



< INTRODUCTION />

WWW.HACKYALE.COM

WELCOME!

The Agenda

- HackYale
- The Internet
- Rapid fire HTML, CSS, and JavaScript
- Teaching lessons
- Q&A



THIS TALK

<http://hackyale.com/pdfs/introduction.pdf>

INTRODUCTIONS

ZACK RENEAU-WEDEEN

- HackYale director, four-time instructor.
- Senior, Trumbull, Computer Science / Econ major
- Worked at Google, Brewster
- Enjoys “PG-rated movies starring Martin Lawrence”



HACKYALE

THROUGH THE AGES

HACKYALE >

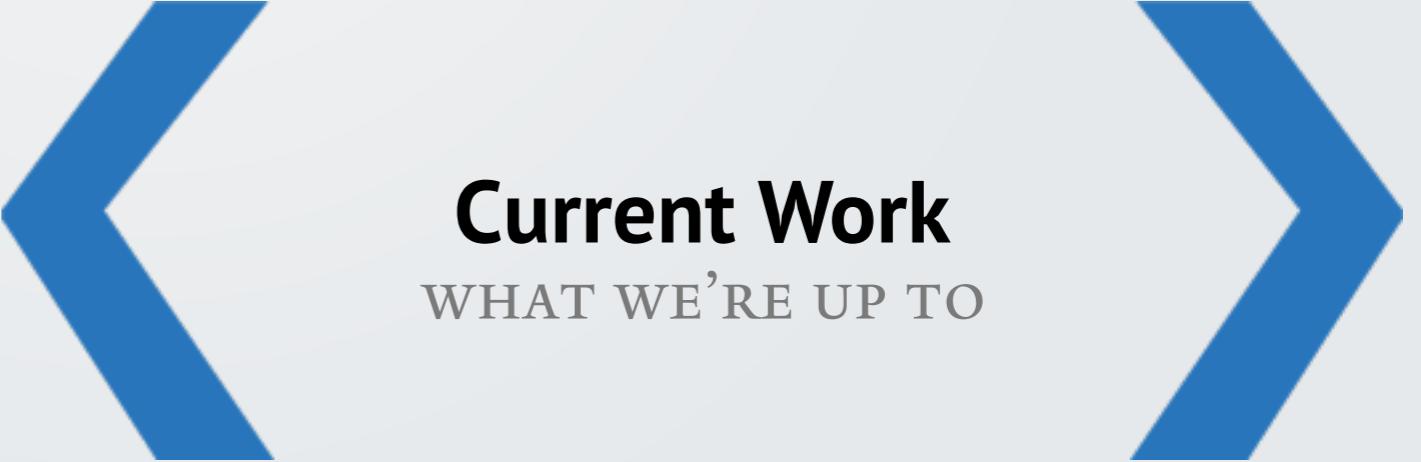
HACKYALE

“

*HackYale provides student-run lectures
in web development, introductory
programming and design.”*

CHRONOLOGY

- Huge demand from the start
 - Considered expansion
 - Settled in as a provider of educational, cultural, and professional resources to Yale students.
-



Current Work

WHAT WE'RE UP TO

CURRENT WORK

- Flagship lecture
 - Workshops
 - Collaboration
 - Jobs
-

LECTURE

- Once a week + office hours
 - Homeworks
-

SHOWCASE

- Yale Film Society
 - Friends With Wheels
 - Inta
 - Yale Taxi Stand
 - Screw Me Yale
 - Yale Travelogue
 - Etc.
-



The Future

WHAT'S IN STORE



THE FUTURE

Incremental Change

- Meeting demand
 - Jobs and opportunities
 - Closer partnership with STC, CEID, etc.
 - Cultural presence (e.g. blog)
 - Moving into the physical realm
-

THE FUTURE

Moon Shots

- HackYale Outreach
 - Hack University
-

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



HACK YALE >

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*”)

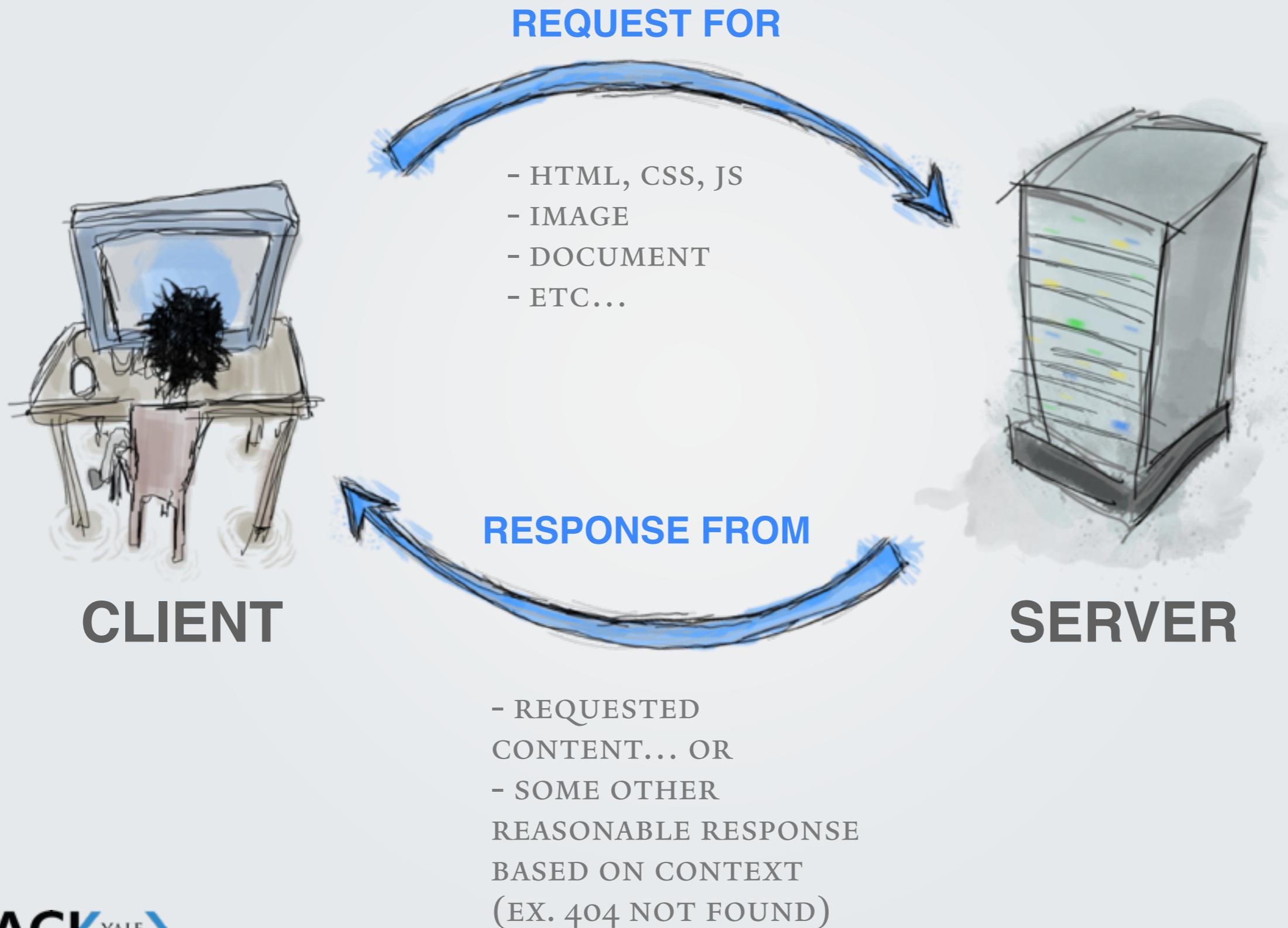
(3) Cycle repeats itself



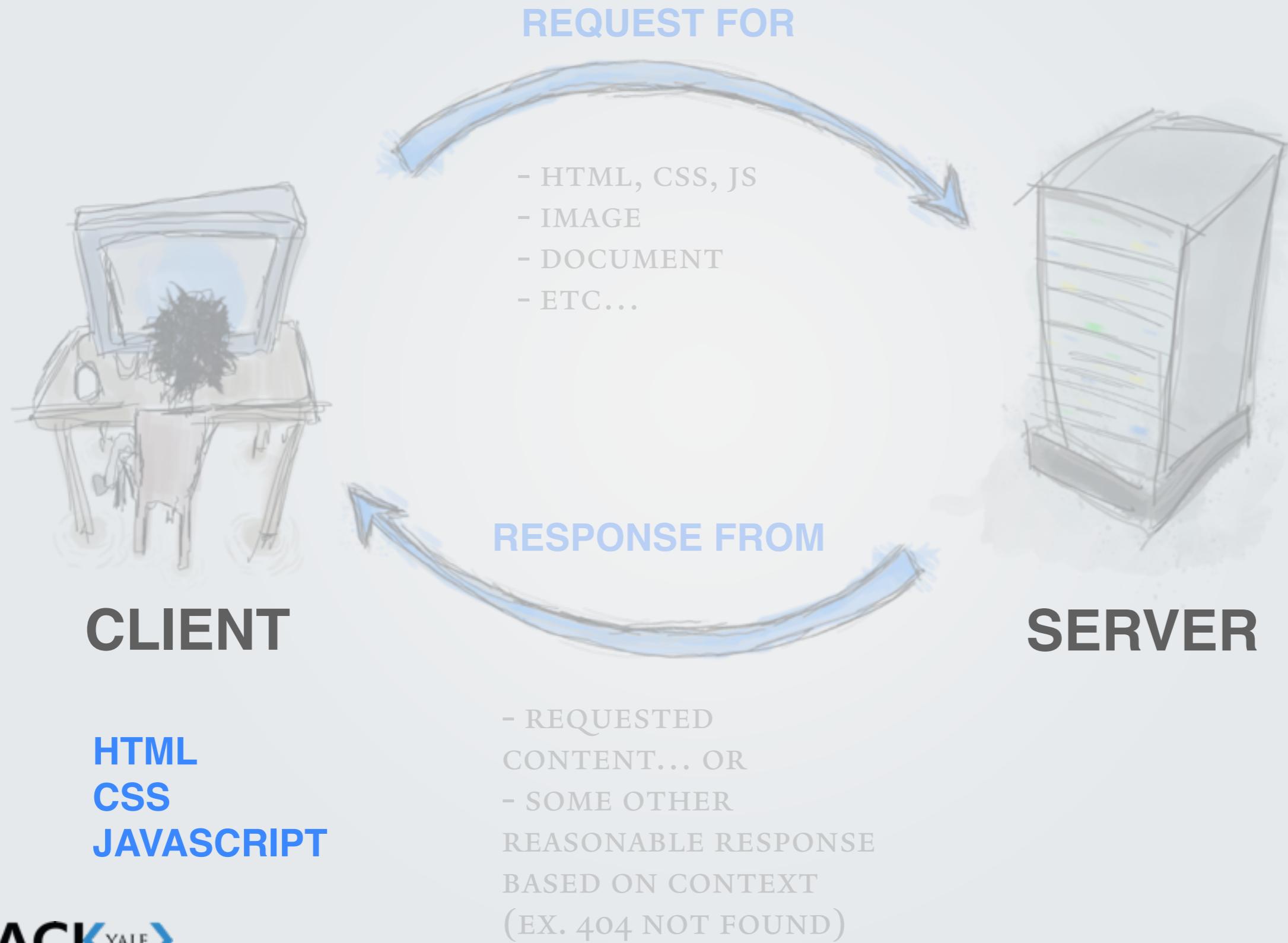
KEY TECHNOLOGIES

WEB DEVELOPMENT 101

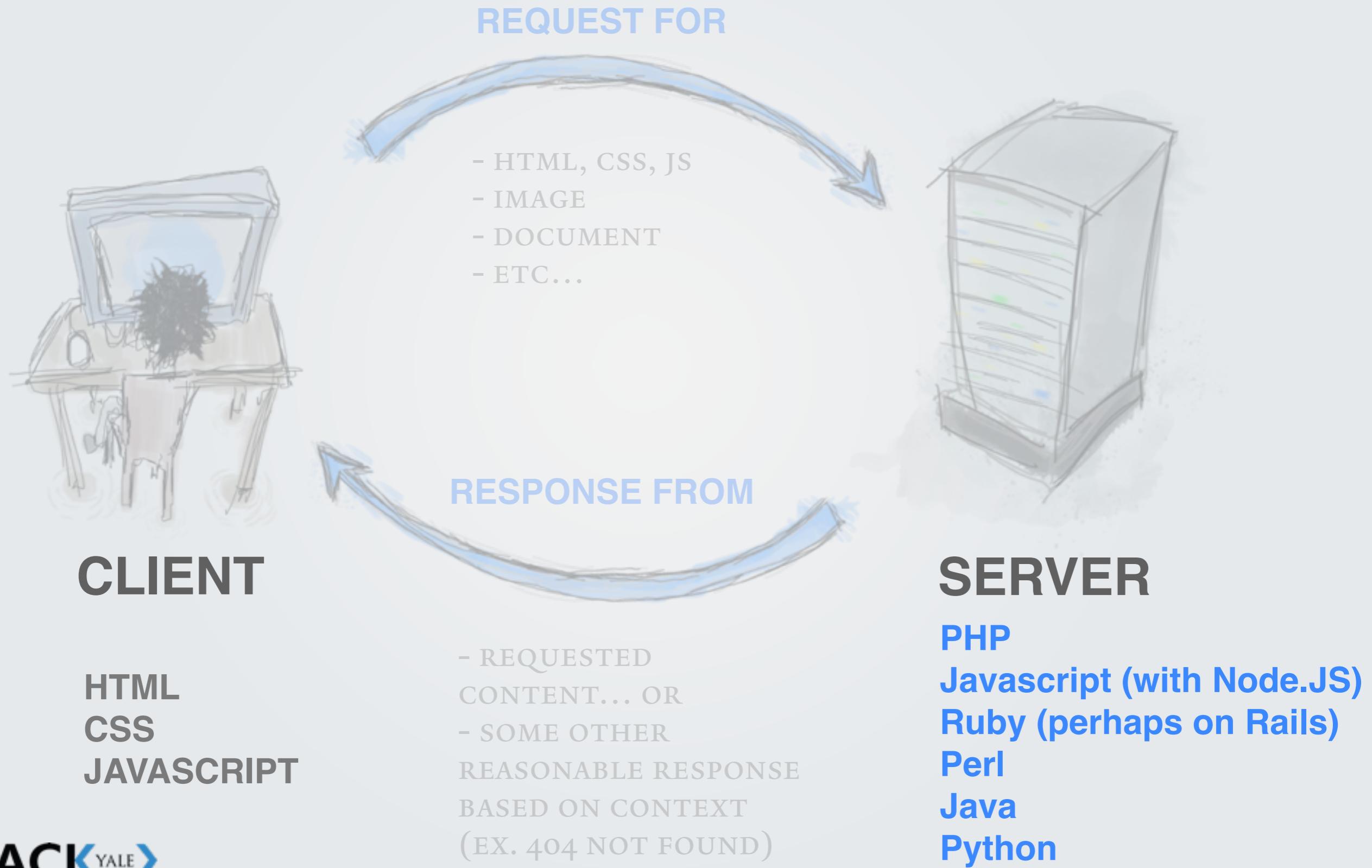
THE CLIENT-SERVER MODEL



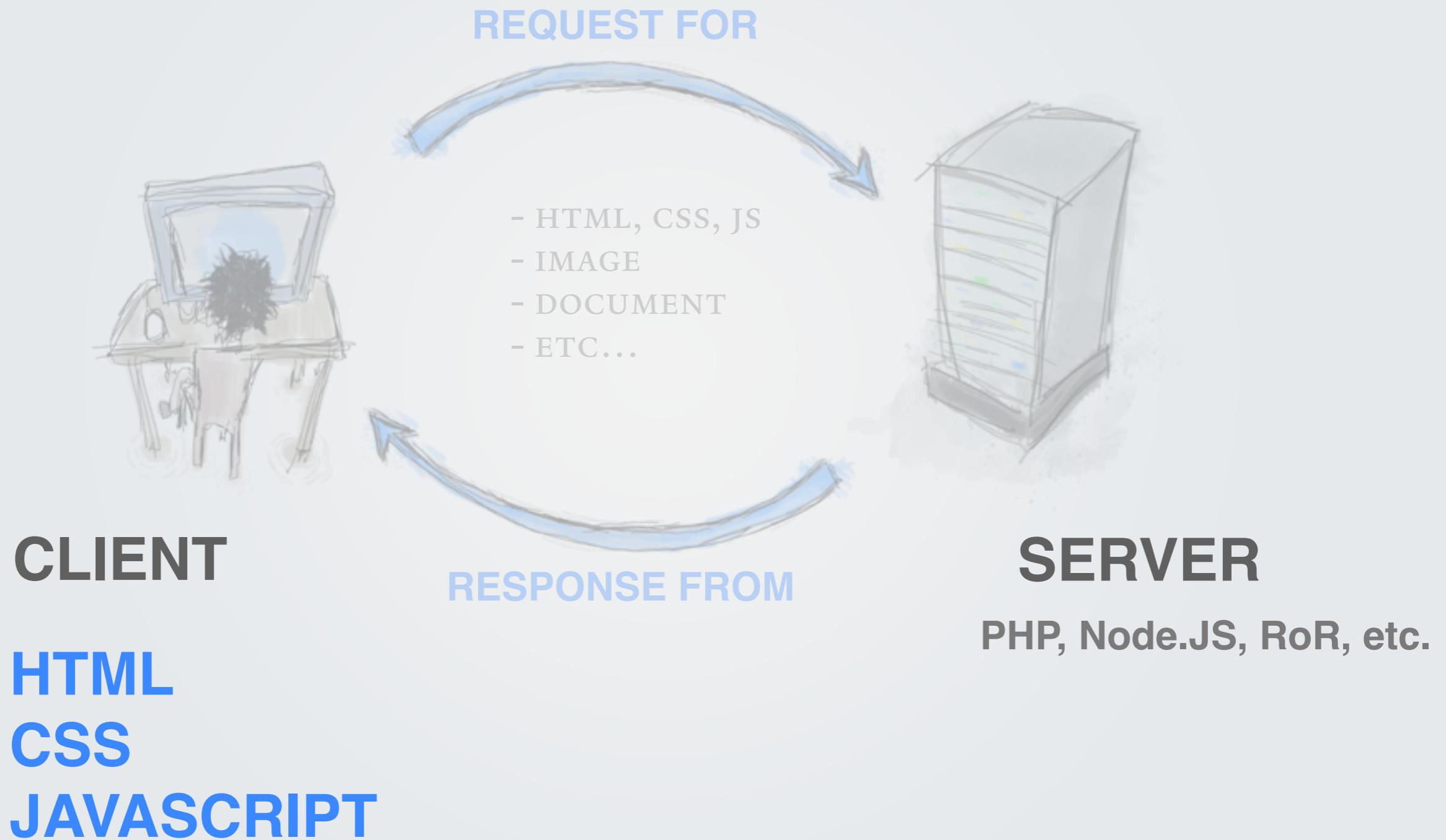
ON THE CLIENT SIDE



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**

OUR FIRST WEBSITE

AN EXERCISE



FRONT-END DEVELOPMENT

An interaction between three “languages”

- HTML - HyperText Markup Language
- CSS - Cascading Style Sheets
- Javascript - A programming language, NOT 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”

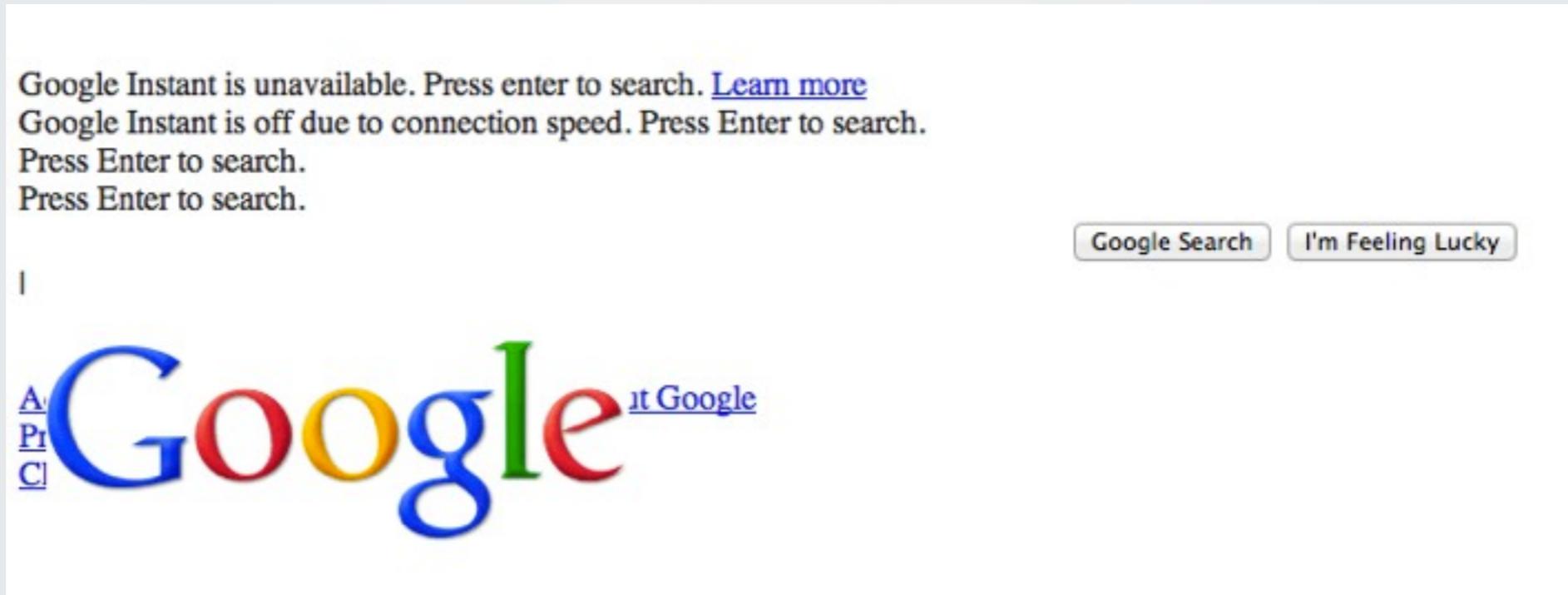
- The “content” of the Internet
- Builds the layout, structure and connections
- Easy to learn, easy to master

THE HTML OF GOOGLE

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



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: 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

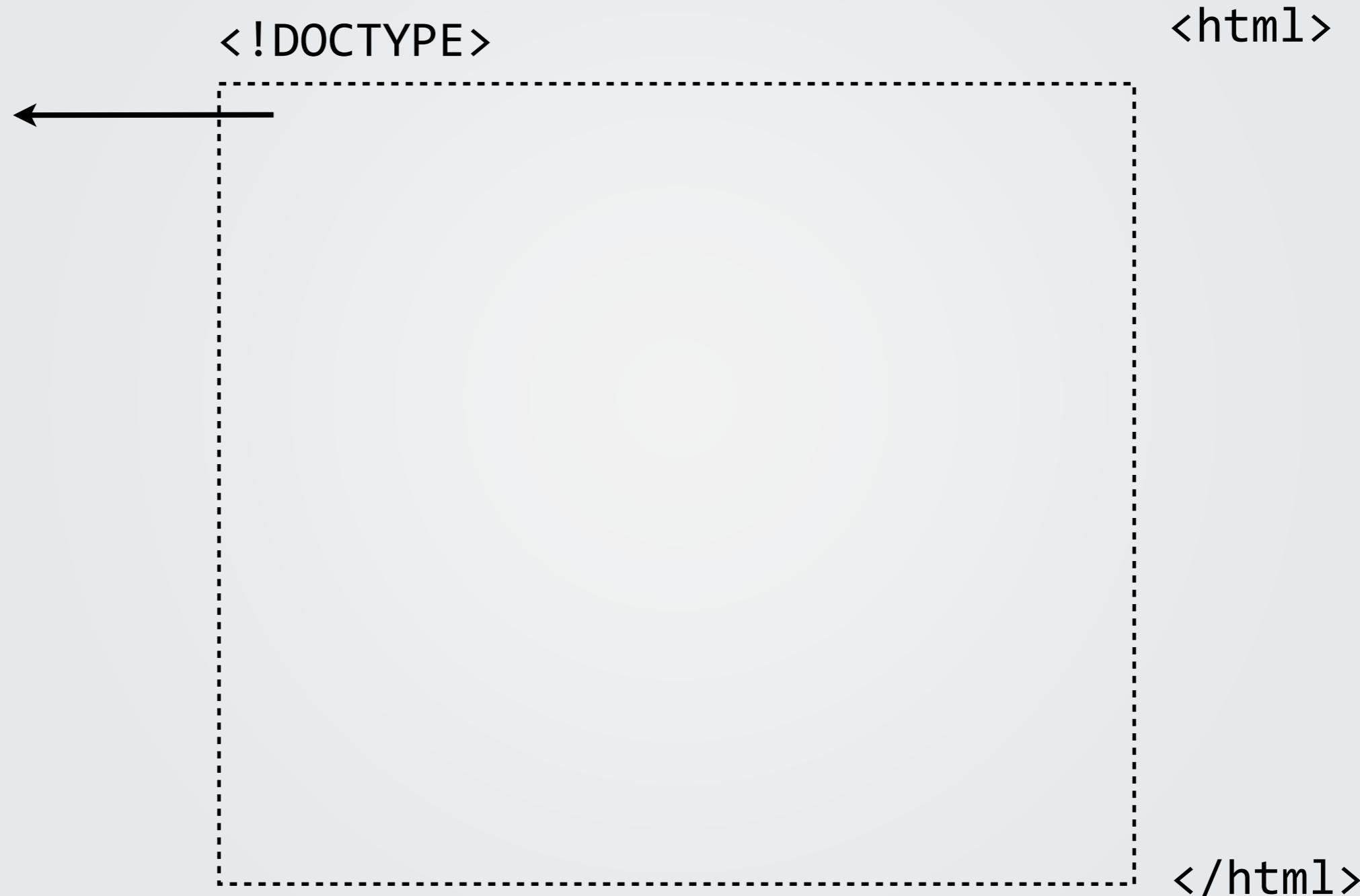


HTML DOCUMENT STRUCTURE

<!DOCTYPE>



HTML DOCUMENT STRUCTURE



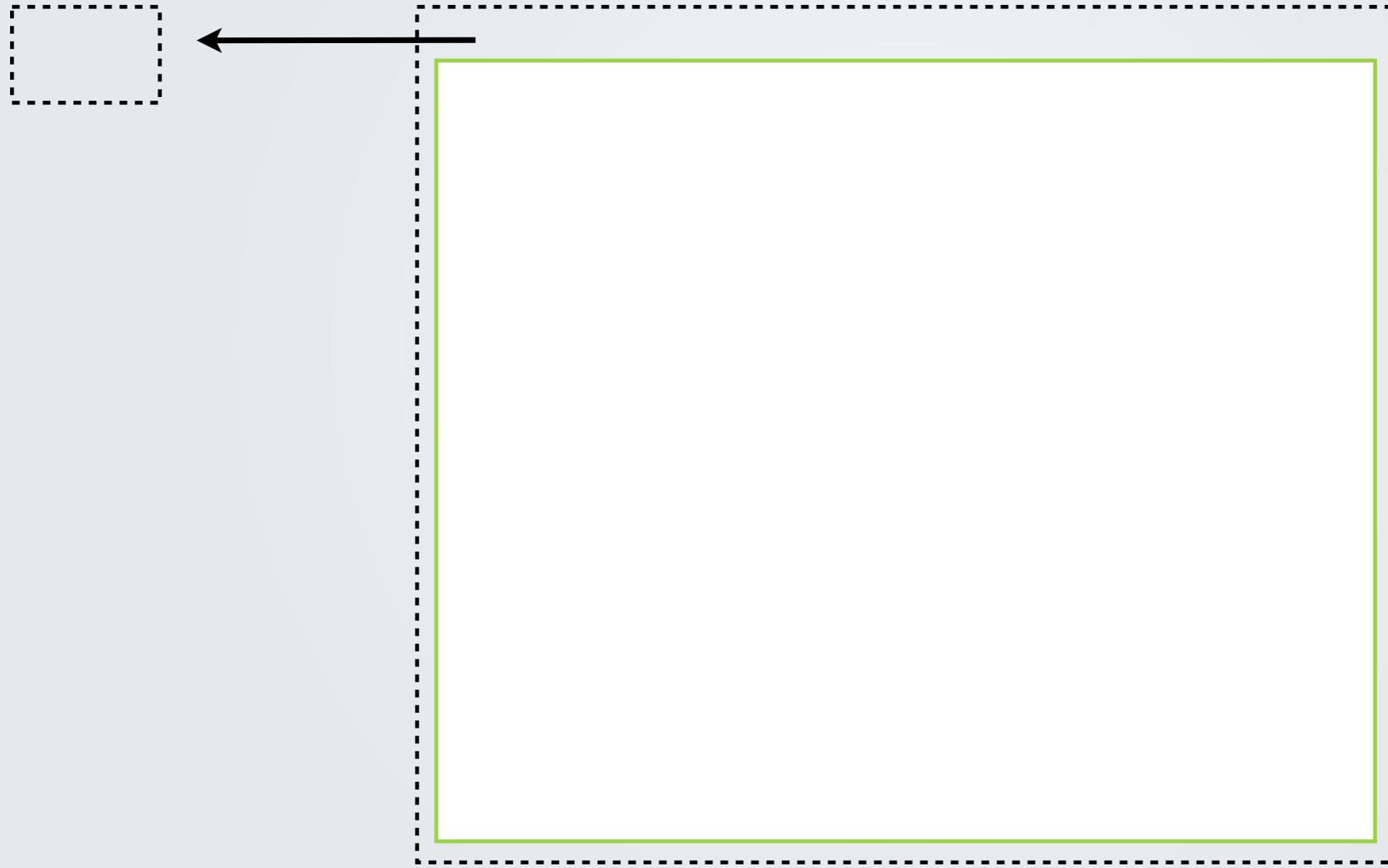
HTML DOCUMENT STRUCTURE



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

➤ Home

 images

➤

<p> paragraphs

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

CASCADING STYLE SHEETS (CSS)

SO YOU'VE BUILT A SITE...

But it's ugly. Not to be mean, but by that I mean:

- Aside from images, not much color
- Not much control of where to put the “boxes”
- Spacing and alignment may not be ideal
- Not too responsive: hovers and clicks don’t do anything

THAT'S WHY WE HAVE CSS!

Style specifies how your HTML elements look

- We want to **select** some elements
- And apply a **style** to them
 - like color, size, alignment...

WAYS TO SELECT ELEMENTS

By tag name:

- Very easy, just `p`, `img`, `div` or what have you
- Very rarely used, and often to define a global ‘feel’ for the site
 - For example, make all links red instead of the default blue
 - Make all images have a border
- But what happens when you don’t want *all* of the `p` tags?

WAYS TO SELECT ELEMENTS

By class name:

- A **class** is an attribute given to one or more HTML elements
 - `<p class="blue-text">Some text</p>`
- Many elements may have the same class
- An element may have many classes
 - Just separate with spaces!
 - `<p class="blue underline">More text</p>`

WAYS TO SELECT ELEMENTS

By ID:

- An ID is an attribute given to **only one** element
 - `<p id="my-only-red-paragraph">ELM0</p>`
- Only one element may have a particular ID
- An element may only have one ID

Why IDs and not classes?

PRACTICAL USE CASES

Tag name

- To override defaults or define global styles

Class

- The most widely used, allows for nice element selection

ID

- HackYale discourages using IDs in CSS selectors
 - Some other people do too

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

JAVASCRIPT

THE CONCEPT

A PROGRAMMING LANGUAGE

But what does that mean?

Why do we program or write code?

To what end?



WHAT IS PROGRAMMING?

PROGRAMMING IS THE PROCESS OF
WRITING *functions* THAT TELL COMPUTERS
TO DO WHAT WE TELL THEM TO DO



FUNCTIONS

A FUNCTION IS A SET OF INSTRUCTIONS
THAT TAKE SOME INPUTS AND PRODUCE A
PREDICTABLE OUTPUT

EXACTLY LIKE IN ALGEBRA!

CONCEPT

By defining and using functions, we can tell computers things we want to do. For example...

- Double a number
- Make all text on a web page double its current size
- Make an image change when a user clicks on it
- ...everything a computer does is implemented by a function

VARIABLES

Sometimes, we want functions to have different outputs when we have different inputs.

- In algebra, you *solve* for a variable.
- But in programming, you get to assign variables values so that we can use those values later.



VARIABLES AND FUNCTIONS

IF YOU UNDERSTAND THESE TWO CONCEPTS,
YOU'VE CLEARED A MAJOR HURDLE IN
UNDERSTANDING PROGRAMMING

JAVASCRIPT SYNTAX



FUNCTIONS

In algebra, we write $f(x) = \dots$

- f is the name of the function
- x is the input
- ... denotes the actual instruction of the function (ex. $x+1$)
- $f(x)$ is shorthand for the output (sometimes called y)
- I hope this sounds familiar.

FUNCTIONS

In Javascript we write $f = \text{function}(x) \{ \dots \};$

- f is the name of the function
- x is the input
- ... denotes the instructions of the function (ex. **return $x+1;$**)
- $f(x);$ is still the output
- I hope this is still familiar!

VARIABLES

Only kind of like algebra

- But instead of finding the values of variables, we use variables to **store values**.
- `a = 7;`
- `a = a + 7;`
- In algebra, the second statement is impossible. In programming, it *happens all the time*.

JAVASCRIPT

A BRIEF HISTORY

“

Are JavaScript and Java the same thing?

No

*Are JavaScript and JSON the same
in any way?*

THE NAME

In 1995, Java was hot stuff, and the Netscape guys wanted their language to be also.

- So they named it JavaScript
- Supposedly this was in exchange for bundling Sun's Java runtime with their market-leading browser: Netscape Navigator.
- Java Applets (e.g. slimezone.com) later fell out of favor, but JavaScript kept kicking.

“Isn’t Javascript that crappy little toy language that was invented in like twenty minutes by some guy at Netscape?”

“But isn’t Javascript that crappy little toy language that was invented in like twenty minutes by some guy at Netscape?...

“But isn’t Javascript that crappy little toy language that was invented in like twenty minutes by some guy at Netscape?”

Sort of...

but not really anymore.

BRIEF HISTORY

Javascript was hurriedly developed by Brendan Eich at Netscape in 1995

- “Mocha” > “LiveScript” > “JavaScript”
- Submitted to Ecma International for standardization in 1996 > “ECMAScript”
- Iterated, updated, and enhanced since then
- For more on Javascript’s colorful history, give this a read: <http://ask.metafilter.com/195482/Lets-assume-that-I-am-the-stupidest-person-that-ever-lived-Explain-to-me-what-JavaScript-is-what-it-does-and-how-a-moron-would-go-about-learning-it#2813956>

JAVASCRIPT AIN'T PERFECT...

Given the rushed conditions of its development, JavaScript has some downsides

- Too permissive. Doesn't complain
- Type coercion (i.e. == vs. ===)
- Awkward typing
 - `if (typeof x == 'undefined') {...}`

BUT IT AIN'T BAD NEITHER!

JavaScript has its advantages

- Functions as first class objects
- Closures
- Event loop
- Prototype chain
- Tail recursion (in theory)
- Fast (thanks to Google's V8 engine)

BUT IT AIN'T BAD NEITHER!

Appreciation for JavaScript has grown in recent years thanks to several key figures

- Douglas Crockford - YUI, Javascript: The Good Parts
- John Resig - jQuery
- Jeremy Ashkenas - Backbone.js
- Ryan Dahl / Joyent - Node.js

DEMO

FIRE UP THE CONSOLE

CHROME CONSOLE

command + option + j
or
view -> developer -> javascript console



A screenshot of the Google Chrome DevTools interface, specifically the Console tab. The top navigation bar includes tabs for Elements, Resources, Network, Scripts, Timeline, Profiles, Audits, Console (which is selected), and Page Speed. A search bar labeled "Search Console" is also present. The main area displays a JavaScript session:

```
> alert("hey");
undefined
> var answer = prompt("what's your name");
undefined
> alert(answer);
undefined
> answer === "Zack";
true
>
```

The bottom navigation bar includes icons for refresh, search, and other developer tools, along with tabs for All, Errors, Warnings, and Logs.

VARIABLE SYNTAX

```
var a = 7;
```

- var tells the computer that a is variable
- = means to **store**. In this case, the number 7
- All Javascript statements end in a semicolon.

VARIABLE SYNTAX

```
a = a + 7;
```

- Don't need to resay that `a` is a variable.
- `=` means to **store**. In this case, the number `7`
- All Javascript statements end in a semicolon.

OTHER EXAMPLES

We covered a lot of examples in class, that I'll list here.

- `var f = function(x) { return 2 * x };`
- `var numCatz = 7; vs just numCatz = 7;`
- `var numDogz = "7"; and numDogz + numCatz;`
- `typeof numCatz; and typeof f;`
- `var g = function(func) { return func(2); };`
- `var ps = document.getElementsByTagName("p");`

THE CONCEPTS

The concepts covered in the examples (respectively)

- Defining functions
- The use of the keyword ‘var’
- The difference in addition between string and numbers
- Types in Javascript (‘var’ is not a type. A variable only gets a type when it is assigned a value). Functions are a type!
- Functions can be inputs to other functions
- Getting elements on your HTML page (in this case, all paragraphs)

UNDERSTANDING THE CONCEPTS

It is *incredibly* important that you understand all of the previous concepts, even if you have programmed before. Javascript does things a bit differently than C or Java, and when we start talking about more advanced things (like callbacks and the Event Loop), we're going to take these things as a given. Use Google or come to office hours...it's just 6 concepts!

INCLUDING JAVASCRIPT

SIMILAR TO CSS

```
<!DOCTYPE html>
<html>
<head>
    <!-- some HTML -->
</head>
<body>
    <!-- some more HTML -->
    <script src="main.js"></script>
</body>
</html>
```

WHAT ARE THE DIFFERENCES?

```
<!DOCTYPE html>
<html>
<head>
    <!-- some HTML -->
</head>
<body>
    <!-- some more HTML -->
    <script src="main.js"></script>
</body>
</html>
```

CSS

```
<html>
<head>
    <!-- some HTML -->
    <link rel="stylesheet" href="/style.css">
    <!-- some more HTML -->
</head>
<body>
    <!-- even more HTML -->
</body>
</html>
```

JAVASCRIPT

```
<!DOCTYPE html>
<html>
<head>
    <!-- some HTML -->
</head>
<body>
    <!-- some more HTML -->
    <script src="main.js"></script>
</body>
</html>
```

LEARNINGS

FIRE UP THE CONSOLE

STRAIGHTFORWARD

One mistake and it was clear

- Have workshops for esoteric topics, not full classes.
- Focus on the products.
- Be lively.

LESS OBVIOUS

Some non-obvious, subtler lessons

- Have class when people *aren't* working.
- Avoid insecurity.
- Prepare to improvise.

STILL WORKING ON

Some things we haven't quite figured out yet

- Ensuring long-term commitment.
- Biting off the right amount.

TRY IT OUT

<http://hackyale-intro.herokuapp.com>



HOMEWORK

YOU THOUGHT WE WERE DONE

HOMEWORK

Begin your education

- Complete project 1 at <https://dash.generalassemb.ly>

THANKS!



QUESTIONS? ZACK@HACKYALE.COM

WWW.HACKYALE.COM