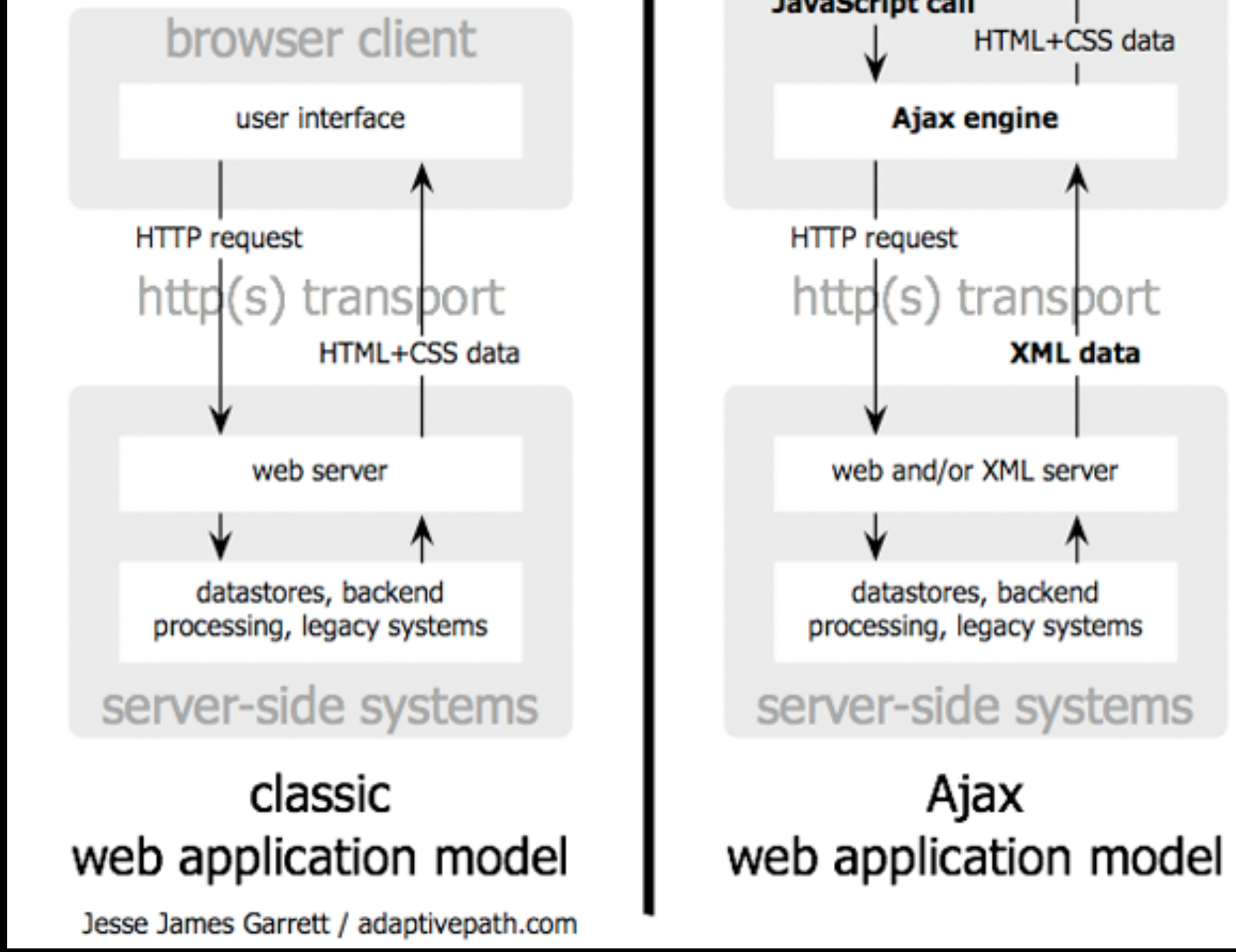
Intro 2 JS

The collection of technologies used by Google was christened **AJAX**, in a seminal essay by Jesse James Garrett of web design firm Adaptive Path.

"**Ajax** isn't a technology. It's really several technologies, each flourishing in its own right, coming together in powerful new ways.

Ajax incorporates:
- standards-based presentation using **XHTML** and **CSS**;
- dynamic display and interaction using the Document Object Model (DOM);
- data interchange and manipulation using XML and XSLT;
- asynchronous data retrieval using XMLHttpRequest;
- + **JavaScript** binding everything together."



Jesse James Garrett, 2005

(To view this link you'll need the Wayback Machine)

Server Side Programming

Unitl now, we have been learning about *client side* web development. The client side, in this case, is the web browser and the technologies that go into it (HTML, CSS, JavaScript).

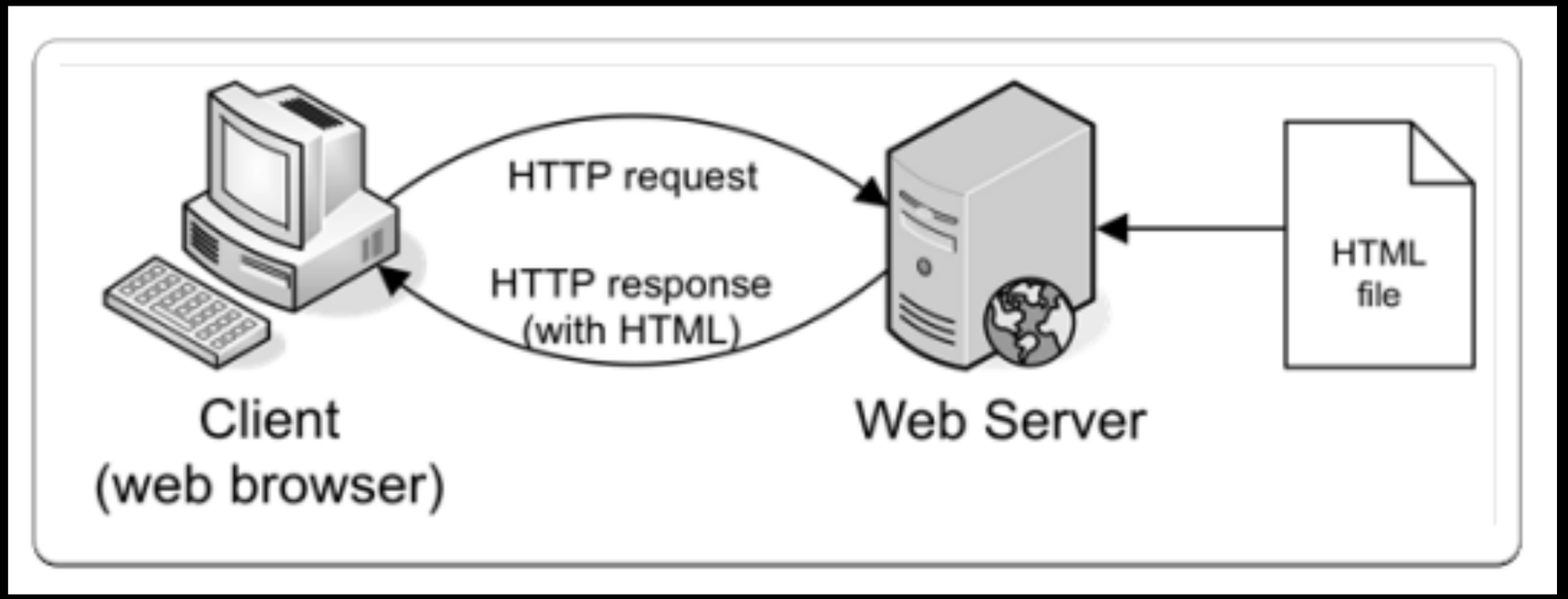
Static vs. Dynamic Web Pages

Server side programming languages (e.x. JavaScript, PHP, .NET, Python, Perl) make creating dynamic web pages possible. Using HTML and CSS we have been creating static web pages in this class. Static web pages will always contain the same content or information in them no matter what, until you change the source code of the page. Dynamic web pages are capable of producing different content for a web page or web site using the same source code, based on the application logic written by the developer. (RSS feeds were one of the first instances of this w/ **Web 2.0**).

Dynamic web pages will often be powered by a server side programming language and a database (e.x. JavaScript, MySQL, SQL Server, Oracle). Together, these technologies can determine what information should be returned to the browser and construct dynamic content for the client side to present to the user.

Request-Response Procedure

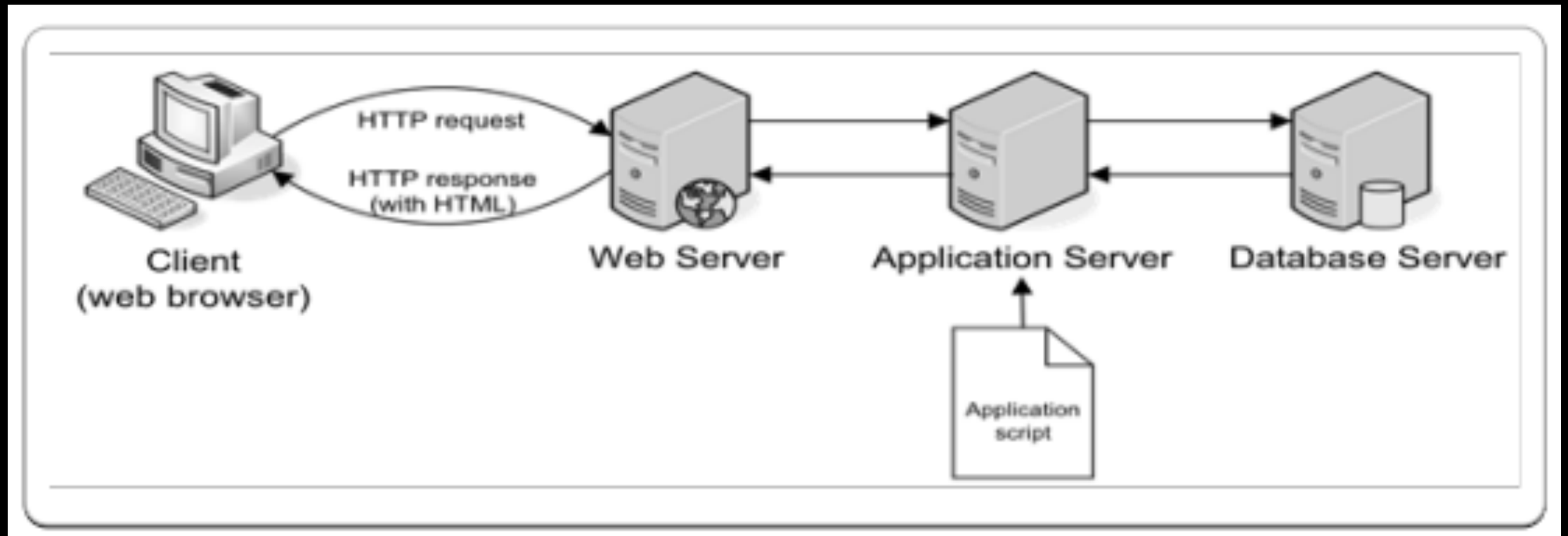
The HTTP request-response procedure explains how you are able to access web sites over the internet. When working with regular client side technologies, the request-response cycle is pretty simple.



1. You enter a website address http://example.com into the browser's address bar.
2. The browser looks up the address for http://example.com and requests the web page associated with it.
3. The request traverses the internet for the server that lives at http://example.com
4. The server at http://example.com receives the request and sends the (HTML) web page for http://example.com.
5. The browser receives the web page sent from the server and shows it to the user.

Server Side

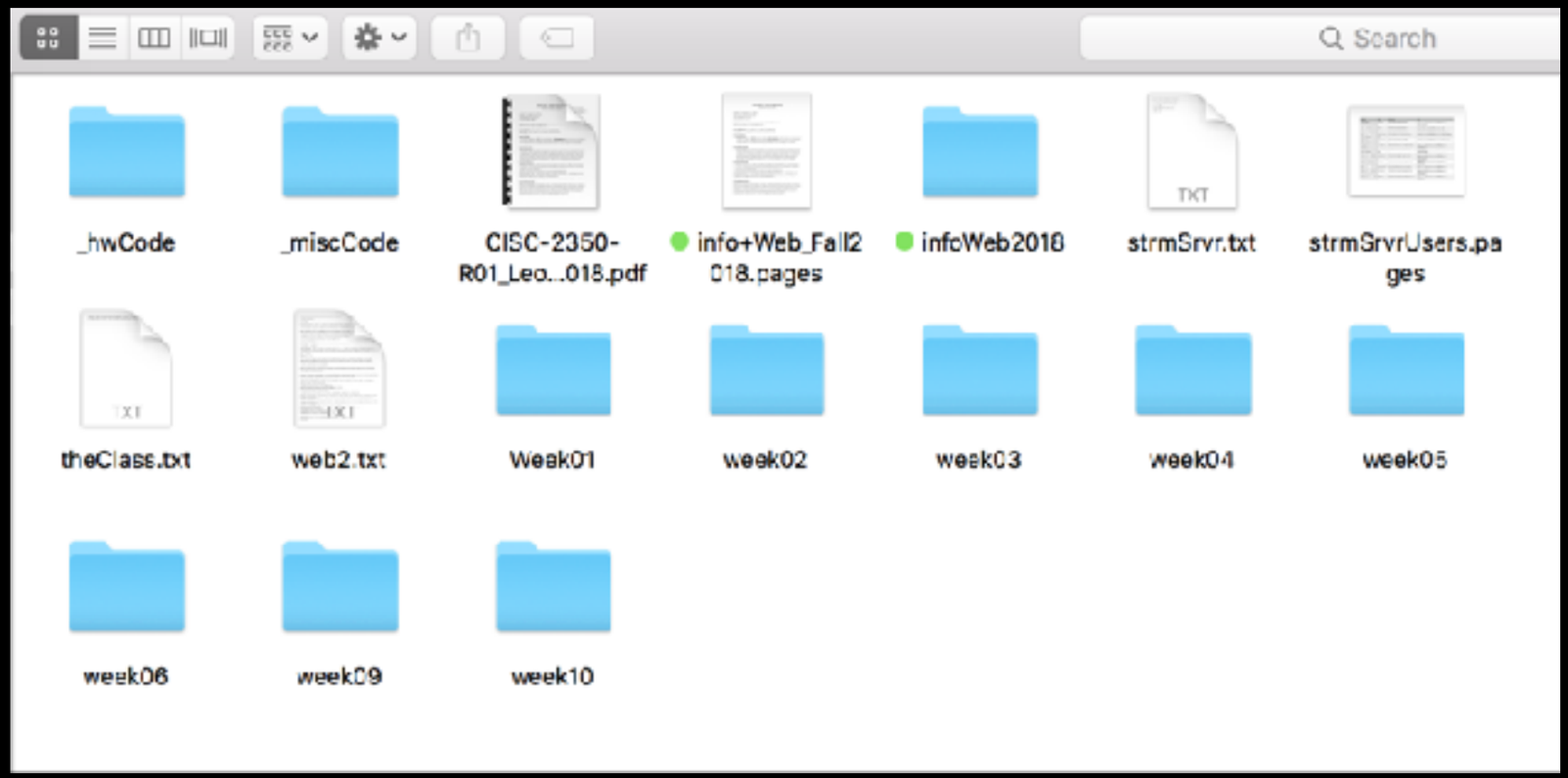
1. You enter a website address http://example.com into the browser's address bar.
2. The browser looks up the address for http://example.com and requests the web page associated with it.
3. The request traverses the internet for the server that lives at http://example.com.
4. The server at http://example.com receives the request and interprets it as a PHP file.
5. The PHP interpreter executes the PHP code and passes any MySQL statements to the database to be processed.
6. The PHP interpreter and MySQL return the resulting data to the web server as HTML.
7. The web server sends the generated page to the browser, which displays it to the user.



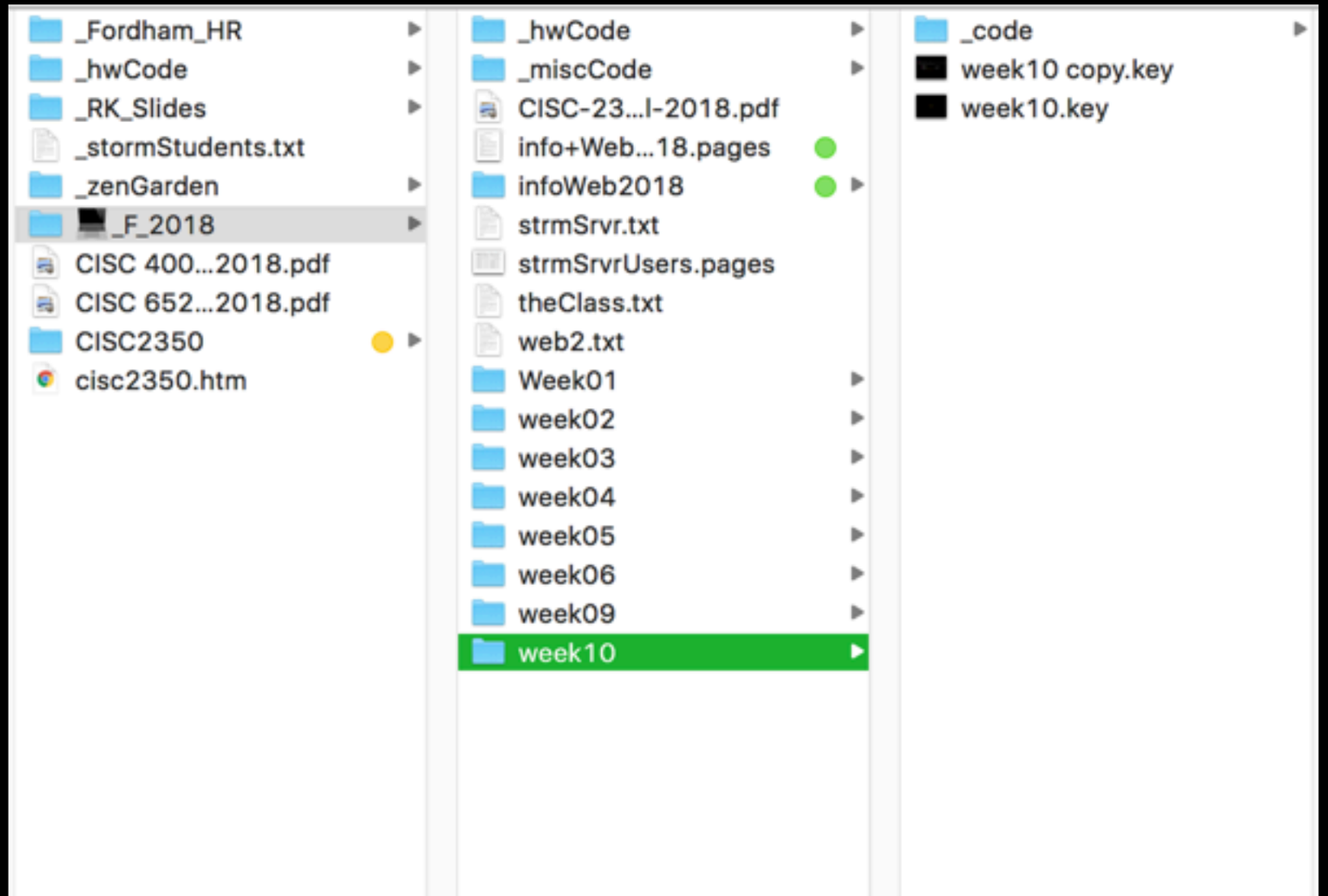
“Think like a computer”

Computers solve problems programmatically; they follow a series of instructions, one after another. The type of instructions they need are often different from the type of instructions you might give to another human.

It’s as simple as the difference btw this



+ this



Computers create models of the world using data.

Object-oriented programming

Objects (Things)

In computer programming, each physical thing in the world can be presented as an object. Each object can have its own:

Properties
Events
Methods

Properties

Each property has a **name** and a **value** + each of these name/value pairs tells you something about each individual instance of that object

Events

Programs are designed to do different thing when users interact with the computer in different ways. For example, clicking on a link could bring up another webpage.

An event is a the computer's way of sticking up its hadn't to say, "hey that just happened."

Programmers choose which events they respond to. When a specific event happens, that event can be use to trigger a speck section of the code.

Methods

Methods represent things people need to do with object. They can retrieve or update the values of an object's properties.

They are questions + instructions that:
- Tell you something about the object (using info stored in its properties)
- Change the value of one or more of the object's properties

The code for a method can contain lots of instructions that together represent one task (ex: changeSpeed()).

When you use a method, you do not always need to know **how** it achieves its task; you just need to know **how to ask the question** and **how to interpret any answers it gives you**.

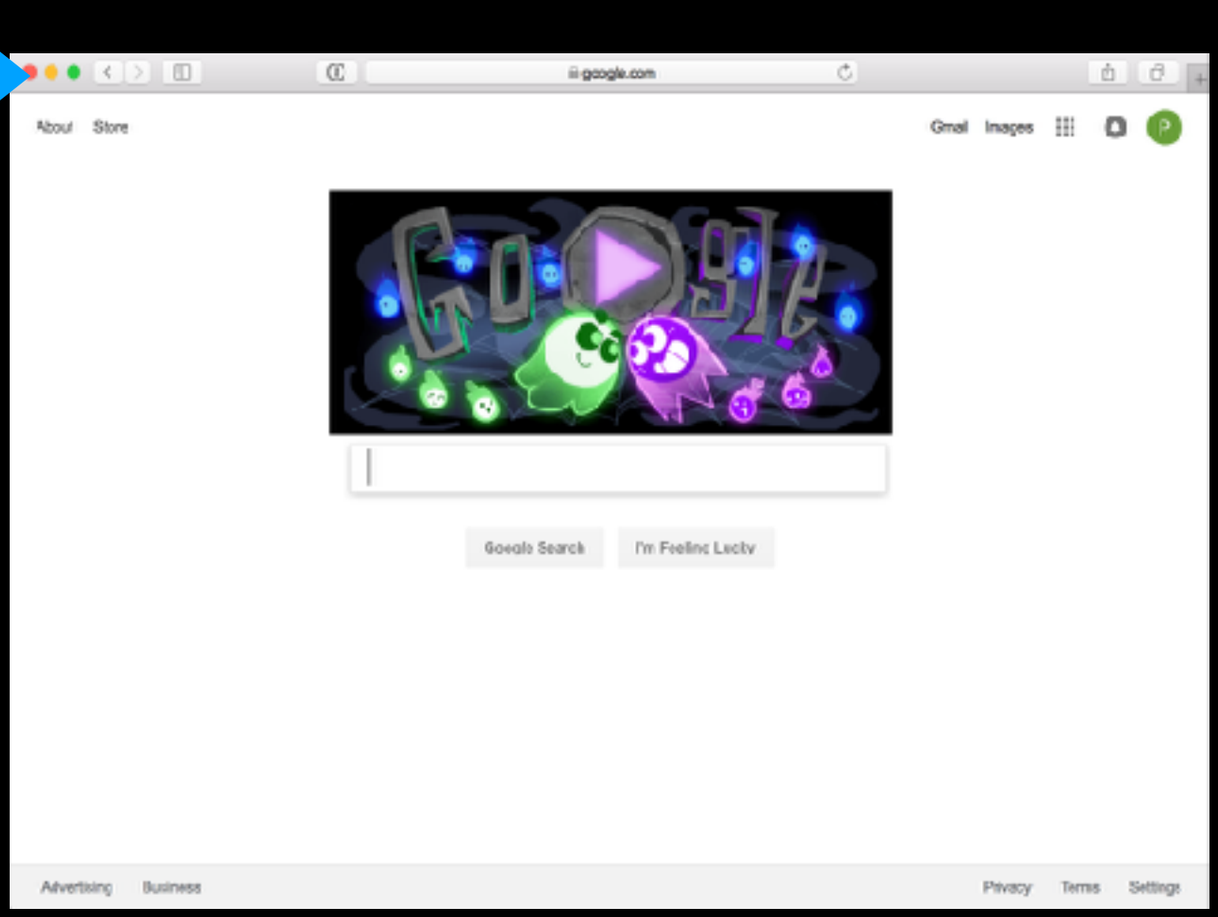
The events, methods + properties of an object all relate to each other. Events can trigger methods, and methods can retrieve or update an objects' properties.

1. Event: button clicked;
2. Calls method changeBackgroundColor();
3. Updates property backgroundColor to blue;

Window Object

The browser represents each window or tab using a **window object**. The **location property** of the **window object** will tell you the URL of the current page.

Object type: window
Properties:
location http://www.google.com



try this command in the window inspect console: **console.log(window.location)**
(it will return the current url)

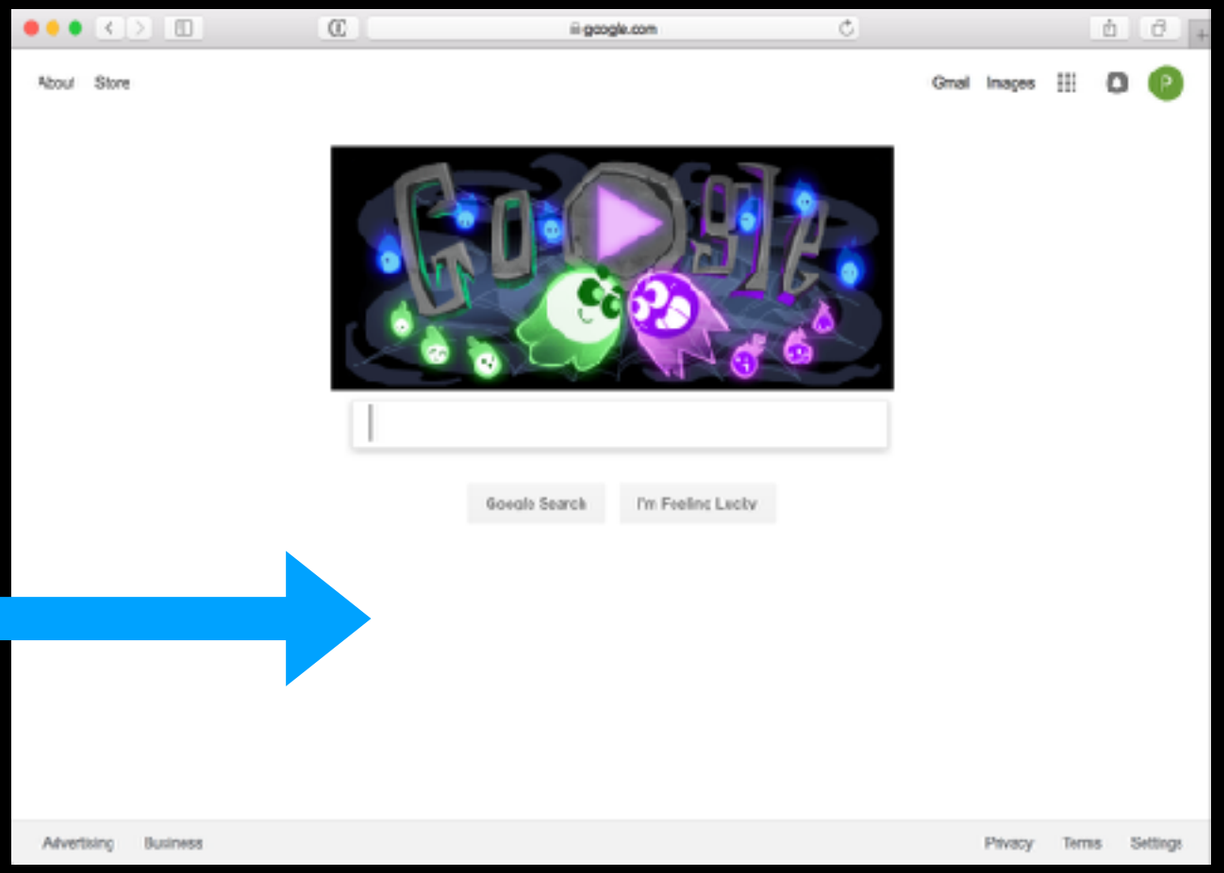
Document Object

The current webpage loaded into each window is modeled using a document object.

Object type: document

Properties:

URL	http://www.google.com
lastModified	10/31/18 6:12 pm
title	Google Search Home Page



try this command in the window inspect console: **console.log(document.title)**
(it will return the current url)

Scripts

A script is a set of instructions that a computer can follow one-by-one. Each individual instruction or step is known as a STATEMENT.

Scripts

You need to start with the big picture of what you want to achieve, and break that down into smaller steps:

1. Define the goal
2. Design the script
3. Code each step

What JavaScript Can Do:

Access content: you can use JavaScript to select any element, attribute or text from an HTML page.

Modify content: you can use JavaScript to add elements, attributes, and text to the page or remove them.

Program rules: you can specify a set of steps for the browser to follow, which allows it to access or change the content of a page.

React to events: you can specify that a script should run when a specific event has occurred.

JS Syntax

1. A semicolon notes that a step is over + JavaScript interpreter should move to the next step
2. Each statement starts on a new line
3. JavaScript is case sensitive: myName is not equal to myname or MYNAME
4. Some statements can be organized into code blocks:

```
if (value > theVariable) {
    //do this;
}
```

notice no ; after the code block

JS Syntax

1. You should ALWAYS write comments to explain what your code does (especially JS logic)
2. There can be single-line and multi-line comments
3. Single-line comments are written with two forward
   slashes //
   Often used for short descriptions of what the code is doing
4. Multi-line comments are written using /* and */ characters
   Often used for descriptions of how the script works, or to prevent a section of the script from running when testing it

JS Syntax

1. Once you created a variable, you need to tell what information it should store (assign a value)

2.To announce it you need to create a variable and give it a name (declare it)
Until you have assigned a value, it’s undefined

```
theSpeed = 7;
```

theSpeed is the variable name
= is the assignment operator
7 is the variable value

JS Syntax

Variables

Variables are containers that you can store values in. You start by declaring a variable with the var keyword, followed by any name you want to call it

```
var theVariable;
var theVariable = 'Bob';                    ——> String Data
var theVariable = 10;                       ——> Numeric Data
var theVariable =  true;                    ——> Boolean
var theVariable = [536, 3, 3354684, 325]    ——> Array
```

JS Syntax

let

allows you to declare variables that are limited in scope to the block, statement, or expression on which it is used. This is unlike the **var** keyword, which defines a variable globally, or locally to an entire function regardless of block scope.

At the top level of programs and functions, let, unlike var, does not create a property on the global object.

** Note: the browser is not able to identify local variables.

JS Syntax

const

Constants are block-scoped, much like variables defined using the let statement. The value of a constant cannot change through reassignment, and it can't be redeclared.

Data Type: Numeric

1. Numeric data type handles numbers
2. Use **typeof** to log what kind of data type you're using

```
var thePrice;
var theQuantity;
var theTotal;

thePrice = 5;
theQuantity= 14;
theTotal = thePrice * theQuantity;

console.log(typeof theTotal);
```

Data Type: Strings

1. The string data type consists of letters and other characters
2. They have to be enclosed with a pair of quotes (single or double)
   Opening quote must match the closing quote
3. They can be used working with any kind of text

```
var myName;
var myMessage;

myName = “Rebecca”;
myMessage = “Atlanta is a really great show y’all should check it out!”;

console.log(typeof myName);
```

Data Type: Boolean

1. Boolean data type can have one of two values: true or false
2. They are helpful when determining which part of a script should run (when code can tak
more than one path);

```
var todayIsCloudy;
todayIsCloudy = true;

if (todayIsCloudy == true) {
        // if true, show this
        console.log("Yes, today is cloudy!”);
        // otherwise show this
 } else {
        console.log("No, it's actually sunny today!");
 }
```

Shorthand for creating variables

You can use these shorthands to create variables:

1:

```
var speed = 5;
var quantity = 14;
var total = speed * quantity;
```

2:

```
var speed, quantity, total;
speed = 5;
quantity = 14;
total = speed * quantity;
```

3:

```
var speed = 5, quantity = 14;
var total = speed * quantity;
```

Changing the value of a variable

Once you have assigned a value to a variable, you can then change it later in a script

```
var speed = 5;
var quantity = 14;
var total = speed * quantity;

//maybe something changed here and the value has to change now

speed = 10;

var textToShow = document.getElementById("hello");
textToShow.innerHTML = "Total cost is: " + "$" + total;
```

Rules for naming variables

1. The name must begin with a letter, $ sign or an underscore _, it cannot start with a number (e.g. name, $name, _name) - I would avoid $, because it might confuse it with jquery

2. You cannot use dash - or a dot . in a variable name (e.g. my-name, my.name)

3. You cannot use keywords or reserved words as variable names (e.g. function, type, this, etc.) - it usually changes the color when you use it

4. All variables are case sensitive (it is bad practice to create the same name using different cases (e.g. myName & Myname) - do not start with a capital letter in general

5. Use a name that describes the kind of information that the variable stores (e.g. firstName, lastName)

6. If a variable name is made up of more than one word use capital letter for the first letter of every word after the first one or underscore (e.g. myFirstName, myLastName, my_first_name, my_last_name)

In your JS file, let's try these commands:

```
console.log("Hello World");
console.logwindow.location);
console.log(document);
```