

`<p>`

`<h1>` - `<h6>`

`<article>`

A document, page or site. This is usually a root container element after body

`<section>`

Generic section of a document

`<header>`

Intro section of a document

`<footer>`

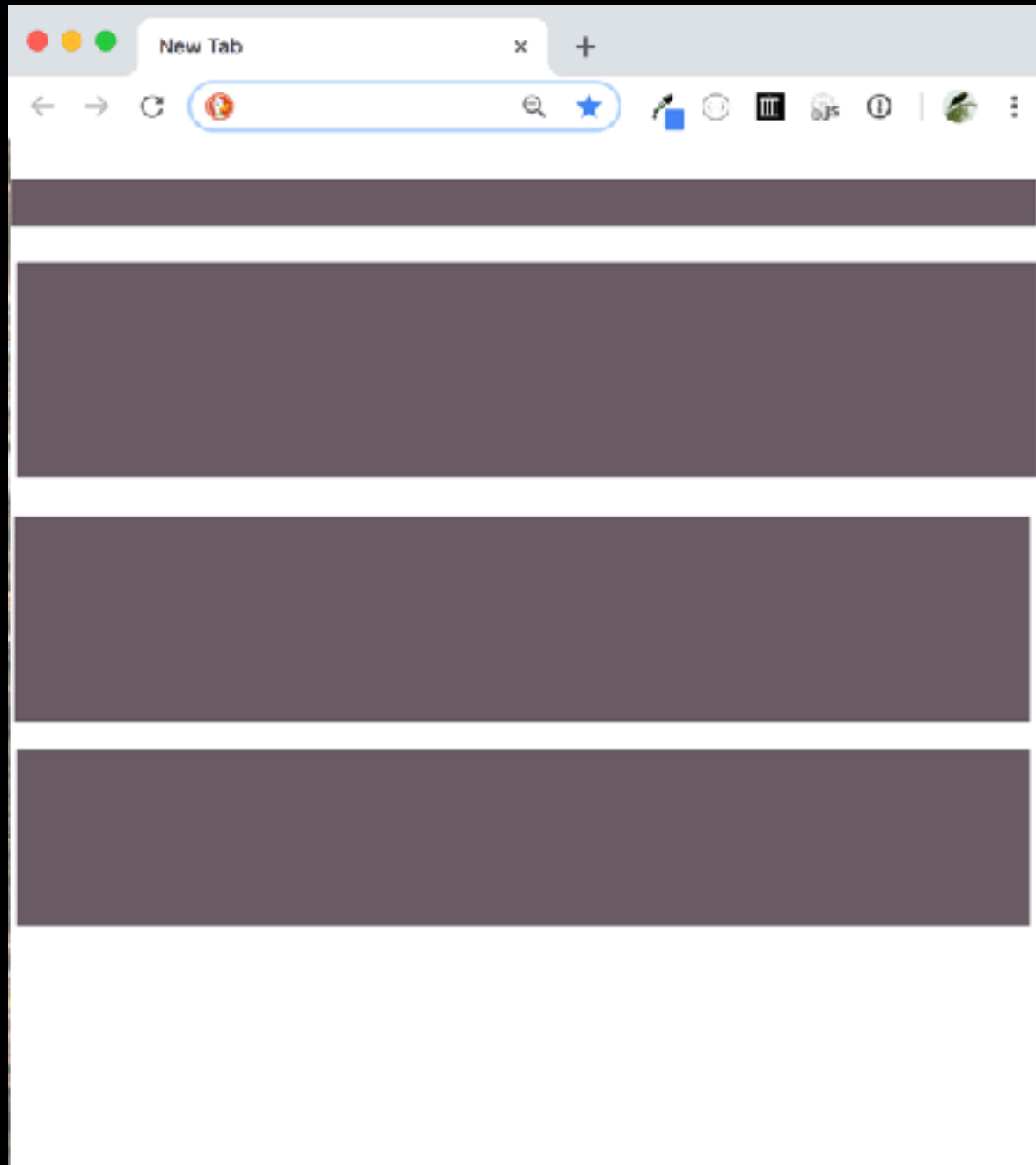
Footer at end of a document or section

`<nav>`

Navigational section

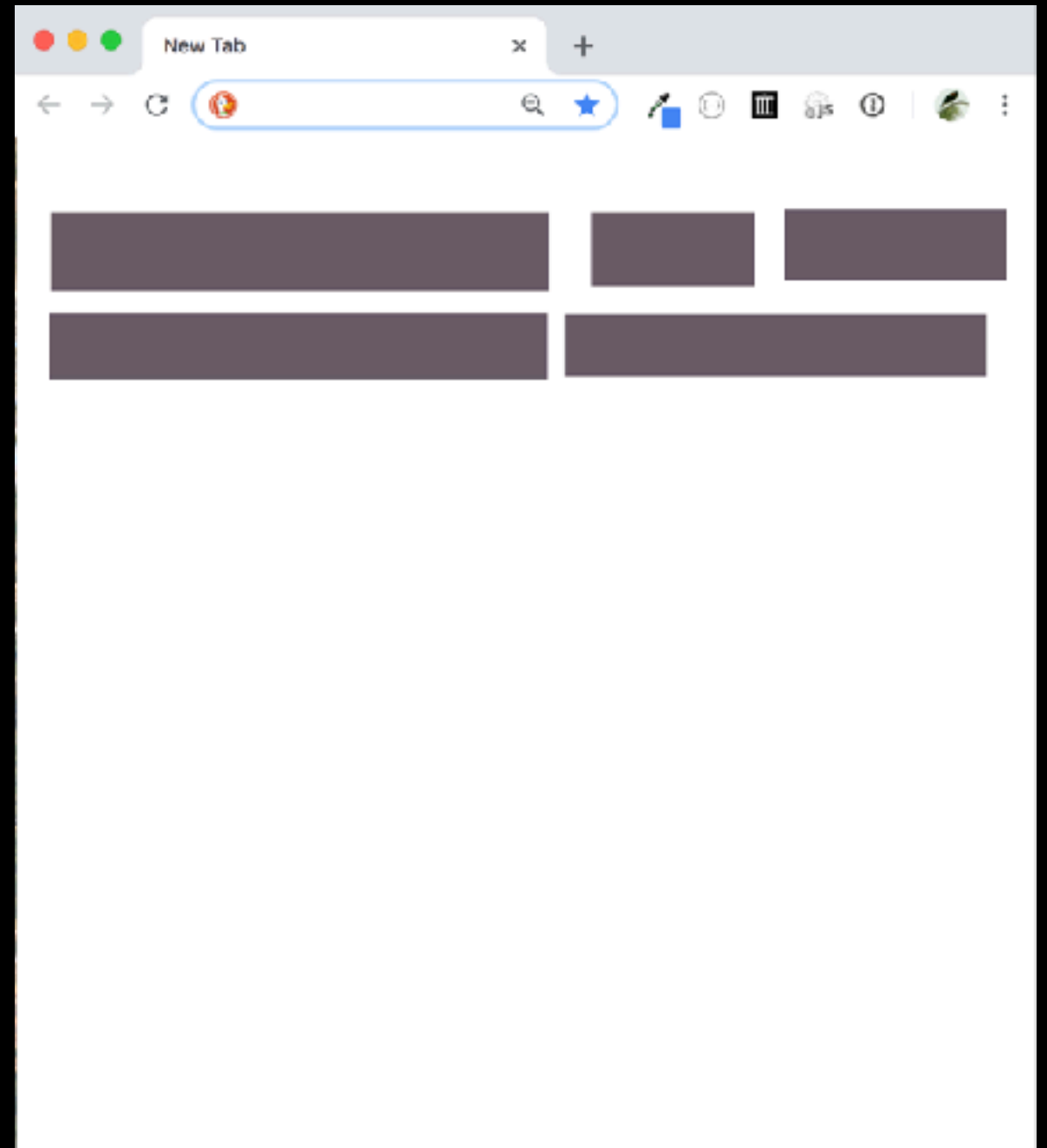
Use these **before** div when appropriate.

css Box Model



Block layout

Laying out large sections of a page



Inline layout

Laying out txt + other inline content within a section

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

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](#)

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;  
}
```

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

Can specify

padding-top

padding-bottom

padding-left

padding-right

There's also a shorthand:

padding: 2px 4px 3px 1px;

padding: 10px 2px;

<- top | right | bottom | left

<- top + bottom | left + right


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

```
.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;  
}
```

Lectures

Homework

Can specify

margin-top

margin-bottom

margin-left

margin-right

There's also a shorthand:

margin: 2px 4px 3px 1px;

margin: 10px 2px;

<- **top** | **right** | **bottom** | **left**

<- **top + bottom** | **left + right**

Margin

Margin is the space btw the border + other elements

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

Lectures

Homework

Q: Why doesn't this look more like this?

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

Lectures

Homework

Margin Collapsing

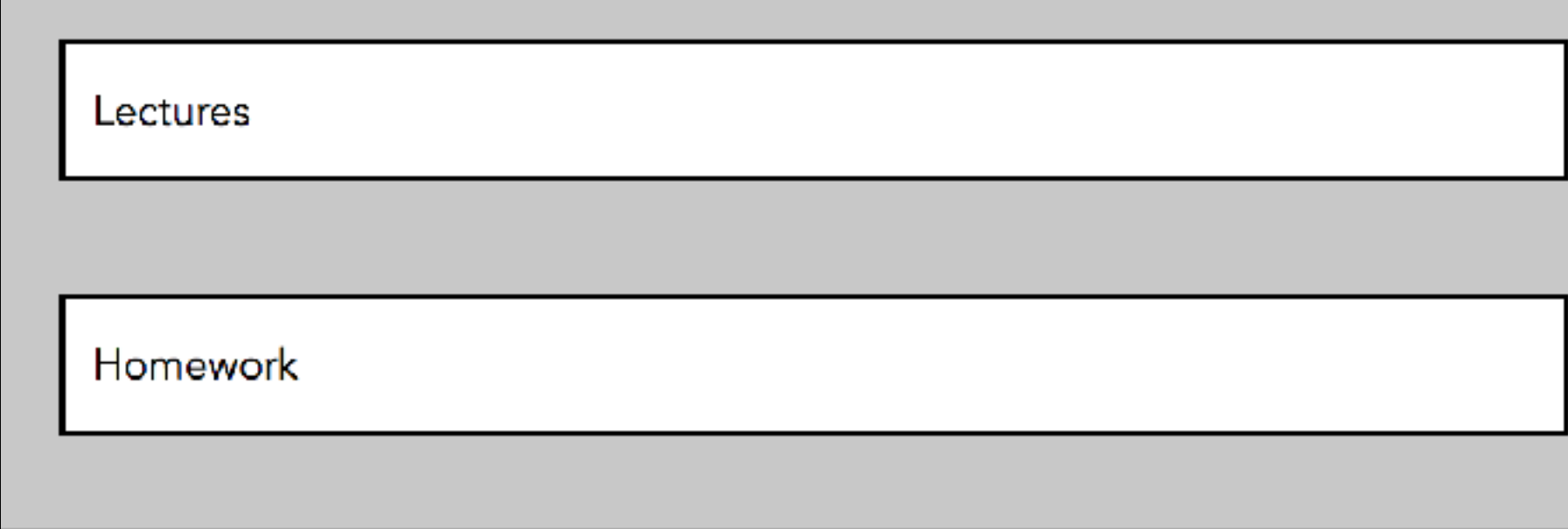
Sometimes the top + bottom margins of block elements are combined ("collapsed") into a single margin.

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?



Lectures

Homework

Negative Margin

Margins can be negative as well.

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

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

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

This is a box of text.

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.

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.

Inline Element Box Model

```
<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>
```

```
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

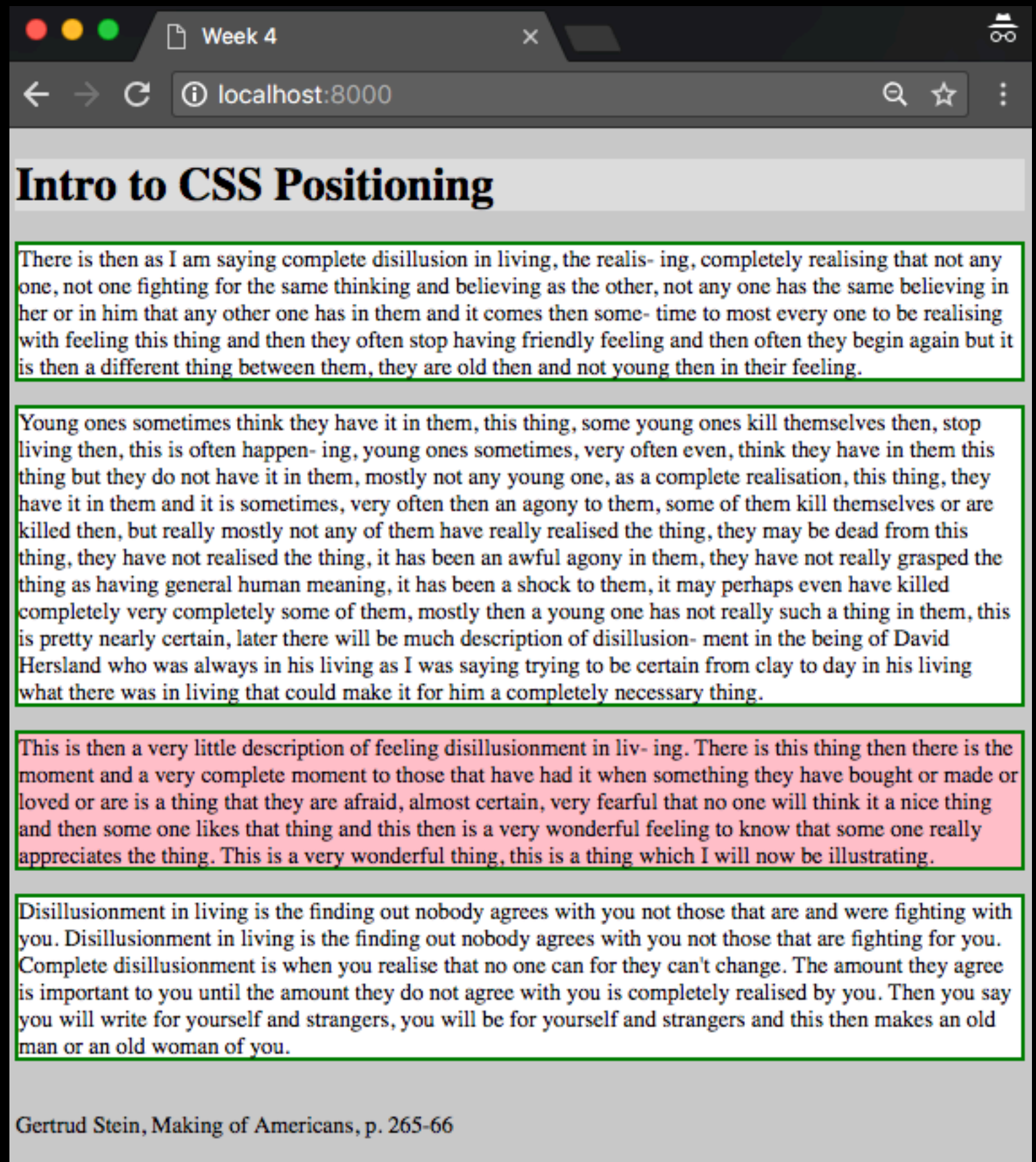
Every block-level element appears on a new line, causing each item to appear lower down on the page. Even if you specify the width of the boxes + there is space they will not appear next to each other.

```
.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 {
```

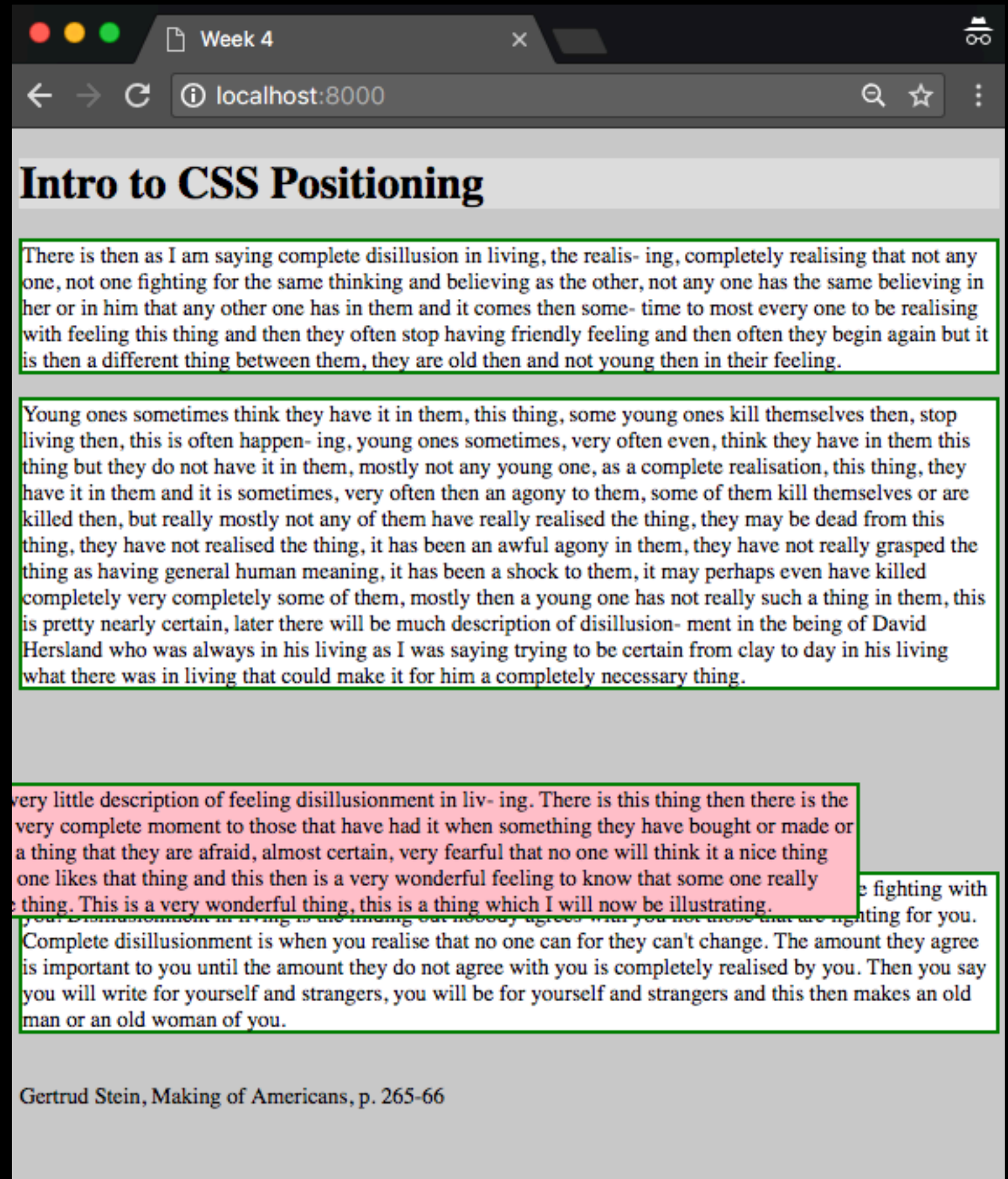
```
background: pink;
```

```
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 {
```

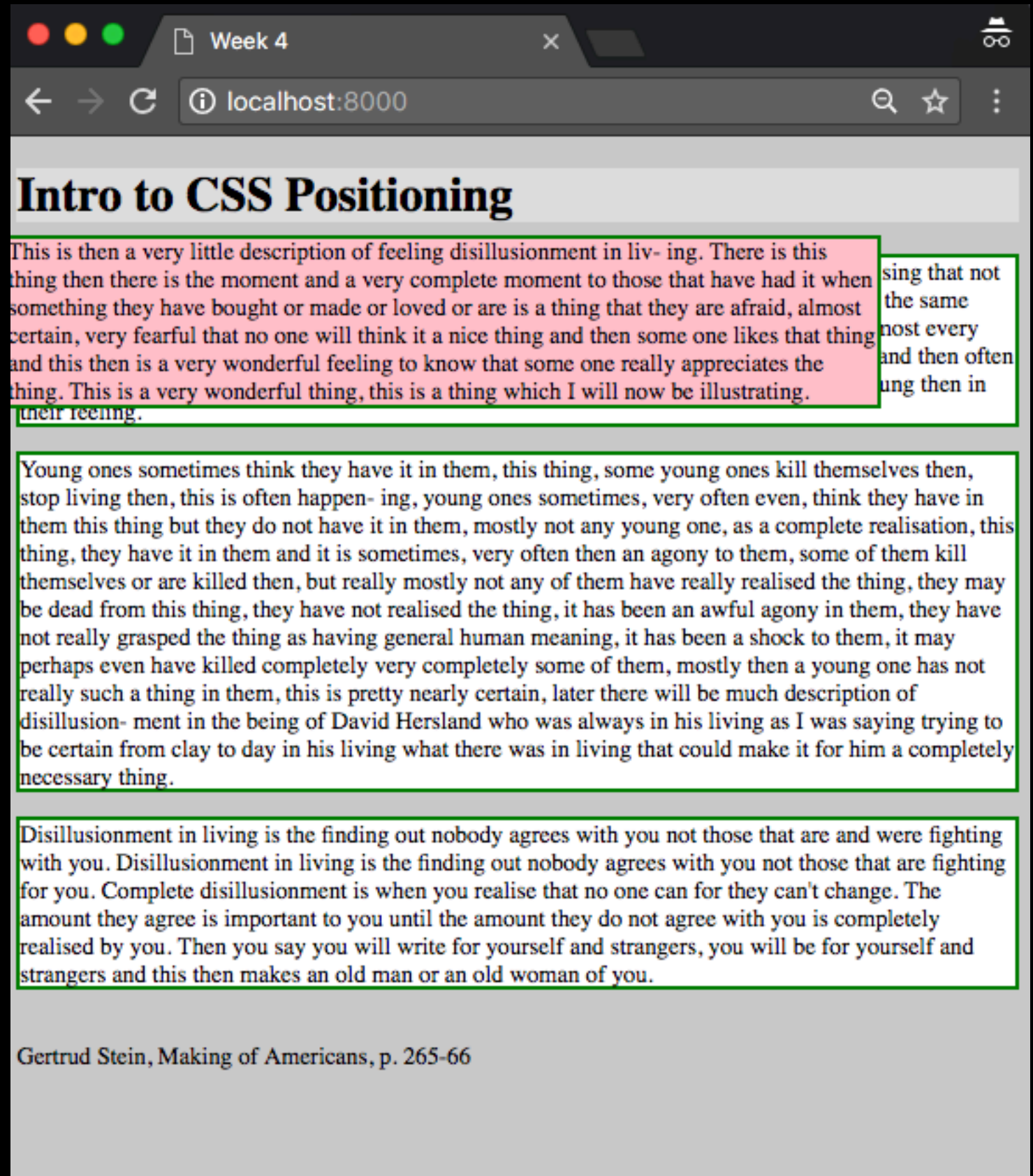
```
background: pink;
```

```
position: absolute;
```

```
top: 50px;
```

```
right: 100px;
```

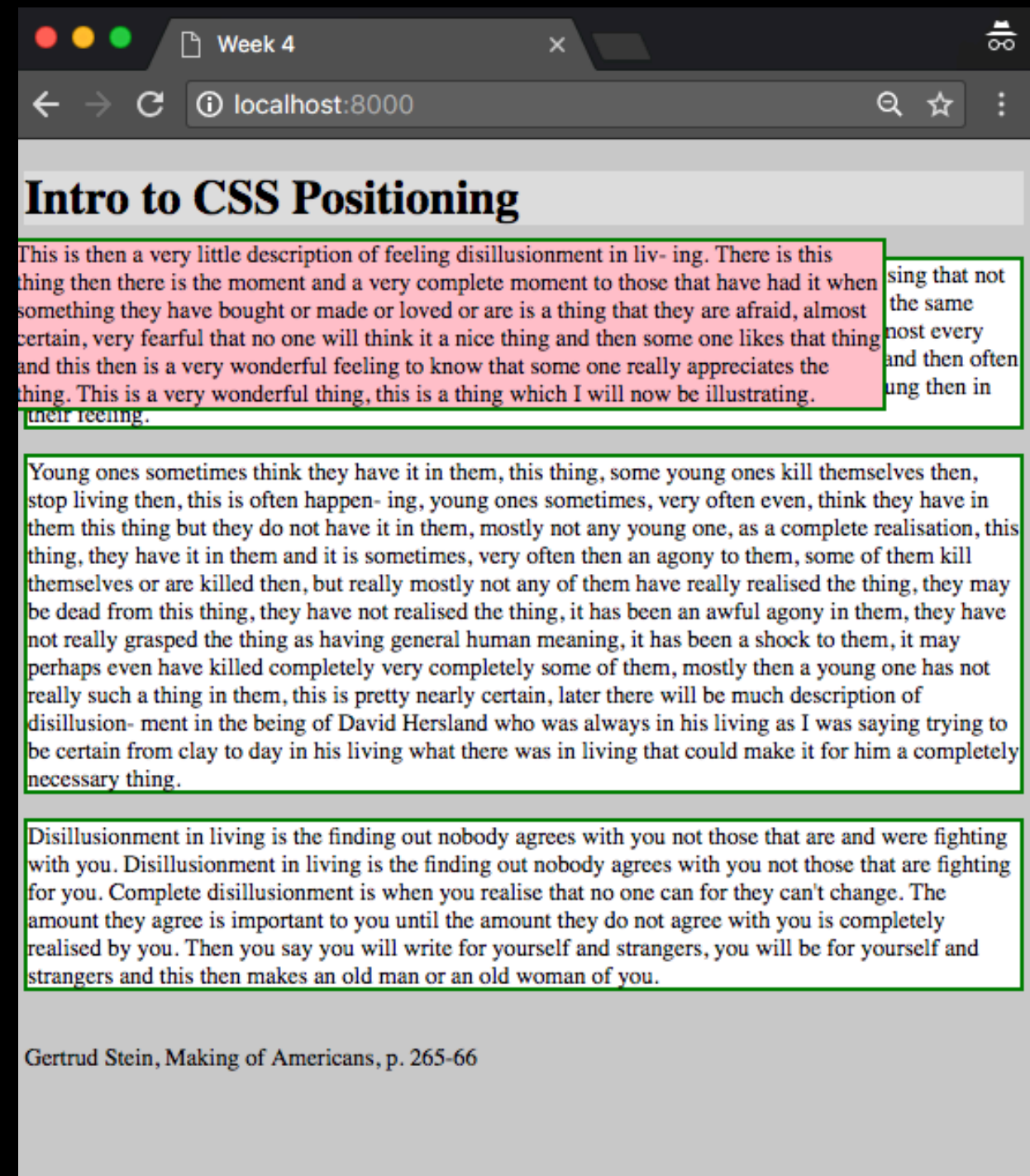
```
}
```

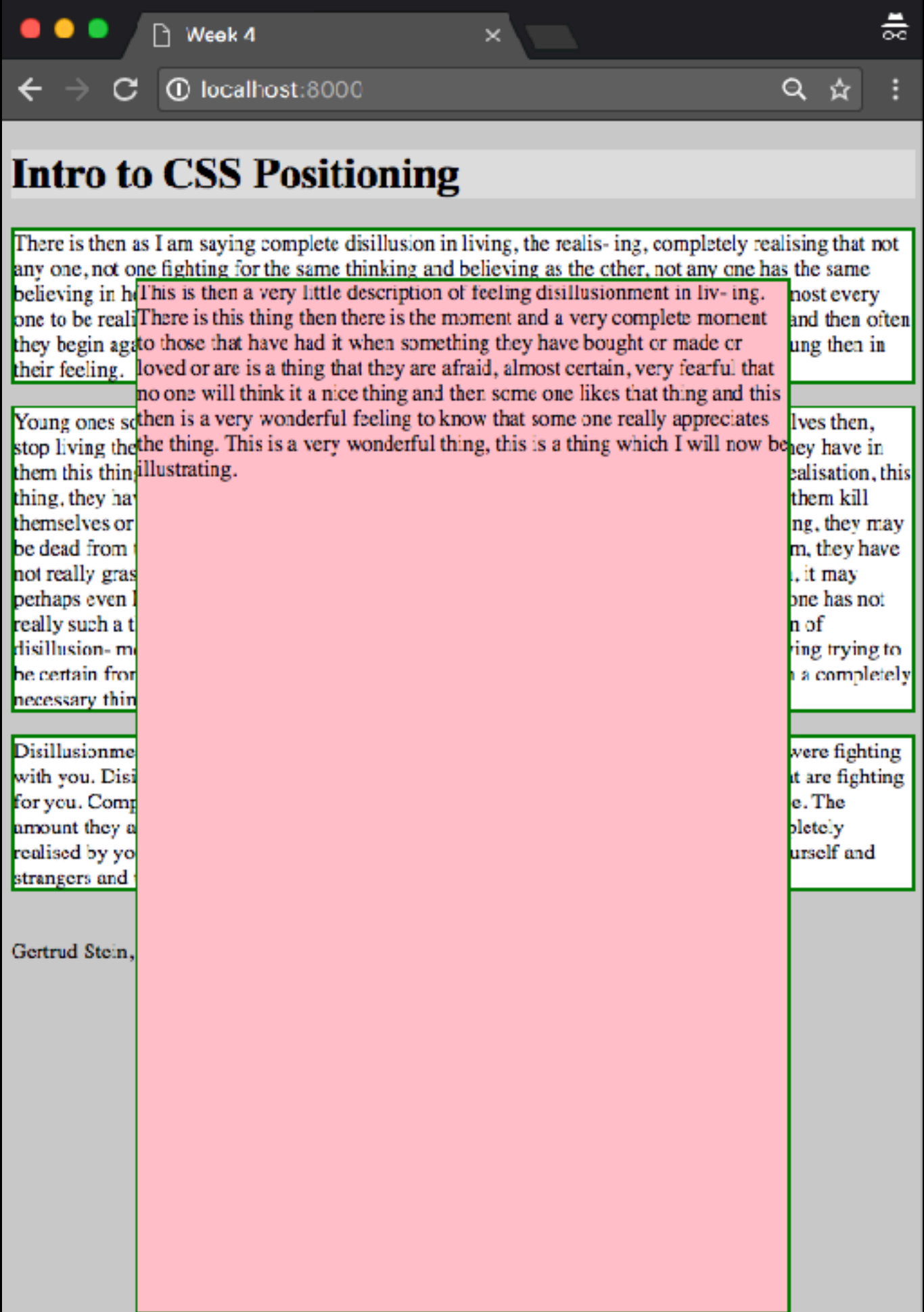


Notice that the position of the element has changed — this is because **top**, **bottom**, **left**, and **right** behave in a different way with absolute positioning.

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

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 Hendersland 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.

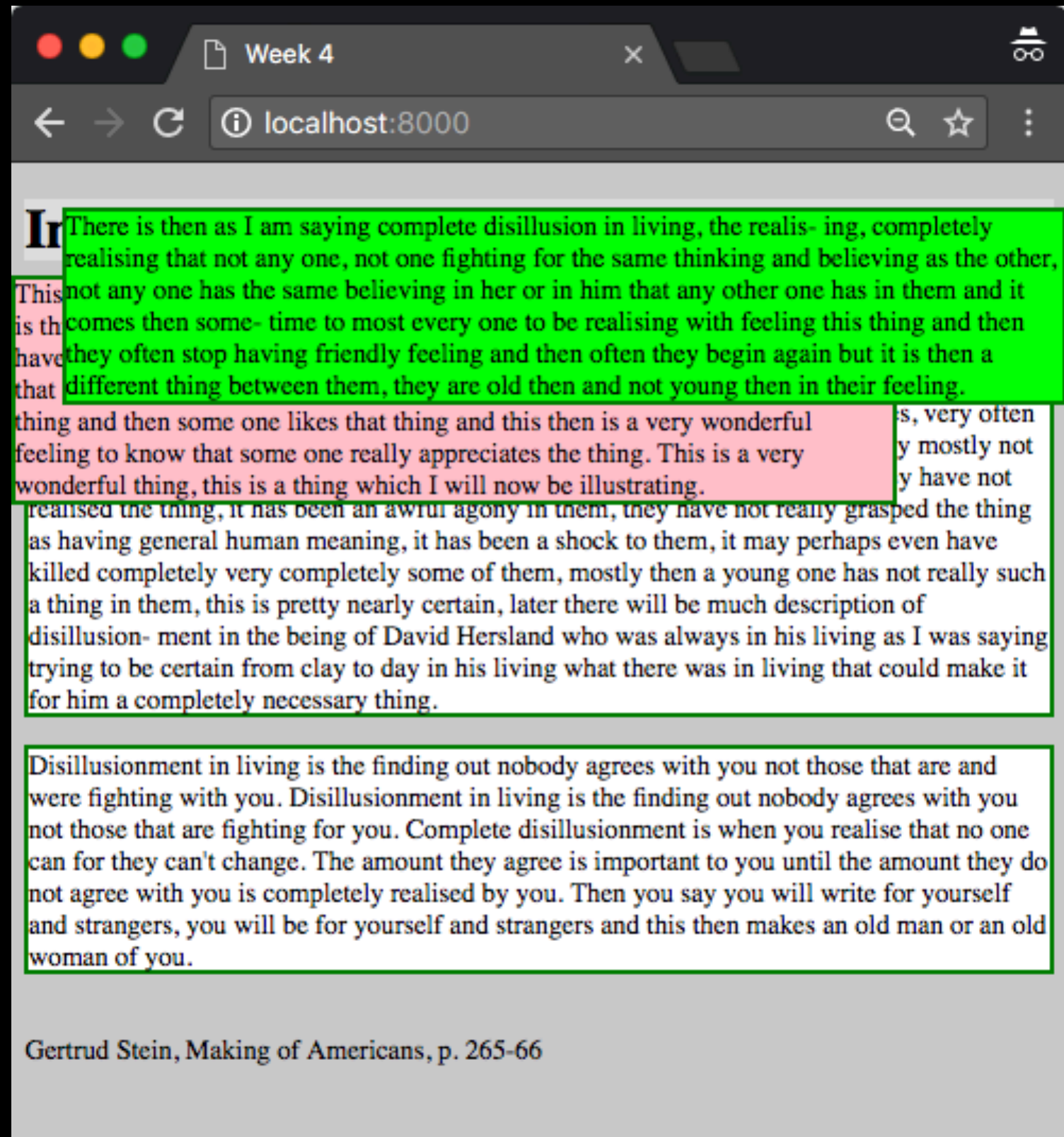
Gertrud Stein, Making of Americans, p. 265-66

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;  
}
```

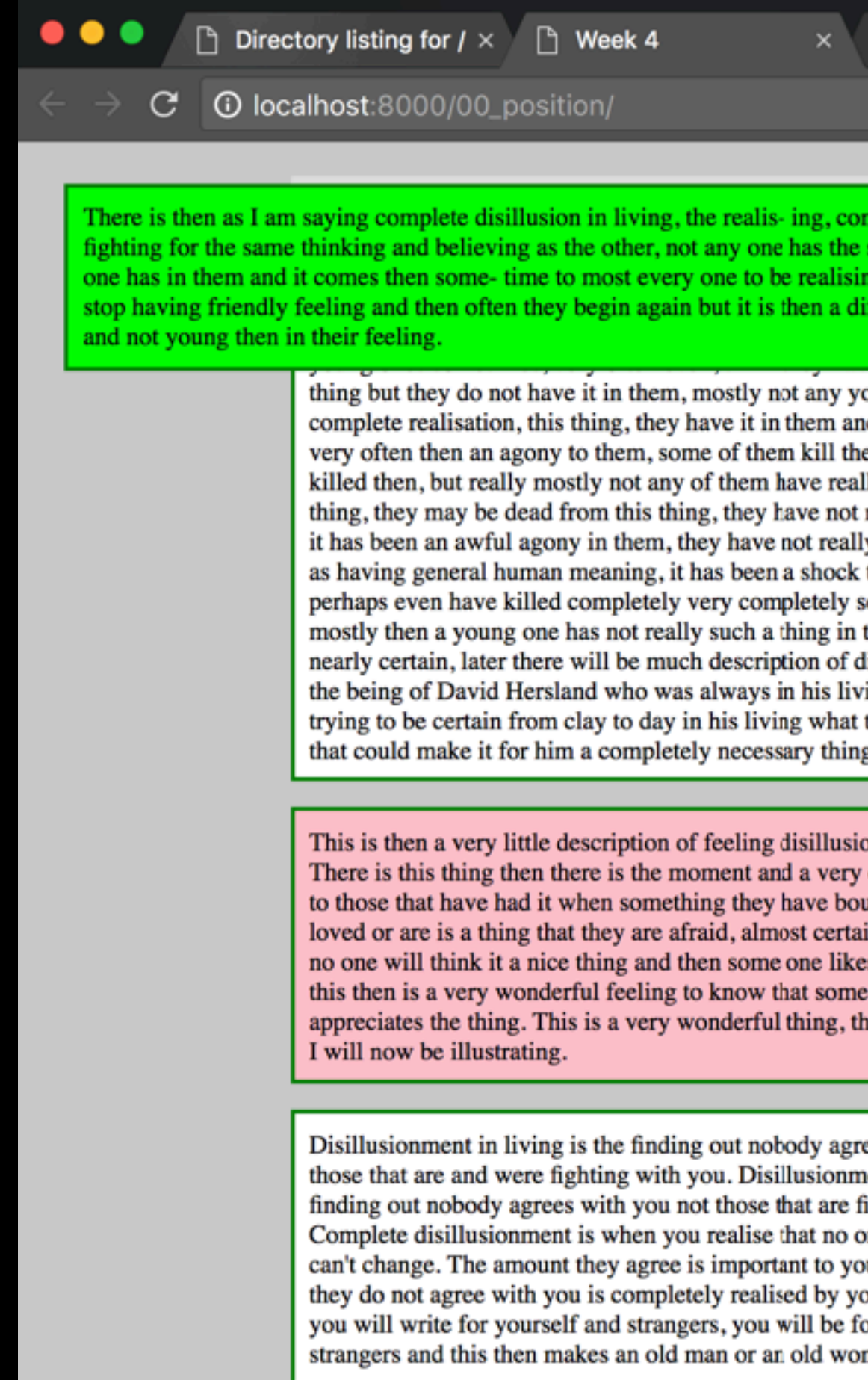


Overlapping Elements

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;
```

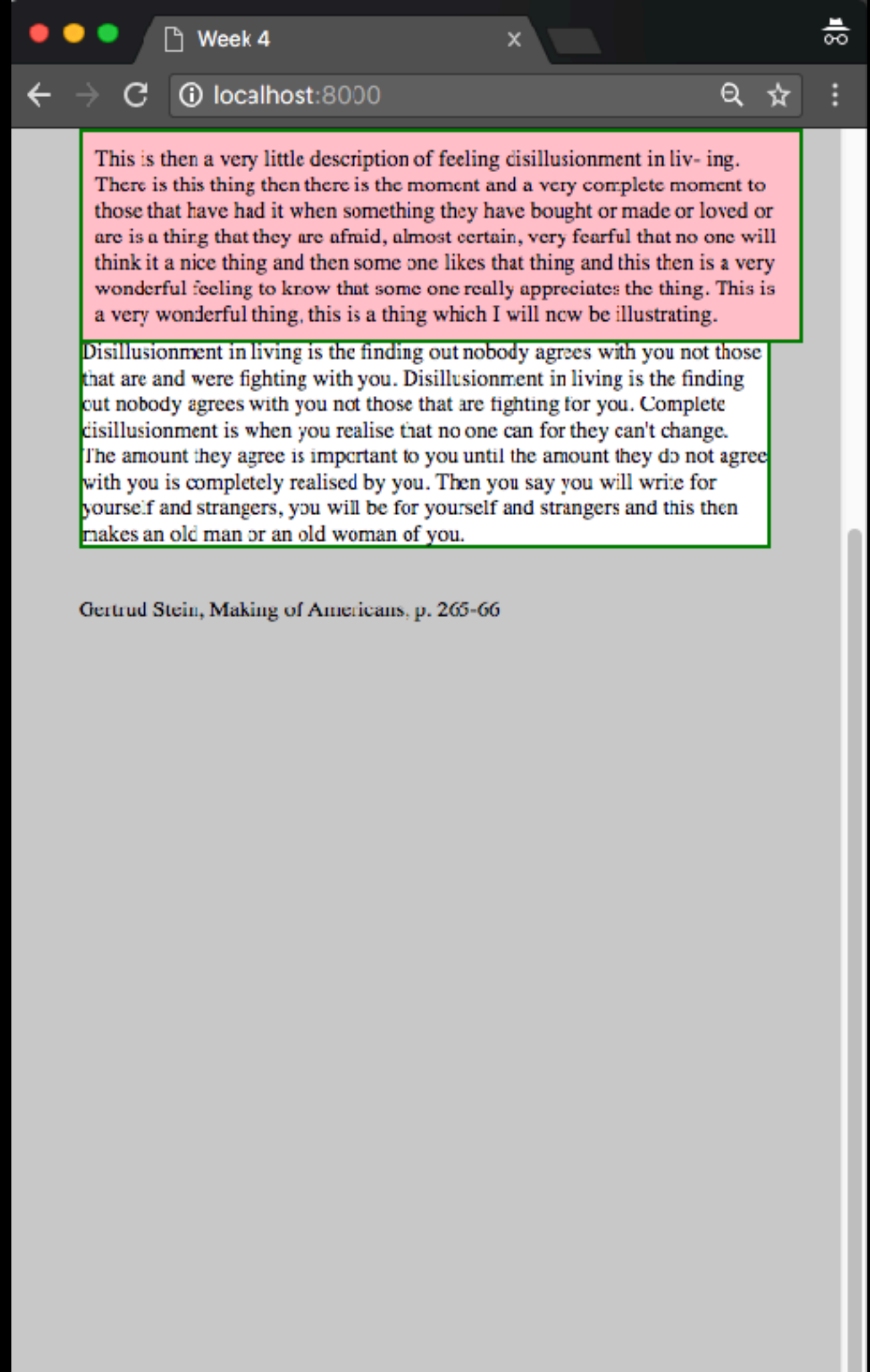
```
top: 0;
```

```
width: 50px;
```

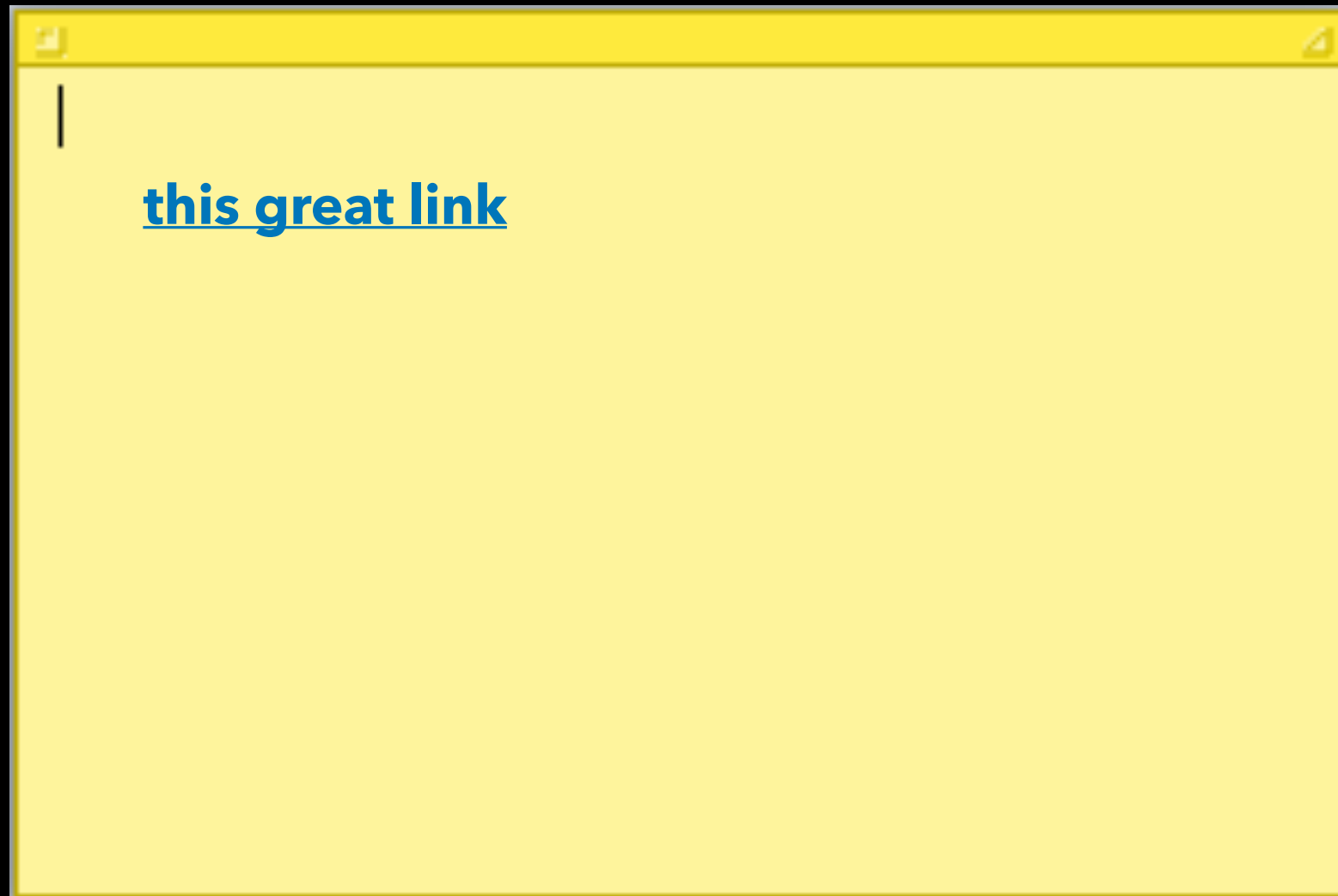
```
margin: 0 auto;
```

```
padding: 10px;
```

```
}
```



Position Sticky



more about web design

Default Web Fonts

Verdana

Arial

Arial Narrow

Arial Black

Helvetica

Century Gothic

Courier

Courier New

COPPERPLATE GOTHIC

Times

Times New Roman

Georgia

Geneva

Gill Sans

Tahoma

Trebuchet

Comic Sans

Impact

Palatino Linotype

Book Antiqua

Lucida Console

Lucida Sans Unicode

Serif

Sans-Serif

Font

Padding is the space btw the border + the content.

Some Properties:

`font-family`

`color`

`font-size`

`line-height`

`text-align`

Font

Padding is the space btw the border + the content.

text-decoration

underline, strike thru or none (eg to unset underline on hyperlinks)

text-transform

change font **case** (eg uppercase, lower, capitalize, none)

font-style

set to italic or normal

font-weight

set to bold or normal

letter-spacing

controls the space btw letters

Font stack

It's important to understand that the browser will only display 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 Font API

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;
```

Responsive Text

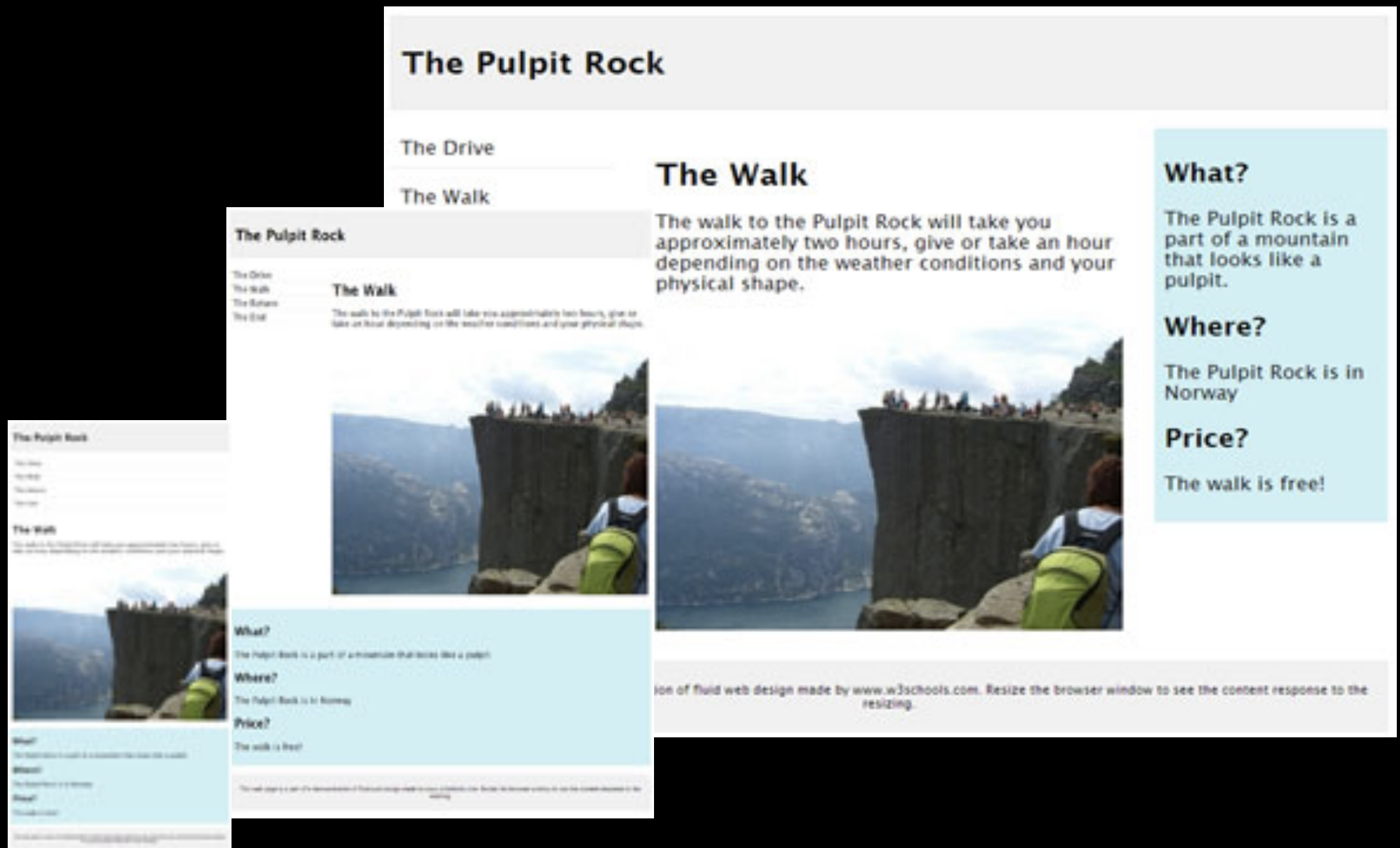
The text size can be set with a "vw" unit, which means the "viewport width".

That way the text size will follow the size of the browser window.

```
<h1 style="font-size:10vw">Hello World</h1>
```

responsive web design

Responsive Web Design



Responsive web design makes your web page look good on all devices.

Responsive web design uses only HTML and CSS.

Responsive web design is not a program or a JavaScript file.

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

Let's breakdown the `content` value of this responsive `<meta>` tag:

Values are comma separated, letting you specify a list of values for `content`

The `width` value is set to `device-width`. This will cause the browser to render the page at the same width of the device's screen size.

`initial-scale` set to `1` indicates the "zoom" value if your web page when it is first loaded. `1` means "no zoom."

There are other values you can specify for the `content` list -

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

Viewport

the rectangle where the webpage shows up, scrollable via a scrollbar

vh and vw

You can define height and width in terms of the viewport

- Use units **vh** and **vw** to set height and width to the percentage of the viewport's height and width, respectively
- $1\text{vh} = 1/100\text{th}$ of the viewport height
- $1\text{vw} = 1/100\text{th}$ of the viewport width

Example:

- height: 100vh; - width: 100vw;

Responsive Text

The text size can be set with a "vw" unit, which means the "viewport width".

That way the text size will follow the size of the browser window.

```
<h1 style="font-size:10vw">Hello World</h1>
```

Mobile First

Common device widths

- 320px (x-small mobile)
- 375px (small mobile)
- 768px (tablet)
- 1024px (laptop)
- 1440px (desktop)

Although it is good practice to create and test your responsive design with these sizes in mind, it is also important that you test beyond the suggested sizes as well.

Users are used to scrolling websites vertically, not horizontally!

Do NOT use large fixed—width elements (such as large images)

Do NOT let the content rely on a particular viewport width

Use CSS *media queries* to apply different styling for small and large screens

Media Queries

Uses the @media rule to include a block of CSS properties only if a certain condition is true

Media Queries

the @media rule tells the browser to include a block of CSS properties only if a certain condition is true.

So this:

```
@media only screen and (max-width: 500px) {  
  body {  
    background-color: light blue;  
  }  
}
```

Translates to:

```
if (the maximum width of the web page is 500 pixels) {  
  then do this stuff  
}
```

Media Queries

Breakpoint

```
/* For mobile phones: */
[class*="col-"] {
    width: 100%;
}

@media only screen and (min-width: 768px) {
    /* For desktop: */
    .col-1 {width: 8.33%;}
    .col-2 {width: 16.66%;}
    .col-3 {width: 25%;}
    .col-4 {width: 33.33%;}
    .col-5 {width: 41.66%;}
    .col-6 {width: 50%;}
    .col-7 {width: 58.33%;}
    .col-8 {width: 66.66%;}
    .col-9 {width: 75%;}
    .col-10 {width: 83.33%;}
    .col-11 {width: 91.66%;}
    .col-12 {width: 100%;}
}
```

add a **breakpoint** where certain parts of the design will behave differently on each side of the breakpoint

many examples: https://www.w3schools.com/Css/css_rwd_mediaqueries.asp

Mobile-first! (Images)

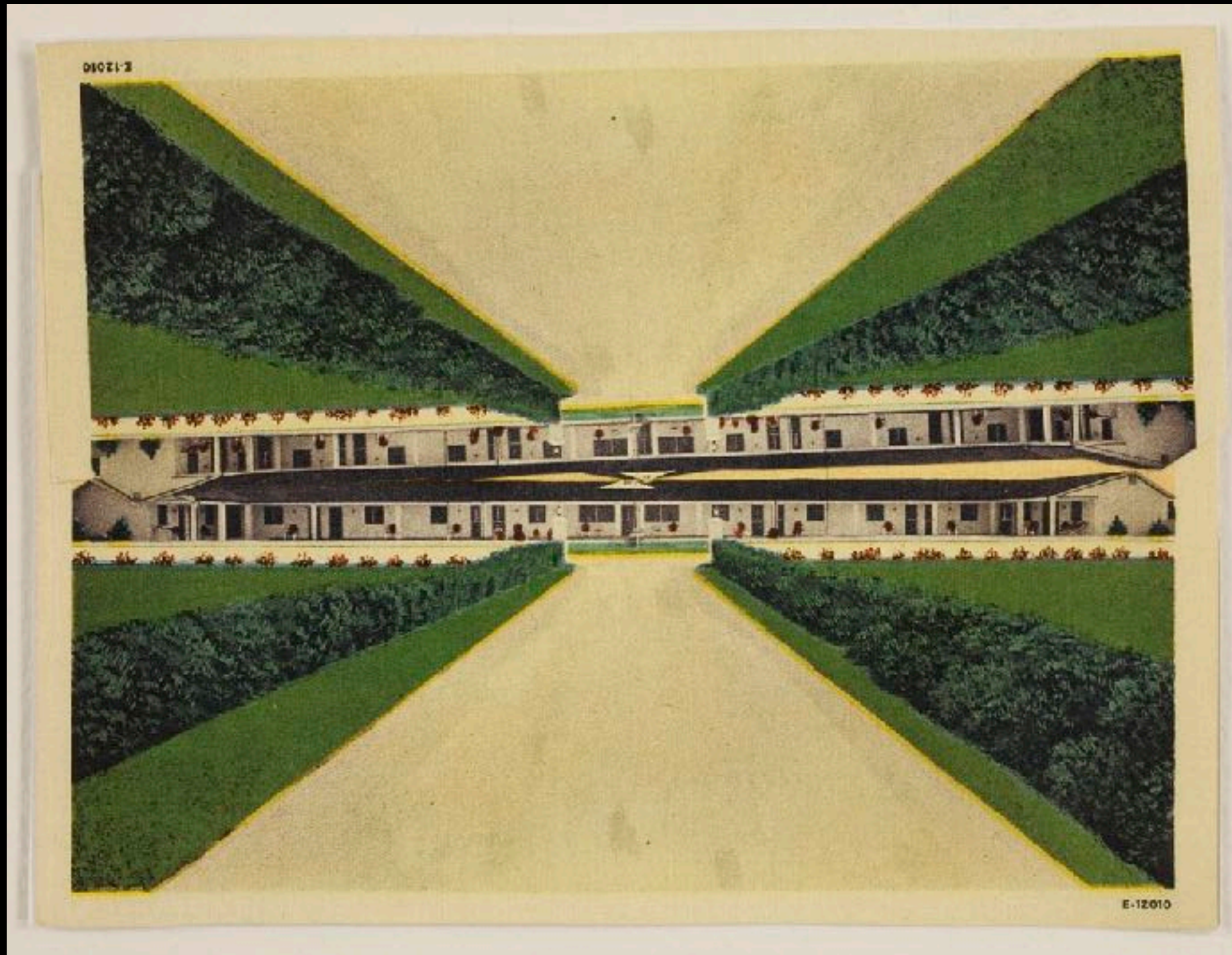


```
/* For width smaller than 400px: */  
body {  
    background-image: url('void_newspaper.jpg');  
}
```



```
/* For width 400px and larger: */  
@media only screen and (min-width: 400px) {  
    body {  
        background-image: url('void.jpg');  
    }  
}
```


If the max-width property is set to 100%, the image will scale down if it has to, but never scale up to be larger than its original size



```

```

Responsive Frameworks

Responsive frameworks are templates of responsive stylesheets

Examples: W3.CSS, Bootstrap (JavaScript)

Examples: https://www.w3schools.com/html/html_responsive.asp

rwd

Responsive Web Design

Most of these notes are verbatim txt from: [Learning Web Design - Jennifer Niederst Robbins](#)

Layout

Rearranging content into different layouts may be the first thing you think of when you picture responsive design, and with good reason. The layout helps form our first impression of a site's content and usability.

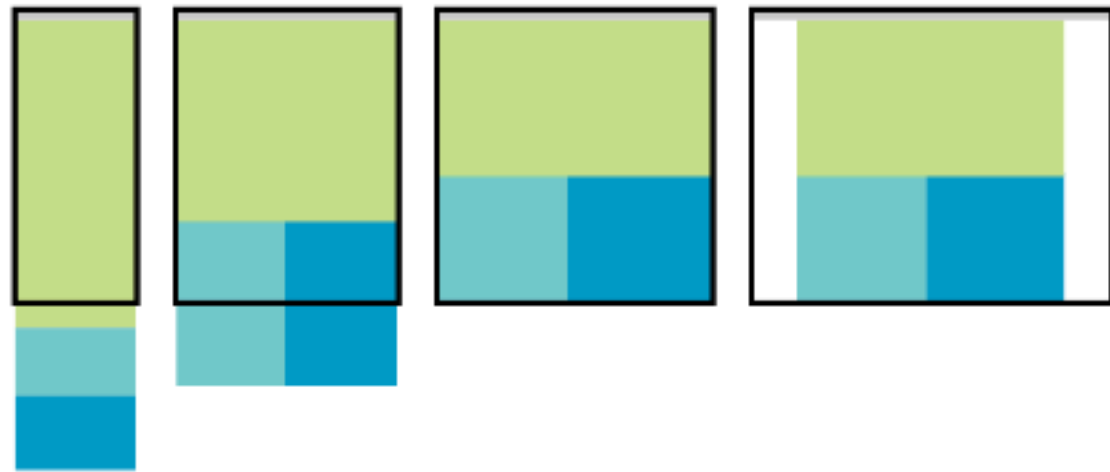
Responsive design is based on fluid layouts that expand and contract to fill the available space in the viewport. One **fluid layout** is usually not enough, however, to serve all screen sizes. More often, two or three layouts are produced to meet requirements across devices, with small adjustments between layout shifts.

Designers typically start with a one-column layout that fits well on small handheld devices and rearrange elements into columns as more space is available. They may also have the design for the widest screens worked out early on so there is an end-point in mind. The design process may involve a certain amount of switching between views and making decisions about what happens along the way.

Responsive layout patterns

The manner in which a site transitions from a small-screen layout to a wide-screen layout must make sense for that particular site, but there are a few patterns (common and repeated approaches) that have emerged over the years. We can thank Luke Wroblewski (known for his “Mobile First” approach to web design, which has become the standard) for doing a survey of how responsive sites handle layout. Following are the top patterns Luke named in his [article](#):

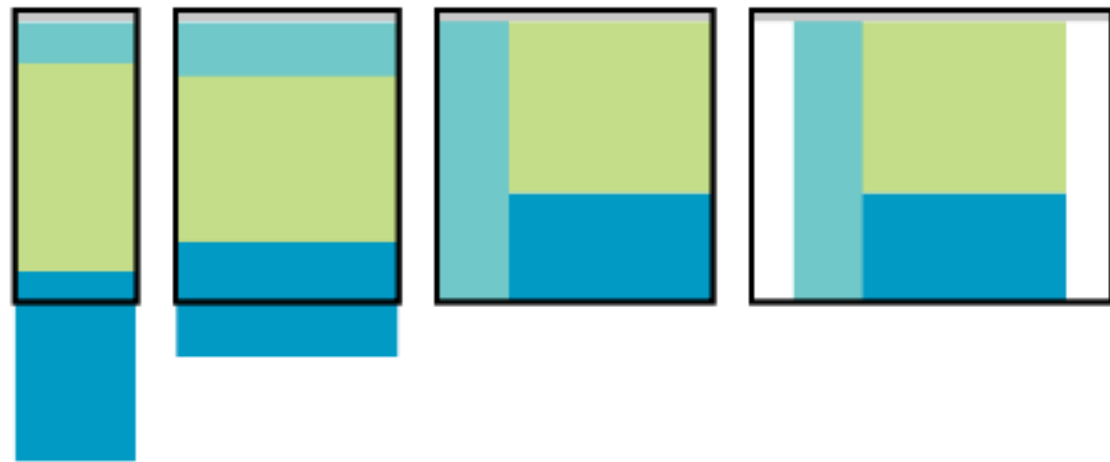
Mostly fluid



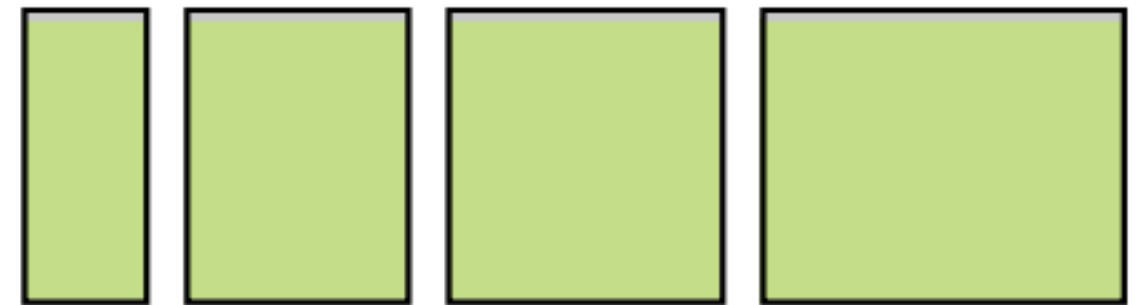
Column drop



Layout shifter



Tiny tweaks



Off canvas

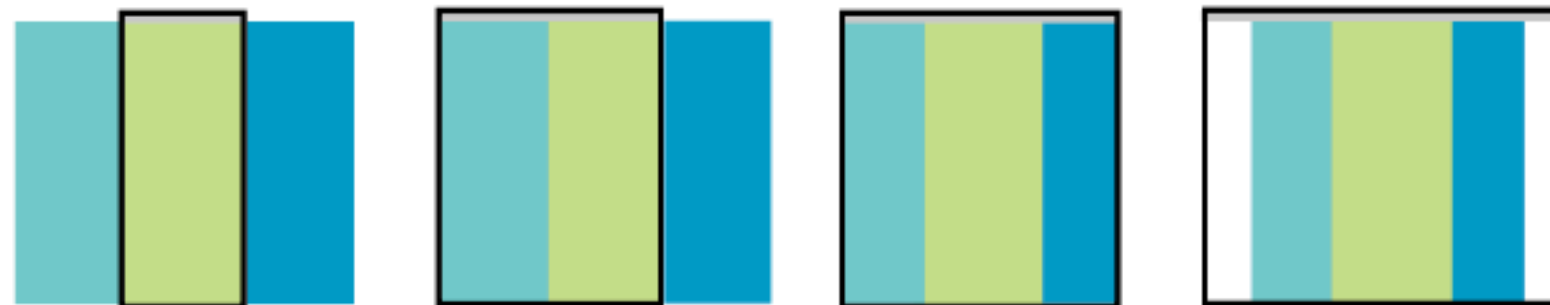


FIGURE 17-9. Examples of the responsive layout patterns identified by Luke Wroblewski.

Mostly fluid

This pattern uses a single-column layout for small screens, and another fluid layout that covers medium and large screens, with a maximum width set to prevent it from becoming too wide. It generally requires less work than other solutions.

Column drop

This solution shifts between one-, two-, and three-column layouts based on available space. When there isn't room for extra columns, the sidebar columns drop below the other columns until everything is stacked vertically in the one-column view.

Layout shifter

If you want to get really fancy, you can completely reinvent the layout for a variety of screen sizes. Although expressive and potentially cool, it is not necessary. In general, you can solve the problem of fitting your content to multiple environments without going overboard.

Tiny tweaks

Some sites use a single-column layout and make tweaks to type, spacing, and images to make it work across a range of device sizes.

Off canvas

As an alternative to stacking content vertically on small screens, you may choose to use an “off-canvas” solution. In this pattern, a page component is located just out of sight on the left or right of the screen and flies into view when requested. A bit of the main content screen remains visible on the edge to orient users as to the relationship of moving parts. This was made popular by Facebook, wherein Favorites and Settings were placed on a panel that slid in from the left when users clicked a menu icon

Navigation

Navigation feels a little like the Holy Grail of Responsive Web Design. It is critical to get it right. Because navigation at desktop widths has pretty much been conquered, the real challenges come in re-creating our navigation options on small screens. A number of successful patterns have emerged for small screens, which I will briefly summarize here

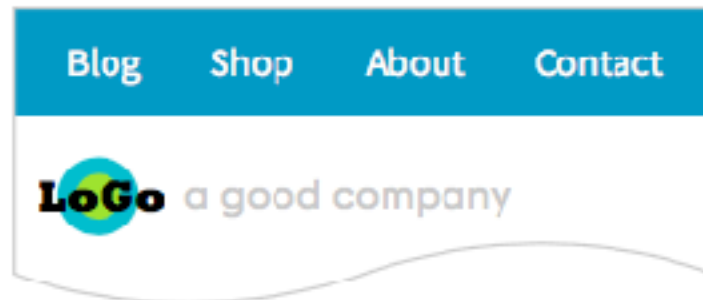
Top navigation

If your site has just a few navigation links, they may fit just fine in one or two rows at the top of the screen.

Priority +

In this pattern, the most important navigation links appear in a line across the top of the screen alongside a More link that exposes additional options. The pros are that the primary links are in plain view, and the number of links shown can increase as the device width increases. The cons include the difficulty of determining which links are worthy of the prime small-screen real estate.

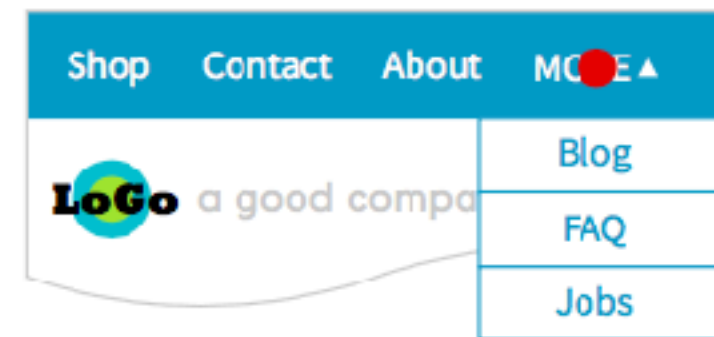
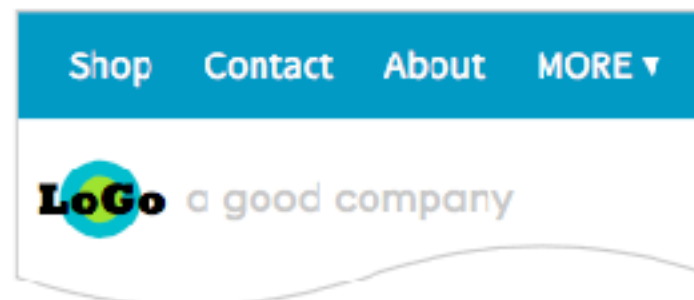
Top navigation



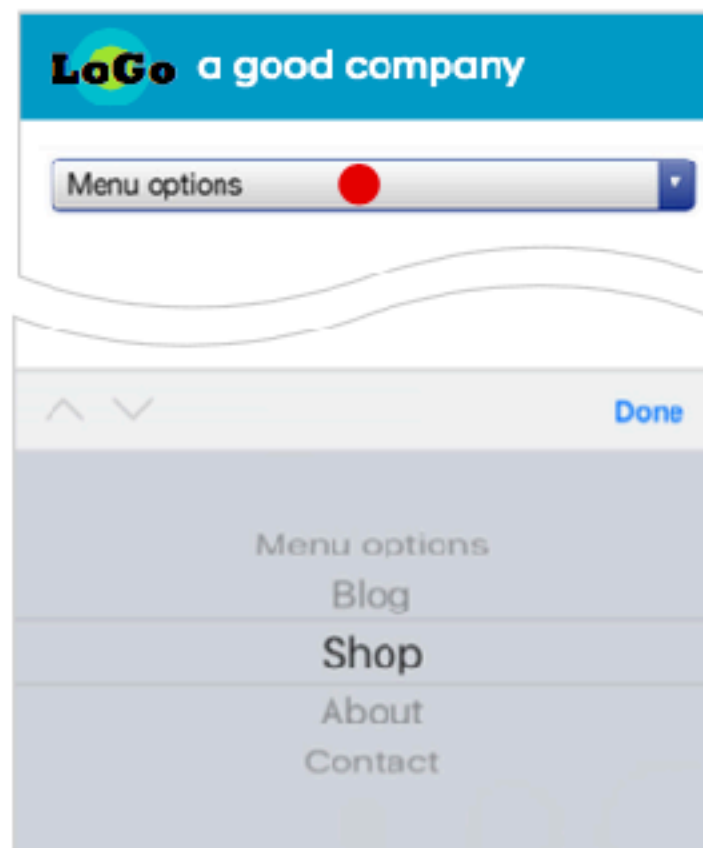
KEY

● = tap

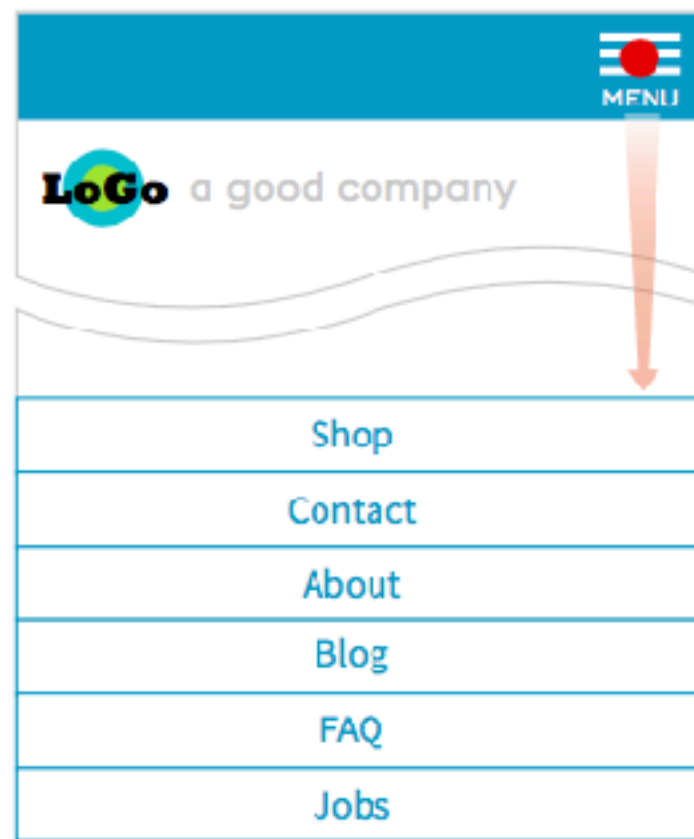
Priority +



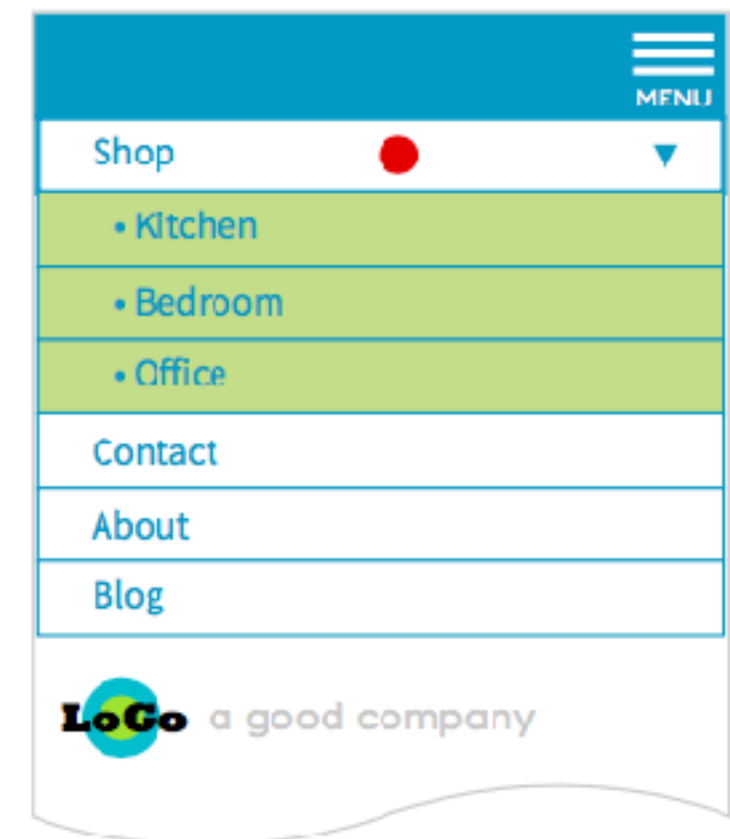
Select menu



Link to footer menu



Accordion sub-navigation



Select menu

For a medium list of links, some sites use a select input form element. Tapping the menu opens the list of options using the select menu UI of the operating system, such as a scrolling list of links at the bottom of the screen or on an overlay. The advantage is that it is compact, but on the downside, forms aren't typically used for navigation, and the menu may be overlooked.

Link to footer menu

One straightforward approach places a Menu link at the top of the page that links to the full navigation located at the bottom of the page. The risk with this pattern is that it may be disorienting to users who suddenly find themselves at the bottom of the scroll.

Accordion sub-navigation

When there are a lot of navigation choices with sub-navigation menus, the small-screen solution becomes more challenging, particularly when you can't hover to get more options as you can with a mouse. Accordions that expand when you tap a small arrow icon are commonly used to reveal and hide sub-navigation. They may even be nested several levels deep. To avoid nesting navigation in accordion submenus, some sites simply link to separate landing pages that contain a list of the sub-navigation for that section.

Overlay toggle (covers top of screen)



Push toggle (pushes content down)



Off-canvas/fly-in



FIGURE 17-11. Responsive navigation patterns.

Display Property

The display property specifies if/how an element is displayed. Every HTML element has a default display value depending on what type of element it is. The default display value for most elements is block or inline. A block-level element always starts on a new line and takes up the full width available (stretches out to the left and right as far as it can). An inline element does not start on a new line and only takes up as much width as necessary.

`display: none;`

commonly used with JavaScript to hide/show elements without deleting and recreating them.

Overriding Default Display

Changing an inline element to a block element, or vice versa, can be useful for making the page look a specific way, and still follow the web standards.

```
li {  
    display: inline;  
}
```

```
span {  
    display: block;  
}
```

Note: Setting the display property of an element only changes how the element is displayed, NOT what kind of element it is. So, an inline element with display: block; is not allowed to have other block elements inside it.

Overflow Property

The CSS overflow property controls what happens to content that is too big to fit into an area. The overflow property specifies whether to clip content or to add scrollbars when the content of an element is too big to fit in a specified area. The overflow property only works for block elements with a specified height. The overflow property has the following values:

- visible** - Default. The overflow is not clipped. It renders outside the element's box
- hidden** - The overflow is clipped, and the rest of the content will be invisible
- scroll** - The overflow is clipped, but a scrollbar is added to see the rest of the content
- auto** - If overflow is clipped, a scrollbar should be added to see the rest of the content

Properties for left and right

overflow-x

specifies what to do with the left/right edges of the content.

overflow-y

specifies what to do with the top/bottom edges of the content.

```
div.theExample1 {  
  background-color: lightblue;  
  height: 40px;  
  width: 200px;  
  overflow-y: scroll;  
}
```

```
div.theExample2 {  
  background-color: lightblue;  
  height: 40px;  
  width: 200px;  
  overflow-y: hidden;  
}
```

code from W3