Front-End v23.0

Back to 'Day 4 | Pre-work'

CSS | Day 4 | Pre-work

CSS

Day 4

Table of contents:

Backgrounds

Background-image

Gradient

CSS3 introduced the ability to specify a gradient for the background of a box. The gradient is created using the background-image property.
Since it is not supported by all browsers, it is possible to specify a background image for the box first (which would represent the gradient) and then provide the CSS alternatives for browsers that support gradients.

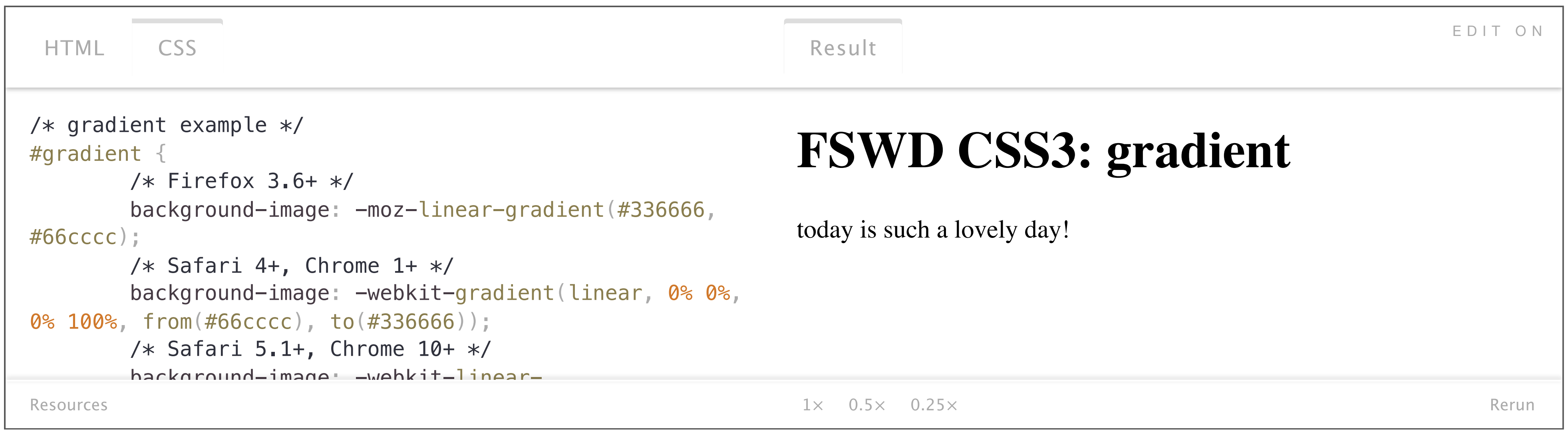
We can create a gradient by using the linear-gradient property:

```
.classname{
   width: 50vw;
   height: 100vh;
   background: purple; /*fall-back if gradients are not supported by browser*/
   background-image: linear-gradient(green,red  );
}
```

Every web browser has an HTML/CSS engine (sometimes called web rendering engine or web layout engine).
The **-webkit-** CSS property is used by WebKit HTML/CSS engine in Apple Safari web browser, or Blink engine which is a forked version of the WebKit engine in Google Chrome. Some non-WebKit browsers also supports -webkit-appearance property for compatibility reasons.
In order to create a linear gradient, we need to specify two colors,we can also do it as follows (without the default fall-back background):

```
#gradient {
  /* Firefox 3.6+ */
   background-image: -moz-linear-gradient(#336666, #66cccc);
  /* Safari 4+, Chrome 1+ */
   background-image: -webkit-gradient(linear, 0% 0%, 0% 100%, from(#66cccc), to(#336666));
  /* Safari 5.1+, Chrome 10+ */
   background-image: -webkit-linear-gradient(#336666, #66cccc);
  /* Opera 11.10+ */
   background-image: -o-linear-gradient(#336666, #66cccc);
  height: 150px;
  width: 300px ;
}
```

From the example above you can see that for each browser, we define the linear gradient properties in different ways (see comments).



```
/* gradient example */
#gradient {
        /* Firefox 3.6+ */
        background-image: -moz-linear-gradient(#336666,
#66cccc);
        /* Safari 4+, Chrome 1+ */
        background-image: -webkit-gradient(linear, 0% 0%,
0% 100%, from(#66cccc), to(#336666));
        /* Safari 5.1+, Chrome 10+ */
        background-image: -webkit-linear-
```

FSWD CSS3: gradient

today is such a lovely day!

If you would like to read up more on gradients, here a link: https://www.w3schools.com/css/css3_gradients.asp

Images

The background-image property will place an image in the background of any HTML element you apply it to. This could be the entire page or a specific element as a div, for example. By default, a background image will repeat itself to fill the entire box.

```
body {
    background-image: url("https://images.pexels.com/photos/2363347/pexels-photo-
2363347.jpeg?auto=compress&cs=tinysrgb&dpr=2&h=750&w=1260" );
}
```

The path to the image follows the letters url and is put inside parentheses and quotes.
In this example, since we apply it to the entire body of the page, you can see a background image being applied to an entire page.

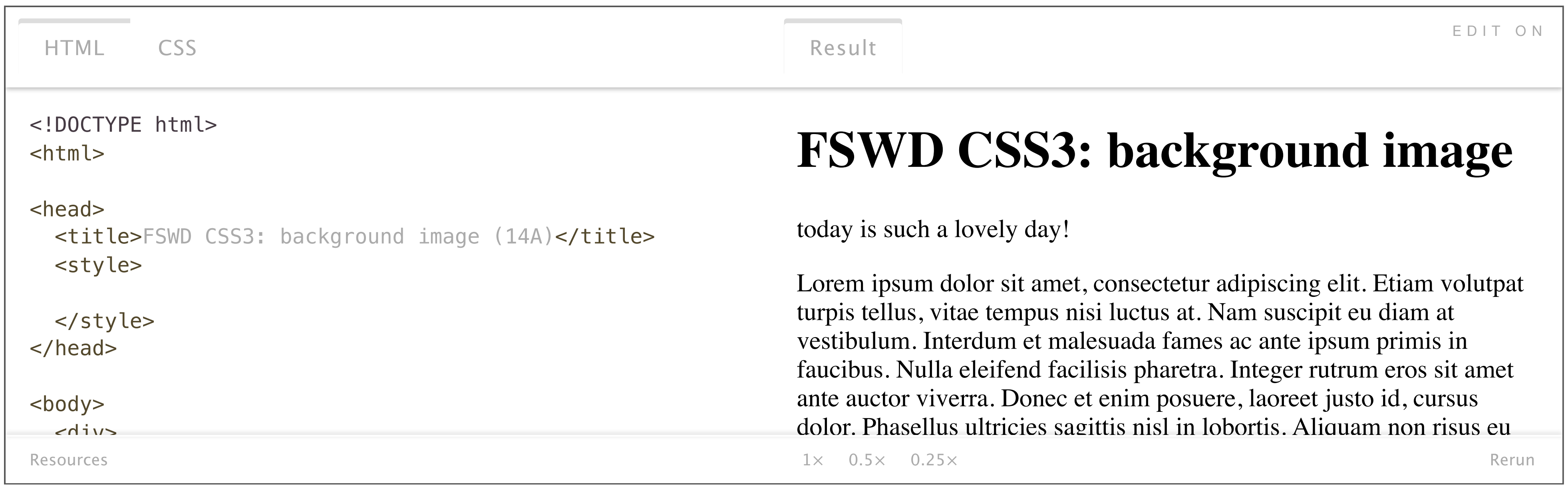
Background-repeat

The background-repeat property can have five values:

1. **repeat-x** - The image is **repeated horizontally only** (as shown in the first example on the left).
2. **repeat-y** - The image is **repeated vertically only**
3. **no-repeat** - The image is only **shown once**.
4. **fixed** - The background image **stays in the same position** on the page.
5. **scroll** - The background image **moves up and down** as the user scrolls up and down the page.

```
body {
background-image: url("https://images.pexels.com/photos/2363347/pexels-photo-2363347.jpeg?
auto=compress&cs=tinysrgb&dpr=2&h=750&w=1260" );
background-repeat: repeat-x;
}
```

The example above will repeat the background horizontally.

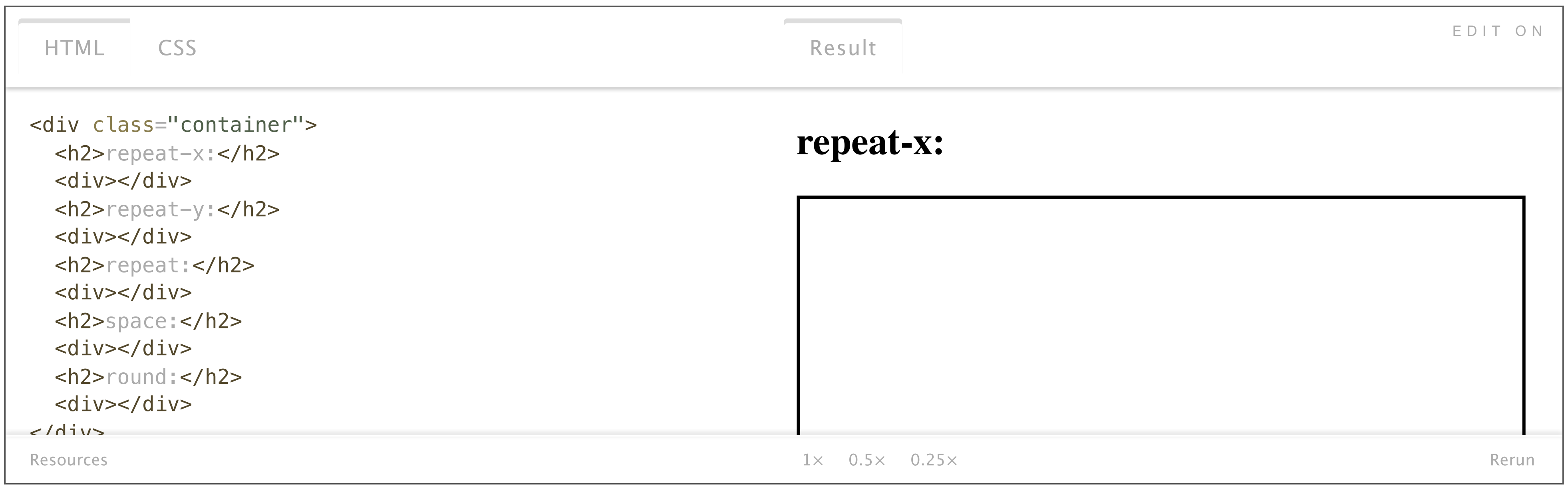


```
<!DOCTYPE html>
<html>

<head>
  <title>FSWD CSS3: background image (14A)</title>
  <style>

  </style>
</head>

<body>
  <div>
```

FSWD CSS3: background image

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 viverra. Donec et enim posuere, laoreet justo id, cursus dolor. Phasellus ultricies sagittis nisl in lobortis. Aliquam non risus eu



```
<div class="container">
  <h2>repeat-x:</h2>
  <div></div>
  <h2>repeat-y:</h2>
  <div></div>
  <h2>repeat:</h2>
  <div></div>
  <h2>space:</h2>
  <div></div>
  <h2>round:</h2>
  <div></div>
</div>
```

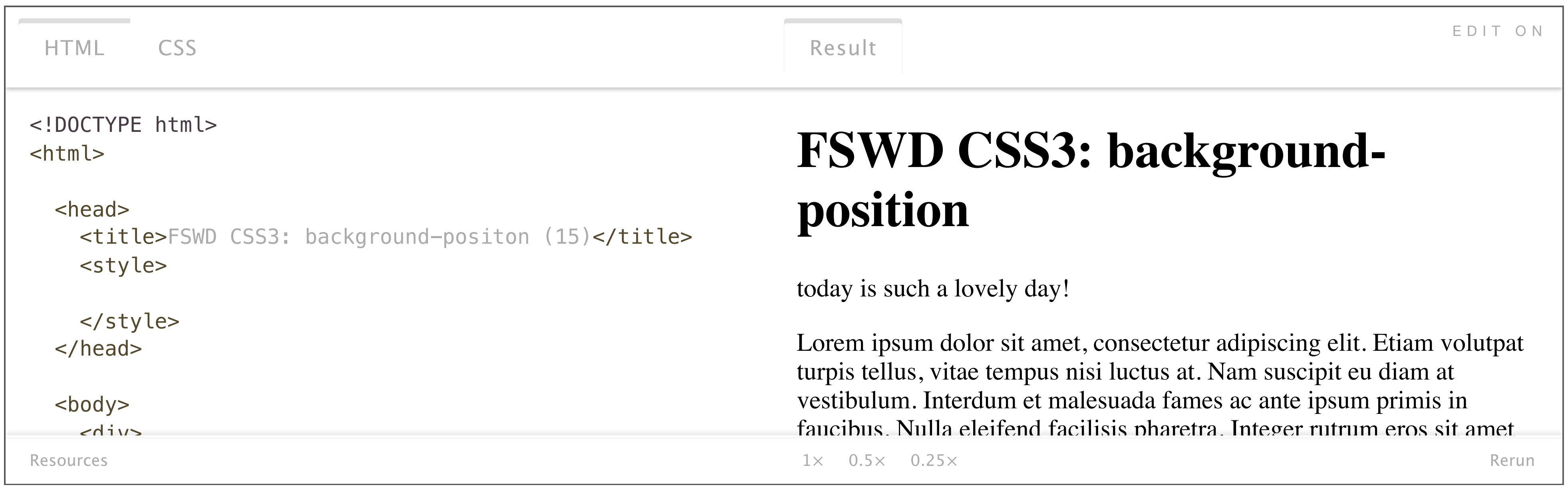
repeat-x:

Background-position

If you set the attribute background-repeat to “no-repeat”, you can use the **background-position** property to specify where in the browser window the background image will be placed. This property usually has a pair of values, representing the horizontal and vertical values respectively.

```
body {
background-image: url("https://images.pexels.com/photos/2363347/pexels-photo-2363347.jpeg?
auto=compress&cs=tinysrgb&dpr=2&h=750&w=1260" );
background-repeat: no-repeat;
background-position: center top;
}
```

This will position the background to the top of the page and it will be centered. We also define the background-repeat property so we avoid repetition of the image.



```
<!DOCTYPE html>
<html>

  <head>
    <title>FSWD CSS3: background-positon (15)</title>
    <style>

    </style>
  </head>

  <body>
    <div>
```

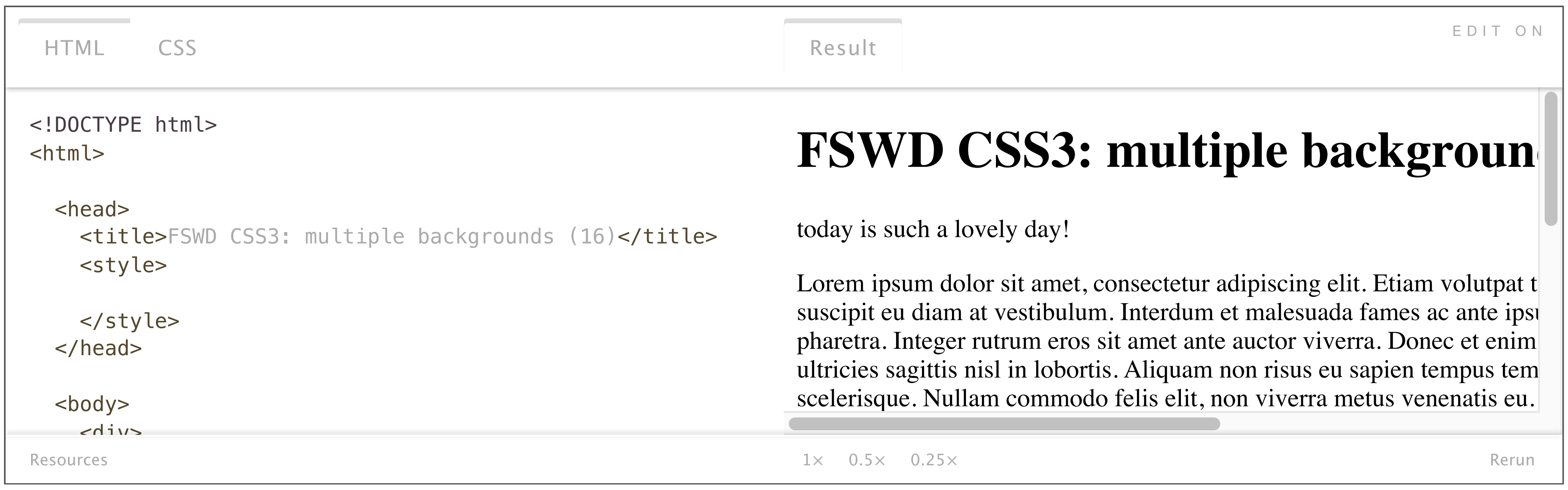
FSWD CSS3: background-position

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

Multiple Backgrounds

If you want to use multiple background images, they need to be specified using a comma-separated list of values for the background-image property. Each of them will generate a separate 'background layer'.



```
<!DOCTYPE html>
<html>

  <head>
    <title>FSWD CSS3: multiple backgrounds (16)</title>
    <style>

    </style>
  </head>

  <body>
    <div>
```

FSWD CSS3: multiple backgroun

today is such a lovely day!

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Etiam volutpat t
suscipit eu diam at vestibulum. Interdum et malesuada fames ac ante ipsu
pharetra. Integer rutrum eros sit amet ante auctor viverra. Donec et enim
ultricies sagittis nisl in lobortis. Aliquam non risus eu sapien tempus tem
scelerisque. Nullam commodo felis elit, non viverra metus venenatis eu.

```
body {
width: 800px;
height  : 430px;
background-image: url("https://images.pexels.com/photos/2363347/pexels-photo-2363347.jpeg?
auto=compress&cs=tinysrgb&dpr=2&h=750&w=1260"  ),
url("https://images.pexels.com/photos/6199961/pexels-photo-6199961.jpeg?
auto=compress&cs=tinysrgb&dpr=1&w=500" );
 background-position: center bottom, left top;
background-repeat: no-repeat;
}
```

The first value in the list represents the top layer, each subsequent layer will be rendered behind the previous ones in order. It can be used e.g. when you want to play safe just in case if the primary background image is no longer available, or if you want to display multiple background images simultaneously by giving them different positions and sizes so they fit next to each other.

Shorthand

The background property has a shorthand way that all above properties could be used in just one line of css:

```
 background: #ffffff url("img/image_name.png") no-repeat right top;
```

Of course the order is important so make sure u follow:

background-color / background-image / repeat / attachment* / position

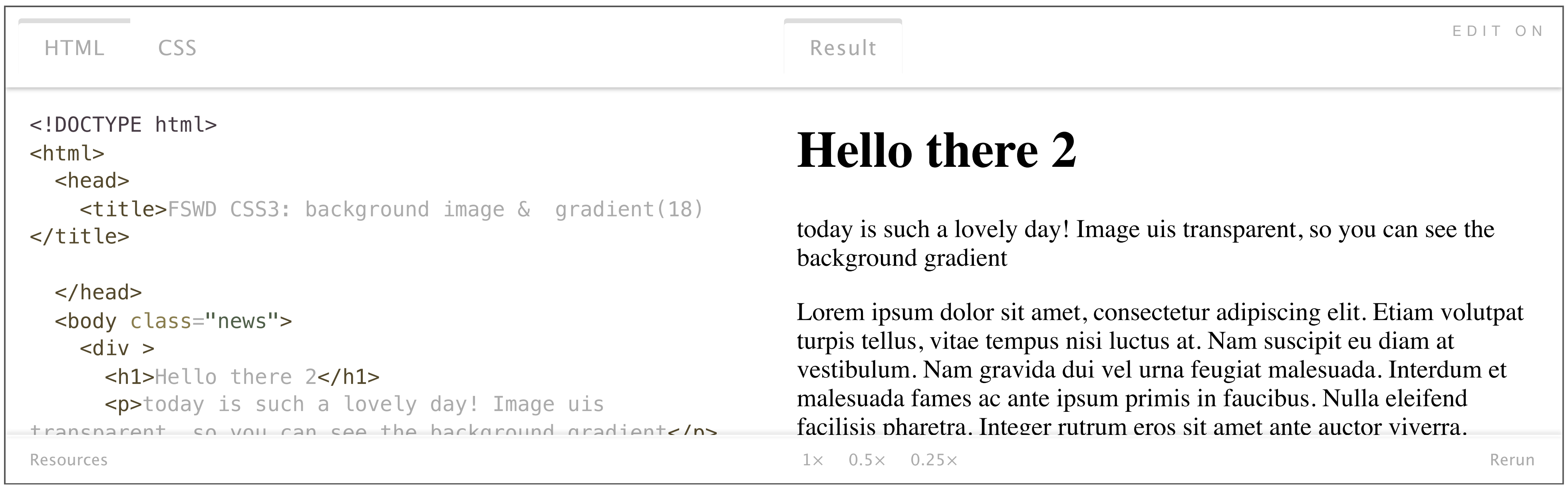
**If one of the values is not given, as long as the others are in the right order, it will work properly.*

Graphics

Background-repeat: While we can set our **background-repeat** to **no-repeat**, and its size to "**cover"(background-size: cover;)** for a regular full-sized background, there is more to this attribute than immediately meets the eye.

If we, for example, set it to **repeat-x**, the background-image will be repeated horizontally along the x-axis (horizontal) of your page. This is perfect for adding things like a graphical banner or decorative borders to your page.
Conversely, **repeat-y**, lets us repeat the image along the y-axis (vertical), and can give us a graphic sidebar to our page. We have already spoken about background-repeat. If we use the **round** value, the background-images are not clipped, but will be distorted to achieve this. The **space** value, in comparison, will introduce space between tiled copies of the image to prevent it from being clipped, thus preserving the image's aspect ratio.

Further, we can use opaque color codes to darken an image, or create transparent overlays, like this:



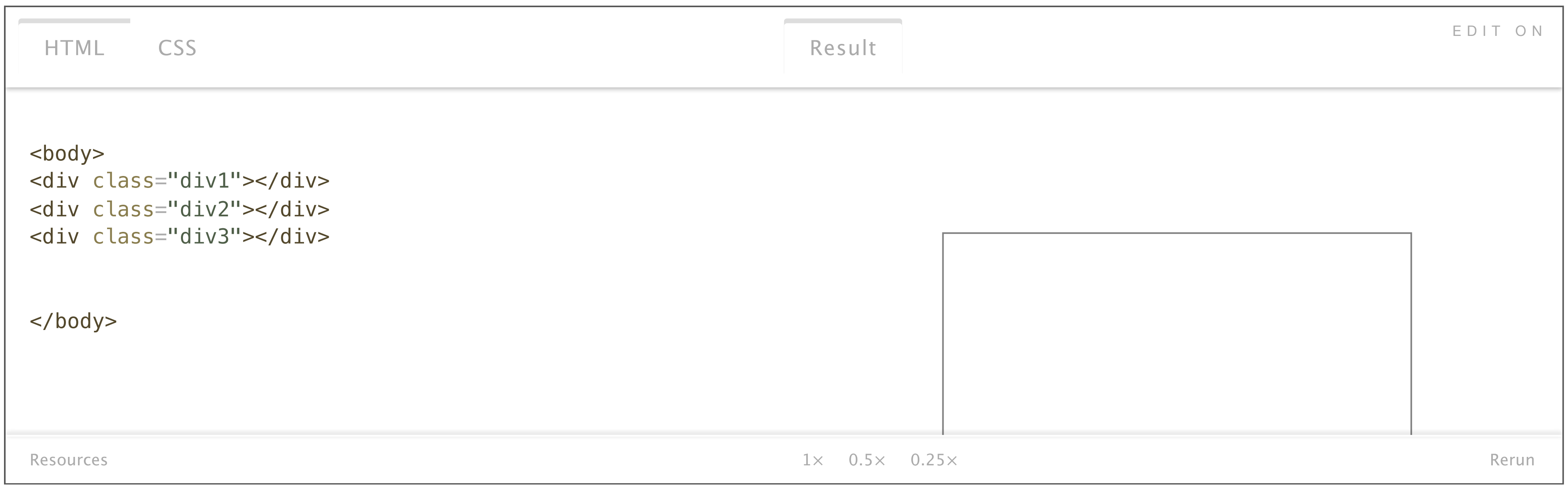
```html
<!DOCTYPE html>
<html>
  <head>
    <title>FSWD CSS3: background image &  gradient(18)
</title>

  </head>
  <body class="news">
    <div >
      <h1>Hello there 2</h1>
      <p>today is such a lovely day! Image uis
transparent, so you can see the background gradient</p>
```

Hello there 2

today is such a lovely day! Image uis transparent, so you can see the background gradient

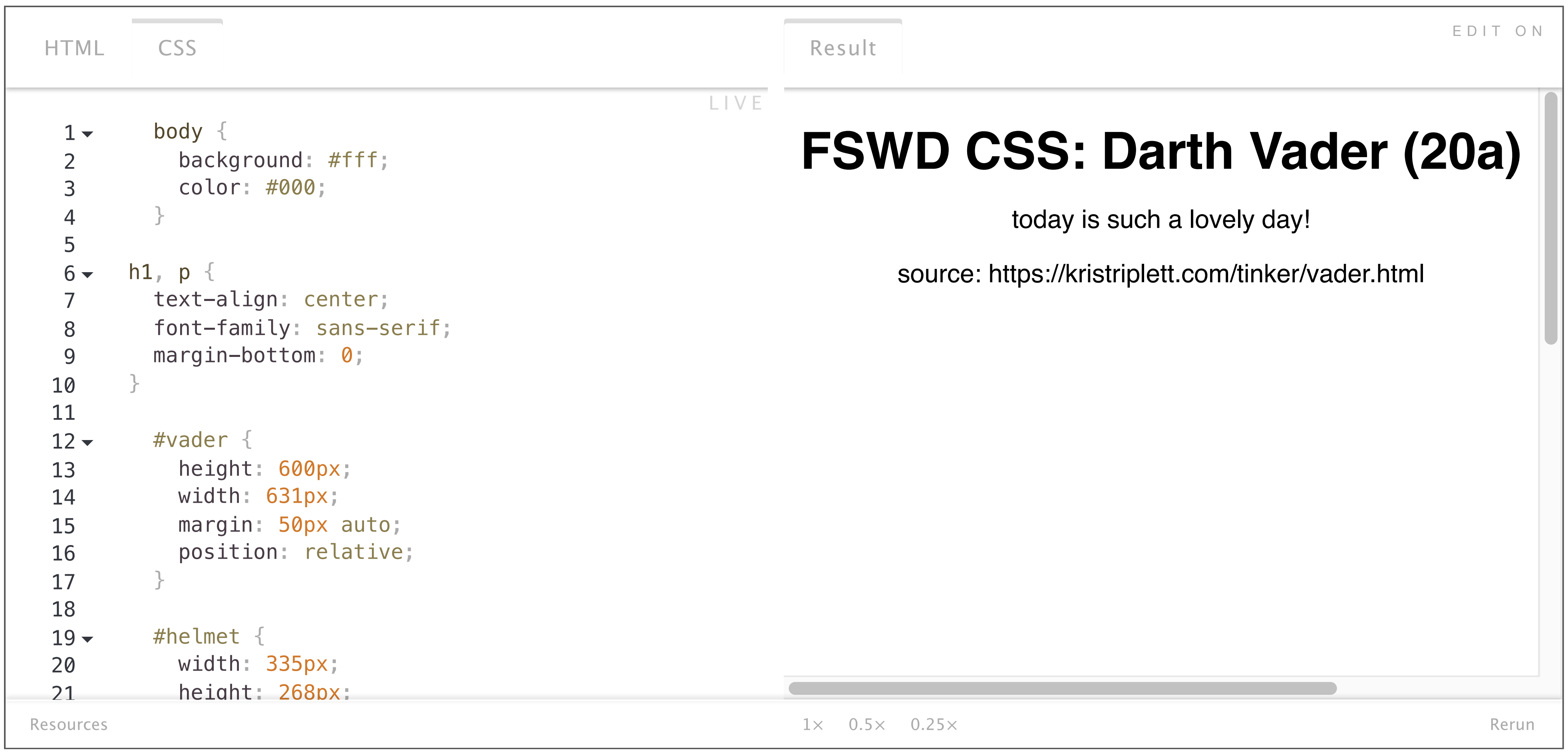
Lorem ipsum dolor sit amet, consectetur adipiscing elit. Etiam volutpat turpis tellus, vitae tempus nisi luctus at. Nam suscipit eu diam at vestibulum. Nam gravida dui vel urna feugiat malesuada. Interdum et malesuada fames ac ante ipsum primis in faucibus. Nulla eleifend facilisis pharetra. Integer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

We can manipulate the background-sizes, by using different values:



```html
<body>
<div class="div1"></div>
<div class="div2"></div>
<div class="div3"></div>


</body>
```

For further information on background-sizes, take a look at: https://www.w3schools.com/cssref/css3_pr_background-size.asp



```css
  body {
    background: #fff;
    color: #000;
  }

h1, p {
  text-align: center;
  font-family: sans-serif;
  margin-bottom: 0;
}

  #vader {
    height: 600px;
    width: 631px;
    margin: 50px auto;
    position: relative;
  }

  #helmet {
    width: 335px;
    height: 268px;
```

FSWD CSS: Darth Vader (20a)

today is such a lovely day!

source: https://kristriplett.com/tinker/vader.html

CSS3: Transition

Generally speaking, CSS3 transitions allow you to change a property value smoothly from one value to another over a predefined time.
For a transition effect, you must specify two things

- The **CSS property** you want to add an effect to
- The **duration** of the effect

So let's say for example you have a div with predefined dimensions and want to increase its height and width if the user triggers a :hover state. This is how it would work:

The original div, as defined by the developer:

```
div {
  width: 150px;
  height : 150px;
  background: lightgrey;
  border-radius: 5px;
  transition: width 2s;
}
```

The hover-state that contains the altered dimensions:

```
div:hover {
  width: 300px;
  height: 300px ;
}
```

Additionally, you can alter multiple values at once and specify the speed curve of the transition with specific keywords for values. Using this alternative, your original div would look like this:

```
div {
  transition: width 2s, height 4s ;
  transition-timing-function: linear;
}
```

Alternatively, you can also use these keywords for values:

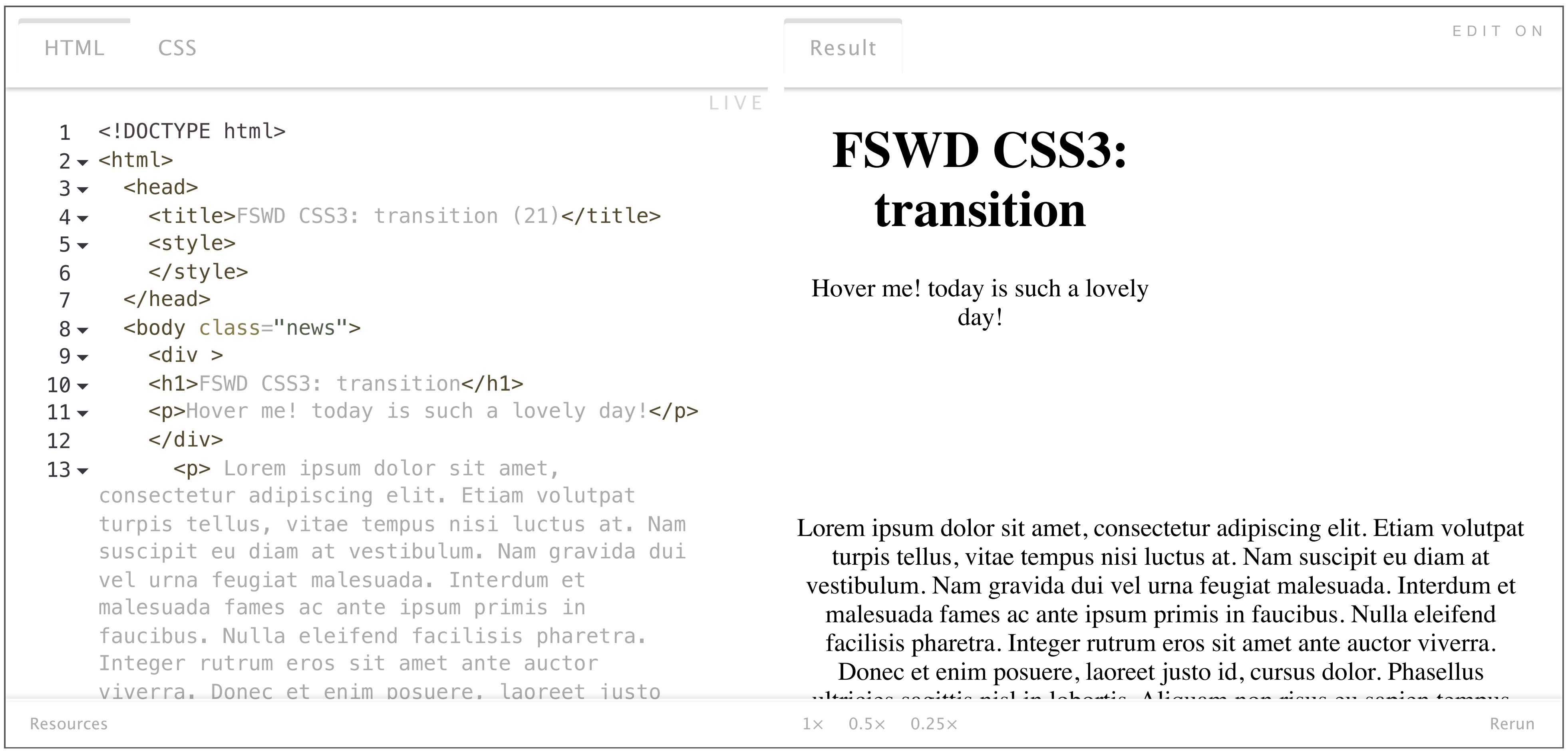
- **ease** - specifies a transition effect with a slow start, then fast, then end slowly (this is the default)
- **ease-in** - specifies a transition effect with a slow start
- **ease-out** - specifies a transition effect with a slow end
- **ease-in-out** - specifies a transition effect with a slow start and end
- **cubic-bezier(n,n,n,n)** - lets you define your own values in a cubic-bezier function

You can also delay the transition effect by adding a "**transition-delay**" attribute with a value in seconds, such as "**3s**".
In total, using longhand, these transition properties could look like this:

```
div {
  transition-property: height;
  transition-duration: 4s;
  transition-timing-function: ease-in;
  transition-delay: 2s;
}
```

Or, using shorthand, like this:

```
div {
  transition: height 4s ease-in 2s ;
}
```



```
<!DOCTYPE html>
<html>
  <head>
    <title>FSWD CSS3: transition (21)</title>
    <style>
    </style>
  </head>
  <body class="news">
    <div >
    <h1>FSWD CSS3: transition</h1>
    <p>Hover me! today is such a lovely day!</p>
    </div>
      <p> Lorem ipsum dolor sit amet,
consectetur adipiscing elit. Etiam volutpat
turpis tellus, vitae tempus nisi luctus at. Nam
suscipit eu diam at vestibulum. Nam gravida dui
vel urna feugiat malesuada. Interdum et
malesuada fames ac ante ipsum primis in
faucibus. Nulla eleifend facilisis pharetra.
Integer rutrum eros sit amet ante auctor
viverra. Donec et enim posuere, laoreet justo
```

FSWD CSS3: transition

Hover me! today is such a lovely day!

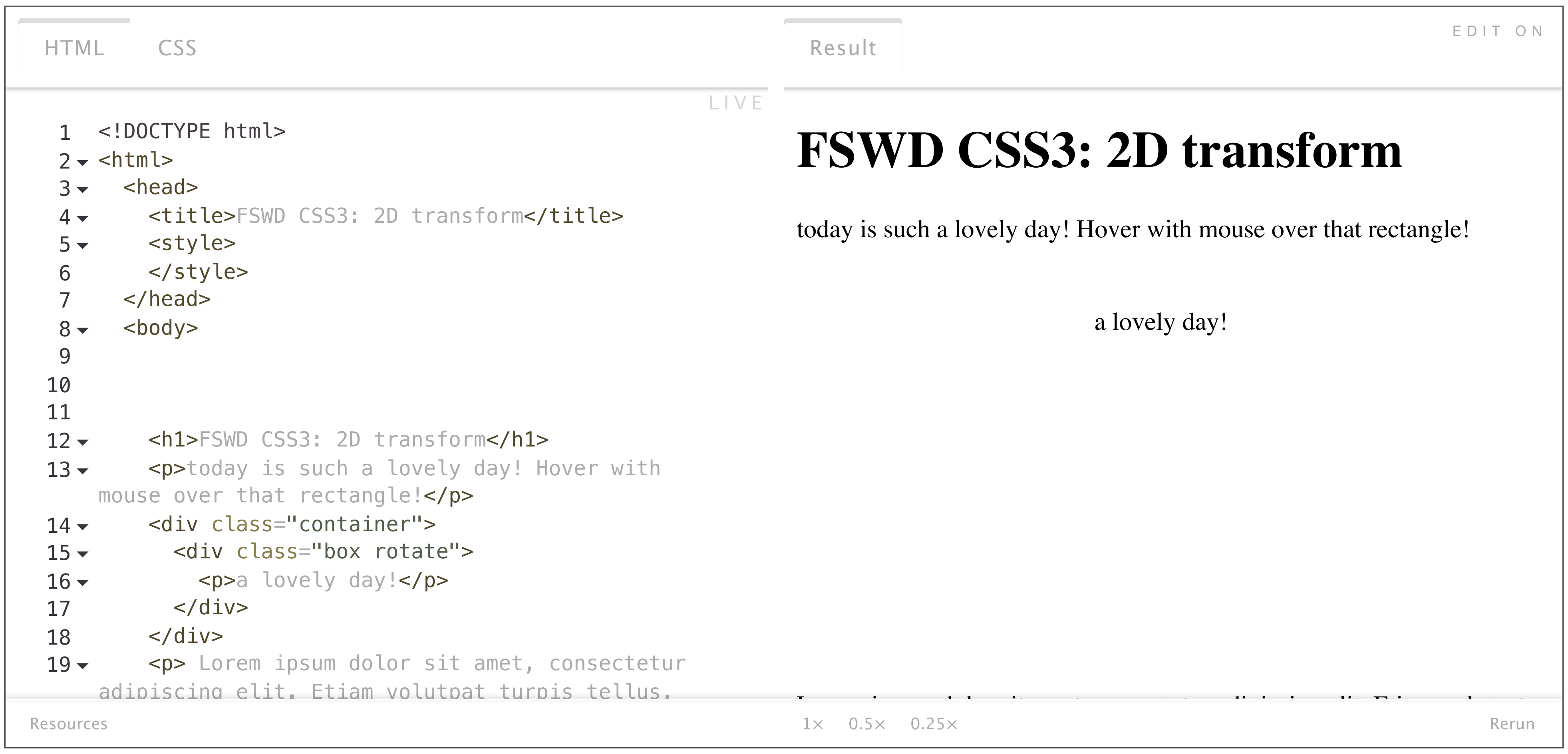
Lorem ipsum dolor sit amet, consectetur adipiscing elit. Etiam volutpat turpis tellus, vitae tempus nisi luctus at. Nam suscipit eu diam at vestibulum. Nam gravida dui vel urna feugiat malesuada. Interdum et malesuada fames ac ante ipsum primis in faucibus. Nulla eleifend facilisis pharetra. Integer rutrum eros sit amet ante auctor viverra. Donec et enim posuere, laoreet justo id, cursus dolor. Phasellus

CSS3: Transform

Closely related to transitions in CSS3 are transformations, which come in versions both for **2D and 3D**.

For 2D-transformations, the following methods are available:

- **translate()** The translate() method moves an element from its current position, given parameters for the x-axis and y-axis, e.g.transform: translate(50px, 100px);
- **rotate()** The rotate() method rotates an element clockwise or counter-clockwise (with a negative prefix) according to a given degree, e.g. transform: rotate(35deg);
- **scale()** The scale() method increases or decreases the size of an element (according to the parameters given for the width and height used as multipliers), e.g. transform: scale(2, 3);
- **skewX()** The skewX() method skews an element along the X-axis by the given angle, similar to the rotate() method.
- **skewY()** The skewY() method skeps an element along the Y-axis.
- **matrix()** The matrix() method combines all the 2D transform methods into one, taking parameters in this order: scaleX(), skewY(), skewX(),scaleY(), translateX(), translateY(). e.g. **transform: matrix(1, -0.3, 0, 1, 0, 0);**
- For 3D-transformations, we add three methods to rotate an element along the X, Y and Z axis:
- **rotateX(), rotateY(), rotateZ()** The rotateY() / rotateY() / rotate() method rotates an element around its relevant axis at a given degree, similar to the 2D rotate() method. It can look like this: **transform: rotateZ(90deg);**



```
<!DOCTYPE html>
<html>
  <head>
    <title>FSWD CSS3: 2D transform</title>
    <style>
    </style>
  </head>
  <body>



    <h1>FSWD CSS3: 2D transform</h1>
    <p>today is such a lovely day! Hover with
mouse over that rectangle!</p>
    <div class="container">
      <div class="box rotate">
        <p>a lovely day!</p>
      </div>
    </div>
    <p> Lorem ipsum dolor sit amet, consectetur
adipiscing elit. Etiam volutpat turpis tellus.
```

FSWD CSS3: 2D transform

today is such a lovely day! Hover with mouse over that rectangle!

a lovely day!

```
<div class="container">
< div class="box rotate"></div >
</ div>
```

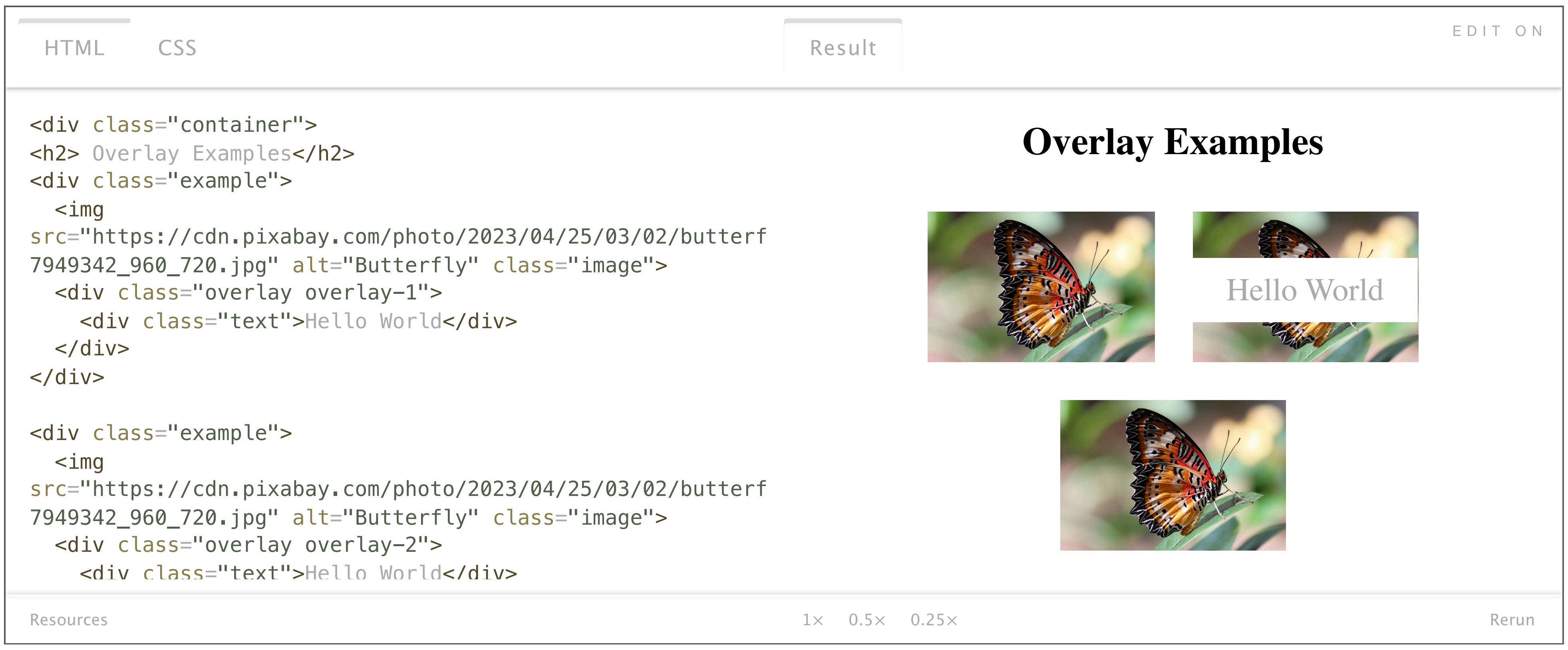
```
.box {
margin: 40px auto;
background  : #005eff;
width: 300px;
height  : 200px;
}

.box:hover {
-webkit-transform : rotate(-360deg);
background: #d1e000;
border-radius  : 100px;
width: 75px;
width : 75px;
}
.rotate {
-webkit-transition : all 0.5s ease-in-out;
}
```

There is also a **matrix3d()** method, accepting a 4*4 matrix of numerical values representing a Cartesian coordinate system for a transformation in 3D space. This method neither has good browser support nor is user-friendly. We will just give you this example of syntax:

```
matrix3d(a1, b1, c1, d1, a2, b2, c2, d2, a3, b3, c3, d3, a4, b4, c4, d4)
```

CSS3: Overlays

As you may have seen it till now, there are many ways to add visual interest to your website, from background images and gradients to graphics and CSS transitions. But one technique that can truly take your design to the next level is overlays. There are different types of overlays that you can use to achieve various effects on your website. Below you can see some examples:

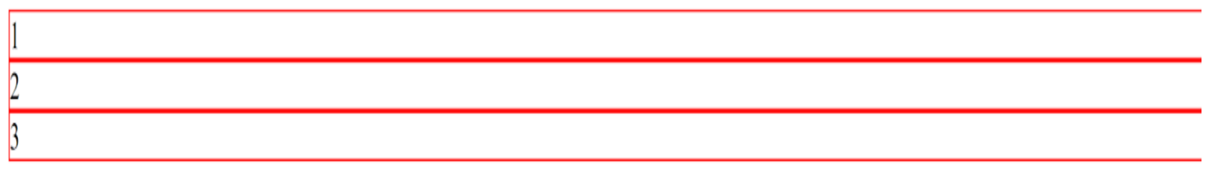


```
<div class="container">
<h2> Overlay Examples</h2>
<div class="example">
  <img
src="https://cdn.pixabay.com/photo/2023/04/25/03/02/butterf
7949342_960_720.jpg" alt="Butterfly" class="image">
  <div class="overlay overlay-1">
    <div class="text">Hello World</div>
  </div>
</div>

<div class="example">
  <img
src="https://cdn.pixabay.com/photo/2023/04/25/03/02/butterf
7949342_960_720.jpg" alt="Butterfly" class="image">
  <div class="overlay overlay-2">
    <div class="text">Hello World</div>
```

Grid Layout

We took a look at Flexbox and its functionality, now let's take it a step further into Grid layout.
What is the difference between Flexbox and Grid layout?
Mainly, Flexbox can be seen as a means to work with a one-dimensional layout, meaning we are working with rows where we place our items along the main-axis, mainly horizontally. Grid, on the other hand, works in a more 2 dimensional way, meaning we need to think in rows and columns.

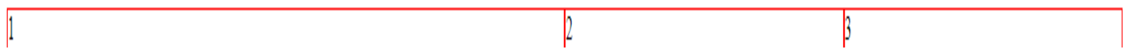
```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta http-equiv="X-UA-Compatible"  content="IE=edge">
  <meta name="viewport"  content="width=device-width, initial-scale=1.0">
  <title>Document</title>
  <style>
    .container{
      display: grid;
    }
    .item{
      border: 1px solid  red ;
    }
  </style>
</head>
<body>
  <div class="container">
    <div class="item" > 1</div>
    <div class="item" > 2</div>
    <div class="item" > 3</div>
  </div>

</body>
</html>
```



As we see in the snippet above, we activate grid layout with **display:grid.**
In this case, we have activated it on the div class "container".
Right now we see no changes.

```
<!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>
        .container{
            display: grid;
            grid-template-columns: 40% 20%   20%;
        }
        .item{
            border: 1px solid  red ;
        }
    </style>
</head>
<body>
    <div class="container">
        <div class="item" > 1</div>
        <div class="item" > 2</div>
        <div class="item" > 3</div>
    </div>

</body>
</html>
```

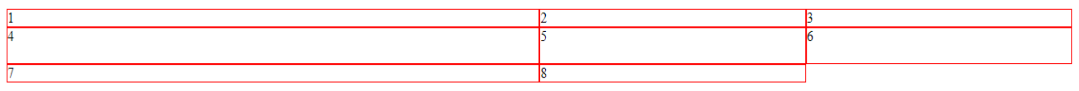


With grid-template-columns we can state what sizes we want to use per row.

```
grid-template-columns: 40% 20% 20% ; /*we can also use auto to fill out the remainder automatically*/
/*or we can use the fraction based method(recommended)*/
grid-template-columns: 2fr 1fr 1fr ;
/*or the repeat fraction based method*/
grid-template-columns: repeat(3, 1fr ); /*first value is how many repeats, second value shows how it should be repeated*/
```

If we fill add some more items, we can see how we can also affect the rows with grid-template-rows:

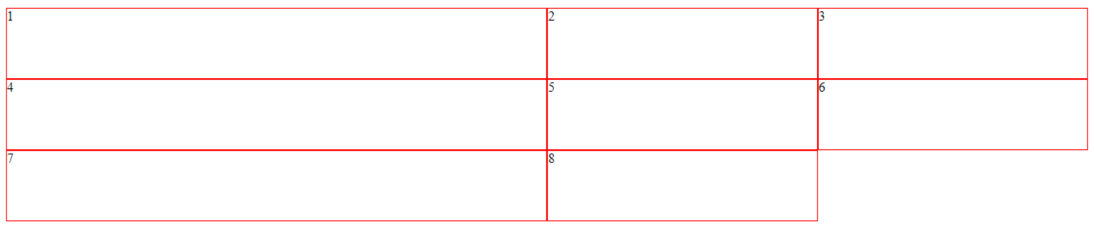
```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible"   content="IE=edge">
    <meta name="viewport"   content="width=device-width, initial-scale=1.0">
    <title>Document</title>
    <style>
        .container{
            display: grid;
            grid-template-columns: 40% 20%   20%;
            grid-template-rows: 2fr 4fr   2fr;
        }
        .item{
            border: 1px solid  red ;
        }
    </style>
</head>
<body>
    <div class="container">
        <div class="item" > 1</div>
        <div class="item" > 2</div>
        <div class="item" > 3</div>
        <div class="item" > 4</div>
        <div class="item" > 5</div>
        <div class="item" > 6</div>
        <div class="item" > 7</div>
        <div class="item" > 8</div>
 </div>

</body>
</html>
```



```
 grid-template-rows: 2fr 4fr  2fr;
```

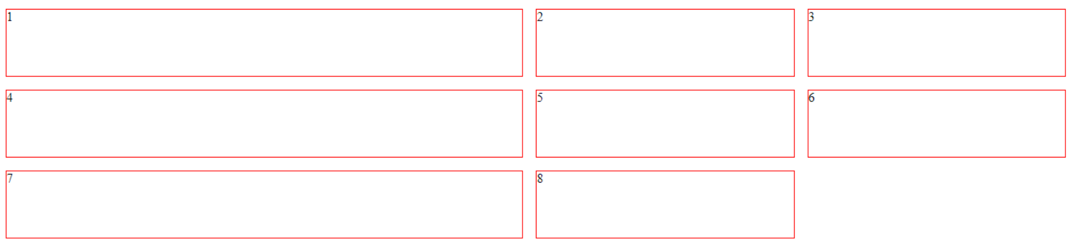
If we place content into the divs and some content is longer than the content in other divs, we can use the following instead of grid-template-rows:

```
grid-auto-rows: minmax(10vh, auto); /*first value is the minimum height of the row, second is the maximum size- in this case: if the content is longer, it will automatically adapt and not flow out of the divs*/
```



We can also state how large the spacing should be between the columns or rows:

```
 .container{
    display: grid;
    grid-template-columns: 40% 20%  20% ;
    grid-auto-rows: minmax(10vh, auto );
    column-gap: 1em;
    row-gap: 1em;
}
```

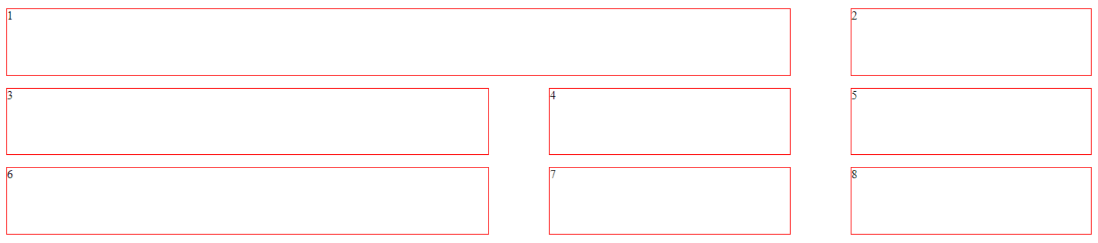


Or use the shorthand:

```
grid: minmax(10vh, auto) / 40%  20% 20%; /*row values / column values*/        gap: 1em 5em; /*first value is row gap, second value is column gap*/
```

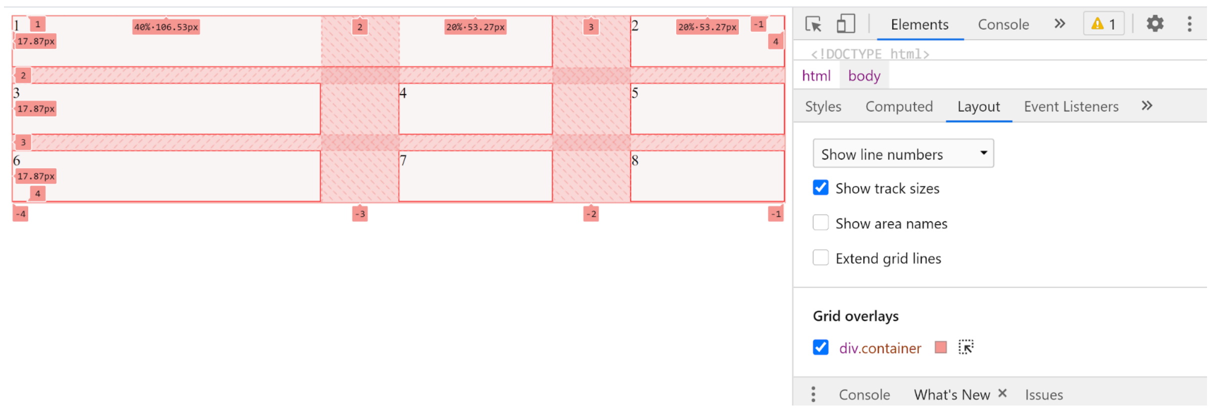
Now let’s take a look at how we can customize specific items in the grid system:

```
<!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>
        .container{
           display: grid;
           grid-template-columns: 40% 20%   20%;
           grid-auto-rows: minmax(10vh , auto);
           gap: 1em 5em;
        }
        .item{
            border: 1px solid  red ;
        }
        .item1{
            grid-column: 1/3;
        }
    </style>
</head>
<body>
    <div class="container">
        <div class="item item1" > 1</div>
        <div class="item" > 2</div>
        <div class="item" > 3</div>
        <div class="item" > 4</div>
        <div class="item" > 5</div>
        <div class="item" > 6</div>
        <div class="item" > 7</div>
        <div class="item" > 8</div>
 </div>

</body>
</html>
```



```
.item1 {
    grid-column: 1/3; /*here this item spans from column 1 to 3*/
}
```

Visually, if we open our developer tools, click on Layout and click on the Grid overlays option, we can see how the grid is set up. We can see that 1 / 3 (as in the example) clearly shows where column 1 begins and column 3 begins.



Show line numbers

[✓] Show track sizes

[ ] Show area names

[ ] Extend grid lines

Grid overlays

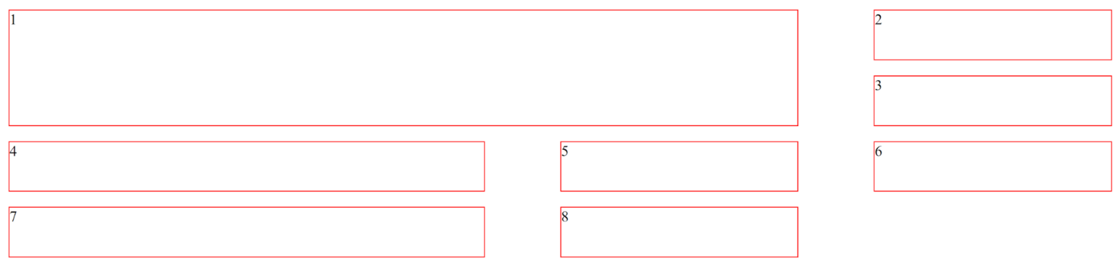
[✓] div.container

We can, of course, also customise how many rows this item should take:

```
grid-row: 1/3;
```

There is also the shorthand to combine both:

```
.item1 {
    grid-area:1/1/3 / 3;/*rowstart columstart rowend columnend */
}
```



Media Queries

Media queries give us the ability to specify how we want our content rendered across different devices and browsers (where the most limiting factor is the screen/viewport size). Here is an example of media queries in action .

Media types & queries

Responsive Web design is the approach that suggests that design and development should respond to the user's behavior and environment based on screen size, platform and orientation, since it is practically impossible to cover all existing devices and all browsers in their different versions.

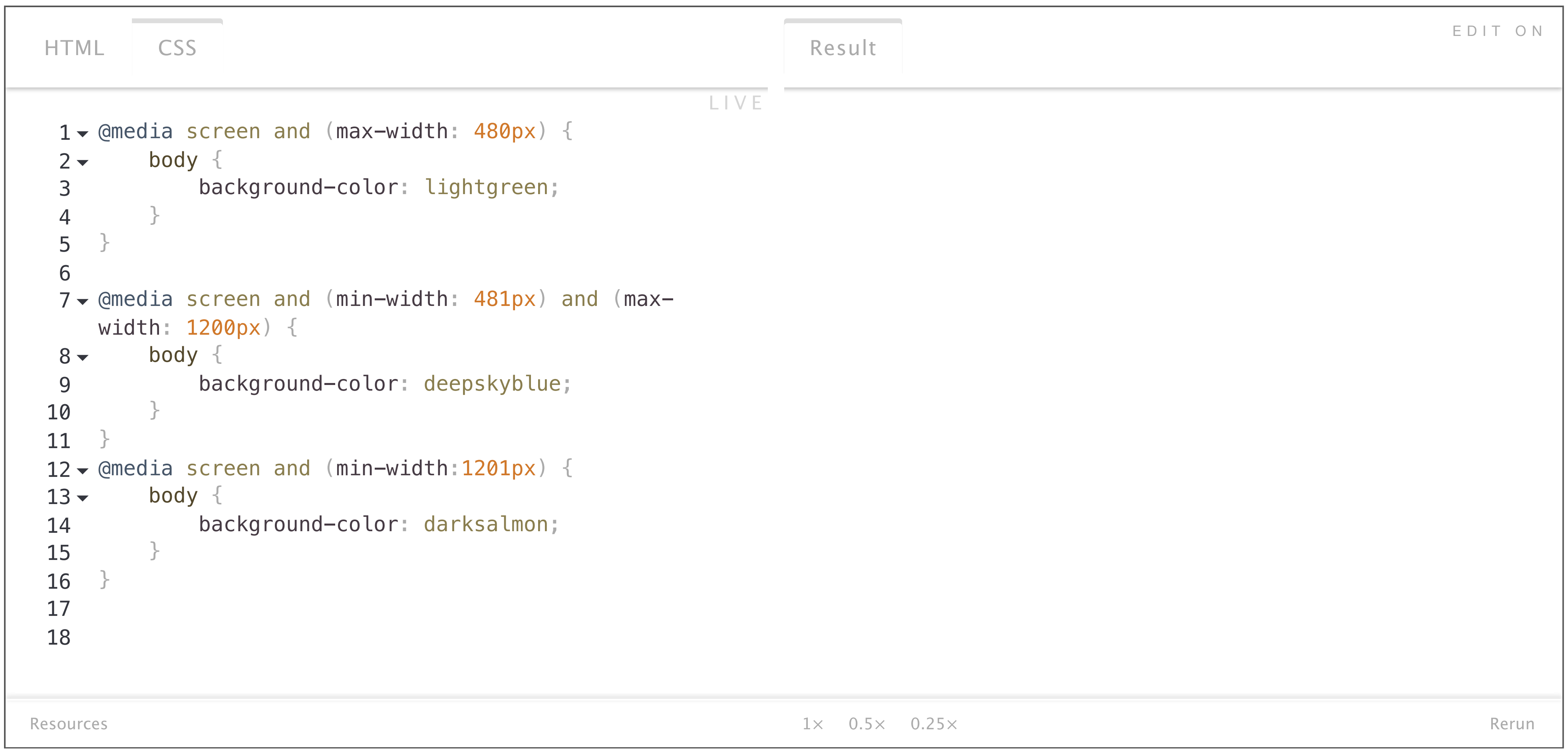
As the user switches from their laptop to smartphone, the website should accommodate for the change in resolution, image size and scripting abilities.

We are looking at **responsive design**, meaning not only should we apply media queries to different use cases, but we should already work in a responsive way when it comes to our dimensions. This is why we should move AWAY from using pixels, and move over to more responsive measurements and %.

Media queries can be used to check many device settings, such as:

- **Width** and **height** of the **viewport**
- **Width** and **height** of the **device**
- **Orientation** (is the tablet/phone in landscape or portrait mode?)
- **Resolution**

In action, it can look like this:



```css
@media screen and (max-width: 480px) {
    body {
        background-color: lightgreen;
    }
}

@media screen and (min-width: 481px) and (max-width: 1200px) {
    body {
        background-color: deepskyblue;
    }
}
@media screen and (min-width:1201px) {
    body {
        background-color: darksalmon;
    }
}


```

Most importantly, we want to set in our <head> section of our HTML documents the following:

```html
<meta name="viewport" content="width=device-width, initial- scale=1.0" >
```

This meta-data will check the viewport settings, where the width of the device will be automatically calculated, and the zoom will start at 1.0(default zoom, meaning at 100%)

In our CSS we will have the following media queries:

```css
@media screen and (max-width: 480px) {
  body {
    background-color: lightgreen;
  }
}
@media screen and (min-width: 481px) and (max-width: 1200px) {
  body {
     background-color: deepskyblue;
  }
}
@media screen and (min-width:1201px) {
  body {
    background-color: darksalmon;
  }
}
```

We have targeted different viewport sizes here (size < 480px, 480px < size <1200px, size > 1200px). Although this example changes only the background colors, more useful syntax targeting font size and block size/position can easily be constructed.

Note: the numbers 480px and 1200px (defined this way) are also known as ***breakpoints***.

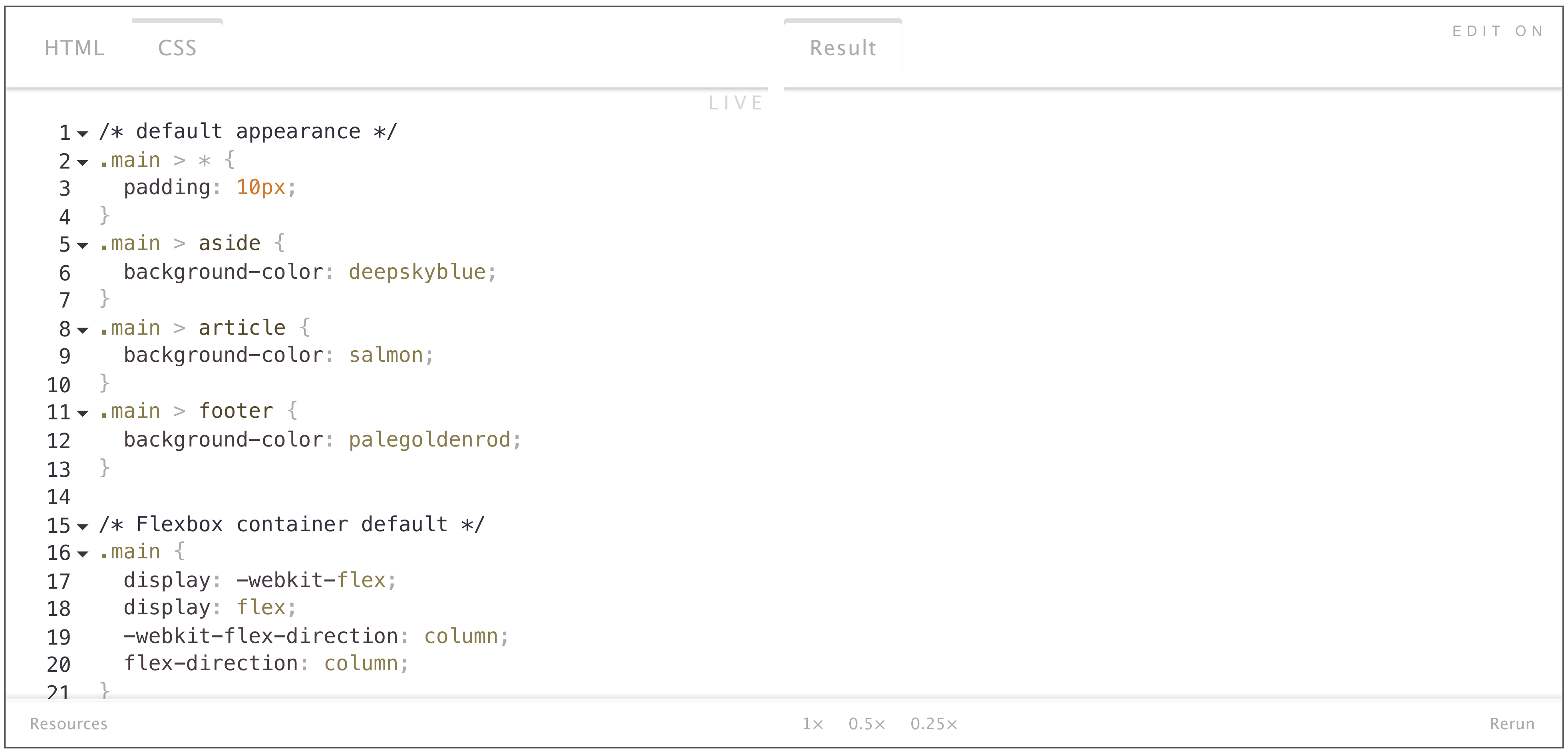
If the **viewport no longer has the properties defined in the media query,** the rules within no longer apply and the relevant nodes/html elements fall back to whatever **CSS rules originally applied** to them!!

Breakpoints: There are comprehensive lists of possibilities to optimize for certain devices that you can find online, for example here .

Our advice is, if you are just practicing, to stick to two or three breakpoints in your responsive design.

It is not necessary to have a continuous track of media-queries through all ranges of viewscreen-size. A rule of thumb: be aware of different devices and their screen sizes, but **implement according to your requirements and design.**

See this simple example that combines FlexBox & media queries to change the layout of a web page depending on viewport size:



```css
/* default appearance */
.main > * {
  padding: 10px;
}
.main > aside {
  background-color: deepskyblue;
}
.main > article {
  background-color: salmon;
}
.main > footer {
  background-color: palegoldenrod;
}

/* Flexbox container default */
.main {
  display: -webkit-flex;
  display: flex;
  -webkit-flex-direction: column;
  flex-direction: column;
}
```

These are all considerations that have to be made when altering the CSS for different screen sizes, (e.g. what is the most important thing for the user, at this specific stage).

Last modified: Tuesday, 10 September 2024, 6:48 PM

◄ Solution | Intermediate Exercise | Nutrition Table

Jump to...

CSS | Day 4 | Quiz ►