

# HACK YALE

< WEB DEVELOPMENT />

[WWW.HACKYALE.COM](http://WWW.HACKYALE.COM)

# HACK

< WEB DEVELOPMENT />

**week\_0**

DIVING IN!

# WELCOME!

## The Agenda

- Introductions
- What is this HackYale?
- Jobs, jobs and mo' jobs
- To the code!



# **INTRODUCTIONS**

LET'S GROW TO LOVE ONE ANOTHER



# INTRODUCTIONS

## RAFI KHAN

- Fifth time teaching HackYale
- Junior, Pierson, Computer Science Major
- Google last summer, teaching at iExperience this summer

# INTRODUCTIONS

## THIS PLACE

- The Center for Engineering, Innovation and Design
- Must be a member! ([ceid.yale.edu](http://ceid.yale.edu))
- Be respectful, etc.

# INTRODUCTIONS

## YOU ALL

- Budding programmers, all-star designers
- Coding is more fun with friends!



# **WHAT IS THIS HACKYALE?**

PROGRAMMING + HACKING CULTURE



# HACKYALE

---

Practical, not as theoretical / academic as a Yale CS class


- Zero -> prototype
  - Not training CTOs
  - Preparing you with the tools to train yourself to do whatever you want
-

# WHY HACKYALE?

---

Good ideas + good developers =  
good tech companies

- Yale  $\supset$  many students with good ideas
  - Yale  $\nexists$  many students who can implement those ideas
-



*“But it’s not that probable that I’m going to make an awesome dynamic web app while I take a classes and go abroad and I double major and I try to set up an internship for next summer/job after college and I stress about my love life and I don’t call my friends from home enough and I worry about whether or not those ants in the corner of my common room are breeding!”*



*“But it’s not that probable that I’m going to make an awesome dynamic web app while I take a classes and go abroad and I double major and I try to set up an internship for next summer/job after college and I stress about my love life and I don’t call my friends from home enough and I worry about whether or not those ants in the corner of my common room are breeding!”*

# Fair Point.

 **GIVE WHAT YOU CAN** 

**AND MAKE TIME TO GIVE A LOT**

# THE 101 CLASS

- Learn how to build pretty, static websites.
- A jumping off point to learn more and build dynamic sites, maybe even this semester.
- For designers, another canvas on which to showcase your craft.

# GOALS

Focus on processes and psychology of web development more than content

- The idea is your responsibility
- Learn by doing; learn by immersion. Lots of implementation, lots of coding
- Memorization as the emergent byproduct of experience

# GOALS

We can't make you successful developers

We can equip you with a kernel of knowledge and key resources with which to make yourselves successful developers



# HACKING CULTURE

THE “DO-IT-YOURSELF” ATTITUDE



# **JOBS, JOBS AND MO' JOBS**

MO' JOBS LESS PROBLEMS



# **JOBS**

HackYale opens a network for finding cool tech jobs

- ▶ Brewster, Google, Art.sy, Microsoft, Palantir, Panorama Education, RedFin, Twitter, SeeClickFix...



**TO THE COMPUTERS!**

*ABOUT TIME!*



## **Quick note: Mac vs. PC**

All examples and screenshots are performed and taken on Macs. However, everything we do in this class should be pretty much the same for those of you on PC.

# THE WORKFLOW

- Create a folder called “hackyale” on your Desktop
- In it, create a folder called “weeko” (that’s a zero)
- Open Sublime and open that folder
- Create index.html
- Type “Hello World” and save it
- Open index.html in Google Chrome (and set as default behavior)



# **WELCOME TO WEB DEV 101**

TIME TO GET LEARNED



# **KEY CONCEPTS**

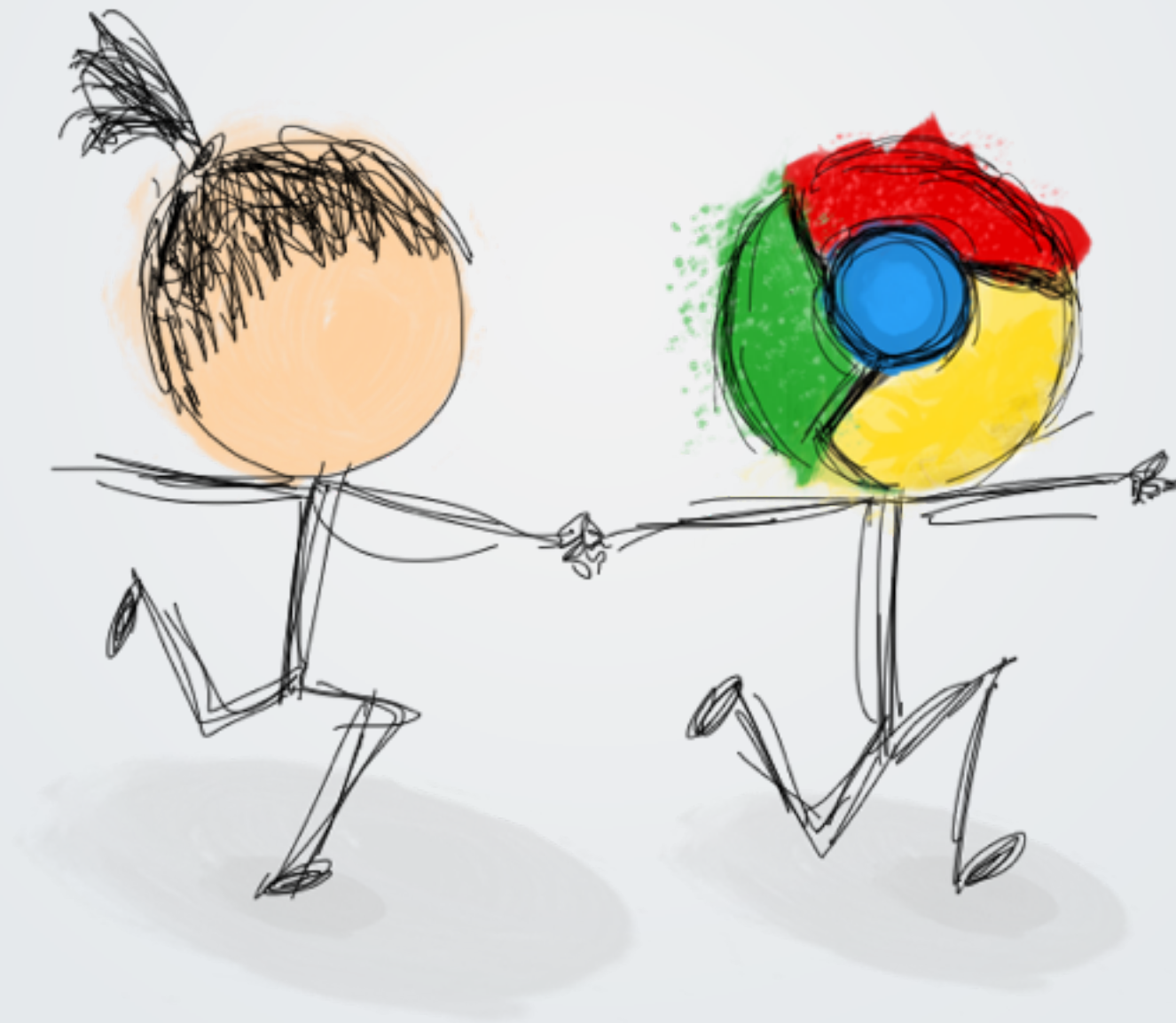
THINGS YOU WANT TO BE AWAKE FOR





# KEY CONCEPT 0

GOOGLE IS YOUR FRIEND



# KEY CONCEPT 0

80% of web development is knowing where to look

Most common answer = Google

- Things to Google:
  - Error messages
  - Syntax
  - Entire problems. Ex: “javascript dropdown menu”
  - Think of whatever question you would ask an expert, and ask it to Google first.

***WHAT DO WE DO  
WHEN WE ENCOUNTER  
A PROBLEM WE CAN'T  
IMMEDIATELY SOLVE?***

GOOGLE GOOGLE GOOGLE GOOGLE GOOGLE  
GOOGLE GOOGLE GOOGLE GOOGLE GOOGLE  
GOOGLE GOOGLE GOOGLE GOOGLE GOOGLE  
GOOGLE GOOGLE GOOGLE GOOGLE GOOGLE  
GOOGLE GOOGLE GOOGLE GOOGLE GOOGLE  
GOOGLE GOOGLE GOOGLE GOOGLE GOOGLE  
GOOGLE GOOGLE GOOGLE GOOGLE GOOGLE  
GOOGLE GOOGLE GOOGLE GOOGLE GOOGLE  
GOOGLE GOOGLE GOOGLE GOOGLE GOOGLE  
GOOGLE GOOGLE GOOGLE GOOGLE GOOGLE  
GOOGLE GOOGLE GOOGLE GOOGLE GOOGLE  
GOOGLE GOOGLE GOOGLE GOOGLE GOOGLE  
GOOGLE GOOGLE GOOGLE GOOGLE GOOGLE  
GOOGLE GOOGLE GOOGLE GOOGLE GOOGLE  
GOOGLE GOOGLE GOOGLE GOOGLE GOOGLE



# ◀ **GOOGLE IS YOUR FRIEND** ▶

BUT WE ARE HERE FOR YOU TOO!

WE'LL GO FAST, BUT THIS IS AN  
INTERACTIVE CLASS.

PLEASE STOP US WHENEVER YOU HAVE A  
QUESTION.

# WE WILL TEACH YOU CONCEPTS

But the implementation is on you!

- The fastest way to learn is practice, practice, practice
  - Making mistakes helps a lot, too
- Please, please, please, follow along examples in class
  - If you're bored, think of how you can make it better
  - If you don't get it, ask for help!

# KEY CONCEPT 2

Code is meant for humans to read

- ▶ *Extremely* important to be clear and concise
- ▶ Rely on conventions
  - ▶ Underscores instead of spaces, start with lower case...
- ▶ Use comments!
  - ▶ Comments are lines that aren't code
  - ▶ For others, and for your future self

# KEY CONCEPT 1

There are two parts to learning to code

- Concept
  - What you can do
- Syntax and implementation
  - How to do it





# **ANATOMY OF A WEBPAGE**

HEAD, SHOULDERS, KNEES AND FOOTER



# WEB DEVELOPMENT

An interaction between three “languages”

- HTML - HyperText Markup Language
- CSS - Cascading Style Sheets
- Javascript - A programming language, NOT related to Java

*To be a successful developer, you'll need to learn all three*

**◀ IMAGINE FOR A SEC... ▶**

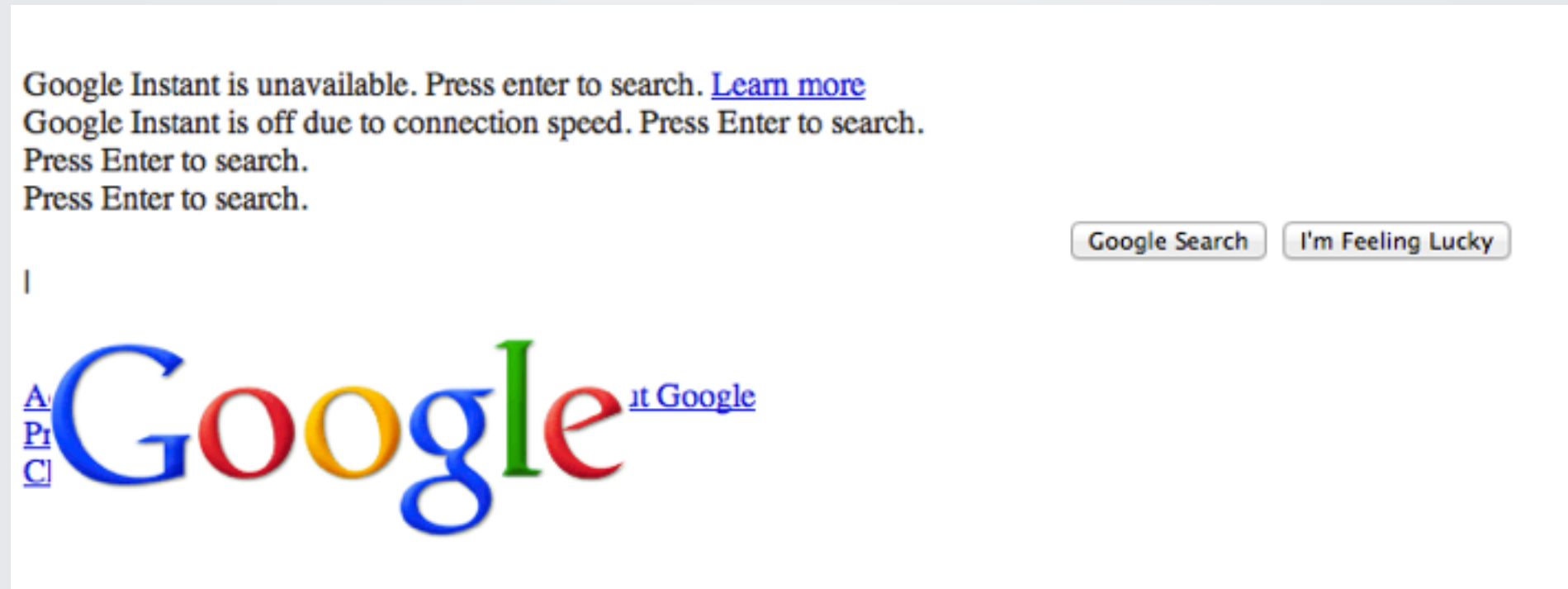
**THAT A WEBSITE IS A HUMAN BODY**

# HTML

## The “bones”

- The “content” of the Internet
- Builds the layout, structure and connections
- Easy to learn, easy to master

# THE HTML OF GOOGLE



IN A WEEK, YOU'LL BE ABLE TO MAKE THIS

# CSS

The “skin” or “physical features”

- The “style” of the Internet
- Defines how HTML elements look
  - Width, height, color, position...
- Easy to learn, difficult to master
- The web designer’s best friend

# JAVASCRIPT

## The “muscles”

- The “interaction” or “animation” of the Internet
- Makes HTML elements interact with one another
  - And with other pages!
- Tougher to learn, and unfortunately, tough to master



# **HTML: THE CONCEPT**

SOME BOARD WORK REQUIRED







# **HTML: IMPLEMENTATION**

TYPE ALONG!

# CODING IN HTML

To program the nested structure of HTML, we use *tags*

- Tags are just 1-4 letters that refer to something
  - Ex. “p” for “paragraph”; “h1” for “header 1”
- To differentiate tags from plain text, we enclose them in brackets
  - <p>, <h1>
  - This *opens* a tag
  - To close a tag, do </h1>

# MORE ABOUT TAGS

---

## Tags delineate content blocks

- `<h1>` *I'm inside a tag!* `</h1>`
  - Tags have “properties,” and these properties are then passed on to the content within the tags
  - Mr. `<tag>` says “abide by my laws until further notice”
  - Then Ms. `</tag>` says “further notice”
-

# TAGS CAN BE NESTED

---

```
<div>  
  <h1>A sick header</h1>  
  <div>  
    <p>My sweet paragraph</p>  
  </div>  
</div>
```

---

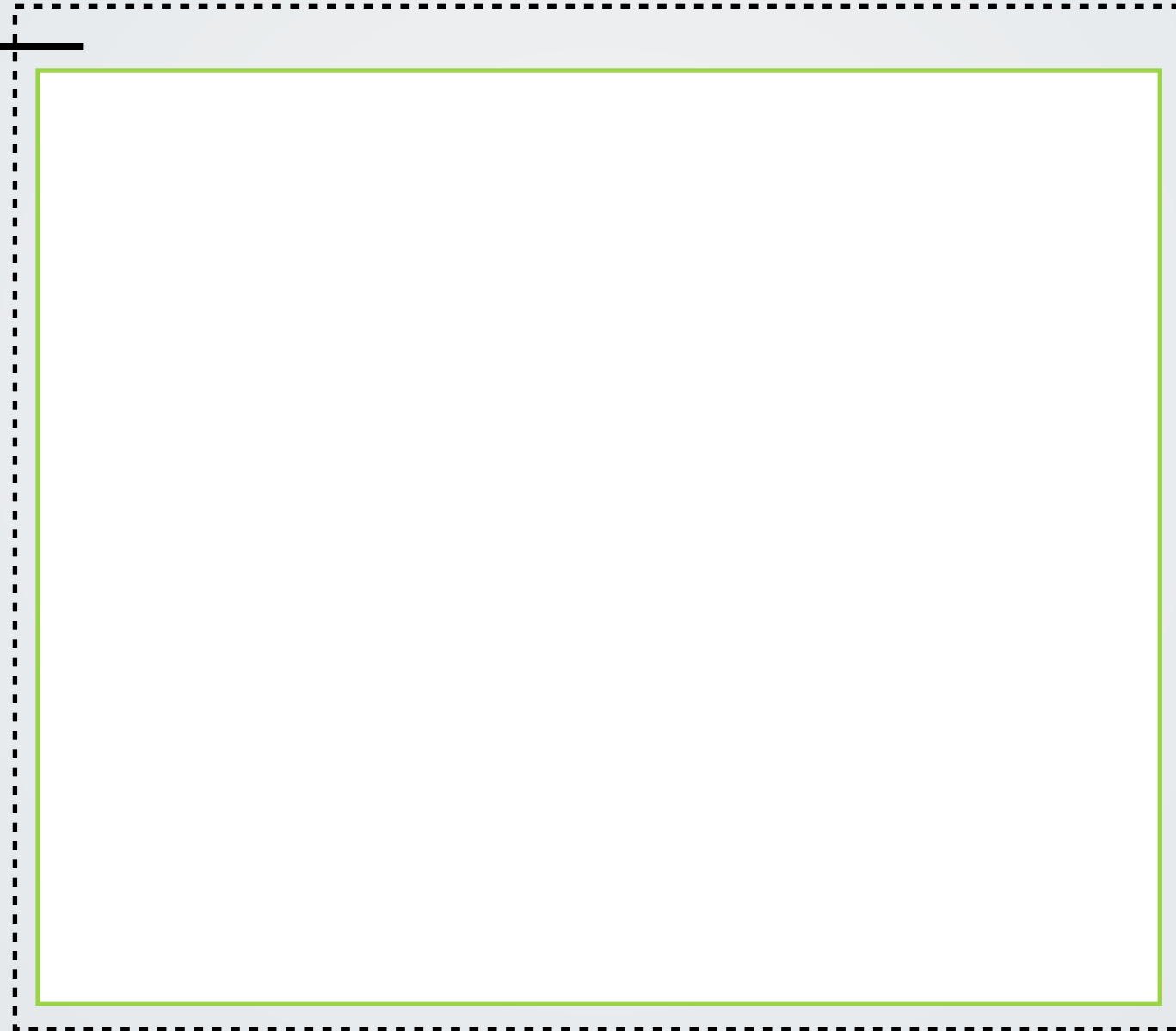
This is how we can create boxes within boxes

# HTML DOCUMENT STRUCTURE

The head



<!DOCTYPE>



<html>

<head>

</head>

<body>

</body>

</html>

# POPULAR TAGS

`<div>` (“division”) the content blocks of HTML (an empty shell)

➤ `<div>This content will be in an div block!</div>`

`<a>` (“anchor”) links

➤ `<a href="http://www.hackyale.com">Home</a>`

`<img>` images

➤ ``

➤ No closing tag—nothing goes *inside* an image...

`<p>` paragraphs

➤ `<p>This text will be in a nice paragraph</p>`

A thick blue diagonal stripe runs from the top right corner towards the bottom left, crossing the center of the slide.

# THE INTERNET

THE LONGEST LINE EVER

**◀ THE VIDEO WAS GREAT ▶**

THANKS, AARON TITUS

HOWEVER, WE'D LIKE TO GET A BIT  
MORE TECHNICAL AND ADD THE  
CONTEXT OF WEB DEVELOPMENT



# TERMINOLOGY

## Client-server model

- ▶ Client == (you and your) browser
- ▶ Server == machine sending (or “serving”) you the data and files you request

HOST ~== “server”

- ▶ “to host” (code, files, applications) ~== “to serve”

LOCAL == hosted on the machine in question

REMOTE == hosted on a different machine



# REQUEST-RESPONSE CYCLE

(1) Client (browser) makes a “request”

- REQUEST == textual message whose syntax and semantics are defined by HyperText Transfer Protocol (*HTTP*)
- Think of a protocol as a “language”

(2) Server issues a “response”

- RESPONSE == textual message defined by HTTP
- Contains status code. Ex: 404 (“*Not Found*”), 200 (“*Okay*”), 500 (“*Internal Server Error*”)

(3) Cycle repeats itself

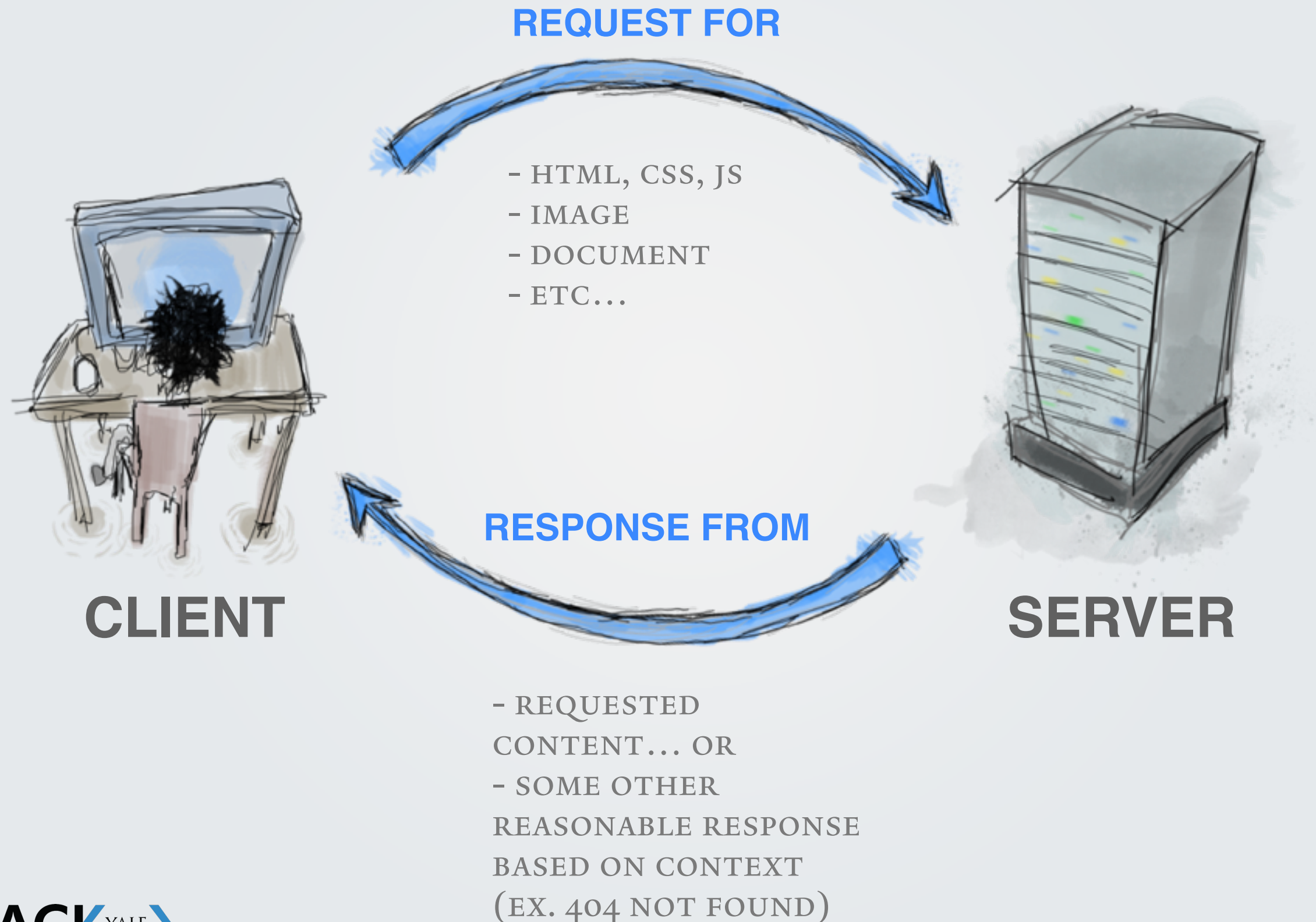


# **KEY TECHNOLOGIES**

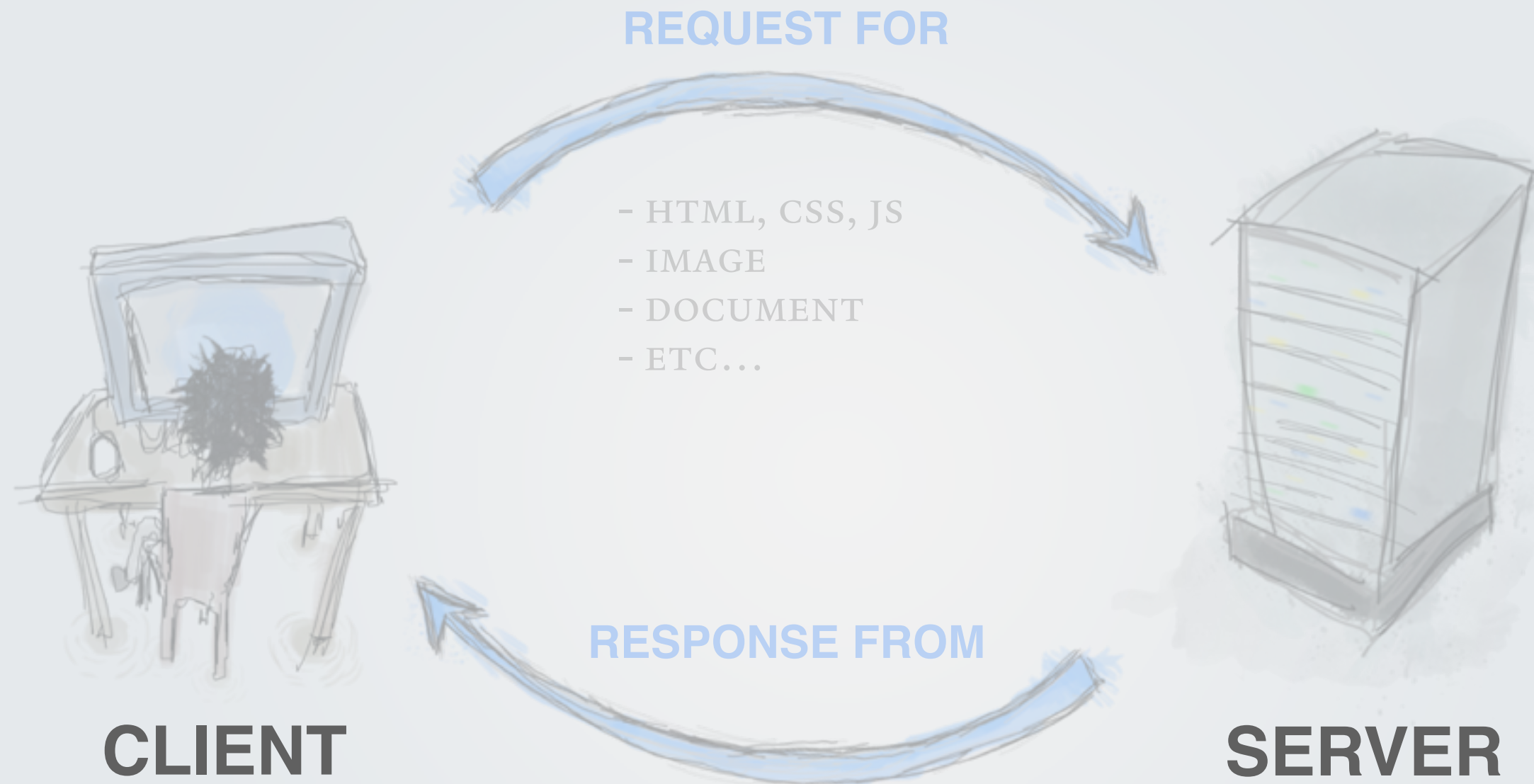
WEB DEVELOPMENT 101



# THE CLIENT-SERVER MODEL



# ON THE CLIENT SIDE



**CLIENT**

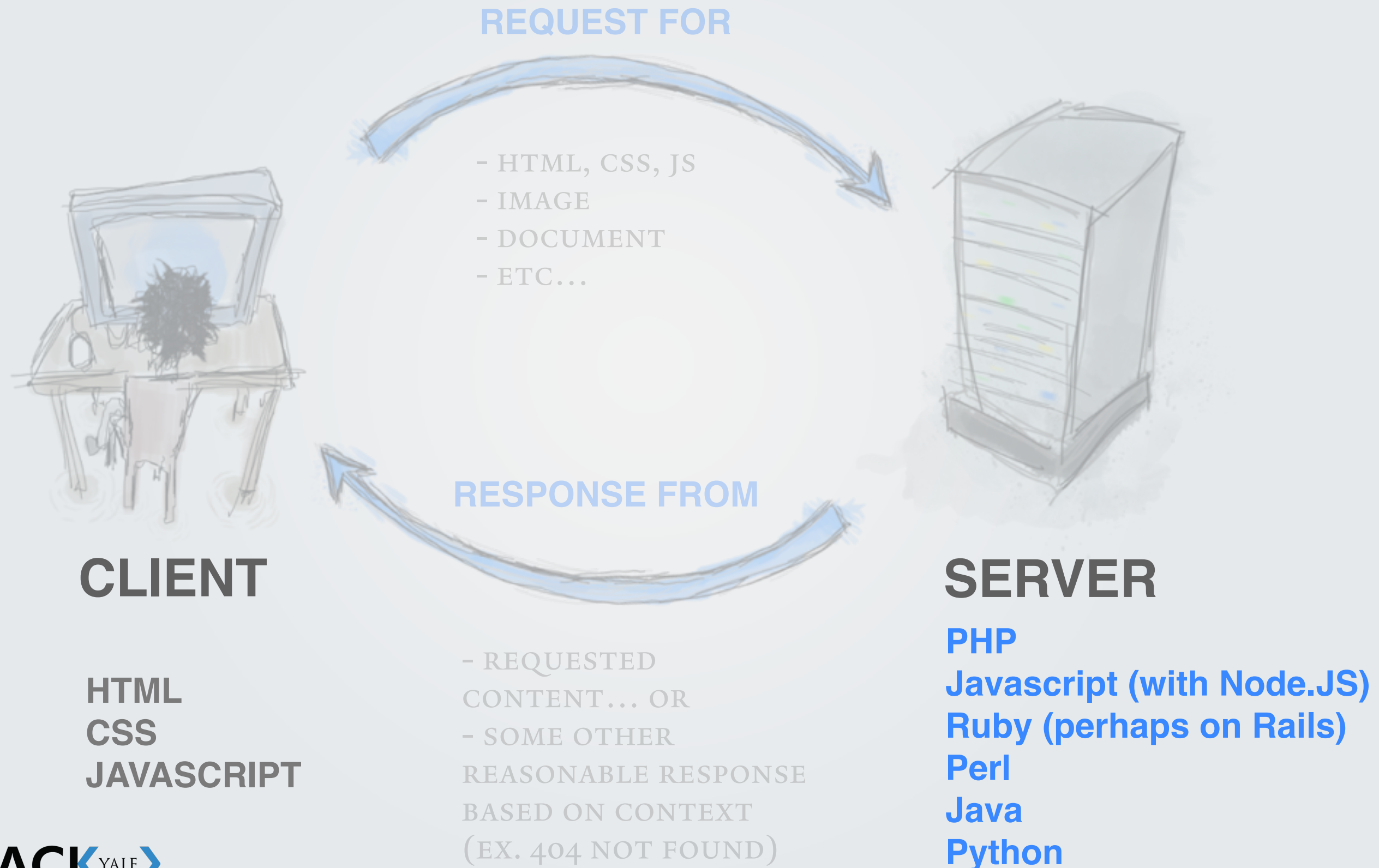
**SERVER**

**HTML**  
**CSS**  
**JAVASCRIPT**

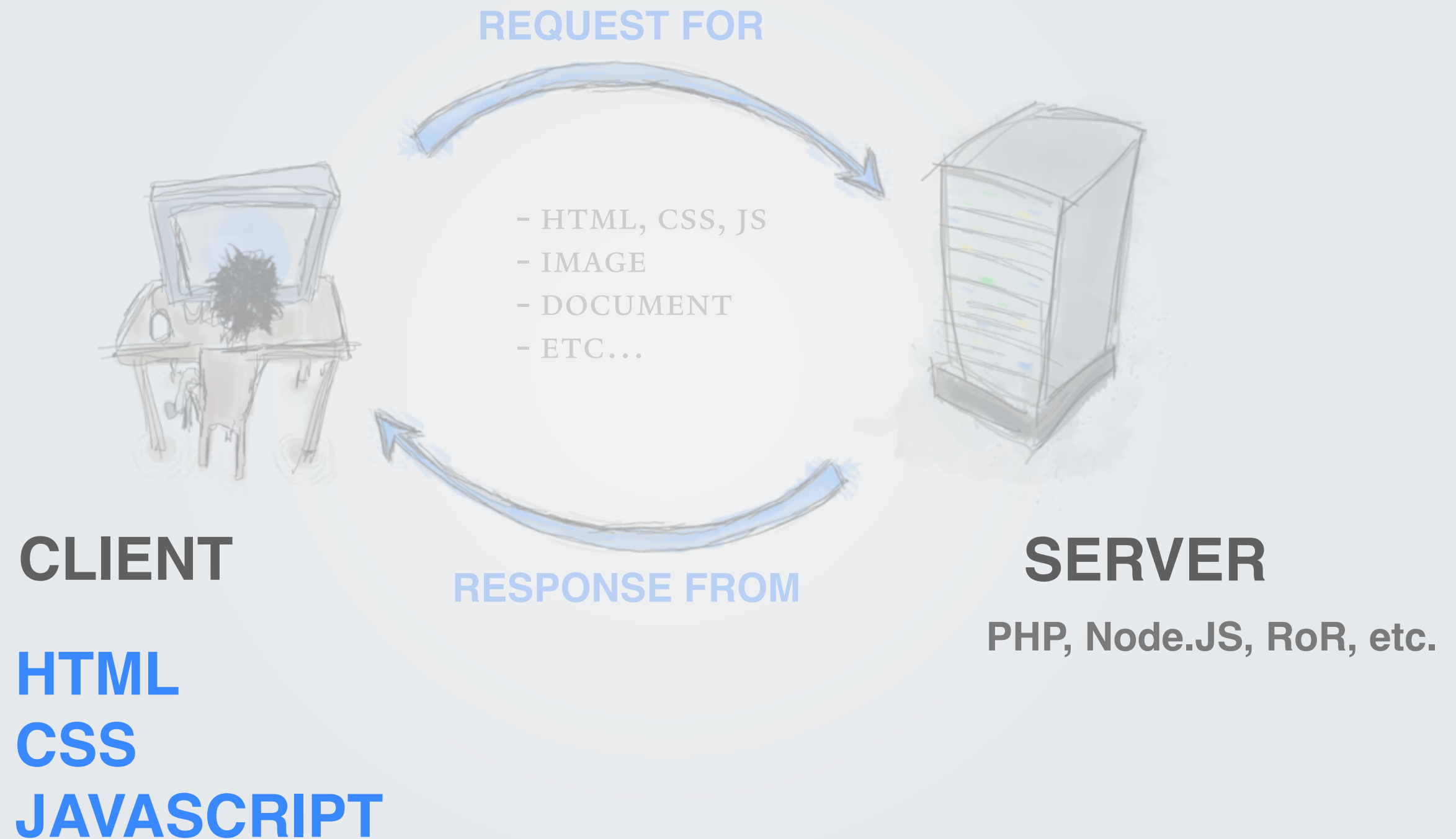
- REQUESTED  
CONTENT... OR  
- SOME OTHER  
REASONABLE RESPONSE  
BASED ON CONTEXT  
(EX. 404 NOT FOUND)



# ON THE SERVER SIDE



# THE FRONT-END IS HOW TO DISPLAY CONTENT FROM THE SERVER





# ◀ **WHOA WHOA, BACK UP** ▶

OKAY, LET'S WALK AND TALK THROUGH WHAT  
GOES ON BEHIND THE SCENES  
WHEN YOU VISIT A WEBSITE LIKE FACEBOOK

A thick blue diagonal stripe runs from the top right corner towards the bottom left, crossing the center of the slide.

# OUR FIRST WEBSITE

AN EXERCISE

# LET'S BUILD OUR FIRST WEBPAGE

There are about 20-25 tags you will need to know

But for today, we'll use:

- !DOCTYPE
- html
- body
- div
- h1, h2, ..., h6
- p
- img
- br
- <!-- ... -->

THAT'S ALREADY 1/2 OF WHAT 'EXPERTS' KNOW!

# ONLY A WEBPAGE? THAT'S SO 1994!

---

So, how about a website?

- ▶ But what is a website? It's just a collection of webpages!

New tag you'll need: `<a>`

---

Okay, that was cool, but my site was still kind of ugly.

# **CASCADING STYLE SHEETS (CSS)**

# ENTER, CSS

```
<p> Hello World! </p>
```

```
<p> Paragraphs are great! </p>
```

```
<p> Totally. </p>
```

Hello World!

Paragraphs are great!

Totally.

Totally.

# INLINE STYLES

```
<p style="color: red;">  
  Hello World!  
</p>
```

```
<p style="color: red;">  
  Paragraphs are great!  
</p>
```

```
<p style="color: red;">  
  Totally.  
</p>
```

Hello World!

Paragraphs are great!

Totally.



# NOW MAKE THEM BLUE

UGH. SO MUCH TYPING!

```
<p style="color: blue;">
```

```
  Hello World!
```

```
</p>
```

```
<p style="color: blu;">
```

```
  Paragraphs are great!
```

```
</p>
```

```
<p style="color: red;">
```

```
  Totally.
```

```
</p>
```

Hello World!

Paragraphs are great!

Totally.

Totally.

# ◀ **INLINE STYLING IS SLOPPY** ▶

CSS LETS US GET **DRY**

DON'T REPEAT YOURSELF, SILLY!

# STYLING IN CSS

---

## Two major benefits

- Cleaner code
    - DRY
  - Flexible code
    - Modular
    - Re-usable
-

# CSS SYNTAX

```
p {  
  font-size: 18px;  
  color: blue;  
}
```

# CSS SYNTAX

```
p {  
  font-size: 18px;  
  color: blue;  
}
```

**SELECTOR**

# CSS SYNTAX

```
p {  
  font-size: 18px;  
  color: blue;  
}
```

SELECTOR

DEFINITION BLOCK

# CSS SYNTAX

```
p {  
  font-size: 18px;  
  color: blue;  
}
```

SELECTOR

DEFINITION BLOCK

PROPERTY

# CSS SYNTAX

```
p {  
  font-size: 18px;  
  color: blue;  
}
```

SELECTOR

DEFINITION BLOCK

PROPERTY

VALUE



# CSS SYNTAX

```
p {  
  font-size: 18px;  
  color: blue;  
}
```

SELECTOR

DEFINITION BLOCK

PROPERTY

VALUE

END OF DEFINITION

```

1 <html>
2 <head>
3   <title>IsItChickenTendersDay?</title>
4 </head>
5 <body>
6
7   <div id="social_block">
8     <iframe src="//www.facebook.com/plugins/like.php?app_id=2774007989367
9   </div>
10
11  <div id="wrapper">
12    <div id="main_text">Yes</div>
13    <div id="sub_text">Get your chicken tenders on.</div>
14
15    <div id="signup_block">
16      Never miss tenders again - we'll text you!<br />
17      <input id="phone" placeholder="(203) 612-5555" />
18      <div id="submit">Ok</div>
19      <br />
20    </div>
21
22    <div id="thanks_block">
23      Ok, I promise you will never cry again.
24    </div>
25  </div>
26 </body>
27 </html>

```

No

But you've still got [options](#).

Never miss tenders again - we'll text you!

Ok

Ok, I promise you will never cry again.

```
1  ▼ img {  
2    border: none;  
3  ▲ }  
4  
5  ▼ a {  
6    color: #999;  
7  ▲ }  
8  
9  ▼ .clear {  
10   padding: 0;  
11   margin: 0;  
12 ▲ }  
13  
14 ▼ #social_block {  
15   position: fixed;  
16   top: 12px;  
17   right: 0px;  
18 ▲ }  
19  
20 ▼ #wrapper #main_text {  
21   color: #477EB4;  
22   font-family: Georgia;  
23   font-size: 140px;  
24 ▲ }  
25
```

# No

But you've still got options.

Never miss tenders again - we'll text you!

# HOMework

Time to practice what you've learned today!

- Make a simple profile website (yep, linked pages!)
- 2 pages: a biography and a page of some of your favorite sites
- Don't worry about styling, unless you want to experiment

Also, do:

- Learn Command Line the Hard Way (<http://cli.learncodethehardway.org/book/>)
- Practical Guide to HTML (<http://learn.shayhowe.com/>)

# NEXT TIME

## More HTML

- History and the new future
- Attributes
- Other tags (forms, lists, ...)

## A lot more CSS

- CSS Selectors with id and class attributes
- Commonly used styles
- Dynamic (or pseudo) selectors

More concrete exercises!



QUESTIONS EVEN GOOGLE CANT ANSWER?

[TEAM@HACKYALE.COM](mailto:TEAM@HACKYALE.COM)

[WWW.HACKYALE.COM](http://WWW.HACKYALE.COM)