Classes and IDs

Two common attributes used to single out certain HTML elements are **class** and **id**, both are used to identify particular elements when adding CSS styling rules.**You author class + id names!!** They have no particular meaning in themselves, besides a puzzle - or code - you are creating.

Use a **class** when you have more than one element you want to share the same styling - perhaps across multiple pages.

Use an **id** when there is only one element on the page with that id, for example id="header"
With a class you can have as many elements with that styling as you like.

An element can have more than one **class**, but not more than one **id**. When there is more than one class, the class names are separated by spaces.

```
<h1 id="myHeader">Hello World!</h1>
```

IDs

Every HTML element can carry the id attribute. It is used to uniquely identify that element from other elements on the page.

Its value should start with a letter or an underscore (not a number or any other character). It is important that no two elements on the same page have the same value for their id attributes (otherwise the value is no longer unique).

More to read on ID naming: https://mathiasbynens.be/notes/css-escapes

IDs

To select these IDs in CSS you would do so with **#myHeader** syntax

(IDs may become particularly useful when it comes to media elements - photos, videos + sound files.)

```
#myHeader{
    color: blue;
}
```

Classes

Every HTML element can also carry a **class** attribute. Sometimes, rather than uniquely identifying one element within a document, you will want a way to identify several elements as being different from the other

```
<div class="cities">
<h2>London</h2>
<p>London is the capital of England. It is the most populous city in the United
Kingdom, with a metropolitan area of over 13 million inhabitants.</p>
<p>Standing on the River Thames, London has been a major settlement for two
millennia, its history going back to its founding by the Romans, who named it
Londinium.</p>
</div>

<div class="cities">
<h2>Paris</h2>
<p>Paris is the capital and most populous city of France.</p>
<p>Situated on the Seine River, it is at the heart of the Île-de-France region, also
known as the région parisienne.</p>
<p>Within its metropolitan area is one of the largest population centers in Europe,
with over 12 million inhabitants.</p>
</div>

<div class="cities">
<h2>Tokyo</h2>
<p>Tokyo is the capital of Japan, the center of the Greater Tokyo Area, and the most
populous metropolitan area in the world.</p>
<p>It is the seat of the Japanese government and the Imperial Palace, and the home of
the Japanese Imperial Family.</p>
<p>The Tokyo prefecture is part of the world's most populous metropolitan area with
38 million people and the world's largest urban economy.</p>
</div>
```

Classes

```
<div class="theAuthor">
  -- from John Duckett's <span><a
  href="https://www.amazon.com/Web-Design-HTML-JavaScript-jQuery/dp/1118907442
  /ref=sr_1_3?ie=UTF8&qid=1526310943&sr=8-3&keywords=html+and+css"
  target="_blank">HTML + CSS</span></a>
  <br>
</div>
```

To select these classes in CSS you would do so with .theAuthor syntax

```
.theAuthor{
  background: rgb(255,255,255);
    /* HSL: Hue, Saturation + Lightness
    Hue - as an angle between 0 + 360
    Saturation - as a precentage
    Lightness - as a precentage: 0% = white, %50 = normal + 100% is bl
    Alpha - expressed btw 0 _ 1.0 : 0.5 = 50% transparency, .75 is 75%
    transparency*/
  background: hsl(0,100%,100%, 0.2);
  text-align: center;
}
```

css Box Model

Box Dimensions

By default the box is sized just big enough to hold its contents. Use the height + width properties to set your own dims.

min-width, max-width
min-height, max-height

- `px` - traditionally, the most popular way of specifying the size of a box

- `%` - the size of the box is relative to the size of the browser window, or if the box is inside another box.

- `em` - the size of the box is are don the size of the txt inside it. Designers have recently started to use % + ems more as they are flexible across platforms + devices

- `rem` - Relative to font-size of the root element. check it here

Border

All boxes have borders even if invisible or 0px wide. It separates the edge of one box from another.

Padding

Padding is the space btw the border + any content contained within it. More padding increases the readability of its contents.

Margin

Margins sit outside the edge of the border. You can set the width to create a gap btw borders of adjacent boxes.

Border

can also specify each border individually:

- **border-top**
- **border-bottom**
- **border-left**
- **border-right**

+ set each property individually:

```
border-style: dotted; (all styles)
border-width: 3px;
border-color: purple;
```

Border

```
<p>
    This is then a very little description of
    feeling disillusionment in living.
</p>
```

```
p {
    border: 4px solid black;

}
```

When we add a border to an element, it sits flush against the text.

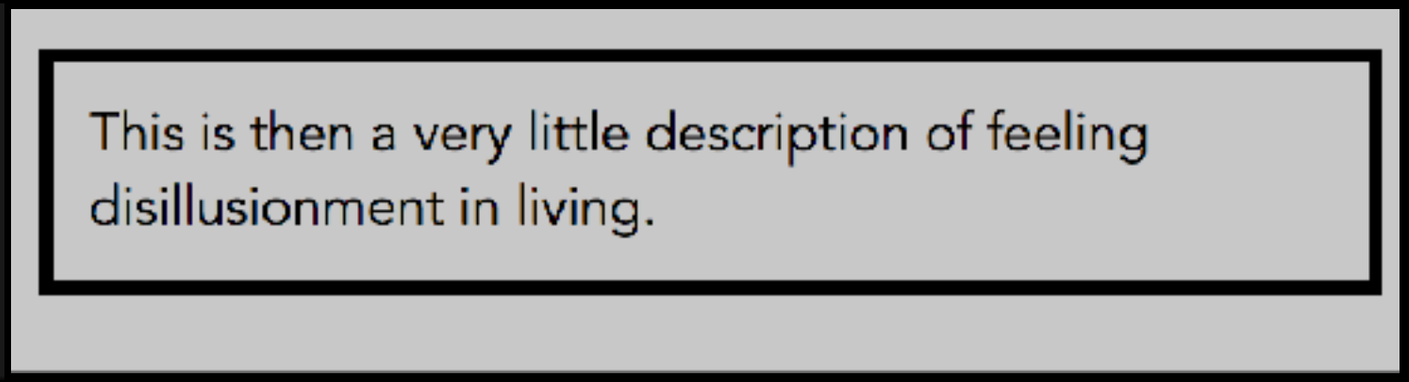
This is then a very little description of feeling disillusionment in living.

Q: How do we add space btw the border + the content of the element?

Padding

Padding is the space btw the border + the content.

```
p {
    border: 4px solid black;
    padding: 10px;
}
```



Can specify

- **padding-top**
- **padding-bottom**
- **padding-left**
- **padding-right**

There's also a shorthand:

padding: 2px 4px 3px 1px; <- top | right | bottom | left

padding: 10px 2px; <- top + bottom | left + right

```html
<div class="marginEx">
Lectures
</div>
<div class="marginEx">
    Homework
</div>
```

```css
.marginEx {
    border: 2px solid black;
    padding: 10px;
}
```

Lectures

Homework

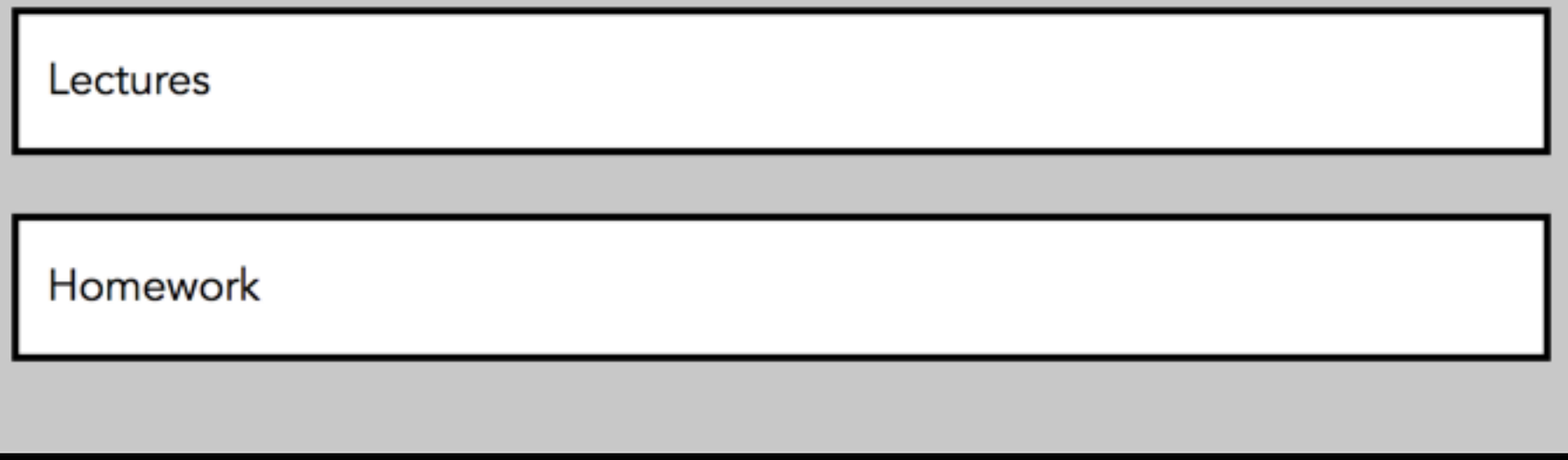
When we add a border to multiple divs (w/ the same class), they sit flush against each other:

Q: How do we add space btw elements?

Margin

Margin is the space btw the border + other elements

```
.marginEx {
    margin: 20px;
    border: 2px solid black;
    padding: 10px;
}
```



Can specify

margin-top
margin-bottom
margin-left
margin-right

There's also a shorthand:

margin: 2px 4px 3px 1px; <- **top** | **right** | **bottom** | **left**
margin: 10px 2px; <- **top + bottom** | **left + right**

Margin Collapsing

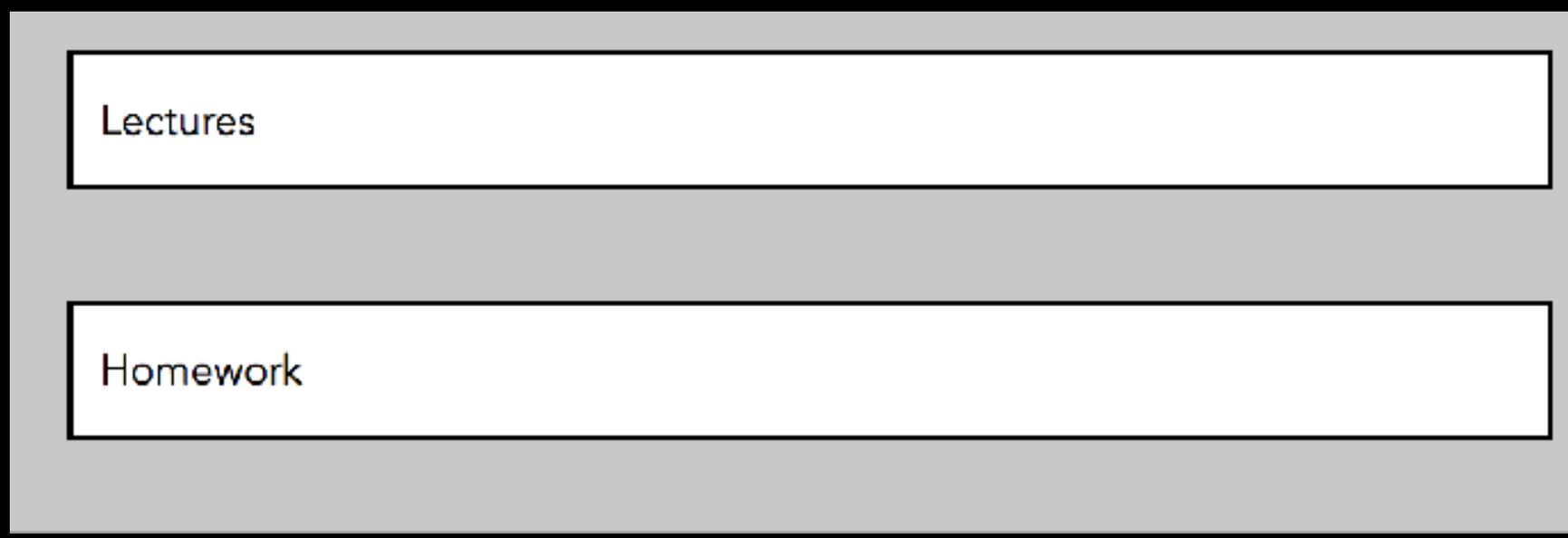
Sometimes the top + bottom margins of block elements are combined ("collapsed") into a single margin.

Generally if:

- the elements are siblings
- the elements are block-level (**not inline**)

Then they collapse into **max** (Bottom Margin, Top Margin).

20px margin-bottom + 20px margin-top = 40 px margin?



Negative Margin

Margins can be negative as well.

```css
img {
    margin-top: -100px;
    margin-left: -100px;
    height: 200px;
    border: 2px solid GREEN;
}
```

-100 px margin-bottom

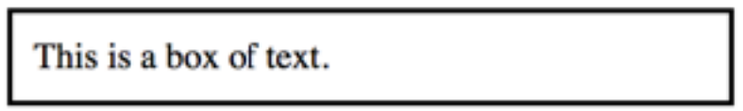
-100px margin-top

Auto Margins

If you set **margin-left** + **margin-right** to auto - you can center a block-level element

```html
<html>
  <head>
    <meta charset="utf-8">
    <title>Auto Margins</title>
  </head>
  <body>
    <div>
      This is a box of text.
    </div>
  </body>
</html>
```

```css
div {
  margin-left: auto;
  margin-right: auto;

  border: 2px solid black;
  padding: 10px;
  width: 300px;
}
```



Box Model for Inline Elements

Box Model applies to inline elements too, but the box is shaped differently.

```
<p> Week 03
    <strong>"There is then as I am saying
    complete disillusion in living, the
    realis- ing, completely realising that
    not any one, not one fighting for the
    same thinking and believing as the
    other... </strong>
Which is a sentence my Gertrude Stein + is
pretty long but is only part of the full one
at that.
</p>
```

```
strong {
    border: 5px solid hotpink;
    color:white;
    padding: 8px;
    margin: 24px;
    background-color:pink;
}
```

Week 03 "There is then as I am saying complete disillusion in living, the realis- ing, completely realising that not any one, not one fighting for the same thinking and believing as the other... Which is a sentence my Gertrude Stein + is pretty long but is only part of the full one at that.
</p>

Inline Element Box Model

Margin is to the left + right of the inline element

- **margin-top** + **margin-bottom** are ignored.

Use **line-height** to manage btw spaces.

```
p {
    width: 500px;
    line-height: 50px;
}

strong {
    border: 5px solid hotpink;
    color:white;
    padding: 8px;
    margin: 24px;
    background-color:pink;
```

Week 03 "There is then as I am saying complete disillusion in living, the realis- ing, completely realising that not any one, not one fighting for the same thinking and believing as the other... Which is a sentence my Gertrude Stein + is pretty long but is only part of the full one at that.

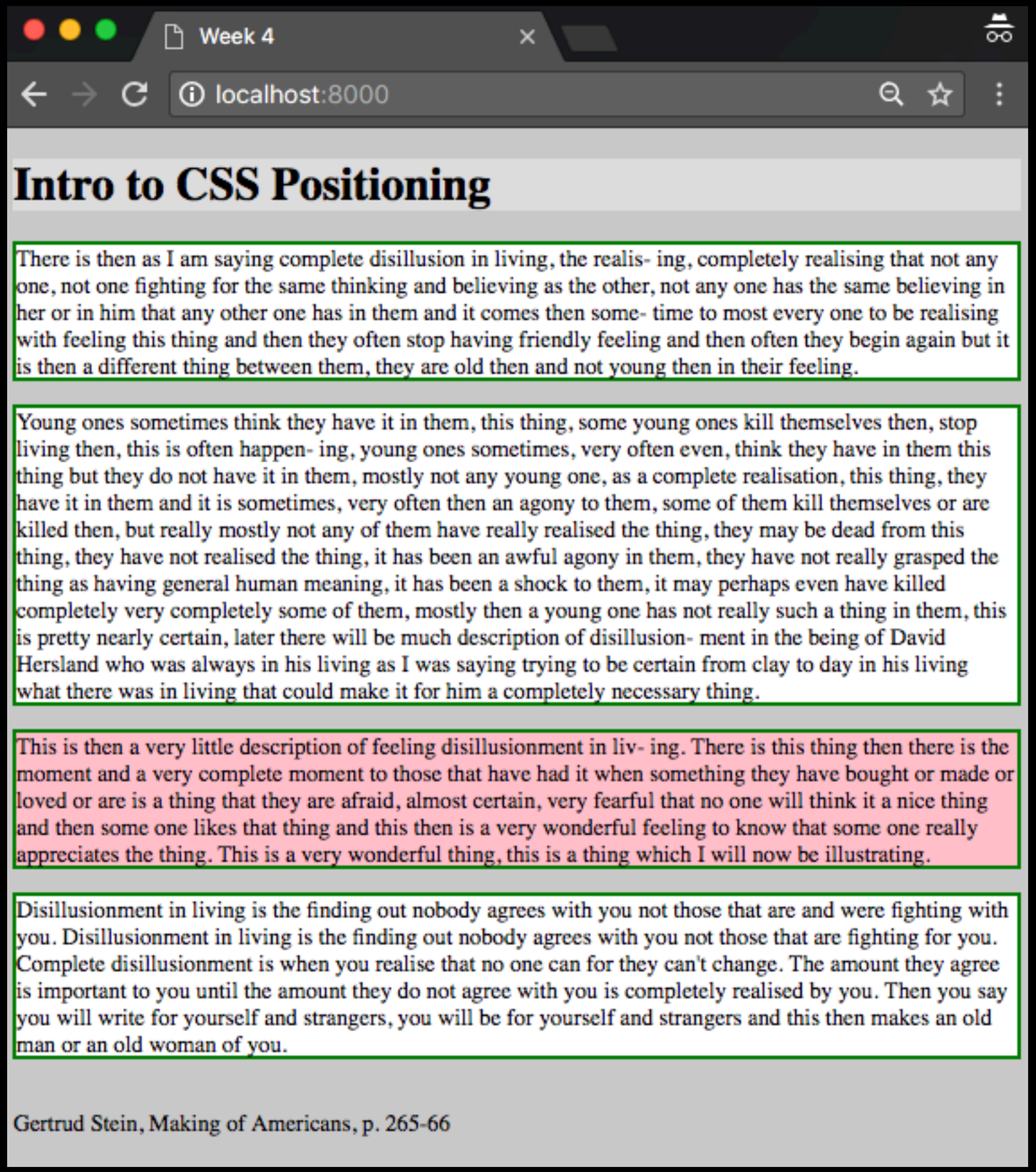
css positioning

Controlling the Position of Elements

Normal Flow

Every block-level element appears on a new line, causing each item to appear lower down on the page. Even if you specify the width of the boxes + there is space they will not appear nxt to each other.

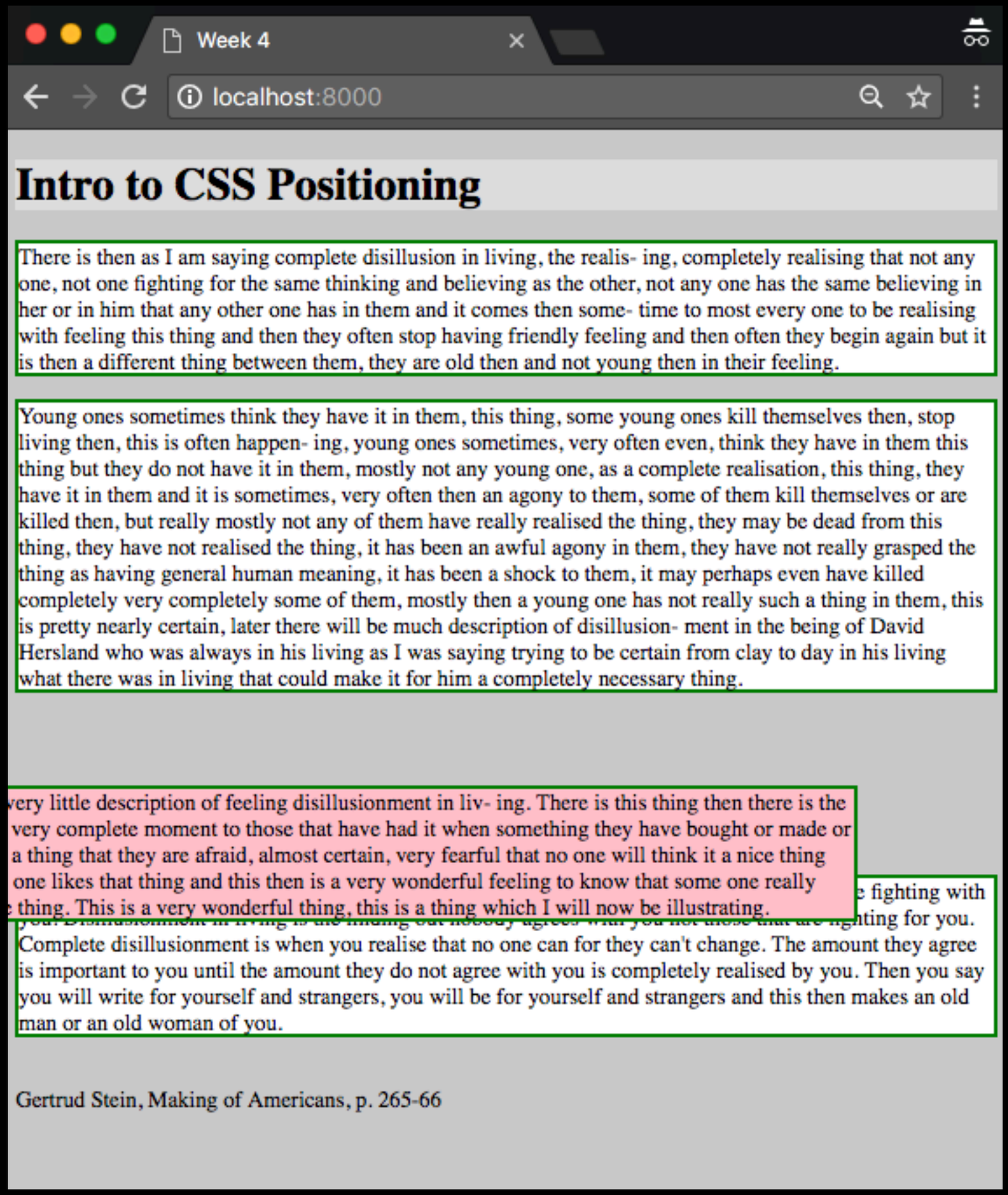
```
.thePosition {

   background: pink;
    position: static;

}
```



Relative Positioning

This moves an element from the position it would be in normal flow, shifting it to the top, right, bottom, or left where it would have been placed. This does not affect the position of surrounding elements; they stay in the position they would be in normal flow.

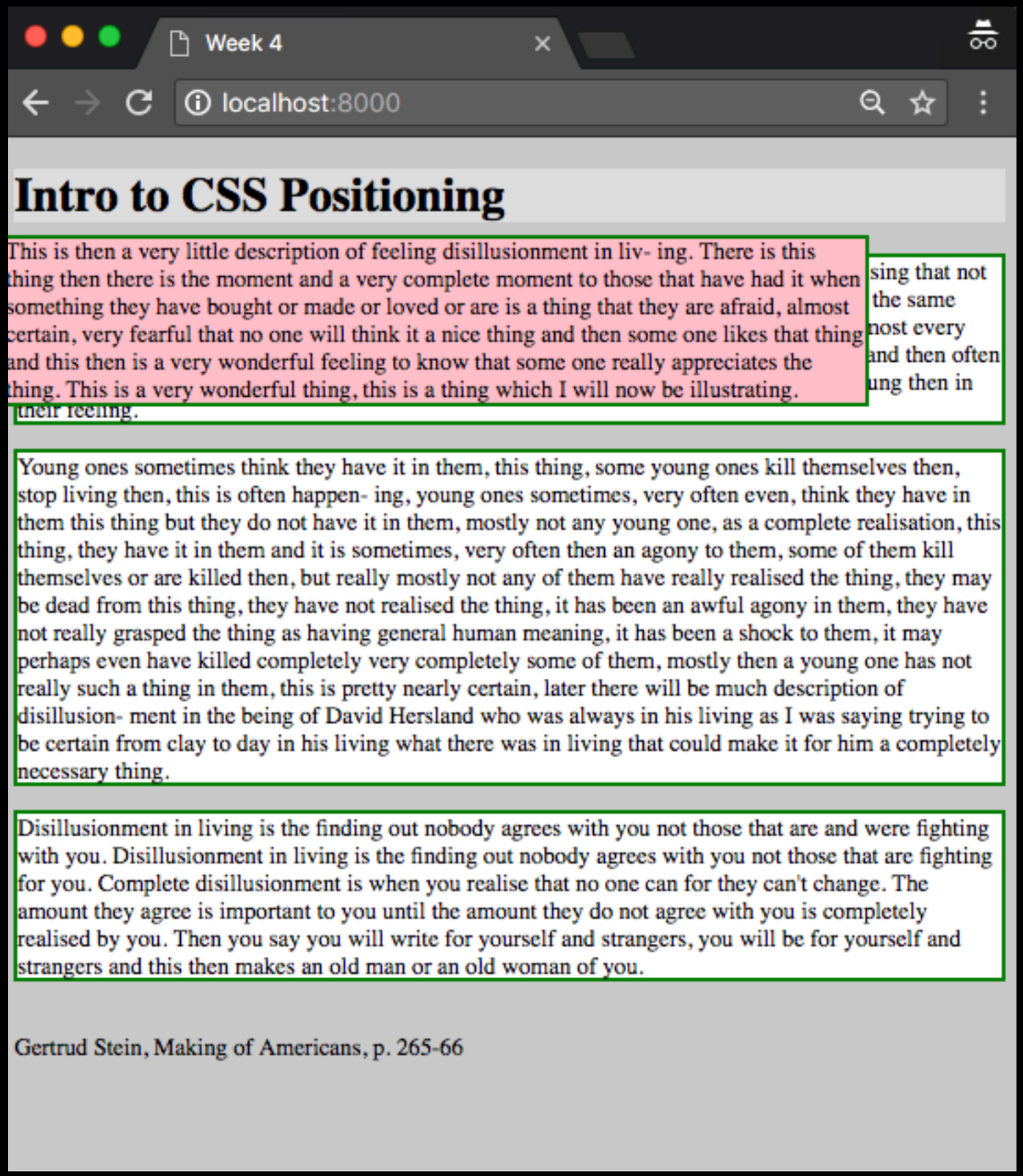
```
.thePosition {

  background: pink;
  position: relative;
  top: 50px;
  right: 100px;

}
```



Absolute Positioning

This positions the element in relation to its containing element. It is taken out of normal flow, meaning that it does not affect the position of any surrounding elements. An absolutely positioned element no longer exists in the normal document layout flow. Instead, it sits on its own layer separate from everything else. Absolutely positioned elements move as users scroll up + down.

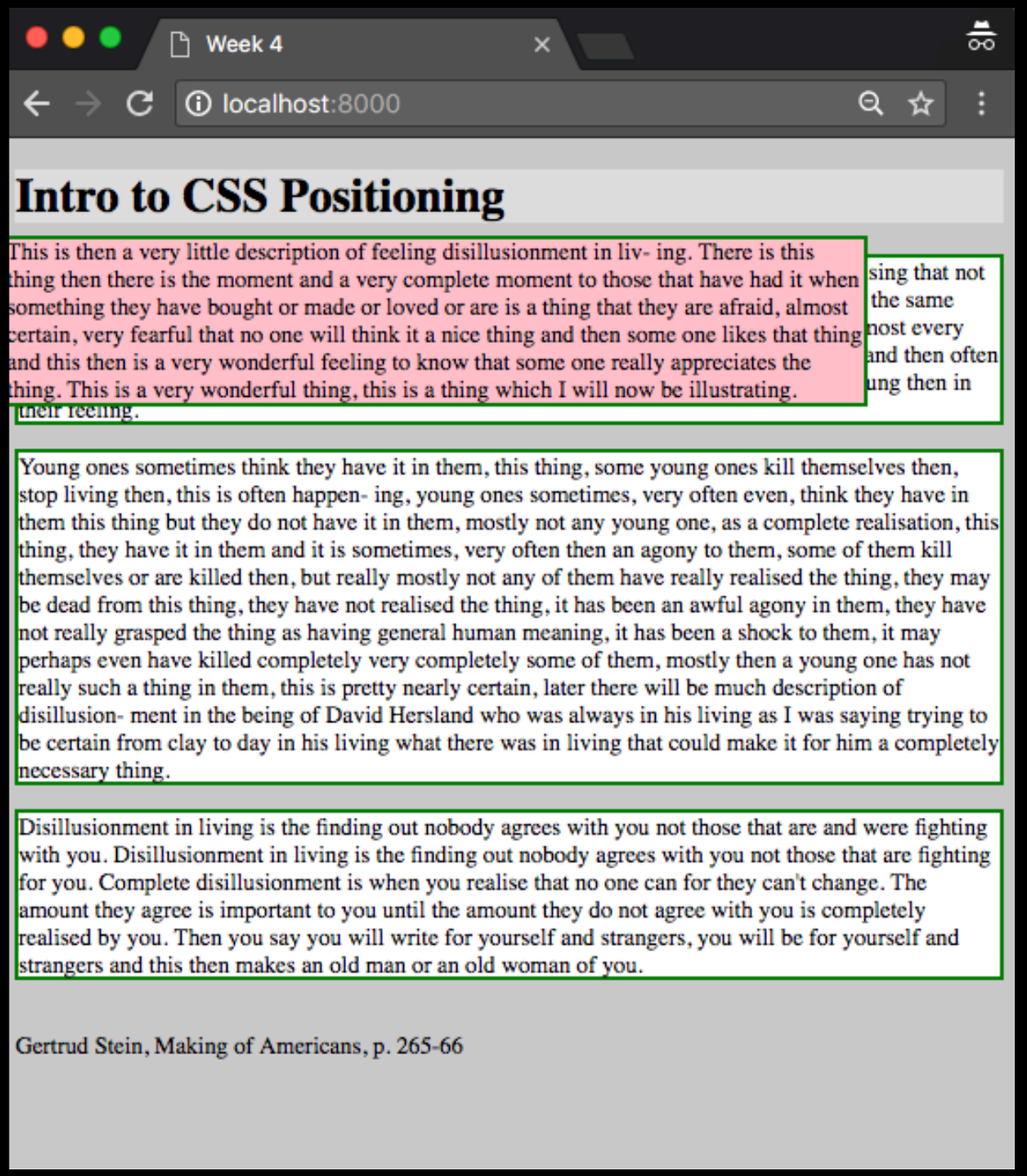
```
.thePosition {

  background: pink;
  position: absolute;
  top: 50px;
  right: 100px;

}
```



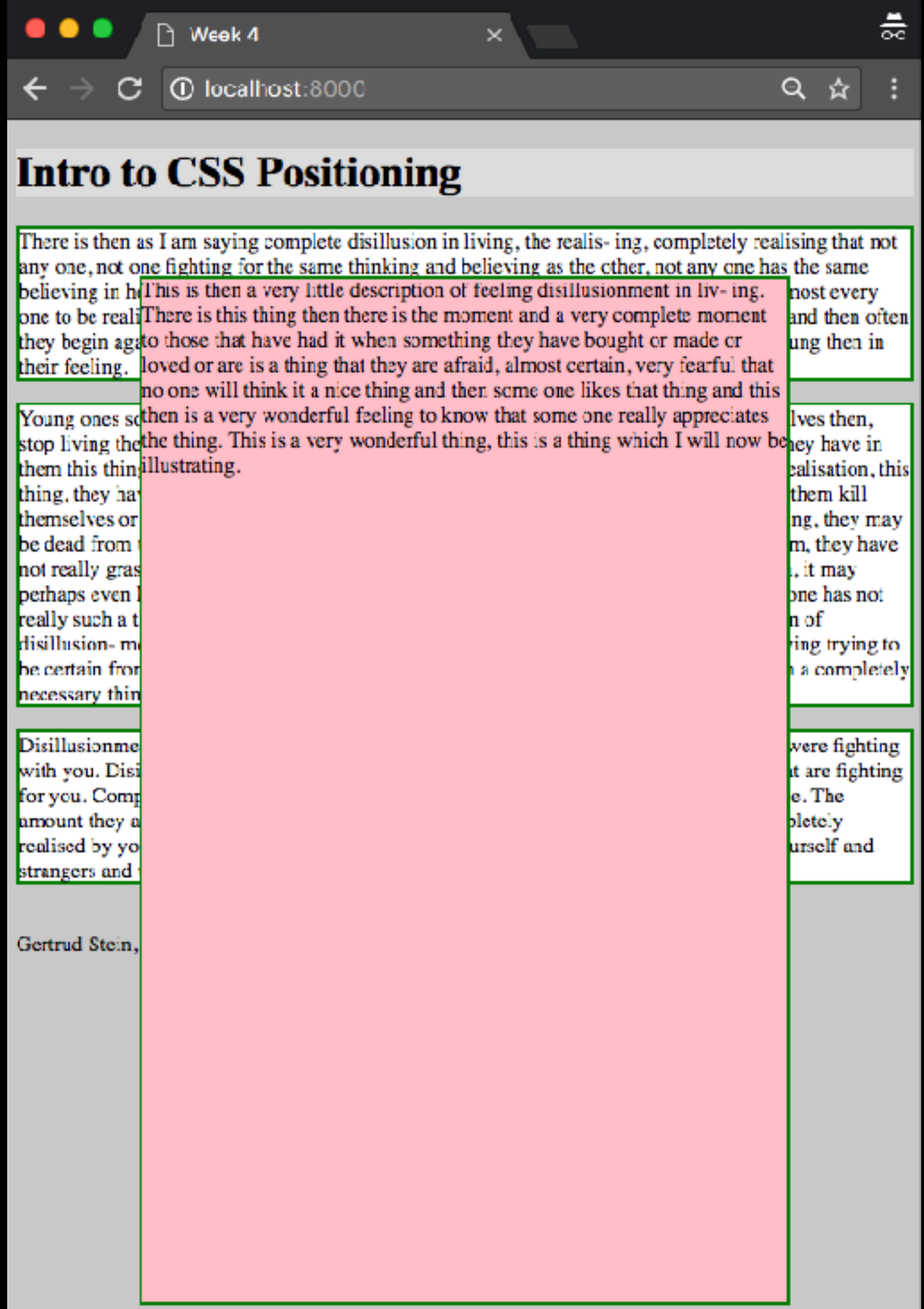
Notice that the position of the element has changed — this is because **top**, **bottom**, **left**, and **right** behave in a different way with absolute positioning.

Instead of specifying the **direction** the element should move in, they specify the **distance** the element should be from each containing element's sides.

So in this case, we are saying that the absolutely positioned element should sit **50px** from the **top** of the "containing element", and **100px** from the **right**.

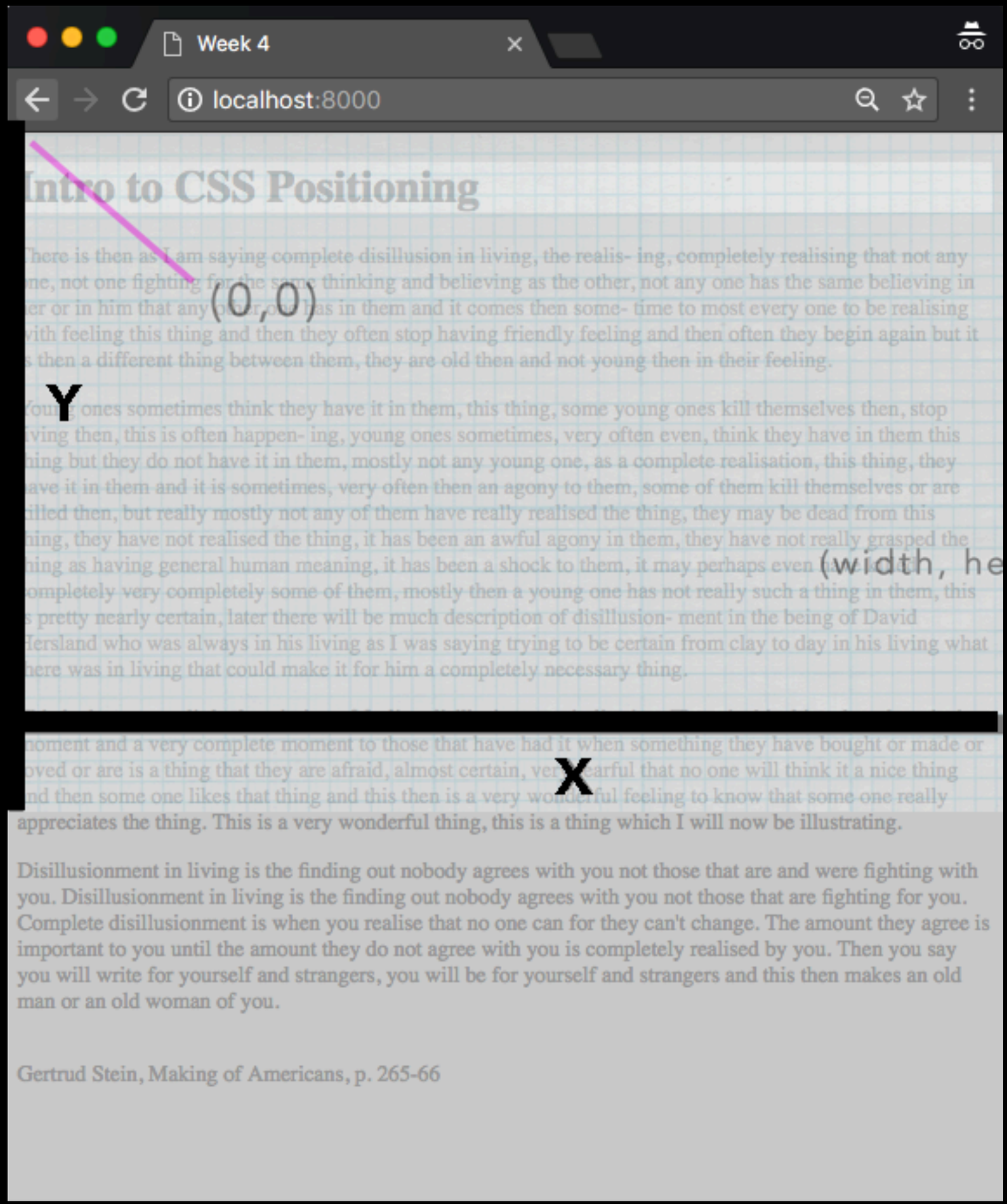


```
.thePosition {

   background: pink;
   position: absolute;
   top: 100px;
   bottom: 100px;
   left: 100px;
   right: 100px;

}
```



left
right
} X axis

top
bottom
} Y axis

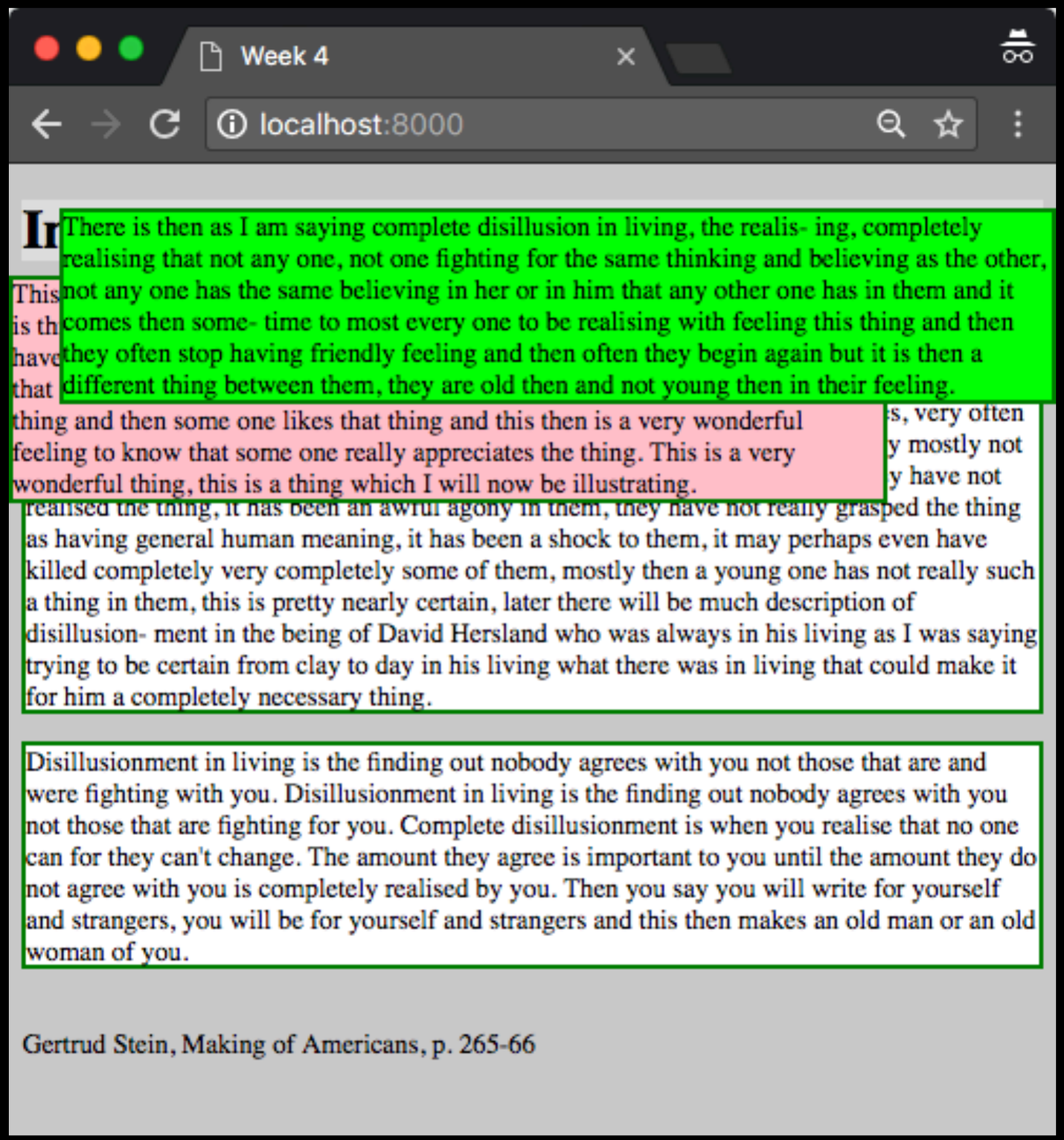


Floating Elements

Floating an element allows you to take that element out of normal flow. The floated element becomes a block-level element around which other content can flow.

Web pages also have a z-axis: an imaginary line that runs from the surface of your screen, towards your face. z-index values affect where positioned elements sit on that axis; positive values move them higher up the stack, and negative values move them lower down the stack. By default, positioned elements all have a z-index of auto, which is effectively 0

```
{
  background: lime;
  position: absolute;
  top: 10px;
  left: 30px;
  z-index: 1;
}
```

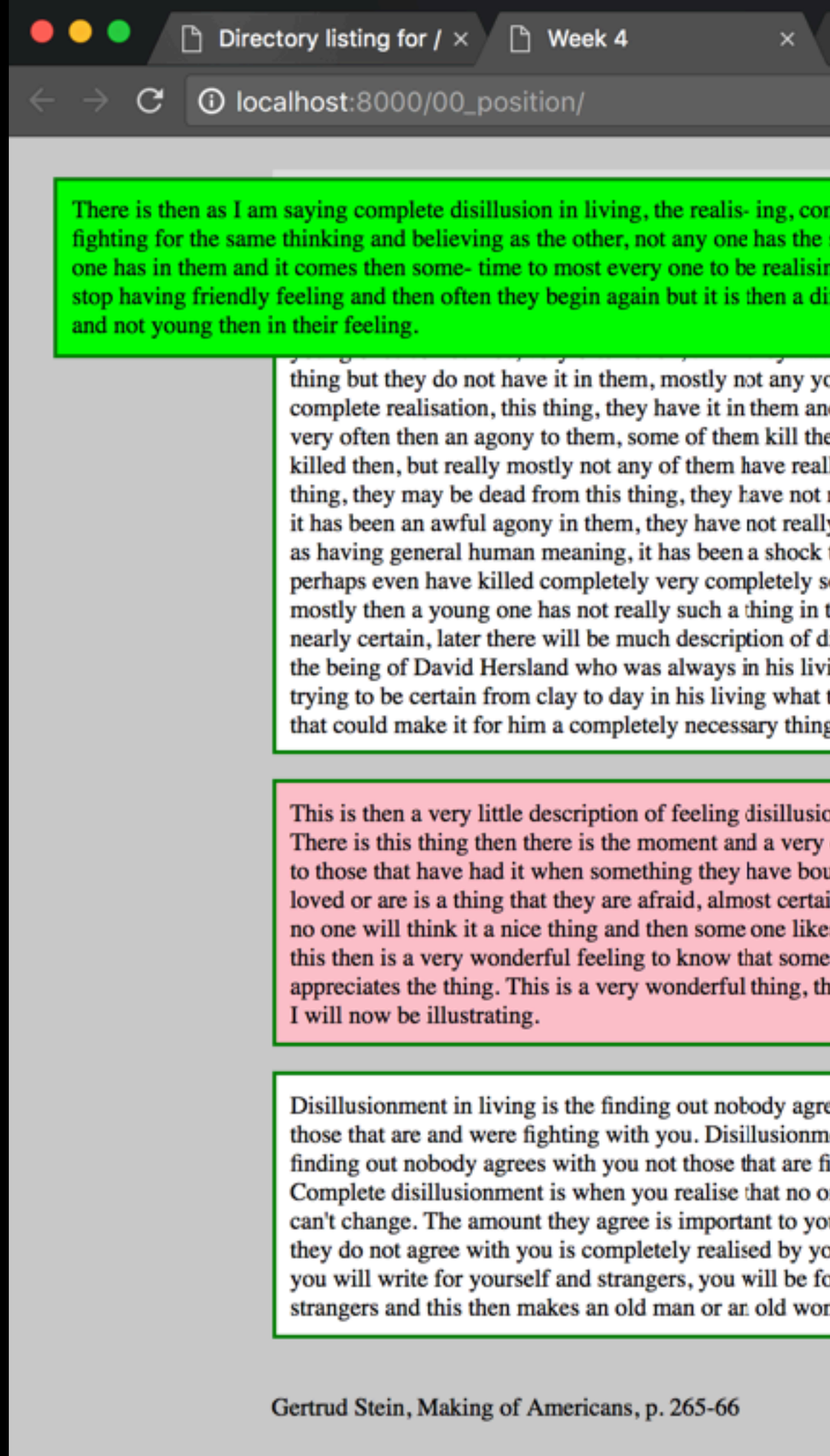


Overlapping Elements

When you use relative, fixed or absolute positioning, boxes can overlap. If they do the elements later in the HTML will sit on top of those that are earlier in the page.

When you move any element form normal flow, boxes can overlap. The **z-index property** allow you to control which box appears on top. It's value is a number, + the hight the number the closer that element is to the top.

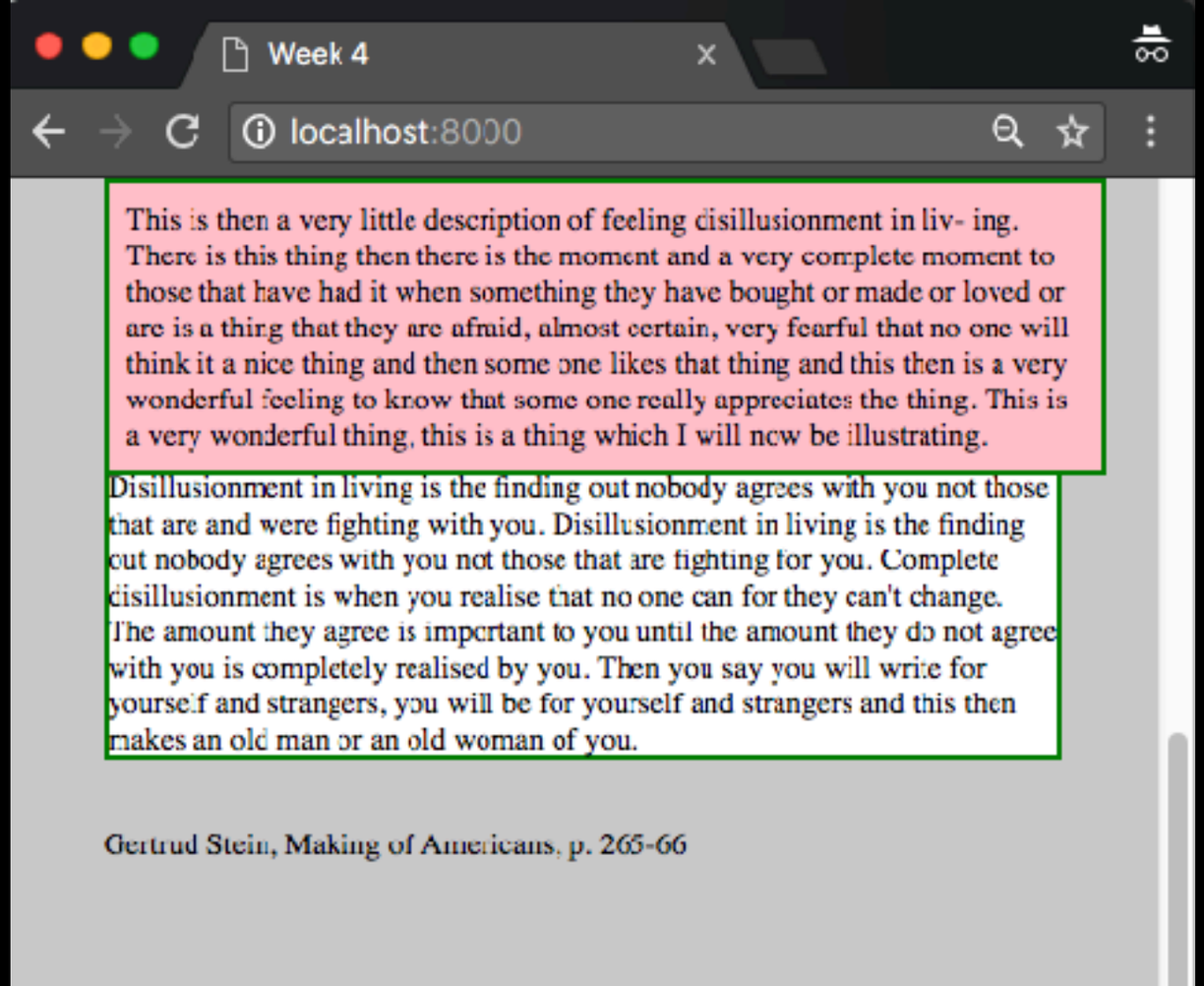
Often referred to as **STACKING CONTEXT** + is similar to "bring to front" + "send to back features."



Fixed Positioning

This is a form of absolute positioning that positions the element in relation to the browser window, as opposed to the containing element.

Elements w/ fixed positioning do not affect the position of surrounding elements + they do not move when the user scrolls up and down the page.




```
.thePosition {

   background: pink;
    position: fixed;
    top: 0;
    width: 50px;
    margin: 0 auto;
    padding: 10px;

}
```

Position Sticky