# Reflow

Reflow has your standard controls for editing boxes and text, but there are 4 unique portions that I'd like to document:

Media Query Bar

*This section allows you to see and edit all your breakpoints for media queries.*

Resize Handle

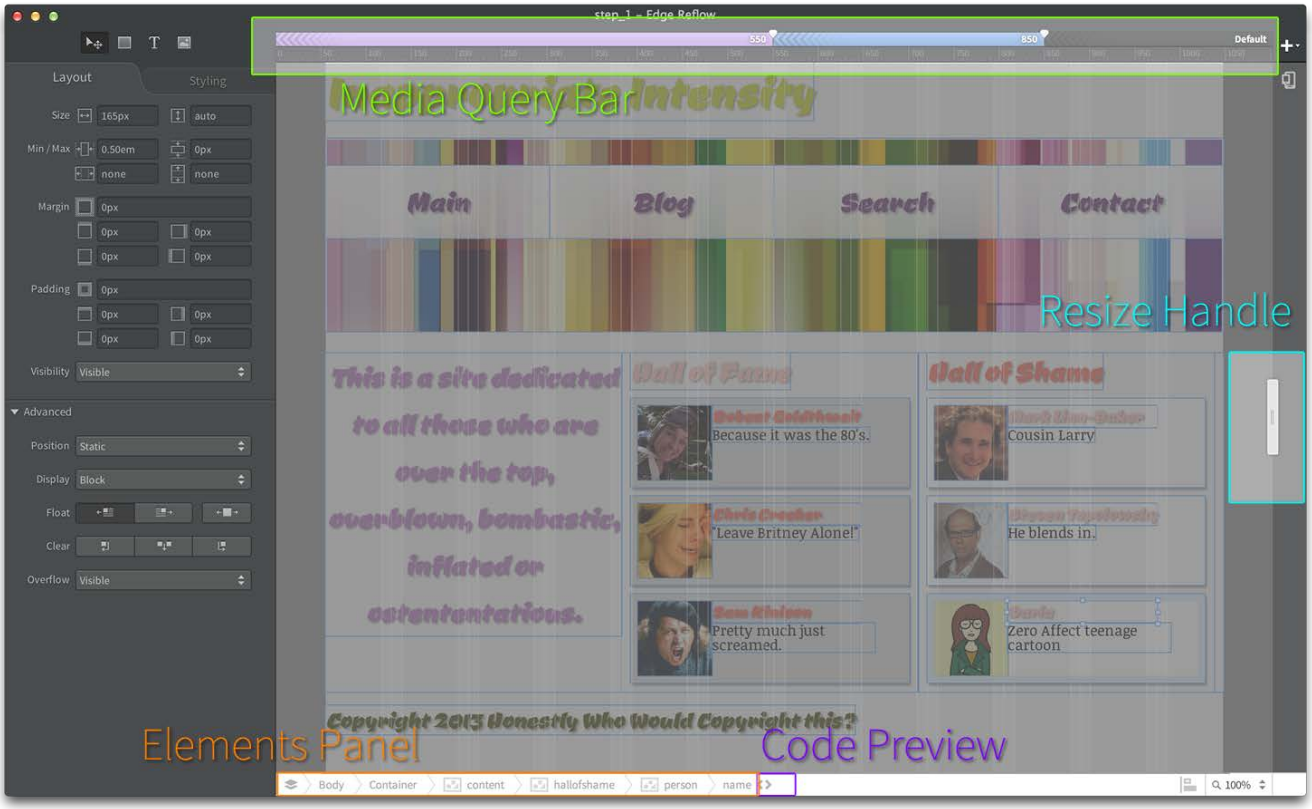
*This handle allows you to resize the design area for either previewing at sizes or for use with the Media Query Bar in setting breakpoints.*

Elements Panel

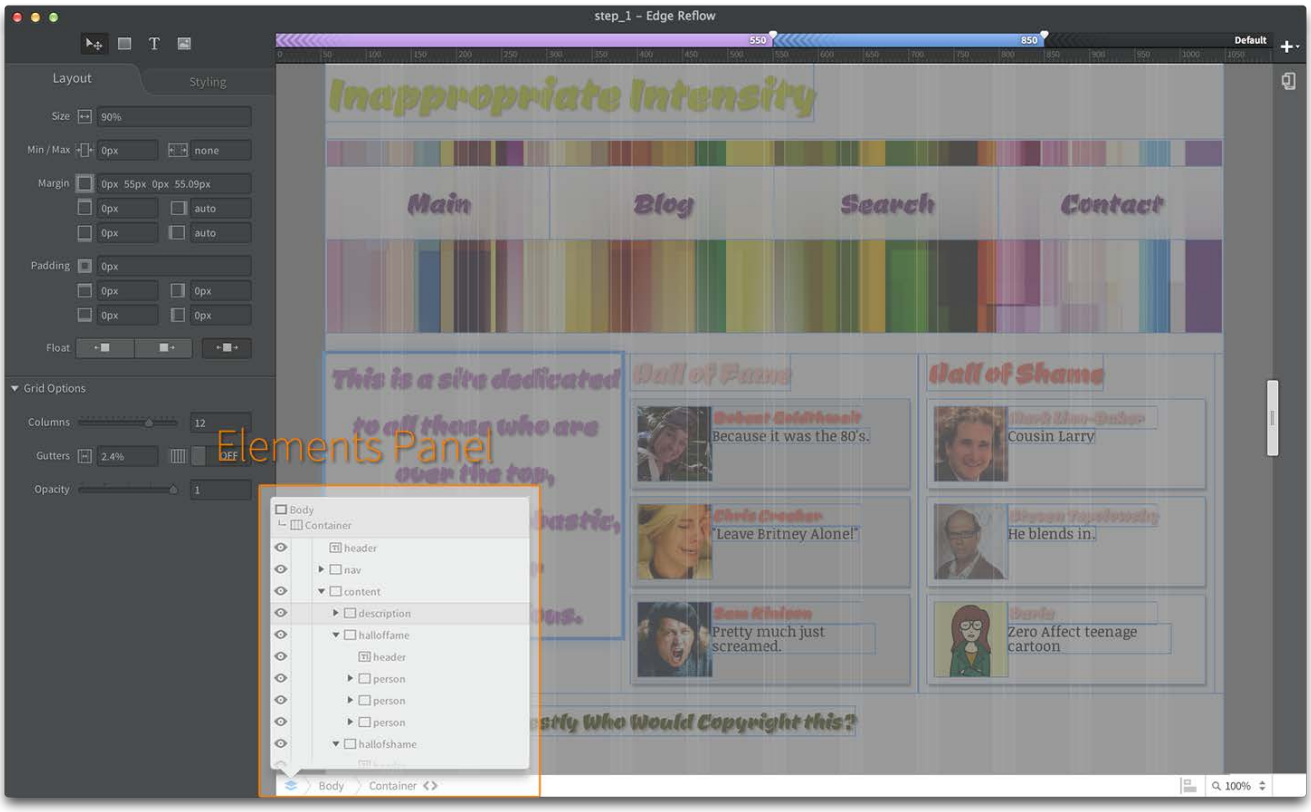
*Gives you a look at the structure you are creating, allowing you to see the hierarchical relationships you are creating in your design.*

Code Preview

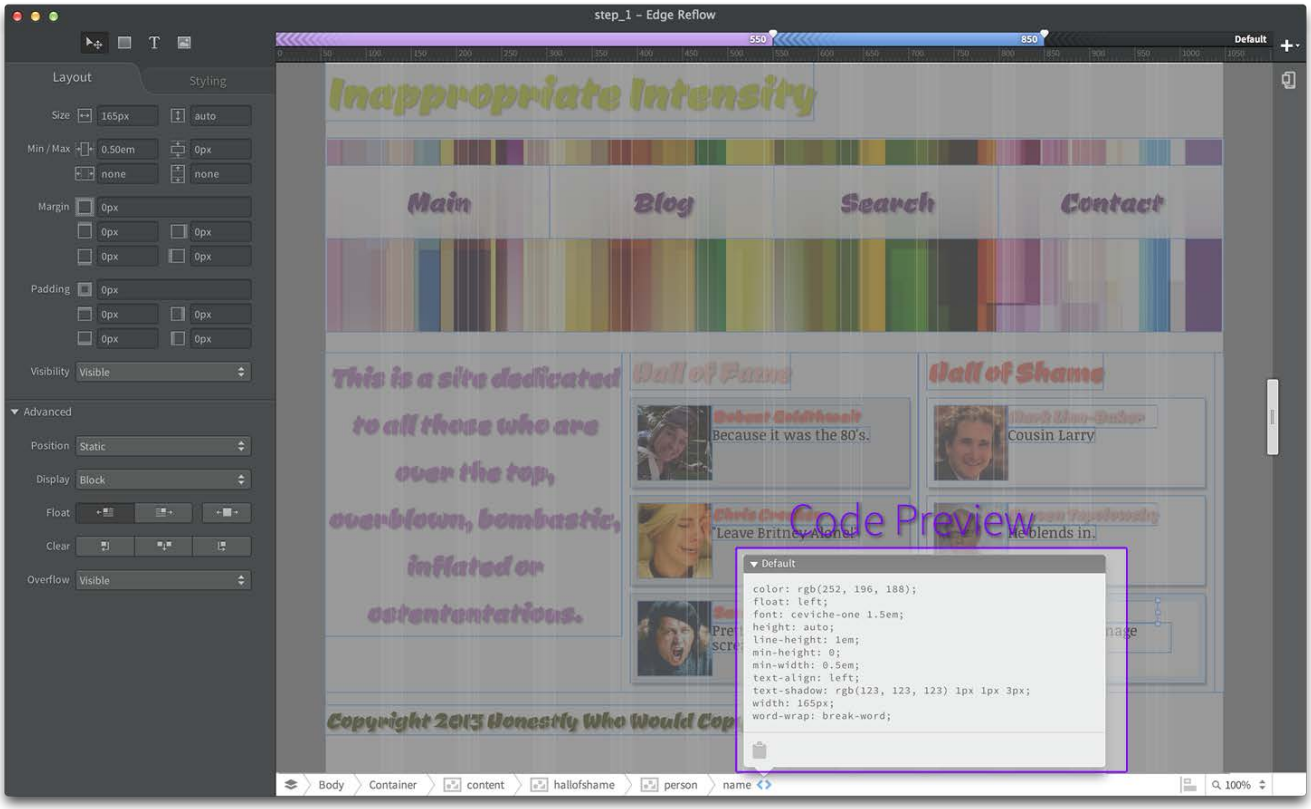
*Allows you to see the CSS code that enables each element of your design.*




*Reflow UI*




*Elements Panel Expanded*




*Code Preview Expanded*

# Design Exercise 1

- Open up Reflow Project It should be in folder: */lab/design/reflow/step_01*
- Play with the resize handler. Move it left and right. The design area screen will resize.

Over the course of this exercise, you will be given pixel targets to hit like "resize the design area to about 850px". **Do not worry about hitting any of these exactly.** Don't spend any time perfectly targeting those exact numbers. You will be frustrated and throw things at me. I would prefer you not. They don't have to be perfect.

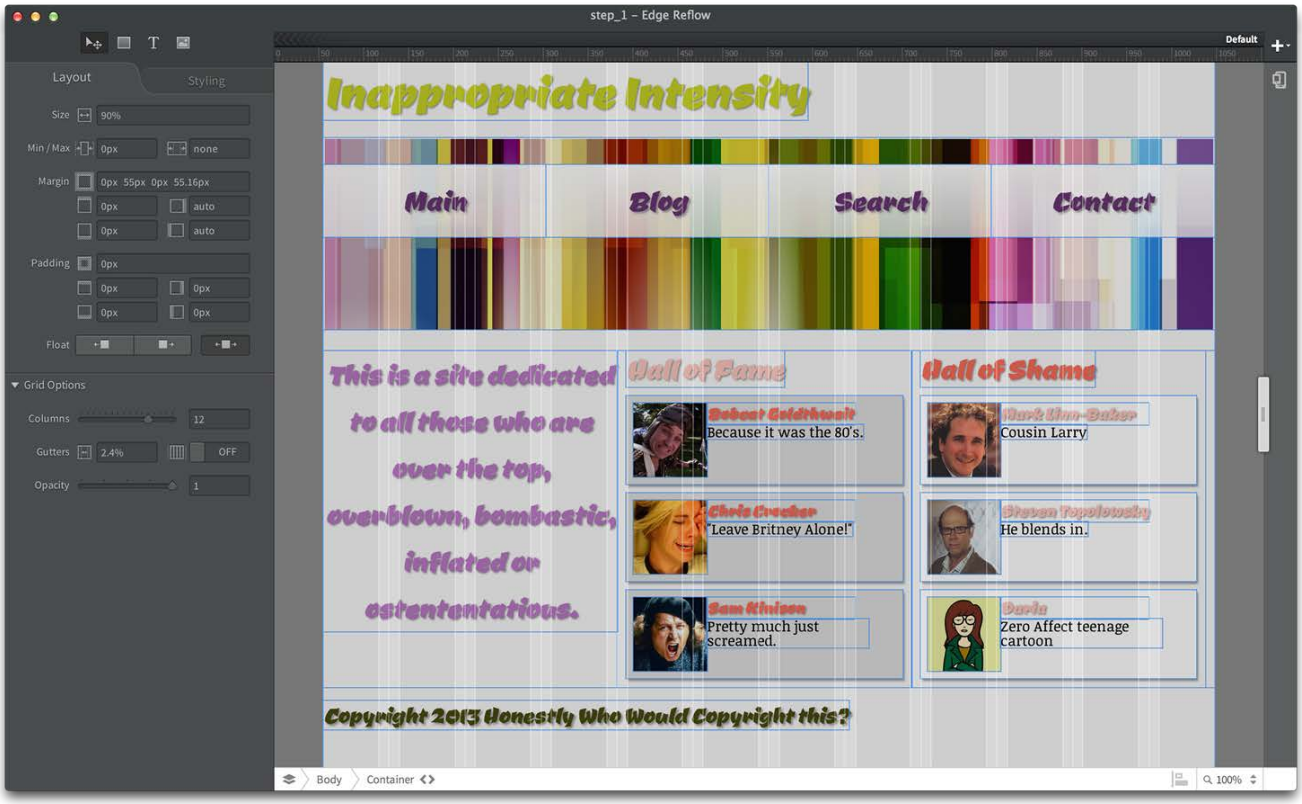



*Figure 1*

Now when your slider gets under about 800 pixels the description (left most section "This is a site…") will start to be covered by the *halloffame* section.

We're going to add a break point at 850px to fix this.

# Add a break point at 850px

- Using the grey slider resize the design area to about 850px.
- Press the plus button in the top right of the screen.
- Don't worry if it isn't exactly 850px, sometimes it gets stuck at 848px.

You should now have a purple bar at the top of the screen that reads "850".

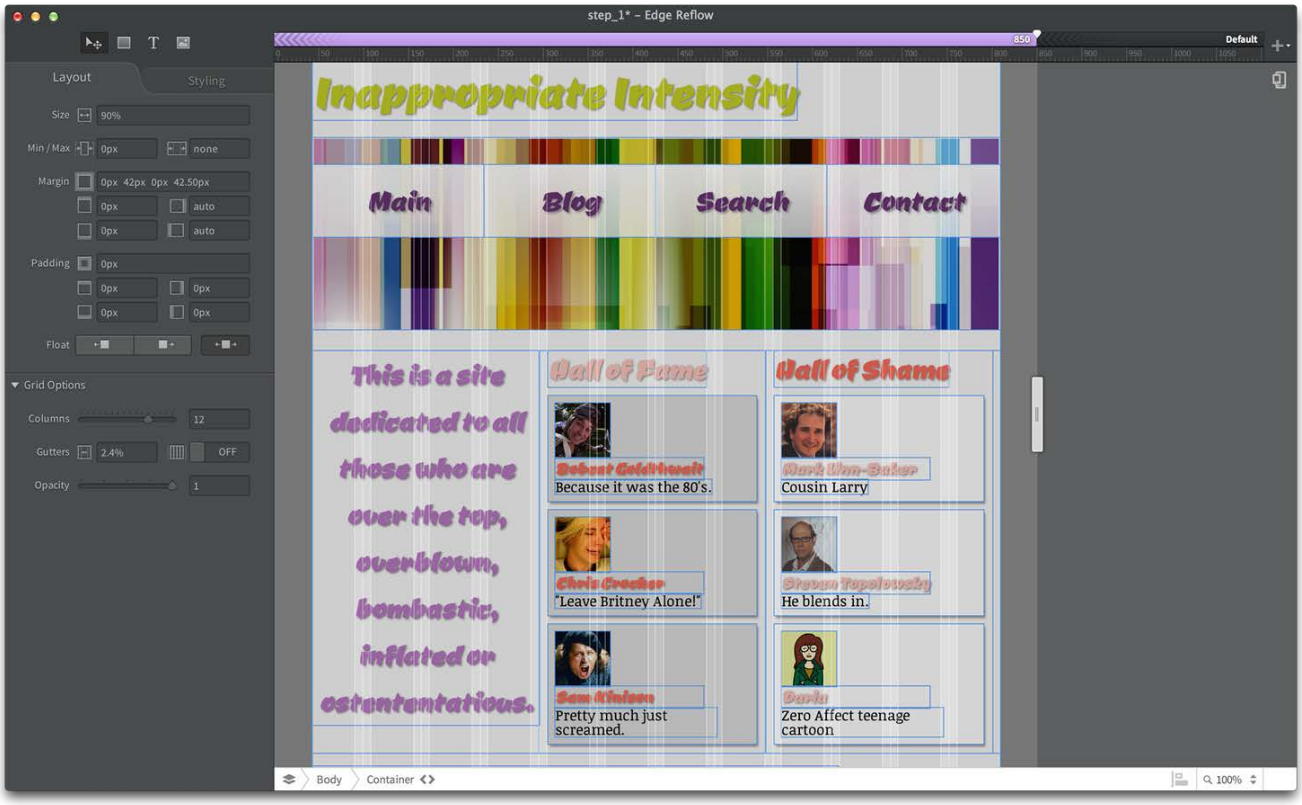



*Figure 2*

To correct the description, we want it to take up 100% of the design area width once the screen gets below 850px.

## Resize description to 100% below 850px

- Move grey slider so that design area is less then 850px wide.
- Click on description. At the bottom you should see:
  - Body > Container > content > description > Text <>
  - This means we targeted the text and not the description container
  - So double click on "description" indicated in bold: Body > Container > content > description > Text <>
  - You should notice that Layout -> Size (Width) is now 33%.

- Change the Layout -> Size (Width) to 100%

Great, but now the *halloffame* and *hallofshame* look awkward, so we want to change them to each be half and not thirds of the design area.


*Figure 3*

## Resize *halloffame* and *hallofshame* to 50% below 850px

- Design Area should still be under 850px.
- Click on *halloffame*. Layout > Size (Width) is now 33%.
- Change Layout -> Size (Width) to 50%.
- Click on the *hallofshame*. Layout > Size (Width) is now 33%.
- Change Layout -> Size (Width) to 50%.


*Figure 4*

Resize the screen again with the grey slider on the right. When your slider gets to about 500 pixels the text in the nav (Main Blog Search Contact) will start to drop to a second line. So we'll want to add another breakpoint and fix the nav appearance.

*Figure 5*

## Add a break point at 550px.

- Using the grey slider resize the design area to about 550px.
- Press the plus button in the top right of the screen.
- Don't worry if it isn't exactly 550px, sometimes it gets stuck at 547px

You should now have a purple bar at the top of the screen that reads "550" followed by a blue bar that reads "850".


*Figure 6*

## Fix nav below 550px

- Move grey slider so that design area is less then 550px wide.
- Click on main. At the bottom you should see:
    - Body > Container > nav > ul > main > Text <>
    - This means we targeted the text and not the main container
    - So double click on "main" indicated in bold: Body > Container > nav > ul > main >

Text <>
    - You should notice that Layout -> Size (Width) is now 25%.
- Change the Layout -> Size (Width) to 100%
- Repeat for blog

After doing blog you should notice that nav is running into the description. We can fix that by making sure that nav resizes.


*Figure 7*

- Click on the multi color vertical striped container. At the bottom you should see:
    - Body > Container > nav <>
    - Layout > Size (Height) should read "213px".
- Change Layout > Size (Height) to "auto".
- Repeat resize to 100% for search and contact.

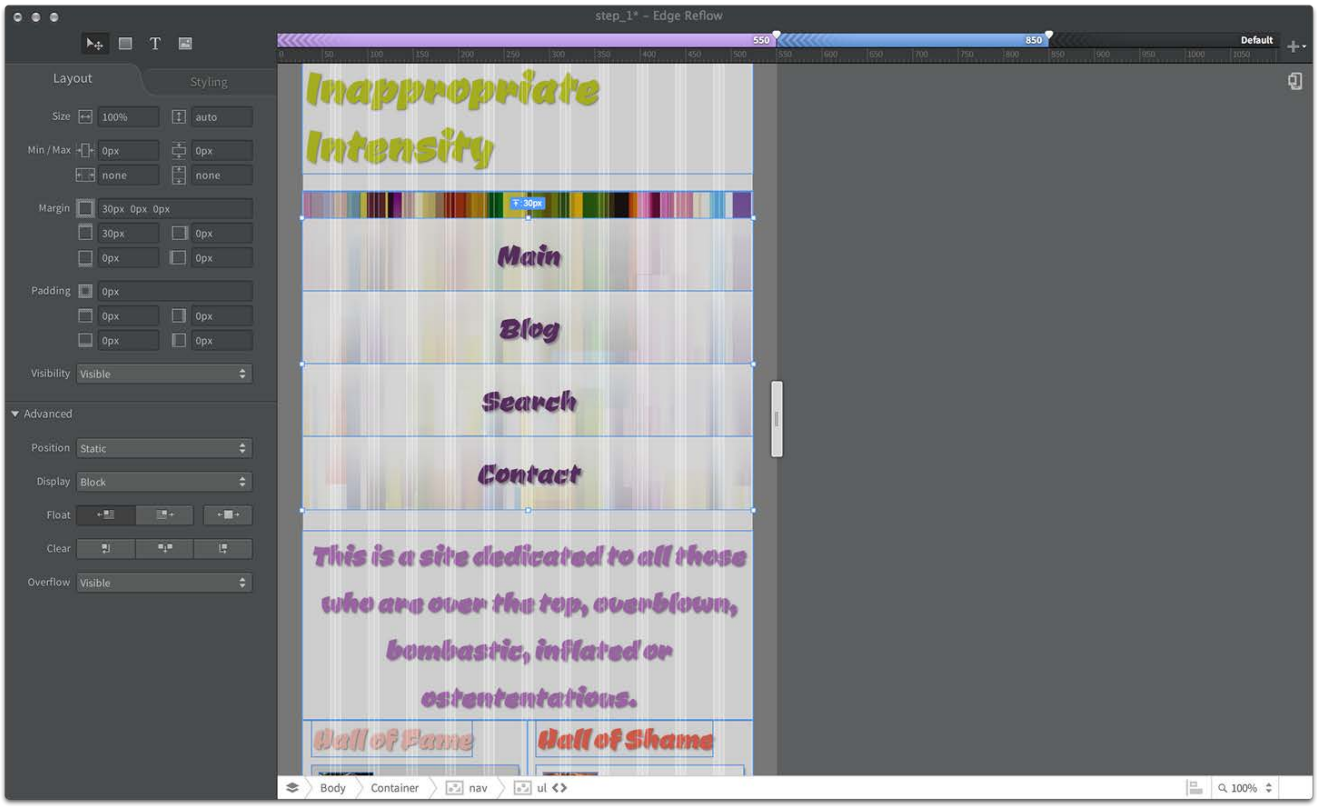



*Figure 7*

At this width, it is most likely that someone is viewing on a phone and not a tablet. Most times having multiple columns is too much for a phone. We want *halloffame* and *hallofshame* to be a single column for each.

*Figure 9*

## Set *halloffame* and *hallofshame* to be single columns.

- Design Area should still be under 550px.
- Mouse scroll down on design area to bring the 2 columns into view
- Click on *halloffame*. Layout > Size (Width) should be 50%.
- Set Layout > Size (Width) to 100%
- Mouse scroll down on design area to bring the *hallofshame* into view
- Click on *hallofshame*. Layout > Size (Width) should be 50%.
- Set Layout > Size (Width) to 100%

When you do this, you will notice that the description for a few of the members of the hall of fame now floats up next to the name. We'll fix this, then we'll be done.


*Figure 10*

## Fix person description layout

- Click on "Leave Britney Alone."

- Layout > Min Width should be 0.50em
- Change Layout > Min Width to 150px.

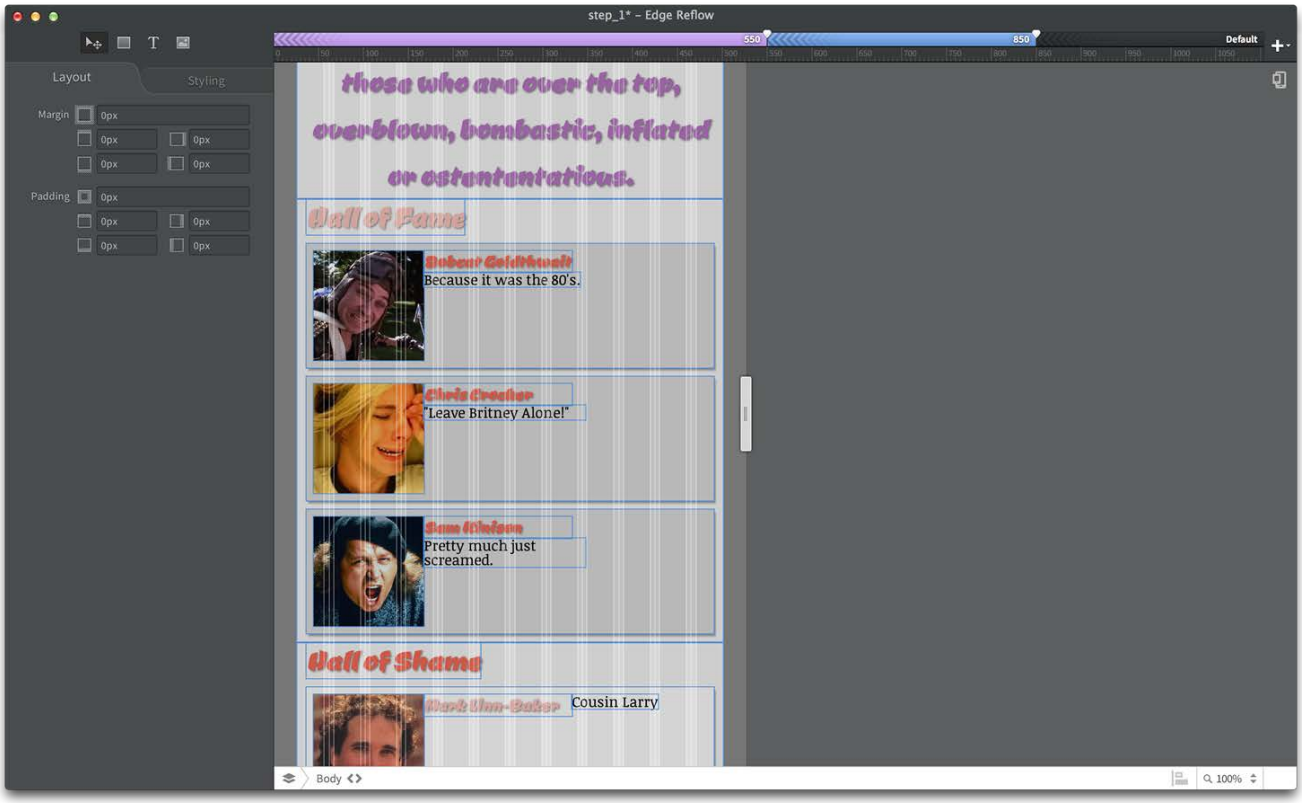



*Figure 11*

All fixed.

# Code Exercise 1 - Flexible Grid

Over the course of this exercise we will be taking the code in the working directly from how it appears in step_0 to how it appears in step_1

- Start with code in */lab/code/working*
- Open *index.html* in Adobe Edge Code (or code editor of your choice)
- Open *index.html* in Chrome.

All of the color, background, and font choices have been brought over from Reflow. None of the layout has been done. We're going to code that by hand.

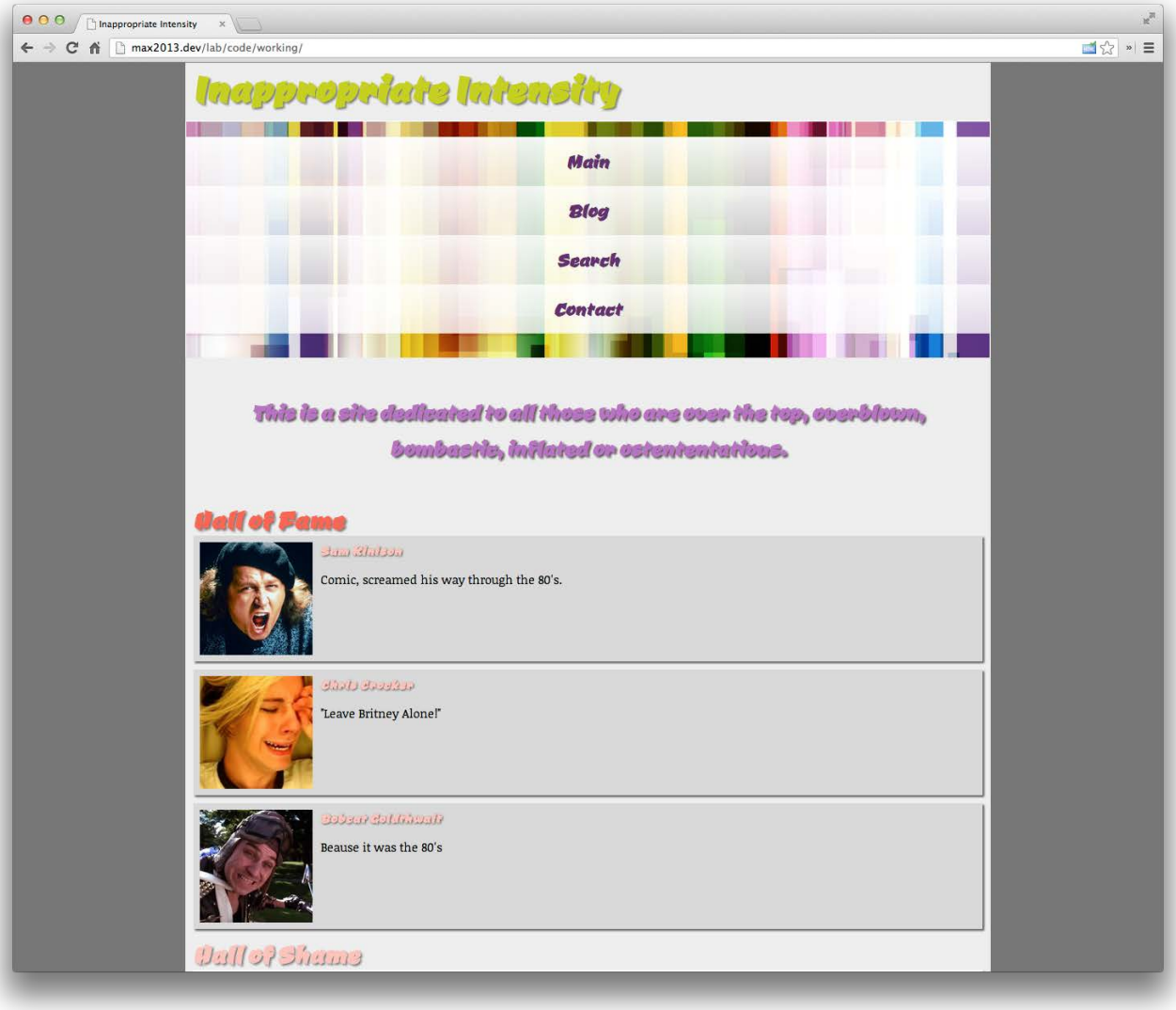



*Figure 1*

# Add a layout.css file to the project

- In the folder lab/code/working/css add a file named layout.css
- In index.html add the following line of code to the HTML header:

```
<link rel="stylesheet" href="css/layout.css"/>
```

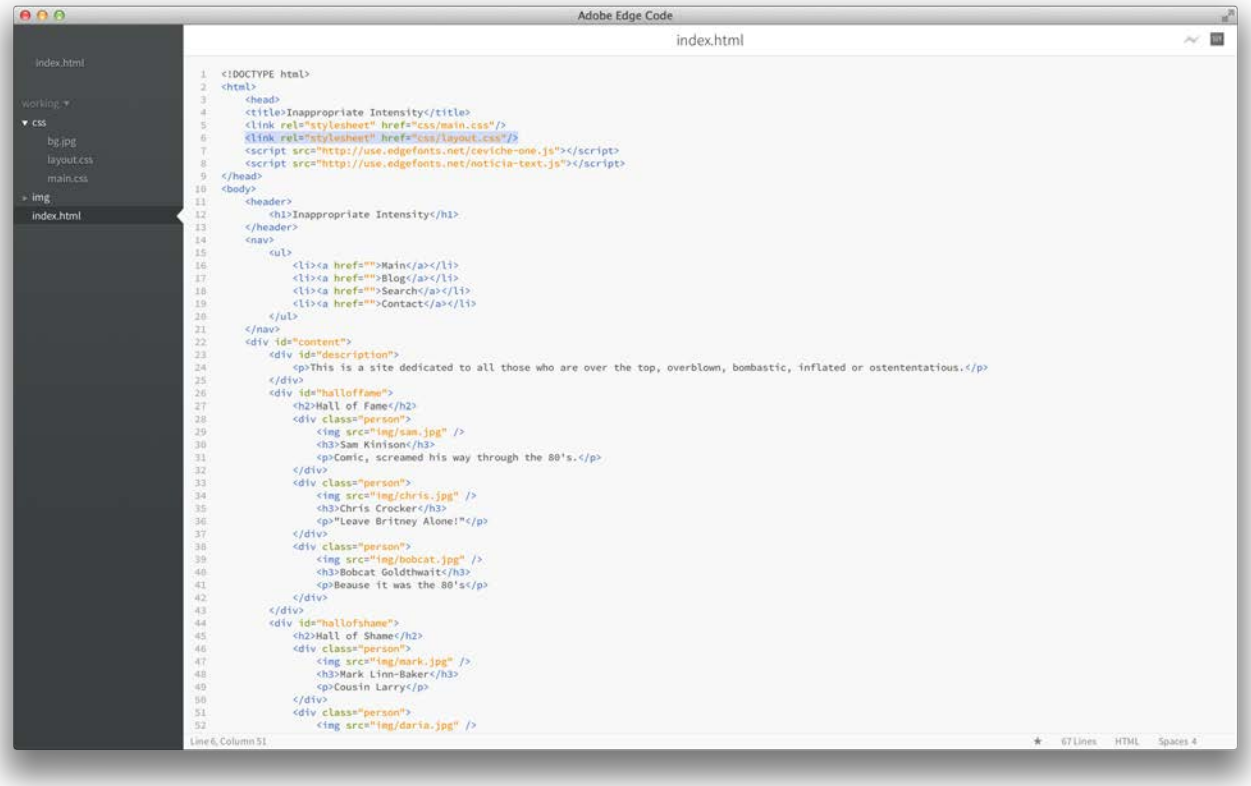



*Figure 2*

Now we're going to add some CSS rules to make things layout as a grid.

# Layout content as a grid

Add any rules below to layout.css.

```
*, *:after, *:before {
  -webkit-box-sizing: border-box;
  -moz-box-sizing: border-box;
  box-sizing: border-box;
}
```

*What is the deal with box-sizing: border-box?*

Basically we're changing the way the width of an item is calculated by the browser. Normally the width excludes padding. So if I make a box, set width: 300px then set padding: 20px in real terms my box will be 340px wide.

```
20px of left padding + 300px of width + 20px of right padding = 340px.
```

Setting border box changes this. It forces the browser to include the padding inside the 300px width. So it changes the equation to this:

```
300px =  20px of left padding + Xpx of width + 20px of right padding
  Solve for X = 260.
```

*What does this have to do with responsive designs?* We normally want to set widths to a percentage so as we resize the browser column sizes resize. However that leads untenable paddings. Like setting all padding to 1% and then taking that 1% out of our column widths so they never add up to above 100%. It starts to get difficult to track, and quite frankly you can make the argument that padding like that should be in pixels and not percents.

Well now you can do that, with Border Box.

More on Border Box

- http://paulirish.com/2012/box-sizing-border-box-ftw/
- http://css-tricks.com/box-sizing/

That's all great but it doesn't change the appearance of the page.

Next we're going to make the 3 sections in content each take up a third of the screen.

```
#description{
     width: 33%;
}

#halloffame{
     width: 33%;
}
```

```
#hallofshame{
      width: 33%;
}
```

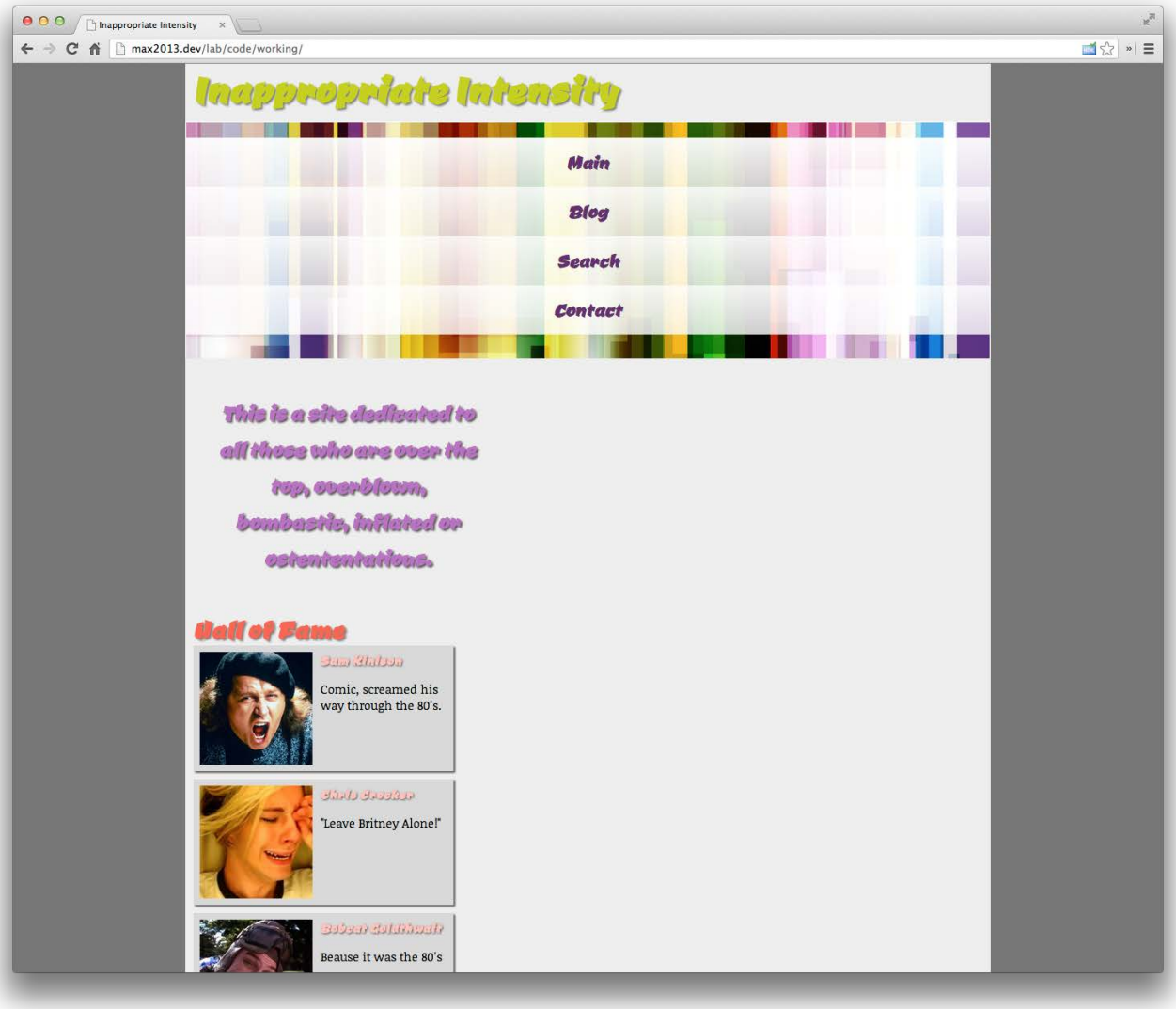



*Figure 3*

Now we're going to make them slide next to each other using float:left;

```
#description{
      width: 33%;
      float:left;
}

#halloffame{
      width: 33%;
      float:left;
}
```

```
#hallofshame{
     width: 33%;
     float:left;
}
```

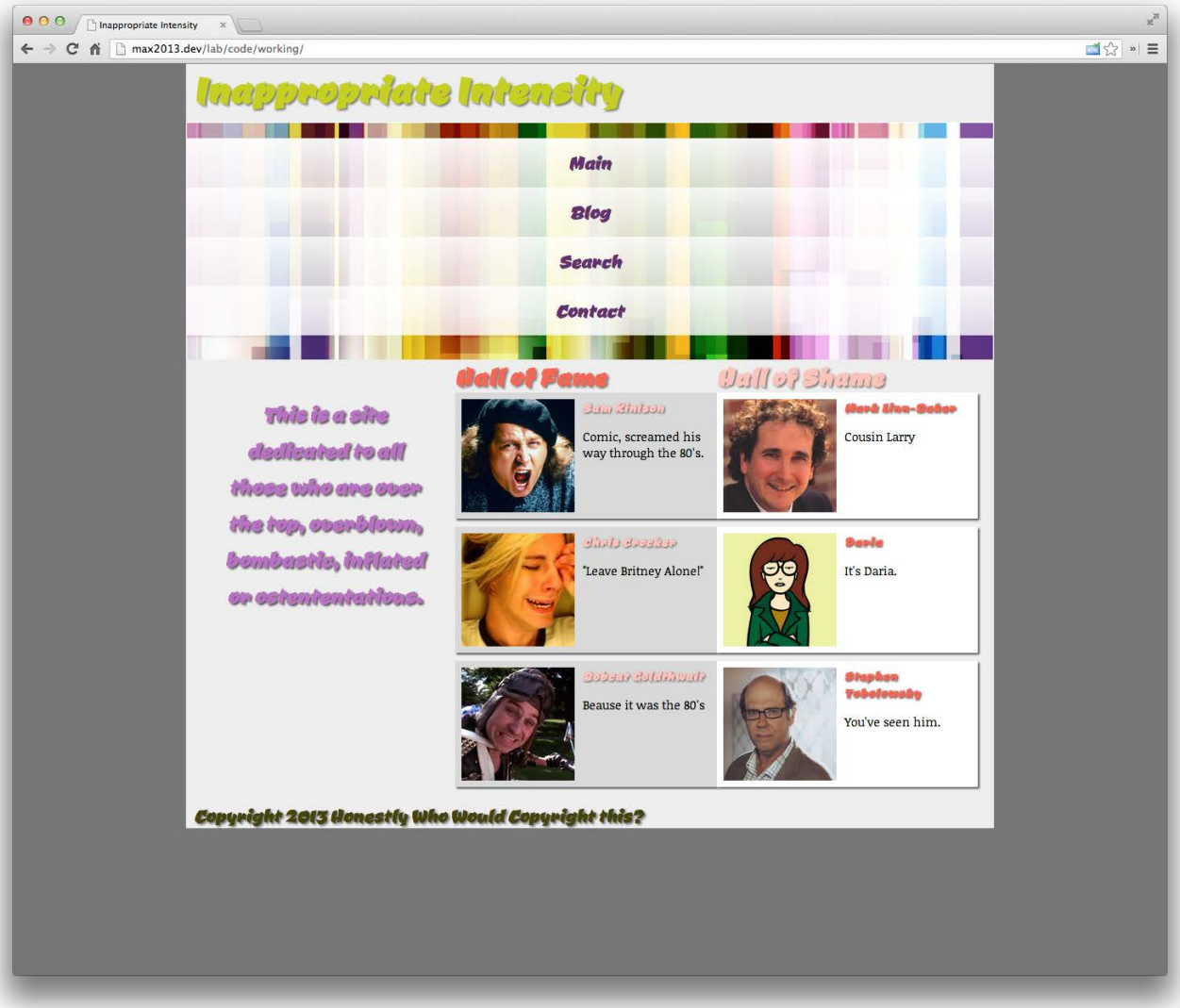



*Figure 4*

Looking good, but we need a little more space between the "halls." We'll do it with padding. Because we are using the border-box model we can add padding without fiddling with column widths.

```
#description{
    width: 33%;
    float:left;
}

#halloffame{
    width: 33%;
```

```
    float:left;
    padding: 0 5px;
}

#hallofshame{
    width: 33%;
    float:left;
    padding: 0 5px;
}
```

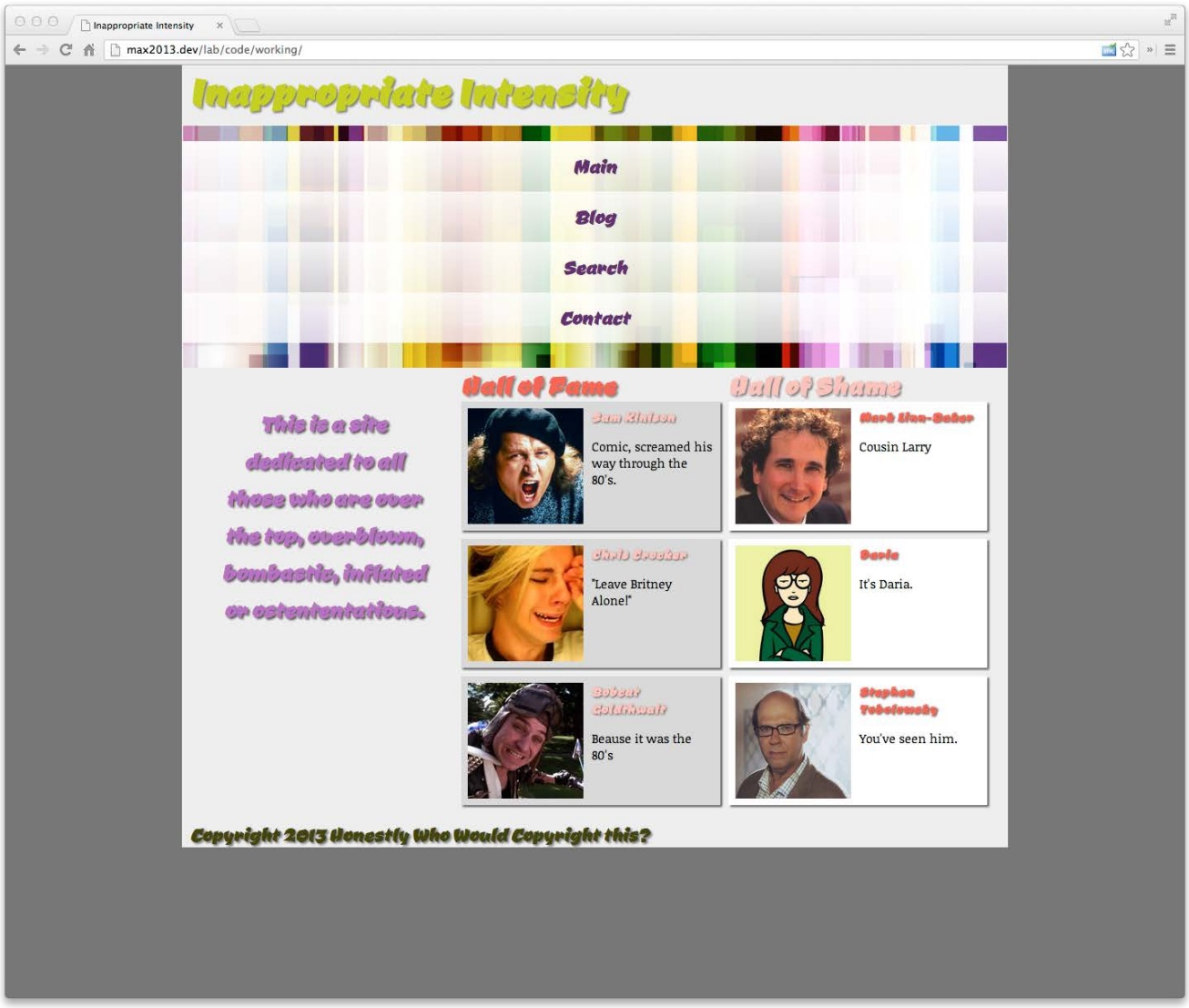



*Figure 5*

Much better. Now we'll take on the navigation. Same idea as with the content columns, but instead of thirds, we'll be using quarters

```
nav ul li a{
     width: 25%;
     float: left;
}
```


*Figure 6*

Looks okay, but because we're floating those navigation links they no longer expand the box they are sitting in. To fix that we'll use a clearfix

When you float an element it no longer has any impact on the height of it's parent. If you float all of the children of element it's height becomes 0. To fix this you need to basically add a fake element past the floated children and clear it completely. This will force the height of the containing element to take up all of the height of the children. A clearfix does this:

```
*:after{
      content: "";
     display: block;
     clear: both;
```

```
}
```

after
: *adds it via css, so no HTML hackery needed.*

content
: *adds a blank piece of content*

display
: *table reduces impact on margins*

clear
: *clears out the float*

More on Clearfix

- http://css-tricks.com/snippets/css/clear-fix/
- http://nicolasgallagher.com/micro-clearfix-hack/

```
nav ul:after{
  content: "";
  display: block;
  clear: both;
}
```

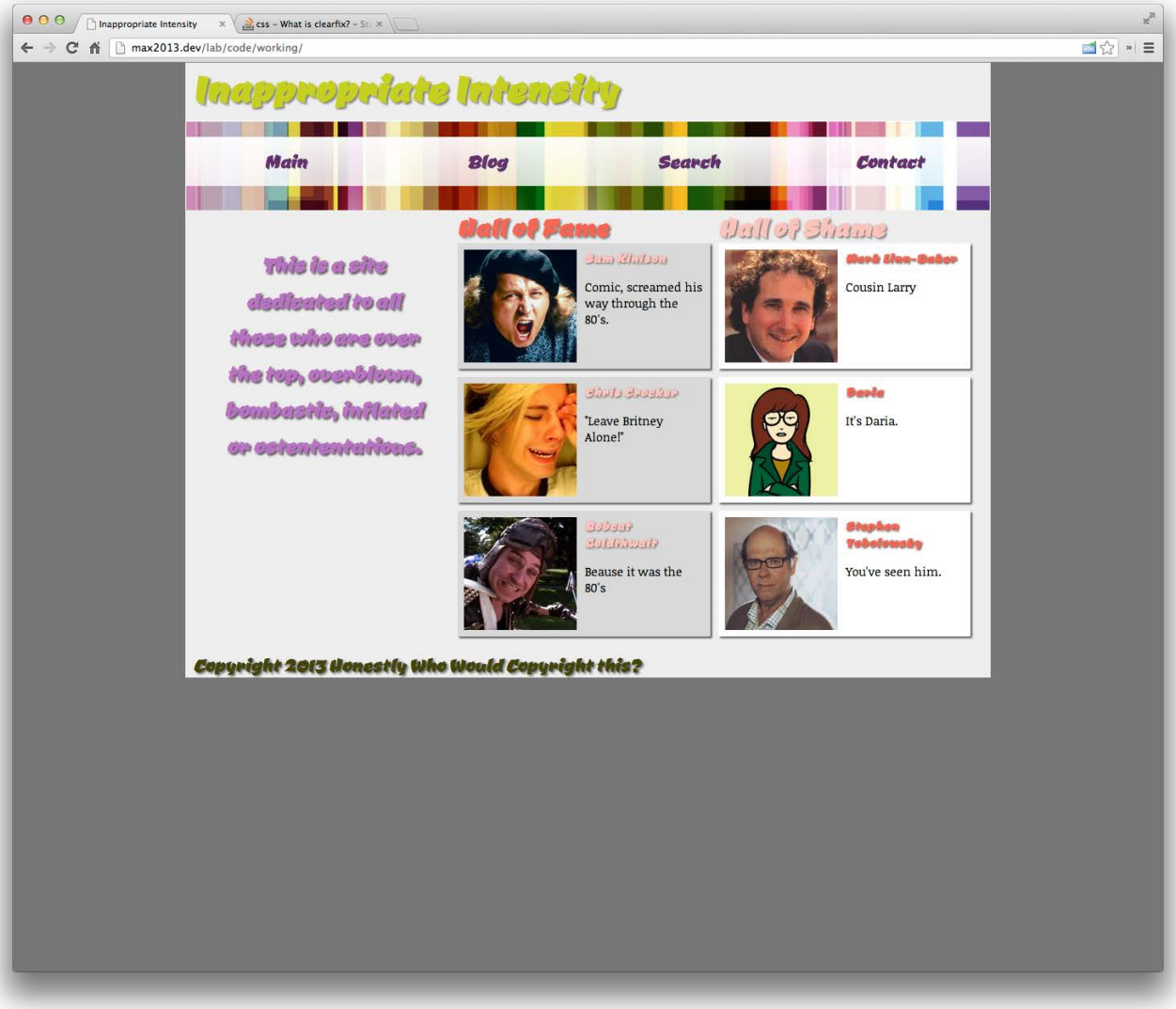



*Figure 7*

There, almost perfect. The last thing we need to do (and this isn't necessarily a responsive thing,) is make the light colored body extend all the way to the end of the page.

```
html { height: 100%; }
body { min-height: 100%; }
```

Once we do that, we should have a fluid grid that flexes and extends the full height of the viewport.

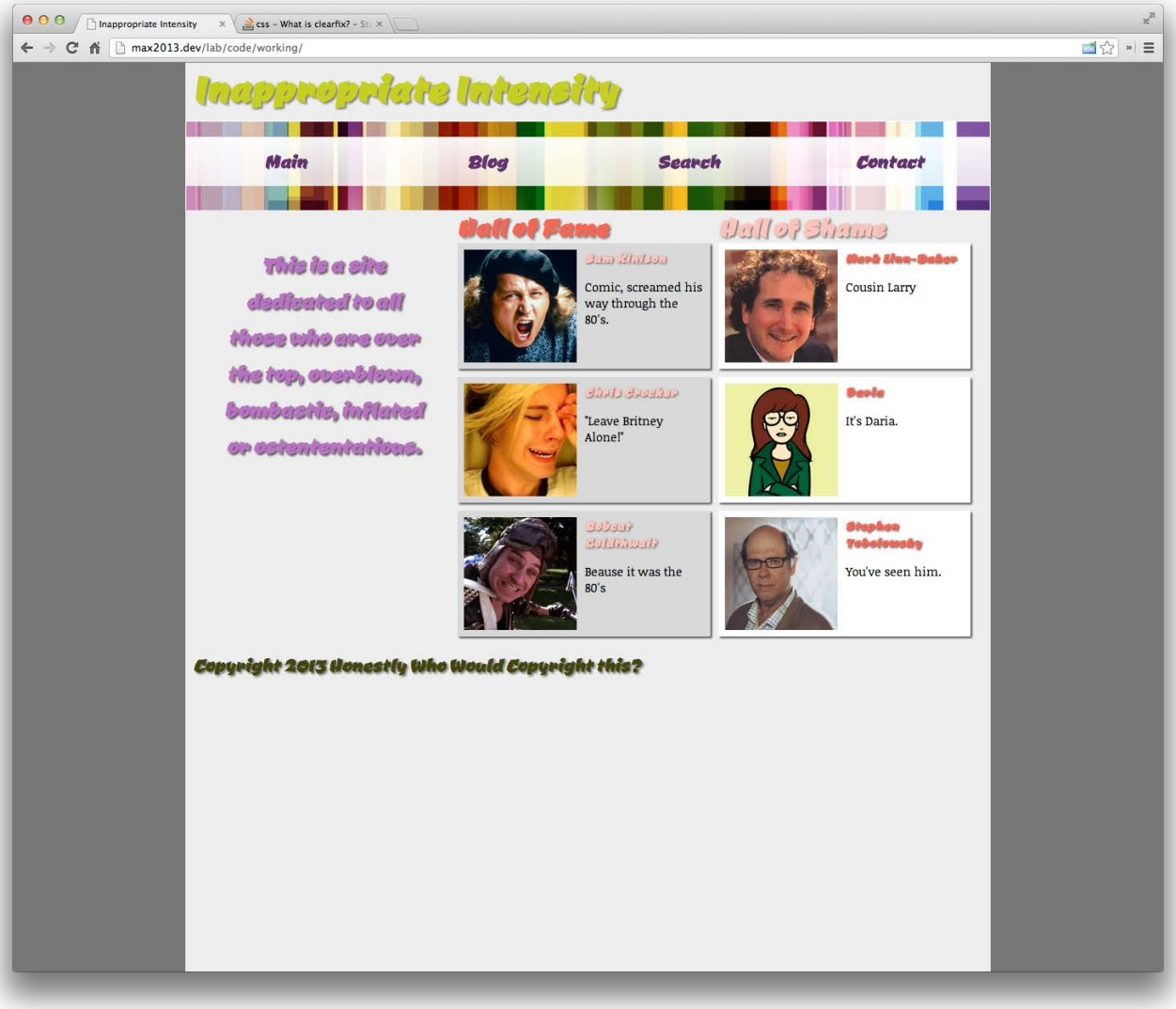



*Figure 8*

If you resize the browser you should see an issue crop up when browsers is about 800px or so. The footer floats up to the hall of shame. It looks terrible. We'll have to fix that with another clearfix.

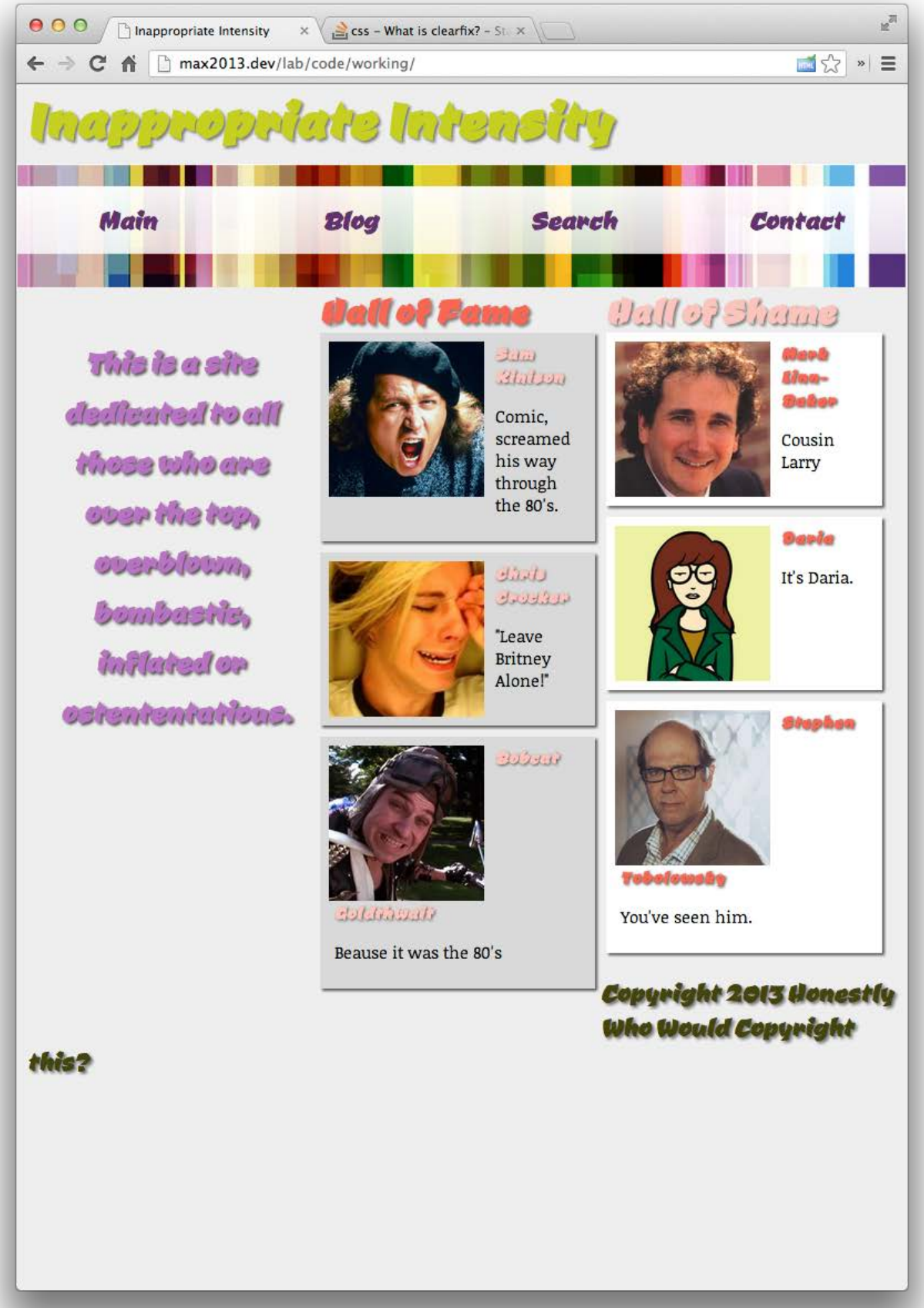

Inappropriate Intensity
css – What is clearfix? – St
max2013.dev/lab/code/working/
Inappropriate Intensity
Main
Blog
Search
Contact
This is a site dedicated to all those who are over the top, overblown, bombastic, inflated or ostententations.
Hall of Fame
Sam Kinison
Comic, screamed his way through the 80's.
Chris Crocker
"Leave Britney Alone!"
Bobcat
Goldthwait
Beause it was the 80's
Hall of Shame
Mark Linn-Baker
Cousin Larry
Daria
It's Daria.
Stephen
Tobolowsky
You've seen him.
Copyright 2013 Honestly Who Would Copyright this?

*Figure 9*

```
#content:after{
  content: "";
  display: block;
  clear: both;
}
```

Now fixed.

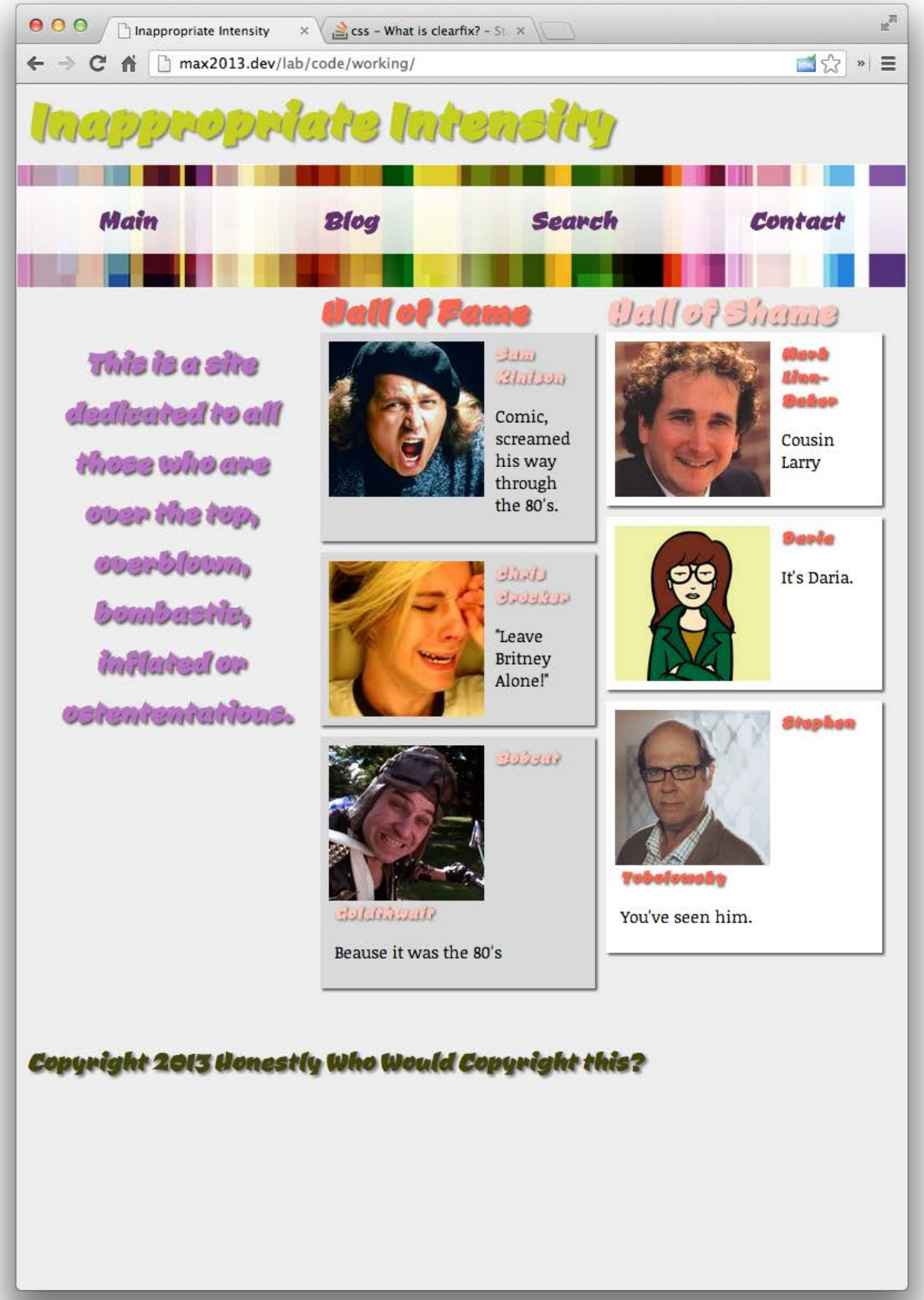

Inappropriate Intensity
css – What is clearfix? – St
max2013.dev/lab/code/working/
Inappropriate Intensity
Main
Blog
Search
Contact
This is a site dedicated to all those who are over the top, overblown, bombastic, inflated or ostententations.
Hall of Fame
Sam Kinison
Comic, screamed his way through the 80's.
Chris Crocker
"Leave Britney Alone!"
Bobcat
Goldthwait
Beause it was the 80's
Hall of Shame
Mark Linn-Baker
Cousin Larry
Daria
It's Daria.
Stephen
Tobolowsky
You've seen him.
Copyright 2013 Honestly Who Would Copyright this?

*Figure 10*

All the CSS for this Exercise:

```
*, *:after, *:before {
  -webkit-box-sizing: border-box;
  -moz-box-sizing: border-box;
  box-sizing: border-box;
}

#description{
    width: 33%;
    float:left;
}

#halloffame{
    width: 33%;
    float:left;
    padding: 0 5px;
}

#hallofshame{
    width: 33%;
    float:left;
    padding: 0 5px;
}

nav ul li a{
     width: 25%;
     float: left;
}

nav ul:after{
  content: "";
  display: block;
  clear: both;
}

html { height: 100%; }
body { min-height: 100%; }

#content:after{
  content: "";
  display: block;
  clear: both;
}
```

*Wait a minute, shouldn't I be using a framework for my grids?*

Well here in code land we like to answer every yes or no question with "It depends."

Grid frameworks like responsive.js, and other UI frameworks that contain a grid system like Bootstrap will make it easier for you to deliver write your code a grid… if you write your code the way they want you to. And that's fine — they are great productivity boosters. But if I used them in this session then I would be teaching you Bootstrap or responsive.js and not the underlying theory. Once you get the underlying theory, feel free to use a framework.

Responsive Grid Systems

- http://responsive.gs/
- http://twitter.github.com/bootstrap
- http://www.getskeleton.com/

# Code Exercise 2 - Flexible Media

Over the course of this exercise we will be taking the code in the working directly from how it appears in step_1 to how it appears in step_2

- Start with code in */lab/code/working*
- Open *index.html* in Adobe Edge Code (or code editor of your choice)
- Open *index.html* in Chrome.

# Add a images.css file to the project

- In the folder lab/code/working/css add a file named images.css
- In index.html add the following line of code to the HTML header:

```
<link rel="stylesheet" href="css/images.css"/>
```

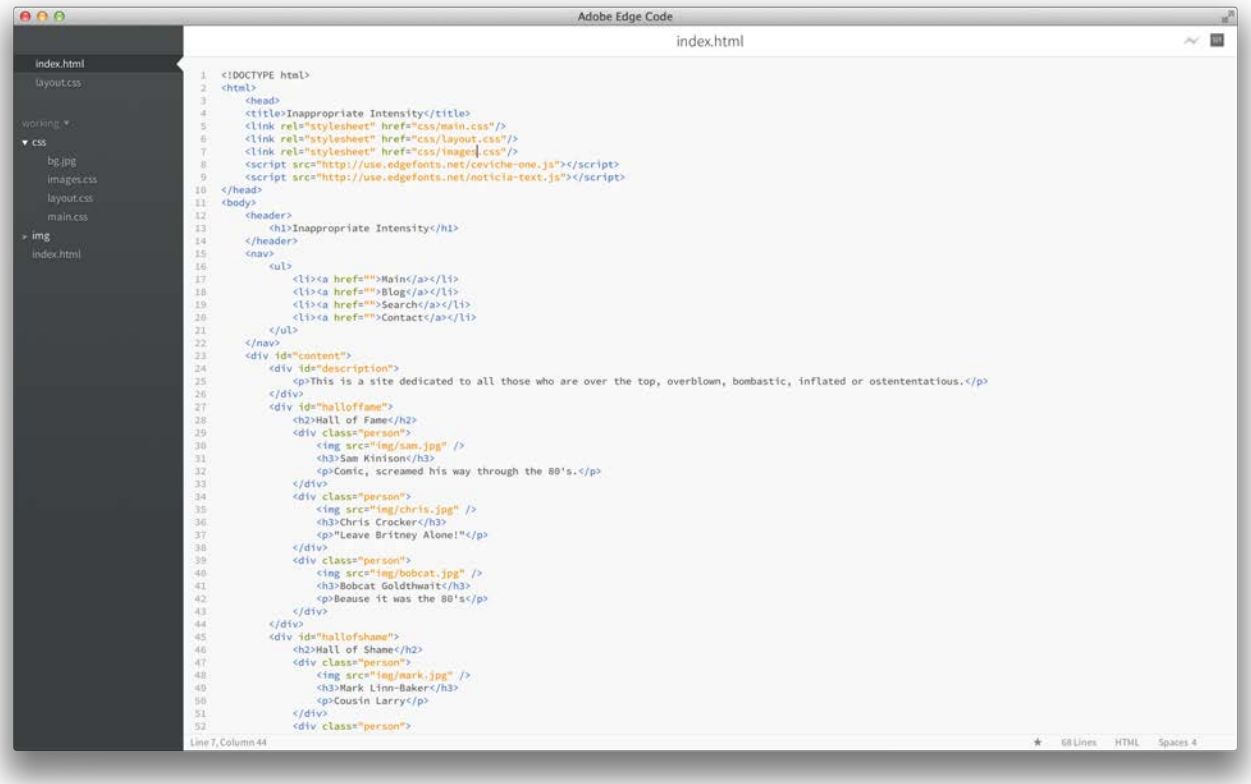



*Figure 1*

Now open images.css.

- In the browser right click on any of the pictures.
- Choose "Open in image in new tab"

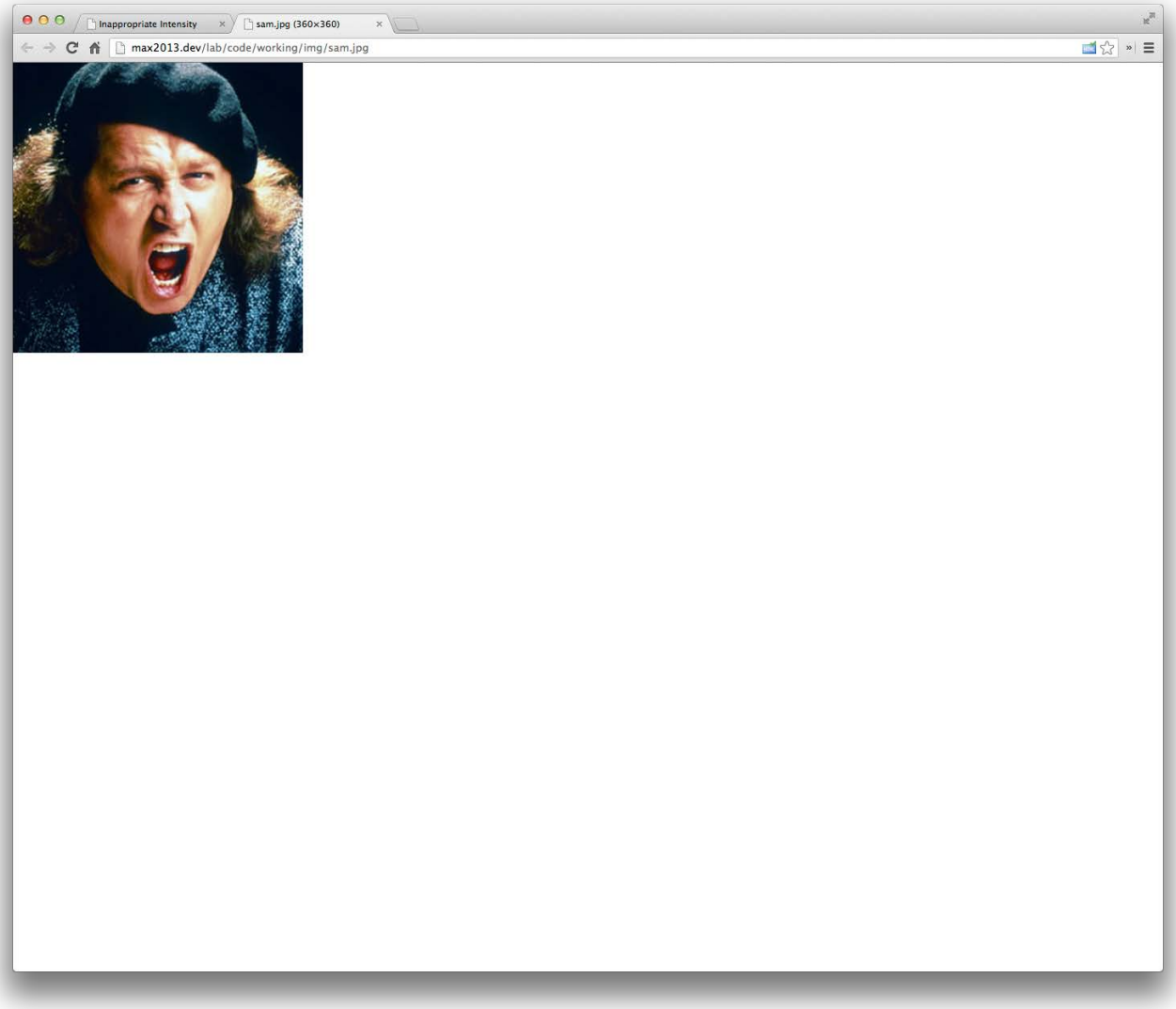



*Figure 2*

You'll notice that the image is shown to be 360px by 360px. However that's not how the image shows up on the page. That's because in the CSS for the main design I set the max-width of the image to 140px. That was just so I could work with the grid in the beginning without tackling images. We'll want to undo that setting.

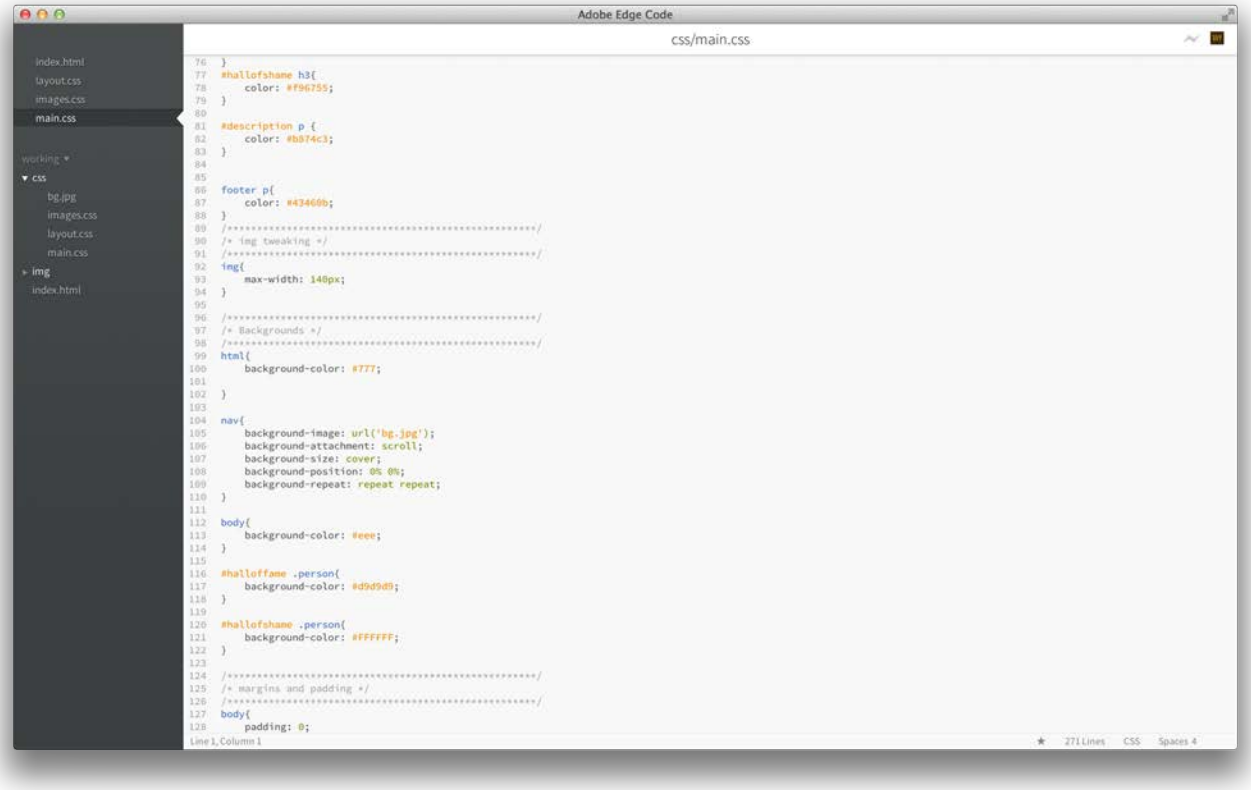



*Figure 3*

Add any rules below to images.css.

```
img{
    max-width:none;
}
```

This overrides the previous directive and results in a messed up page.

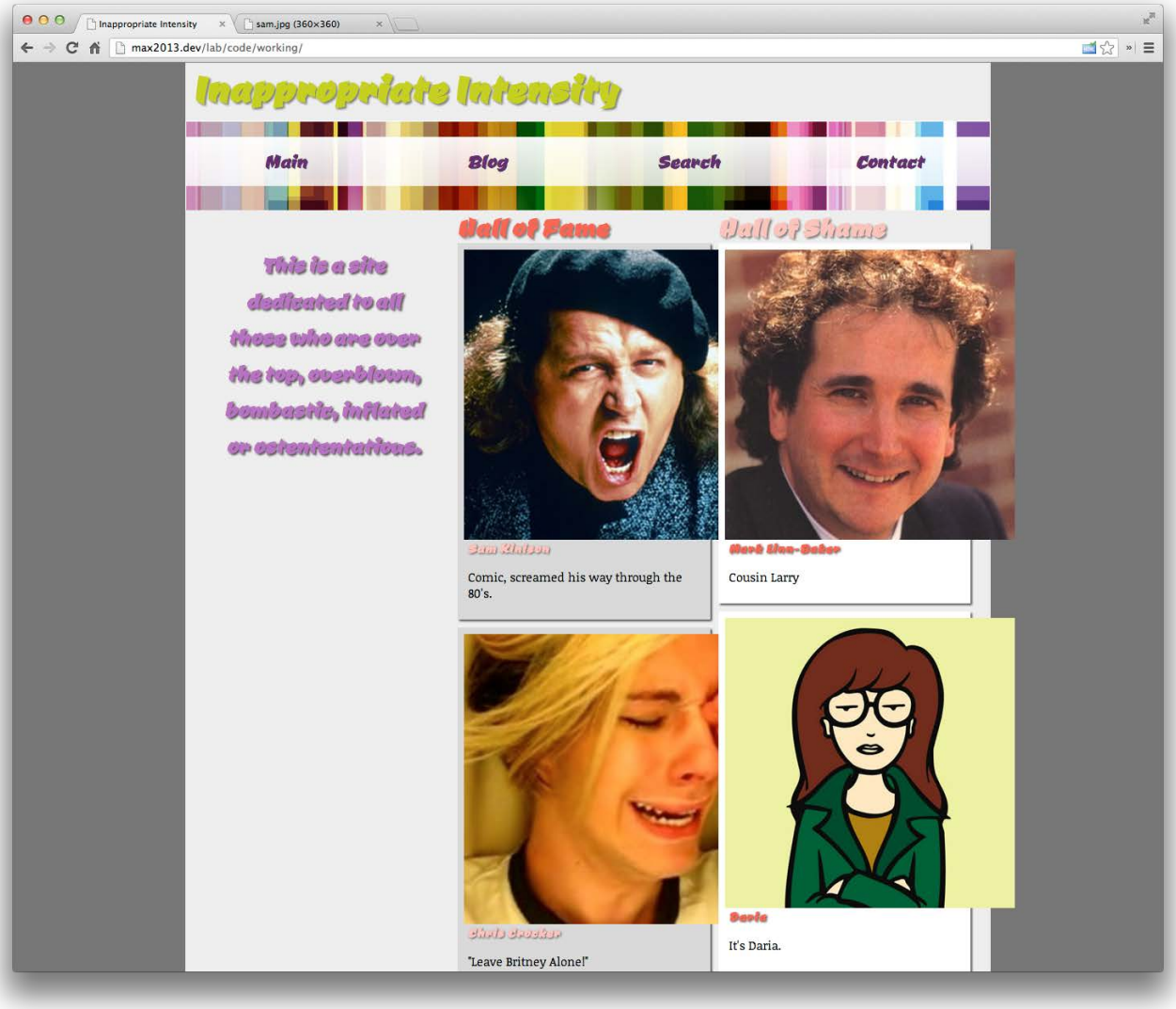



*Figure 4*

So we want to make sure that an image can never be larger then it's container. This is done with an pretty easy line of CSS.

```
img{
    max-width:100%;
}
```

Now the images behave properly.

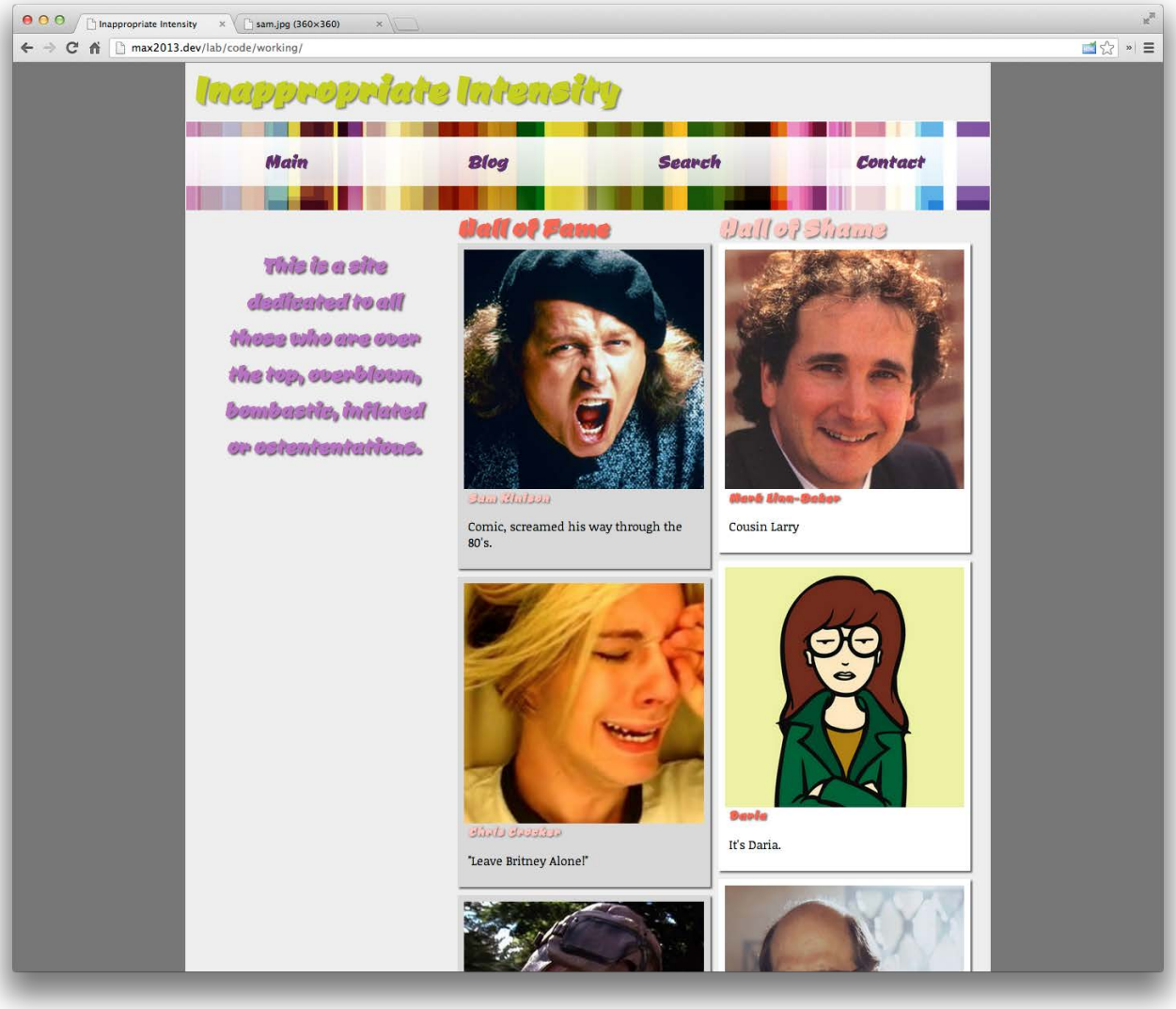



*Figure 5*

It looks better, but if you were viewing on a High Density pixel screen or "Retina" screen, the image would be appear to be lower quality. This is because these screens have 2x as many pixels at most other screens and do some trickery to make images look a lot better, but when forced to render things at scale, they can look a little artifacted.

There is an easy fix to this, you can use a media query to force images on Retina screens to be half sized, so they will look correct.

Media queries are a way of selectively applying CSS. Media queries basically ask a question "Is the answer to this question, true?" You can test a number of features of a browser including the height and width, the orientation, and in the case in this exercise, the pixel density. Typically in responsive web design you are responding to the width of

the screen, but in this case we can respond to something more specific.

More on Media Queries

- http://en.wikipedia.org/wiki/Media_queries
- http://www.adobe.com/devnet/dreamweaver/articles/introducing-media-queries.edu.html
- http://css-tricks.com/snippets/css/retina-display-media-query/

```
@media (-webkit-min-device-pixel-ratio: 2){
    .person img{
          max-width: 180px;
     }
}
```

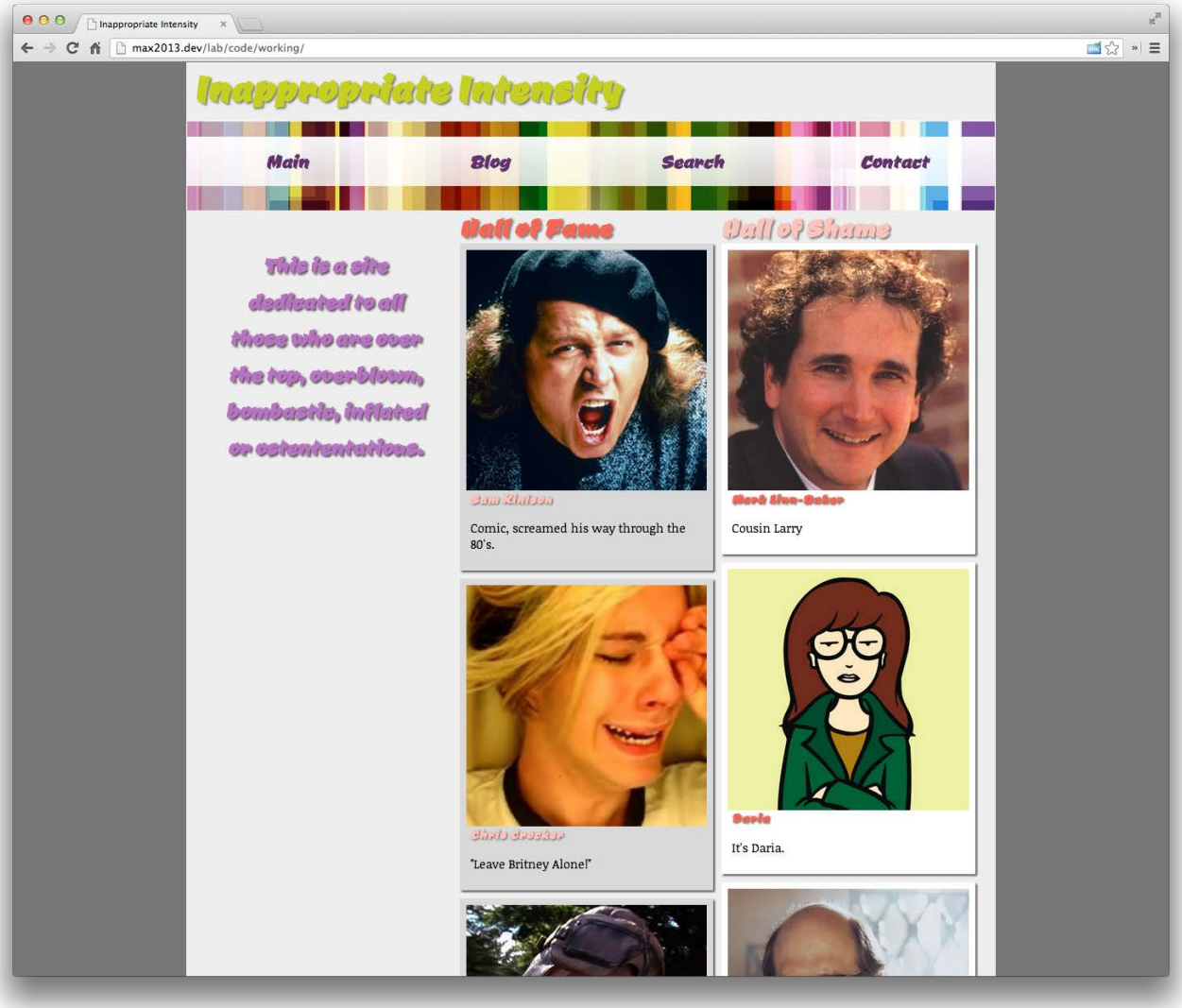



*Figure 6 - Normal*

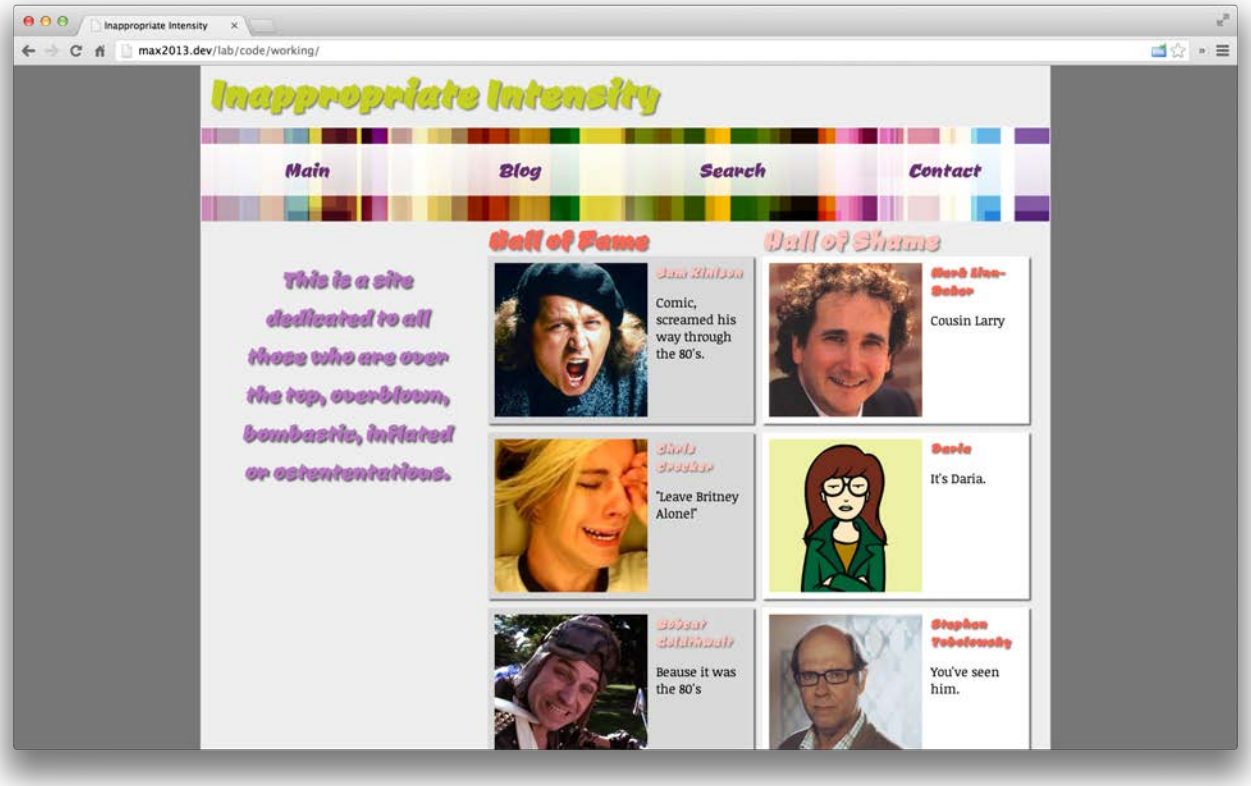



*Figure 7 - Retina*

Finally, we want those images to only take up half of the box they are on all screens, which would agree with the original design.

```
.person img{
    width:50%;
}
```

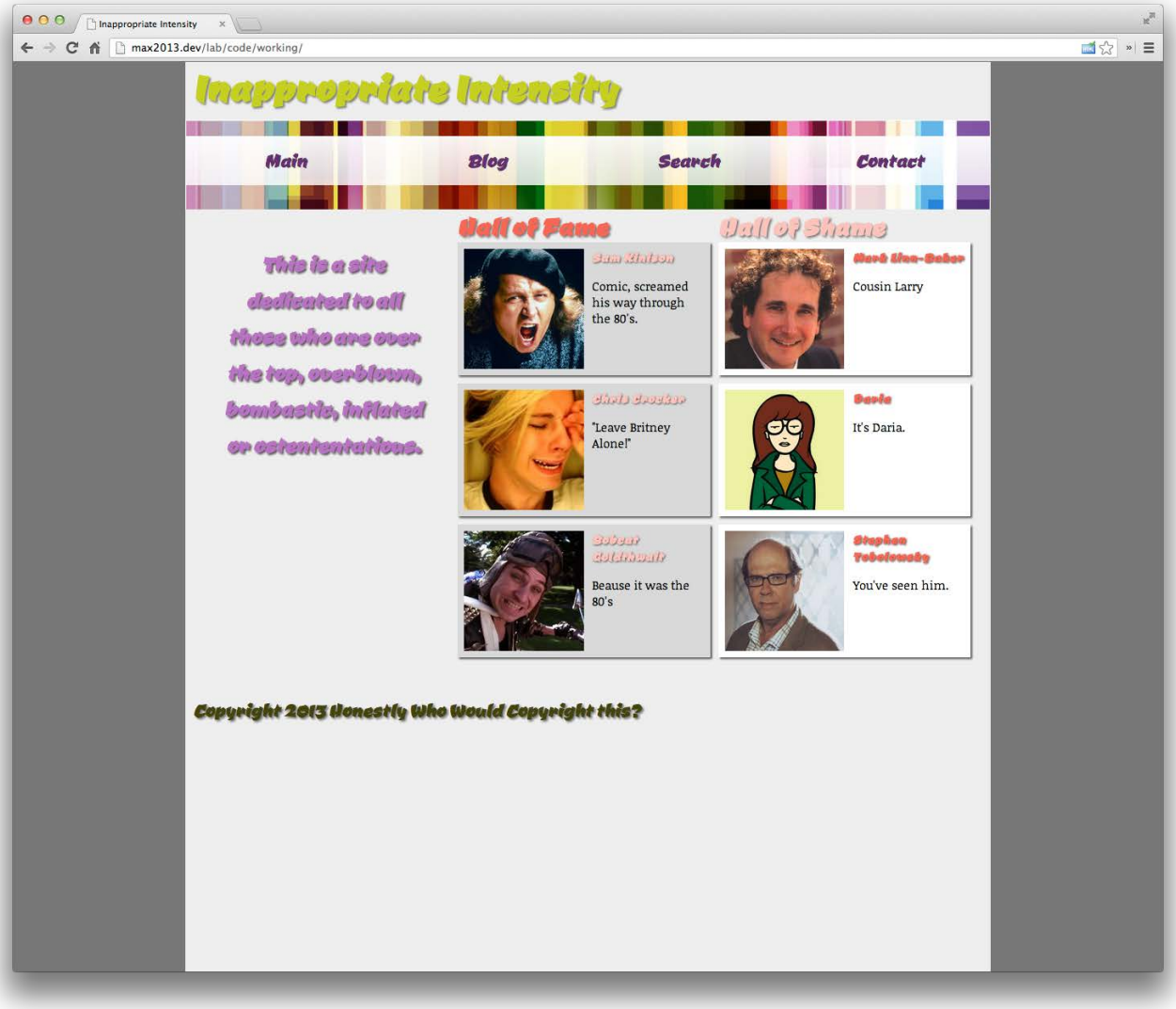
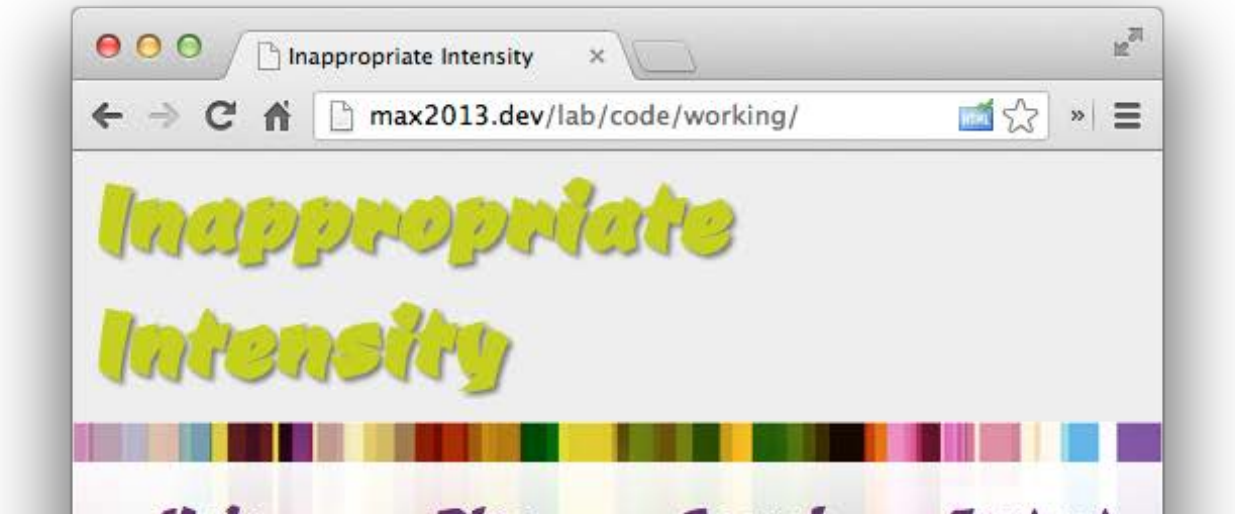



*Figure 8*

And there, we have a flexible grid, with images that scale properly when the browser shrinks, and won't blow past the outside of any containers they are in.



This is a site dedicated to all those who are over the top, overblown, bombastic, inflated or ostentatious.

## Hall of Fame

**Sam Kinison**

Comic, screamed his way through the 80's.

**Chris Crocker**

"Leave Britney Alone!"

**Bobcat Goldthwait**

Beause it was the 80's

## Hall of Shame

**Mark Linn-Baker**

Cousin Larry

**Davie**

It's Daria.

**Stephen Tobolowsky**

You've seen him.

*Figure 9*

All the CSS for this Exercise:

```
img{
    max-width:none;
}

img{
    max-width:100%;
}

@media (-webkit-min-device-pixel-ratio: 2){
    .person img{
          max-width: 180px;
     }
}

.person img{
    width:50%;
}
```

I'll cop to it, the solution I use for images is a bit simplistic. Basically, we serve up the biggest images we can and then down scale for smaller image footprints. It's simple but it has one major problem: I am sending the biggest picture over the wire that I can. On a network connected desktop machine, that's no problem regardless of the display abilities of the device. But on mobile devices often with poor screens and poor connections that's expensive for an image that the user won't be able to fully appreciate.

This is a common problem, and there are a number of solutions out there. There is also an effort from the W3C to create a standard that handles this. But dealing with this is outside the scope of this lab. Rest assured there are other minds at work on this problem, and here are some resources to help you explore this topic.

- http://decodize.com/html/simple-responsive-image-technique/
- http://adaptive-images.com/
- http://css-tricks.com/which-responsive-images-solution-should-you-use/
- https://github.com/scottjehl/picturefill/

# Code Exercise 3 - Media Queries

This section will go deeper into media queries which we touched on in the last section.

Over the course of this exercise we will be taking the code in the working directly from how it appears in step_2 to how it appears in step_3.

- Start with code in */lab/code/working*
- Open *index.html* in Adobe Edge Code (or code editor of your choice)
- Open *index.html* in Chrome.

# Add a queries.css file to the project

- In the folder lab/code/working/css add a file named queries.css
- In index.html add the following line of code to the HTML header:

```
<link rel="stylesheet" href="css/queries.css"/>
```

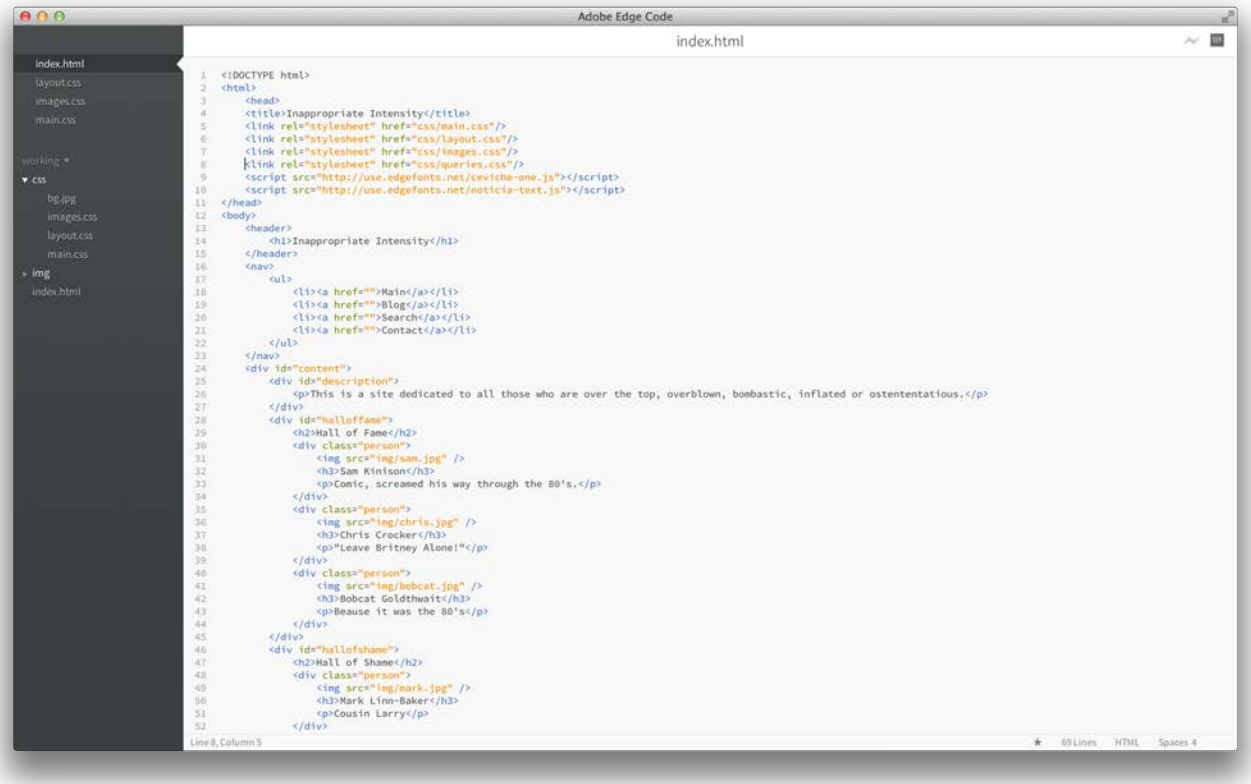


*Figure 1*

We're going to resize the browser to about 700px. When we get there we should see that the description starts to flow under the "Hall Of Fame" similar to what happened in the design portion of this lab. We need to tell the browser to do something different once we get this low on screen width. To do that we'll add a media query based on screen width.

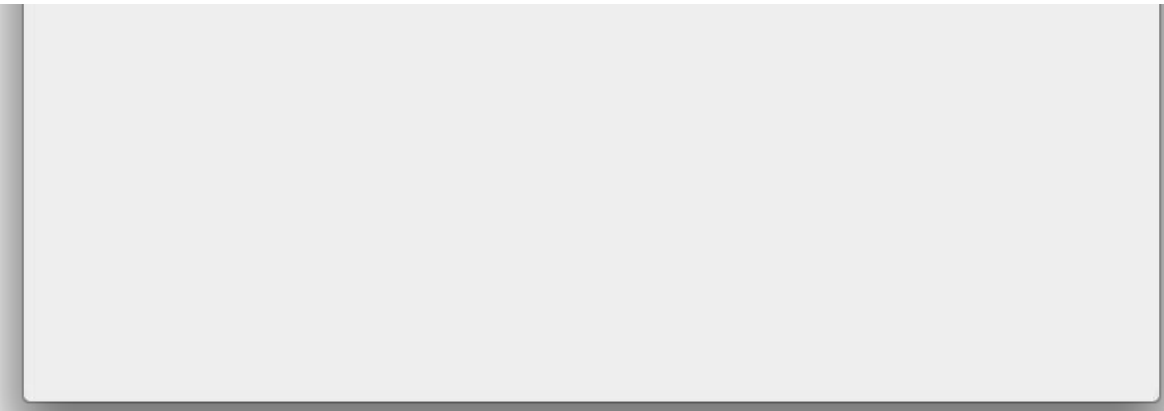


*Figure 2*

Add any rules below to queries.css.

```
@media all and (max-width: 850px) {
     #description{
          width: 100%;
     }
}
```

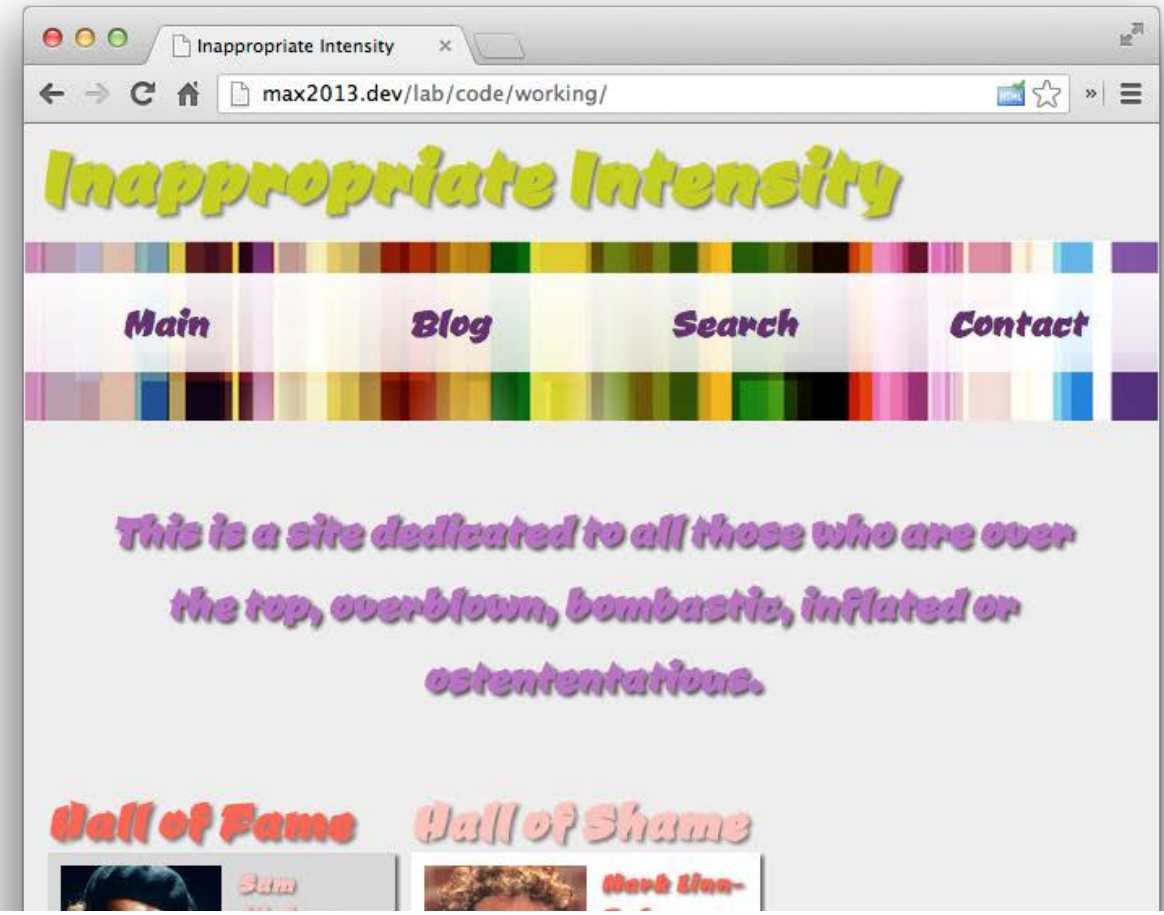
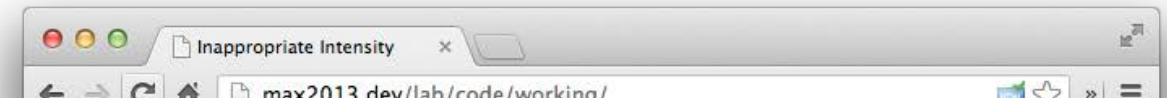



*Figure 3*

Now we'll want to go back and make sure that the other columns fill up their space properly.

```
@media all and (max-width: 850px) {
     #description{
          width: 100%;
     }
    #hallofshame, #halloffame{
          width: 50%;
     }
}
```



# Inappropriate Intensity

Main | Blog | Search | Contact

**This is a site dedicated to all those who are over the top, overblown, bombastic, inflated or ostentatious.**

## Hall of Fame

### Sam Kinison

Comic, screamed his way through the 80's.

### Chris Crocker

"Leave Britney Alone!"

### Bobcat Goldthwait

Beause it was the 80's

## Hall of Shame

### Mark Linn-Baker

Cousin Larry

### Daria

It's Daria.

### Stephen Tobolowsky

You've seen him.

*Figure 4*

Resizing looks good, but we'll want to go even lower. Take the browser down as skinny as it will go. On Chrome this is 400px.


*Figure 5*

It's not unusable, but on the skinny columns just really don't feel right. At some point we'll want to just go to a one column layout. (It's not a coincidence that the original un-laid out

version of the site was one long column.)

We'll add another media query

```
@media all and (max-width: 550px) {


}
```

The the rest of the CSS will go inside that query:

```
#hallofshame, #halloffame{
      width: 100%;
 }
```

# top, overblown, bombastic, inflated or ostententatious.

## Hall of Fame

### Sam Kinison

Comic, screamed his way through the 80's.

### Chris Crocker

"Leave Britney Alone!"

### Bobcat Goldthwait

Beause it was the 80's

*Figure 6*

We'll also want to adjust the navigation.

```
nav ul li a{
          width: 100%;
 }
```

This is a site dedicated to all those who are over the top, overblown, bombastic, inflated or ostentatious.

## Hall of Fame

### Sam Kinison

Comic, screamed his way through the 80's.

### Chris Crocker

"Leave Britney Alone!"

*Figure 7*

Looks good, but if we were to shrink down the height of the screen to a small size (400px by 600px) we'd see that a large amount of the screen is taken up just by header and navigation. Perhaps we want to tone down some of that

Let's get rid of the background that peeks through in the navigation.

```
nav ul {
        padding: 0;
    }
```

And let's tweak some of the font-sizes to make a little more content fit on the screen.

```
h1{
        font-size: 4em;
    }

    #description p{
        font-size: 3em;
    }
    h2{
        font-size: 3.5em;
    }
```

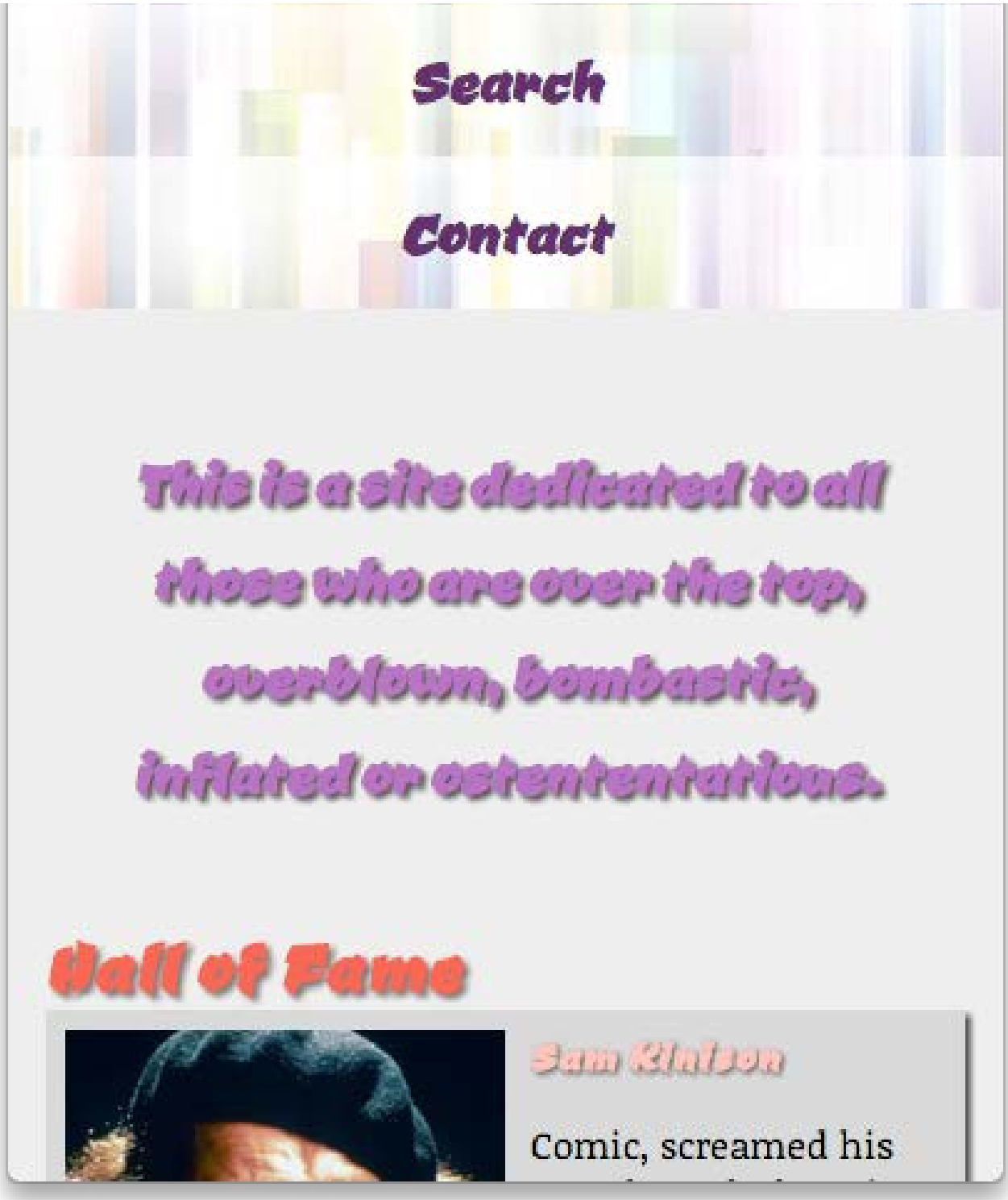



*Figure 8*

Looking much more appropriate on smaller devices. However, if we scale the browser larger then 1000px we'll find that the screen is locked into a max-width of 1000px. We can do a little better for users of large screens.

```
@media all and (min-width: 1100px) {
    body{
```

```
        max-width: 90%;
    }
}
```

There we have defined experiences for larger desktop screens down to the smartphone screen.

Complete CSS for this exercise:

```
@media all and (max-width: 850px) {
     #description{
          width: 100%;
     }
    #hallofshame, #halloffame{
          width: 50%;
     }
}

@media all and (max-width: 550px) {
    #hallofshame, #halloffame{
          width: 100%;
     }
    nav ul li a{
          width: 100%;
     }

    nav ul {
        padding: 0;
    }

    h1{
        font-size: 4em;
    }

    #description p{
        font-size: 3em;
    }
    h2{
        font-size: 3.5em;
    }

}
@media all and (min-width: 1100px) {
    body{
        max-width: 90%;
    }
}
```

# Code Exercise 4 - Tweaks

Okay so we've implemented a responsive design using:

- Flexible Grid
- Flexiible Media
- Media Queries

But there is a little more to consider. We've been showing this design in a browser on a desktop under what are almost ideal conditions. On a real mobile device things like glare and reduced attention will make us want to make sure there is a little more contrast. Consider where the halloffame meets the hallofshame.

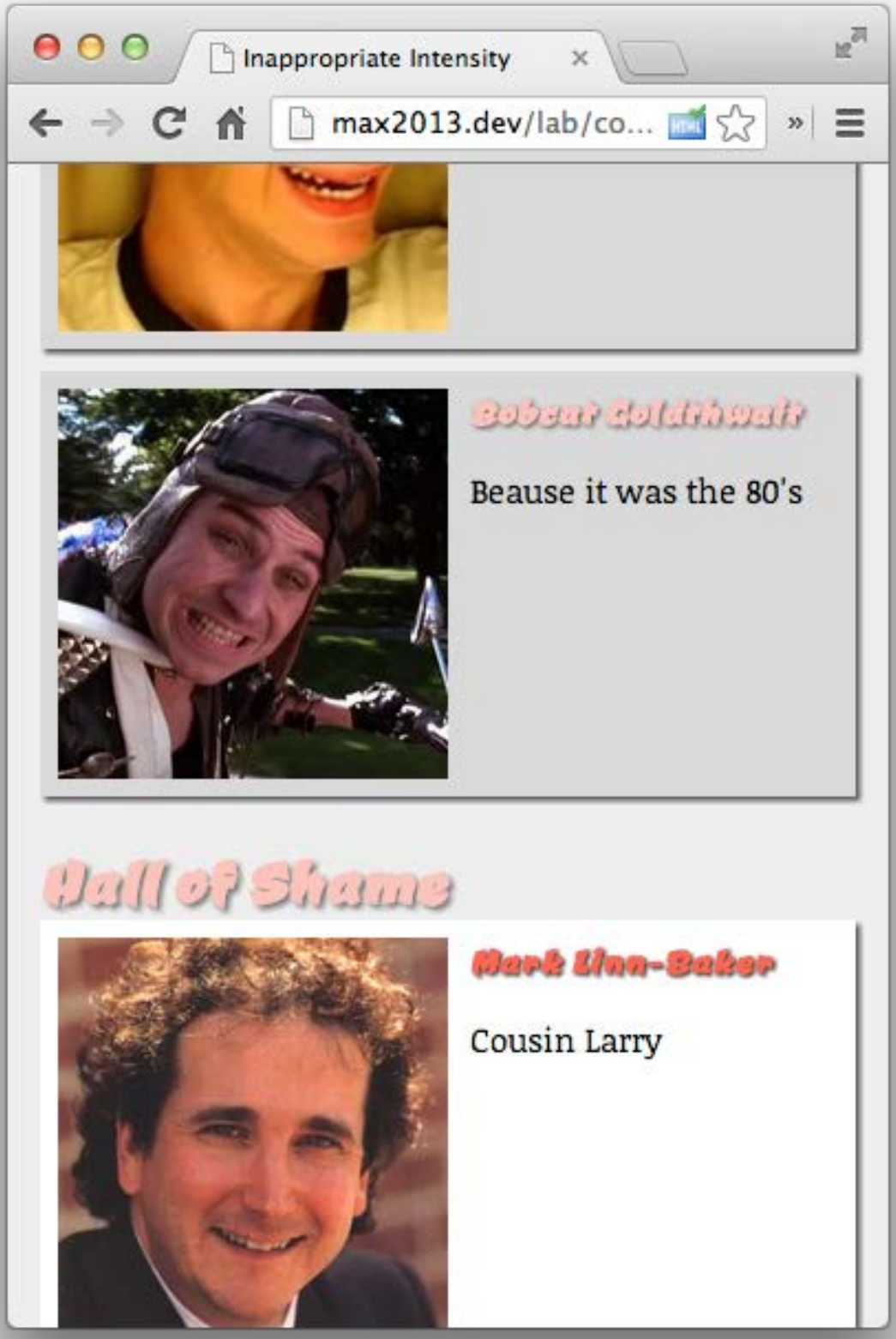

Inappropriate Intensity
max2013.dev/lab/co...
Bobcat Goldthwait
Beause it was the 80's
Hall of Shame
Mark Linn-Baker
Cousin Larry

*Figure 1*

You have a body with a #eeeeee background with Cousin Larry on a #ffffff background and Bobcat on a #d9d9d9. In short there is little difference between these and maybe we want to increase the contrast for a better on the go experience.

Over the course of this exercise we will be taking the code in the working directly from how it appears in step_3 to how it appears in step_4.

- Start with code in */lab/code/working*
- Open *index.html* in Adobe Edge Code (or code editor of your choice)
- Open *index.html* in Chrome.

## Add a tweaks.css file to the project

- In the folder lab/code/working/css add a file named tweaks.css
- In index.html add the following line of code to the HTML header:

```
<link rel="stylesheet" href="css/tweaks.css"/>
```

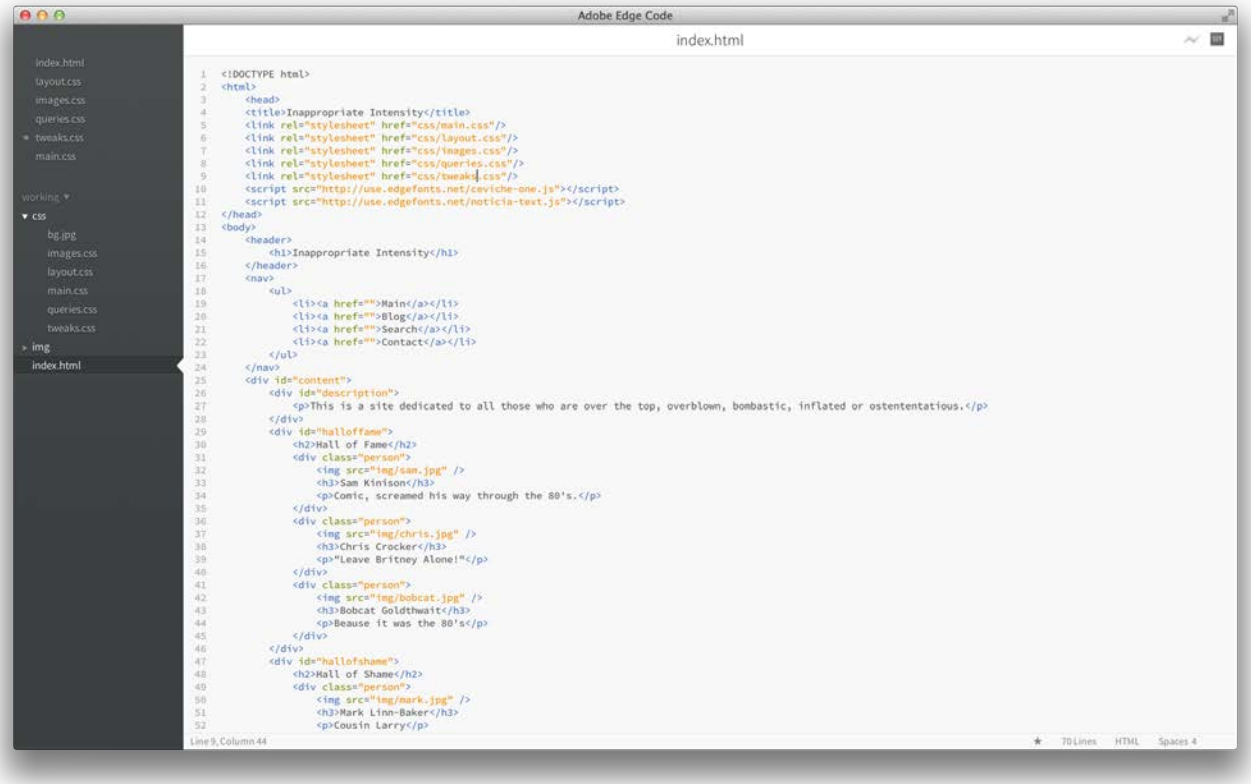



*Figure 2*

Let's add some css to tweaks.css. Start with a media query.

```
@media all and (max-width: 550px) {

}
```

Add let's adjust the contrast for those sections.

```
body{
        background-color: #CCCCCC;
    }

    #halloffame .person{
        background-color: #eeeeee;
    }

    #hallofshame .person{
        background-color: #FFFFFF;
    }
```

We get that extra contrast we want.

Normally you cannot get Chrome to resize below 400px, which is aggravating if you need to test on resolutions lower then that. However there is a little trick you can do to get it to.

Open Chrome → View → Developer → Developer Tools

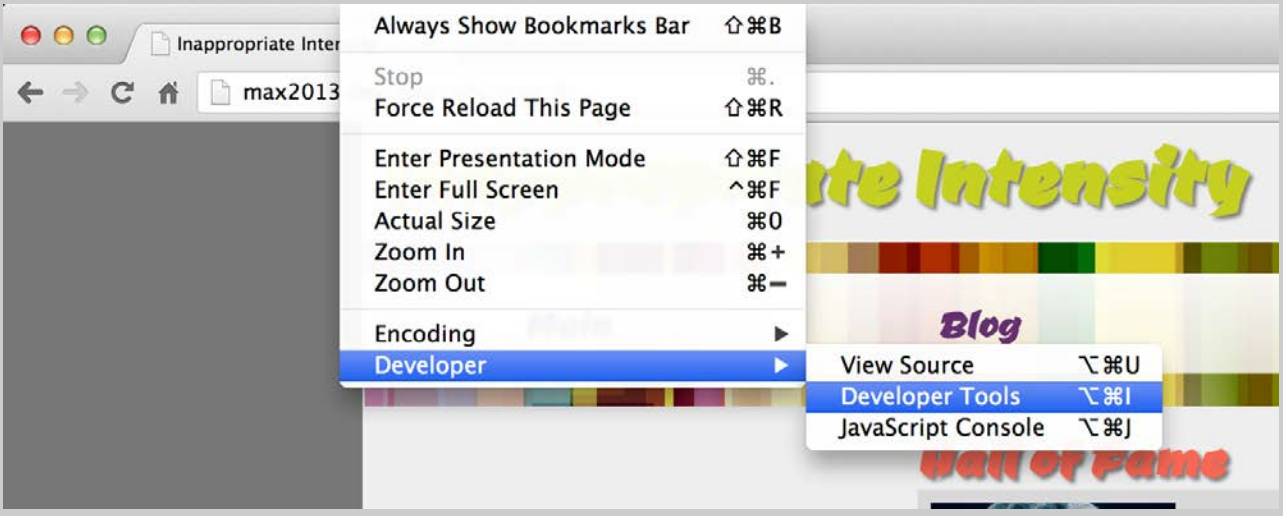



*Figure A1*

This will open the Chrome Developer Tools

In the lower left corner there is an icon that allow you to undock the panel.

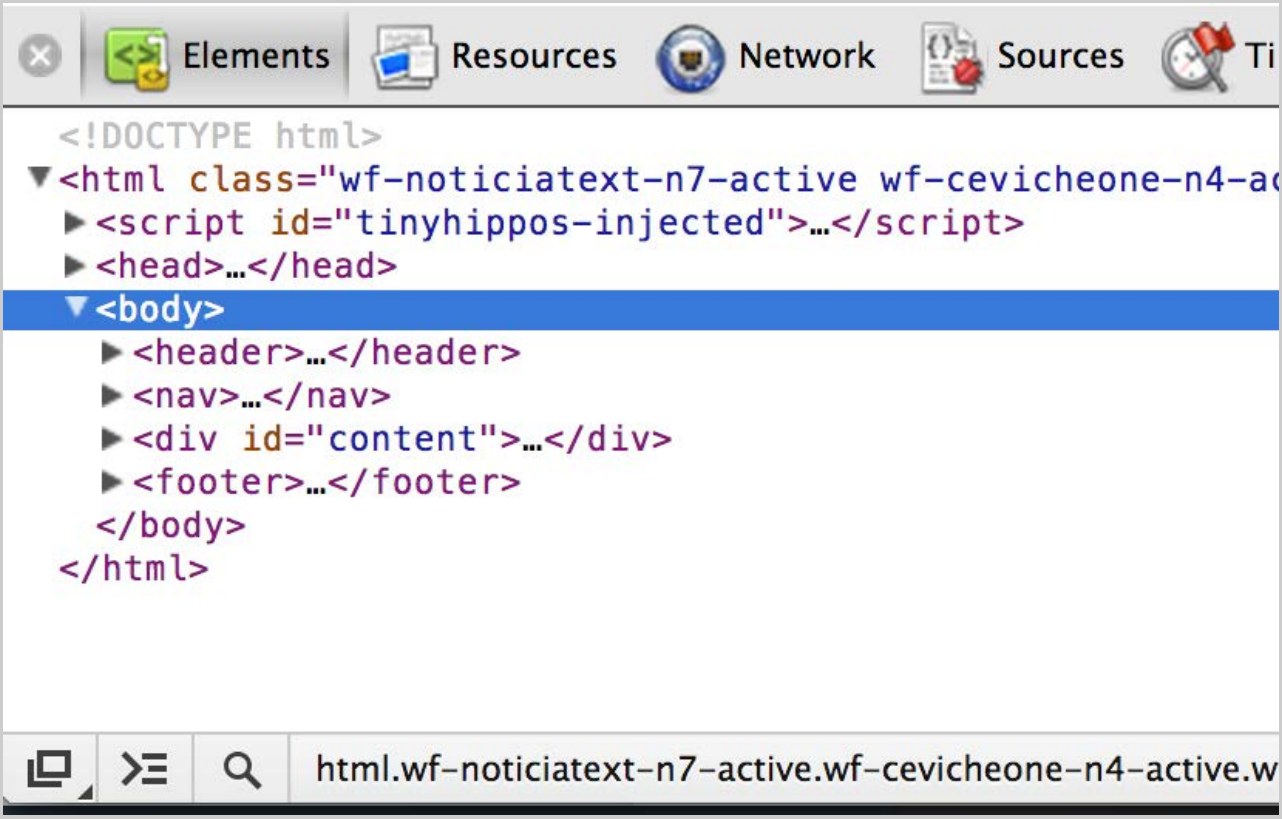



*Figure A2*

If you hold the button down you have the option of docking the tools on the right of the browser. Do that.

.

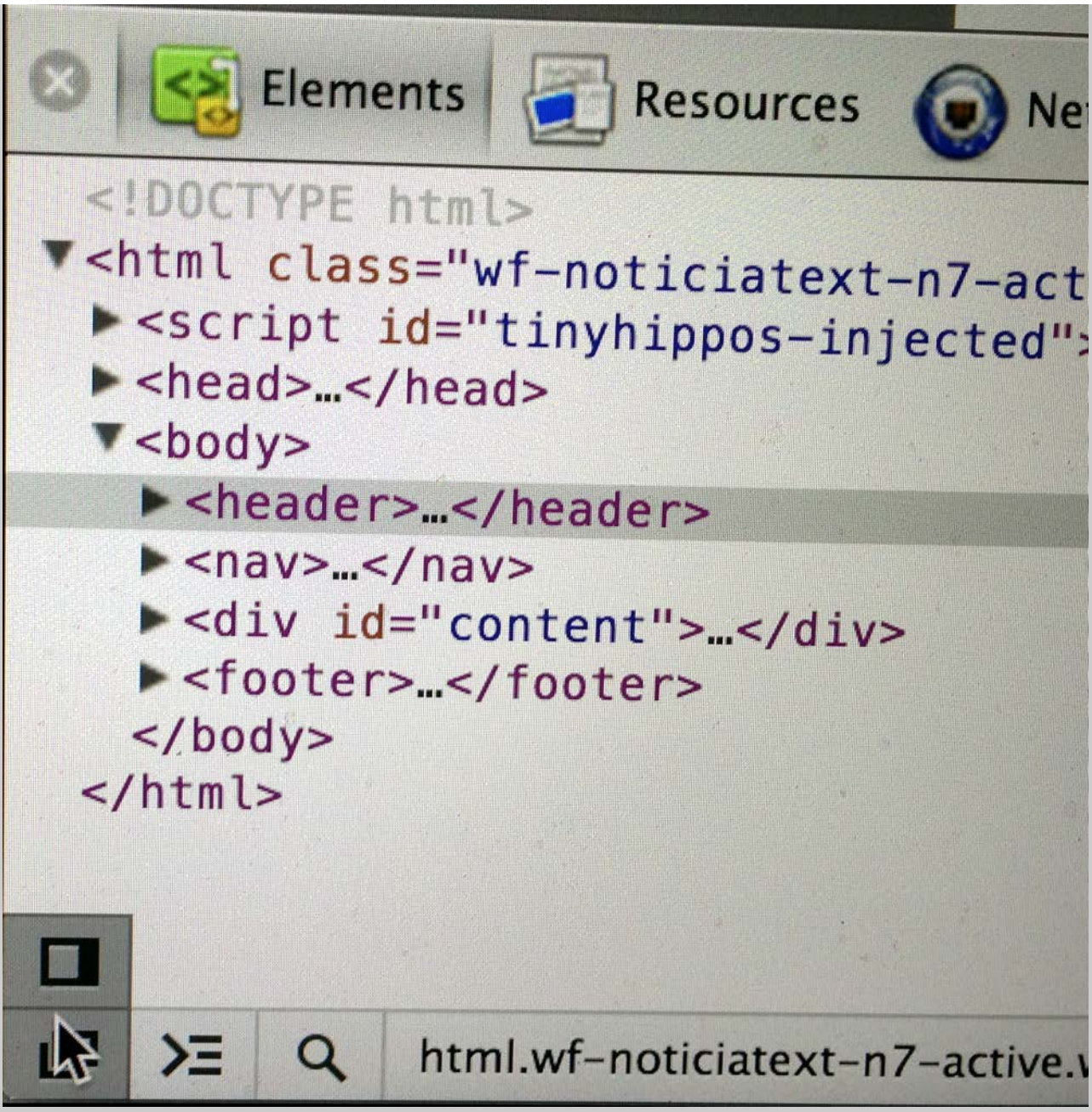



*Figure A3*

Now that the panel is on the right, you can resize the viewport below 400px.


*Figure A4*

*How do you know its width?*

Hover over the HTML code in the Developer Tools. Specifically hover over the <body>. When you do this you will get a popover that tells you the dimensions of the body.

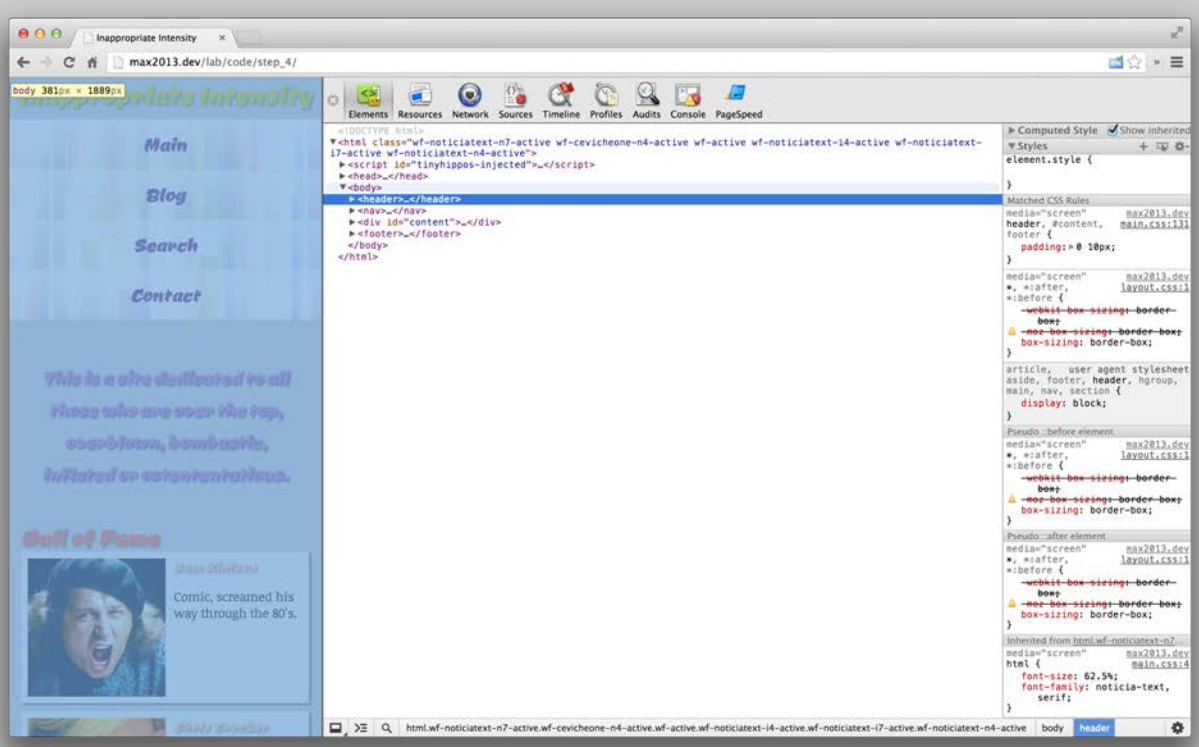



*Figure A5*

Let's go to a smaller extreme. Let's talk about the iPhone 3g generation of devices. A lot of them are still out there. The standard resolution for those devices is 320px X 480px. At that size the screen is still usable, but leaves a little bit to be desired.

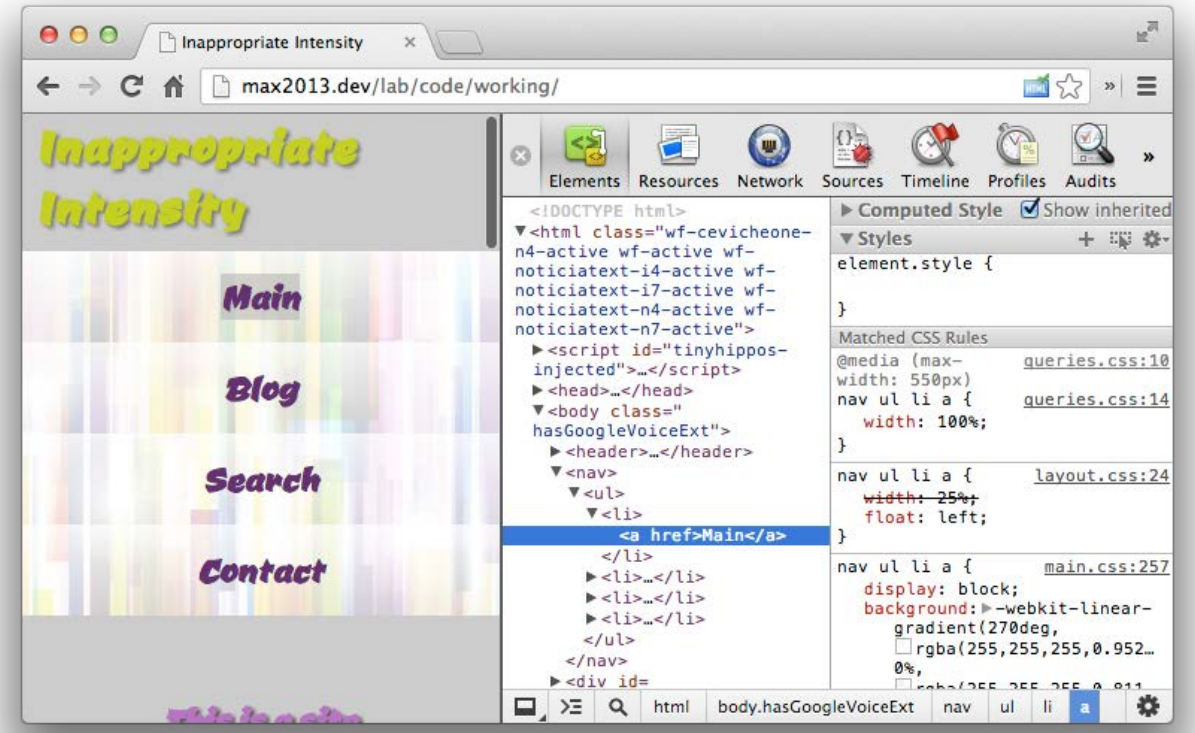



*Figure 3*

So we should really shrink down the heading and other top places items again.

```
@media all and (max-width: 330px) {

    h1{
        font-size: 3em;
    }
    #description p {
        font-size: 2.5em;
    }
    nav ul li {
        font-size: 1.9em;
    }
}
```

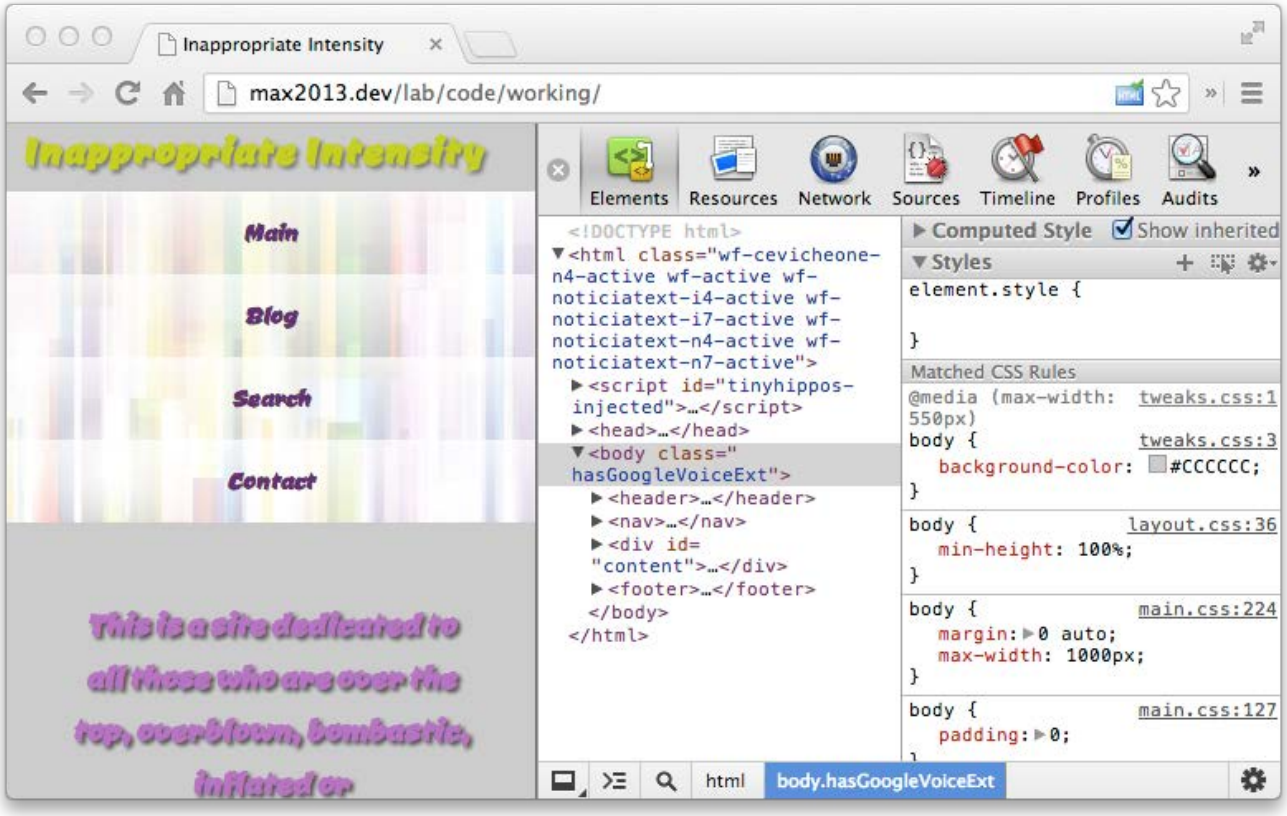



*Figure 4*

Looks better, but I'm not happy with how the font looks at this size. On a iPhone this small, I'm better off using a font that has been optimized for the iPhone.

```
@media all and (max-width: 330px) {

    h1{
        font-size: 3em;
        font-family:MarkerFelt-Thin, sans-serif;
    }
    #description p {
        font-size: 2.5em;
        font-family:MarkerFelt-Thin, sans-serif;
    }
    nav ul li {
        font-size: 1.9em;
        font-family:MarkerFelt-Thin, sans-serif;
    }

}
```

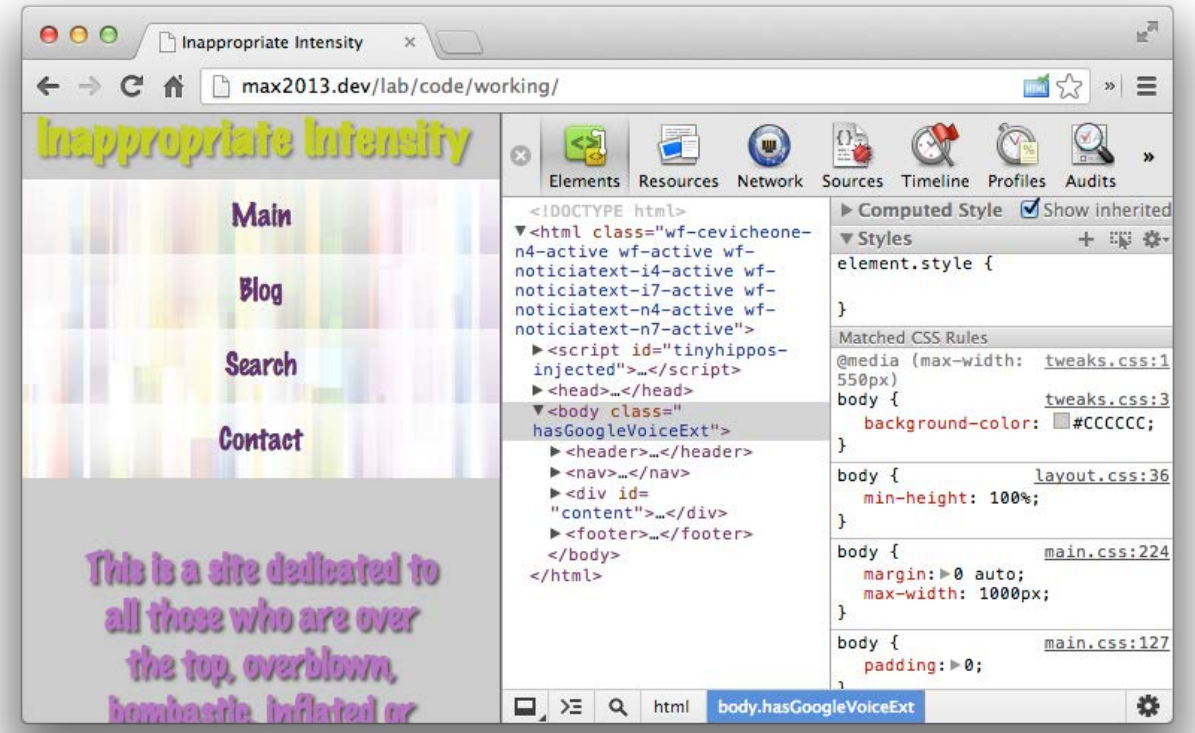



*Figure 5*

You now have a design that can stretch and scale over a wide variety of devices. It's usable at all views. It's tweaked to deal with certain edge cases: very small screens, retina displays. And it's a great starting spot for your own responsive sites.

Complete CSS for this exercise:

```
@media all and (max-width: 550px) {

    body{
     background-color: #CCCCCC;
    }

    #halloffame .person{
        background-color: #eeeeee;
    }

    #hallofshame .person{
        background-color: #FFFFFF;
    }


}
```

```
@media all and (max-width: 330px) {

    h1{
        font-size: 3em;
        font-family:MarkerFelt-Thin, sans-serif;
    }
    #description p {
        font-size: 2.5em;
        font-family:MarkerFelt-Thin, sans-serif;
    }
    nav ul li {
        font-size: 1.9em;
        font-family:MarkerFelt-Thin, sans-serif;
    }

}
```