


☐

I'm not robot


reCAPTCHA

Continue

Html and css for dummies

Freelance Full-stack DeveloperThis article helps you to start in CSS3, explaining the basics of how to display elements in an HTML document. When I started working as a software developer, it was boring for me to encrypt the web user interface. My weakness has always been CSS because it has been difficult for me to do well showing elements in an HTML document. I had some experience using CSS (version 2), but I was disappointed by the compatibility in the browser. When I have to work with the web user interface, I just copy some layouts that can be found on the Internet. Then it changed when I knew Bootstrap. 1. My previous experienceA. What did I have? As I said before, I just copied some nice layouts that I could find on the Internet, until the day I knew Bootstrap. With Bootstrap my work has been easier because CSS classes are compatible with all browsers (even IE8), there are decent templates and it can be used with JQuery to make a useful user interface. B. It's all easy with a framework and I'm importantI've been working for a school as a software developer, meanwhile, I've got some customers who want electronic invoice software for their companies. So when I started these projects, I used Bootstrap for the UI with some changes and I always applied I'm important if a class doesn't work. C. ConsequentlyThe projects of my clients are growing, so I am glad that I make more money. The downside is that I had to change my interface a lot according to the client's requirements. Using Bootstrap and my interface plugins was too heavy, slow and many times with bugs.2. Practice I did it!A. GridWith Grid, we can assemble the layout (or skeleton, as I call it here) of our sample. We just put: Header, Footer, Main and a Left Sidebar (Aside) as a puzzle. The key to using Grid is to know which elements are parents (as containers) and which elements are children. So the example below shows a father container and three children. <!DOCTYPE html><html lang=en><head><meta charset=UTF-8> <meta name=viewport content=width=device-width, initial-scale=1.0><title>Documents</title><style> h1 { margin: 0; padding: 0; } .grid-father { display: grid; grid-template-columns: 1fr 1fr 1fr 1fr; gap: 5px; } .grid-child-1 { grid-row: 1; grid-column: 1; background-color: green; height: 150px; } .grid-child-2 { grid-row: 2; grid-column: 2; background-color: orange; height: 150px; } .grid-child-3 { grid-row: 3; grid-column: 3; background-color: red; height: 150px; } </style></head><body><div class=grid-father><div class=grid-child-1><h1>Row 1 - Colonel 1</h1></div><div class=grid-child-2><h1>Row 3 - Col 3.34</h1></div></div></body></html>Run the previous example in our browser. We can see how children show up inside the father: The father is a Grid with 4 columns in which each fr represents a fair share of the screen. The first child is displayed on the first column and the first row because its class has two obvious attributes (grid row: 1; grid column: 1); also applies to the second child and its properties (grid row: 2; grid column: 2). Especially in the third child, located in the third row and the third column, but it occupies 2 columns because the grid column attributes defined with the number of columns to expand.B. FlexboxS using Flexbox can place several elements in different positions and displays. It was very important to me because many times I had problems with the vertical and horizontal positions of divs, stretching, and images. Grid, for example, with Flexbox, the key is to know where parents and children are: Display effects only apply from parent to child, not to your child. So the next example shows three types of Flexbox displays: rows, wraps, and columns. <!DOCTYPE html><html lang=en><head><meta charset=UTF-8> <meta name=viewport content=width=device-width, initial-scale=1.0><title>Hòp flexbox</title><style> h1 { padding: 0; margin: 0; } .square-container { padding: 5px; margin: 5px; border: 2px solid #000; text-align: center; height: 50px; width: 350px; } .flex-row { display: flex; flex-direction: row; background-color: gray; color: #000; margin: 5px; } .flex-wrap { display: flex; flex-direction: row; flex-wrap: wrap; background-color: cadetblue; color: #000; margin: 5px; } .flex-column { display: flex; flex-direction: column; background-color: darkgoldenrod; color: #000; margin: 5px; } </style></head><body><div class=flex-row><div class=square-container><h1>Hàng Flex</h1></div><div class=square-container><h1>Hàng Flex</h1></div></div><div class=square-container><h1>Hàng Flex</h1></div><div class=flex-wrap><div class=square-container><h1>Hàng Flex</h1></div><div class=square-container><h1>Hàng Flex</h1></div><div class=square-container><h1>Hàng Flex</h1></div><div class=square-container><h1>Hàng Flex</h1></div></div><div class=flex-column><div class=square-container><h1>Boc flex</h1></div><div class=square-container><h1>Boc flex</h1></div><div class=square-container><h1>Boc flex</h1></div><div class=square-container><h1>Boc flex</h1></div></div></body></html>If we Previous code, it can be seen the difference: Using element display attribute with flexbox as row (flex-direction: row), all internal elements occupy a position in the same line.

