

# Intro to the Web

Lecture 1



**ALEX HINDS | FRONTEND DEVELOPER | @AL\_HINDS**

# Course Overview

Intro to the Web

---

Language Overview

Introduction to the DOM

DOM API and Events

Asynchronous JavaScript

Extended JavaScript

# Why learn JavaScript?

## It's kinda a big deal

Ubiquitous in modern web development, runs server side and in the browser

---

## Poorly understood

Very rarely taught well, and has evolved from a messy origin

## Flexible

A simple, weakly typed, scripting language that is easy to pick up and develop with

# Why learn JavaScript?

It's kinda a big deal

Ubiquitous in modern web development, runs server side and in the browser

Poorly understood

Very rarely taught well, and has evolved from a messy origin

---

Flexible

A simple, weakly typed, scripting language that is easy to pick up and develop with

# Why learn JavaScript?

**It's kinda a big deal**

Ubiquitous in modern web development, runs server side and in the browser

**Poorly understood**

Very rarely taught well, and has evolved from a messy origin

**Flexible**

A simple, weakly typed, scripting language that is easy to pick up and develop with

---

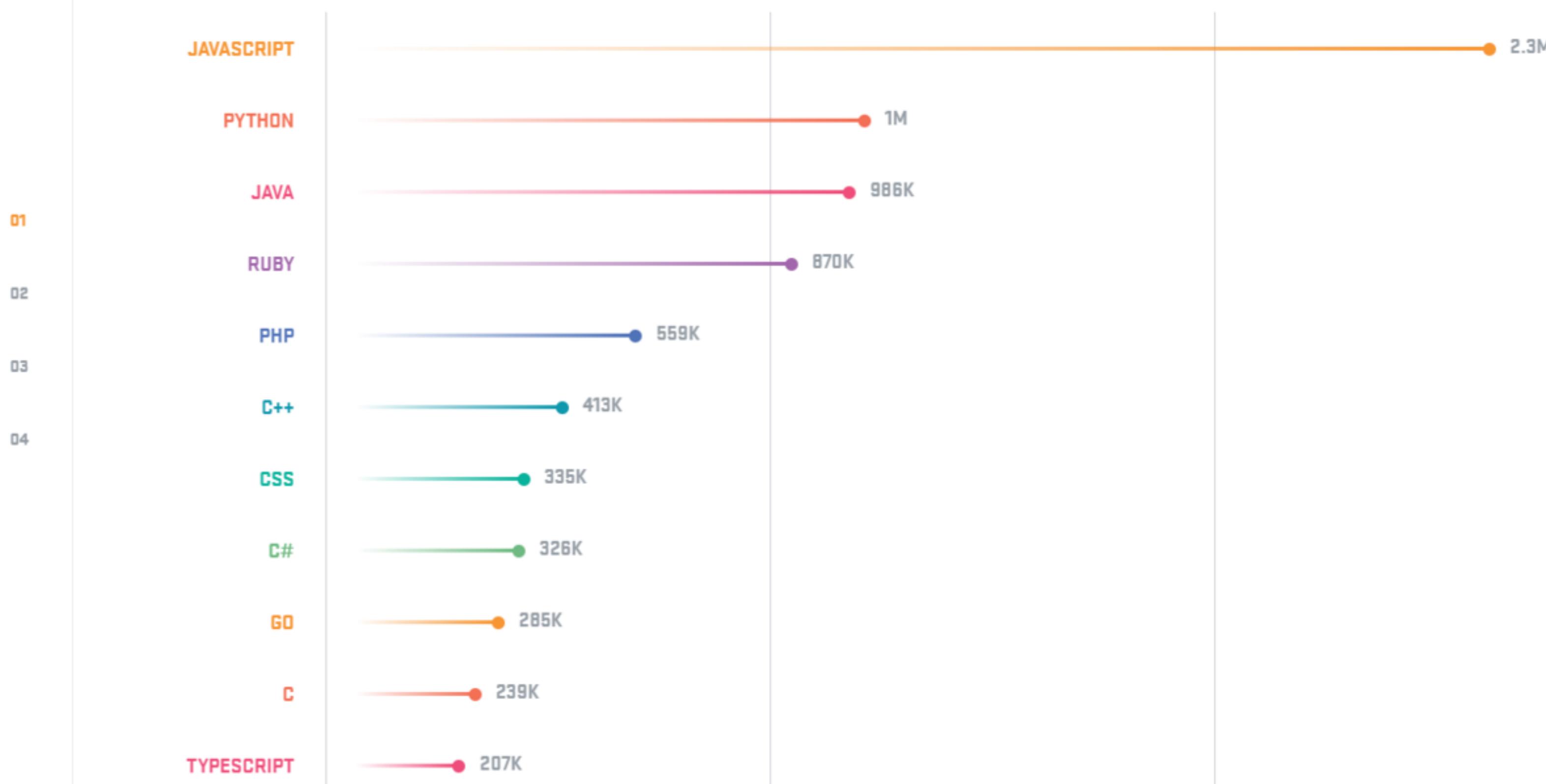


# The fifteen most popular languages on GitHub

by opened pull request

GitHub is home to open source projects written in 337 unique programming languages—but especially JavaScript.

GitHub



## **FOCUS**

---

**JavaScript in the Browser**

**Language Fundamentals**

**Frontend Fundamentals**

## **WE WON'T**

---

**Cover NodeJS in depth**

**Cover HTML & CSS in depth**

**Teach you JS frameworks**

Subjects not covered are **not less**  
**important** - we just don't have  
time to do them justice!

# Getting Setup

The right tools for the job

# RECOMMENDED RESOURCES

---

## MDN

Mozilla's Developer Network has comprehensive documentation for web development.

<https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference>

## Course Material

We've put together a selection of examples and additional reading which we hope helps.

## You Don't Know JS

Free ebook available which has particularly good explanations on the JS language.

<https://github.com/getify/You-Dont-Know-JS>

## The web...

There are countless resources on the web for web development these days. Very few bad choices.

# Development Browser Choice



## Google Chrome

Modern, bleeding edge support of JS and CSS plus great dev tools



## Mozilla Firefox

Not built by Google, and committed to the open web.  
Still a great browser.



## Safari

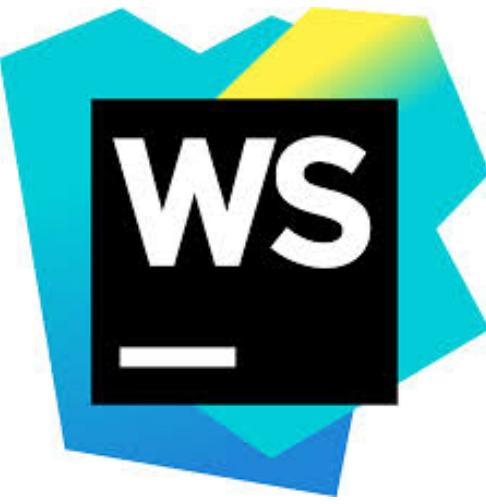
Same JS engine as Chrome, with a slightly different DevTools experience

# Use an IDE



**VSCode**

Built for web development  
and has some nifty type  
inference built in



**WebStorm**

Will likely require a license  
but a trial period should be  
available.



# Node.js

Installing Node.js on your system will be helpful even if you're developing for the web.

You can download and install the node binary from the link below.

<https://nodejs.org/en/>

# The modern web

Some background to how we got here

# The modern web



## HTML

The scaffold for any modern web page. A form of markup language used to format web documents.



## CSS

Cascading Style Sheets. After a few false starts the de facto standard for applying styles and basic animations



## JavaScript

Envisioned initially to help supercharge CSS, but has since evolved to facilitate 'dynamic web pages'

## **HTML**

---

- **Lightweight, scaffold for web pages**
- **Collection of tags and attribute-value pairs**
- **Limited but core functionality**



# HTML Crash Course

# HTML

Structure

Tags

Attributes

## HTML Elements

The most basic selector, apply to the corresponding HTML element. eg:

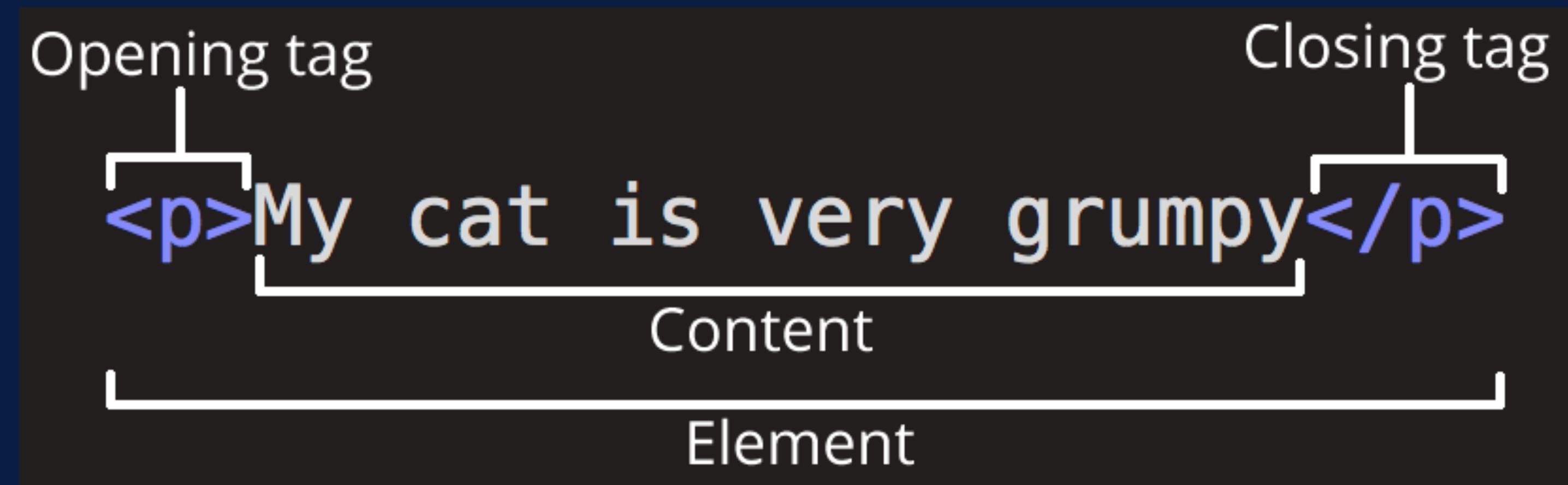


Image Source: MDN

# HTML

Structure

Tags

Attributes

## Tags

Tags define the type of element we are rendering, and help to semantically structure the document.

```
<html></html>, <body></body>, <main></main>  
<!-- This is a comment -->
```

```
<!-- These tags are for structuring text -->  
<p></p>, <h1></h1>, <h2></h2>
```

```
<!-- These tags are for lists -->  
<ul>unorderd</ul> or <ol>ordered</ol>  
<li>list items</li>
```

```
<!-- Thest tags are for content structure -->  
<nav></nav>, <article></article>, <section></section>, <form></form>
```

```
<!-- These tags are special tags -->  
<img/>, <input/>, <button></button>  
<!--
```

Some people just prefer to use divs everywhere.  
Those people are bad people  
-->  
<div></div>

# HTML

Structure

Tags

Attributes

## Attributes

Attributes control how an element behaves. There are many 'standard' attributes that apply to all elements. Eg. class, style, id.

```
<!-- Will render a text input -->
<input type="text" name="firstName"
placeholder="Jeff">
<!-- Will render a password field -->
<input type="password" name="password">
```

```
<!-- When this form is submitted it'll do a get
request on /some/path -->
<form action="/some/path" method="get"></form>
```

```
<!-- This image has a src attribute of fake.jpg -->

```

# The modern web



## HTML

The scaffold for any modern web page. A form of markup language used to format web documents.



## CSS

Cascading Style Sheets. After a few false starts the de facto standard for applying styles and basic animations



## JavaScript

Envisioned initially to help supercharge CSS, but has since evolved to facilitate 'dynamic web pages'

- Arose to extend HTML functionality
- ‘Style elements’ were a poor, cumbersome way to style - CSS abstracted out style from markup using selectors



# CSS Crash Course

# CSS Structure

---

## Property/Value Pairs

CSS uses a selector combined with a key-value property pair to style elements. There are hundreds of possible styles, but common one's include: margin, padding, color, font... to name a few.

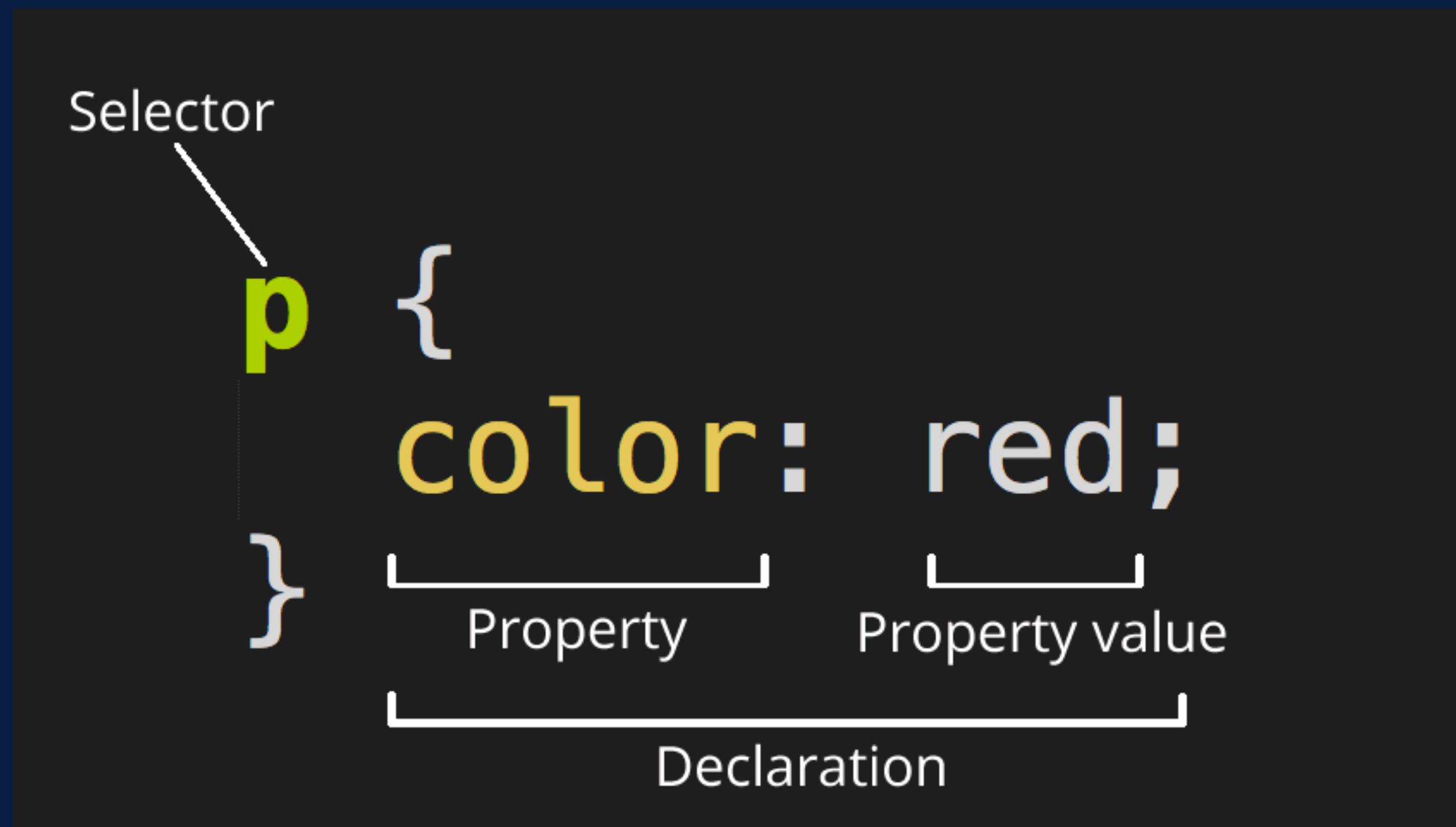


Image Source: MDN

# CSS Selectors

Tags

Classes

Pseudo

ID

## Tag Selectors

The most basic selector, apply to the corresponding HTML element. eg:

```
p {  
    color: black;  
    font-weight: normal;  
}
```

```
h1 {  
    color: red;  
    font-weight: bold;  
}
```

# CSS Selectors

Tags

Classes

Pseudo

ID

## Class Selectors

A more extensible approach to styling, denoted by the leading ‘.’ in the selector. eg:

```
.box {  
    width: 100px;  
    height: 100px;  
}
```

```
.red {  
    background-color: red;  
}
```

```
.blue {  
    background-color: blue;  
}
```

# CSS Selectors

Tags

Classes

Pseudo

ID

## Pseudo Selectors

Make use of the `:` operator and must be combined with another selector.  
Examples, include element states or positions like `:hover, :focus, :after, :before`.

```
.box {  
    width: 100px;  
    height: 100px;  
}  
  
/* box with our red class will turn red on hover */  
.box.red:hover {  
    background-color: red;  
}  
  
/* A normal box will turn black */  
.box:hover {  
    background-color: black;  
}
```

# CSS Selectors

Tags

Classes

Pseudo

ID

## IDs and special selectors

IDs are denoted by a leading **#**, other special selectors include **>** and **+**.

```
/*  
   Highest precedence but applies to only  
   _one_ element.  
*/
```

```
#nav {  
  font-weight: bold;  
}
```

```
/*  
   The p child of the logo will have opacity 0.5  
*/  
#logo > p {  
  opacity: 0.5;  
}
```

“**Specificity** is a weight that is applied to a given CSS declaration, determined by the number of each selector type in the matching selector.”

---

Mozilla Developer Network

<https://developer.mozilla.org/en-US/docs/Web/CSS/Specificity>

By combining selectors and specificity we can apply style rules as generally or specifically as we choose.

---

Mozilla Developer Network

<https://developer.mozilla.org/en-US/docs/Web/CSS/Specificity>

# Same HTML: different CSS



A demonstration of what can be accomplished through [CSS](#)-based design. Select any style sheet from the list to load it into this page.

Download the example [HTML FILE](#) and [CSS FILE](#)

## THE ROAD TO ENLIGHTENMENT

Littering a dark and dreary road lay the past relics of browser-specific tags, incompatible DOMs, broken [CSS](#) support, and abandoned browsers.

We must clear the mind of the past. Web enlightenment has been achieved thanks to the tireless efforts of folk like the W3C, WASP, and the major browser creators.

The CSS Zen Garden invites you to relax and meditate on the important lessons of the masters. Begin to see with clarity. Learn to use the time-honored techniques in new and invigorating fashion. Become one with the web.

MID CENTURY MODERN  
by Andrew Lohman

GARMENTS  
by Dan Mall

STEEL  
by Steffen Knoeller

APOTHECARY  
by Trent Walton

SCREEN FILLER  
by Elliot Jay Stocks

CSS Zen Garden

<http://www.csszengarden.com>

# The modern web



## HTML

The scaffold for any modern web page. A form of markup language used to format web documents.



## CSS

Cascading Style Sheets. After a few false starts the de facto standard for applying styles and basic animations



## JavaScript

Envisioned initially to help supercharge CSS, but has since evolved to facilitate 'dynamic web pages'

## JAVASCRIPT

---

- Initially; facilitates animation, form validation, warnings, prompts.
- Extended in mid-2000s to support XHR/AJAX to become core part of modern web

## QUESTION

---

How do these  
three components  
fit together in  
practice?

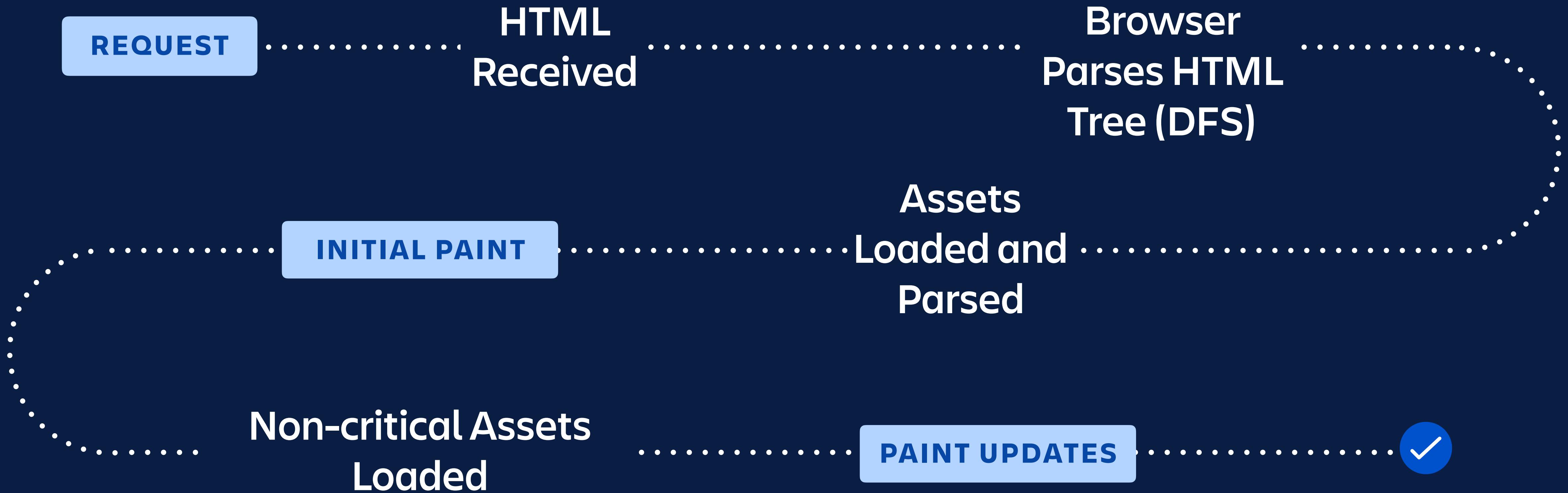




# The Web Document

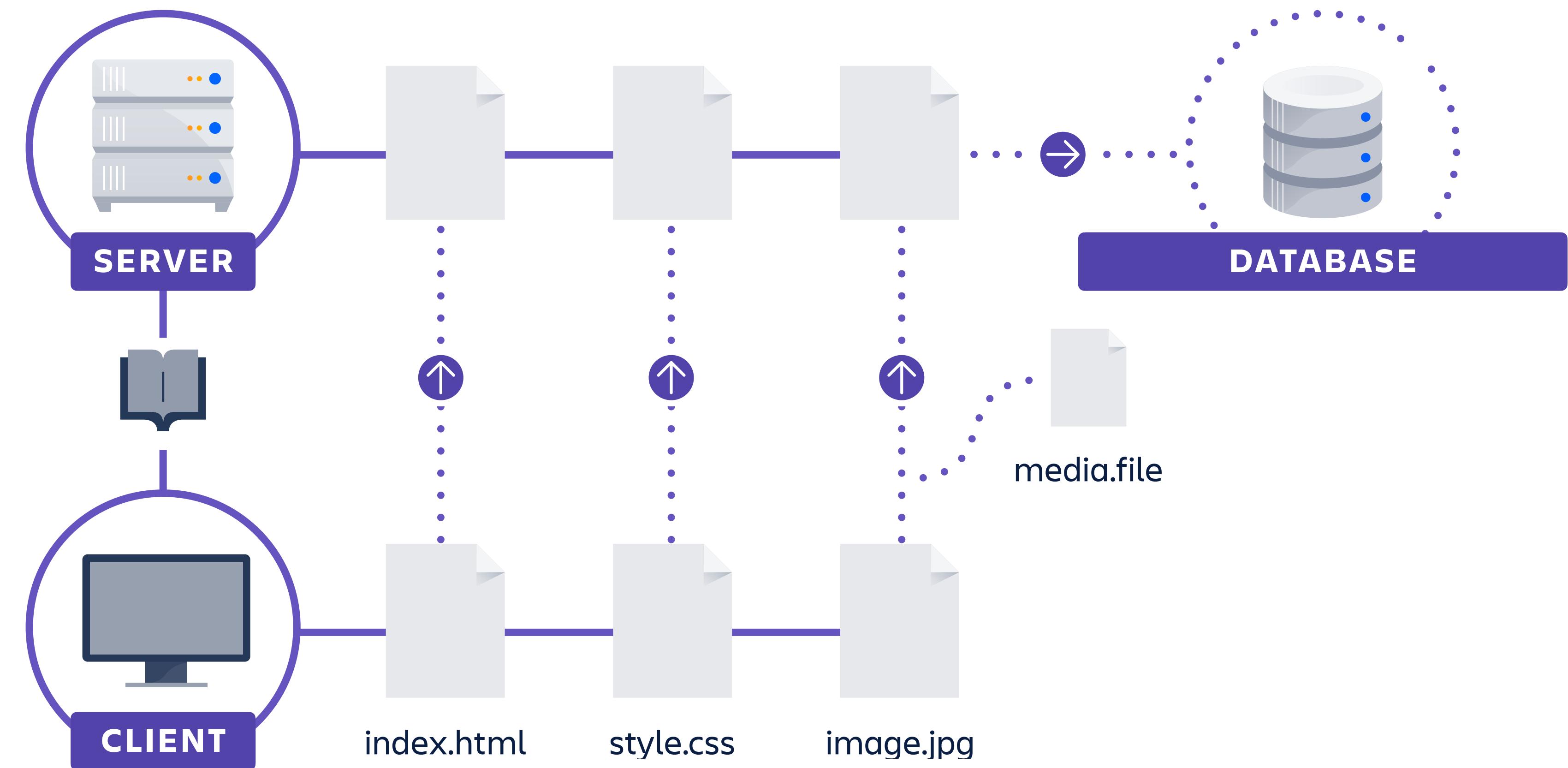
These three building blocks come together to form a web document which our browser can interpret and update.

# BASIC WEB DOCUMENT TIMELINE



# WEB BASICS

Server gets a request from your browser, serves HTML, which in turn asks for any additional assets.



## QUESTION

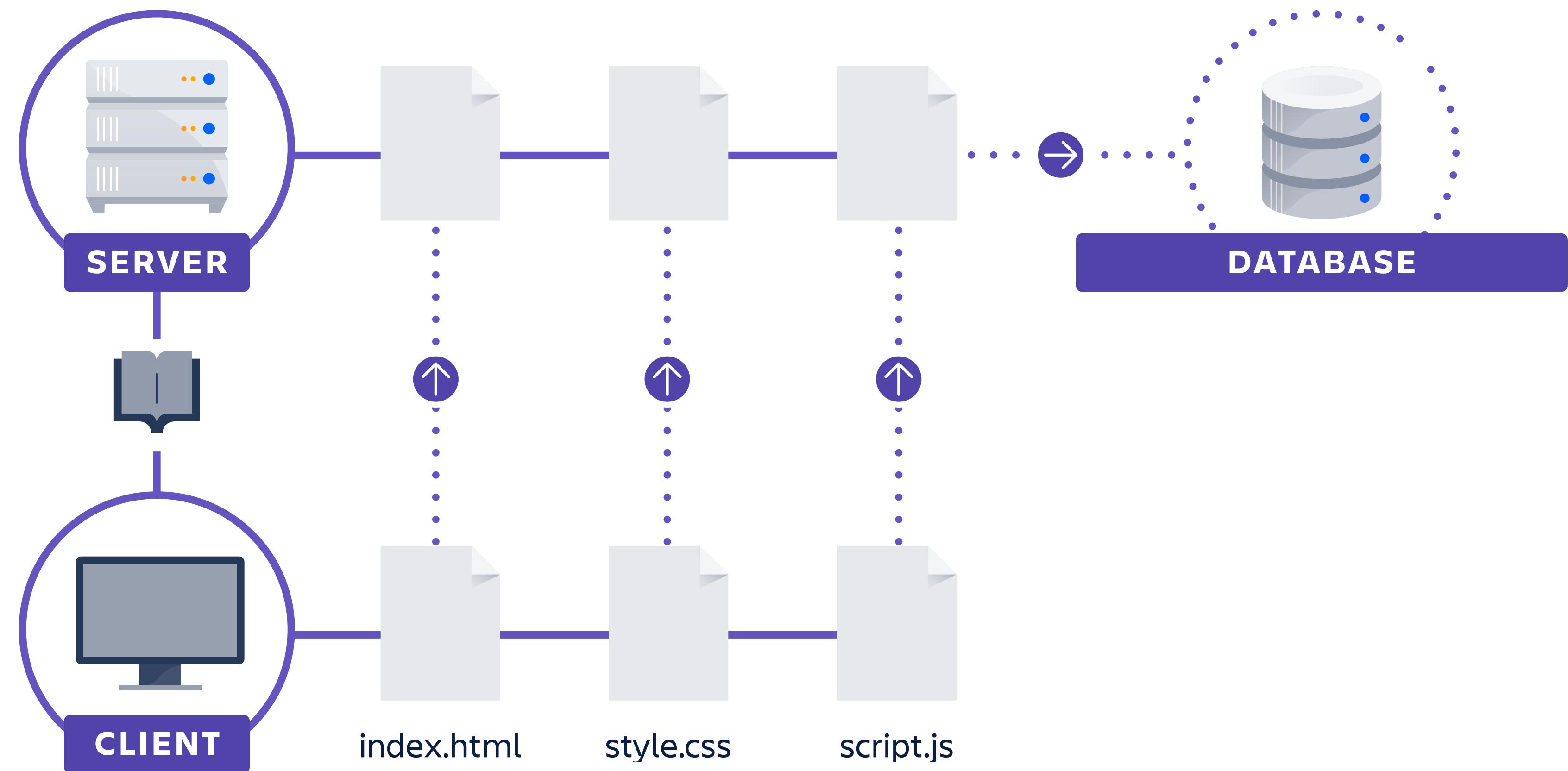
---

**CSS and HTML  
seem pretty nifty -  
why do I need  
JavaScript?**



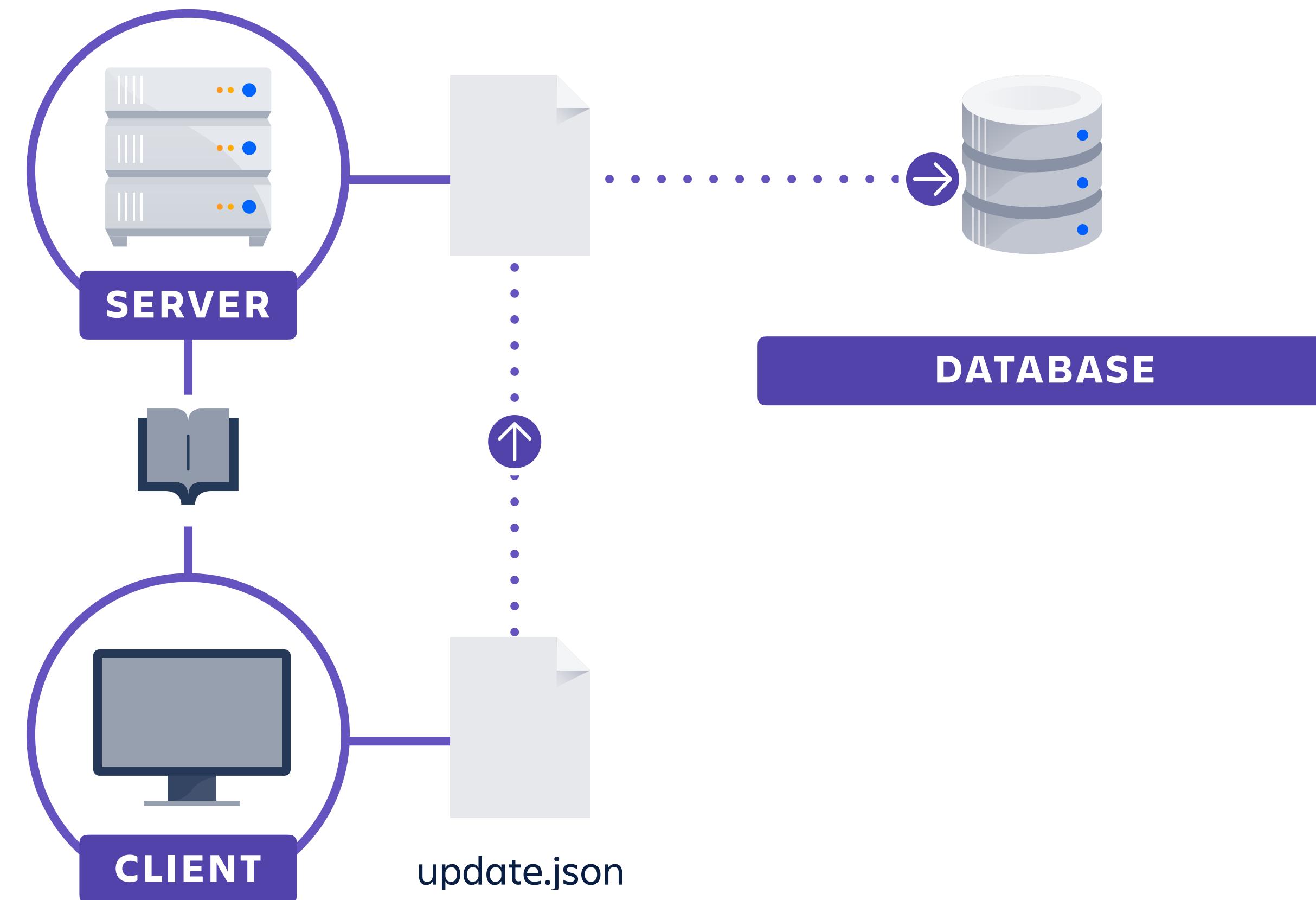
# MODERN WEB

Similar initial setup but less reliant on server for CPU processing. Updates can happen in the background.



# MODERN WEB

Updates only need to send critical information; significantly smaller payloads, no need to re-render the whole page.





You don't need to understand all of  
this now. By the end of the course it  
will make more sense!