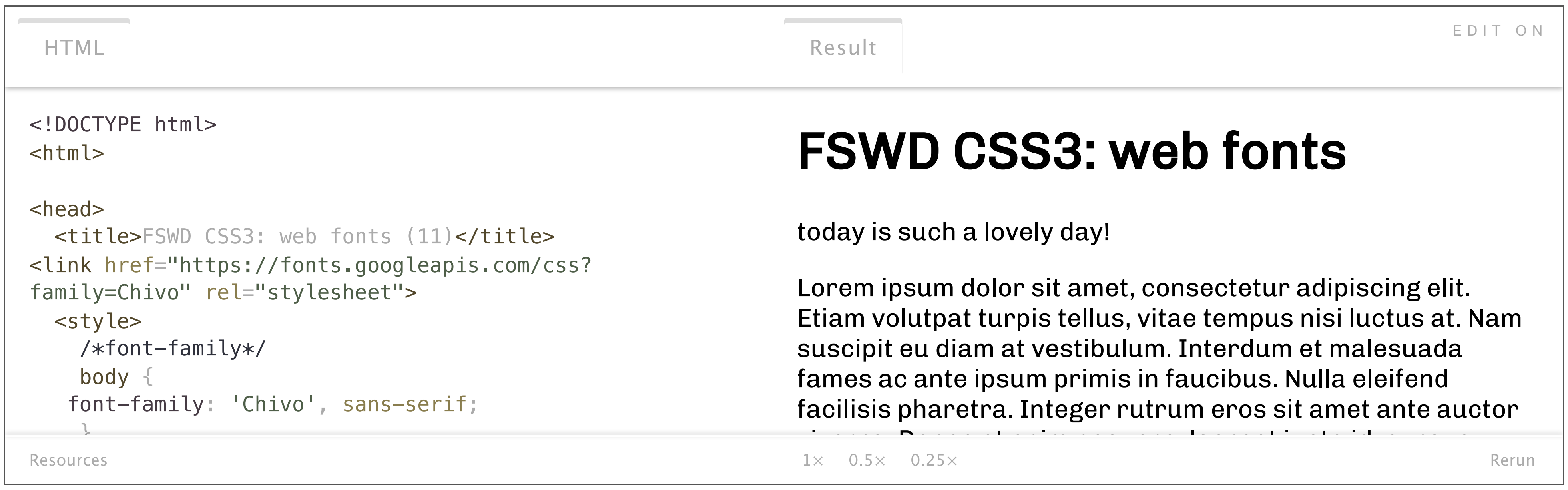


● <link> ○ @import

```
<link rel="preconnect" href="http
s://fonts.gstatic.com">
<link href="https://fonts.googleapi
s.com/css2?family=Chivo:wght@300&dis
play=swap" rel="stylesheet">
```

7. Simply paste this code in your HTML document inside the HEAD section
8. Apply the font to the body section, so it can affect the whole page

```
body {
      font-family: 'Chivo', sans-serif;
}
```



```
<!DOCTYPE html>
<html>

<head>
  <title>FSWD CSS3: web fonts (11)</title>
<link href="https://fonts.googleapis.com/css?
family=Chivo" rel="stylesheet">
  <style>
    /*font-family*/
    body {
   font-family: 'Chivo', sans-serif;
```

FSWD CSS3: web fonts

today is such a lovely day!

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Etiam volutpat turpis tellus, vitae tempus nisi luctus at. Nam suscipit eu diam at vestibulum. Interdum et malesuada fames ac ante ipsum primis in faucibus. Nulla eleifend facilisis pharetra. Integer rutrum eros sit amet ante auctor

Colors

Before we learn how to assign colors with css, let's see how the color are available to be used:

Color name:

red	blue	aqua	orange	yellow	lime

Hex number

#FF0000	#0000FF	#00FFFF	#FF8000	#FFF00	#00FF00

*css accepts the alphanumeric code in small capital too.

Rgb number

rgb(255,0,0)	rgb(0,0,255)	rgb(0,255,255)	rgb(255,128,0)	rgb(255,255,0)	rgb(0,255,0)

If you would like to add some transparency to the element, the option rgb allows you to do that using the "alpha channel" which specifies the opacity of a color. You simply need to add a new value from 0.0 (totally transparent) to 1.0(totally opaque) at the end of the sequence, i.e.:

- color: rgb(0,0,255, 0.5) /* blue with 50% opacity */
- background-color: rgb(255, 128, 0, .8) /* orange with 20% transparency */

Font-color

In order to change the font color, the attribute **color** must be used, and a value that can be chosen from the three options above.

```
.class-name {
      color: #f700ff;
}
```

Background-color

To assign a color to the background the syntax will be **background-color** and a value must be given.

```
.class-name {
      background-color: rgb(255,128,0);
}
```

There are many tools on the internet that can help you find the ideal colors. Just search "color pallet" on your browser and many will show:

https://colorhunt.co/

https://coolors.co/

https://colorpalettes.net/

Last modified: Friday, 19 April 2024, 11:51 AM

◄ CSS | Day 1 | Exercises

Jump to...

CSS | Day 2 | Quiz ►