

# HW4 Programming: Styling and Populating Your Crowdfunding Site:

## Instructions

---

When you're ready to submit your solution, go to the [assignments list](#).

[UP](#) | [HOME](#)

## Homework 4

### Background

In this programming assignment we are going to modify the crowdfunding site we deployed the last time in the following ways.


1. Set up EC2 and Chrome to allow instant desktop and mobile previews.
2. CSS style `index.html` and add basic mobile compatibility with Twitter Bootstrap.
3. Start populating your product front page with marketing copy.

This is the second step towards our final project. [Lecture 5](#) and [6](#) will be helpful on marketing and HTML/CSS/JS respectively. You will be graded mildly on the technical aspects of this process (presence of CSS styles and mobile compatibility), but it is the qualitative aspects that you want to start focusing on for your final project.

### Preliminaries: Set up an EC2 Instance which allows instant previews

First, set up an EC2 instance which allows HTTP connections on ports 80 and 8080. This will allow you to debug your `index.html` page while previewing in a browser, rather than copying it locally or always pushing to Heroku. To do this you will want to go to the AWS Dashboard, set up a new security group, and then launch an instance with that security group (you can also change the security group of an existing instance). Take a look at the screenshots below, which create a security group named `quicklaunch-1` that allows the EC2 box to be accessed via ports 22 and 80.

← → ↺ <https://console.aws.amazon.com/ec2/home?region=us-east-1#s=SecurityGroups>

 **Services** ▾ **Edit** ▾

EC2 Dashboard

Events

INSTANCES

- Instances
- Spot Requests
- Reserved Instances

IMAGES

- AMIs
- Bundle Tasks

ELASTIC BLOCK STORE

- Volumes
- Snapshots

NETWORK & SECURITY

- Security Groups**
- Elastic IPs
- Placement Groups
- Load Balancers
- Key Pairs
- Network Interfaces

Create Security Group Delete





Viewing: EC2 Security Groups Search

	Name	VPC ID	Description
<input type="checkbox"/>	default		default group
<input checked="" type="checkbox"/>	quicklaunch-1		quicklaunch-1
<input type="checkbox"/>	ElasticMapReduce-slave		Slave group for Elastic MapReduce
<input type="checkbox"/>	ElasticMapReduce-maste		Master group for Elastic MapReduce

↓

Edit security group ports below.

1 Security Group selected

 **Security Group: quicklaunch-1**   

Details **Inbound**

Create a new rule:

Port range:

Source:

- ✓ Custom TCP rule
- Custom UDP rule
- Custom ICMP rule
- All TCP
- All UDP
- All ICMP
- SSH
- SMTP
- DNS
- HTTP
- POP3
- IMAP
- LDAP
- HTTPS
- SMTPS
- IMAPS
- POP3S
- MS SQL
- MYSQL
- RDP

TCP	Port (Service)	Source	Action
	22 (SSH)	0.0.0.0/0	Delete

## 1 Security Group selected

## Security Group: ssh-http-https

Details

Inbound

Outbound

Create a new rule: Custom TCP rule

Port range: 8080  
(e.g., 80 or 49152-65535)

Source: 0.0.0.0/0  
(e.g., 192.168.2.0/24, sg-47ad482e, or 1234567890/default)

SSH, HTTP, HTTPS are built in options  
But for port 8080 you need a  
Custom TCP rule as shown.

+ Add Rule

Apply Rule Changes

## 1 Security Group selected

## Security Group: ssh-http-https

Details

Inbound\*

Outbound

Create a new rule: Custom TCP rule

Port range:   
(e.g., 80 or 49152-65535)

Source: 0.0.0.0/0  
(e.g., 192.168.2.0/24, sg-47ad482e, or 1234567890/default)

+ Add Rule

Your changes have not been applied yet.

Apply Rule Changes

TCP Port (Service)	Source	Action
22 (SSH)	0.0.0.0/0	Delete
80 (HTTP)	0.0.0.0/0	Delete
443 (HTTPS)	0.0.0.0/0	Delete
8080 (HTTP*)	0.0.0.0/0	Delete

1) This is what your final list of rules should look like.

2) Then click here.

## Create a New Instance

Cancel

## Ubuntu Server 12.04.1 LTS (ami-3d4ff254)

Platform: Ubuntu  
Architecture: x86\_64

Ubuntu Server 12.04.1 LTS with support available from Canonical  
(http://www.ubuntu.com/cloud/services).

Please review your settings and click **Launch** to finish or **Edit details** to make changes.

## Instance Details

Name:  Type: t1.micro

Detailed Monitoring: No Availability Zone: No preference

Shutdown Behaviour: Stop Termination Protection: No

Launch into a VPC: No

## Security Details

Key Pair: cs184-john-stanford-edu

Security Group: quicklaunch-0

## Advanced Details

Kernel ID: Default

Ramdisk ID: Default

User Data: 

IAM Role: ?

MAKE SURE to click edit details, as you want the new security group you just created (not the quicklaunch-0 that was selected by default, but the quicklaunch-1 we just set up in our example)

&lt; Go Back

Edit detailsLaunch

## Create a New Instance

Cancel X

## Ubuntu Server 12.04.1 LTS (ami-3d4ff254)

Platform: Ubuntu  
Architecture: x86\_64

Ubuntu Server 12.04.1 LTS with support available from Canonical  
(<http://www.ubuntu.com/cloud/services>).

Click **Save details** in order to save your changes and return to the review screen.

☐ Instance Details☐ Modify Tags☒ Security Settings

Security groups determine whether a network port is open or blocked on your instances. You may use an existing security group, or we can help you create a new security group to allow access to your instances.

☐ Create new Security Group☒ Select Existing Security Groups

default  
quicklaunch-1  
ElasticMapReduce-slave  
quicklaunch-0

(Selected groups: quicklaunch-1)

☐ Advanced Details☐ Storage Device Configuration

Save details

Launch ▶

Note again that you don't need to launch a new instance, but can apply this to the security group of a running instance. If you do launch a new instance, you'll want to run `setup.sh` to reinstall everything and use `git clone` to pull down your bitstarter repository again from github. You should be in the directory with your `index.html` again, as shown:

```
ubuntu@ip-172-31-36-95:~/bitstarter — ssh — 80x24
[ubuntu@ip-172-31-36-95:~/bitstarter]$ head index.html
<!DOCTYPE>
<html lang="en">
  <head>
    <title>My First Bitstarter</title>
    <meta charset="utf-8">
    <link type="text/css" rel="stylesheet"
      href="http://twitter.github.com/bootstrap/assets/css/bootstrap.css">
    <style type="text/css">
      body {
        padding-top: 20px;
[ubuntu@ip-172-31-36-95:~/bitstarter]$
```

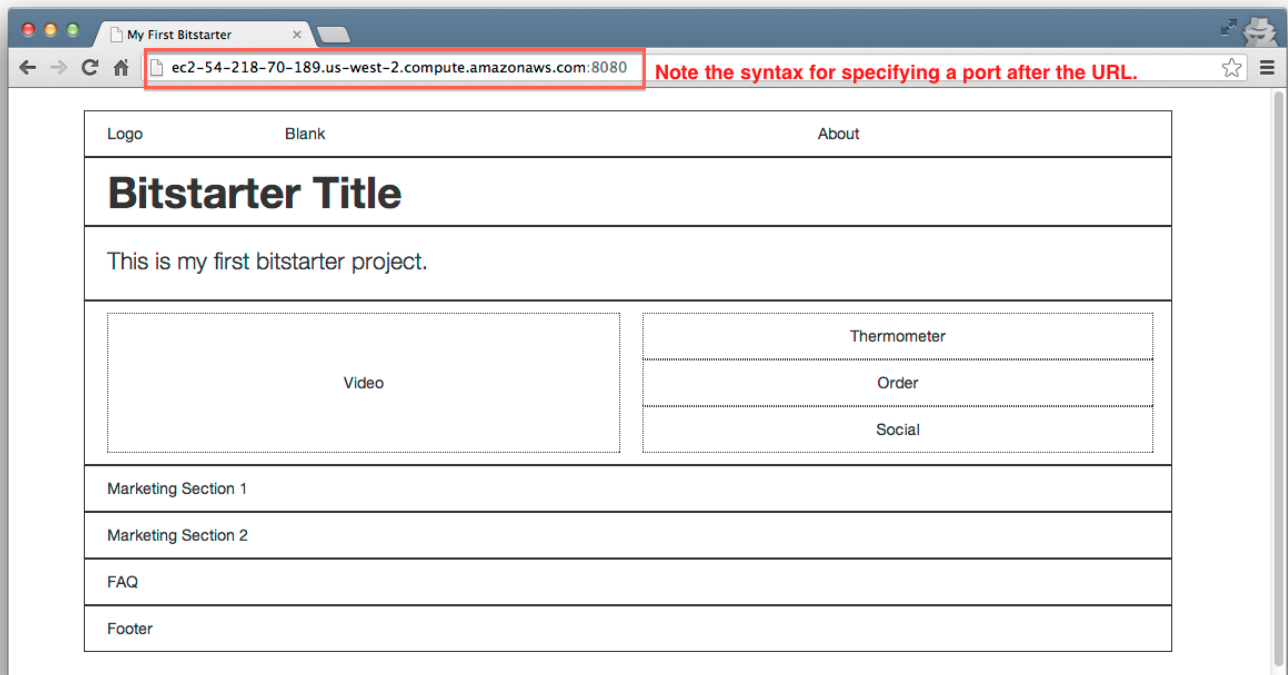
Once your EC2 instances is once again provisioned and available, edit your `web.js` file to configure it to port 8080 and then run `node web.js`. You should now be able to see your `index.html` in the browser, as shown in the following screenshots:

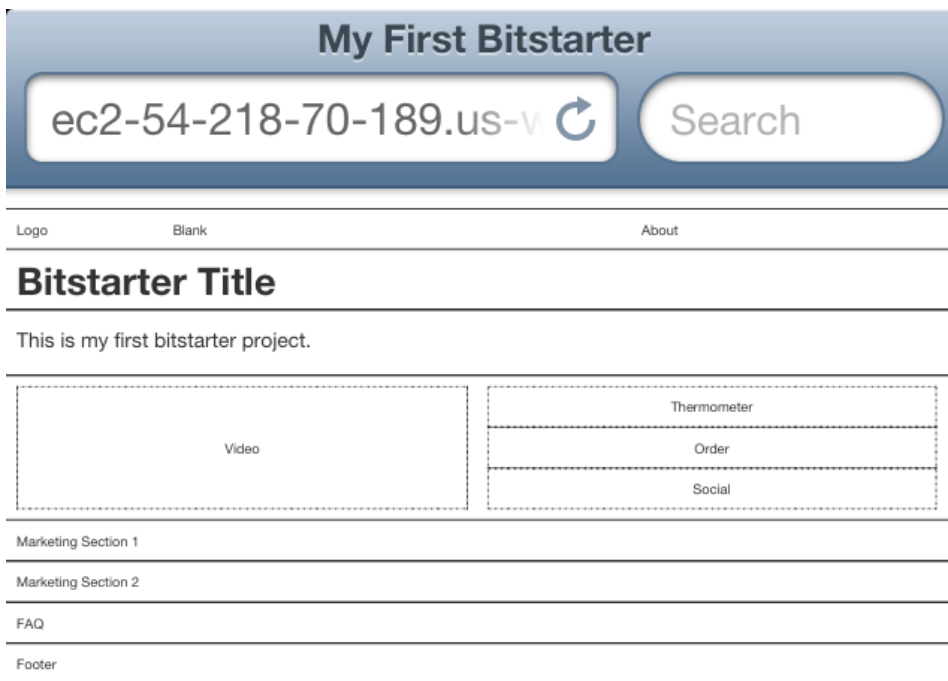
```
ubuntu@ip-172-31-36-95:~/bitstarter — ssh — 80x24
[ubuntu@ip-172-31-36-95:~/bitstarter]$cat web.js
var express = require('express');
var fs = require('fs');
var htmlfile = "index.html";

var app = express.createServer(express.logger());

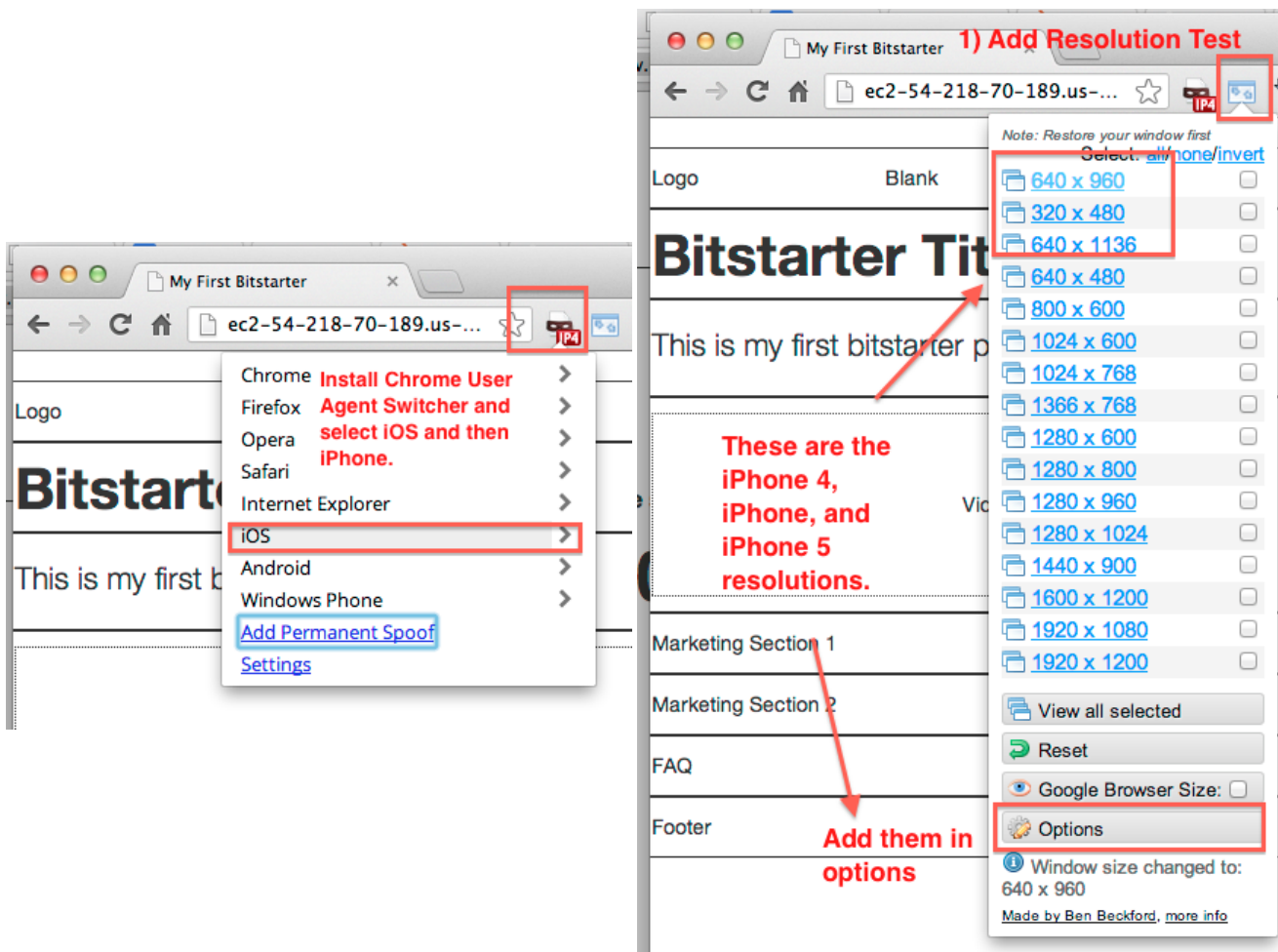
app.get('/', function(request, response) {
  var html = fs.readFileSync(htmlfile).toString();
  response.send(html);
});

var port = process.env.PORT || 8080;
app.listen(port, function() {
  console.log("Listening on " + port);
});
[ubuntu@ip-172-31-36-95:~/bitstarter]$node web.js
Listening on 8080
```





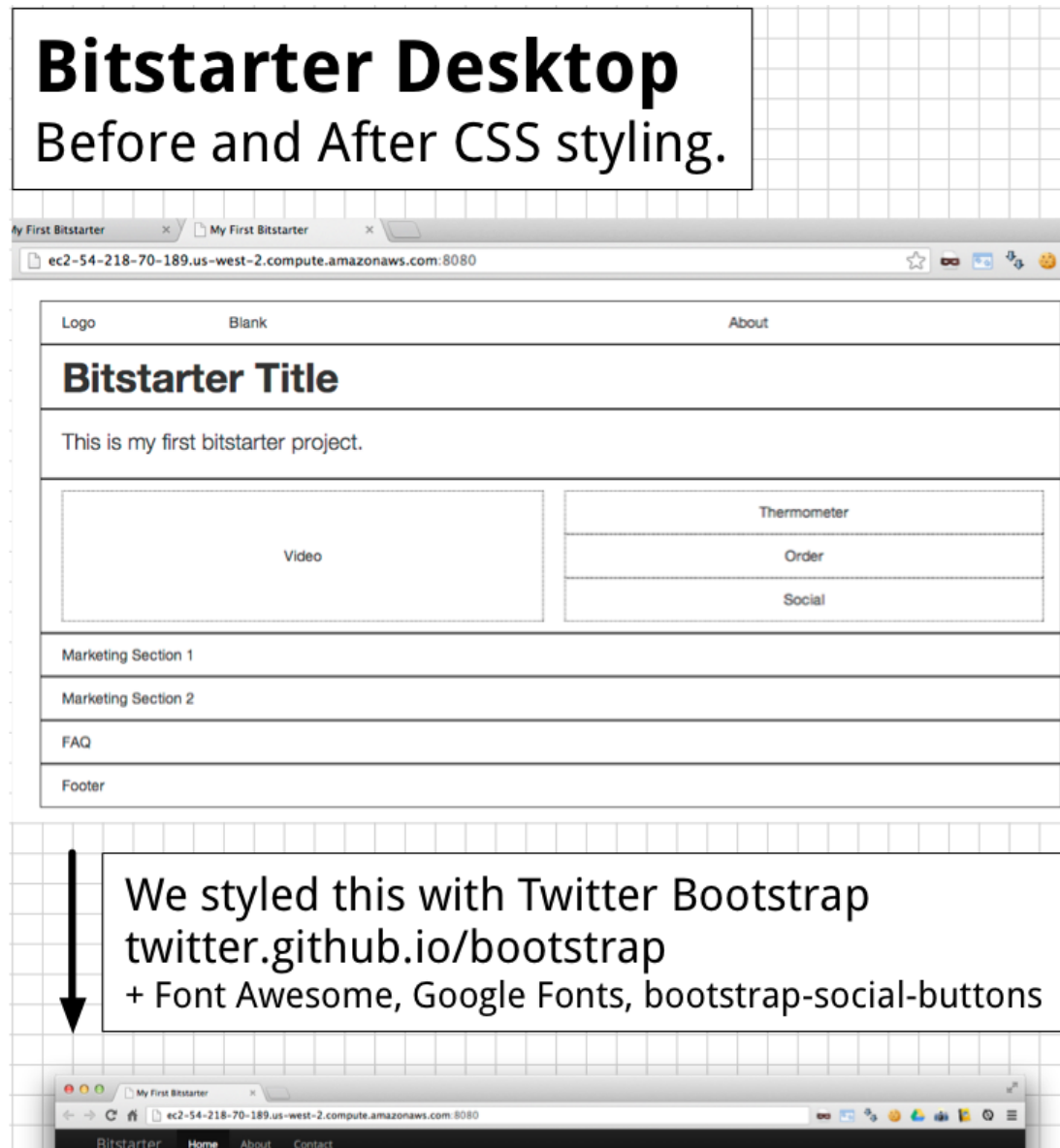
Note that the last screenshot shows a mobile phone browsing to the URL. This works because by opening up the security group, you made your EC2 instance accessible over port 8080 over the public internet. However, it's a little inconvenient to keep refreshing your mobile phone. Instead, one of the best ways to preview how your website will look with a mobile client is by installing Chrome's [User Agent Switcher](#) and its [Resolution Test](#) tools. Then navigate to the same URL and set things up as follows:

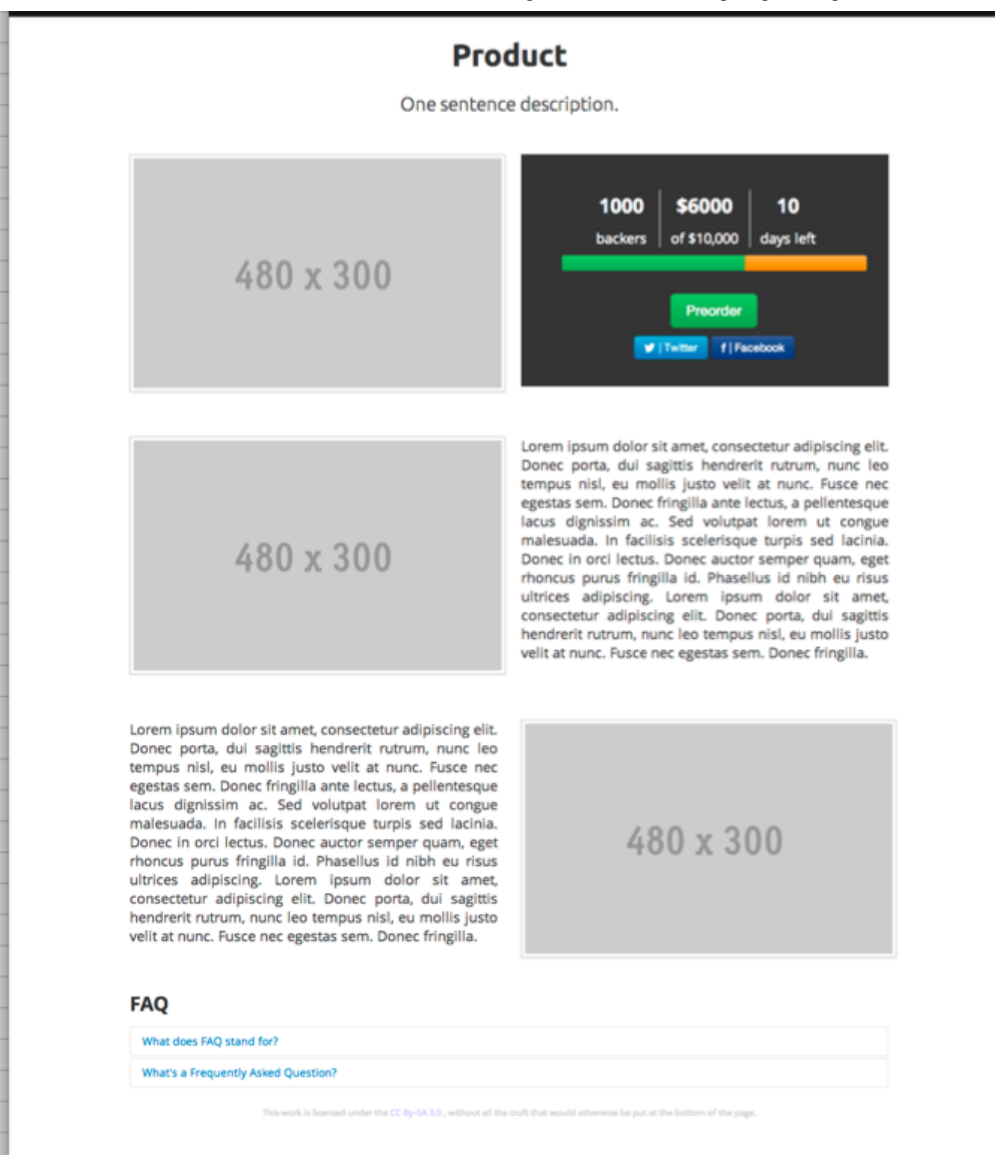


One way to test that you've configured this properly is to go to [airbnb.com](http://airbnb.com) and see if you get automatically redirected to [m.airbnb.com](http://m.airbnb.com). If so, great work - you've just set things up so that you can edit and preview your mobile site.

## Part 1: Add CSS styles and enable responsive design via Twitter Bootstrap

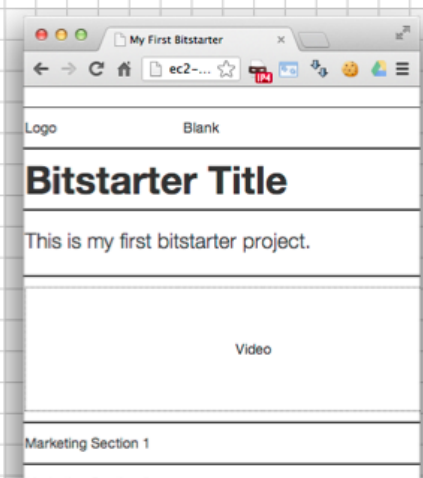
1. In this part we will modify the HTML file from the previous lecture to make it look reasonably good on mobile and to prettify it via CSS. We'll do both of these things with [Twitter Bootstrap](#). Use the content from Lectures 2-4 to edit the file and review Lecture 6 for an overview of how HTML, CSS, and JS interact. We assume throughout that by this point you are reasonably familiar with git, emacs, screen and the like and will be using git add/commit/push to save your work periodically and push it to github, so we won't remind you to do this each time.
2. To start, review the Twitter Bootstrap documentation, paying particular attention to the [examples](#) and the [components](#). We're going to use this framework to style the index.html page, setting up various buttons that will be nonfunctional for now but will be connected to backend events later.
3. First, though, we'll set up the page scaffold for mobile support via responsive design. The concept here is that the majority of your users will be doing most of their website views from mobile browsers via tablets and smartphones, and soon also through new devices like Google Glass. However, outside of the relatively few screen sizes of iPads and iPhones, it's difficult to make assumptions about mobile [screen sizes](#). [Here is a picture](#) from more than a year ago which gives some idea of the diversity of mobile screen sizes. Thus, rather than try to do a custom UI that is pixel perfect for a fixed-width screen, we try to set up the site so that it naturally scales up and down to different screen resolutions. You can get very fancy here (see <http://mediaqueri.es> in particular for inspiration), but for the purposes of this class we'll keep it simple by using Bootstrap's tools for [responsive design](#).
4. At the risk of being accused of asking you to [draw the Tick](#), let's go through in mechanistic detail exactly how we did the conversion from our original wireframe to this responsive design. First, take a look at the before/after images below to see what our goal was in the styling. Then click [the link](#) to see how we did the styling.





## Bitstarter Smartphone

### Before and After CSS styling.



Viewed in Chrome with  
User Agent Switcher  
and Resolution Test

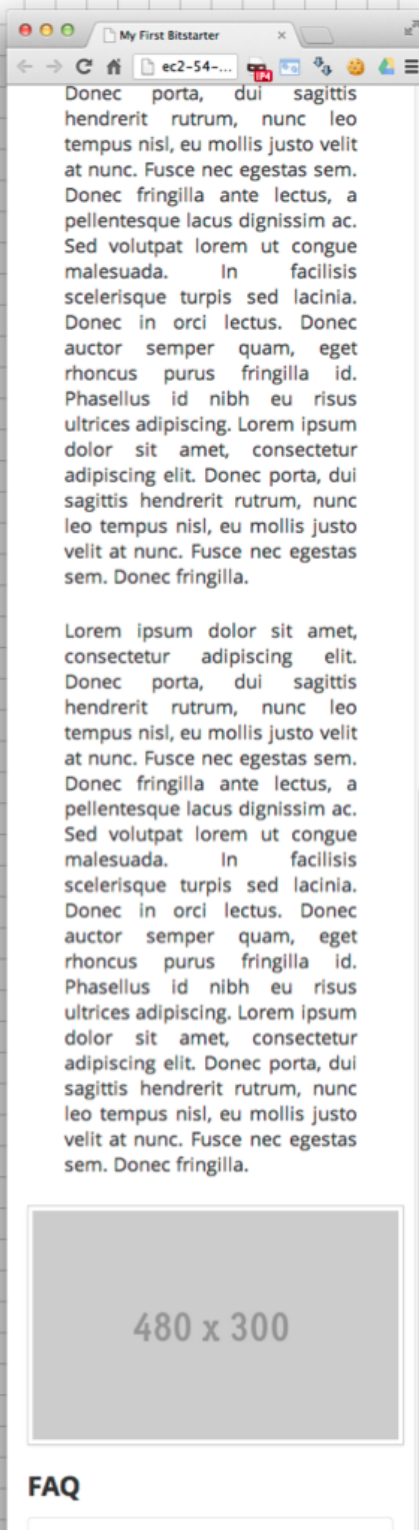
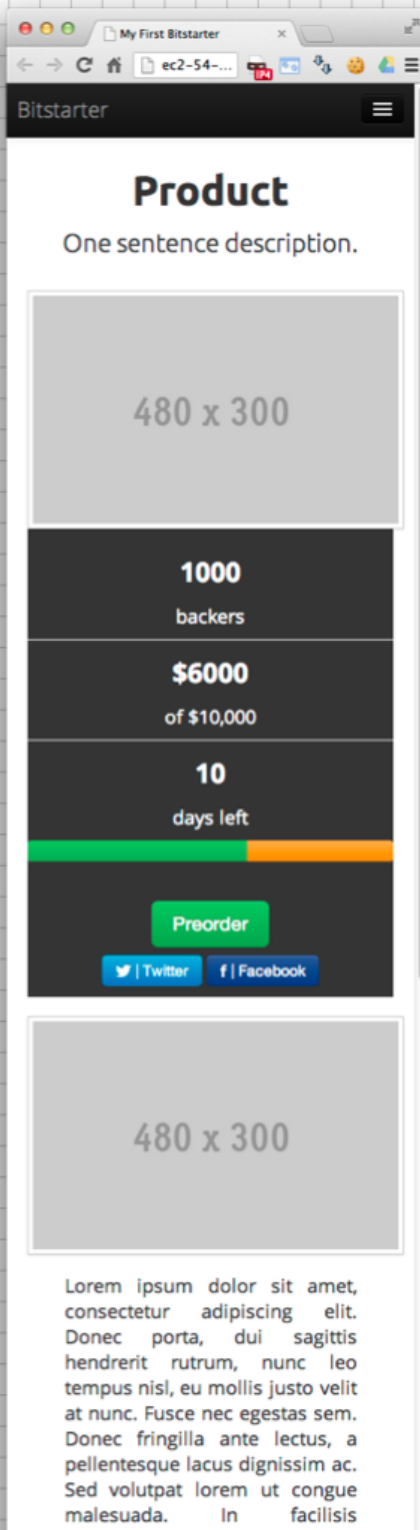


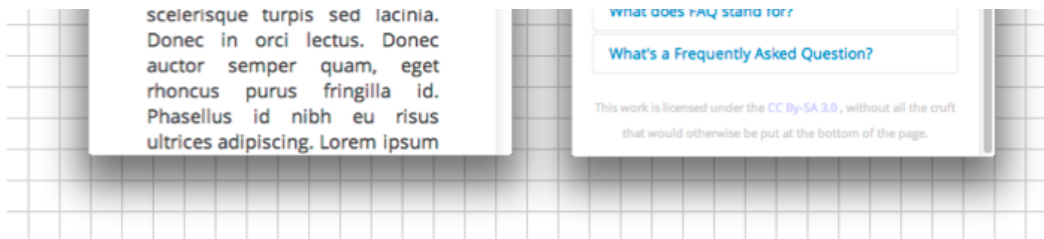
Marketing Section 2

FAQ

Footer

We styled this with Twitter Bootstrap  
[twitter.github.io/bootstrap](https://twitter.github.io/bootstrap)  
+ Font Awesome, Google Fonts, bootstrap-social-buttons





## Part 1: Responsive Design and CSS Styling

We've annotated corresponding portions of the desktop and smartphone render. Apply the changes step-by-step to transform your unstyled index.html. Along the way, preview your changes and save them with git. Finally, when done, submit the github URL to the index.html of your styled file.

This image is very large, so go here for the PDF version:

<https://spark-public.s3.amazonaws.com/startup/images/part1-instructions.pdf>

**DOWNLOAD [THE BIG DIAGRAM \(PDF\)](#) TO SEE HOW TO STYLE THE PAGE.**

**DON'T MISS THAT LINK, IT IS A HIGH RES PDF, TOO LARGE AS A PNG.**

### Assignment

Your mission is to use the source code annotations in the [big diagram](#) and your text editor to gradually transform your old `index.html` from HW3 into a new styled version, previewing the results in the browser along the way. Try to go div by div and see how the page changes over time. If you have a vision or reading disability, click [here](#) for the new index.html, but you'll get much more out of the process by typing in these changes step by step, seeing the results in the browser, and then doing a git commit.

### Submit

Submit the direct github URL to the commit with your styled, responsive `index.html`. It should look something like this:

```
https://raw.githubusercontent.com/your-username/bitstarter/90fd58e343c29e54fc1d4e19036595500b2bdc22/index.html
```

## Part 2: Populate Your Styled CSS page with your own content

Here's the part where your creativity comes in. *You should figure out what product you want to market for the final crowdfunder project.* You don't actually need to build the entire thing (though you can) to sell preorders; you just need to be able to bring it to life enough to determine whether a market exists. This might seem backwards, but the basic idea is to confirm through empirical measurement of preorder volume that there is genuine demand *before* investing enormous amounts of time and effort. Preorder volume need not be monetary; it can be sharing volume on social media if you don't want to collect real money.

### Choose a project which you can get to a crowdfunder

People in the class have widely varying backgrounds, ranging from relative neophytes to fairly advanced students. Your goal is to figure out a final project that you can get to the level of a reasonable crowdfunding site. This can mean anything from verbal description and mockups to fully working code, or anything in between: whatever is necessary to make the sale and prove that you have a market. Note that you have control over who's seeing the crowdfunder (e.g. the people in your email list or social circle), so you are also determining the difficulty of the sale itself by choosing the market and the initial crowdfunding audience. Note also that you can just ask for someone to share your content on social media rather than pay you real money for a preorder. Open source projects are ok as well, not just for-profit businesses. The main guideline is to set your success bar at the point that it'll be an achievable challenge for you. That might be a target of \$100 or it might be \$100,000.

### Some sample ideas

If you can't think of anything, here are a few ideas that we consider interesting and potentially fundable. Important: we do not expect you to

necessarily build these out in a full fledged way during the class (though we don't want to stop you from doing that - it would be very impressive!). Instead, the expectation is that you can go from ideas like this to a [crowdfunder page](#), complete with your own images, video, marketing copy, etc. Coming up with your own idea is the ideal, but here are a few to get you started:

1. A telepresent tourist service that uses supervised [Double Robotics](#) units to offer snap world tours as a cheaper alternative to the expense and logistical overhead of world travel and the hassle of tourist visas. So, in one day your customers would wander around the Louvre, the Smithsonian, the Grand Canyon, and the Pyramids, taking photos from the comfort of their home, with an on-site tour guide opening doors and packing up the Double Robotics units at the end of the day. Market: people who can't afford travel but are interested in tourism.
2. A 2.0 version of [Amazon Mechanical Turk](#). Specifically: a native mobile app, which takes payments in Bitcoin and which allows customers to select workers with particular skills as endorsed by [LinkedIn](#) to allocate work. This would allow an end user to pay for higher skill tasks by (e.g.) selecting only lawyers and paying \$20/hour in BTC for document review. Market: on one side engineers who need repetitive work automated, and on the other side casual workers or workers in developing countries.
3. Jurisdiction selector. Given your personal circumstances and your desired goals, recommend a rank-ordered list of the optimal places in the world to [relocate](#) to and give an estimated cost of relocation. Market: initially market to college students and people starting out in their careers; ultimately market to people at various transition points (e.g. MDs graduating from medical school, veterans returning home) and expand the [executive relocation](#) services market.
4. A review site for federal regulators in your country, along the lines of [thefunded.com](#) but for regulation. Market: advertising and consulting referrals for executives in regulated industry.
5. A Kinect-based app which [3D scans your face](#) and those of your parents, brother, and relatives. Outputs a score based on facial morphology which determines who you resemble the most, both in terms of general morphology and particular expressions (smiling, frowning). Market: this one is for fun, but could be interesting to eventually incorporate as part of a video game or perhaps for applications in plastic surgery.
6. Find new customers. App for service providers like lawyers, doctors, plumbers, mechanics, and the like. Mine Twitter, Facebook, etc. to find concentrations of people nearby who are likely to need the services of these people. Use these same social networks to rank order said potential customers by social proximity. Hit a button to approach a few of them electronically with customized emails. Note: while you want to assuredly stay far away from enabling spam, you will realize when doing this that for many people "spam is a marketing message they found irrelevant". Market: any service provider who can type in some keywords that characterize their prospective customer.
7. Secure phone. A Linux based, completely open sourced phone which gives feedback (via blinking red, vibrating, etc.) before transmitting any kind of signal that could be wiretapped. Kind of a Geiger counter for the NSA. Market: any foreign country (e.g. Russia) which is concerned about information security. This concept is a bit of a lark but could be surprisingly huge if one could also deploy it as a background app on existing smartphones (which would essentially blink red, vibrate, etc. all the time).
8. A neighborhood drone watch. Take a fleet of [Parrot AR Drones](#), equip them with cameras, microphones, speakers, and floodlights, and orchestrate the whole shebang with the [ar-drone](#) node module. Set up a command center and sell the package to high crime areas (the kinds of places that [ShotSpotter](#) might already be in use). Now one person can see and interrupt crimes via remote control without fear of personal harm. Market: any area with a mix of high crime and medium-to-high income residents.
9. An API that shows all regulations that are applicable to a given object, along with the local personnel responsible for enforcing them and their contact information. So, for example, scan a barcode or submit the [EIN](#) of a business to see all applicable local, state, and federal laws as well as the points of contact. It'd be really fascinating if you could make this into an augmented reality like app, to get a [Yelp Monocle](#)-like view of the world. Market: any executive in regulated industry, small business owner, or consultant.
10. Your own idea. Come on, we're sure you have something!

### Assignment: fill in your bitstarter with your own marketing copy and images.

Once you've got your idea and you've finished applying CSS styles to your bitstarter template, it's time to start filling in your bitstarter with your own marketing copy and images. Take a look at the [Kickstarter School](#) for inspiration. Just focus on the front page for now. Use the wireframing tools described in lecture 5 to make graphics that help potential customers understand what it is you're building. Once you've zeroed in on a product and have some visuals, start thinking about product tiers as well using "Product Tier Selection: A Conceptual Framework" from Lecture 5. A video, though not mandatory, is a very important piece of a crowdfunder (see "[Making Your Video](#)"); if you want to get ahead of the curve you should start doing some [storyboarding](#) with some [templates](#).

At the end of this section you should have taken your `index.html` and modified it to include your own product name, tagline, and some images and text that describe what your product is about. Make these edits in your git repository and push them to github. If you need image hosting, use [imgur.com](#) or a similar free image host for now, and then use the `img src` syntax that we used previously. We'll get into this soon, but you generally don't want to serve static files like images via `node` (something like [nginx](#) is preferable for non-dynamic content).

## Part 2: Populate your styled page.

Now that you have a scaffold, edit your crowdfunder to include your own content. You should aim for a title, a one sentence description, three placeholder images, two body paragraphs, a FAQ, and a fundraising target. Along the way edit `index.html`, git push your edits, and submit the URL.

**Populate  
with your  
own  
text and  
images.**

**Product**  
One sentence description.

480 x 300



480 x 300

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Donec porta, dui sagittis hendrerit rutrum, nunc leo tempus nisl, eu mollis justo velit at nunc. Fusce nec egestas sem. Donec fringilla ante lectus, a pellentesque lacus dignissim ac. Sed volutpat lorem ut congue malesuada. In facilisis scelerisque turpis sed lacinia. Donec in orci lectus. Donec auctor semper quam, eget rhoncus purus fringilla id. Phasellus id nibh eu risus ultrices adipiscing. Lorem ipsum dolor sit amet, consectetur adipiscing elit. Donec porta, dui sagittis hendrerit rutrum, nunc leo tempus nisl, eu mollis justo velit at nunc. Fusce nec egestas sem. Donec fringilla.

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Donec porta, dui sagittis hendrerit rutrum, nunc leo tempus nisl, eu mollis justo velit at nunc. Fusce nec egestas sem. Donec fringilla ante lectus, a pellentesque lacus dignissim ac. Sed volutpat lorem ut congue malesuada. In facilisis scelerisque turpis sed lacinia. Donec in orci lectus. Donec auctor semper quam, eget rhoncus purus fringilla id. Phasellus id nibh eu risus ultrices adipiscing. Lorem ipsum dolor sit amet, consectetur adipiscing elit. Donec porta, dui sagittis hendrerit rutrum, nunc leo tempus nisl, eu mollis justo velit at nunc. Fusce nec egestas sem. Donec fringilla.

## FAQ

What does FAQ stand for?

What's a Frequently Asked Question?

This work is licensed under the CC BY-SA 3.0, without all the cruft that would otherwise be put at the bottom of the page.

**Determine  
min capital to raise  
(from econ of scale)  
and think about  
product tiers**

## Design Tips

Note that you can also feel free to restyle the page to your heart's content; just make sure that you have earlier versions in your github repository. If you do restyle the page, the following are good things to keep in mind:

1. Generally avoid extreme colors (pure black/white/blue/red/green) and sharp gradients. Go with gradual gradients.
2. If you can't draw, use Google Fonts and iconfinder to do "font-based design". That is, pick a pair of nice fonts (here's a sample [pairing](#) page).

Then minimize the use of bitmap/raster graphics, and go with fonts and vector icons throughout the site. You'll find that this imposes interesting constraints and gives a definite geometric look to the site that can be complemented with a bit of raster at a later date.

3. Try to line things up along vertical edges (as per the Non-Designer's Design Guide).
4. Don't strive for the design to be pixel perfect in every intermediate shape. Just see that it looks reasonably good on a laptop, a tablet, and a phone.
5. Use [lorem ipsum](#) and <http://placeholder.it> fairly liberally.
6. For best compatibility with responsive designs, only use percentages for padding/margin, rather than absolute px.
7. If you do use padding, don't pad horizontally as this will often mess up the grid.
8. Get things looking reasonably nice in one perspective (e.g. desktop). Then resize your browser window and view it with a tablet or smartphone User Agent. If you need to make a few tweaks, that's fine, but keep them scoped within `@media` queries as shown.
9. Using `.row-fluid` within a `.container` actually gives a reasonably good experience; as noted [here](#) ("You can use row-fluid inside a container to get a 12-column (nestable) fluid grid within a pseudo-fixed-width, centred container that steps down in size as the window resizes. This is the best way to achieve a resizable layout.")
10. It's a lot easier to design optimally for a few key widths (Macbook width, iPad landscape/portrait, iPhone landscape/portrait). Even if things don't look perfect between widths, that could be ok if you don't have extremely high polish requirements (and before you have revenue/profit, you probably don't have such requirements). Make sure the site never looks outright broken with intermediate widths, but otherwise don't micro-optimize.
11. If you are doing a serious design, it's a good idea to [pseudo-namespace](#) your CSS to avoid inadvertent (and hard to debug) collisions. We'll introduce some techniques to help automate this in the future, but for now you can just use a prefix (e.g. 'myappname-statistics' rather than 'statistics' for the CSS classes).

## Submit

Submit the direct github URL to the commit with your very latest `index.html`, with the product name and tagline filled in with your own ideas. It should look like this:

```
https://raw.githubusercontent.com/your-username/bitstarter/90fd58e343c29e54fc1d4e19036595500b2bdc22/index.html
```

Note that this is different from what you submitted above. The previous one was just the index.html with CSS styles applied. This one is your first version with your own marketing content and images.