

# Typeface terminology

**Serif**

A serif is a small decorative flourish on the end of the strokes that make up letters and symbols.

**Sans Serif**

Sans Serif fonts do not have any flourishes at the end of strokes.

**Monospaced**

Monospaced fonts, letters, and characters each occupy the same amount of horizontal space.

**Georgia**

**Times**

**Times New Roman**

**Arial**

**Verdana**

**Helvetica**

**Courier**

**Courier New**

# TYPE OF WORK



PRINT

SERIF!



WEB

SANS



# Font stack

It's important to understand that the browser will only display a font if it's installed on user's computer.

**Font stack** - a collection of more than one typeface in an order of preference to be displayed in the browser if some of the typefaces are not found.

```
{  
font-family: Georgia, Courier, serif;  
}
```

# Setting the font-family

- **font-family** property sets the font in your CSS
- Presented as a hierarchy of choices (1st choice, 2nd choice, 3rd choice) so it's good to have a fallback for older browsers that can't render

```
body {
```

```
    font-family: Georgia, Courier, serif;
```

```
}
```

```
h1, h2, h3 {
```

```
    font-family: Arial, Verdana, sans-serif;
```

```
}
```

# Google Fonts: step 1

The screenshot shows the Google Fonts website with the Roboto font selected. The browser's address bar displays the URL <https://fonts.google.com/specimen/Roboto>. The page features a navigation bar with links to DIRECTORY, FEATURED, and ABOUT. The main heading is "Roboto", accompanied by a red "+ SELECT THIS FONT" button and a red rectangular box. Below the heading, the page is divided into three sections: "Glyph", "Characters", and "Designer".


**Glyph**

Rr

**Characters**

ABCĆČDĎEFGHIJKLMNOPQRSŠTUVWXYZŽ  
abcčćdďefghijklmnopqrsštuvwxyzžАБВГГ'  
ДЪЕЁЄЖЗСИІЙЈКЛЉМНЊОПРСТЋУЎФХ  
ЦЧЏШЩЪЫЬЭЮЯабвггдђеёєжзсиіійјклљ  
мњњопрстћуўфхцчџшщъыьэюяАВГДЕЗН  
ΘΙΚΛΜΝΞΟΠΡΣΤΥΦΧΨΩαβγδεζηθικλμνξοπ  
ρστυφхψωάΆέΈεΉίϊΐόΌύϋϋΥΎΩǼǼÊÔŮǻ  
âêôσϣ1234567890'?"'!"(%)[#]{@}/&<-+÷×=  
>®©\$€£¥¢:;,.\*

**Designer**

 Christian Robertson  
Principal design

**About**

Roboto has a dual nature. It has a mechanical skeleton and the forms are largely geometric. At the same time, the font features friendly and open curves. While some grotesks distort their letterforms to force a rigid rhythm, Roboto doesn't compromise, allowing letters to be settled into their natural width. This makes for a more natural reading rhythm more commonly found in humanist and serif types.

This is the regular family, which can be used alongside the **Roboto Condensed**

# Google Fonts: step 2

Google Fonts

DIRECTORY FEATURED ABOUT

Roboto

Glyph

Characters

Rr

ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz  
ДТЪЕЁЄЖЗСИІЙЈКЛЉМНЦЧЏШЩЪЫЬЭЮЯабвггд  
мнњопрстћуўфхцчшщѣ  
ΘΙΚΛΜΝΞΟΠΡΣΤΥΦΧΨΩα  
ρστυφхψωάΆέΈέΉίϊΐόΌ  
âêôσϣ1234567890'?"'!"(%  
>®©\$€£¥¢:;,.\*

1 Family Selected

Your Selection [Clear All](#)

Roboto

EMBED CUSTOMIZE

Load Time **Fast**

**Embed Font**

To embed your selected fonts into a webpage, copy this code into the <head> of your HTML document.

**STANDARD @IMPORT**

```
<link href="https://fonts.googleapis.com/css?family=Roboto" rel="stylesheet">
```

**Specify in CSS**

Use the following CSS rules to specify these families:

```
font-family: 'Roboto', sans-serif;
```

For examples of how fonts can be added to webpages, see the [getting started guide](#).

## Custom web fonts: Google Fonts

Add link in **<head>** of HTML

```
<link href="https://fonts.googleapis.com/css?family=Roboto" rel="stylesheet">
```

Use with font-family property in CSS

```
font-family: 'Roboto', sans-serif;
```



```
2
3 ▼ <head>
4     <title>Week 4</title>
5     <!-- you will need to link to the css file -->
6     <link href="style.css" rel="stylesheet" type="text/css">
7     <!-- the following link is to the Google Font API
8     You should put any REQUESTS like this in the head of yr html
9     so your site grabs the DATA it needs before loading the content of the
10    page -->
11    <link href="https://fonts.googleapis.com/css?family=Roboto" rel="
12    stylesheet">
13 </head>
```

```
body {
    /*font-family: Avenir;*/
    font-family: 'Roboto', sans-serif;
    font-size: 15px;
    background: rgb(200,200,200);
}
```



**remember to  
return to the  
wiki for  
tools,  
resources +  
inspiration !!**

**class repo**

#### Tools:

- [CSS Syntax Validation](#)
- SFTP Software - [Cyber Duck](#)

#### Inspiration + Resources:

- Good website design!
  - [SiteInspire](#)
  - [Brutalist Websites](#)
  - [SiteSee](#)
  - [Awwwards](#)
- Good Web typography!
  - [Google Font API](#)
  - [Google Chrome Plug-in - What Font is that?](#)
  - [Typewolf](#)
  - [Fonts in Use](#)
  - [Canva - Font Pairing](#)
  - [Canva - Font Pairing Tool](#)
  - [Type.io](#)
- [Chrome Developer Tools](#)
- [Responsive Meta Tag](#)
- [w3 CSS Box Model](#)
- [MDN - CSS Box Model](#)
- [HTML Dog - CSS3 Properties](#)
- [MDN Size/Length](#)
- [MDN Styling Boxes](#)
- [MDN CSS Positioning](#)
- [MDN Positioning Example Code](#)
- [MDN More Floating Elements](#)

**box model**

**margin**

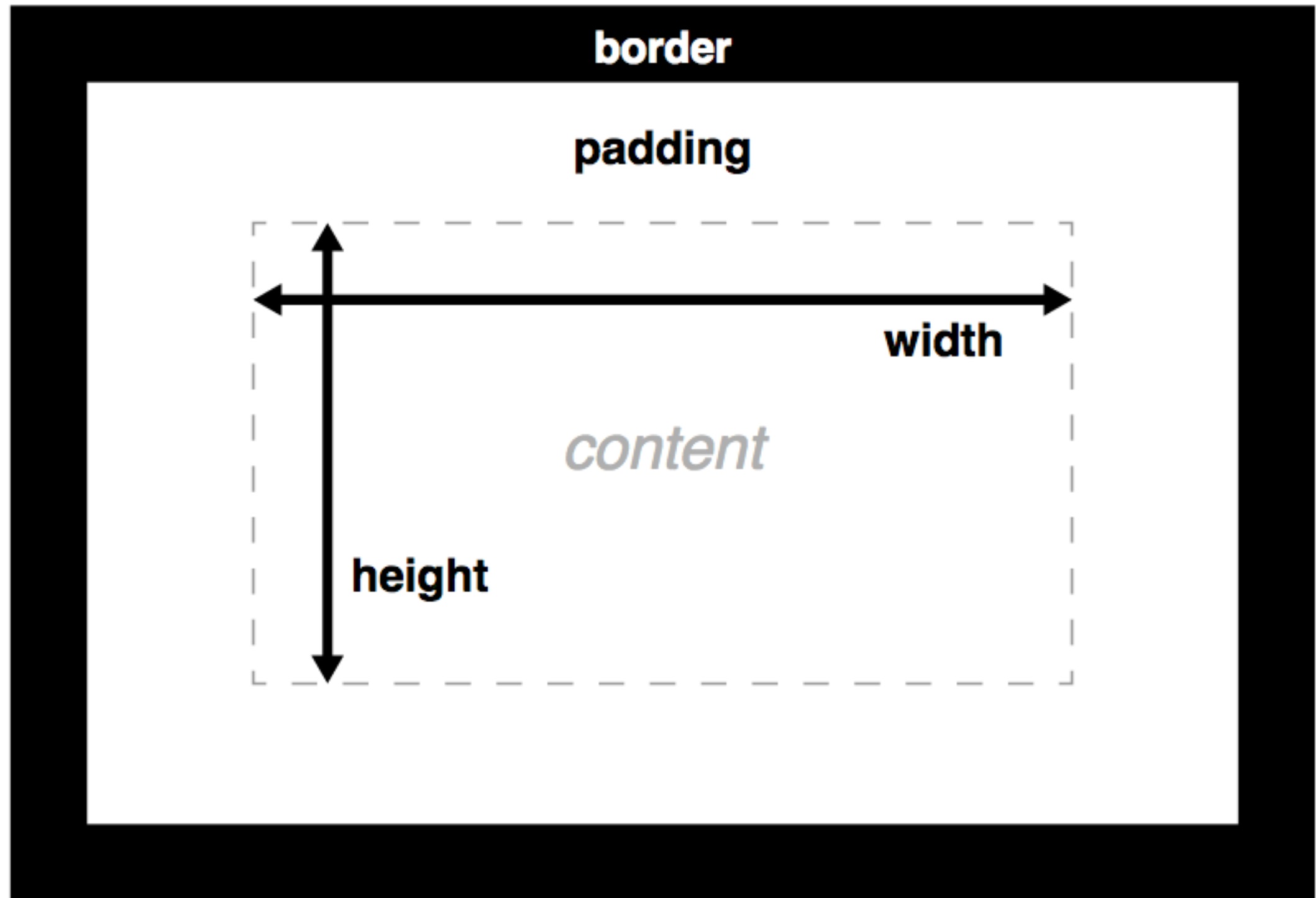
**border**

**padding**

**width**

*content*

**height**



## 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 based on the size of the text 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](#)

css positioning

# Padding

Padding is the space btw the border + any content contained within it. the INNER margin of a CSS box - btw the outer edge of the content box + the inner edge of the border. More padding increases the readability of its contents. Use the [padding](#) property.

## Border

The boarder of a CSS box sits btw the outer edge of the padding + the inner edge of the margin. It's default is 0px. It separates the edge of one box from another. Use the [border](#) property.

## Margin

Margins surround a CSS box + pushes up against other boxes in the layout. You can set the width to create a gap btw borders of adjacent boxes. Use the [margin](#) property.





# Containing Elements

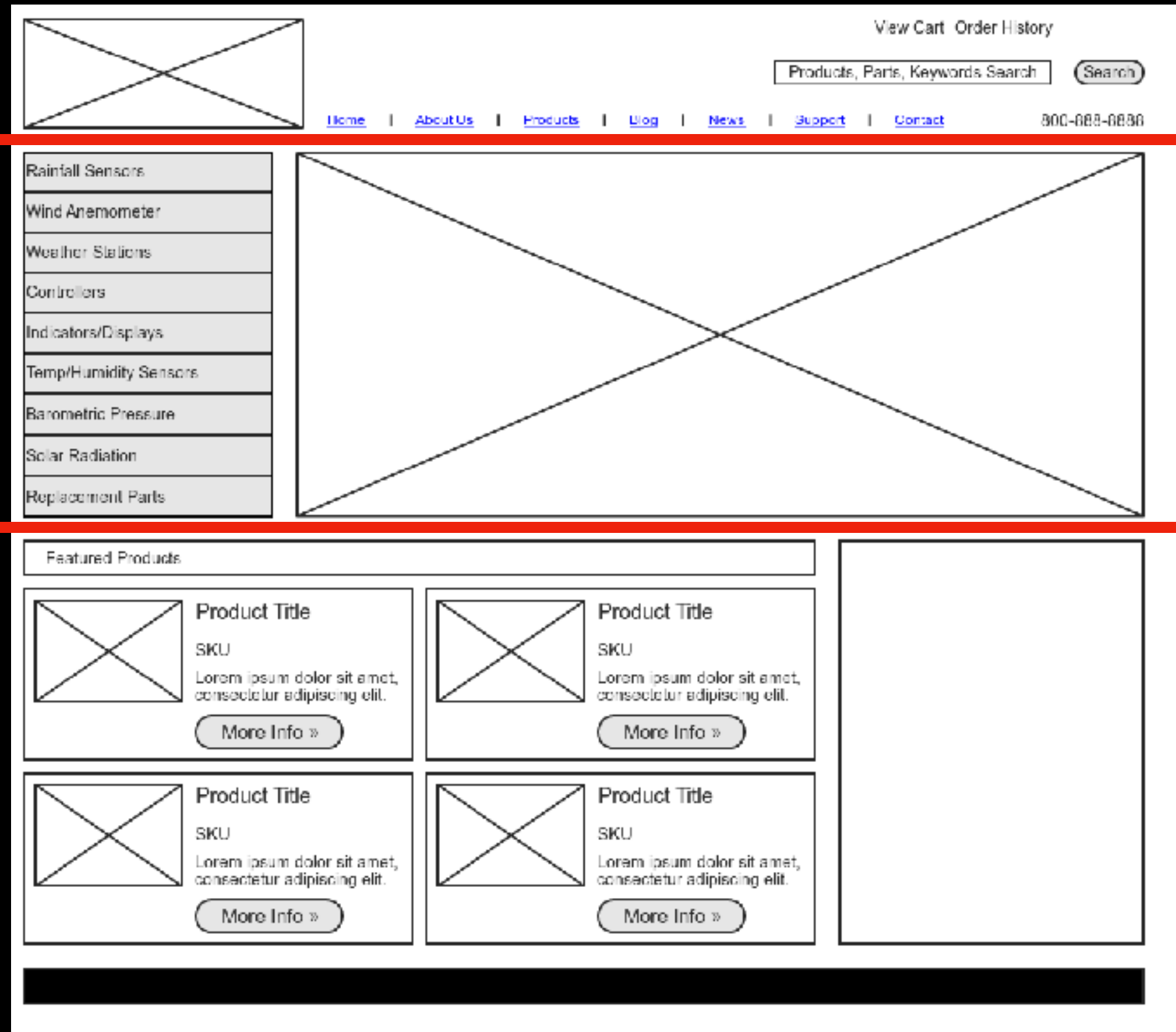
level 1

level 2

level 3

## Block level

elements start on a new line – if a block-level element sits inside another then the outer box is the containing or parent element.



# Controlling the Position of Elements

## Normal Flow

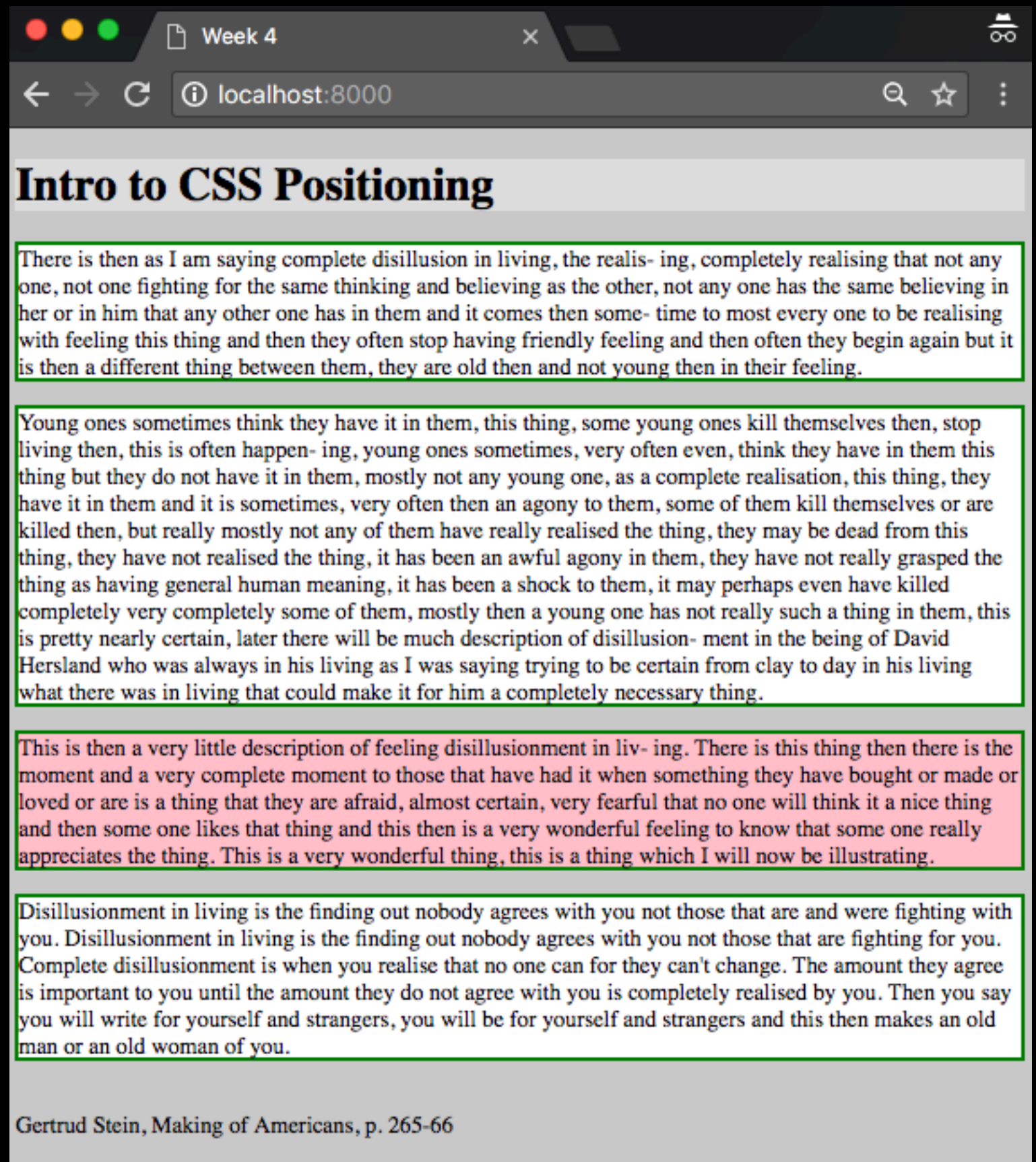
Every block-level element appears on a new line, causing each item to appear lower down on the page. Even if you specify the width of the boxes + there is space they will not appear next to each other.

```
.thePosition {
```

```
background: pink;
```

```
position: static;
```

```
}
```



## MDN Types of CSS Positioning

A **positioned element** is an element whose computed position value is either **relative**, **absolute**, **fixed**, or **sticky**. (In other words, it's anything except **static**.)

A **relatively positioned element** is an element whose computed position value is **relative**. The **top** and **bottom** properties specify the vertical offset from its normal position; the **left** and **right** properties specify the horizontal offset.

An **absolutely positioned element** is an element whose computed position value is **absolute** or **fixed**. The **top**, **right**, **bottom**, and **left** properties specify offsets from the edges of the element's containing block. (The containing block is the ancestor relative to which the element is positioned.) If the element has margins, they are added to the offset.

A **stickily positioned element** is an element whose computed position value is **sticky**. It's treated as relatively positioned until its containing block crosses a specified threshold (such as setting **top** to value other than auto) within its flow root (or the container it scrolls within), at which point it is treated as "stuck" until meeting the opposite edge of its containing block.

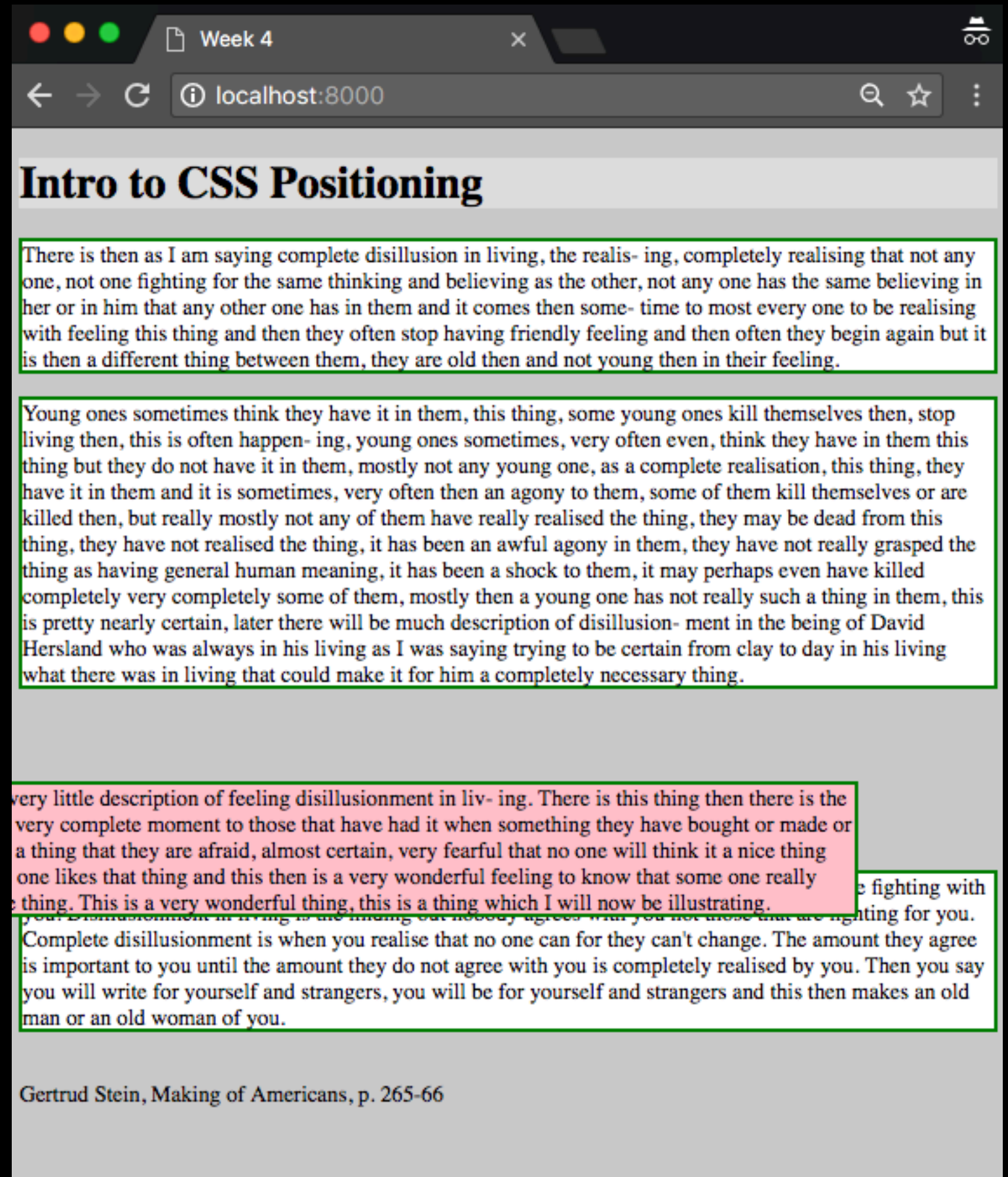


## Relative Positioning

**This moves an element from the position it would be in normal flow, and then offset relative to itself based on the values of top, right, bottom, and left. 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 {
```

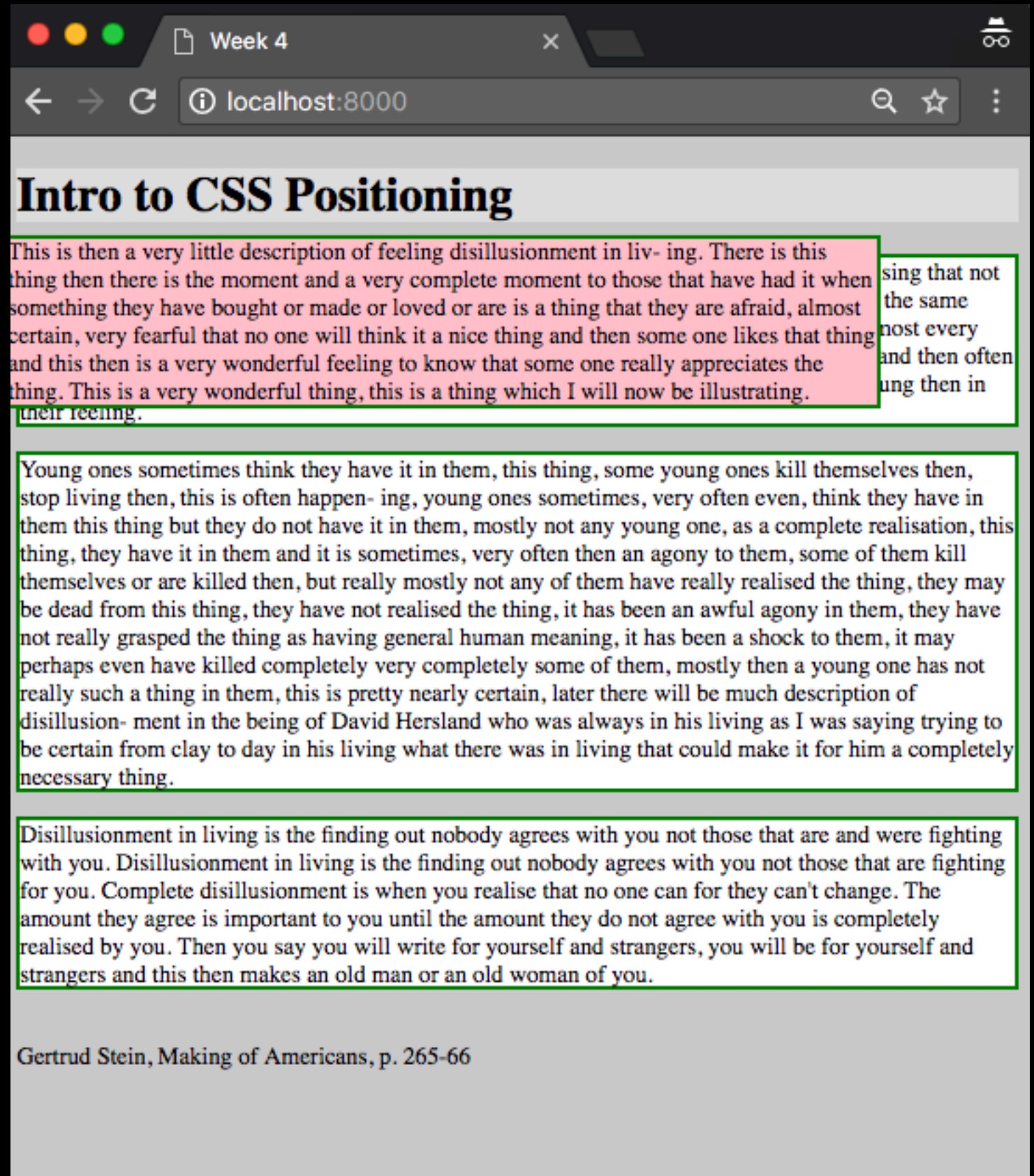
```
background: pink;
```

```
position: absolute;
```

```
top: 50px;
```

```
right: 100px;
```

```
}
```

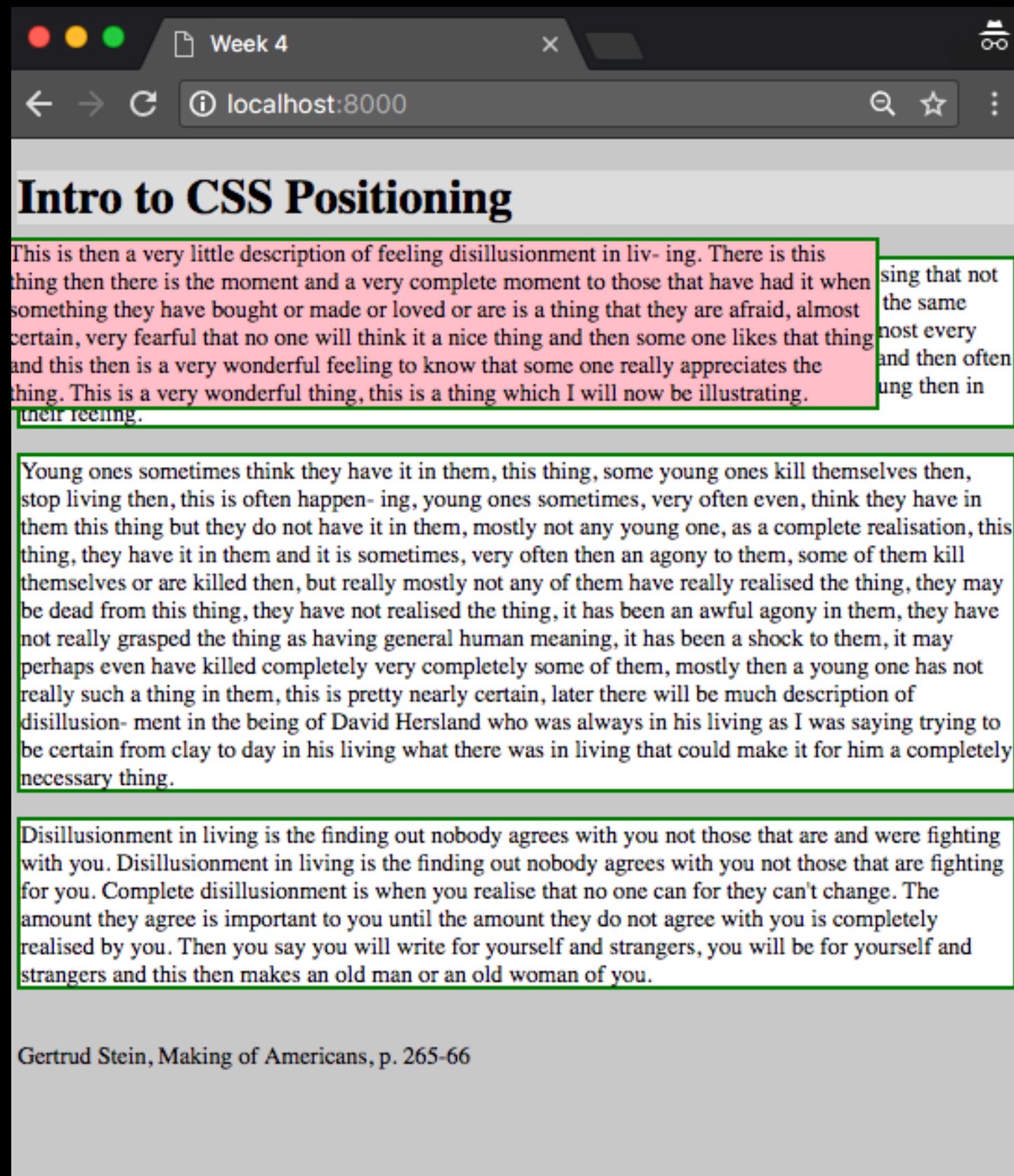




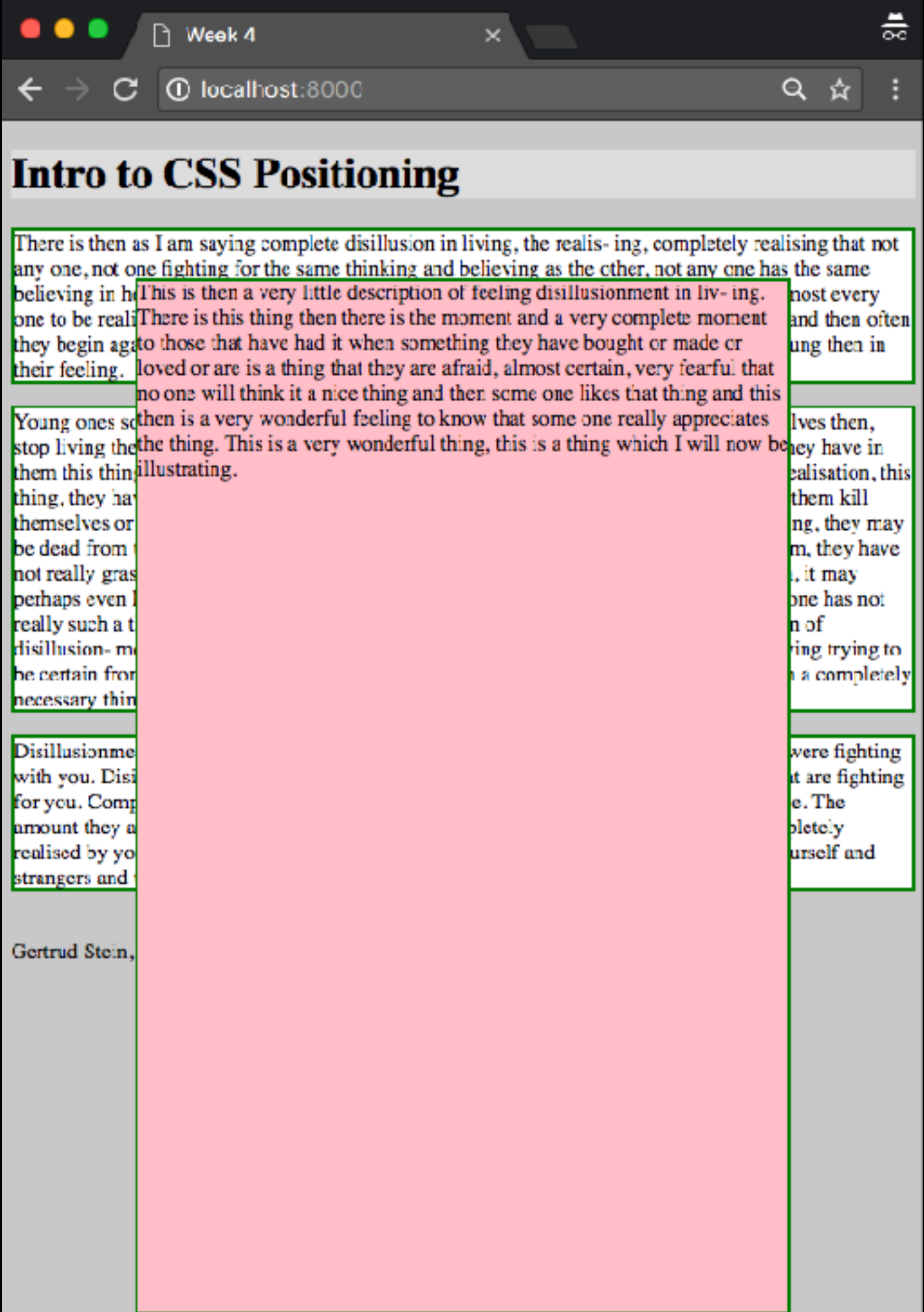
Notice that the position of the element has changed — this is because **top**, **bottom**, **left**, and **right** behave in a different way with absolute positioning.

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

Week 4

localhost:8000

# Intro to CSS Positioning

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, not any one has the same believing in her or in him that any other one has in them and it comes then some- time to most every one to be realising with feeling this thing and then they often stop having friendly feeling and then often they begin again but it is then a different thing between them, they are old then and not young then in their feeling.

Young ones sometimes think they have it in them, this thing, some young ones kill themselves then, stop living then, this is often happen- ing, young ones sometimes, very often even, think they have in them this thing but they do not have it in them, mostly not any young one, as a complete realisation, this thing, they have it in them and it is sometimes, very often then an agony to them, some of them kill themselves or are killed then, but really mostly not any of them have really realised the thing, they may be dead from this thing, they have not realised the thing, it has been an awful agony in them, they have not really grasped the thing as having general human meaning, it has been a shock to them, it may perhaps even be a shock to them completely very completely some of them, mostly then a young one has not really such a thing in them, this is pretty nearly certain, later there will be much description of disillusion- ment in the being of David Gersland who was always in his living as I was saying trying to be certain from clay to day in his living what there was in living that could make it for him a completely necessary thing.

moment and a very complete moment to those that have had it when something they have bought or made or loved or are is a thing that they are afraid, almost certain, very fearful that no one will think it a nice thing and then some one likes that thing and this then is a very wonderful feeling to know that some one really appreciates the thing. This is a very wonderful thing, this is a thing which I will now be illustrating.

Disillusionment in living is the finding out nobody agrees with you not those that are and were fighting with you. Disillusionment in living is the finding out nobody agrees with you not those that are fighting for you. Complete disillusionment is when you realise that no one can for they can't change. The amount they agree is important to you until the amount they do not agree with you is completely realised by you. Then you say you will write for yourself and strangers, you will be for yourself and strangers and this then makes an old man or an old woman of you.

Gertrud Stein, Making of Americans, p. 265-66

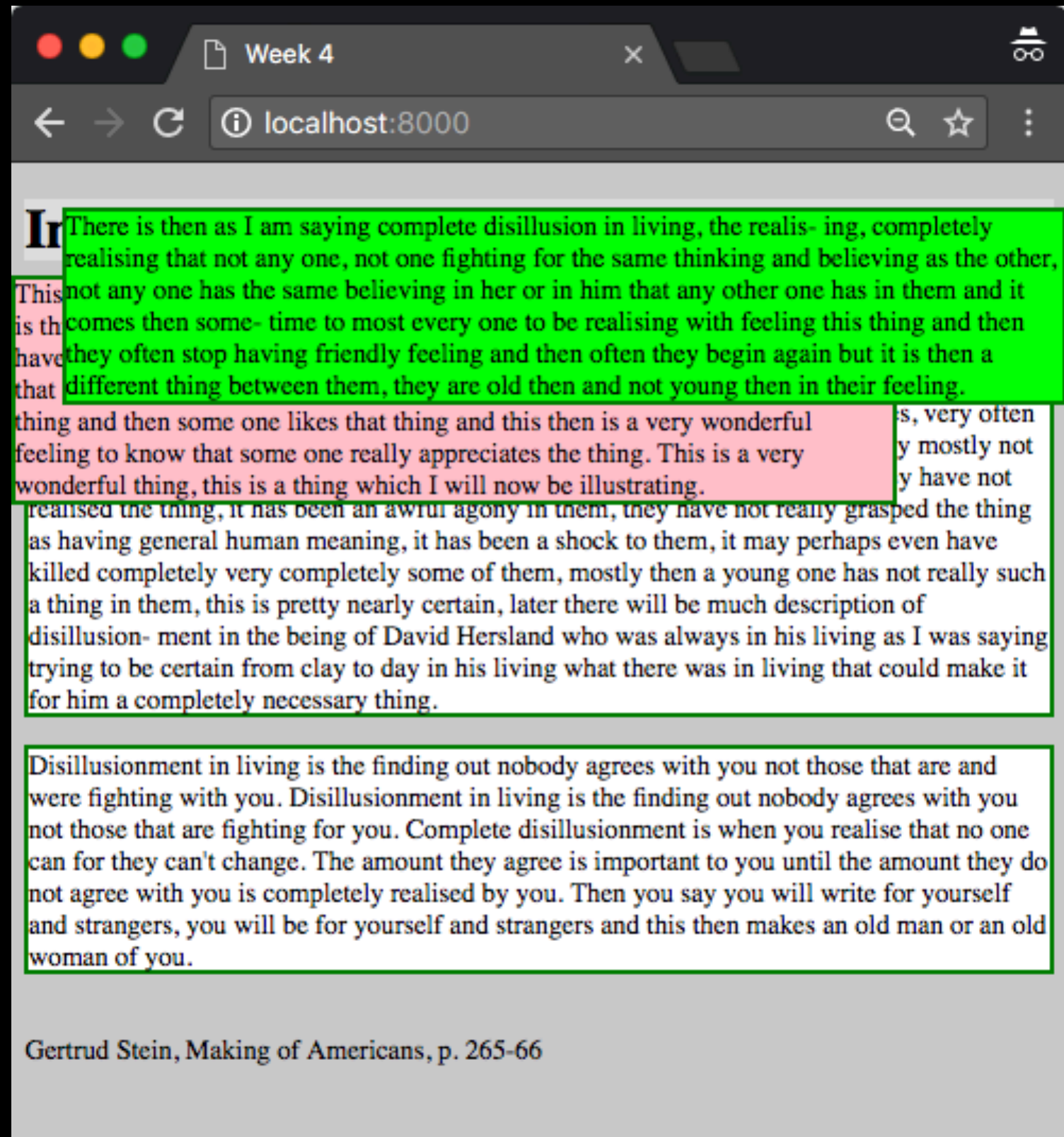


# Floating Elements

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;  
}
```

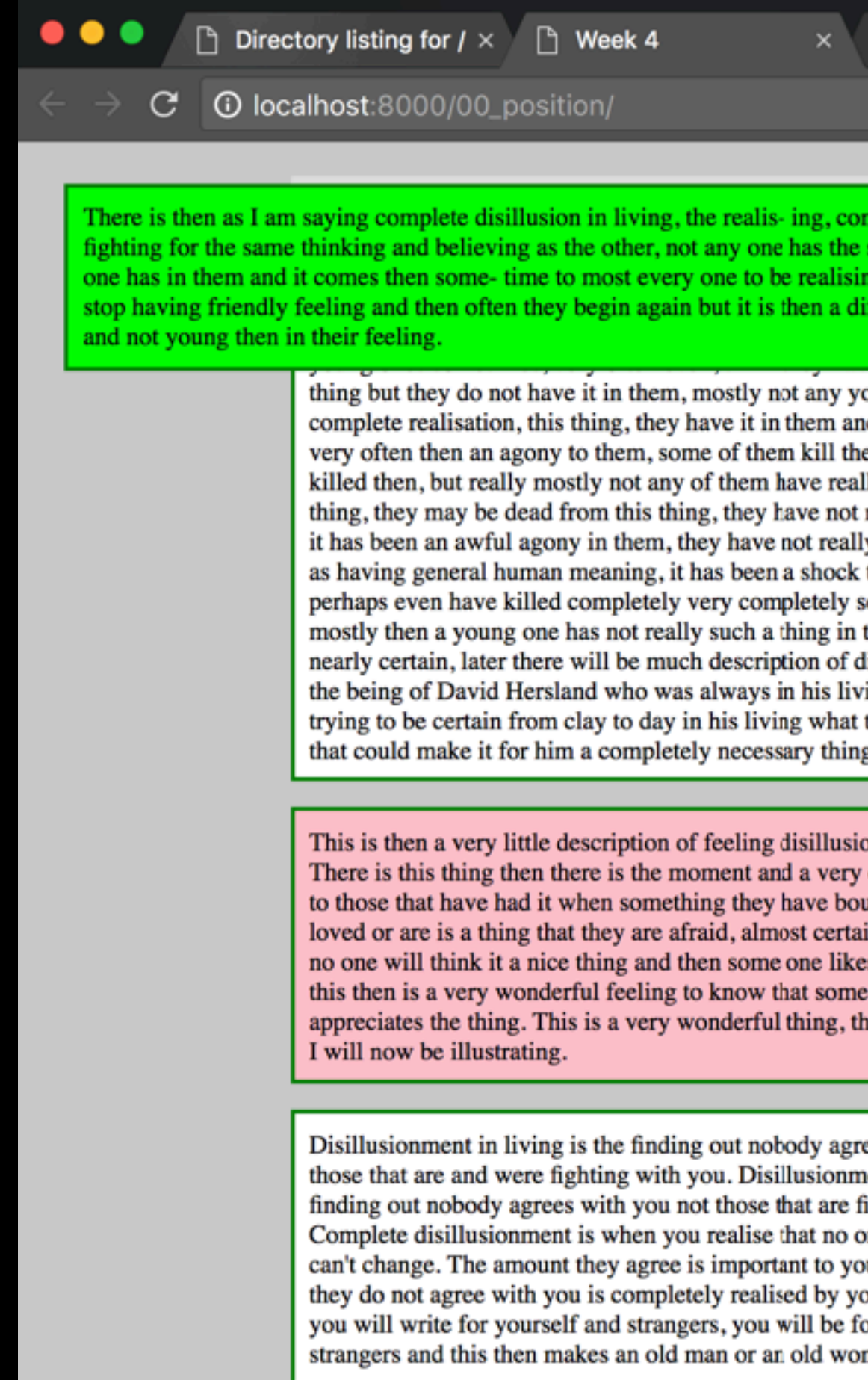


# Overlapping Elements

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 from 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 higher 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;
```

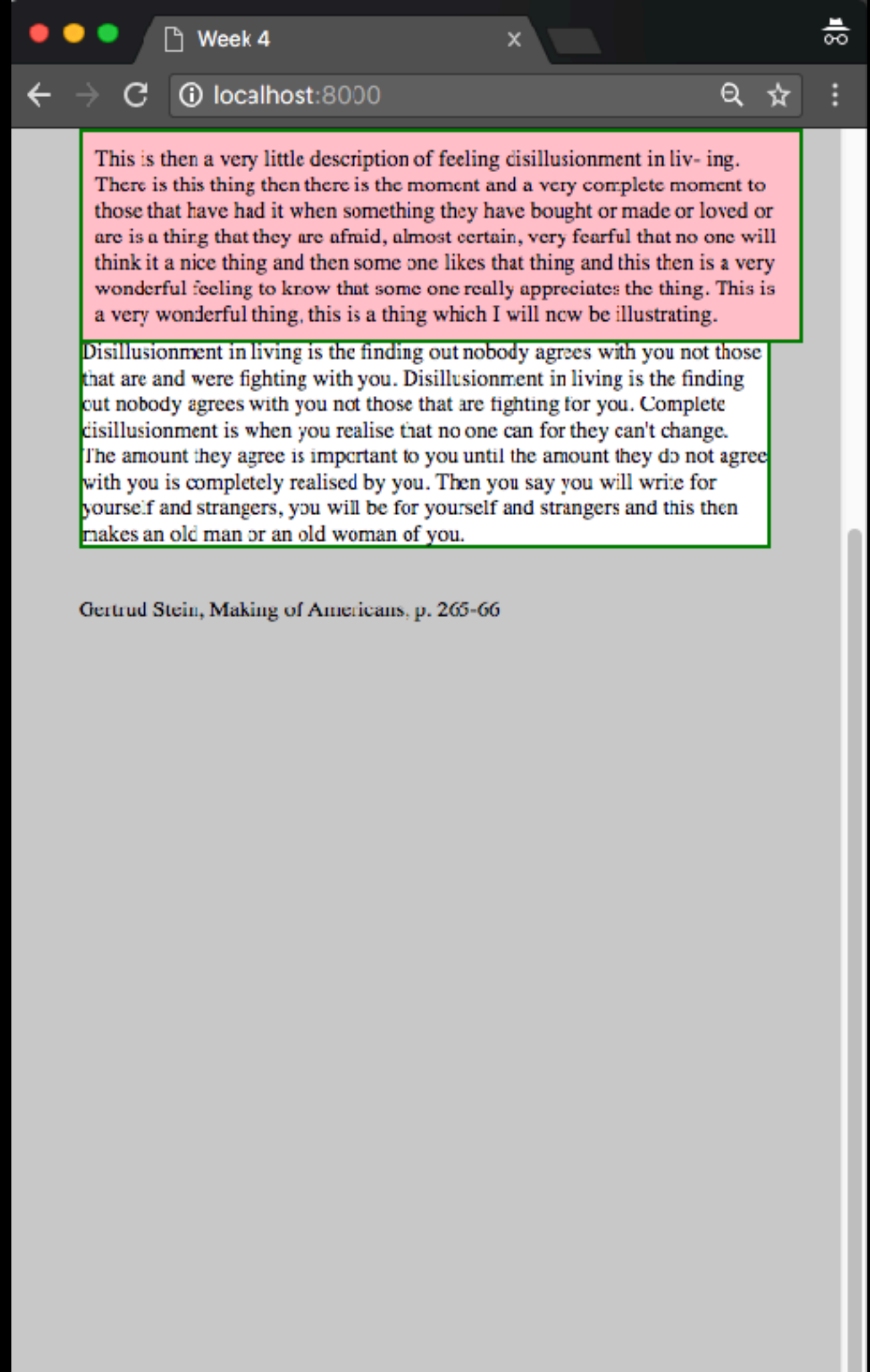
```
top: 0;
```

```
width: 50px;
```

```
margin: 0 auto;
```

```
padding: 10px;
```

```
}
```





## Position Sticky

