

---

---

*is there a breakdown anywhere of the  
semantic of what terminal commands mean  
as it relates to github*

---

---

<http://adriana-ga.github.io/#resources>

---

---

*Will the work automatically save to github after the initial set-up? Or must we type it on the terminal when saving our projects?*

---

---

You need to follow workflow every time!

```
git add .  
git commit -m "what changes you made"  
git push origin master
```

---

---

*how do you do the Nav bar from the travel  
blog?*

---

---

[https://github.com/alcastaneda/  
FEWD10\\_solutions/tree/master/  
lesson3\\_code/travel\\_blog\\_solution\\_ga](https://github.com/alcastaneda/FEWD10_solutions/tree/master/lesson3_code/travel_blog_solution_ga)

---

---

*Still searching for a good list of windows terminal commands. Still need lots more practice with git commands. Not sure how to employ versioning - I understand you can get the log and access the "receipt" for a version of your code, but I don't understand how to apply the older version.*

---

---

git revert commit#



---

---

*how to make images show up for a specific view of the page. I know we're learning it in the next class, but then why is it on our homework?*

---

---

It wasn't part of your homework  
But now it is! The completed lab is due this  
week. And we will learn how to work with the  
picture today

---

---

*Will we explore github and learn how to  
incorporate sections of code we find online  
into our own files?*

---

---

SUUUURE

and what do you mean incorporate code you  
found?

# AGENDA

---



- Final Project Overview
- Review
- Refactoring
- Advanced CSS Positioning
- Lab — Startup Matchmaker

# **LEARNING OBJECTIVES**

- Practice web development by transforming a design comp into a webpage. (Startup Matchmaker)
- Be able to use positioning and z-index in front end

---

**FEWD**

---

# FINAL PROJECTS

---

# FINAL PROJECTS

---

## WHERE CAN I GET SOME INSPIRATION FROM WHAT PAST STUDENTS HAVE DONE?

- Visit the General Assembly [Gallery](#)

## WHERE SHOULD I BE RIGHT NOW?

- By the end of this week you should have looked through the gallery and jotted down a few ideas. Wireframes/project proposals will be due at the end of Week 5!



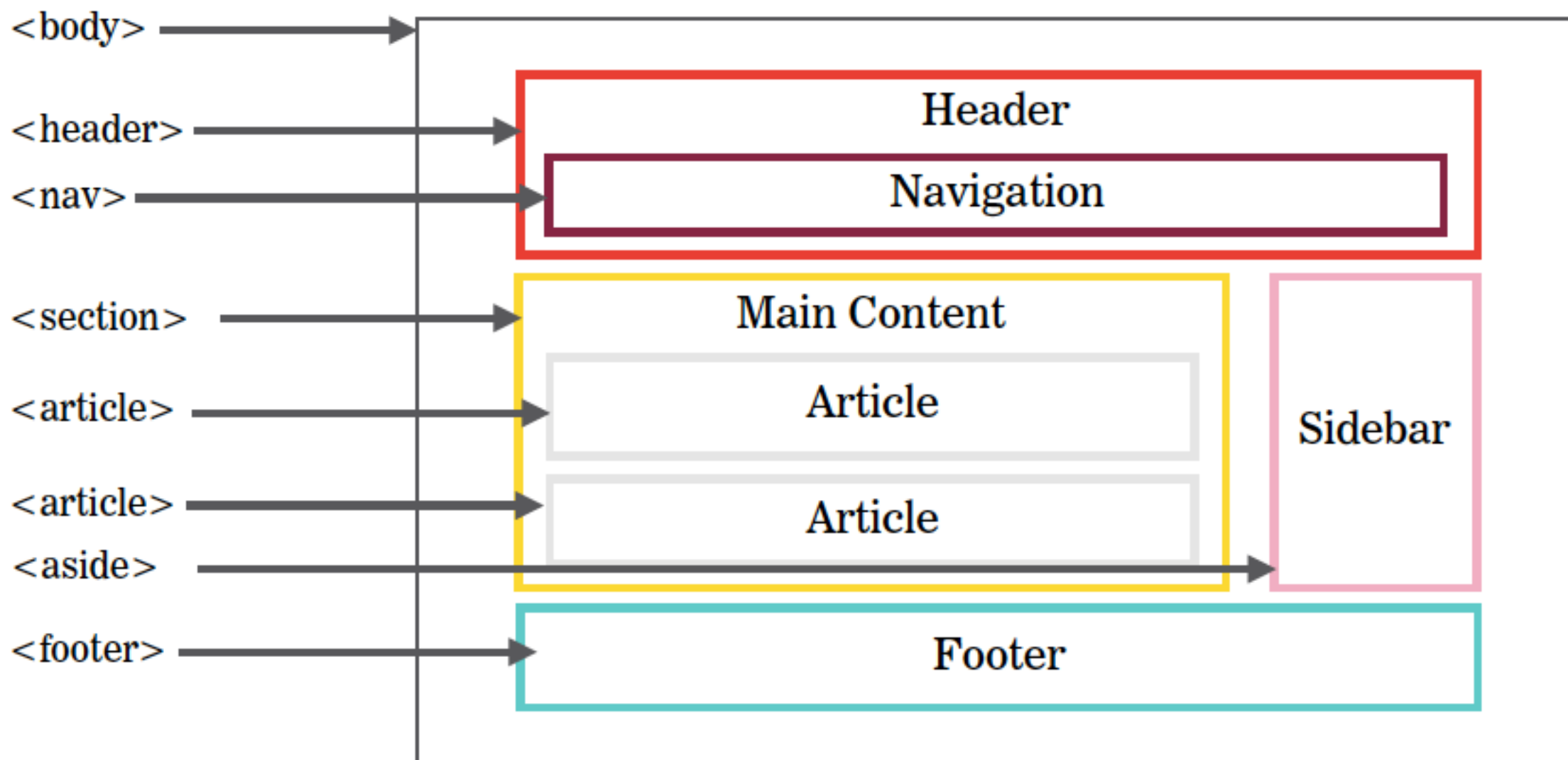
---

**FEWD**

---

**REVIEW**

# STRUCTURAL ELEMENTS



# FLOATS

---

Values for the float property:

```
p {  
  float: left;  
}
```

▸ **float: left;** Floats an element to the left side

```
h1 {  
  float: right;  
}
```

▸ **float: right;** Floats an element to the right side

```
div {  
  float: none;  
}
```

▸ **float: none;** (the default) ensures the element will not float

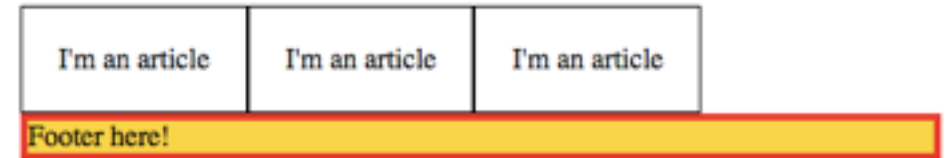
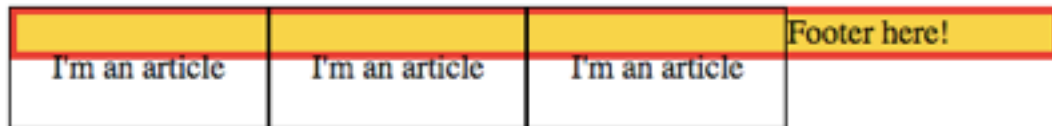
---

# CONFUSING NAMES — KEEPING THINGS STRAIGHT

---

## CLEAR: BOTH;

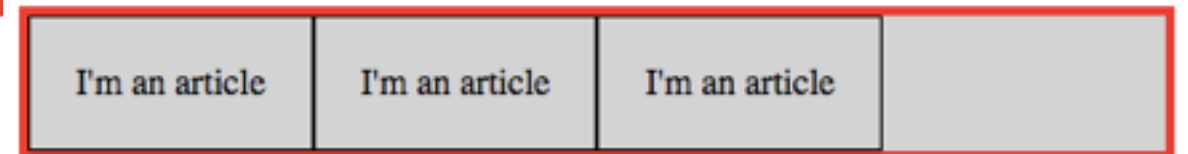
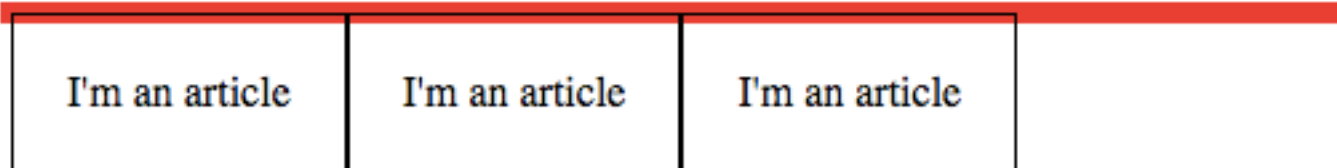
*Make sure an element starts on a new line*



```
footer {  
  clear: both;  
}
```

## CLEARFIX:

*Fixes collapsed parent*



### PT. 1 — ADD CSS CLASS:

```
.clearfix:after {  
  content: "";  
  display: table;  
  clear: both;  
}
```

### PT. 2 — ADD CLASS TO HTML:

```
<div class="clearfix">  
  <p>1</p> <!-- float: left -->  
  <p>2</p> <!-- float: left -->  
  <p>3</p> <!-- float: left -->  
</div>
```

---

# STEPS TO ACHIEVE A MULTI-COLUMN LAYOUT

---

**1. MAKE SURE EACH COLUMN HAS A WRAPPER AROUND IT IN YOUR HTML**

**2. ADD BOX-SIZING: BORDER-BOX; TO EVERYTHING (USE THE \* CSS SELECTOR)**

**3. GIVE A WIDTH TO EACH COLUMN (PREFERABLY IN %)**

**4. FLOAT EACH COLUMN TO LEFT**

**5. USE PADDING TO ADD SPACE BETWEEN COLUMNS**

**6. CLEAR ANYTHING UNDERNEATH YOUR COLUMNS I.E. A FOOTER  
USING THE CSS CLEAR PROPERTY (CLEAR: BOTH;)**

---

## **ACTIVITY**

---

**Floats, Clear, and Clearfix**  
**Time: 10min**

---

**FEWD**

---

# CHROME DEV TOOLS!

---

## LET'S USE THE INSPECTOR!

---

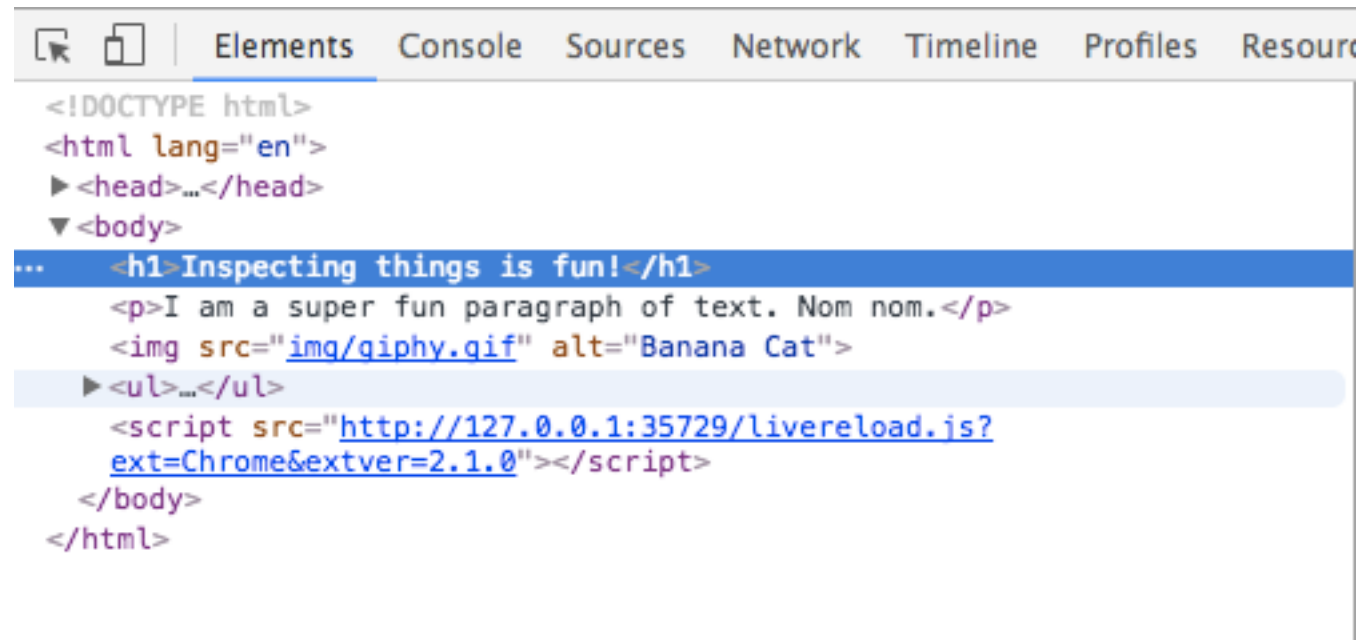
There are several ways to open Chrome dev tools:

- Right click on an element and click "inspect"
- In Chrome, go to view > developer > Developer Tools
- Keyboard shortcut: Mac: Cmd + Opt + I Windows: F12, Ctrl + Shift + I



# PICKING AN ELEMENT

Choose an element to inspect by clicking on the element in the "Elements" tab

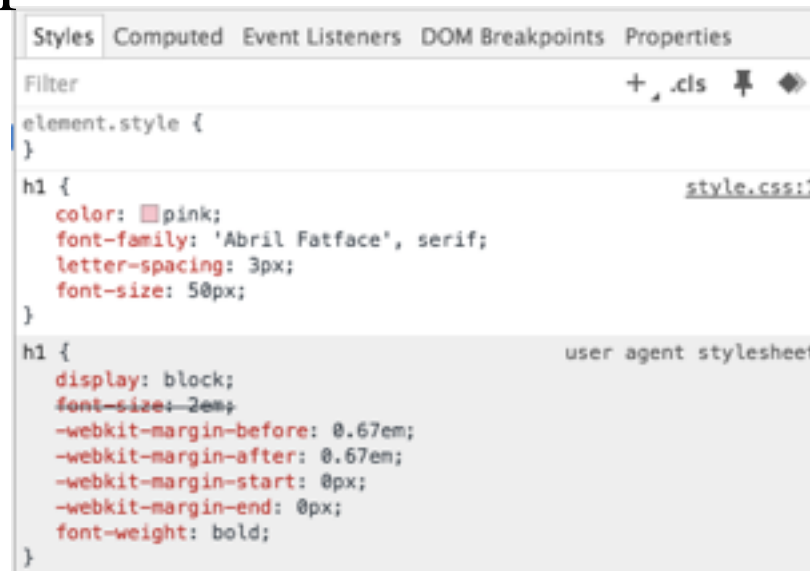


---

# SEEING STYLES AND EXPERIMENTING

---

You can see what styles you've added (and default styles added by the browser) to the element in the "styles" panel.



You can also experiment here!

```
✓ color: pink;
✓ font-family: 'Abril Fatface', serif;
✓ letter-spacing: 3px;
  font-size: 100px;
}
```

# Lesson 6

---

**FEWD**

---

# REFACTORING

---

# REFACTORING

---

## WHAT IS REFACTORING?

- Code refactoring is the process of restructuring existing computer code—changing the factoring—without changing its external behavior
- Refactoring improves nonfunctional attributes of the software

---

# REFACTORING

---

## THE GOLDEN RULE

**Write D.R.Y. Code!!!  
(Don't Repeat Yourself!)**

---

# REFACTORING

---

## OTHER TIPS

**Code should be easy to read!**

---

# REFACTORING

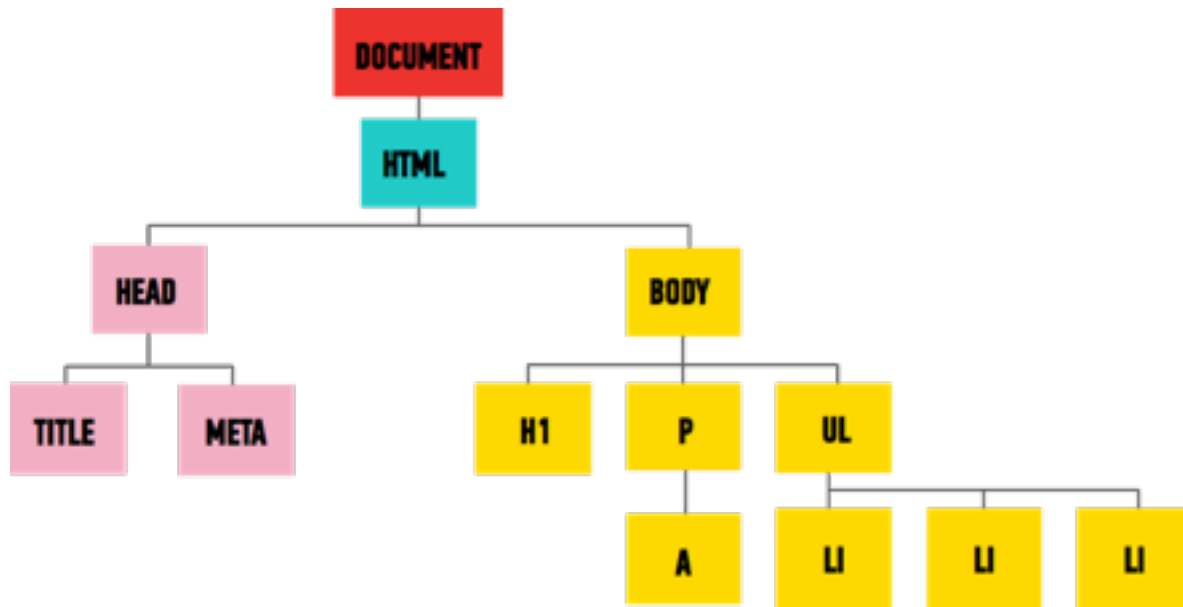
---

## What is easy to read?

- Indented Code
- Helpful comments
- Organized
- Clear naming



# REFACTORING -- INDENTED CODE



```
1 <!DOCTYPE html>
2 <html lang="en">
3   <head>
4     <meta charset="UTF-8">
5     <title>Document</title>
6   </head>
7   <body>
8
9     <h1>Site title</h1>
10    <p>Bacon ipsum dolor amet brisket tail
    frankfurter cupim pig salami. Fatback
    porchetta strip steak doner chicken <a href="
    http://www.jamieoliver.com/recipes/pork-recipes
    /pork-belly-roast/">pork belly</a></p>
11    <ul>
12      <li>Bacon</li>
13      <li>Chicken</li>
14      <li>Meatloaf</li>
15    </ul>
16
17  </body>
18 </html>
```

---

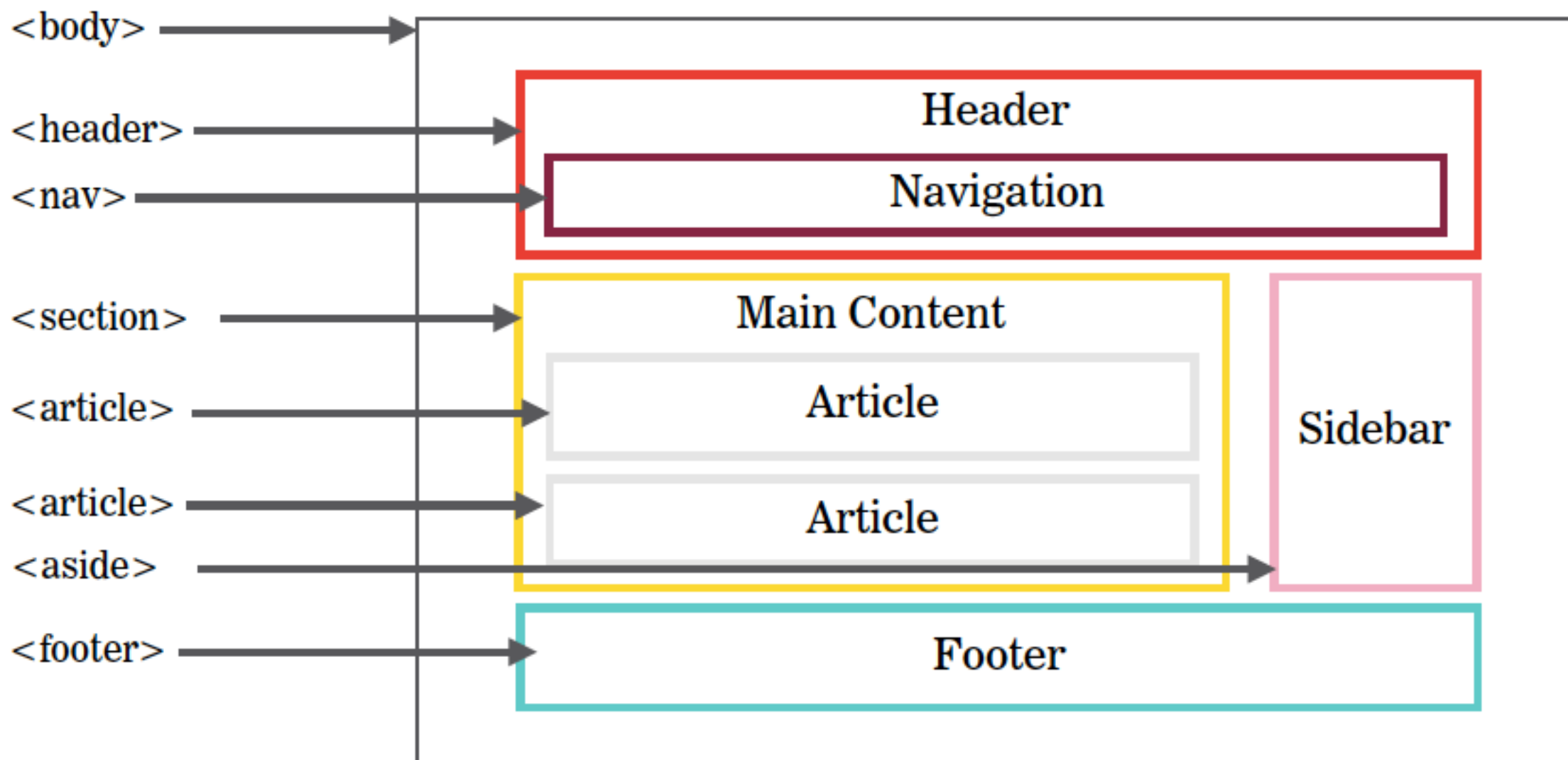
## REFACTORING — HELPFUL COMMENTS

---

Leave a comment for what part of html you're styling

If something is confusing, leave a comment that explains what code does

# STRUCTURAL ELEMENTS



---

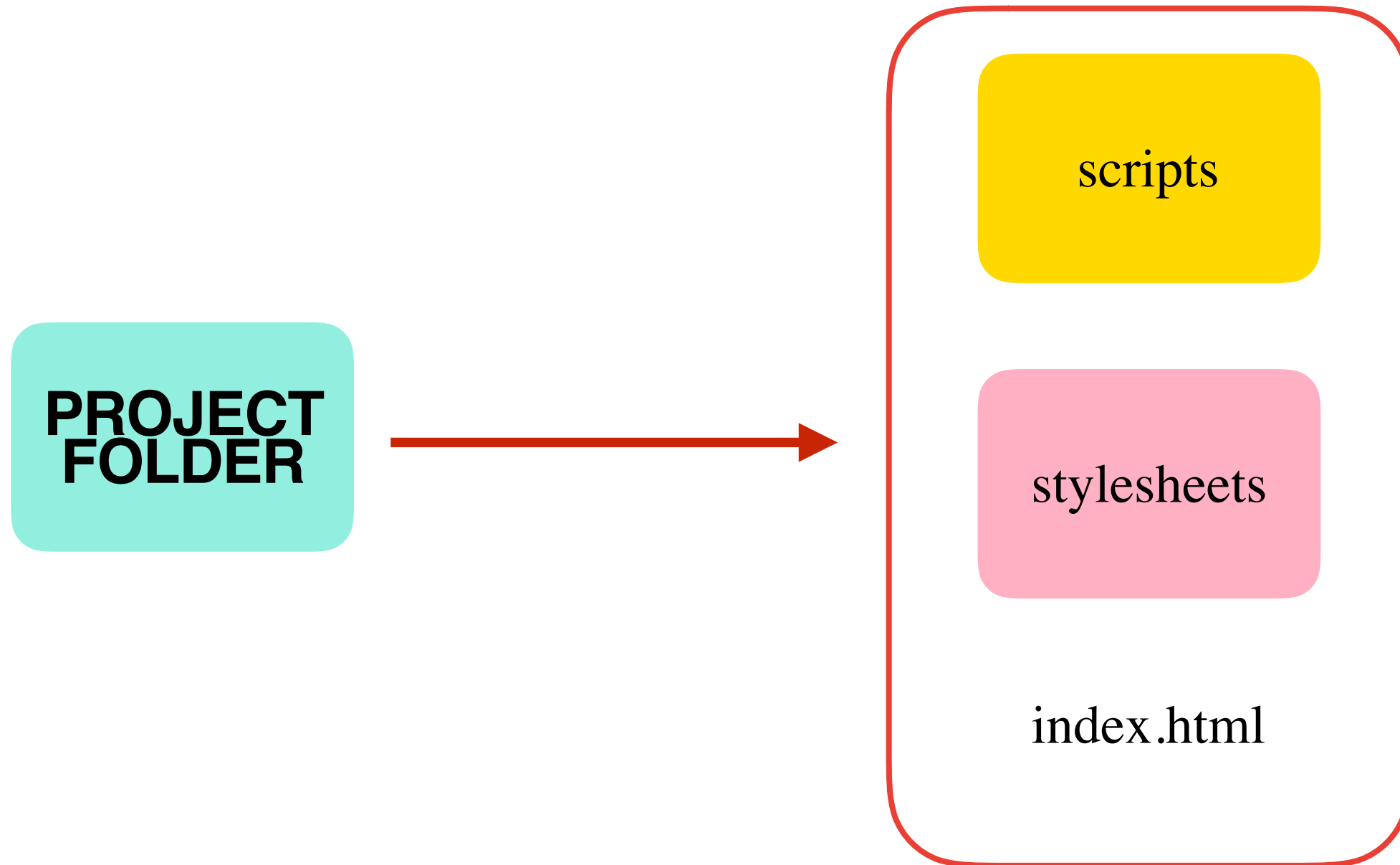
# REFACTORING — HELPFUL COMMENTS

---

```
1  *{
2    box-sizing: border-box;
3  }
4
5  /*HEADER STYLING*/
6  h1{
7    padding-top: 50px;
8  }
9
10 /*NAV STYLING*/
11 nav{
12   z-index: 2;
13   position: fixed;
14   width: 100%;
15 }
16
17 nav p, ul{
18   width: 49%;
19   display: inline-block;
20 }
```

# REFACTORING — ORGANIZED

---



---

## REFACTORING — CLEAR NAMING

---

This applies mostly to javascript but in general, it's not obscure.

For example, if I want to make a class to style my images a certain way, I would name my class to reflect that ex. class = “center-img” versus naming it something confusing like class = “ci”

---

**FEWD**

---

# ADVANCED CSS POSITIONING

# STATIC POSITIONING

---

- This is the normal flow of the document, the **default**
- Elements render in order, as they appear in the document flow.

```
.my-class {  
  position: static;  
}
```



---

# RELATIVE POSITIONING

---

- Relative positioning moves an element *relative to where it would have been in normal flow*.
- For example, "left: 20px" adds 20px to an element's **left** position
- **Creates a coordinate system for child elements.**

```
.my-class {  
  position: relative;  
  top: 20px;  
  left: 30%;  
}
```

---

# ABSOLUTE POSITIONING

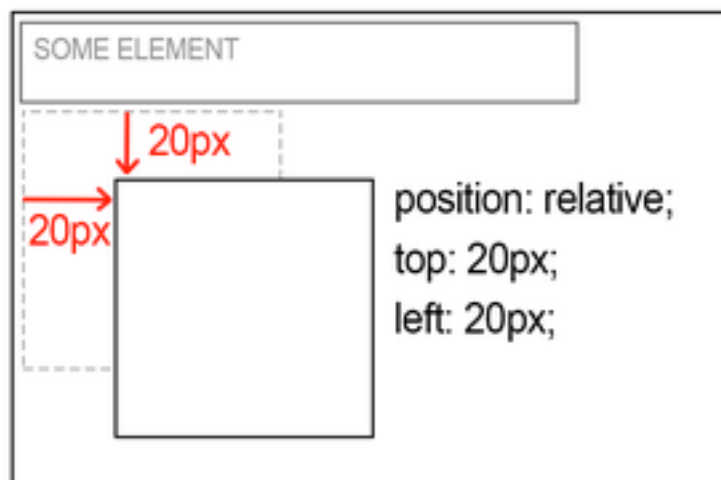
---

- ▶ When the *position* property is given a value of *absolute*, an element is taken out of the normal flow of the document.
- ▶ This element no longer affects the position of other elements on the page (they act like it's not there).
- ▶ You can add the *right*, *top*, *left* and *bottom* properties to specify where the element should appear relative to its first positioned (not static) ancestor element

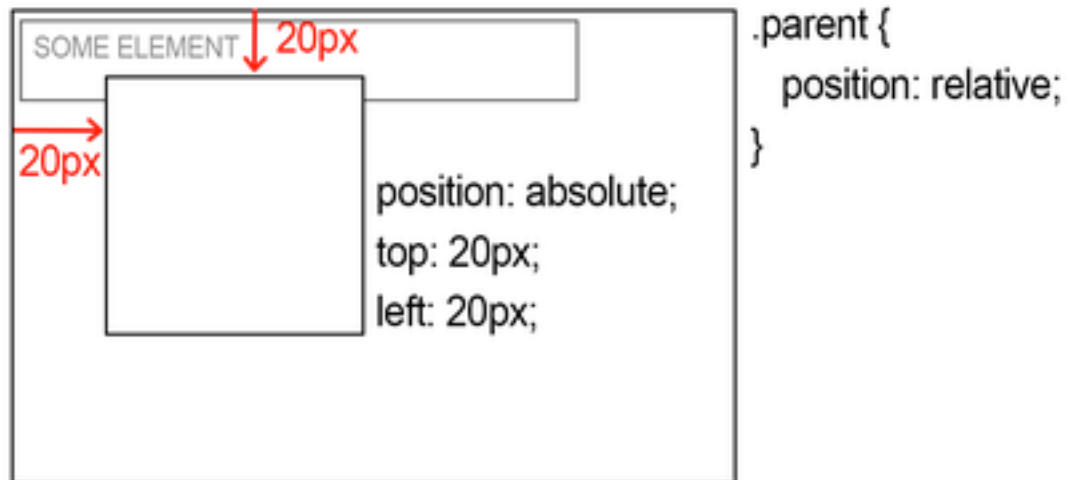
```
.my-class {  
  position: absolute;  
  top: 0;  
  left: 500px;  
}
```

# RELATIVE VS ABSOLUTE POSITIONING

## RELATIVE POSITIONING:



## ABSOLUTE POSITIONING:



---

# FIXED POSITIONING

---

- ▶ When the *position* property is given a value of *fixed*, the element is positioned in relation to the browser window
- ▶ When the user scrolls down the page, it stays in the same place.
- ▶ You can add the *right*, *top*, *left* and *bottom* properties to specify where the element should appear in relation to the browser window.

```
.my-class {  
  position: fixed;  
  top: 0;  
  left: 500px;  
}
```

---

## OVERLAPPING ELEMENTS — Z-INDEX

---

- When using relative, fixed or absolute positioning, elements can overlap.
- When elements overlap, the elements that appear later in the HTML code sit on top of those that appear earlier in the page.
- If you want to control which elements are layered on top of each other, you can use the z-index property.
- This property takes a number — the higher the number the closer that element is to the front.
- Similar to 'bring to front' and 'send to back' in programs like *Adobe Illustrator*.

```
.my-class {  
  z-index: 10;  
}
```

---

## **ACTIVITY**

---

**Positioning Fun**  
**Time: 10min**

# LAB — TRAVEL BLOG

---

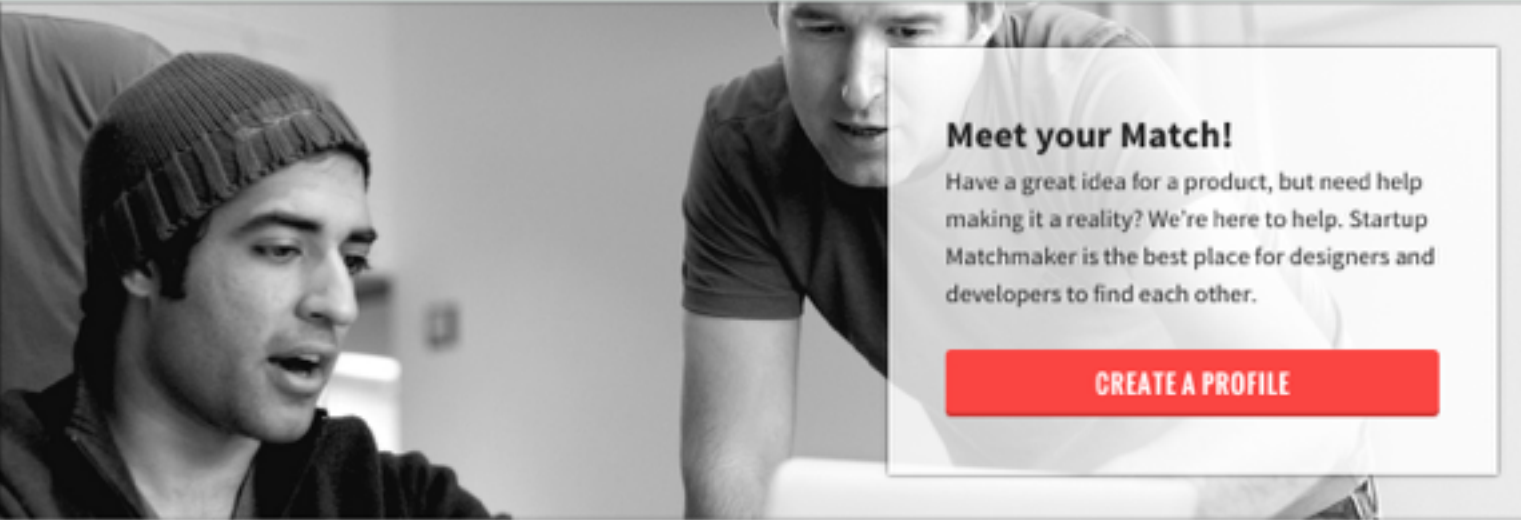


# LAB — STARTUP MATCHMAKER

Startup Matchmaker

DEVELOPERS DESIGNERS How it Works Our Team Blog

*Because two heads are better than one.*



### Meet your Match!

Have a great idea for a product, but need help making it a reality? We're here to help. Startup Matchmaker is the best place for designers and developers to find each other.

[CREATE A PROFILE](#)

### Create a Profile

Are you a Designer? Put yourself out there so that others can find you!

[SIGN UP NOW](#)

### Find a Developer

Looking for a developer to work with on the next big thing? Look no further.

[START YOUR SEARCH](#)

### Find a Designer

Need someone who can make a product intuitive and appealing? Get ready.

[START YOUR SEARCH](#)

© 2013 Startup Matchmaker. Made in NY.



---

**FEWD**

---

# **HOMEWORK**

---

# **HOMEWORK**

---

- **Continue working through the lab we started on today**

---

**ADVANCED CSS**

---

# EXIT TICKETS