

Software Development

---

# Node.js

Module 07

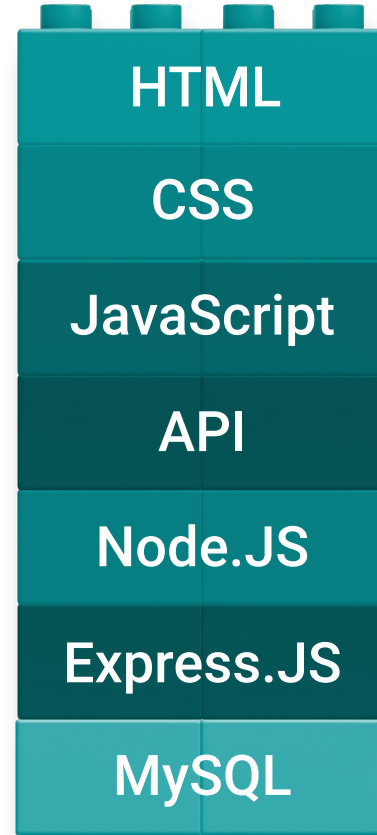


What is **full-stack** web development?



# Full-Stack Web Development

Full-stack web development encompasses the suite of tools required to build both the front and back ends of a web application.



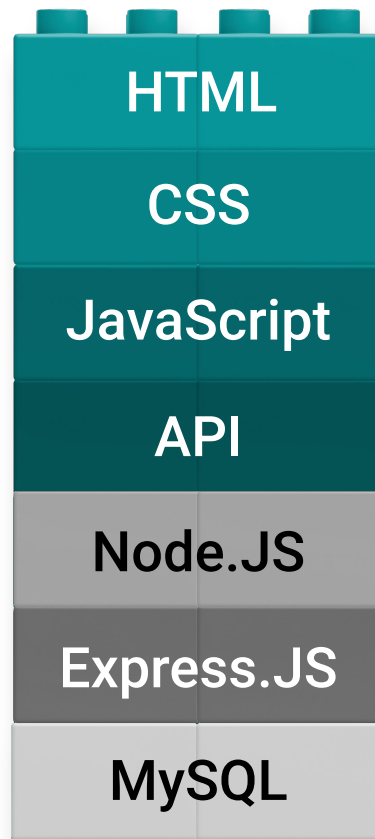


How much of the  
**stack** do we know?



# Client-Side

So far, we have learned about client-side development. The three primary components of client-side code are HTML, CSS, and JavaScript. We can also fetch data from an API for use in the client.



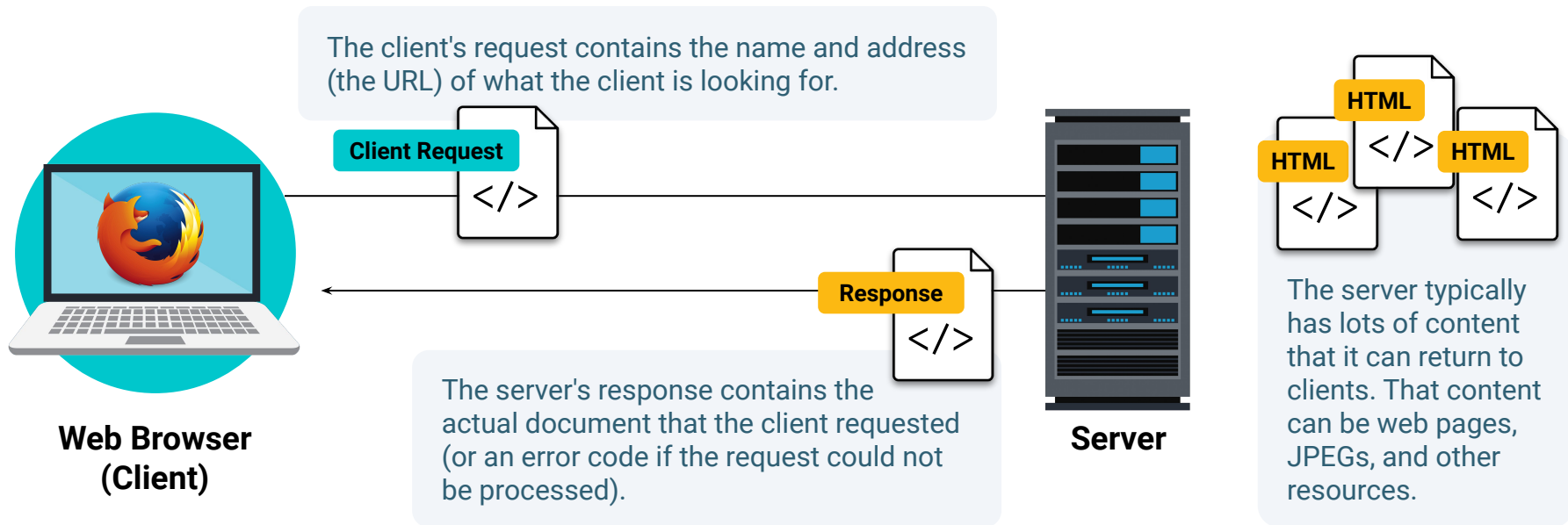


What is the **client-server** model?



# The Client-Server Model

In modern web applications, there is constant back-and-forth communication between the visuals displayed on the user's browser (the front end) and the data and logic stored on the server (the back end). Clients make requests, and servers respond.





Key Question:  
So what is  ?





# Definition of NodeJS

**Node.js** is an open source, cross-platform JavaScript runtime environment designed to be run outside of the browser.

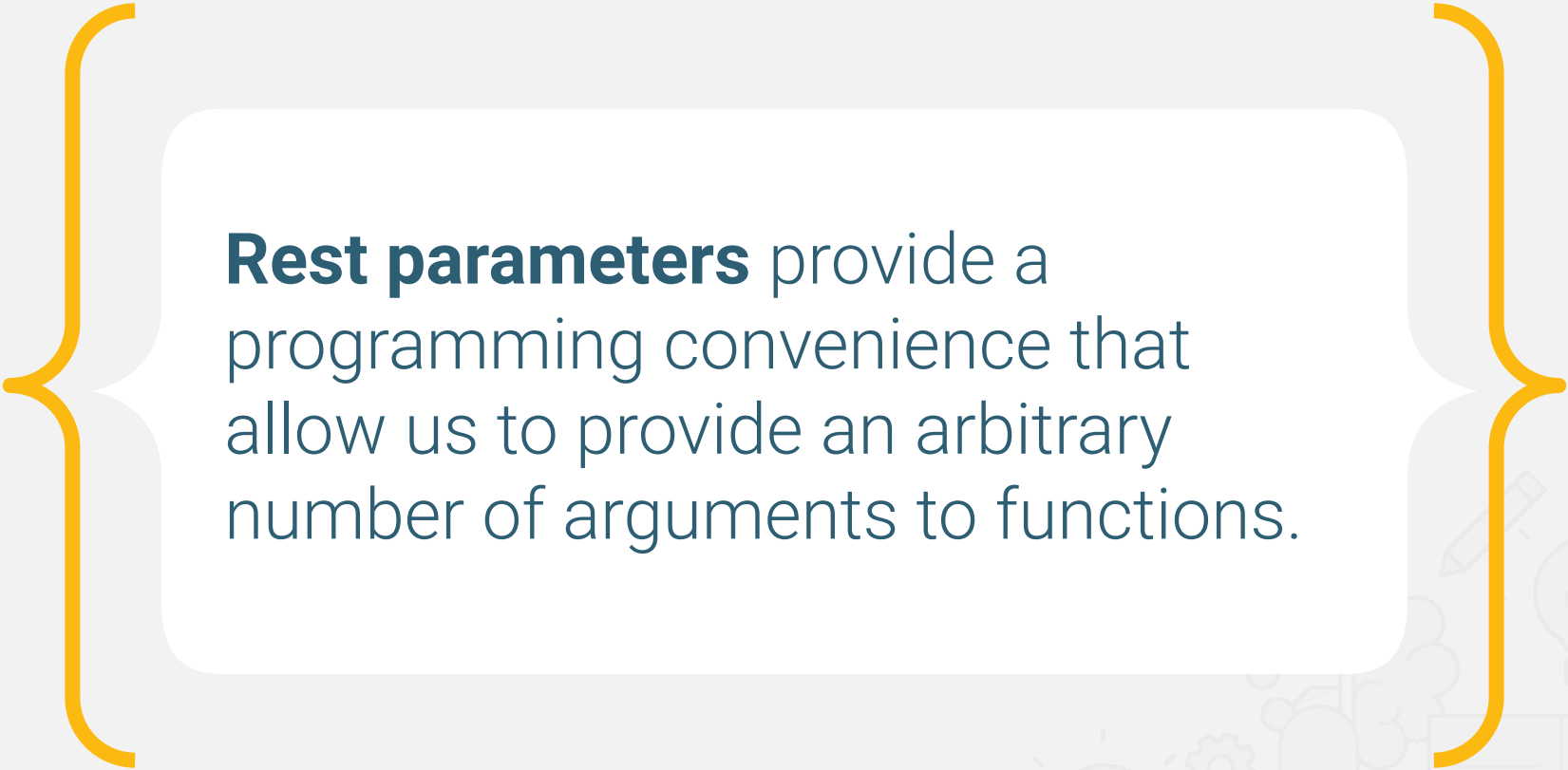
It is a general utility that can be used for a variety of purposes including asset compilation, scripting, monitoring, and **most notably as the basis for web servers.**






What are **rest**  
**parameters?**





**Rest parameters** provide a programming convenience that allow us to provide an arbitrary number of arguments to functions.



# Understanding Rest Parameters



With rest parameters, we can create functions that accept any number of parameters.



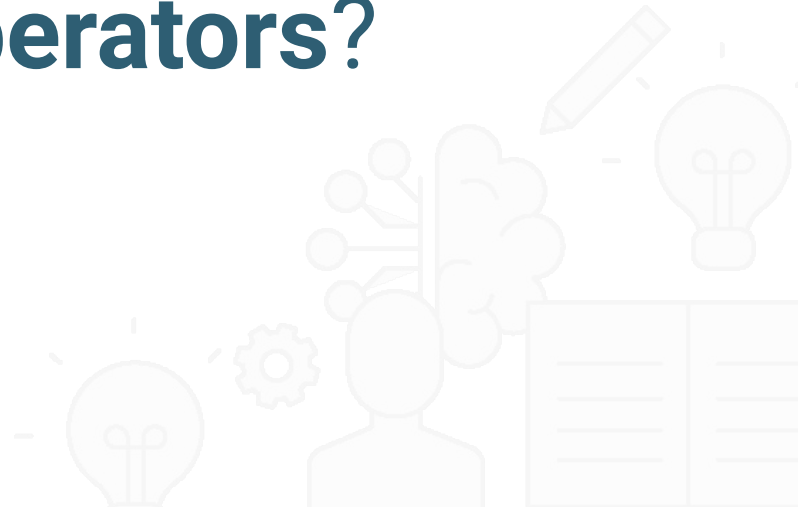
For example, with the function definition `multiply(...nums) {}`, we can write code that multiplies together any number of supplied numbers.

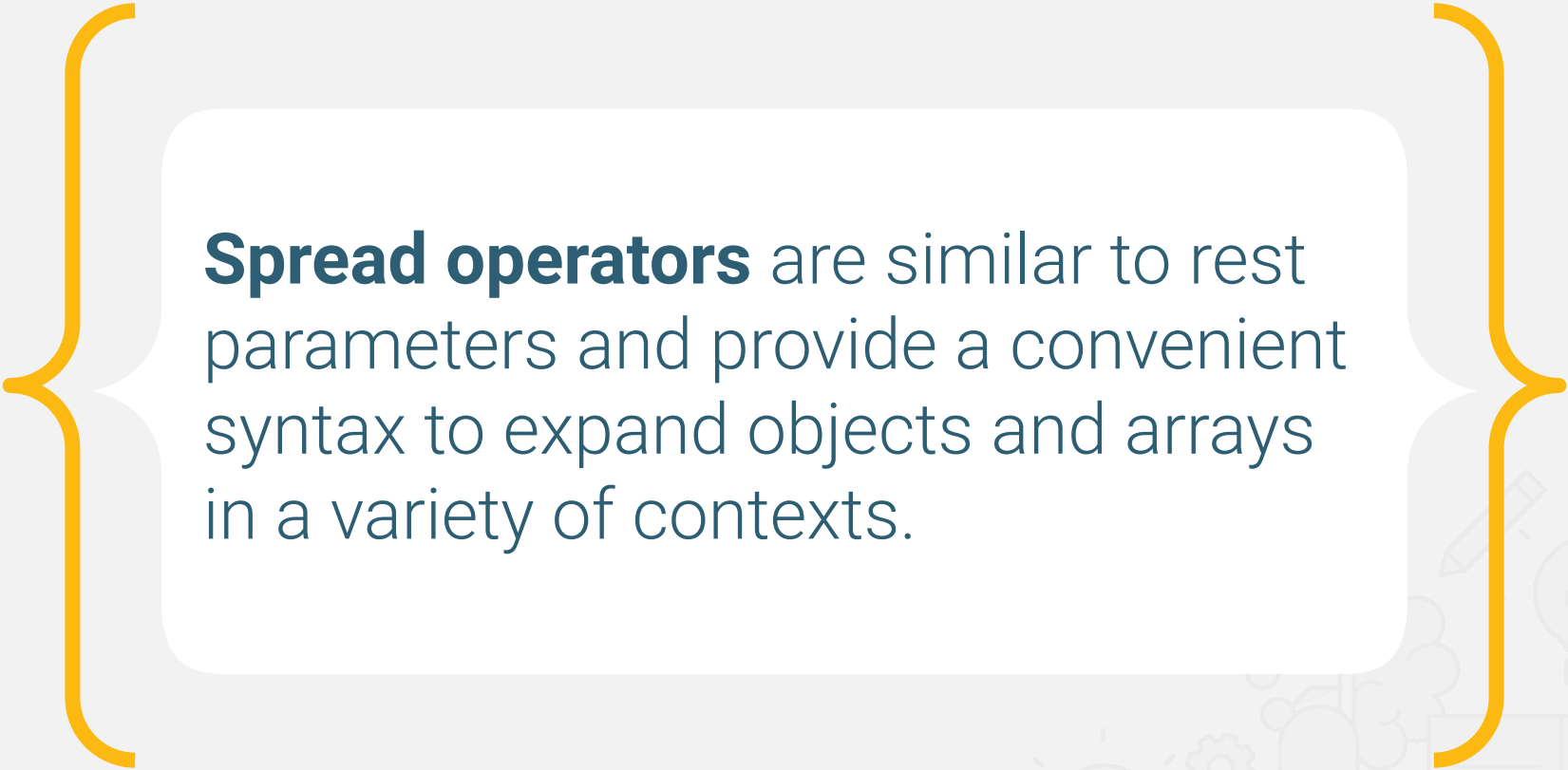


The caller of the function then has the flexibility to provide any number of parameters, such as `multiply(1, 2, 3)` or `multiply(1, 2, 3, 4, 5)`.

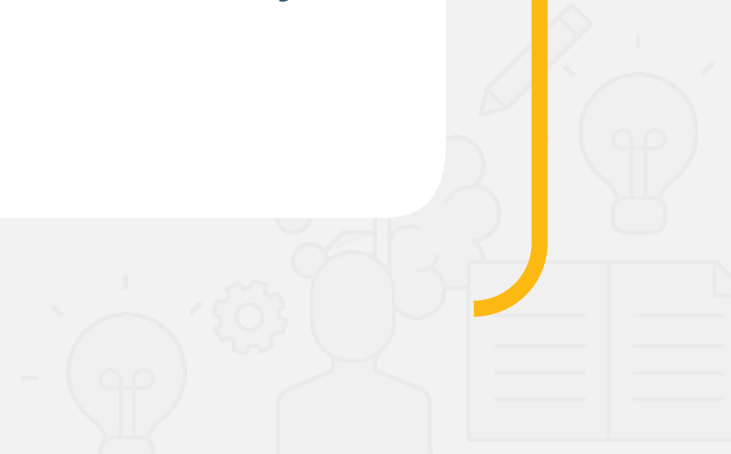


What are  
**spread operators?**





**Spread operators** are similar to rest parameters and provide a convenient syntax to expand objects and arrays in a variety of contexts.



# Understanding Spread Operators

We can use spread operators in many contexts, including:



We can use spread operators in many contexts, including:



Populating one array using the contents of another



Constructing one object using the contents of another



Supplying multiple function arguments at once using the ellipsis (`...`) operator



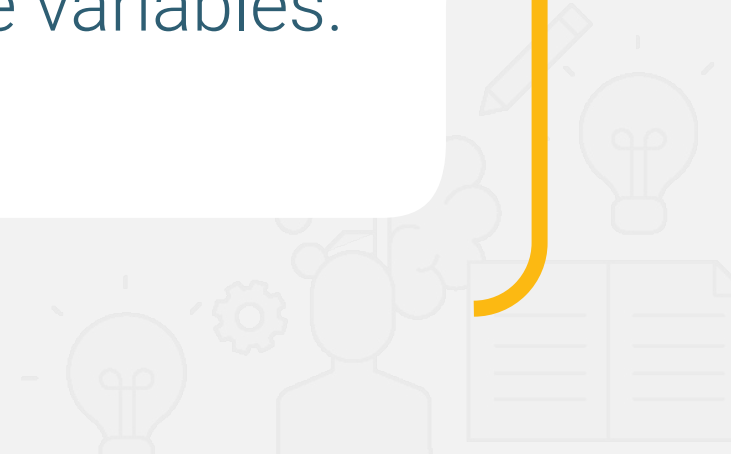
What is **destructuring**?







**Destructuring** provides a convenient way to extract values from arrays and objects into separate variables.



# Understanding Destructuring

Destructuring can be used in a variety of contexts, including:



Arrays



Objects

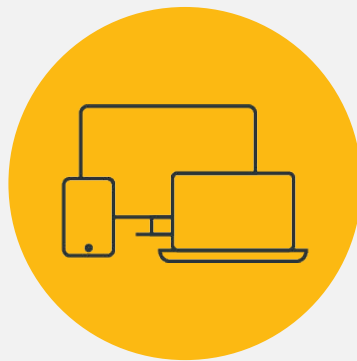


Function arguments



Module imports

With destructuring, we can concisely extract multiple elements or properties using less code than iterating through the entire object or array.



# Instructor **Demonstration**

Mini-Project