The Web Server

2 min

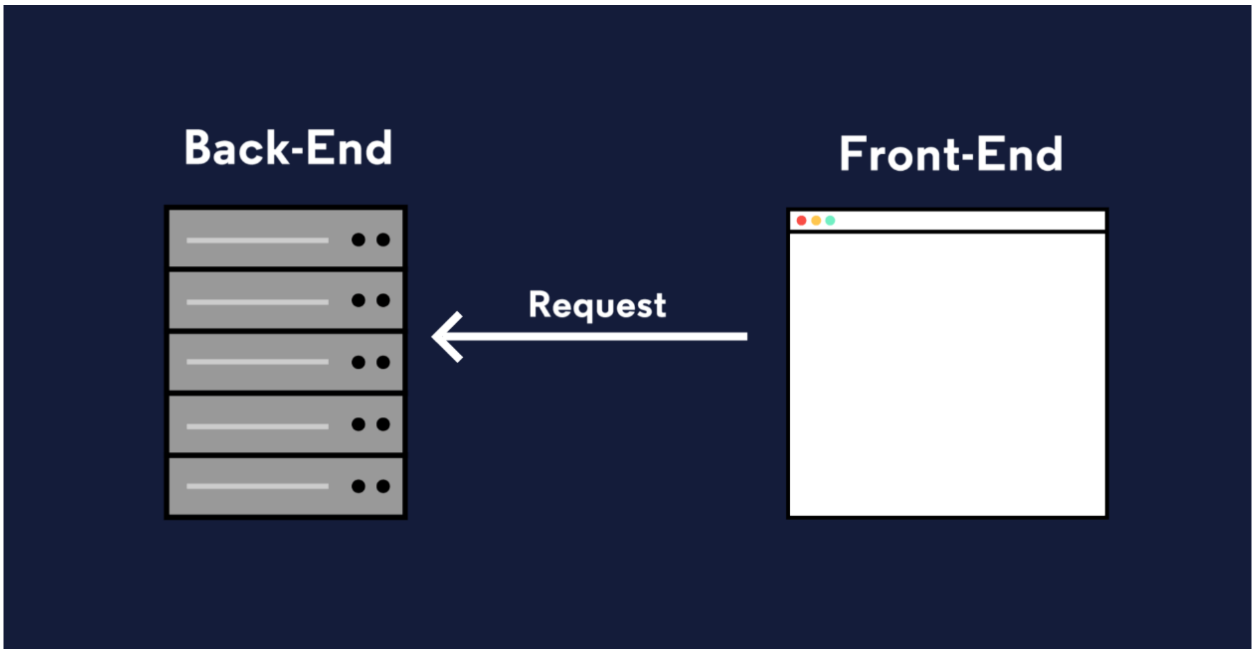
We talked about how the front-end consists of the information sent to a client so that a user can see and interact with a website, but where does the information come from? The answer is a *web server*.

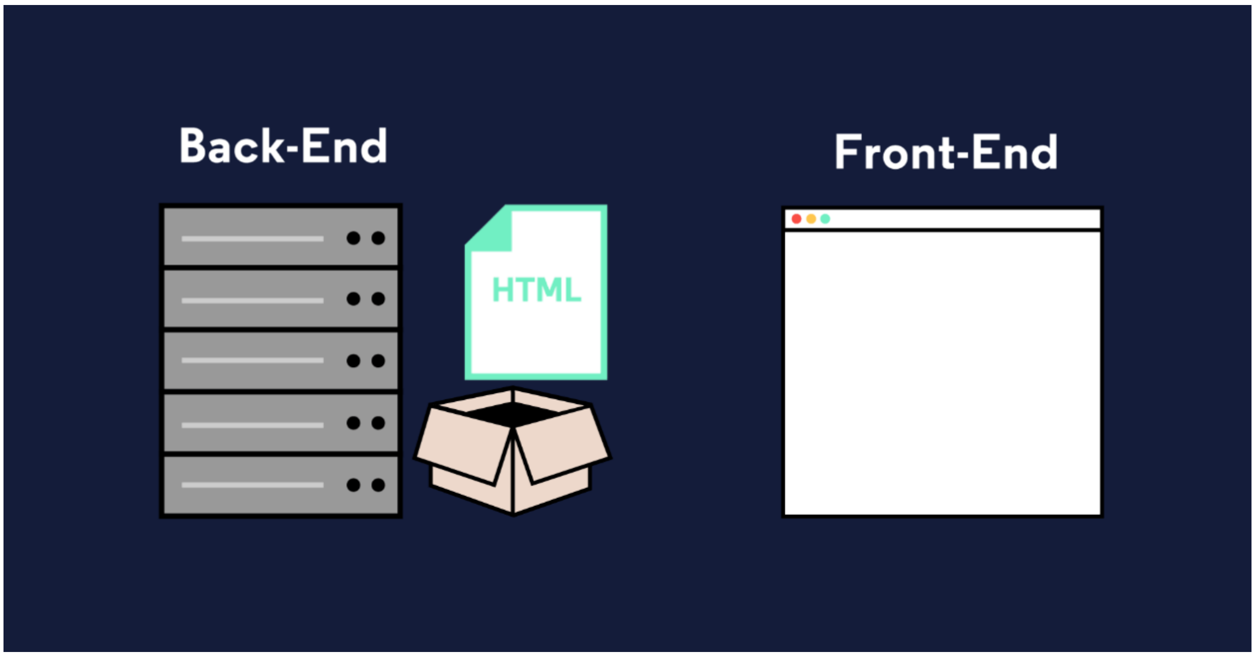
The word "server" can mean a lot of things in computing, but we're going to focus on web servers specifically. A *web server* is a process running on a computer that listens for incoming requests for information over the internet and sends back responses. Each time a user navigates to a website on their browser, the browser makes a request to the web server of that website. Every website has at least one web server. A large company like Facebook has thousands of powerful computers running web servers in facilities located all around the world which are listening for requests, but we could also run a simple web server from our own computer!

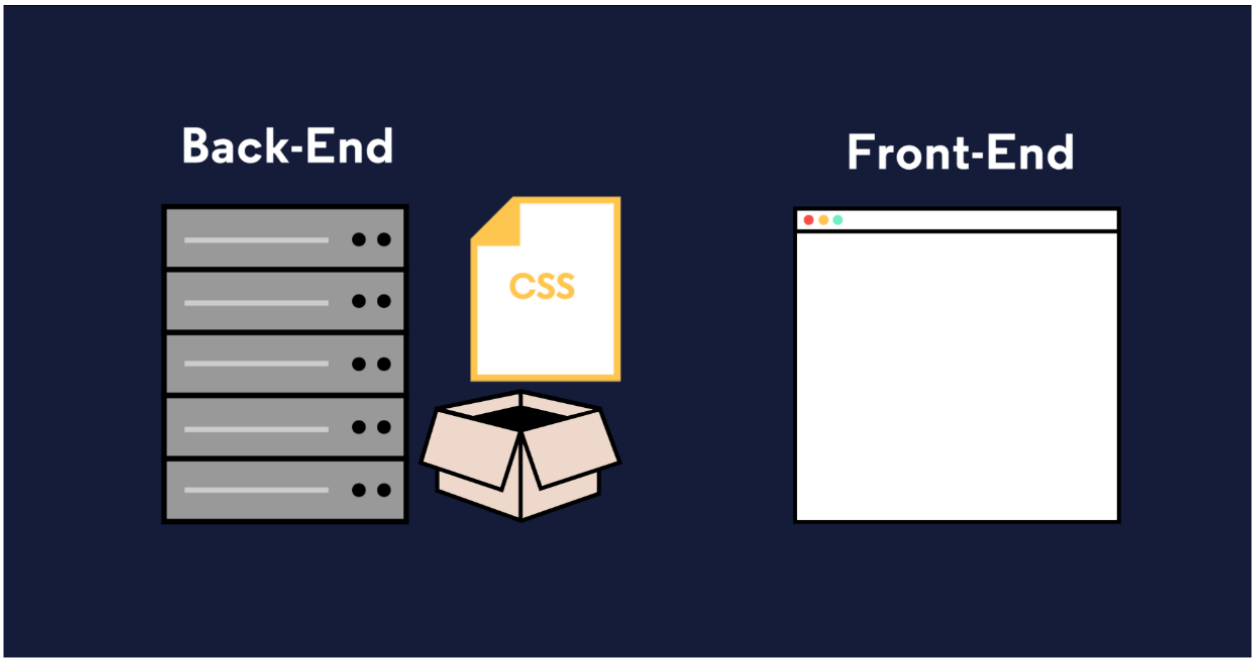
The specific format of a request (and the resulting response) is called the *protocol*. You might be familiar with the protocol used to access websites: HTTP. When a visitor navigates to a website on their browser, similarly to how one places an order for takeout, they make an HTTP request for the resources that make up that site.

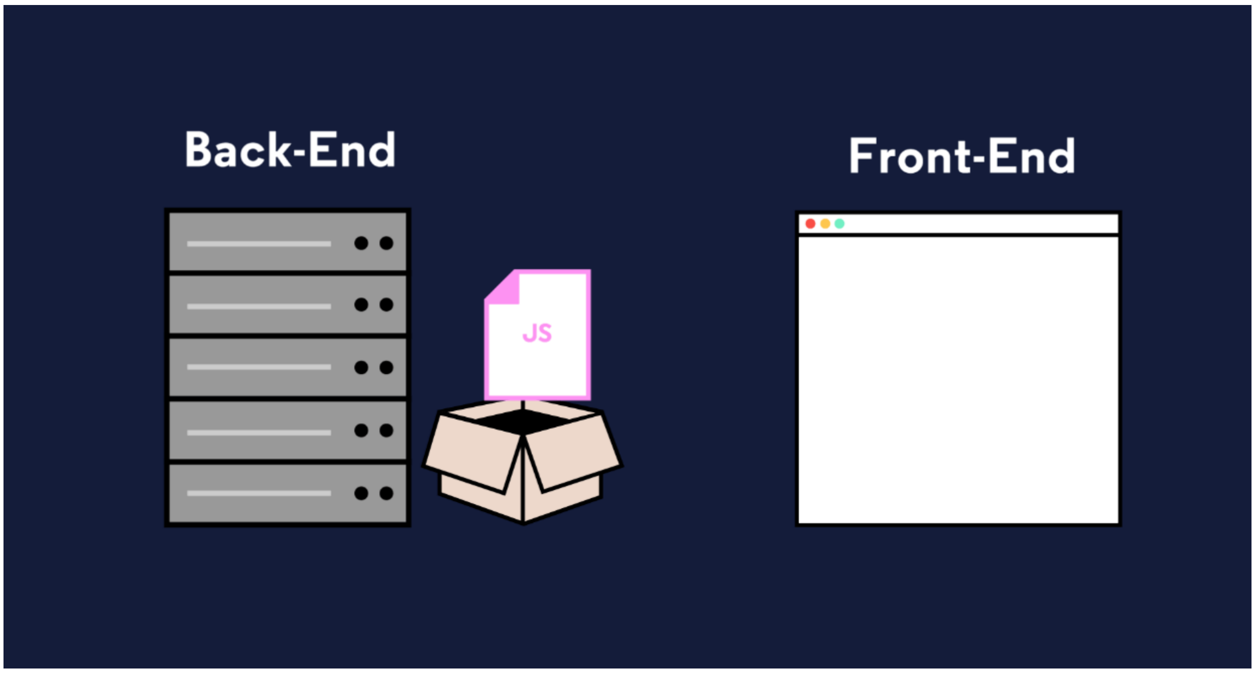
For the simplest websites, a client makes a single request. The web server receives that request and sends the client a response containing everything needed to view the website. This is called a *static website*. This doesn't mean the website is not interactive. As with the individual static assets, a website is static because once those files are received, they don't change or move. A static website might be a good choice for a simple personal website with a short bio and family photos. A user navigating Twitter, however, wants access to new content as it's created, which a static website couldn't provide.

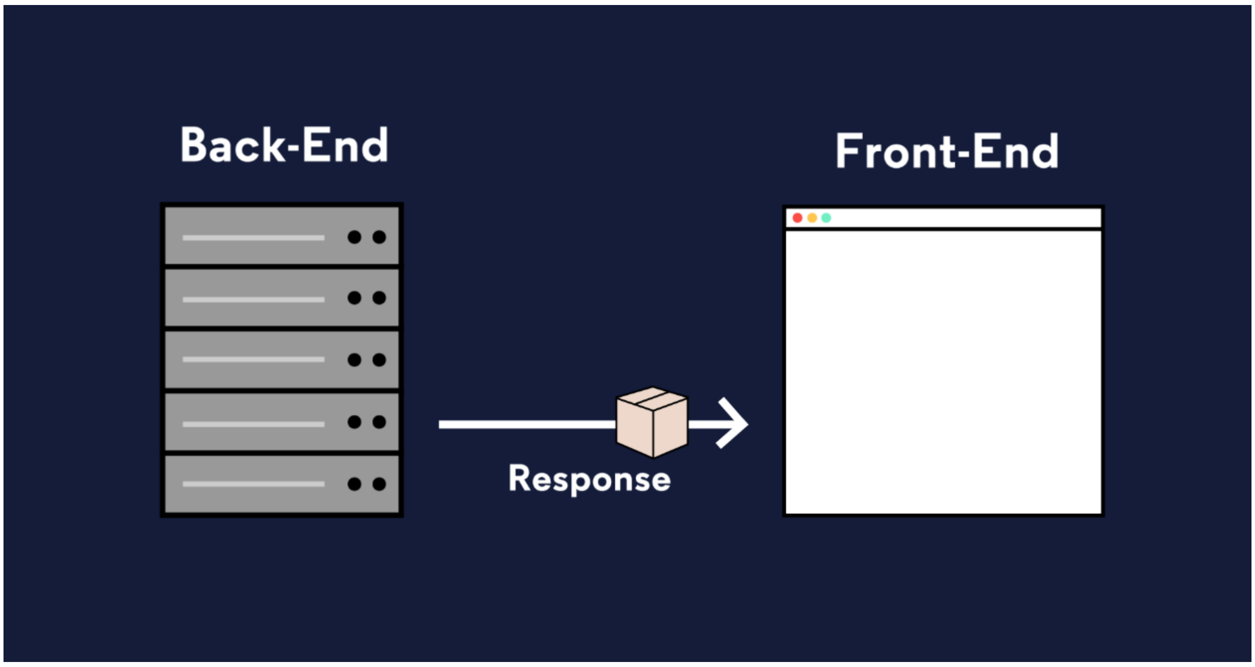
A static website is like ordering takeout, but modern web applications are like dining in person at a sit-down restaurant. A restaurant patron might order drinks, different courses, make substitutions, or ask questions of the waiter. To accomplish this level of complexity, an equally complex back-end is required.

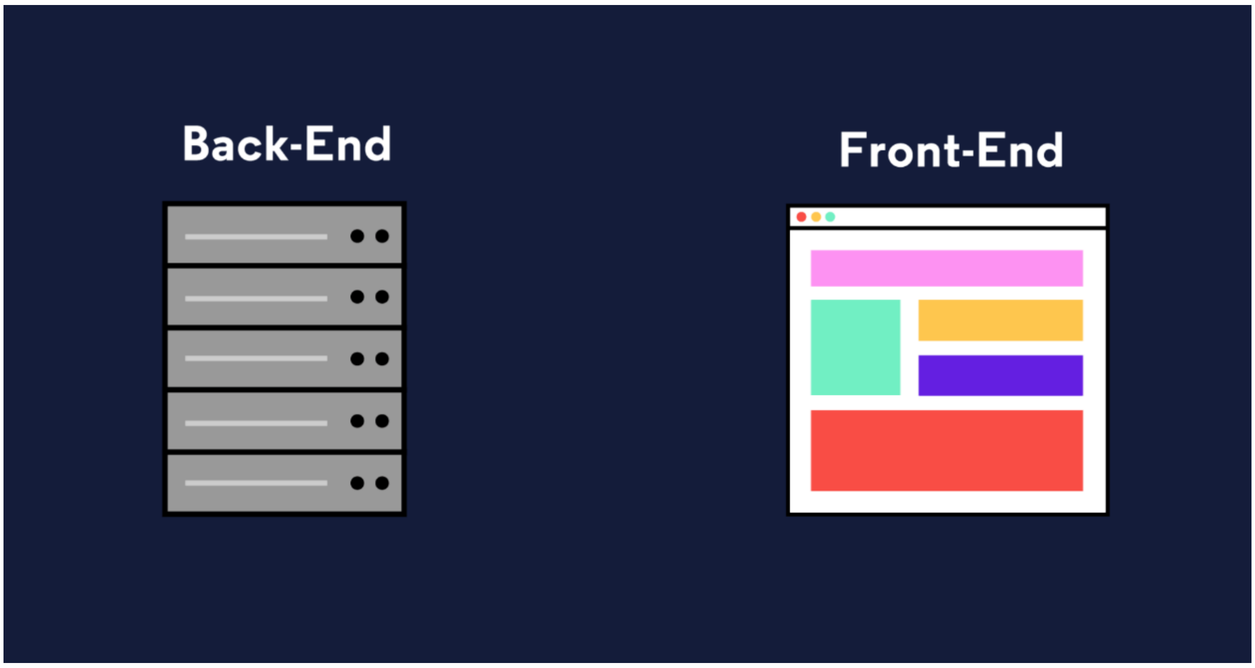

Back-End
Front-End
Request


Back-End
Front-End
HTML


Back-End
CSS
Front-End


Back-End
JS
Front-End


Back-End
Front-End
Response


Back-End
Front-End