Eric Price's Technical Writing Answers
8-23-18

1) From start to finish how does that data reach you to be rendered in the browser? So what happens, exactly?

When you type a web address into your browser, the browser goes to the DNS server and finds the real address (IP address) of the server that the website lives on (you find the address of the shop). First, the browser sends an HTTP request message to the server, asking it to send a copy of the website to the client (you go to the shop and order your goods). This message, and all other data sent between the client and the server, is sent across your internet connection using TCP/IP.
Next, provided the server approves the client's request, the server sends the client a "200 OK" message, which means "Of course you can look at that website! Here it is", and then starts sending the website's files to the browser as a series of small chunks called data packets.
Next, the browser assembles the small chunks into a complete website and displays it to you.
A website is made up of many different files, which are like the different parts of the goods you buy from a shop. These files come in two main types, Code files and Assets
Code files: Websites are built primarily from HTML, CSS, and a scripting language, such as JavaScript.
Assets: This is a collective name for all the other stuff that makes up a website, such as images, music, video, Word documents, and PDFs.

2) What code is rendered in the browser?

HTML, CSS, JavaScript and Assets (images, music, video, Word documents, and PDFs) are rendered in the browser. Image content from a server is interpreted based on web standards and specifications
The rendering engine has a very crucial job as it displays what you see on your screen. It communicates with the networking layer of the browser to grab HTML code and other items passed from a remote server. Then it follows these steps:
1. Parsing HTML and creating the DOM Tree: These HTML elements are parsed and turned into a "DOM tree" by the rendering engine. It is a tree like structure made out of the HTML, where each tag is a branch starting at the root element.
2. Render Tree Construction: CSS attributes are parsed and combined with the DOM tree to create a "render tree". This is a tree of visual elements such as height, width and color ordered in the hierarchy in which they are to be displayed in the browser.
3. Layout Process: Once the render tree is constructed, the rendering engine recursively goes through the HTML elements in the tree and figures out where they should be placed on the screen.
4. Painting: Each node (branch) of the render tree is drawn out on the screen by communicating with the Operating System Interface which contains designs and styles for how UI elements should look.
Sending the website's files to the browser as a series of small chunks called data packets.
JavaScript is a programming language which allows things to happen inside of the browser and makes web pages interactive. Things like popup windows, actions that occur on a button press and elements that move across the page are all things accomplished with JavaScript. This means that JavaScript code executes after the web page has been rendered and painted onto the screen, and when it executes it triggers a re-render to account for changes made.

3) What is the server-side code's main function?

The server-side code's main function is to manipulate the data and to keep the application running using CRUD (Create, Read, Update and Delete) via Post, Get, PUT, PATCH and Delete. Server-side website programming mostly involves choosing which content is returned to the browser in response to requests. The server-side code handles tasks like: validating submitted data, authentication and requests, using databases to store and retrieve data and sending the correct data to the client as required. Most large-scale websites use server-side code to dynamically display different data when needed, generally pulled out of a database stored on a server and sent to the client to be displayed via some code (e.g. HTML and JavaScript).
Maybe the most significant benefit of server-side code is that it allows us to tailor website content for individual users. Dynamic sites can highlight content that is more relevant based on user preferences and habits. It can also make sites easier to use by storing personal preferences and information. It can even allow interaction with users of the site, sending notifications and updates via email or through other channels. All of these capabilities enable much deeper engagement with users.

4) What is the client-side code's main function?

Client-side code is primarily concerned with improving the appearance and behavior of a rendered web page. This includes selecting and styling UI components, creating layouts, navigation, form validation, etc. Client-side code is written using HTML, CSS, and JavaScript and it's run inside a web browser that has little or no access to the operating system.
Web developers can't control what browser every user might be using to view a website and browsers provide inconsistent levels of compatibility with client-side code features. A challenge of client-side programming is handling differences in browser support gracefully.
Next, I'd like to compare client-side and server-side just so you know I know the difference and that you can connect the two. The main advantage to server-side scripting is the ability to highly customize the response based on the user's requirements, access rights, or queries into data stores. The client-side environment required to run scripts is usually a browser. The processing takes place on the user's computer. The source code is transferred from the web server to the user's computer over the internet and run directly in their browser. The scripting language needs to be enabled on the client's computer. Occasionally, if a user is conscious of security risks they may switch the scripting facility off. When this is the situation, a message usually pops up to alert the user when script is attempting to run. The server-side environment that runs a scripting language is a web server. A user's request is fulfilled by running a script directly on the web server to generate dynamic HTML pages. This HTML is then sent to the client browser. It is usually used to provide interactive web sites that interface to databases or other data stores on the server. Again, this is different from client-side scripting where scripts are run by the viewing web browser, usually in JavaScript.

5) How many instances of the client-side assets (HTML, CSS, JS, Images, etc.) are created?

One instance of the client-side assets is created upon loading and all of the assets can be seen via sources after inspecting the home page. An instance is defined as a case or occurrence of anything. This could be an element, document type, or a document that conforms to a particular data type definition. For each client-side asset, such as a GET request, one instance is created. This is because only one file (or one of something else) can be rendered per GET request.

6) How many instances of the server-side code are available at any given time?

As much as is available and it depends on the functionality written into the code. There can only be one output per input, but for the server-side code, there are multiple outputs for one input. A dynamic website is one where some of the response content is generated dynamically only when needed. On a dynamic website, HTML pages are normally created by inserting data from a database into placeholders in HTML templates. Dynamic sites can return different data for a URL based on information provided by the user or stored preferences and thus can perform other operations as part of returning a response. Sending notifications is an example. Most of the code to support a dynamic website must run on the server. Client-side web frameworks simplify layout and presentation tasks while server-side web frameworks provide a lot of “common” web server functionality that you might otherwise have to implement yourself. For example, support for sessions, support for users and authentication, easy database access and templating libraries.

7) What is runtime?

For a number of years, technical writers resisted "runtime" as a term, insisting that something like "when a program is run" would obviate the need for a special term. Gradually, the term crept into general usage. Runtime is when a program is running (or being executable). That is, when you start a program running in a computer, it is runtime for that program. In some programming languages, certain reusable programs or "routines" are built and packaged as a "runtime library." These routines can be linked to and used by any program when it is running. Programmers sometimes distinguish between what gets embedded in a program when it is compiler and what gets embedded or used at runtime. The former is sometimes called "compile time."

8) How many instances of the the databases connected to the server application are created?

Unlimited instances. A lot of this depends on what you’re inputing into the database. Each instance manages several system databases and one or more user databases. Each computer can run multiple instances of the Database Engine independently of other instances. The database is the set of files where application data (the reason for a database) and meta data is stored. An instance if the software and memory that the program uses to manipulate data in the database.