1) The first thing when a user types a URL in a browser is that the browser recognizes the various parts of the URL. For example, if the URL has http in it, the browser recognizes it as the protocol to use when communicating with the server. It then separates out the other parts and extracts the actual domain name (ex: google.com) from the URL. Once it has the domain name, it will ask the ISP's DNS server to initiate a DNS query, meaning it will ask the DNS database for the particular IP address associated with the domain name. After this query, the client has access to the destination IP address. It can now use that to connect to the server which the IP address belongs to using whatever protocol it happened to get from the URL. Then, it will initiate a GET request to the server (which usually contains the protocol and the path from the URL). Once the server locates whatever path is requested by the GET request, it will typically respond with a confirmation for the protocol (this signifies that the server actually understood what protocol the GET request specified) and a code (there are various codes which mean different things) signifying it found the content [1]. It will usually respond with a 404 Not Found error if it couldn't find the content/resource requested. It then responds with the HTML page requested and then the browser will display the HTML page.

   In essence:

   Browser -> DNS Query -> TCP Connection -> Get Request -> Server handling -> Server Response -> Browser Display

   Once the client (browser) receives the response it will build a DOM object to work with, so it can map out the elements and their properties. Once the DOM is created, it will display the HTML code, then apply the CSS associated with it, and then load the JS. The order is usually HTML -> CSS -> JS (assuming the script tag is at the end of the body; if not, then it will load depending on where it's placed in the file). So, the browser usually loads things from top to bottom in the file. So, if a script tag is in middle, it will usually stop the HTML processing and get the script and process it before continuing with the HTML processing [2].

   In essence:

   HTML structure -> Place elements -> Apply CSS -> Map out JS (DOM) -> Process JS

2) When designing a website, one should consider the following:
    - UI – The website should look clean, well laid out, and consistent. The colors should be complementary, the font should be easy to read, and everything should be neatly structured, so it’s not cluttered. You should have a good balance of text and images (and images should be places appropriately and the size should be set so it’s not blurry).
    - UX – Navigation must be made intuitive and any content should be easily reached quickly (search bars are great for this). The website should be interactive so the user will be interested in exploring your website. The pages should load fast; no one like slow websites. Website should have good content (websites with good content and load times will usually be better than a slow website with lots of effects and animations).
    - Accessibility – The website should use web safe colors and large enough fonts and legible fonts so users can easily read the content. Include a “dark mode” toggle so users with bad eyesight or users accessing the website at night will have a good experience. Always include alternative captions for images and media, in case they don’t load so there’s some description there instead of blank spaces.
    - Responsiveness – The website should be responsive, that is, it should work well on any device with any screen size. The layout and experience should be optimized for different screen sizes (ex: you might want to display background videos on desktop but use a static image on mobiles). The content should be resized, and nothing should be out of place.
    - General considerations – Place scripts at the end of the document so it doesn’t load first and slow down the display of the content. Try compressing images so web sites are not slow. Add hover and active properties to links and nav bars so it’s not plain and confusing. Divide content into sections, don’t add too much text/images, clearly label each section, and don’t overdo the effects and animations.

I believe that content should be created first since without knowing that should go in the website, all you will have is the design. You will then have to trim useful content or add useless content to fit your design. Nowadays content is extremely important for SEO and site rankings so that’s also another reason to create content first. Once content is created, you can carefully plan the layout and structure of the page. You will know if you have a long paragraph, you can include a “read more” option rather than making a small box for the content and be stuck with it or edit it later. Also, if you have lots of images, you might design a separate gallery page rather than creating a layout with specific image locations and have to end up not adding all the images.

I don’t think one developer should handle both content and styling. You will want a professional content writer for the content and then a developer should create the layout and design the webpage with the content.

References:

[1] Codecademy, HTTP Requests. “HTTP Requests | Codecademy.” *Codecademy*, 2019, www.codecademy.com/articles/http-requests.

[2] “How Browsers Load and Process JavaScript.” *Innoq.Com*, 16 Mar. 2017, www.innoq.com/en/blog/loading-javascript/.