

Build a Sophisticated Webpage: Step-by-Step

In this class we've covered a lot about the technical skills required to build a website. We've discussed asset preparation, basic HTML skills such as writing basic HTML tags, HTML attributes, and container tags. We have discussed the basics of CSS including syntax, CSS typography and CSS Box Model. Recently we've applied an advanced technique known as CSS grids which allows us to set up rows and columns for our designs.

In this lab we will combine EVERYTHING we've learned to go step-by-step of how to turn a design into a working website using HTML and CSS.

OVERVIEW

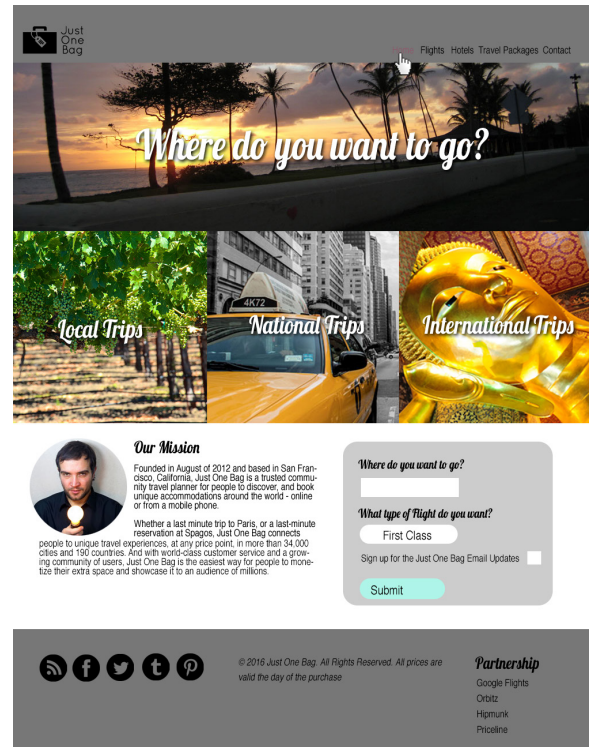
In this lab we will go through the steps of creating a complex webpage layout for a fictitious luxury travel company called "Just One Bag". The process will go through the following steps:

- Design Overview
- HTML Core tag layout
- HTML Typography tags
- Images and Links
- HTML Container Tags / Ids
- CSS typography
- CSS Backgrounds
- Box Model
- CSS Grids

GETTING STARTED

For this project we will be building a sophisticated webpage for a fictitious company called 'Just One Bag'. We have previously met with the client and discussed the needs of the website with them. They have hired an art director who put together a wireframe and mockup for the webpage. You can find these items in the "resource" folder of the main Just-One-Bag folder. You can also view the finished webpage with the link below:

<http://davidhurwich.com/teaching/des117/just-one-bag/>



Take a moment to look over both the wireframe and the mockup for the webpage. You'll see that design includes a logo, navigation bar, splash area, a mission statement by the CEO, a form for ordering flights tickets and a footer section containing legal information, a social media bar and list of links to partnership companies.

HYPER TEXT MARKUP LANGUAGE

Hyper Text Markup Language

FOUNDATIONS

Let's begin with the foundations. Create a new document in Brackets and save it as "justonebag-yourname.html". When you save the file save it into the Just-One-Bag folder into the top level of hierarchy, meaning do not save it into a subfolder. See the image on the right for reference:

As per usual add in all the standard HTML core tags including `<!doctype>`, `<html>`, `<head>`, `<title>` and `<body>`.

NAVIGATION BAR

Let's add the navigation bar with a group of hyperlinks. You can make navigation links dummy links for now by doing the following:

```
<a href="#"> Home </a>
```

Add the following hyperlinks for the nav bar

- Home
- Flights
- Hotels
- Vacation Packages
- Contact

THE REST OF THE TEXT

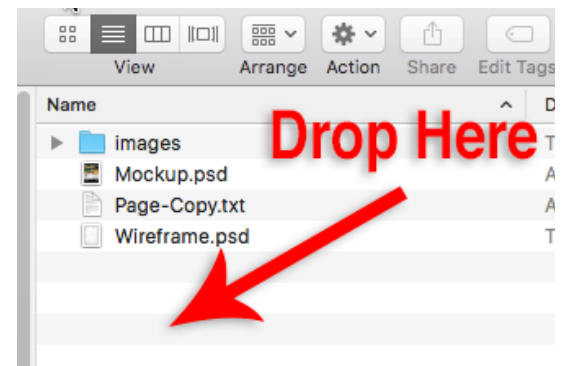
Let's add the text for the header splash area. Put "Where do you want to go" in a `<h2>`.

Put the travel destinations, Local Trips, national Trips etc as seen in the mockup inside `<h3>`.

Put the phrase "Our Mission" inside an `<h3>`.

Open Just-One-Bag-Page-Copy.txt and copy and paste the text for the mission section into your page. Convert the paragraphs into `<p>` tags.

We'll skip the form for now and come back to it later.



```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>Just One Bag</title>
</head>
<body>
</body>
```

```
<body>
<nav>
  <a href="#"> Home </a>
  <a href="#"> Flights </a>
  <a href="#"> Hotels </a>
  <a href="#"> Travel Packages </a>
  <a href="#"> Contact </a>
</nav>
<h2> Where do you want to go? </h2>
<h3> Local Trips </h3>
```

Copy and paste the legal copy into your page. Convert it into <p> tags.

Add an h3 called "Our Partnerships".

Save and preview. Your page should look roughly like the one on the right.

ADD IMAGES AND HYPERLINKS

Alright, we have are text setup, let's add in our images.

BEFORE the navigations links add in the logo for Just One Bag. It can be found in the "images" folder under the name logo.png.

Remember that your HTML file is saved in the top heirarchy, so to reference images in the "images" folder we will need to go inside the images folder. The syntax would look like this:

```

```

Add the logo BEFORE the navigation links.

Let's add a photo of the CEO into the mission statement area. Add ceo.png BEFORE the mission statement text.

Let's look at the social media bar in the mockup. We'll create this by creating an unordered list of images. Create a BEFORE the legal text and add in the social images icon images.

HYPERLINKS

Alright, let's put in some hyperlinks that actually work. In the footer under 'Partnerships' we'll have an unordered list of hyperlinks to other websites that partnered with Just One Bag.

Create a of hyperlinks. Use the following as reference:

Google Flights: <http://www.google.com/flights>
Orbitz: <http://www.orbitz.com>
Hipmunk: <http://www.hipmunk.com>
Priceline: <http://www.priceline.com>

Save and preview, your page should be similar to the one on the

[Home Flights Hotels Travel Packages Contact](#)

Where do you want to go?

Local Trips

National Trips

International Trips

Our Mission

2016 Copyright Just One Bag. All Rights Reser



[Home Flights Hotels Travel Packages](#)

Where do you want to g



Our Partnerships

- [Google Flights](#)
- [Orbitz](#)
- [Hipmunk](#)
- [Priceline](#)

right.

FORM

Alright, text, images and links complete, let's add in a form so customers can purchase a ticket through the webpage. Normally this would require connecting our webpage to a database to get and store information like credit cards and addresses. This is normally very complicated, for this lab we'll just create the form, but it won't actually connect to a database. This means our data we type won't be stored unfortunately.

Begin by creating a form with the `<form>` tag.

Add in questions, put each in an `<h3>`

- Where do you want to go?
- What type of flight would you like?

Set the "Where do you want to go?" to an `<input type="text">`

Set the "What type of flight would you like?" to a `<select>`. Set the `<option>` to

First Class
Business Class
Standard
Standby

Add a paragraph asking "Sign up for Just One Bag email updates".
Add a checkbox

Finally, let's finish it off with a submit button. You can add this using an `<input type="submit">`

Save and preview. Your page should roughly look like the image on right.

HTML5 CONTAINERS

Alright, lastly let's begin to organize areas of our page together in groups. Having areas organized into containers such as `<header>`, `<main>` and `<section>` will make it much easier to stylize and layout the content when we begin to use CSS.

Take a look at the Wireframe.pdf located in the Resources folder.

```
<form>

</form>
```

```
<h3> Where do you want to go? </h3>
<input type="text">
<h3> What type of flight do you want? </h3>
<select>
  <option> First Class </option>
  <option> Business Class </option>
  <option> Standard </option>
  <option> Standby </option>
</select>
<p>Sign up for Just One Bag email updates </p>
<input type="checkbox">
<p>
  <input type="submit">|
</p>
```

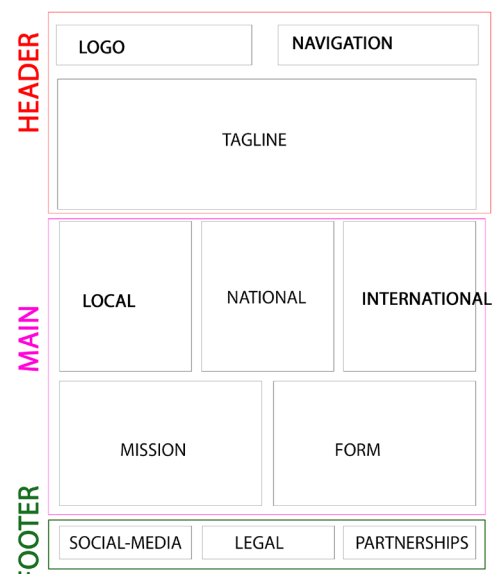
Where do you want to go?

What type of flight do you want

First Class

Sign up for Just One Bag email updates ☐

Submit



The wireframe gives us a view of how the webpage breaks down into major sections.

Below is the list of major sections

- logo
- navigation
- tagline
- local
- national
- international
- mission
- form
- social-media
- legal
- partnerships

<SECTION>

Begin to organize the existing text and image tags into the appropriate sections. For instance, your logo would look something like this

```
<section>
  
</section>
```

Your navigation would look something like this

```
<section>
  <a href="#"> Home </a>
  <a href="#"> Flights </a>
  ....
</section>
```

Continue to put each major area into a <section> tag.

<NAV>

In addition to adding the navigation links inside a section, let's structure them a bit more with a <nav> tag. This will give us a bit more control later on.

MAJOR ORGANIZING TAGS

So in addition to organizing HTML into groups or <section>s, we

can group the groups (crazy right?). In a document we can have major areas that hold multiple blocks or sections of content. We will use <header>, <main> and <footer>.

Place the logo, navigation and tagline into a <header> tag.

Place the local, national, international, mission statement and form into the <main> tag.

Place the social-media, legal information and partnerships into a <footer> tag.

Ids

Organizing our HTML is great, it will allow us to style it using CSS. Let's take it another step by labeling each major block. The reason for labeling each block is to pinpoint an exact area. For instance, perhaps we want to style just the navigation area, or just the Mission area. Ids make this much easier. By labeling a block of content it will make it easier to style with CSS later on. Each area gets it's own unique id that will get applied in the HTML.

For instance we would label the logo section as follows

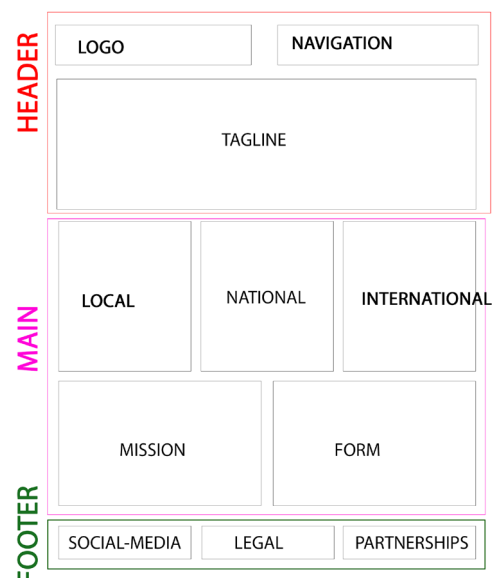
```
<section id="logo">
  
</section>
<section id="nav">
  <nav>
    <a href="#"> Home </a>
    <a href="#"> Flights </a>
    <a href="#"> Hotels </a>
    ....
  </nav>
</section>
```

Continue to label the remaining sections, use the wireframe as reference.

```
#logo
#nav
#tagline
#local
#national
```

```
<header>
  <section>
    
  <section>
    <nav>
      <a href
      <a href
      <a href
```

```
<section id="logo">
  
  <nav>
    <a href="#">Home </a>
    <a href="#">Flights <
    <a href="#">Hotels</a>
    <a href="#">Travel Pa
    <a href="#">S
```



#international
#mission
#form
#social-media
#legal
#partnerships

Save and preview. You can view a template page the link below to see give you an idea of how the webpage should look.

You can view the HTML below:

<http://davidhurwich.com/teaching/des117/just-one-bag/just-one-bag-html.html>

CASCADING STYLESHEETS

CASCADING STYLESHEETS

So, we've structured everything, yay! Now that the tedious part is done let's do the fun part and actually get to begin styling our webpage.

Let's start by creating a new document in Brackets and saving it as style.css. Save it into the main heirarchy level as the just-onebag-username.html file.

In your HTML file add a `<link rel="stylesheet" href="file">` to connect the CSS document to your HTML document.

ORGANIZATION

Complicated pages can have ALOT of CSS, so let's try to organize it into sections as we write it. They will make it MUCH easier to go back and make changes later on.

Let's also write our CSS in a specific order, starting with simpler things such as type and colors and getting more complicated as we continue. For this lab let's go in this order:

- CSS Typography
- CSS backgrounds
- Box Models
- CSS Grids
- Odds and Ends

CSS TYPOGRAPHY

Let's organize this by adding the following comment

```
/* === CSS Typography == */
```

Let's begin by defining all of typography for this document. To do that let's look at the mockup in Adobe Photoshop.

Open mockup.psd into Adobe Photoshop. If you use the type tool you'll see that the design uses two main typefaces, Helvetica and Lobster. We can also use Photoshop and the Character palette to figure out how large text is, whether it is bold or not, and leading information as well.

```
head>  
<meta charset="UTF-8">  
<title>Just One Bag</title>  
<link rel="stylesheet" href="style.css">  
/head>
```



Let's figure out font-size and font-leading information for the

- navigation links
- header text
- local, national and international text
- mission statement body copy
- Form text
- and legal copy

It's a good idea to write this information into a text-document or on a piece of paper to refer back to later.

SETTING BASE TYPEFACES

Rather than writing rules for EVERY heading and paragraph HTML tag, let's use CSS inheritance to our advantage. Remember that CSS inheritance is the idea that CSS properties are passed down from parent tags to child tags.

Helvetica seems to be the main typeface so let's make that font the default for the entire page. Set the following

```
body {  
font-family: 'Helvetica', 'Arial', sans-serif;  
}
```

This will set the <body> font to Helvetica and pass it down to all the child elements inside of the <body>. Save and preview.

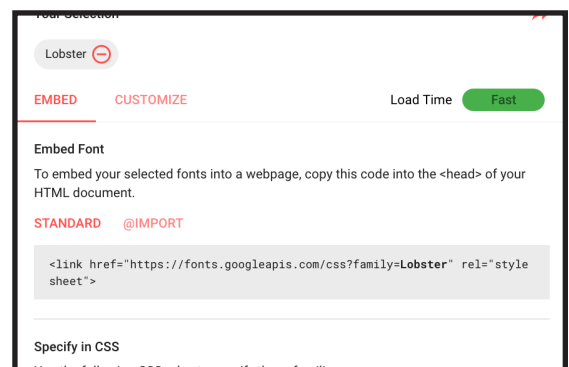
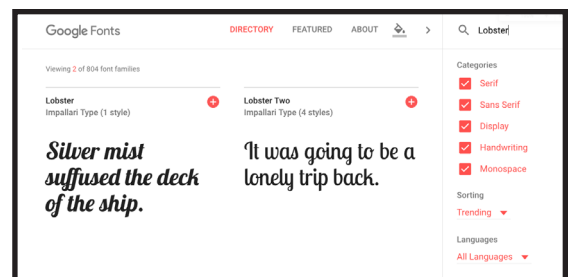
Let's set the other major typeface which is Lobster. Lobster is not a standard font, but instead a Google Webfont, so we'll need to grab the code for it from google.com/fonts. Visit the site and type 'Lobster' into the search.

From the results click the "+". From the menu on the bottom of the webpage click the 'embed' option and then choose the '@import' option.

Copy the code and paste it into the top of your CSS document.

Now let's set the h1, h2, and h3 to Lobster. If we do the following

```
h2 {  
font-family: 'Lobster';  
}
```



That would only set heading 2 to Lobster, to set all three let's use multiple selectors. Apply the following:

```
h1, h2, h3 {  
  font-family: 'Lobster';  
}
```

SET FONT SIZES

Let's say we want to set the size of the typeface inside the #header area. If you refer to the mockup the text should be 60pt size. So we could write

```
h2 {  
  font-size: 60pt;  
}
```

but there are other <h2> tags on the webpage, the result would turn them ALL to 60pt size. So let's use CSS compounds to be specific. We'll use the id to pinpoint the exact text we want to style.

```
#tagline h2 {  
  font-size: 60pt;  
}
```

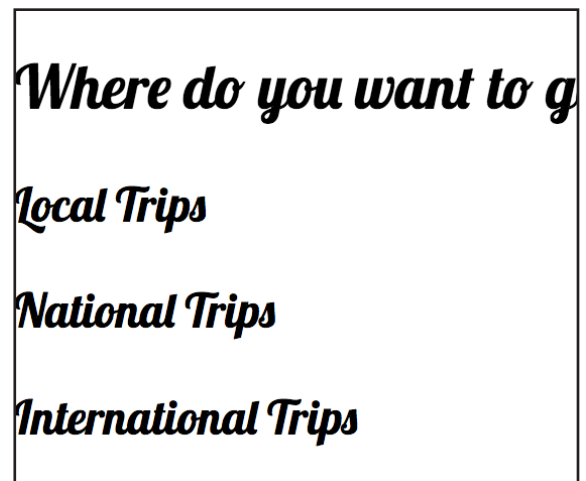
MUCH better, now we'll only affect the text within the #tagline area.

Using the information you got from the mockup use CSS compounds for the remain font-sizes on the page.

Don't forget about setting leading using the CSS property line-height for areas where text is multiple lines. You also will want to use font-weight to set text thinner or bolder. You may want to apply text-transform to make sure that the text is capitalized as well.

Also make sure to set a specific CSS rule for <a> links.

Refer to the textbook or w3schools.com if you have questions about syntax.



CSS BACKGROUNDS

```
/* === CSS Backgrounds == */
```

Ok, so we've gotten type taken care of, let's now tackle CSS backgrounds. Backgrounds are things such as background colors, background images and background gradients for our webpage.

Let's take a look at the mockup again. We can see that the top or <header> area of the page is gray, as is the bottom or <footer> area.

Open up the mockup and use the eyedropper tool from the toolbox. If you sample the mockup you'll see that the color is #777777.

Thus, we can set the background color for the <header> and <footer> as such:

```
header, footer {  
background-color: #777777;  
}
```

The other background images in the design are the background images for the #header, #local, #national and #international. The images for that are located in the "images" folder.

For this let's use the CSS property background-image to set the background image. Let's start with #header, set the following:

```
#tagline {  
background-image:url('tagline.png');  
}
```

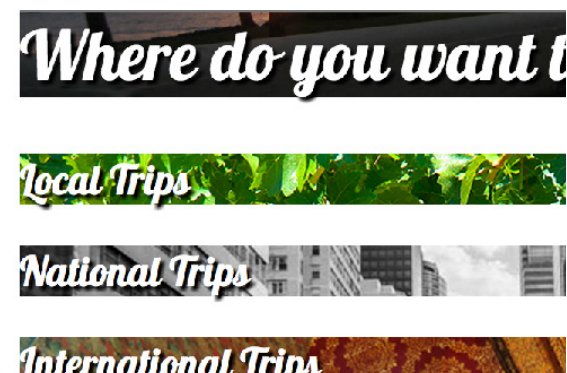
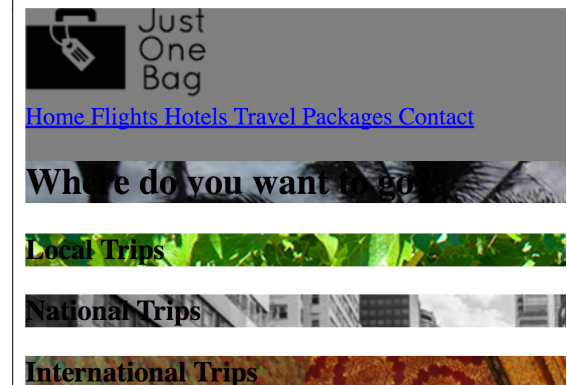
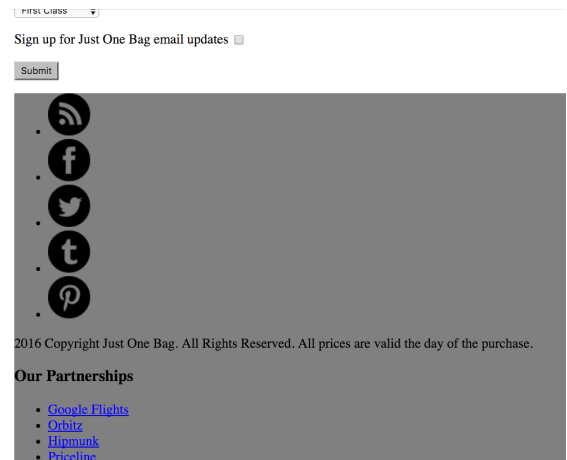
Save and preview.

Apply the other three image backgrounds as well for #local, #national and #international.

To help make the headings more visible set their font color to white (#ffffff) and add the CSS property text-shadow.

Box-Model

So, if we take a look at the our page it feels a bit 'off'. For starters



the header and destination areas like #local, #national, #international are very short. If we look at the mockup they should be much taller. Other areas such as the form feel very tight and contracted. Let's fix this by adding in box model properties such as width, height and padding. Add the following comment to your CSS.

```
/* === CSS Box-Model === */
```

Let's begin by setting a height for the #tagline Use the CSS height property for this:

```
#header {  
height: 400px;  
}
```

Let's set the destination areas to a height of 300px. You can either do this by using multiple CSS selectors, or by creating a class and applying it to multiple <section> tags.

Create a class. Create a class called .destinations

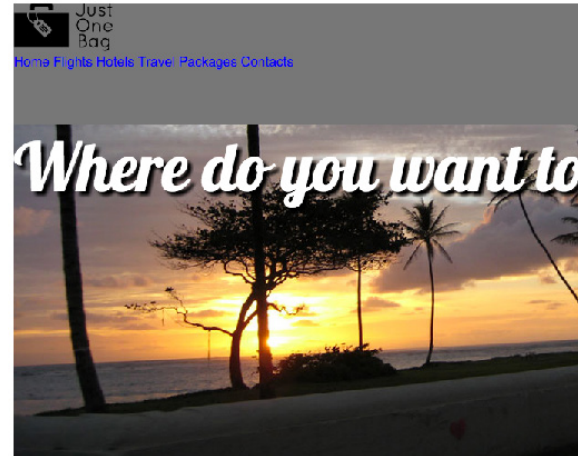
```
.destinations {  
height: 300px;  
}
```

Save and apply the class to the #local, #national and #international <section> tags. You should see each area grow taller.

Now that the containers are taller you'll notice that the background images repeat. You can fix that by adding the CSS property background-size and setting it to cover.

For the tagline area you'll notice we are using the top of the photo when the bottom of the photo is more interesting. We can change the starting place of the image by using the CSS property background-position and then using locations like left, right, center for the horizontal placement and top, center and bottom for vertical placement.

Let's apply a bit of padding to a few elements. We can apply padding to the <form> so that the text and input areas are not right up against the edge of the form container.



```
form {  
padding: 25px;  
}
```

See, doesn't that look a bit nicer?

STYLING THE FORM

The form is a bit tricky so we'll do this part together. Remember that the form visual style is based on the `<form>` tag. Begin by setting the background color for the form itself.

To achieve the rounded corners we will use `border-radius` on the `<form>`

Forms have `<input>` tags. These inputs are where users type information. By default the inputs are a bit cramped, let's apply a bit of padding to the input tag. This will give a bit of breathing room to the input.

Finally, you may have noticed that in the mockup that the submit button is a bit more styled than the default one the browser sets.

Let's start by creating a class called `.submit`. Apply it to the HTML submit input like below:

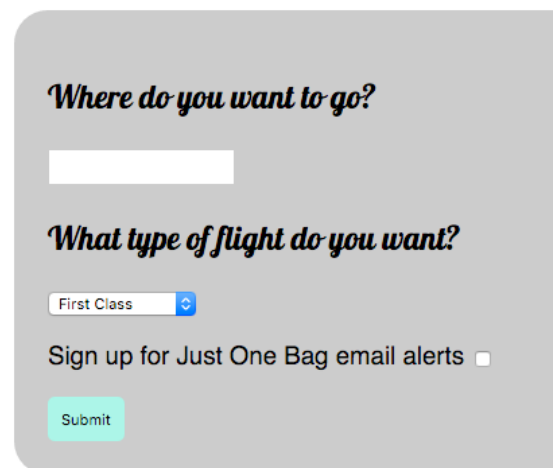
```
<input type="submit" class="submit">
```

Use the mockup to set the text color and background-color. Use padding and `border-radius` to space and round the button as well. Finally you can remove the default border by setting `border` to 0.

CSS GRIDS

A grid system in web development is a technique of separating your page into rows and columns. The idea is based on graphic design principles, specifically the idea that a page can be divided into 12 horizontal units. If something goes 1/3 of the webpage width, that would be 4 units, half of the webpage width would be 6 units, and so on.

We will write a simple grid system. It will combine setting a CSS width to a column based on the 12 unit idea; it will also combine the notion of CSS floating to have columns stack up against one another horizontally, creating page columns.



Where do you want to go?

What type of flight do you want?

First Class

Sign up for Just One Bag email alerts ☐

Submit

Add the following CSS code

```
.column {  
float: left;  
}  
.col-12 {  
width: 100%;  
}  
.col-9 {  
width: 75%;  
}  
.col-6 {  
width: 50%;  
}  
.col-4 {  
width: 33.33%;  
}  
.col-3 {  
width: 25%;  
}  
.row:after {  
clear: both;  
content: " ";  
display: table;  
}
```

This grid system will set the width of each major section. Use the wireframe.pdf as a reference to set the grid classes. For instance, in the wireframe the #logo area takes up 50% width of the page. You would apply the following:

```
<section id="logo" class="col-6 column">  
    
</section>
```

The .col-6 sets the width of the #logo section. The .column make the section float to the left, causing the content underneath to float up, the result is creating actual columns that stack horizontally.

Apply the rest of the CSS grid structure. Use wireframe.pdf as a reference.

Rows

You may notice that we lost the background color for the header. You may also have noticed that some areas are accidentally wrapping around other areas causing some odd layout. We need to separate content into rows, rows will divide content vertically for us.

What we will do is wrap all of the columns for a row inside a `<div>` labelled with CSS class called `.row`.

The `.row:after` uses the CSS clear property. Clears prevent content from wrapping around floated items. Thus, having the clear applied to the row prevents rows from wrapping around one another, the result being they separate vertically. See below as an example:

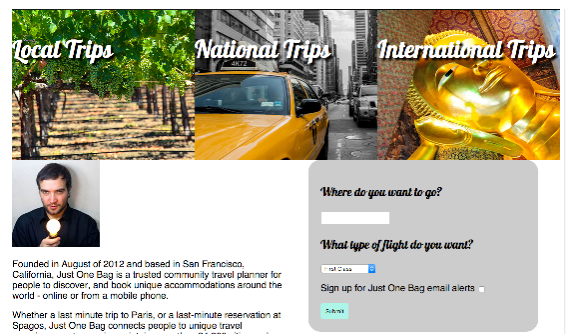
```
<div class="row">
  <section id="logo">
    ...
  </section>
  <section id="nav">
    ...
  </section>
</div>
<div class="row">
  <section id="tagline">
    ...
  </div>
```

Use Wireframe.pdf as a reference to where the rows should be placed.

Your results should roughly match the image on the right. Check the link below to see how your page should look at this point.

POSITIONING THE FORM

The form is a special case in this mockup because we have both the container for the form AND the form itself. In the mockup the form area takes up 50% of width of the page, however, the actual form is less than 50%. There are many ways we could solve this but we'll use a width WITHIN a width to solve this.



Set the width of the <form> to 75%.

```
form {  
  width: 75%;  
}
```

That will make the form 75% of whatever the form block is. For example if the form block was 100px wide then the form would be 75px wide. If the form block was 200px wide then the form would be 150px wide and so on.

Setting the width is great, but the result is a form that is always on the left side of the form section. To make the form sit in the middle we will use the css technique known as margin:auto.

When margin: auto is applied to a block element that is less than the width of its parent it will automatically be centered within the parent. In this case the form <section id="form"> is the parent and the <form> is the child.

```
form {  
  width: 75%;  
  margin: auto;  
}
```

HELPER CLASSES

Ok, we've really come far with this webpage. Let's begin to use CSS to make small changes such as aligning text or moving items within a column. To do this we'll make what are usually referred to as "helper classes". Helper classes are non-specific classes that designers create to apply anywhere in a website when they are needed. Let's make a few!

Add `*/ == Helper Classes ==*/` to your stylesheet and add the following:

```
.float-left {  
  float: left;  
}  
.float-right {  
  float: right;  
}
```

```
4  
5 ▼ .float-right {  
6   float: right;  
7 }  
8  
9 ▼ .float-left {  
10  float: left;  
11 }  
12  
13 ▼ .text-center {  
14   text-align: center;  
15 }  
16  
17 ▼ .text-left {  
18   text-align: left;  
19 }  
20  
21 ▼ .text-right {  
22   text-align: right;  
23 }  
24
```

```

.align-right {
text-align: right;
}
.align-center {
text-align: center;
}
.align-left {
text-align: left;
}
.padding-small {
padding: 10px;
}
.padding-large {
padding: 25px;
}
.img-circle {
border-radius: 50%;
}

```

The idea is that we keep the names generic like *padding-small* so we can modify the value later if need be. 'Small' is subjective so changing it from 10px to 15px still makes sense to others reading your code.

Let's begin with the nav bar. You'll notice in the mockup that the navigation links are aligned to the right side of the webpage. Defaultly they are aligned to the left side of the #nav container. Let's fix this with a helper class. Add the following:

```

<nav class="align-right">
...
</nav>

```

The result is that all of your navigation links have moved to the right side of the page. Let's continue with adding center alignment to each destination area: #local, #national, #international.

Let's also add the float-left to the ceo image so the remaining text will wrap around it. Save and preview.

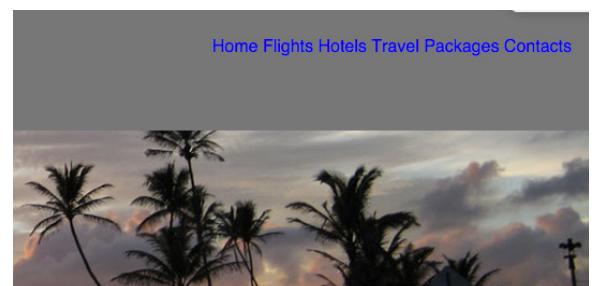
Awesome! Now let's use the helper class we made for turning images into circles. That was the .img-circle class we created.

Finally, let's add a bit of padding so that the logo and the nav-

```

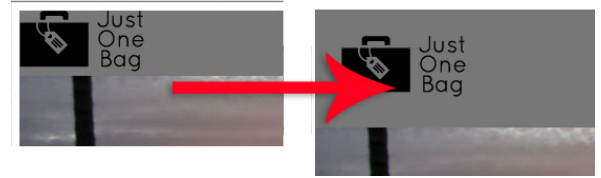
ction id="nav" class="col-6
<nav class="text-right">
  <a href="#">Home </a>
  <a href="#">Flights </a>
  <a href="#">Hotels</a>
  <a href="#">Travel Pack
  <a href="#">Contacts</a>
</nav>
ction>

```



igation do not touch against the side of the page. Let's do that by applying the padding-large to the `<div class="row">` that is containing the columns.

Apply the class and you'll see that the row now has a bit of padding pushing the elements away from the edge.



Let's apply the padding class to the footer columns as well. As you may notice the columns in the row are tight against one another. Let's apply padding-small to each section, that will give EACH column a bit of padding

ODDS AND ENDS

Alright, we are almost done. We just have to put on the finishing touches to the webpage. So let's begin:

SOCIAL MEDIA BAR

By default lists render content vertically, or one on top of another. That is because ``s items are block elements, and by default block elements are displayed vertically. However, if we look at the mockup we want the social media bar to go horizontally. To fix this we can change how `` render on a webpage.

To do this we will use the CSS property `display`. `display` allows us to change block elements to inline elements, which will cause items displayed vertically to display horizontally, and vice versa.

Add the following code:

```
#social-media li {  
    display: inline;  
}
```

Save and you should see your image displayed in a horizontal line as opposed to a vertical line. We can space each image from one another using `margin-right`.

To remove the move the bullets use the CSS property `list-style-type` and set it to `none`. Defaultly there is margin and padding on the `` which causes it to be pushed a bit to the right. You can remove that using `margin-left: 0` and `padding-left: 0`.



ARRANGE TEXT TO VERTICAL CENTER

As you may have noticed that our destination text sits at the top of the section, but in the mockup it sits in the vertical center of the section. Unfortunately there is no vertical-align property BUT what we can do is use the property line-height.

Add the following

```
.destination {  
  height: 300px;  
  line-height: 200px;  
}
```

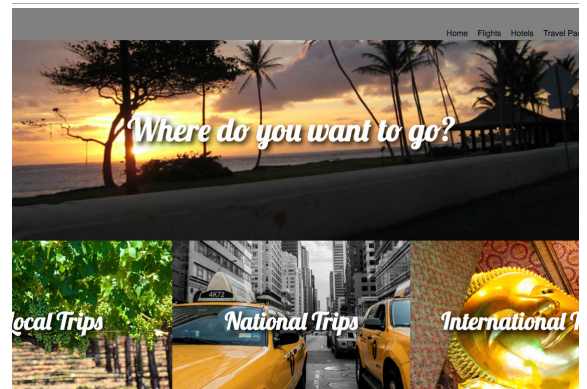
this will position the text to the center of the box.

REMOVE MARGINS

Sometimes there will be margins and paddings that are applied by the user agent. These are typically applied to block tags like `<body>` and headings. Sometimes we want to remove the default padding or margin to improve the design.

In our footer you'll notice that the text "Partnerships" is pushed down a bit from the other items in the footer. That is because Partnerships is inside an `<h3>`, `<h3>` have default margin on the top and bottom. Let's remove that by specifically targeting JUST the partnership h3

```
#partnerships h3 {  
  margin-top: 0;  
  margin-bottom: 0;  
}
```



NAVIGATION BAR

If we look at the mockup we can see that the navigation bar sits along the baseline of the logo. However, in our webpage that navigation links are sitting at the top. We can fix that by applying a margin to the entire navigation bar. To do this we'll target the `<nav>` tag.

```
nav {  
  margin-top: 100px;  
}
```

That will push the navigation down to the baseline of the logo.

We can give a bit of space to our navigation links by apply a bit of margin-right to each navigation link.

HOVER EFFECTS

Hover effects are when an item changed it's look or style when the mouse is hovered over it. This is typically applied to hyperlinks and buttons.

Let's begin by making a hover effect for our links. We'll use the pseudo class hover for this. To make sure we only affect the hyperlinks in the navigation area and in the footer let's use CSS compounds

```
nav a:hover, footer a:hover {  
  color: #8C5A6E;  
}
```

Save and test in a browser.

Let's also set this up for the submit button in our form. Make a psuedo class for our `.submit` class we made earlier.

```
.submit:hover {  
  background-color: #8C5A6E;  
}
```

Congratulations, hopefully this helped understand the process of creating a sophisticated webpage from start-to-finish. There is no 'Right' way, but its a good idea to start with small issues like type and color and work our way up to position and placement.

