

CSS works by associating rules with **HTML** elements. These rules govern how the content of specified elements should be displayed.

A **CSS** rule contains two parts: a **selector** and a **declaration**.

**** pro tip:** It takes 5% to learn how to write CSS rule and 95% to learn different properties that you can use.

intro 2 CSS

* which is super awesome

**bc CSS is all about:

DESIGN



Metadata: `viewport`

The user's visible area of a web page

HTML5 introduced a method to let web designers take control over the viewport, through the `<meta>` tag.

<!

- - Tells the browser to match the device's width for the viewport
- Sets an initial zoom value -->

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

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



without



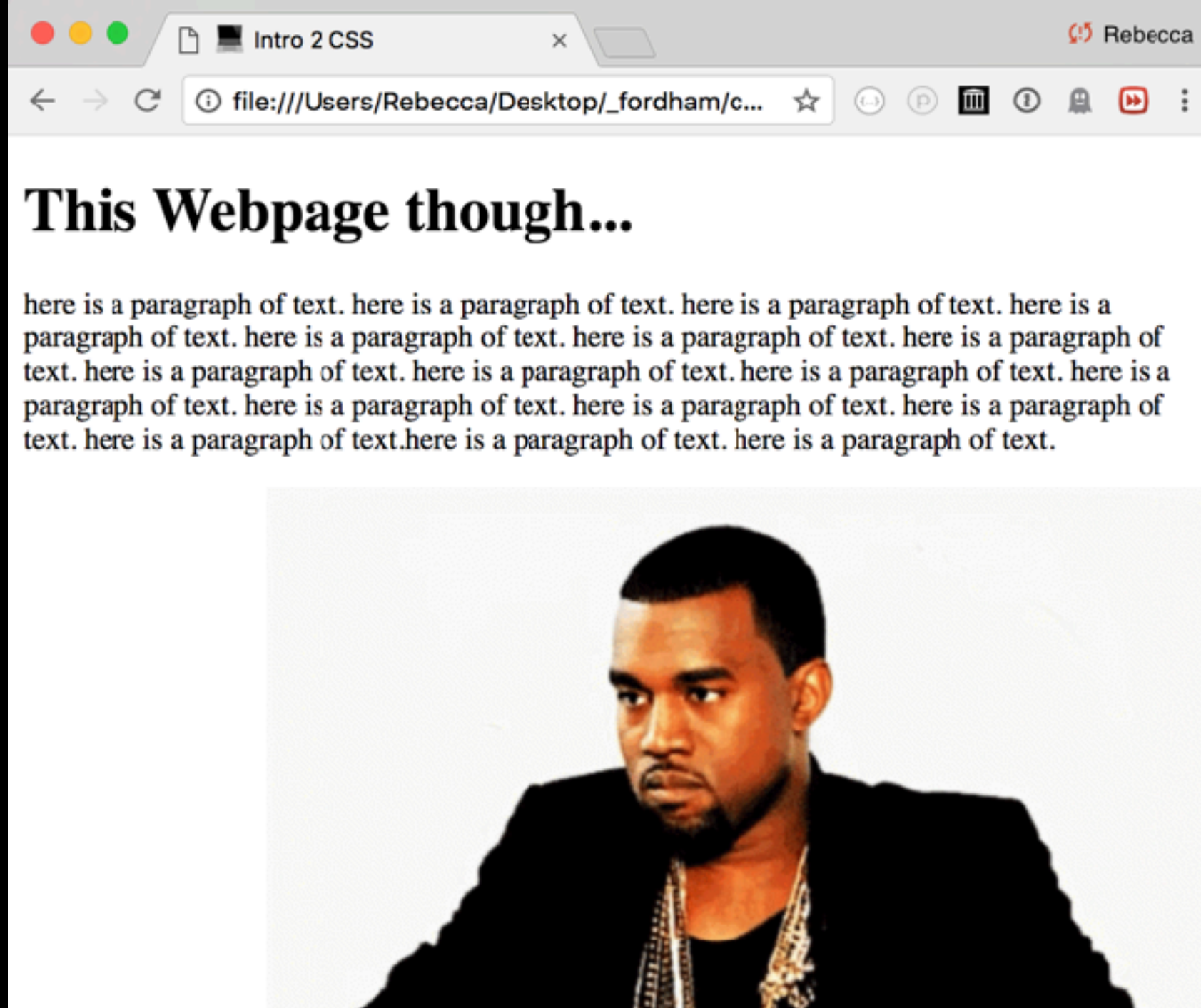
with

CSS works by associating rules with HTML elements. These rules govern how the content of specified elements should be displayed.

A CSS rule contains two parts: a **selector** and a **declaration**.

It takes 5% to learn how to write CSS rule and 95% to learn different properties that you can use.

the world w/out css



The key to understanding how **CSS** works is to imagine that there is an invisible box around every **HTML** element.

Block level elements are outlined w/ red + inline elements in green.

<body> creates 1st box, then **<h1>**, **<h2>**, **<p>**, **<i>** + **<a>** each create their own boxes within it.

The Cottage Garden

The *cottage garden* is a distinct style of garden that uses an informal design, dense plantings, and a mixture of ornamental and edible plants.

The Cottage Garden originated in England and its history can be traced back for centuries, although they were re-invented in 1870's England, when stylized versions were formed as a reaction to the more structured and rigorously maintained English estate gardens.

The earliest cottage gardens were more practical than their modern descendants, with an emphasis on vegetables and herbs, along with some fruit trees.

The Cottage Garden

The *cottage garden* is a distinct style of garden that uses an informal design, dense plantings, and a mixture of ornamental and edible plants.

The Cottage Garden originated in England and its history can be traced back for centuries, although they were re-invented in 1870's England, when stylized versions were formed as a reaction to the more structured and rigorously maintained English estate gardens.

The earliest cottage gardens were more practical than their modern descendants, with an emphasis on vegetables and herbs, along with some fruit trees.

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.

Box Model



Content

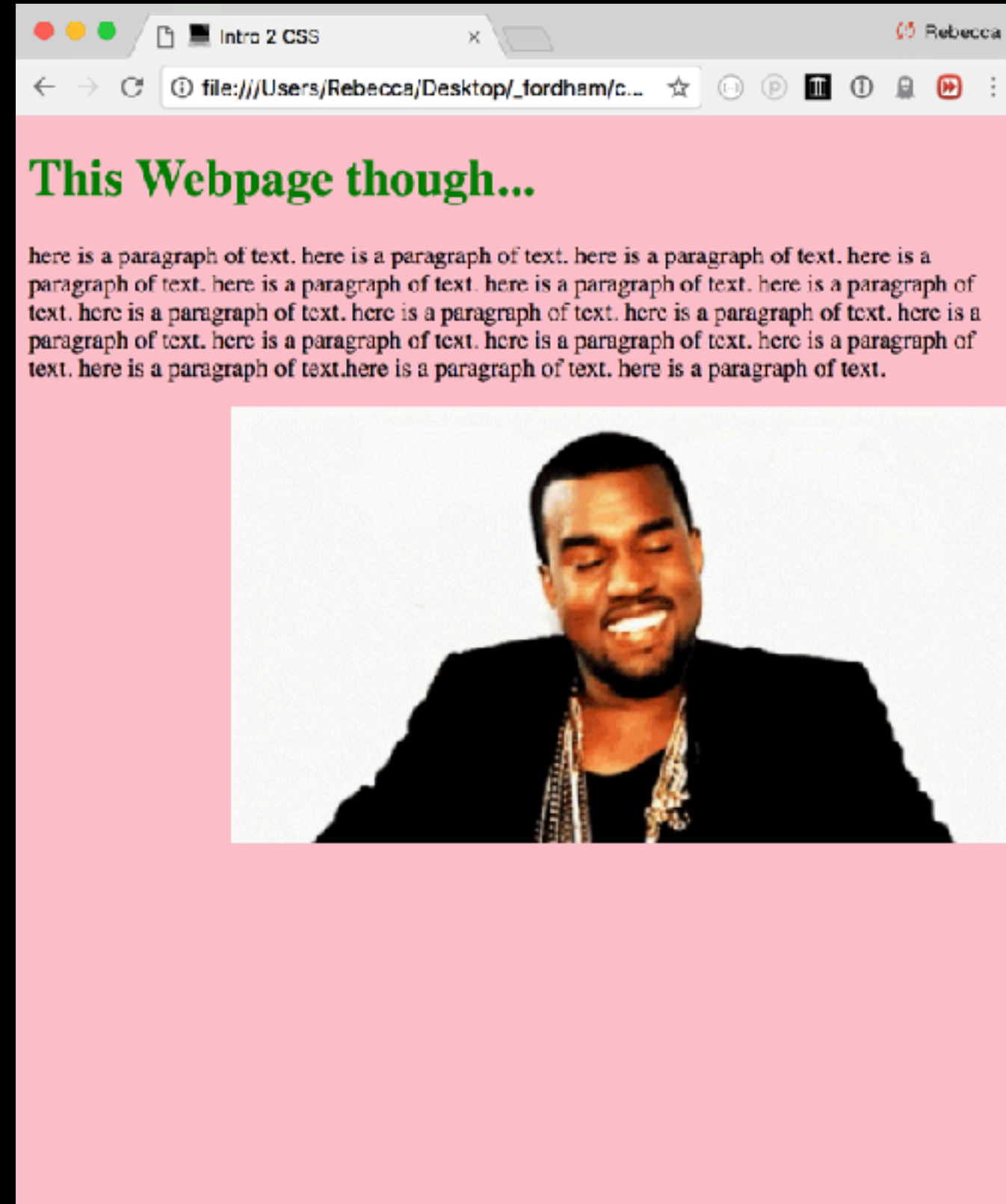
You can write CSS 3 Different Ways:

Inline Styles

```
<h1 style="color:green;">This Webpage though...</h1>  
<body style="background-color: pink;">
```

Embedded Styles

Externals Styles



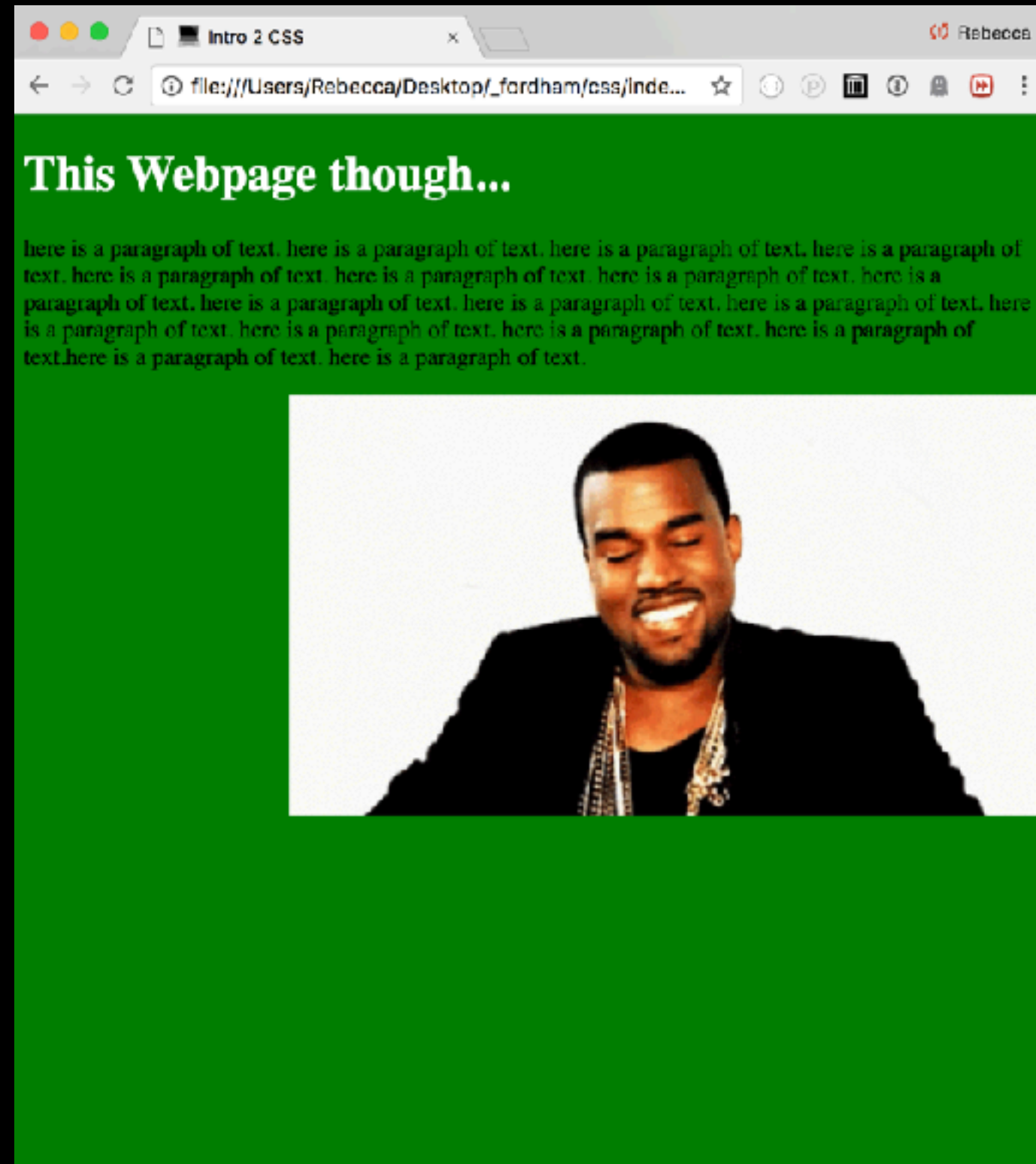
Inline Styles

```
<h1 style="color:white;">This Webpage though...</h1>  
<body style="background-color: green;">
```

Embedded Styles

```
<html>  
  <head>  
    <title>🖥 Intro 2 CSS </title>  
    <style type="text/css">  
      h1 {  
        color: white  
      }  
  
      body {  
        background: green;  
      }  
    </style>  
  </head>
```

Externals Styles



Inline Styles

```
<h1 style="color:#FF4500;">This Webpage though...</h1>  
<body style="background-color: #000080;">
```

Embedded Styles

```
<html>  
  <head>  
    <title>🖥 Intro 2 CSS </title>  
    <style type="text/css">  
      h1 {  
        color: #FF4500  
      }  
  
      body {  
        background: #000080;  
      }  
    </style>  
  </head>
```

External Styles *

```
<head>  
  <title>🖥 Intro 2 CSS </title>  
  <link rel="stylesheet" type="text/css" href="theStyle.css">  
</head>
```



* Why Use External Style Sheet:

Same CSS for each HTML page

No need to copy or rewrite styling

Make changes to CSS automatically + can apply to the entire website

Faster download time for subsequent pages

style.css

CSS SYNTAX:

selectors are used to find (select) HTML elements based on their element name, id, etc...

```
h1 {  
  color: #FF4500  
}  
  
body {  
  background: #000080;  
}
```

selector { declaration }

```
selector {  
  
  property: value ;  
  
}
```



Selector is a term such as **p**, **h1**, **div** that identifies the HTML element you want to format or apply a rule to. You can add multiple selectors in a declaration.

Selector

Meaning

Example

Universal Selector

Applies to all elements in the document

***** { }

Type Selector

Matches element names

h1, h2, h3 { }

Class Selector

Matches an element whose class attribute has a value that matches the one specified after the period (or full stop) symbol

.note { }

targets any element whose class attribute has a value of "note"

p.note { }

targets only **<p>** elements whose class attribute has a value of "note"

ID Selector

Matches an element whose id attribute has a value that matches then specified after the # symbol

#introduction { }

targets the element whose id attribute has value of "introduction"

Selector

Meaning

Example

Child Selector

Matches an element that is a direct child of another

`li > a { }`

targets any `<a>` element that are children of an `` element (but not other `<a>` elements in the page).

Descendant Selector

Matches an element that is a descendent of another specified element (not just a direct child of that element)

`p a { }`

targets any `<a>` elements that sit inside a `<p>` element, even if there are other elements nested btw them

Selector

Meaning

Example

Adjacent Sibling Selector

Matches an element that is the next sibling of another

h1+p { }
targets the first **<p>** element after any **<h1>** element (but not other **<p>** elements)

General Sibling Selector

Matches an element that is a sibling of another, although it does not have to be the directly preceding element

h1~p { }
tif you have two **<p>** elements that are siblings of an **<h1>** element, this rule would apply to both

```
/* type/element selector */
```

```
p {  
  color: blue;  
}
```

```
/* class attribute selector */
```

```
.blue-text {  
  color: blue;  
}
```

```
/* id attribute selector */
```

```
#blue-par {  
  color: blue;  
}
```

```
/* BONUS: grouping  
selector */
```

```
p,  
.blue-text,  
#blue-par {  
  color: blue;  
}
```

selecting multiple elements:

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

```
    color: red;
```

```
    background-color: blue;
```

```
    width: 500px;
```

```
}
```

```
p,
```

```
li {
```

```
    background-color: red;
```

```
    font-color: blue;
```

```
}
```

HTML comments are written like this

```
<!-- This is a comment -->
```

CSS comments are written like this

```
/* This is a comment */
```

```
{  
text-align:  
  
    left ;  
    right ;  
    center ;  
    justify ;  
  
}
```

```
{  
vertical-align:  
  
    baseline ;  
    sub ;  
    super ;  
    top ;  
    text-top ;  
    middle ;  
    bottom ;  
    text-bottom ;  
}
```

This property is NOT intended to allow you to vertically align text in the middle of a block level elements such as `<p>` + `<div>`, although it does have this effect when used with table cells `<td>` + `<th>` elements.

It is more commonly used w/ inline elements such as ``, `` or ``. When used with these elements, it performs a task very similar to the HTML align attribute used on the `` element.

a: link {

a: visited {

: hover { Applied when a user hovers over an element w/ a mouse. This changes the appearance of links and buttons when a user places their cursor over them. Does not work on mobile.

: active { Applied when an element is being activated by a user, like when a button is pressed or a link clicked. This added to UX. Applied when an element has focus. Any thing you can interact with.

: focus { Focus occurs when a browser discovers that you are ready to interact w/ an element. For example when yr cursor is in an input - that element is said to have focus.

}

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 positioning

Controlling the Position of Elements

Normal Flow

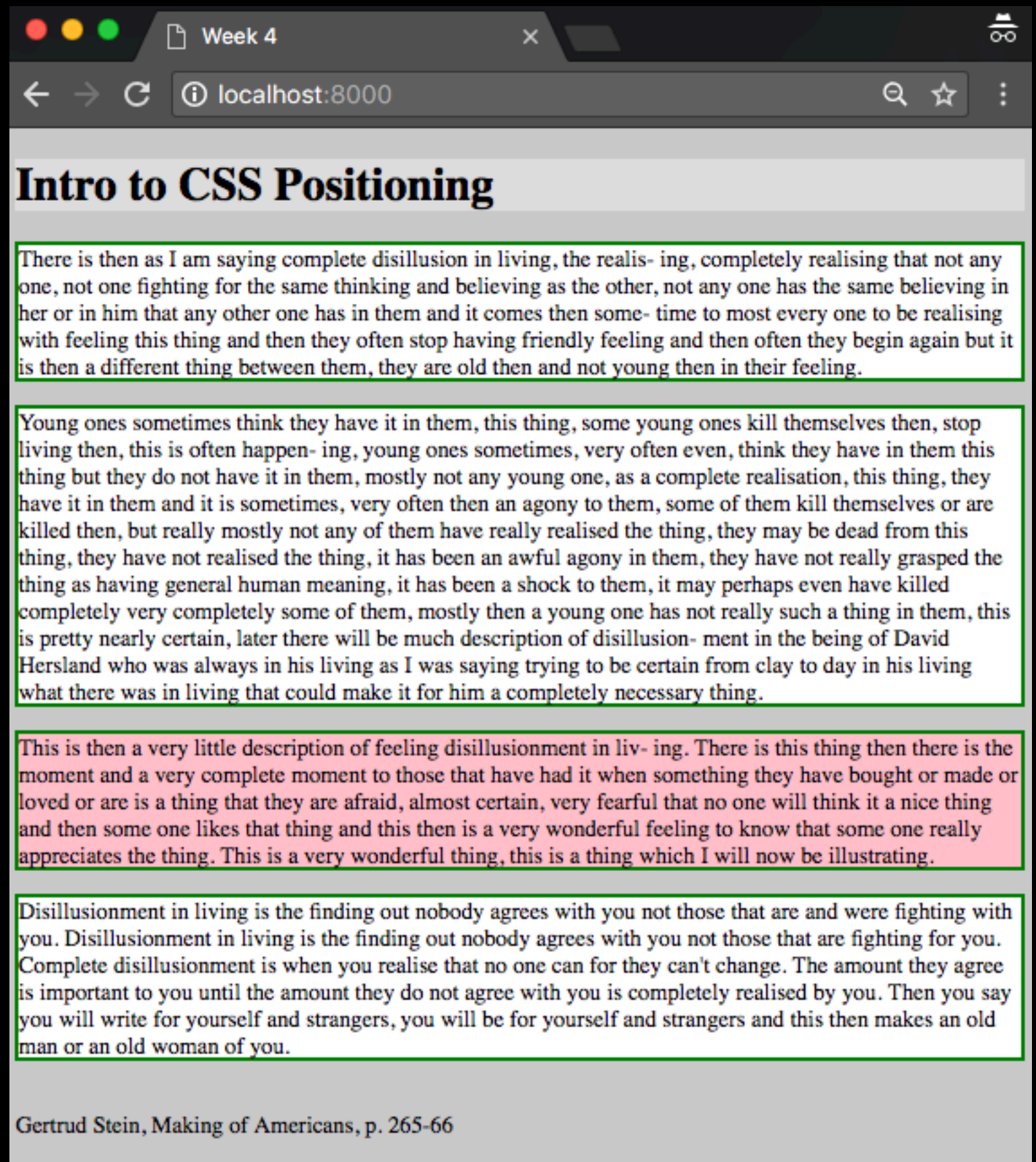
Every block-level element appears on a new line, causing each item to appear lower down on the page. Even if you specify the width of the boxes + there is space they will not appear next to each other.

```
.thePosition {
```

```
background: pink;
```

```
position: static;
```

```
}
```

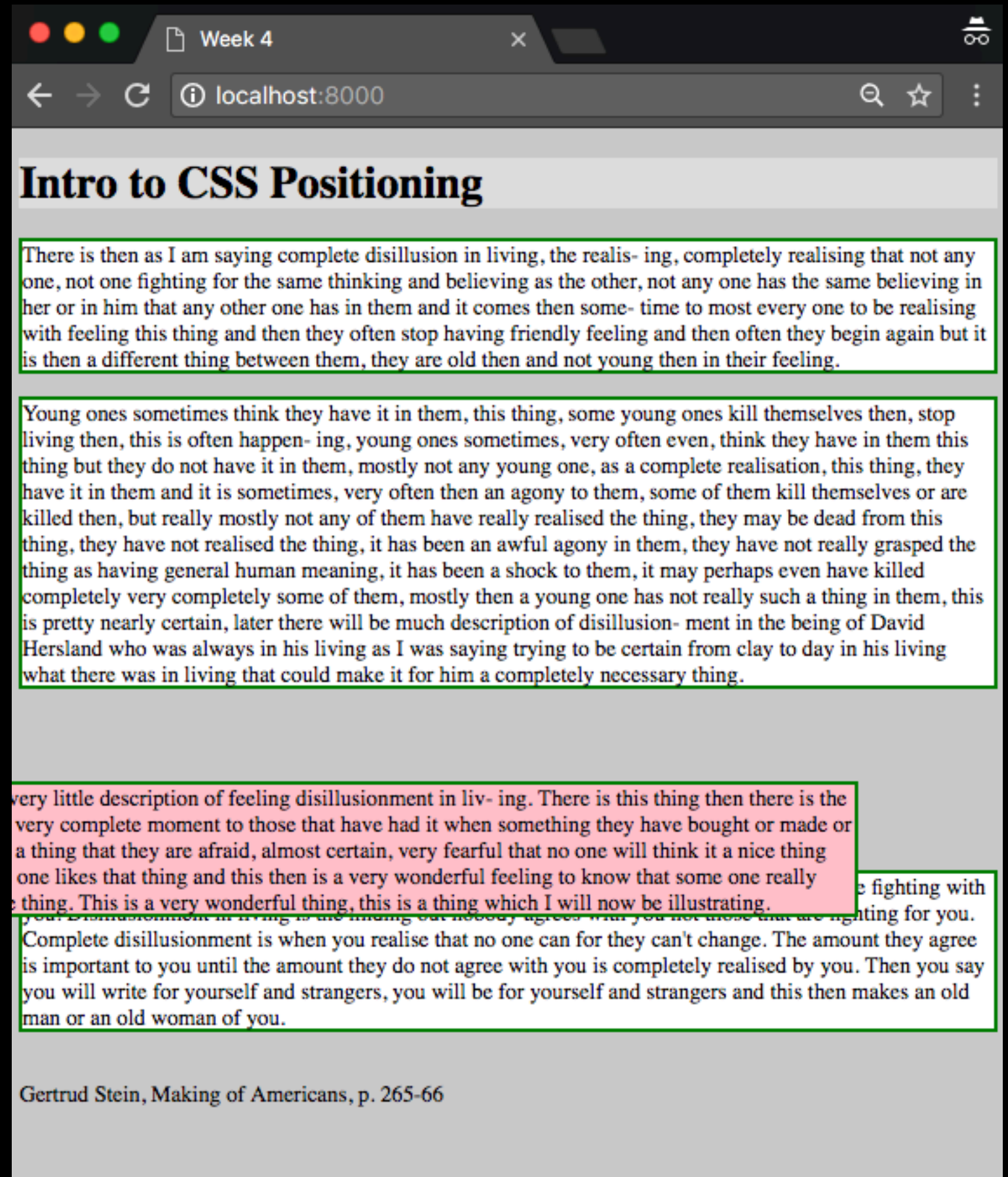


Relative Positioning

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 {
```

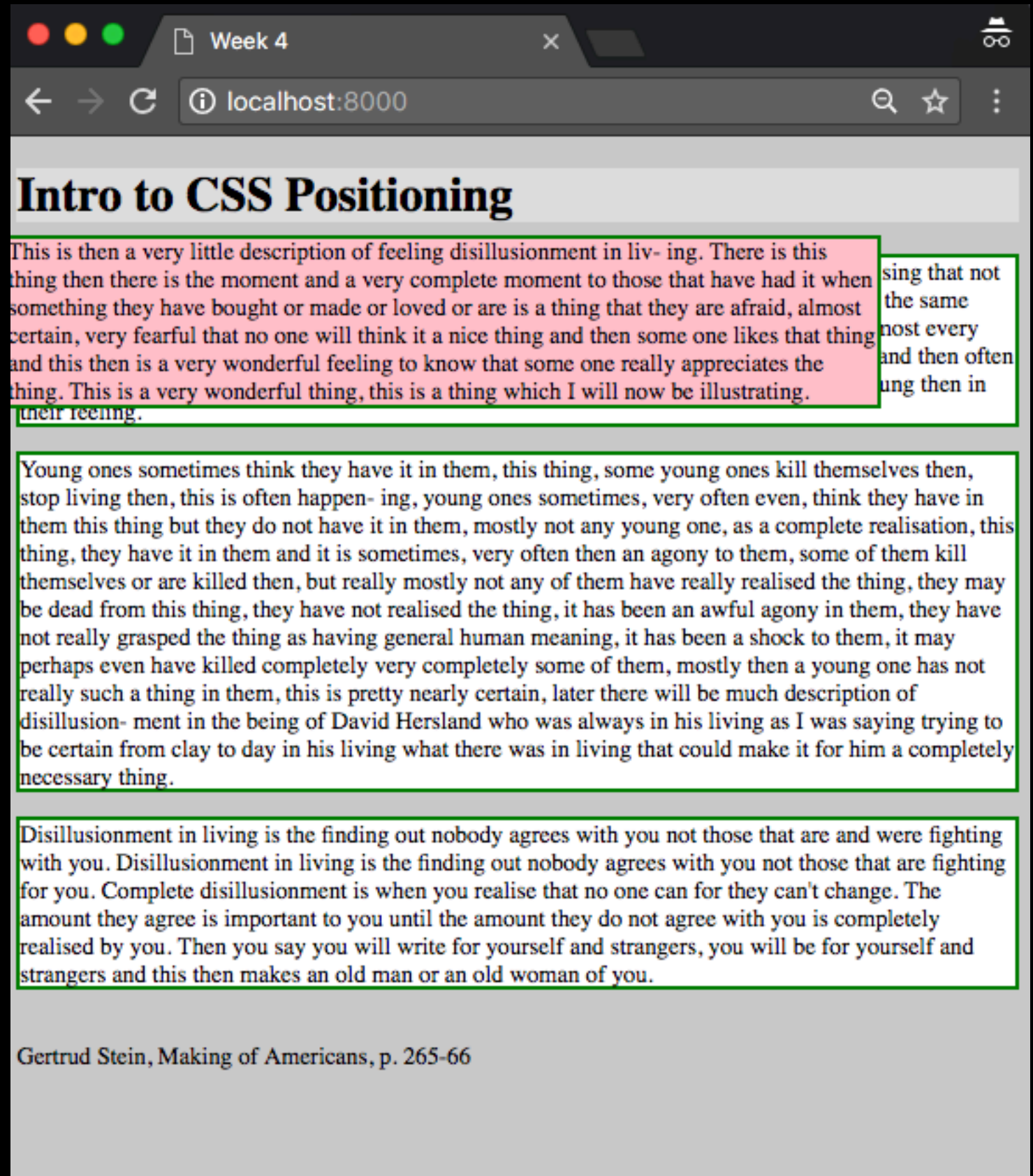
```
background: pink;
```

```
position: absolute;
```

```
top: 50px;
```

```
right: 100px;
```

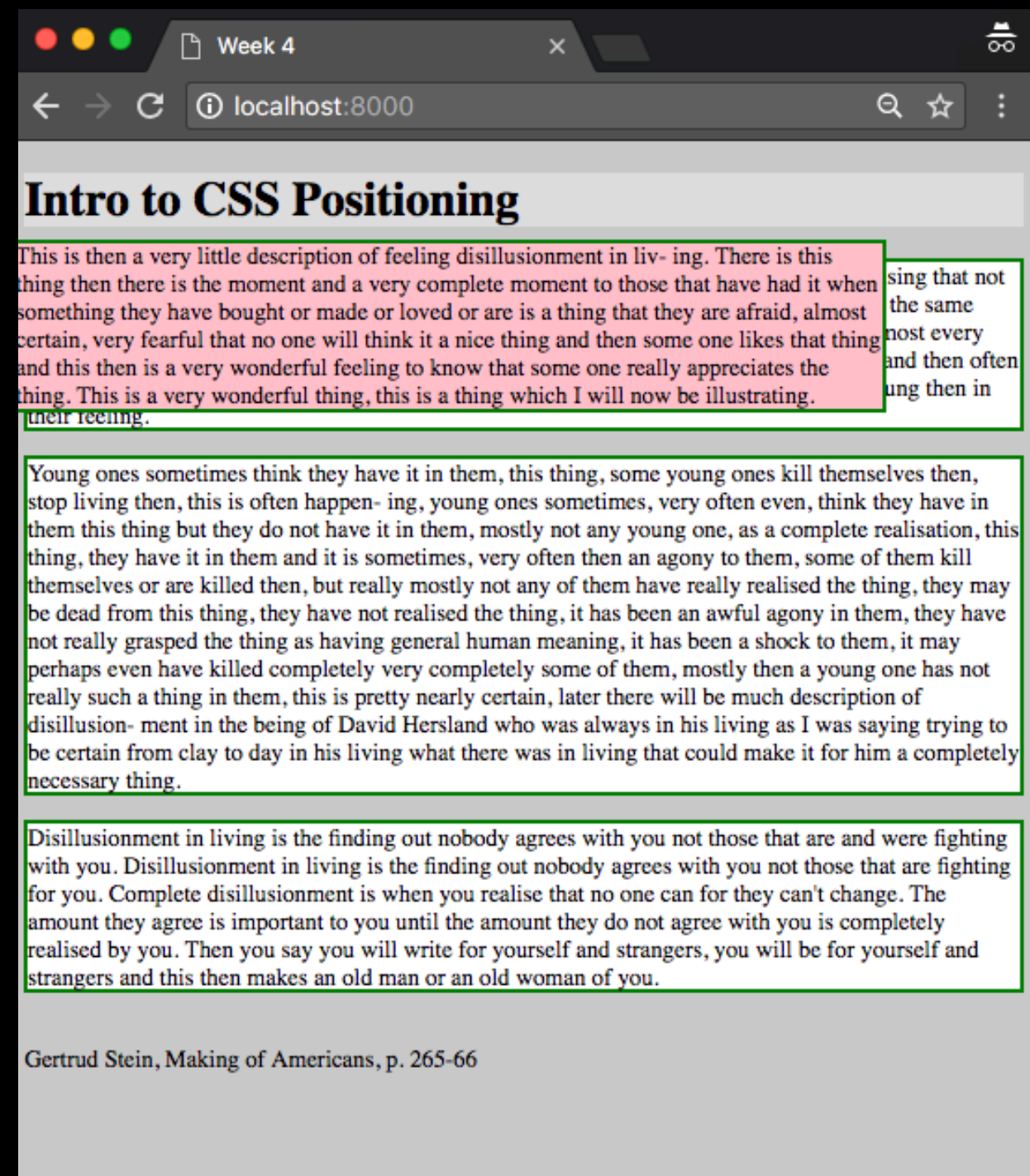
```
}
```

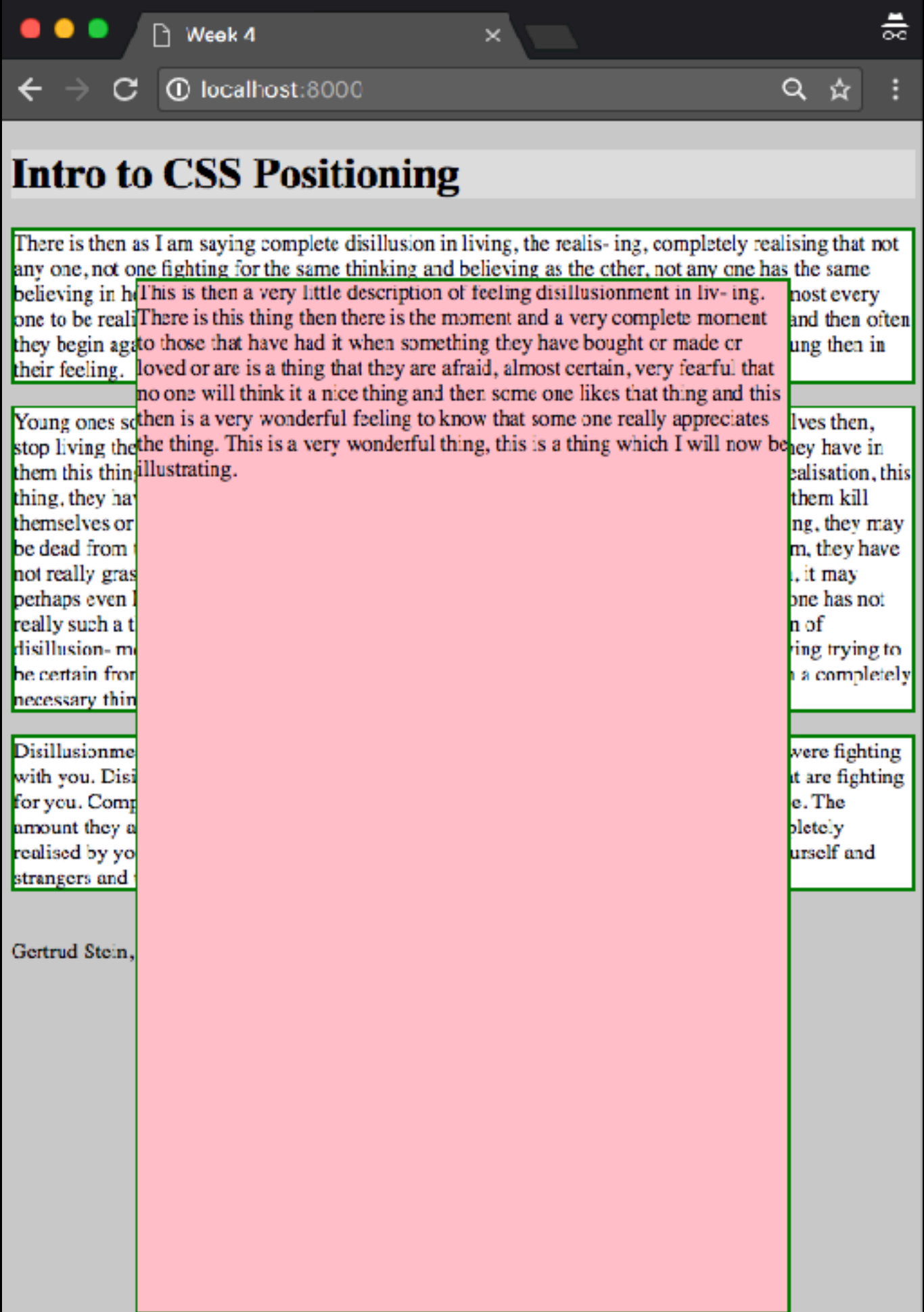


Notice that the position of the element has changed — this is because **top**, **bottom**, **left**, and **right** behave in a different way with absolute positioning.

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

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 Hirschland 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.

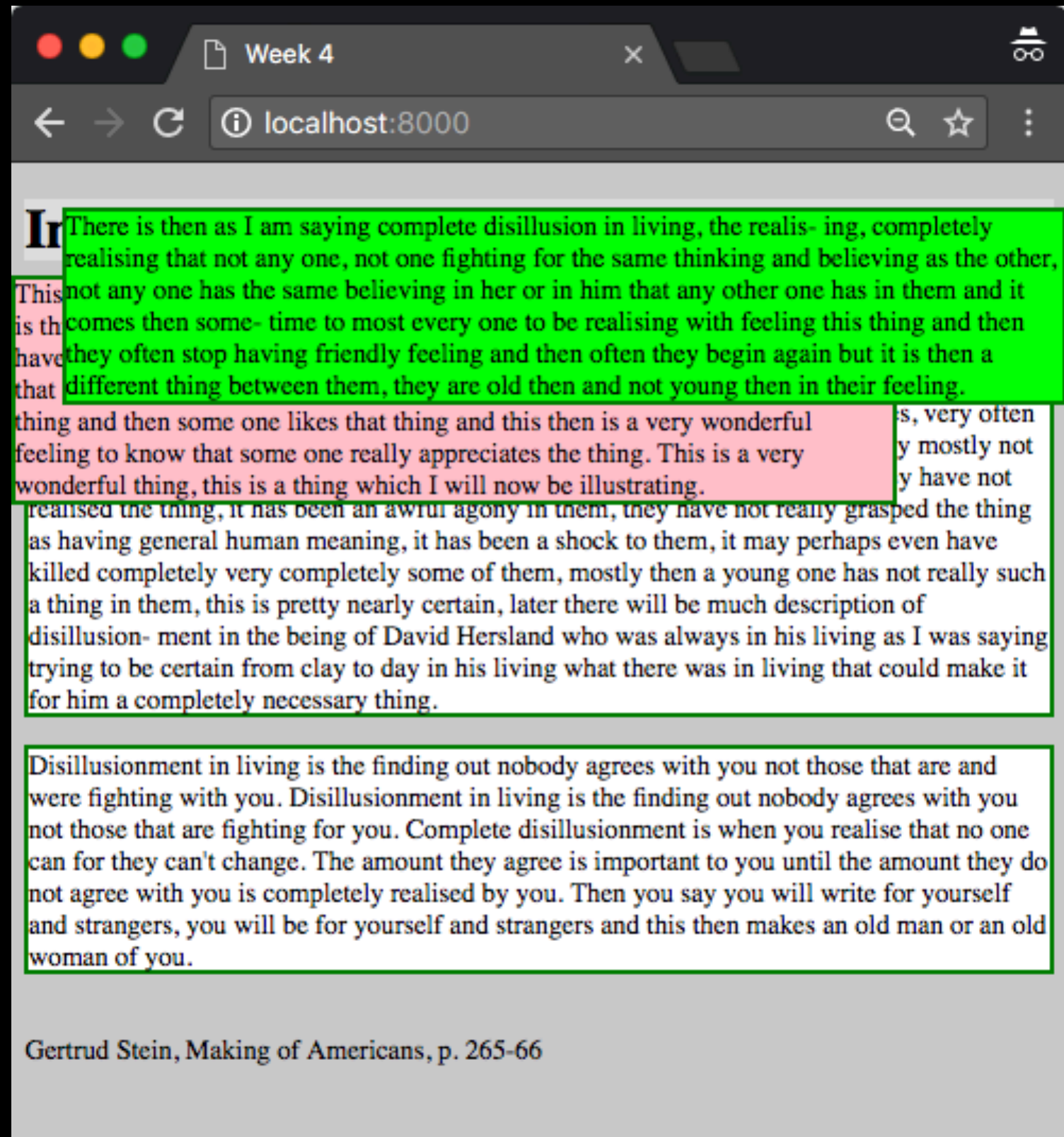
Gertrud Stein, Making of Americans, p. 265-66

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;  
}
```

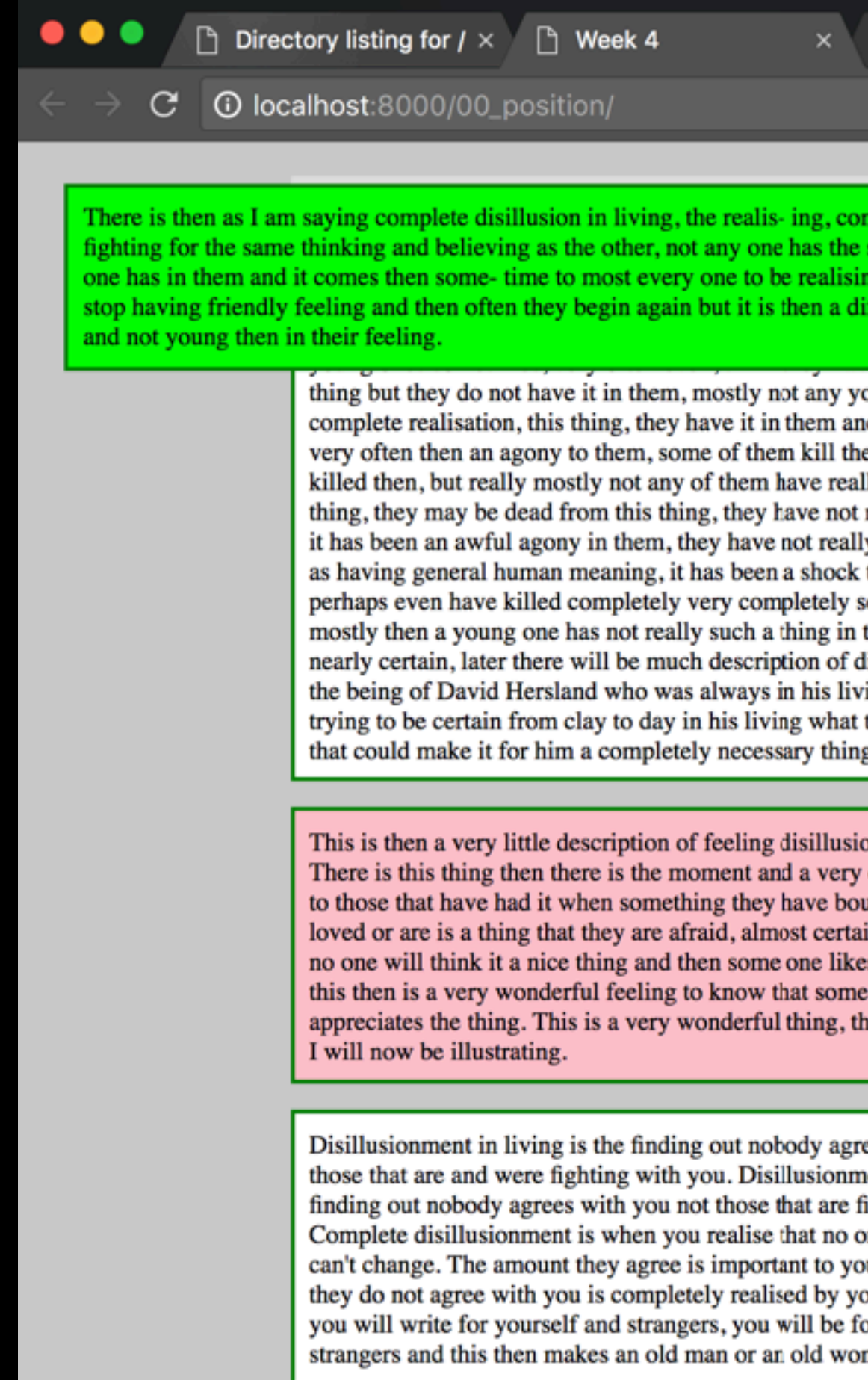


Overlapping Elements

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;
```

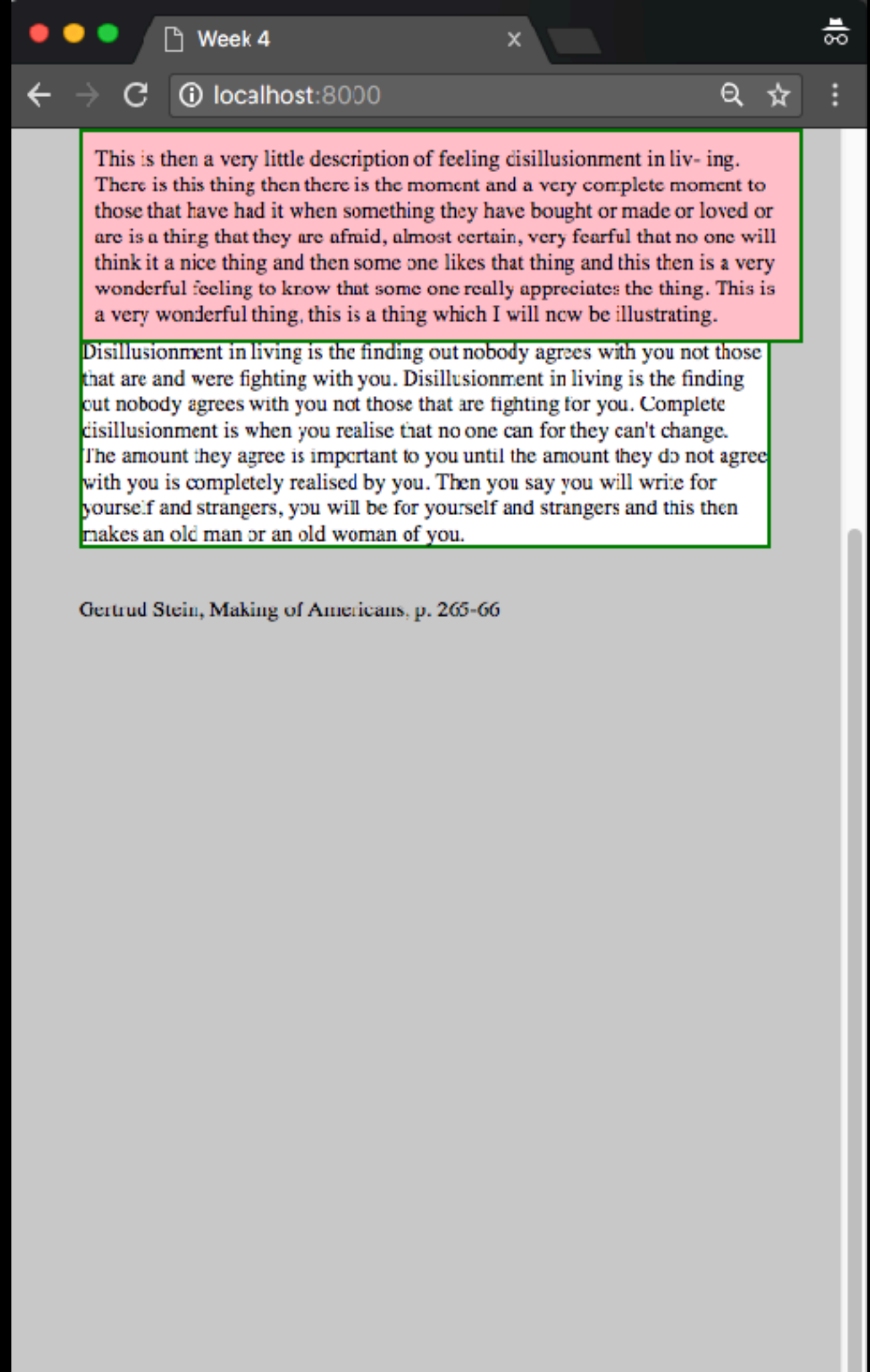
```
top: 0;
```

```
width: 50px;
```

```
margin: 0 auto;
```

```
padding: 10px;
```

```
}
```



Position Sticky

