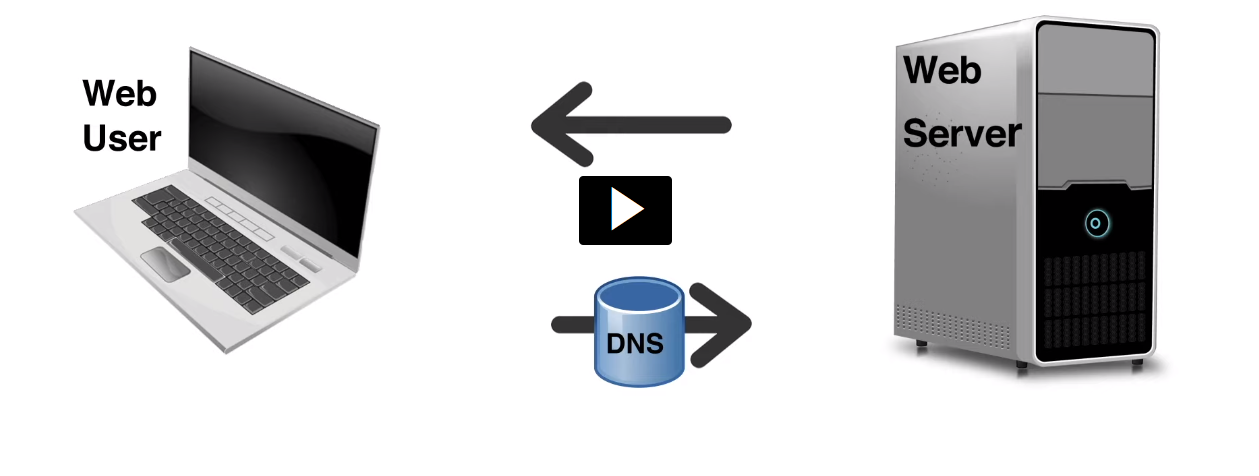
Let’s go over some basics that are good to know as we begin studying Web Development. Let’s start with the Internet. **The Internet is a global network of computers that can talk to each other**. They all use the same rules, or protocols, so they can communicate. The Internet protocol is called ***TCP/IP. It stands for: Transmission Control Protocol/Internet Protocol***.

**The World Wide Web is a whole lot of resources like documents and applications on the Internet.** These documents can be linked together with hypertext links that when clicked can take you to other documents and applications on the web. The protocol used on the web is called **HTTP**. It stands for: **Hypertext Transfer Protocol**.

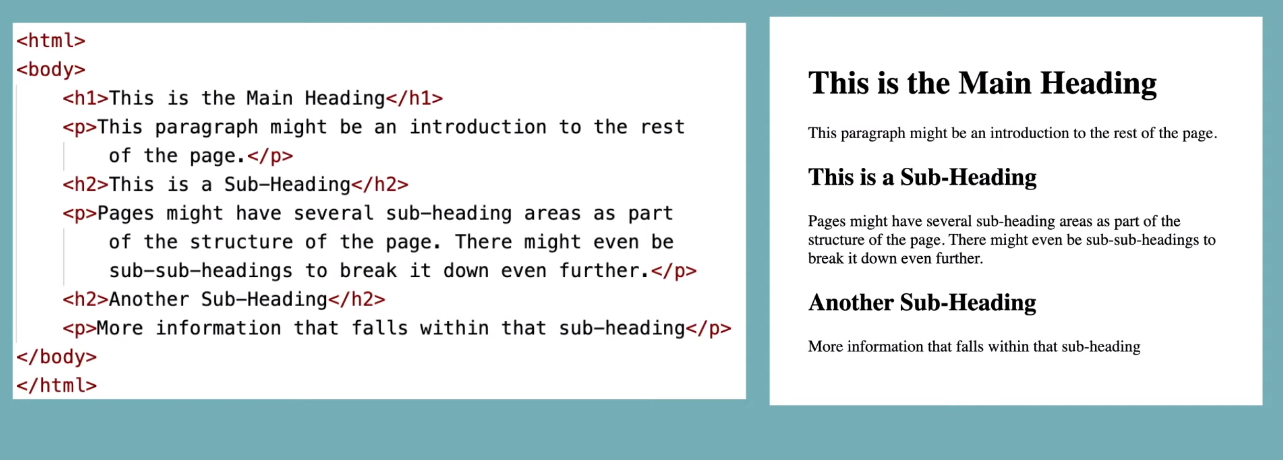
When a web user opens a website or web application a request is made to the web server that holds those web files**. The user and the browser they use (like Chrome, Edge, or Safari) is referred to as the client on the front-end**. **The server is the back-end.** The user might type in a web address, also called a **URL** or **uniform resource locator**, or they might click a link or open a bookmark. The right server will then get the request because of this URL or web address. This URL matches up to an **IP address** of where the website or app files are located on the server. This is done with the **DNS Domain Name System**. A system where all web addresses or domain names are matched up with the server location of the files requested. The server then sends the files that were requested back to the client. 

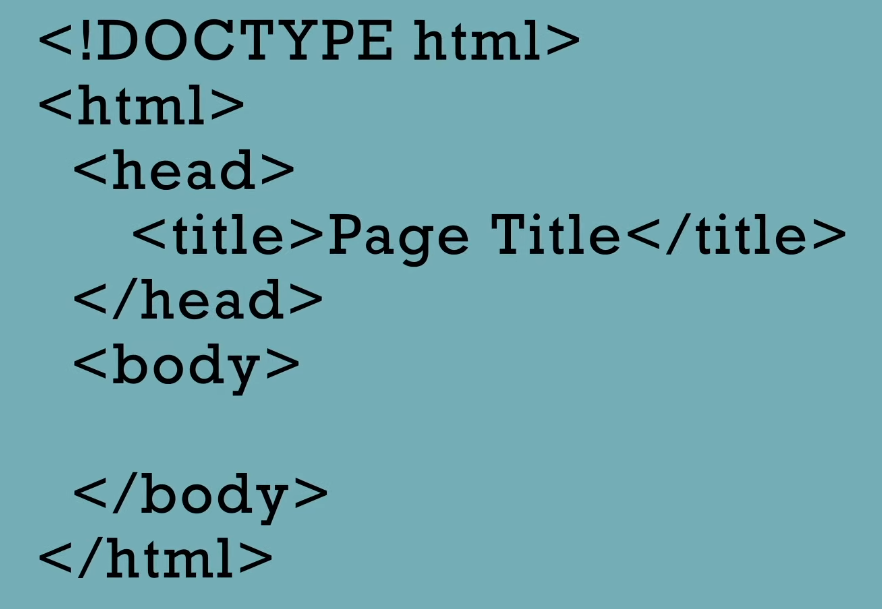
Web developers often use hosting companies that already have servers in place to store their files. Then they might purchase a domain name such as mycompany.com and register that domain name with the IP address of the hosting company’s servers where they store their files. Developers could also use their own web server if they don’t use a hosting service. Different languages or scripts run on the client (front-end) and the server (back-end). **HTML and CSS are front-end languages**, the code runs on the client or user’s computer**. PHP and Python are examples of back-end languages, the back-end code runs on a server before it gets to the user**. Even though the HTML and CSS are front-end languages, the actual files are stored on the back-end server until requested, they just aren’t processed or interpreted there. When a request is made the files are sent to the client. Then browser that is being used, will run or interpret the code in the files and present the website or application on the browser’s screen. So, the device that made the request has to interpret or render the pages or application that is sent. This is referred to as front-end processing. Things like logging a user in, processing data from a form, or using a database with a website would be examples of back-end processes and that is where the backend languages would be used. When considering the users of our sites and applications, it’s good to keep in mind that many different devices are going to be used to access our web files, like phones, tablets, laptops, etc. And they all have different connection speeds. Not only do the files like HTML and CSS have to be sent, but all the images and videos that go with them. The goal is to make our sites as small and efficient as possible and still look and run well on any device. Also, some users may use screen readers that will read the site to them. Often these are visually impaired users. Developers must also take into account the needs of these users and make sure their code is written with accessibility in mind. So there we have just a brief overview of some web development basics.

---------------------------------------------------------------------------------------------

What is HTML?

With any document we read, the structure of the page helps us understand the message. The headings and subheading give some structure to this page. It’s HTML’s job to give structure like this to a web page. This HTML code shows a few different elements that make up the structure of this page.



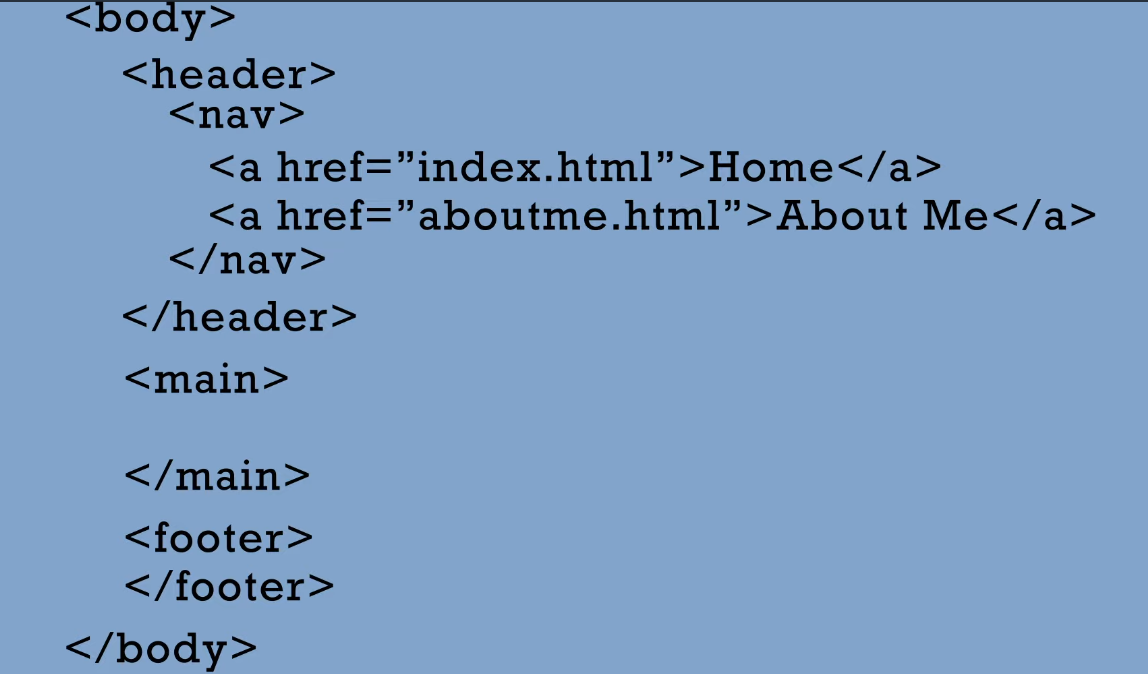
**The HTML elements are shown in red with angle brackets.** They usually have an opening and closing tag(<>). The closing tags have a slash after the first angle bracket(</). **Each element tells the browser what kind of information is between its opening and closing tag**. Some tags contain other tags. The html tag has all the other tags between its opening and closing tag. **The body tag** contains all the tags that will display in the page of the browser. **This is called nesting. You can have tags inside of other tags or parent and child tags.** The text between the h1 tags, or heading level 1 tags, is the main heading. h2 tags are one level down from the main heading, in this case they are used as subheadings and the text between the p tags is the paragraph text. **HTML stands for hypertext markup language.** Hypertext refers to the way we can place hypertext links in our document that allows our users to move from one page to another**. Markup is a set of symbols or codes for displaying content on the Internet. It’s not a programming language, it’s a markup language** **that tells web browsers (like Chrome, Safari, or Edge) the structure of a web page's words and images**. This markup language includes the tags we make to code around the content, then the browser will use this code to display or render our pages correctly. This is a simple HTML framework or beginning of a HTML file or page. 

Notice there is nothing inside the body tag yet, so nothing would actually show up on the browser page. But these HTML elements begin an HTML file. An HTML file can be created in any simple text editor, (like word). There is no need for any special program to write HTML code. There are good code editors, however, that can be used that might be helpful when writing your code (VScode and others). **The first line here lets the browser know which version of HTML is being used**, it is the document type declaration. This declaration lets the browser know to expect an HTML version 5 document. The html element is the container for all other html tags. **The head** element contains elements that tell about the page, rather than what’s on the page. The head here has one title element that contains the text ‘Page Title’. **That text will show up in the title bar or tab at the top of the browser window not on the page itself.** The **body** tag is where **all the elements that make up the page will go**. If you look at the source of every web page you will see this HTML structure.

Let’s take a look at a few different types of elements. We’ve seen <**h1>, <h2> and <p>** elements. Let’s look at a few more. There are 3 common elements you might see inside of the body. **They are common parent elements for all other elements**. They are the <**header>,<main> and<footer> .** They are the child elements of **<body**>.

When you have child elements its good practice to indent them inside the parent element. Indenting each child element helps the HTML code be more readable. The <**reader**>element is different from the<head> element. The<**header**> **element will show up on the browser window. This is where elements at the top of the web page will be placed**. **Everything in the <header> will remain the same from one web page to anothe**r. Things like a company logo or a web menu would be items that would belong in the **<header>. The <main> element will hold web page content that is unique to each page**. And the **<footer>element will hold elements that will show up at the bottom of every page of the website.** Things like copyright or contact info or maybe more links. Like the <header> element, the <footer>elements will remain the same from one web page to another.

A common child element inside the<header> is the <nav>element. The <nav> element is used to hold the menu. It defines a set of navigation links. It can hold different types of elements, but it will usually hold<**a**> elements.

An <**a**>**element defines a hyperlink. Meaning when the user clicks on the text or content between the <a> tags, it will take them somewhere else.** Additional information is needed inside the opening tag to specify where to send the user once they click the content of the **<a>** element. Additional information inside an opening tag is called an attribute. Attributes usually come in name/value pairs, with the name followed by an equal sign and then the value in quotes. 

This is an attribute with the name of **href** and the value in quotes of the URL or web address of where the link will take the user**. Href stands for hypertext reference**. Notice that tags and attribute names are **lowercase. HTML is not case sensitive but it’s good practice to leave them lowercase**. These **<a>**elements are nested inside the <nav>element and so they are the children of the <nav> .

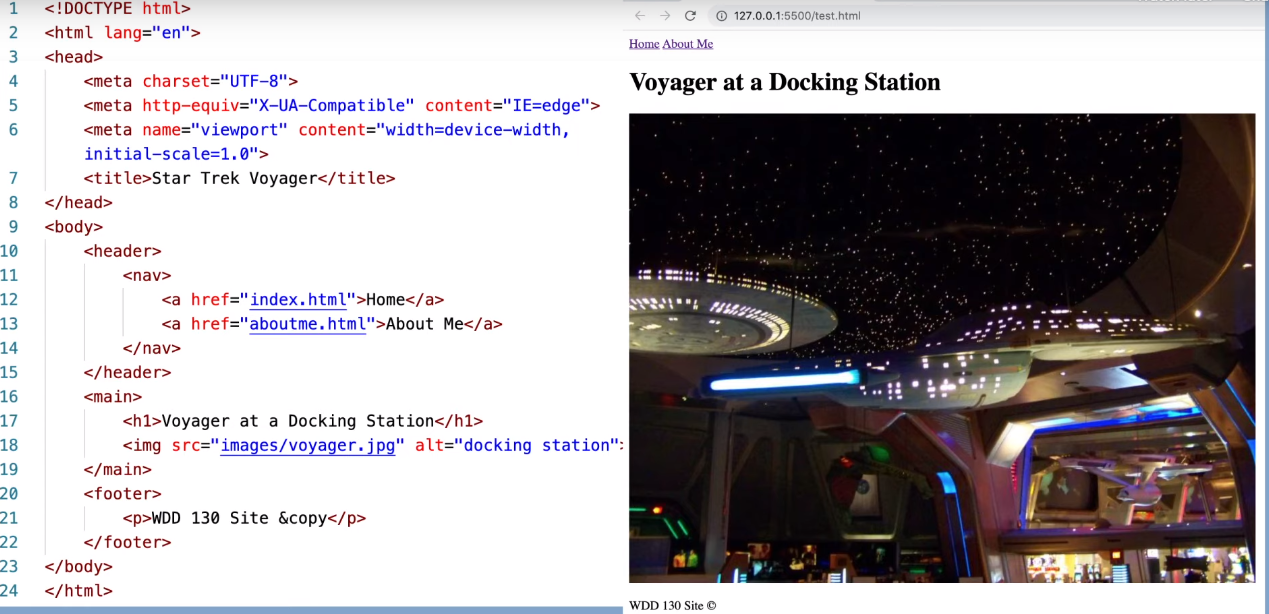
Here’s an **img** or image element. This is one, of only a few elements, that doesn’t have an open and closing tag.

 That’s because there is no content that is needed between tags. All the information to show the image is handled by the attributes. This image element has two attributes the **src** or source attribute and the alt or alternative text attribute. The **src** attribute has the value of the path of where the image is located along with the name of the image. The **alt** **attribute shows alternative text that will be used by screen readers or if the image is not available(error message)**. The name part of the name value pairs are src and alt. The values are in the quotes following the equal sign. Here is an html file as you’d see it in a code editor. Some editors will color coordinate different types of code. This is showing the Visual Studio Code editor.



**The tags are a dark red and the attribute names a bright red with the values of those attributes as blue.** Take a look at the html element. It has an attribute called **lang** that indicates the language that will be used for the content of html. In this case it is English. We see that our HTML skeleton has meta elements in the head section (we’ll learn more about **these later).** This is what our html page would look like after the browser renders or interprets the code. It’s interesting to note that the

# <h1>element and <a>element look very different. The <h1>is large and bold and the <a>link is smaller and underlined with blue or purple text depending on if the link has been visited or not. The way these elements look after being rendered is not the job of HTML that is the job of CSS. HTML handles the structure of the page and CSS handles how it is presented or how it looks. The reason the h1 is big and bold and the link is smaller, colored, and underlined is because our browser has some internal default CSS that is being applied to different types of elements. For example, most browsers have default CSS for h1 elements of being bigger and bolder than other text. We will learn more about CSS later and how we can override the internal browser CSS defaults to change things to look a certain way. But remember that is not the job of HTML. HTML is just about structure—knowing what element to use for what content and how to organize that on our page. For HTML the <h1> element simply means that it is the main heading of the page and to see its content as such. This is referred to as semantics. Meaning you can tell what type of content it is because of the element used. Semantic tags describe the content you are adding to the page. An <h1> will be a main heading, an <a> tag will be a link, an <img> tag will display an image, etc.



Esqueleto como fica

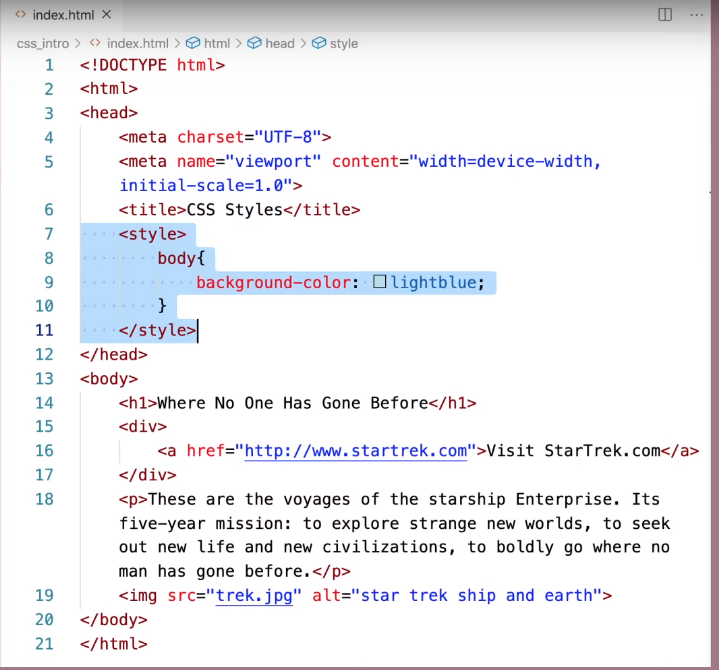
What is CSS?

CSS is the language used to style an HTML document. **CSS is a separate language from HTML**. It is a rule-based language with its own syntax (or way of typing the code). **It describes how the HTML should be displayed**. It’s what will make our web pages more attractive. Remember HTML is about structure and what element is used for what content, and **CSS is all about presentation and how the elements on the page will look and how they will be laid out on the page**. A great example to see the power of CSS is on csszengarden.com . Every page featured uses the exact same HTML but with different **CSS** applied to it. **CSS stands for Cascading Style Sheet s**. The term ‘cascading’ refers to the fact that one style rule can be set up for multiple elements and even work on multiple pages of your website. This makes it very efficient and requires less coding to style your pages. Cascade can also refer to the hierarchical order in which different styles interact with elements. More on this later. Here is a page rendering with just HTML, no CSS. Here is the same page with CSS applied to it. There are a few different ways we can use CSS with HTML. The first is with inline CSS. This is not the preferred way because it takes out the benefit of reusing our CSS rules in many different areas and can be hard to edit later.

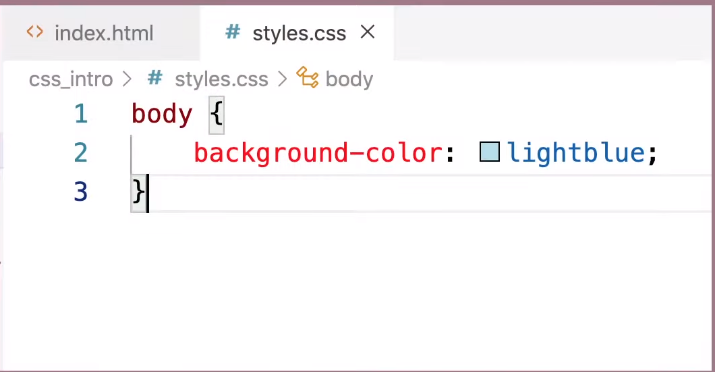


For example, if I wanted to make the background of my entire page a light blue, I could go into the body element’s opening tag and add style as an attribute. As is shown here. This is referred to as **inline CSS**. I would avoid using inline CSS if possible.

Another way to apply CSS to your HTML is to embed the CSS in the <head> section of your HTML page inside of <style> tags. Again, this is also not the preferred way because you can’t reuse your CSS on multiple pages of your website.



The best way to apply CSS to our HTML is in a separate CSS file. **You should always keep your HTML and CSS separate if possible. They both have different purposes and it’s best to keep them in separate files**. Also, other advantages, of having a separate CSS file, is so **different web pages can share the same style sheet, then the same code wouldn’t have to be repeated on every page**. It makes less code and faster load times. Also, **if you have a style change to make on several pages, the CSS would only have to be changed in one place.**

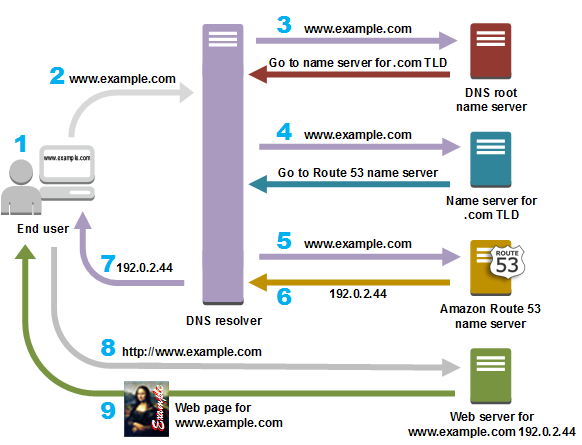
****

The HTML<link> element is used to tell our HTML page which CSS file to use to style our page**. You can have multiple CSS files effecting one HTML page but our site will be simple enough for just one CSS file**. Notice the <link> element has no closing tag. Everything it needs is inside the tag. **The link element is placed in inside the head section of our page**. It has two required attributes that show the relationship of the linked file and the hypertext reference of what CSS file we are using. Now we can type all our CSS in the external file and it will apply to the HTML.

Fundamentos do DNS

Todos os computadores da internet, abrangendo de smartphones ou laptops a servidores que distribuem conteúdo para grandes websites do comércio, se encontram e se comunicam entre si usando números. Esses números são conhecidos como endereços IP. Ao abrir um navegador e acessar um site, você não precisará lembrar-se de um longo número nem digitá-lo. Em vez disso, você poderá informar um nome de domínio, como exemplo.com, e ainda assim encontrar o que deseja.

Um serviço DNS, como o Amazon Route 53, é um serviço globalmente distribuído que converte nomes legíveis por humanos, como www.exemplo.com, em endereços IP numéricos, como 192.0.2.1, usados pelos computadores para se conectarem entre si. O sistema DNS da internet funciona praticamente como uma agenda de telefone ao gerenciar o mapeamento entre nomes e números. Os servidores DNS convertem solicitações de nomes em endereços IP, controlando qual servidor um usuário final alcançará quando digitar um nome de domínio no navegador da web. Essas solicitações são chamadas consultas.



1. Um usuário abre um navegador, digita www.exemplo.com na barra de endereços e aperta Enter.
2. A solicitação de www.exemplo.com é direcionada para um resolvedor DNS, que geralmente é gerenciado pelo ISP (Internet service provider – Provedor de serviços de internet) do usuário, como um provedor de internet a cabo, um provedor de banda larga DSL ou uma rede corporativa.
3. O resolvedor DNS do ISP encaminha a solicitação, que sai de www.exemplo.com e passa para um serviço de nome raiz DNS.
4. O resolvedor DNS do ISP encaminha novamente a solicitação de www.exemplo.com, mas desta vez para um dos servidores de nome TLD de domínios .com. O servidor de nome dos domínios .com responde a solicitação com os nomes dos quatro servidores de nome do Amazon Route 53 que estão associados ao domínio exemplo.com.
5. O resolvedor DNS do ISP escolhe um servidor de nome do Amazon Route 53 e encaminha a solicitação de www.exemplo.com para o servidor de nome em questão.
6. O servidor de nome do Amazon Route 53 procura na zona hospedada exemplo.com pelo registro de www.exemplo.com, obtém o valor associado, como o endereço IP de um servidor web (192.0.2.44) e retorna o endereço IP ao resolvedor DNS.
7. Por fim, o resolvedor DNS do ISP obtém o endereço IP de que o usuário precisa. O resolvedor retorna este valor para o navegador. O resolvedor DNS também armazenará em cache o endereço IP de exemplo.com durante um período que você especificará para que ele possa responder mais rapidamente da próxima vez que alguém pesquisar por exemplo.com. Para obter mais informações, consulte a TTL (time to live – vida útil).
8. O navegador enviará uma solicitação de www.exemplo.com para o endereço IP que obteve do resolvedor DNS. O seu conteúdo poderá ser encontrado lá. Por exemplo, um servidor web em execução em uma instância do Amazon EC2 ou um bucket do Amazon S3 que seja configurado como um endpoint do site.
9. O servidor web ou outro recurso em 192.0.2.44 retornará a página web de www.exemplo.com para o navegador que, por sua vez, exibirá a página.