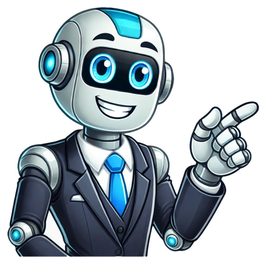
Click to verify



Modern HTML FormWith the proliferation of free services in exchange for personal information, a modern HTML form has become more important in the capture of email addresses, passwords, usernames, etc. The inability to offer a seamless and pleasant experience in the informational exchange might cause your users to throw up their hands and rethink their want to use your service. In addition, today's web is leaps and bounds better looking than years before. With more and more website builders becoming mainstream and offering modern themes it'll be harder to convince users that your product is exciting if your HTML form is bland and unthoughtful. In this tutorial, I'll show you how you can go from this …Simple HTML FormTo this …Mordern HTML FormPlan Of AttackThe steps for this tutorial fall into three categories:Create HTML SkeletonPosition HTML FormStyle HTML FormThankfully forms aren't all that hard to write so we can get to the fun stuff (Styling the HTML form) quicker. However, there are a couple of things we need to discuss before we rush off to do our styling. Let's begin…To start off we'll just create the basic structure of an HTML Document A Better Form Once we have the basic structure down for our HTML Document lets build out the form within the body tag… Sign Up First name:

Last name:

Favorite Color:

Favorite Food:

function formSubmit(event) { event.preventDefault(); }…Nothing too interesting here. We've created a top-level container named "formContainer" to hold our form. This will be essential for step two when we try positioning the form. In regards to our form, we have specified that when we click the submit button we want to call "formSubmit(event)" with the event that triggered the function. Within the function, we just call "event.preventDefault();" so that if we accidentally hit the submit button our page doesn't get refreshed. Just a minor thing, not really important to the overall purpose of this article.Your form should now look something like this …Step 2: Position HTML FormHere is the portion which people might have the hardest time getting their form to look just right. However, a lot of layout issues/hacks can be resolved with CSS Flexbox and that's what we'll be using today.We'll be centering the form both vertically and horizontally. This is easily achievable with just three attributes: "display: flex", "justify-content: center", "align-items: center".Our first attribute "display: flex" signifies that we'll be using CSS Flexbox. The next one "justify-content: center" horizontally centers the element within our form container. Lastly, we have "align-items: center" which vertically centers our content.Thumbs UpLet's actually apply this to our code. At the top of the CSS file add this …body { display: flex; justify-content: center; align-items: center;}You might notice that the form doesn't seem to be quite centered. That's because of the height of our body tag. Let's fix that with one line …body { . . . height: 100vh; }Cool. Now we have a vertically and horizontally centered form. The next thing we'll do is start tweaking the layout of our form container.Go ahead and add this to the bottom of your CSS file ….formContainer { display: flex; justify-content: center; align-items: center;}You will now see that the Sign Up header is to the left of our form instead of on top. This is actually expected. The way CSS Flexbox handles its elements by default is putting each child element in a row type direction. We can change this behavior with the "flex-direction: column" attribute. Add it to the bottom of the formContainer class like so ….formContainer { display: flex; justify-content: center; align-items: center; flex-direction: column;}Awesome so now we should see the Sign Up header at the top of our form.With that addition, we have a final product that looks like this …Step #3: Style HTML FormStyling A SuitHere is the real focus of our article. Spicing the form up with some extra CSS styles.Because we'll be switching the font color to white let us change our background color first.A really good way to spruce up your form's page is to have a compelling high-quality background image. We can achieve this with just linear gradients and a selection of complementary colors. For the color selection, I'll be using www.material-ui.com/style/color/I generally find a diagonal gradient to be best. Go ahead and drop these three lines in our body classbody { . . . background: linear-gradient(to top left, #4A148C, #C51162); background-repeat: cover; color: white;}Because minimalism is a great way to make your design feel modern lets get rid of every line in each of our input boxes except the bottom one, creating a nice white underline. Add this to the bottom of the CSS fileinput { margin-bottom: 5vh; border-right: none; border-top: none; border-left: none; background-color:transparent; outline: none; color: white; caret-color: white;}And because we don't want any wasted space lets go ahead and change up our form to use placeholder text for the inputs titles

This is good but we still need to style the placeholder text as it doesn't look good grayed out. Place this block at the bottom of your CSS file.input[type=text]::placeholder { text-align: center; color: white; font-family: 'Roboto', sans-serif;}Almost ThereWe are almost done.The next item we need to change is the submit button. Add this to the bottom of your CSS fileinput[type=submit] { background: rgba(255,255,255,.2); width: 100%; padding: 2vh; border: none; cursor: pointer; border-radius: 4px 4px;}Adding opacity to your buttons can make them blend in more to the background yet still have some sort of separation. We also add in a change to the buttons borders. Essentially sanding off the edges to make it a smoother design.Our final adjustment is to make sure our form is always a relative size to the screen. Add this to the bottom of the CSS file ….formContainer { . . . height: 80vh; width: 20vw;}This gives us an upright rectangle which suits the column layout we specified for our usage of CSS Flexbox.Our final product should look like this …Spongebob Dusting Off HandsConclusionLike stated earlier, forms are becoming more essential as we transition to a world of information collection. A good way to entice your users is to make sure when you are asking them for their personal information the process is as pleasant and frictionless as possible. Creating modern beautiful looking forms can help alleviate any reservations a person might have about forking over private data. You could go so far as to inject CSS animations into your form if you wanted too. However, with this tutorial, you should now at least have an understanding of what goes into making a simple modern HTML form. In web development, forms are essential because they enable users to engage with websites by providing information like survey answers, feedback, and login credentials. A well-designed form guarantees smooth data gathering, increases accessibility, and improves user experience.In this blog, we will cover:The basics of using HTML to create forms Various kinds of form input fields. Learn MoreThe best ways to organize formsUsing CSS to style forms for improved usability. Read CSS GuideImproving the user experience through accessibility and form validationYou will know how to make forms for your website that are both aesthetically pleasing and easy to use by the end of this tutorial.A organized area of a website where users may enter and submit data is called an HTML form. Typically, a form's element has buttons, labels, selection options, and many input fields.Data is transmitted to a server for processing when a user submits a form, typically via the HTTP GET or POST protocols.Learn more about HTML Forms: HTML Form TutorialA simple HTML form consists of:A element that acts as a container for all input fieldsInput fields (such as text boxes, radio buttons, and dropdowns)Labels that describe each input fieldA submit button to send the data to a server Full Name: Email: SubmitHTML offers various input elements to collect user data efficiently. Here are some of the most commonly used form fields:Collects single-line text input (e.g., name, username)Collects an email address and ensures proper formatMasks input characters for secure password entryAllows users to select one option from multiple choicesAllows users to select multiple optionsCollects multi-line text input (e.g., comments, messages)Creates a dropdown list for selecting a valueEach input field should be paired with a to enhance readability and accessibility.By default, HTML forms appear plain and unstyled. CSS allows us to enhance their appearance, improving readability and usability.Padding and Margins: Improve spacing between form elements for better readability.Borders and Backgrounds: Enhance input field visibility.Font Styles: Customize text appearance for labels and input fields.Hover and Focus Effects: Highlight fields when users interact with them.Button Styling: Make submit buttons more noticeable.Example CSS for Form Styling Contact Form body { background: #a1c4fd; font-family: Arial, sans-serif; text-align: center; } form { background: white; width: 300px; margin: auto; padding: 15px; border-radius: 10px; box-shadow: 0 4px 8px rgba(0, 0, 0, 0.1); } input, textarea, button { width: 100%; margin: 8px 0; padding: 10px; border-radius: 5px; border: 1px solid #ccc; } button { background: #2c3e50; color: white; border: none; cursor: pointer; } button:hover { background: #1a252f; } Contact Us Send See CSS Styling in Action: CSS Form Styling GuideMake use of appropriate spacing: Organize form components consistently and rationally.Draw attention to the targeted fields: To show the user's typing location, use the :focus pseudo-class.Assure responsiveness on mobile devices: Make use of adaptable layouts to suit various screen sizes.Employ contrasting hues: Make it easier to read and make sure that everyone can access it.Make good use of placeholders: Labels shouldn't be replaced with placeholder text; it should just offer suggestions.A well-styled form not only looks professional but also enhances the overall user experience.Form validation ensures that users enter correct and complete information before submitting a form.HTML provides built-in validation attributes such as required, minlength, maxlength, and pattern to enforce input rules.Learn more: HTML Form ValidationServer-side validation is crucial for security, even when client-side validation enhances user experience. All submitted data should be validated by the backend prior to processing.When consumers enter wrong data, errors are clearly indicated.To draw attention to mistakes and successful messages, use colors and icons.Usability must be carefully considered while creating an effective form. The following are recommended procedures to adhere to:Don't complicate: Steer clear of fields that are superfluous and add to the form's length.Use clear labels: To help the user, make sure that each input field has a relevant label.Put similar fields together: Form elements can be logically arranged using fieldsets or sections.Give out error messages: Provide users with informative error messages when they enter inaccurate or missing data.Make sure it's accessible: Use ARIA properties and semantic HTML to create forms that are usable by people with disabilities.Make your site mobile-friendly: Make sure buttons and input fields are the right size for touch interactions.Forms that are properly designed increase completion rates and lessen user annoyance, which results in improved user experience.Because they facilitate user interaction and data entry, forms are an essential component of web development. You can make forms that are both aesthetically pleasing and easy to use by properly structuring them using HTML and then improving their design with CSS. A smooth user experience on all devices is also guaranteed by putting accessibility and validation into practice.Whether you're creating a straightforward contact form or a sophisticated registration system, knowing how to create forms that work can enhance both functionality and usability.Want to become an expert in web design, HTML, and CSS? Enroll in our Web Development Course now to discover how to easily create beautiful, interactive websites.To learn more about the course and begin your path to become a professional web developer, click here. Creating a survey form is a great way to understand the basics of web development. In this tutorial, we will build a Survey Form that allows users to easily submit their responses. The form will include different input types such as text fields, checkboxes, and radio buttons, all designed using HTML for the structure and CSS for styling. What We're Going to Create:Survey Form Interface: We will design a simple form where users can easily input their responses using text fields, checkboxes, and radio buttons.User-Friendly Design: The form will be visually appealing and easy to navigate, ensuring a smooth user experience.Data Collection: The form will gather user responses in an organized way, ready for submission.Responsive Layout: The form will be styled to adapt to different screen sizes, ensuring it looks great on both desktop and mobile devices.Project PreviewBuild a Survey Form using HTML and CSSSurvey Form - HTML and CSS CodeThis survey form has a simple and flexible design that works well on any device. It includes different input options like text fields, checkboxes, and radio buttons, making it easy to fill out. The form also looks nice and has smooth effects, making it easy to use. index.html GeeksforGeeks Survey Form Name Email Age Which option best describes you? Student Intern Professional Other Would you recommend GeeksforGeeks to a friend? Yes No Maybe Languages and Frameworks known (Check all that apply) C C++ C# Java Python JavaScript React Angular Django Spring Any comments or suggestions Submit style.css /* Styling the Body element */ body { background-color: #05c46b; font-family: Verdana; text-align: center; } /* Styling the Form */ form { background-color: #fff; max-width: 500px; margin: 50px auto; padding: 30px 20px; box-shadow: 2px 5px 10px rgba(0, 0, 0, 0.5); } /* Styling .form-control */ .form-control { text-align: left; margin-bottom: 25px; } /* Styling .form-control label */ .form-control label { display: block; margin-bottom: 10px; } /* Styling .form-control input, select, textarea */ .form-control input, .form-control select, .form-control textarea { border: 1px solid #777; border-radius: 2px; font-family: inherit; padding: 10px; display: block; width: 95%; } /* Styling Radio button and Checkbox */ .form-control input[type="radio"], .form-control input[type="checkbox"] { display: inline-block; width: auto; } /* Styling Button */ button { background-color: #05c46b; border: 1px solid #777; border-radius: 2px; font-family: inherit; font-size: 21px; display: block; width: 100%; margin-top: 50px; margin-bottom: 20px; } Output:Build a Survey Form using HTML and CSSIn this code:The form collects user details like name, email, age, role, feedback, and more.Uses text, email, and radio button inputs for user data.Each input field is labeled for clarity.Allows users to select one or multiple options for feedback.A button to submit the survey.Body Styling: Sets a green background with centered text and a simple font.Form Styling: The form has a white background, padding, and a shadow for a clean, modern look.Form-Control: Inputs, selects, and text areas are styled for consistency and usability.Button Styling: The submit button has a green background, with padding and full width for easy clicking.Responsive Layout: The form is centered and adapts to different screen sizes for better user experience.ConclusionIn this article, we built a simple and attractive survey form using HTML and CSS. The form collects basic information like name, email, and feedback using different types of input fields such as text boxes, radio buttons, and checkboxes. We also added styling to make the form look clean and easy to use. In this tutorial, you will learn how to create a registration form using HTML and CSS. Whether you are a complete beginner or already have some basic knowledge of HTML and CSS, this guide will help you build a functional and stylish registration form. When it comes to designing websites, one of the most essential elements is a registration form. It's the first step users take to sign up for a service or application. A registration form ensures that your users have an excellent experience on any device, whether they're on a phone, tablet, or desktop. In this guide, we'll walk you through the steps of creating a registration form using only HTML and CSS.This tutorial will provide all the necessary details to create a smooth and functional registration form. If you're looking to create a registration form using HTML and CSS, this guide is exactly what you need. Even if you're a complete beginner with just a basic understanding of HTML and CSS, you will be able to build this registration form by following the steps outlined here. A registration form is important for any website or app. It's where users fill in key details like their name, email, and password. This is the first step to creating an account or signing up for a service. How to structure the clock with HTML How to style the clock using CSS Start by creating a new file called (index.html). Then, copy and paste the HTML code provided into this file. Remember to save it with the (.html) extension. Registration Form

Registration Form

Full Name: Email: Password: Confirm Password: Register Create a new file called (style.css) and copy the provided code into this file. Don't forget to save it with the (.css) extension. /* Global styles */ * { margin: 0; padding: 0; box-sizing: border-box; } body { font-family: 'Arial', sans-serif; background: linear-gradient(135deg, #6e7dff, #8e4ae6); display: flex; justify-content: center; align-items: center; height: 100vh; margin: 0; } .form-container { background-color: #fff; padding: 30px; border-radius: 12px; box-shadow: 0 8px 16px rgba(0, 0, 0, 0.15); width: 100%; max-width: 400px; text-align: center; } h2 { font-size: 24px; margin-bottom: 20px; color: #333; } form { display: flex; flex-direction: column; } label { font-size: 14px; color: #555; margin-bottom: 8px; text-align: left; } input { padding: 12px; margin-bottom: 15px; border: 1px solid #ddd; border-radius: 8px; font-size: 14px; outline: none; transition: border 0.3s; } input:focus { border-color: #6e7dff; } button { padding: 12px; background-color: #6e7dff; color: white; border: none; border-radius: 8px; font-size: 16px; cursor: pointer; transition: background-color 0.3s; } button:hover { background-color: #5a66cc; } input::placeholder { color: #aaa; } In this tutorial, we've shown you how to create a simple, yet effective registration form using HTML and CSS. By following the steps, you learned how to structure the form with HTML and style it with CSS to make it both functional and visually appealing. Creating forms is a fundamental skill for web development, and with the knowledge you've gained here, you can build various types of forms for different purposes, such as login forms, contact forms, or survey forms. The techniques used in this tutorial can be extended and customized to meet your specific design requirements, helping you enhance the user experience on your website. Whether you're a beginner or looking to brush up on your web development skills, building forms is a great way to practice HTML and CSS. Keep experimenting with different layouts, colour, and styles to make your forms more interactive and user-friendly. Happy Coding! The look of an HTML form can be greatly improved with CSS: Use the width property to determine the width of the input field: First Name Try it Yourself » The example above applies to all elements. If you only want to style a specific input type, you can use attribute selectors: input[type=text] - will only select text fields input[type=password] - will only select password fields input[type=number] - will only select number fields etc.. Use the padding property to add space inside the text field. Tip: When you have many inputs after each other, you might also want to add some margin, to add more space outside of them: First Name Last Name input[type=text] { width: 100%; padding: 12px 20px; margin: 8px 0; box-sizing: border-box; } Try it Yourself » Note that we have set the box-sizing property to border-box. This makes sure that the padding and eventually borders are included in the total width and height of the elements. Read more about the box-sizing property in our CSS Box Sizing chapter. Bordered Inputs Use the border property to change the border size and color, and use the border-radius property to add rounded corners: First Name If you only want a bottom border, use the border-bottom property: First Name Colored Inputs Use the background-color property to add a background color to the input, and the color property to change the text color: Focused Inputs By default, some browsers will add a blue outline around the input when it gets focus (clicked on). You can remove this behavior by adding outline: none; to the input. Use the :focus selector to do something with the input field when it gets focus: Input with icon/image If you want an icon inside the input, use the background-image property and position it with the background-position property. Also notice that we add a large left padding to reserve the space of the icon: input[type=text] { background-color: white; background-image: url('searchicon.png'); background-position: 10px 10px; background-repeat: no-repeat; padding-left: 40px; } Try it Yourself » Animated Search Input In this example we use the CSS transition property to animate the width of the search input when it gets focus. You will learn more about the transition property later, in our CSS Transitions chapter. input[type=text] { transition: width 0.4s ease-in-out;}input[type=text]:focus { width: 100%;} Try it Yourself » Styling Textareas Tip: Use the resize property to prevent textareas from being resized (disable the "grabber" in the bottom right corner): Some text... textarea { width: 100%; height: 150px; padding: 12px 20px; box-sizing: border-box; border: 2px solid #ccc; border-radius: 4px; background-color: #f8f8f8; resize: none; } Try it Yourself » Styling Select Menus select { width: 100%; padding: 16px 20px; border: none; border-radius: 4px; background-color: #f1f1f1; } Try it Yourself » Styling Input Buttons input[type=button], input[type=submit], input[type=reset] { background-color: #04AA6D; border: none; color: white; padding: 16px 32px; text-decoration: none; margin: 4px 2px; cursor: pointer; }/* Tip: use width: 100% for full-width buttons */ Try it Yourself » For more information about how to style buttons with CSS, read our CSS Buttons Tutorial. Responsive Form Resize the browser window to see the effect. When the screen is less than 600px wide, make the two columns stack on top of each other instead of next to each other. Advanced: The following example uses media queries to create a responsive form. You will learn more about this in a later chapter. Try it Yourself » Nearly all browsers nowadays support CSS and many other applications do, too. To write CSS, you don't need more than a text editor, but there are many tools available that make it even easier. Of course, all software has bugs, even after several updates. And some programs are further ahead implementing the latest CSS modules than others. Various sites describe bugs and work-arounds. More » For beginners, Starting with HTML + CSS teaches how to create a style sheet. For a quick introduction to CSS, try chapter 2 of Lie & Bos or Dave Raggett's intro to CSS. Or see examples of styling XML and CSS tips & tricks. Another page also has some books, mailing lists and similar fora, and links to other directories. The history of CSS is described in chapter 20 of the book Cascading Style Sheets, designing for the Web, by Håkon Wium Lie and Bert Bos (2nd ed., 1999, Addison Wesley, ISBN 0-201-59625-3) More » CSS inspired 'quayjn' to write the song 'CSS is OK'. Site navigation When the first CSS specification was published, all of CSS was contained in one document that defined CSS Level 1. CSS Level 2 was defined also by a single, multi-chapter document. However for CSS beyond Level 2, the CSS Working Group chose to adopt a modular approach, where each module defines a part of CSS, rather than to define a single monolithic specification. This breaks the specification into more manageable chunks and allows more immediate, incremental improvement to CSS. Since different CSS modules are at different levels of stability, the CSS Working Group has chosen to publish this profile to define the current scope and state of Cascading Style Sheets as of 2024. Cascading Style Sheets (CSS) CSS is a language for writing style sheets, and is designed to describe the rendering of structured documents (such as HTML and XML) on a variety of media. CSS is used to describe the presentation of a source document, and usually does not change the underlying semantics expressed by its document language. Style sheet A set of rules that specify the presentation of a document. Style sheets are written by an Author, and interpreted by a User Agent, to present the document to the User. Source document The document to which one or more style sheets apply. A source document's structure and semantics are encoded using a document language (e.g., HTML, XHTML, or SVG). Author An author is a person who writes documents and associated style sheets. An authoring tool is a User Agent that generates style sheets. User A user is a person who interacts with a user agent to view, hear, or otherwise use the document. User Agent (UA) A user agent is any program that interprets a document and its associated style sheets on behalf of a user. A user agent may display a document, read it aloud, cause it to be printed, convert it to another format, etc. For the purposes of the CSS specifications, a User Agent is one that supports and interprets Cascading Style Sheets as defined in these specifications. This section is non-normative. In the W3C Process, a Recommendation-track document passes through three levels of stability, summarized below: Working Draft (WD) This is the design phase of a W3C spec. The WG iterates the spec in response to internal and external feedback. The first official Working Draft is designated the "First Public Working Draft" (FPWD). In the CSSWG, publishing FPWD indicates that the Working Group as a whole has agreed to work on the module, roughly as scoped out and proposed in the editor's draft. The transition to the next stage is sometimes called "Last Call Working Draft" (LCWD) phase. The CSSWG transitions Working Drafts once we have resolved all known issues, and can make no further progress without feedback from building tests and implementations. This "Last Call for Comments" sets a deadline for reporting any outstanding issues, and requires the WG to specially track and address incoming feedback. The comment-tracking document is the Disposition of Comments (DoC). It is submitted along with an updated draft for the Director's approval, to demonstrate wide review and acceptance. Candidate Recommendation (CR) This is the testing phase of a W3C spec. Notably, this phase is about using tests and implementations to test the specification: it is not about testing the implementations. This process often reveals more problems with the spec, and so a Candidate Recommendation will morph over time in response to implementation and testing feedback, though usually less so than during the design phase (WD). Demonstration of two correct, independent implementations of each feature is required to exit CR, so in this phase the WG builds a test suite and generates implementation reports. The transition to the next stage is "Proposed Recommendation" (PR). During this phase the W3C Advisory Committee must approve the transition to REC. Recommendation (REC) This is the completed state of a W3C spec and represents a maintenance phase. At this point the WG only maintains an errata document and occasionally publishes an updated edition that incorporates the errata back into the spec. An Editor's Draft is effectively a live copy of the editors' own working copy. It may or may not reflect Working Group consensus, and can at times be in a self-inconsistent state. (Because the publishing process at W3C is time-consuming and onerous, the Editor's Draft is usually the best (most up-to-date) reference for a spec. Efforts are currently underway to reduce the friction of publishing, so that official drafts will be regularly up-to-date and Editor's Drafts can return to their original function as scratch space.) A list of all CSS modules, stable and in-progress, and their statuses can be found at the CSS Current Work page. This profile includes only specifications that we consider stable and for which we have enough implementation experience that we are sure of that stability. Note: This is not intended to be a CSS Desktop Browser Profile: inclusion in this profile is based on feature stability only and not on expected use or Web browser adoption. This profile defines CSS in its most complete form. As of 2024, Cascading Style Sheets (CSS) is defined by the following specifications. CSS Level 2, latest revision (including errata) [CSS2] This defines the core of CSS, parts of which are overridden by later specifications. We recommend in particular reading Chapter 2, which introduces some of the basic concepts of CSS and its design principles. CSS Syntax Level 3 [CSS-SYNTAX-3] Replaces CSS2§4.1, CSS2§4.2, CSS2§4.4, and CSS2§G, redefining how CSS is parsed. CSS Style Attributes [CSS-STYLE-ATTR] Defines how CSS declarations can be embedded in markup attributes. Media Queries Level 3 [CSS3-MEDIAQUERIES] Replaces CSS2§7.3 and expands on the syntax for media-specific styles. CSS Conditional Rules Level 3 [CSS-CONDITIONAL-3] Extends and supersedes CSS2§7.2, updating the definition of @media rules to allow nesting and introducing the @supports rule for feature-support queries. Selectors Level 3 [SELECTORS-3] Replaces CSS2§5 and CSS2§6.4.3, defining an extended range of selectors. CSS Namespaces [CSS3-NAMESPACE] Introduces an @namespace rule to allow namespace-prefixed selectors. CSS Cascading and Inheritance Level 4 [CSS-CASCADE-4] Extends and supersedes CSS2§1.4.3 and CSS2§6, as well as [CSS-CASCADE-3]. Describes how to collate style rules and assign values to all properties on all elements. By way of cascading and inheritance, values are propagated for all properties on all elements. CSS Values and Units Level 3 [CSS-VALUES-3] Extends and supersedes CSS2§1.4.2.1, CSS2§4.3, and CSS2§A.2.1–3, defining CSS's property definition syntax and expanding its set of units. CSS Custom Properties for Cascading Variables Module Level 1 [CSS-VARIABLES-1] Introduces cascading variables as a new primitive value type that is accepted by all CSS properties, and custom properties for defining them. CSS Box Model Level 3 [CSS-BOX-3] Replaces CSS2§8.1, §8.2, §8.3 (but not §8.3.1), and §8.4. CSS Color Level 4 [CSS-COLOR-4] Extends and supersedes CSS2§4.3.6, CSS2§14.1, and CSS2§18.2, also extends and supersedes [CSS-COLOR-3], introducing an extended range of color spaces beyond sRGB, extended color values, and CSS Object Model extensions for color. Also defines the opacity property. CSS Backgrounds and Borders Level 3 [CSS-BACKGROUNDS-3] Extends and supersedes CSS2§8.5 and CSS2§14.2, providing more control of backgrounds and borders, including layered background images, image borders, and drop shadows. CSS Images Level 3 [CSS-IMAGES-3] Redefines and incorporates the external 2D image value type, introduces native 2D gradients, and adds additional controls for replaced element sizing and rendering. CSS Fonts Level 3 [CSS-FONTS-3] Extends and supersedes CSS2§15 and provides more control over font choice and feature selection. CSS Writing Modes Level 3 [CSS-WRITING-MODES-3] Defines CSS support for various international writing modes, such as left-to-right (e.g. Latin or Indic), right-to-left (e.g. Hebrew or Arabic), bidirectional (e.g. mixed Latin and Arabic) and vertical (e.g. Asian scripts). Replaces and extends CSS2§8.6 and §9.10. CSS Multi-column Layout Level 1 [CSS-MULTICOL-1] Introduces multi-column flows to CSS layout. CSS Flexible Box Module Level 1 [CSS-FLEXBOX-1] Introduces a flexible linear layout model for CSS. CSS Basic User Interface Module Level 3 [CSS-UI-3] Extends and supersedes CSS2§18.1 and CSS2§18.4, defining cursor, outline, and several new CSS features that also enhance the user interface. CSS Containment Module Level 1 [CSS-CONTAIN-1] Introduces the contain property, which enforces the independent CSS processing of an element's subtree in order to enable heavy optimizations by user agents when used well. CSS Transforms Level 1 [CSS-TRANSFORMS-1] Introduces coordinate-based graphical transformations to CSS. CSS Compositing and Blending Level 1 [COMPOSITING] Defines the compositing and blending of overlaid content and introduces features to control their modes. CSS Easing Functions Level 1 [CSS-EASING-1]. Describes a way for authors to define a transformation that controls the rate of change of some value. Applied to animations, such transformations can be used to produce animations that mimic physical phenomena such as momentum or to cause the animation to move in discrete steps producing robot-like movement. CSS Counter Styles Level 3 [CSS-COUNTER-STYLES-3] Introduces the @counter-style rule, which allows authors to define their own custom counter styles for use with CSS list-marker and generated-content counters [CSS-LISTS-3]. It also predefines a set of common counter styles, including the ones present in CSS2 and CSS2.1. Note: Although we don't anticipate significant changes to the specifications that form this snapshot, their inclusion does not mean they are frozen. The Working Group will continue to address problems as they are found in these specs. Implementers should monitor www-style and/or the CSS Working Group Blog for any resulting changes, corrections, or clarifications. The following specifications are considered to be in a reliable state, meaning they have largely stable implementations and specifications, but are not yet at the Recommendation level due to minor issues or the need for additional implementation reports. Media Queries Level 4 [MEDIAQUERIES-4] Extends and supersedes [CSS3-MEDIAQUERIES], expanding the syntax, deprecating most media types, and introducing new media features. CSS Scroll Snap Module Level 1 [CSS-SCROLL-SNAP-1] Contains features to control panning and scrolling behavior with "snap positions". CSS Scrollbars Styling Module Level 1 [CSS-SCROLLBARS-1] Defines properties to influence the visual styling of scrollbars, introducing controls for their color and width. CSS Grid Layout Module Level 1 [CSS-GRID-1] Introduces a two-dimensional grid-based layout system, optimized for user interface design. In the grid layout model, the children of a grid container can be positioned into arbitrary slots in a predefined flexible or fixed-size layout grid. CSS Grid Layout Module Level 2 [CSS-GRID-2] Extends and supersedes [CSS-GRID-1], introducing "subgrids" for managing nested markup in a shared grid framework. The following modules have completed design work, and are fairly stable, but have not received much testing and implementation experience yet. We hope to incorporate them into the official definition of CSS in a future snapshot. Media Queries Level 4 [MEDIAQUERIES-4] Extends and supersedes [CSS3-MEDIAQUERIES], expanding the syntax, deprecating most media types, and introducing new media features. CSS Display Module Level 3 [CSS-DISPLAY-3] Replaces CSS2§9.1.2, §9.2.1 (but not §9.2.1.1), §9.2.2 (but not §9.2.2.1), §9.2.3, and §9.2.4 (and lays the foundations for replacing §9.7), defining how the CSS formatting box tree is generated from the document element tree and defining the display property that controls it. CSS Writing Modes Level 4 [CSS-WRITING-MODES-4] Extends and supersedes [CSS-WRITING-MODES-3], adding more options for vertical writing. CSS Fragmentation Module Level 3 [CSS-BREAK-3] Describes the fragmentation model that partitions a flow into pages, columns, or regions and defines properties that control it. Extends and supersedes CSS2§13.3. CSS Box Alignment Module Level 3 [CSS-ALIGN-3] Introduces properties to control the alignment of boxes within their containers in the various CSS box layout models: block layout, table layout, flex layout, and grid layout. CSS Shapes Module Level 1 [CSS-SHAPES-1] Extends floats (CSS2§9.5) to effect non-rectangular wrapping shapes. CSS Text Module Level 3 [CSS-TEXT-3] Extends and supersedes CSS2§16 excepting §16.3, defining properties for text manipulation and specifying their processing model. It covers line breaking, justification and alignment, white space handling, and text transformation. CSS Text Decoration Module Level 3 [CSS-TEXT-DECOR-3] Extends and supersedes CSS2§16.3, providing more control over text decoration lines and adding the ability to specify text emphasis marks and text shadows. CSS Masking Module Level 1 [CSS-MASKING-1] Replaces CSS2§11.1.2 and introduces more powerful ways of clipping and masking content. CSS Scroll Snap Module Level 1 [CSS-SCROLL-SNAP-1] Contains features to control panning and scrolling behavior with "snap positions". CSS Speech Module Level 1 [CSS-SPEECH-1] Replaces CSS2§A, overhauling the (non-normative) speech rendering chapter. CSS View Transitions Module Level 1 [CSS-VIEW-TRANSITIONS-1] Defines the View Transition API, along with associated properties and pseudo-elements, which allows developers to create animated visual transitions representing changes in the document state. Although the following modules have been widely deployed with rough interoperability, their details are not fully worked out or sufficiently well-specified and they need more testing and bugfixing. We hope to incorporate them into the official definition of CSS in a future snapshot. CSS Transitions Level 1 [CSS-TRANSITIONS-1] and CSS Animations Level 1 [CSS-ANIMATIONS-1]. Introduces mechanisms for transitioning the computed values of CSS properties over time. CSS Will Change Level 1 [CSS-WILL-CHANGE-1] Introduces a performance hint property called will-change. Filter Effects Module Level 1 [FILTER-EFFECTS-1] Introduces filter effects as a way of processing an element's rendering before it is displayed in the document. CSS Font Loading Module Level 3 [CSS-FONT-LOADING-3] Introduces events and interfaces used for dynamically loading font resources. CSS Box Sizing Level 3 [CSS-SIZING-3] Overlays and extends CSS§10., expanding the value set of the sizing properties, introducing more precise sizing terminology, and defining with more precision and detail various automatic sizing concepts only vaguely defined in CSS2. CSS Transforms Level 2 [CSS-TRANSFORMS-2] Builds upon [CSS-TRANSFORMS-1] to add new transform functions and properties for three-dimensional transforms, and convenience functions for simple transforms. CSS Lists and Counters Module Level 3 [CSS-LISTS-3] Contains CSS features related to list counters: styling them, positioning them, and manipulating their value. CSS Logical Properties and Values Level 1 [CSS-LOGICAL-1] Introduces logical properties and values that provide the author with the ability to control layout through logical, rather than physical, direction and dimension mappings. Also defines logical properties and values for the features defined in [CSS2]. These properties are writing-mode relative equivalents of their corresponding physical properties. CSS Positioned Layout Module Level 3 [CSS-POSITION-3] Contains defines coordinate-based positioning and offsetting schemes of CSS: relative positioning, sticky positioning, absolute positioning, and fixed positioning. Resize Observer [RESIZE-OBSERVER-1] This specification describes an API for observing changes to element's principal box's size. Web Animations [WEB-ANIMATIONS-1] Defines a model for synchronization and timing of changes to the presentation of a Web page. Also defines an application programming interface for interacting with this model. CSS Fonts Module Level 4 [CSS-FONTS-4] Extends and supersedes CSS Fonts 3 and provides more control over font choice and feature selection, including support for OpenType variations. CSS Color Adjustment Module Level 1 [CSS-COLOR-ADJUST-1] This module introduces a model and controls over automatic color adjustment by the user agent to handle user preferences and device output optimizations. CSS Conditional Rules Module Level 4 [CSS-CONDITIONAL-4] Extends CSS Conditional 3 to allow testing for supported selectors. CSS Cascading and Inheritance Level 5 [CSS-CASCADE-5] Extends CSS Cascade 4 to add cascade layers. Motion Path Module Level 1 [MOTION-1] This module allows authors to position any graphical object and animate it along an author specified path. CSS Scroll Anchoring Module Level 1 [CSS-SCROLL-ANCHORING-1] This module aims to minimize content shifts by locking the scroll position of a scroll container to a particular anchor element. CSS Object Model (CSSOM) [CSSOM-1] This module defines APIs for parsing, serializing, and manipulating CSS, Media Queries, and Selectors. CSS Color Module Level 5 [CSS-COLOR-5] Extends CSS Color 4 to add color spaces and color modification functions. Selectors Level 4 [SELECTORS-4] Extends Selectors Level 3 by introducing new pseudo-classes, pseudo-elements, and combinators, enhancing the ability to select elements based on more complex criteria and states. Cascading Style Sheets does not have versions in the traditional sense; instead it has levels. Each level of CSS builds on the previous, refining definitions and adding features. The feature set of each higher level is a superset of any lower level, and the behavior allowed for a given feature in a higher level is a subset of that allowed in the lower levels. A user agent conforming to a higher level of CSS is thus also conformant to all lower levels. CSS Level 1 The CSS Working Group considers the CSS1 specification to be obsolete. CSS Level 1 is defined as all the features defined in the CSS1 specification (properties, values, at-rules, etc), but using the syntax and definitions in the CSS2.1 specification. CSS Style Attributes defines its inclusion in element-specific style attributes. CSS Level 2 Although the CSS2 specification is technically a W3C Recommendation, it passed into the Recommendation stage before the W3C had defined the Candidate Recommendation stage. Over time implementation experience and further review has brought to light many problems in the CSS2 specification, so instead of expanding an already unwieldy errata list, the CSS Working Group chose to define CSS Level 2 Revision 1 (CSS2.1). In case of any conflict between the two specs CSS2.1 contains the definitive definition. Once CSS2.1 became Candidate Recommendation—effectively though not officially the same level of stability as CSS2—obsoleted the CSS2 Recommendation. Features in CSS2 that were dropped from CSS2.1 should be considered to be at the Candidate Recommendation stage, but note that many of these have been or will be pulled into a CSS Level 3 working draft, in which case that specification will, once it reaches CR, obsolete the definitions in CSS2. The CSS2.1 specification defines CSS Level 2 and the CSS Style Attributes specification defines its inclusion in element-specific style attributes. CSS Level 3 CSS Level 3 builds on CSS Level 2 module by module, using the CSS2.1 specification as its core. Each module adds functionality and/or replaces part of the CSS2.1 specification. The CSS Working Group intends that the new CSS modules will not contradict the CSS2.1 specification: only that they will add functionality and refine definitions. As each module is completed, it will be plugged in to the existing system of CSS2.1 plus previously-completed modules. From this level on modules are levelled independently: for example Selectors Level 4 may well be completed before CSS Line Module Level 3. Modules with no CSS Level 2 equivalent start at Level 1; modules that update features that existed in CSS Level 2 start at Level 3. CSS Level 4 and beyond There is no CSS Level 4. Independent modules can reach level 4 or beyond, but CSS the language no longer has levels. ("CSS Level 3" as a term is used only to differentiate it from the previous monolithic versions.) Not all implementations will implement all functionality defined in CSS. In the past, the Working Group published a few Profiles, which were meant to define the minimal subset of CSS that various classes of user agents were expected to support. This effort has been discontinued, as the Working Group was not finding it effective or useful, and the profiles previously defined are now unmaintained. Note: Partial implementations of CSS, even if that subset is an official profile, must follow the forward-compatible parsing rules for partial implementations. The following sections define several conformance requirements for implementing CSS responsibly, in a way that promotes interoperability in the present and future. So that authors can exploit the forward-compatible parsing rules to assign fallback values, CSS renderers must treat as invalid (and ignore as appropriate) any at-rules, properties, property values, keywords, and other syntactic constructs for which they have no usable level of support. In particular, user agents must not selectively ignore unsupported property values and honor supported values in a single multi-value property declaration: if any value is considered invalid (as unsupported values must be), CSS requires that the entire declaration be ignored. To avoid clashes with future stable CSS features, the CSSWG recommends the following best practices for the implementation of unstable features and proprietary extensions to CSS: Implementations of unstable features that are described in W3C specifications but are not interoperable should not be released broadly for general use; but may be released for limited, experimental use in controlled environments. Why? We want to allow both authors and implementors to experiment with the feature and give feedback, but prevent authors from relying on them in production websites and thereby accidentally "locking in" (through content dependence) certain syntax or behavior that might change later. For example, a UA could release an unstable features for experimentation through beta or other testing-stage builds; behind a hidden configuration flag; behind a switch enabled only for specific testing partners; or through some other means of limiting dependent use. A CSS feature is considered unstable until its specification has reached the Candidate Recommendation (CR) stage in the W3C process. In exceptional cases, the CSSWG may additionally, by an officially-recorded resolution, add pre-CR features to the set that are considered safe to release for broad use. See §4 Safe to Release pre-CR Exceptions. Note: Vendors should consult the WG explicitly and not make assumptions on this point, as a pre-CR spec that hasn't changed in awhile is usually more out-of-date than stable. To avoid clashes with future CSS features, the CSS2.1 specification reserves a prefixed syntax [CSS2] for proprietary and experimental extensions to CSS. A CSS feature is a proprietary extension if it is meant for use in a closed environment accessible only to a single vendor's user agent(s). A UA should support such proprietary extensions only through a vendor-prefixed syntax and not expose them to open (multi-UA) environments such as the World Wide Web. Why? The prefixing requirement allows shipping specialized features in closed environments without conflicting with future additions to standard CSS. The restriction on exposure to open systems is to prevent accidentally causing the public CSS environment to depend on an unstandardized proprietary extensions. For example, Firefox's XUL-based UI, Apple's iTunes UI, and Microsoft's Universal Windows Platform app use extensions to CSS implemented by their respective UAs. So long as these UAs do not allow Web content to access these features, they do not provide an opportunity for such content to become dependent on their proprietary extensions. Even if a feature is intended to eventually be used in the Web, if it hasn't yet been standardized it should still not be exposed to the Web. If a feature is unstable (i.e. the spec has not yet stabilized), but at least three UAs implement the feature (or a UA has broken the other rules and shipped for broad use an unstable or otherwise non-standard feature in a production release), and the implementations have rough interoperability, and the CSS Working Group has recorded consensus that this feature should exist and be released, implementers may ship that feature unprefixed in broad-release builds. Rough interoperability is satisfied by a subjective judgment that even though there may be differences, the implementations are sufficiently similar to be used in production websites for a substantial number of use cases. Note that the CSSWG must still be consulted to ensure coordination across vendors and to ensure coherency review by the CSS experts from each vendor. Note also that rough interoperability still usually means painful lack of interop in edge (or not-so-edge) cases, particularly because details have not been ironed out through the standards review process. Why? If a feature is sufficiently popular that three or more browsers have implemented it before it's finished standardization, this clause allows releasing the pressure to ship. Also, if a feature has already escaped into the wild and sites have started depending on it, pretending it's still "experimental" doesn't help anyone. Allowing others to ship unprefixed recognizes that the feature is now de facto standardized and encourages authors to write cross-platform code. When exposing such a standards-track unstable feature to the Web in a production release, implementations should support both vendor-prefixed and unprefixed syntaxes for the feature. Once the feature has stabilized and the implementation is updated to match interoperable behavior, support for the vendor-prefixed syntax should be removed. Why? This is recommended so that authors can use the unprefixed syntax to target all implementations, but when necessary, can target specific implementations to work around incompatibilities among implementations as they get ironed out through the standards/bugfixing process. The lack of a phase where only the prefixed syntax is supported greatly reduces the risk of stylesheets being written with only the vendor-prefixed syntax. This in turn allows UA vendors to retire their prefixed syntax once the feature is stable, with a lower risk of breaking existing content. It also reduces the need occasionally felt by some vendors to support a feature with the prefix of another vendor, due to content depending on that syntax. Anyone promoting unstable features to authors should document them using their standard unprefixed syntax, and avoid encouraging the use of the vendor-prefixed syntax for any purpose other than working around implementation differences. In order to preserve the open nature of CSS as a technology, vendors should make it possible for other implementors to freely implement any features that they do ship. To this end, they should provide spec-editing and

testing resources to complete standardization of such features, and avoid other obstacles (e.g., platform dependency, licensing restrictions) to their competitors shipping the feature. Once a specification reaches the Candidate Recommendation stage, implementers should release an unprefixed implementation of any CR-level feature they can demonstrate to be correctly implemented according to spec, and should avoid exposing a prefixed variant of that feature. To establish and maintain the interoperability of CSS across implementations, the CSS Working Group requests that non-experimental CSS renderers submit an implementation report (and, if necessary, the testcases used for that implementation report) to the W3C before releasing an unprefixed implementation of any CSS features. Testcases submitted to W3C are subject to review and correction by the CSS Working Group. Further information on submitting testcases and implementation reports can be found from on the CSS Working Group’s website at . Questions should be directed to the public-css-testsuite@w3.org mailing list. The following features have been explicitly and proactively cleared by the CSS Working Group for broad release prior to the spec reaching Candidate Recommendation. See § 3.2.1 Experimentation and Unstable Features. The following features have been explicitly and retroactively cleared by the CSS Working Group for broad release prior to the spec reaching Candidate Recommendation: Everything in CSS Animations Level 1 and CSS Transitions Level 1. These sections are non-normative. = ~= 1st 2d matrix 2nd 3rd 4th absolute length absolute length unit absolutely positioned element abstract dimensions :active activeborder activecaption active duration active (pseudo-class) actual value in css-cascade-4 in css21 additive tuple adjoining margins advance measure :after after after-change style aliceblue alignment baseline alignment container alignment context alignment subject 'all' media group alphabetic baseline ambiguous image url an+b ancestor anchor anchor unit animation origin animation-tainted anonymous in css-display-3, for CSS in css21 anonymous box anonymous inline boxes antiquewhite apply to appworkspace aqua aquamarine are a valid escape aspect value atomic inline atomic inline box atomic inline-level box at-rule attr() attribute 'audio' media group auditory icon augmented grid aural box model author authoring tool author origin author-origin author presentational hint origin author style sheet automatic column position automatic grid position automatic numbering automatic placement automatic position automatic row position auto-placement auto-placement cursor available font faces available grid space avoid break values axis-lock axis value azure backdrop background background color background image background image layer background painting area background positioning area backslash escapes baseline baseline alignment baseline alignment preference baseline content-alignment baseline self-alignment baseline set baseline-sharing group baseline table base size bearing angle :before before before-change style before flag beige bfc bidi formatting characters bidi-isolate bidi-isolated bidi isolation bidi paragraph bidirectionality bidirectionality (bidi) bi-orientational bi-orientational transform bisque 'bitmap' media group black blanchedalmond ()-block []-block block in css-display-3 in css21 {}-block block at-rule block axis block-axis block box in css-display-3 in css21 block container block container box in css-display-3 in css21 block dimension block end block-end block flow direction block formatting context block formatting context root blockification blockify block layout block-level block-level box in css-display-3 in css21 block-level content block-level element block scripts block size block-size block start block-start blue blueviolet blur radius boolean context border box border color border edge border image border image area border image region border::of a box border radius border style border width bottom box box alignment properties box::border box::content box::content height box::content width box-corner box fragment box::margin box::overflow box::padding box tree break brown burlywood buttonface buttonhighlight buttonshadow buttontext cadetblue cancel canonical unit canvas canvas background canvas surface captiontext captures snap positions cascade in css-cascade-4 in css21 cascade-dependent keyword cascaded independently cascaded value cascade origin central baseline character character encoding character map "@charset" chartreuse check if three code points would start an ident sequence check if three code points would start a number check if three code points would start a unicode-range check if two code points are a valid escape child child combinator child selector chinese chocolate circled-lower-latin clamp a grid area clearance. clipping path clipping region in css-masking-1 in css21 closest-side clustered scripts collapse collapsed collapsed flex item collapsed grid track collapsed gutter collapsed track collapse through collapsible white space collapsing margin color color stop color stop list color transition hint column box column break column gap column height column rule column width combinator combinators combined duration compatible baseline alignment preferences compatible units complete completed transition component value composite face computed computed track list computed value in css-cascade-4 in css21 concrete object size conditional group rule conditional import conditionally hang conformance consecutive constraint rectangle consume a block consume a block's contents consume a component value consume a declaration consume a function consume a list of component values consume a list of declarations consume a list of rules consume an at-rule consume an escaped code point consume an ident-like token consume an ident sequence consume a number consume a numeric token consume a qualified rule consume a simple block consume a string token consume a style block's contents consume a stylesheet's contents consume a token consume a unicode-range token consume a url token consume comments consume the next input token consume the remnants of a bad declaration consume the remnants of a bad url consume the value of a unicode-range descriptor contain constraint containing block in css-display-3 in css21 containing block chain containing block for all descendants containing block::initial containment content content-based minimum size in css-flexbox-1 in css-grid-1 content box content distribution content-distribution content-distribution properties content edge content language content::of a box content::rendered content size suggestion in css-flexbox-1 in css-grid-1 content writing system continuous media 'continuous' media group convert a string to a number coordinated self-alignment preference coral cornflowerblue cornsilk counter() counters counter style counter symbol cover constraint crimson cross axis cross-axis cross-axis baseline set cross dimension cross-end cross size cross-size cross size property cross-start css bracketed range notation css-connected css feature queries cssfontfacerule cssfontfeaturevaluesrule css ident css identifier css ident sequence css qualified name css value definition syntax css-wide keywords cubic bézier easing function currentcolor current input code point current input token current transformation matrix current value cursive cursive script custom property cyan darkblue darkcyan darkgoldenrod darkgray darkgreen darkgrey darkkhaki darkmagenta darkolivegreen darkorange darkorchid darkred darksalmon darkseagreen darkslateblue darkslategray darkslategrey darkturquoise darkviolet declaration in css-syntax-3, for CSS in css21 declaration block declared declared value decode bytes decorating box deeppink deepskyblue default default face default namespace default object size default sizing algorithm default style sheet definite definite column position definite column span definite grid position definite grid span definite position definite row position definite row span definite size definite span descendant descendant-selectors descriptor descriptor declarations destination determine the fallback encoding device pixel dice digit dimension dimgray dimgrey directional embedding directional override discard a mark discard a token discard whitespace display type distributed alignment distribute extra space document in css-speech-1 in css-style-attr document language document order document tree document white space document white space characters dodgerblue dominant baseline easing function effective character map element in css-display-3, for CSS in css21 element::following element::preceding element tree emoji presentation participating code points empty in css-syntax-3, for token stream in css21 em (unit) encapsulation contexts end ending point ending shape ending token endmost end time end value environment encoding eof code point escaping establish an independent formatting context establish an orthogonal flow established an independent formatting context establishes an independent formatting context establishing an independent formatting context exact matching expanded name explicit grid explicit grid column explicit grid properties explicit grid row explicit grid track explicitly-assigned line name ex (unit) fallback alignment false in the negative range fantasy farthest-side fetch a font fetch an @import fictional tag sequence filter code points filtered code points filter primitive filter primitive attributes filter primitive subregion filter primitive tree filter region find the matching font faces fire a font load event firebrick :first first available font first-baseline alignment first-baseline content-alignment first baselines first-baseline self-alignment first baseline set :first-child first-child first cross-axis baseline set first formatted line :first-letter first-letter :first-line first-line first main-axis baseline set first symbol value fixed sizing function flex base size flex basis flex container flex direction flex factor flex factor sum flex formatting context flex fraction flex grow factor flexible flexible length in css-flexbox-1 in css-grid-1 flexible sizing function flexible tracks flex item flex layout flex-level flex line flex shrink factor float area float rules floralwhite flow layout flow of an element flow-relative flow-relative direction :focus focus focus (pseudo-class) following element font block period font download timer font failure period font feature value declaration font-feature-value-type font-relative lengths font source font specific font-stretch font swap period footnote forced break forced break values forced line break forced paragraph break forestgreen formatting context in css-display-3 in css21 formatting structure forward-compatible parsing fragment fragmentainer fragmentation fragmentation break fragmentation container fragmentation context fragmentation direction fragmentation root fragmented flow free space fuchsia full-size full-size kana full-width fully inflexible function functional notation gainsboro generate a counter generate a counter representation generate baselines generated content ghostwhite go gold goldenrod gradient-average-color gradient box gradient center gradient function gradient line grapheme cluster gray graytext green greenyellow grey grid grid area grid cell grid column grid column line grid container grid formatting context grid item grid item placement algorithm grid layout grid layout algorithm grid-level grid line 'grid' media group grid-modified document order grid order grid placement grid-placement property grid position grid row grid row line grid sizing algorithm grid span grid track growth limit guaranteed-invalid value gutter half-width hang hanging glyph height hex digit highlight highlighttext honeydew horizontal axis horizontal block flow horizontal dimension horizontal offset horizontal-only horizontal script horizontal typographic mode horizontal writing mode hotpink :hover hover (pseudo-class) hyphenate hyphenation hyphenation opportunity hyphen-separated matching hypothetical cross size hypothetical fr size hypothetical main size ident ident code point identifier in css-values-3, for CSS in css21 identity transform function ident sequence ident-start code point ignore ignored illegal implicit grid implicit grid column implicit grid lines implicit grid properties implicit grid row implicit grid track implicitly-assigned line name implicitly-named area @import important import conditions inactiveborder inactivecaption inactivecaptiontext indefinite indefinite size independent formatting context index indianred indigo infinitely growable in flow in-flow in css-display-3 in css21 infobackground infotext inherit in css-cascade-4 in css-cascade-4, for CSS in css21 inheritance in css-cascade-4 in css-cascade-4, for CSS inherited property inherited value initial containing block in css-display-3 in css21 initial free space initial representation for the counter value initial value in css-cascade-4 in css21 inline in css-display-3 in css21 inline axis inline-axis inline base direction inline block inline-block inline block box inline box in css-display-3 in css21 inline dimension inline end inline-end inline formatting context inline-level inline-level box in css-display-3 in css21 inline-level content inline-level element inline size inline-size inline start inline-start inlinification inlinify inner box-shadow inner display type inner edge input progress value input stream installed font fallback integer intended direction intended direction and end position intended end position 'interactive media group internal ruby box internal ruby element internal table box in css-display-3 in css21 internal table element in css-display-3 in css21 interpreter in css-namespaces-3 in css-style-attr intrinsic dimensions intrinsic sizing function invalid invalid at computed-value time invalid image invalid rule error invisible isolated sequence isolation iteration order ivory japanese justification opportunity keyword khaki known korean :lang lang (pseudo-class) last-baseline alignment last-baseline content-alignment last baselines last-baseline self-alignment last baseline set last cross-axis baseline set last main-axis baseline set lavender lavenderblush lawngreen laying out in-place layout containment layout containment box layout-internal :left left leftover space legacy name alias legacy shorthand legacy value alias lemonchiffon letter in css-syntax-3 in css-text-3 lightblue lightcoral lightcyan lightgoldenrodyellow lightgray lightgreen lightgrey lightpink lightsalmon lightseagreen lightskyblue lightslategray lightslategrey light source lightsteelblue lightyellow lime limegreen limited max-content contribution limited min-content contribution linear easing function linear timing function line box line break in css-break-3 in css-text-3 line breaking line breaking process line-left linen line name line name set line orientation line-over line-relative line-relative direction line-right line-under :link link (pseudo-class) list-item list properties loading image local coordinate system local url flag logical height logical width longhand longhand property lowercase letter magenta main axis main-axis main-axis baseline set main dimension main-end main size main-size main size property main-start margin box margin edge margin::of a box mark marked indexes maroon mask border mask border image mask border image area mask image mask layer image mask painting area mask-position mask positioning area mask-size match matching transition delay matching transition duration matching transition-property value matching transition timing function max cross size max cross size property maximum allowed code point max inner height max inner width max main size max main size property max track sizing function may media media condition media-dependent import media feature media group media query media query list media query modifier media type mediumaquamarine mediumblue mediumorchid mediumpurple mediumseagreen mediumslateblue mediumspringgreen mediumturquoise mediumvioletred menu menutext message entity midnightblue min cross size min cross size property minimum contribution min inner height min inner width min main size min main size property mintcream min track sizing function mistyrose moccasin monolithic monospace multicol container multi-col line multicol line multi-column container multi-column formatting context multi-column layout multi-column line multi-column spanner multi-column spanning element multi-line flex container multiple declarations multiply must must not named cell token named grid area namespace prefix name-start code point natural aspect ratio natural dimension natural end-point natural height natural size natural width navajowhite navy nearest neighbor newline next input code point next input token next-sibling combinator next token non-ascii code point non-ascii ident code point 'none'::as display value non-overridable counter-style names non-printable code point non-replaced non-replaced element normal normalize into a token stream null cell token number numeric data types objects object size negotiation occupied oldlace olive olivedrab opacity operating coordinate space optimal viewing region optional orange orangered orchid order-modified document order in css-display-3 in css-flexbox-1 orthogonal orthogonal flow other space separators outer box-shadow outer display type outer edge outline out of flow in css-display-3 in css21 out-of-flow output of the cascade output progress value over overflow overflow alignment overflow columns padding box padding edge padding::of a box @page page area page box page break page-context paged media 'paged' media group page selector pagination paint containment paint containment box palegoldenrod palegreen paleturquoise palevioletred papayawhip parent parent box parse parse a block's contents parse a comma-separated list according to a css grammar parse a comma-separated list of component values parse a component value parse a css stylesheet parse a declaration parse a list parse a list of component values parse a list of declarations parse a list of rules parse a rule parse a style block's contents parse a stylesheet parse a stylesheet's contents parse error parse something according to a css grammar parsing a list participates in baseline alignment pass through filter peachpuff pending on the environment pending-substitution value percentage peru physical physical bottom physical dimensions physical direction physical left physical right physical top physical unit pink pixel pixel unit plum positional alignment positioned element/box positioning scheme post-multiplied post-multiply powderblue preceding element pre-multiplied pre-multiply preserved tokens preserved white space primary filter primitive tree principal block-level box principal box principal writing mode process propagate propagation proper table child proper table row parent property in css-cascade-4, for CSS in css21 property declarations pseudo-classes pseudo-classes:::active pseudo-classes:::focus pseudo-classes:::hover pseudo-classes:::lang pseudo-classes:::link pseudo-classes:::visited pseudo-class:::first pseudo-class:::left pseudo-class:::right pseudo-elements pseudo-elements:::after pseudo-elements:::before pseudo-elements:::first-letter pseudo-elements:::first-line purple quad width qualified rule range context recommended reconsume the current input code point reconsume the current input token red reference box in css-shapes-1, for in css-transforms-1 reference pixel region break relative length relative length unit relative positioning relative units remaining fragmentainer extent remaining free space rendered content render with a fallback font face render with an invisible fallback font face replaced replaced element in css-display-3 in css21 representation required reset implicitly reset-only sub-property re-snap resolved type restore a mark reversing-adjusted start value reversing shortening factor :right right root root element rosybrown row group box row groups royalblue rule run-in in css-display-3 in css21 run-in box run-in sequence running transition saddlebrown salmon sandybrown sans-serif scaled flex shrink factor scope screen reader scrollbar scroll snap scroll snap area scroll snap container scroll snapport scroll snap position seagreen seashell segment break selector selector::match selector::subject of self-alignment self-alignment properties semitone sequence of simple selectors serialize an value serif set entries set explicitly shall shall not shared alignment context sheet shorthand shorthand property in css-cascade-4 in css21 should should not sibling sideways typesetting sienna silver simple block simple selector single-line flex container size containment size containment box sizing as if empty sizing function skyblue slateblue slategray slategrey small small kana snow soft wrap break soft wrap opportunity source source document spaces space-separated matching space to fill span count specified size specified size suggestion in css-flexbox-1 in css-grid-1 specified value in css-cascade-4 in css21 'speech' media group spread break spread distance springgreen stacking context stack level start starting point startmost starts with an ident sequence starts with a number starts with a valid escape start time start value start with an ident sequence start with a number statement at-rule 'static' media group static-position rectangle in css-align-3 in css-flexbox-1 steelblue step easing function step position steps stop or comma stretched strictness value stroke bounding box structural pseudo-classes strut size stuck on the environment style attribute style change event style rule style sheet in css-namespaces-3 in css-speech-1 in css21 stylesheet subject (of selector) subjects of the selector sub-property subsequent-sibling combinator substitute a var() support in css-conditional-3, for CSS in css-fonts-4 supports queries switch the fontfaceset to loaded switch the fontfaceset to loading synthesize baseline synthesized baseline system fonts table caption box table element tables tabs tab size tab stop tabular container 'tactile' media group tan target main size teal text/css text node text sequence textual data types thistle threeddarkshadow threedface threedhighlight threedlightshadow threedshadow timing function tokenization tokenize tokenizer tokens token stream tomato top tracking track list track section track sizing algorithm track sizing function transfer function element transfer function element attributes transferred size suggestion in css-flexbox-1 in css-grid-1 transformable element transformation matrix transformed element transitionable transition origin transparent trash token triangle trinary turquoise type selector typeset sideways typesetting sideways typesetting upright typeset upright typographic character typographic character unit typographic letter unit typographic mode ua in css-speech-1 in css21 in css21 ua origin ua-origin ua style sheet under unforced break universal selector unknown unoccupied upper-alpha-legal uppercase letter upright typesetting url use a negative sign used value in css-cascade-4 in css21 in css21 user user agent in css-speech-1 in css21 in css21 user-agent origin user-agent style sheet user coordinate system user origin user-origin user style sheet uses a negative sign valid image validity valid style sheet value value definition syntax var() substitution vertical axis vertical block flow vertical dimension vertical offset vertical-only vertical script vertical typographic mode vertical writing mode viewport viewport-percentage lengths violet :visited visited (pseudo-class) visual angle unit visual formatting model 'visual' media group volume wheat white whitesmoke white space whitespace white space characters width window windowframe windowtext word separator word-separator character would start an ident sequence would start a number would start a unicode-range wrap in css-shapes-1 in css-text-3 wrapping in css-shapes-1 in css-text-3 writing mode x-axis x-height y-axis yellow yellowgreen absolute in css-speech-1, for voice-pitch in css-speech-1, for voice-range add additive alias all allow-end all-petite-caps all-scroll all-small-caps alpha alphabetic alternate alternate-reverse always in css-scroll-snap-1, for scroll-snap-stop in css-speech-1, for speak annotation() anywhere in css-text-3, for line-break in css-text-3, for overflow-wrap arabic-indic armenian in css-counter-styles-3, for in css21 aural auto in css-align-3, for align-self in css-align-3, for justify-self in css-backgrounds-3, for background-size in css-backgrounds-3, for border-image-width in css-break-3, for break-before, break-after in css-break-3, for break-inside, page-break-inside in css-counter-styles-3, for @counter-style/range in css-counter-styles-3, for @counter-style/speak-as in css-flexbox-1, for align-items, align-self in css-flexbox-1, for flex-basis in css-fonts-4, for @font-face/font-display in css-fonts-4, for font-kerning in css-fonts-4, for font-optical-sizing in css-fonts-4, for font-synthesis-position in css-fonts-4, for font-synthesis-small-caps in css-fonts-4, for font-synthesis-style in css-fonts-4, for font-synthesis-weight in css-grid-1, for in css-grid-1, for grid-template-columns, grid-template-rows in css-images-3, for image-rendering in css-multicol-1, for column-count in css-multicol-1, for column-fill in css-multicol-1, for column-width in css-scroll-snap-1, for scroll-padding, scroll-padding-inline, scroll-padding-inline-start, scroll-padding-inline-end, scroll-padding-block, scroll-padding-block-start, scroll-padding-block-end in css-speech-1, for speak in css-speech-1, for voice-duration in css-text-3, for hyphens in css-text-3, for line-break in css-text-3, for text-align-last in css-text-3, for text-justify in css-text-decor-3, for text-underline-position in css-ui-3, for caret-color in css-ui-3, for cursor in css-will-change-1, for will-change in filter-effects-1, for color-interpolation-filters auto-fill auto-fit [auto-flow && dense?] ? / avoid avoid-column avoid-page avoid-region backwards balance balance-all baseline bengali bidi-override blink block bold bolder border-box both bottom braille break-all break-spaces break-word in css-text-3, for overflow-wrap in css-text-3, for word-break bullets cambodian capitalize caption cell center in css-align-3, for , , justify-self, align-self, justify-content, align-content in css-backgrounds-3, for background-position in css-flexbox-1, for align-content in css-flexbox-1, for align-items, align-self in css-flexbox-1, for justify-content in css-scroll-snap-1, for scroll-snap-align in css-speech-1, for voice-balance in css-text-3, for text-align in css-transforms-1, for transform-origin ch character-variant(#) child ch unit circle cjk-decimal cjk-earthly-branch cjk-heavenly-stem cjk-ideographic clip clone close-quote closest-corner closest-side cm coarse collapse color color-burn color-dodge col-resize column column-reverse common-ligatures condensed contain in css-backgrounds-3, for background-size in css-images-3, for object-fit content in css-contain-1, for contain in css-flexbox-1, for flex-basis content-box contents context-menu contextual copy cover in css-backgrounds-3, for background-size in css-images-3, for object-fit crisp-edges crosshair cursive cyclic dark in css-fonts-4, for base-palette in css-fonts-4, for font-palette darken dashed decimal in css-counter-styles-3, for in css21 decimal-leading-zero in css-counter-styles-3, for in css21 default deg dense devanagari diagonal-fractions difference digits digits ? digits ? disc in css-counter-styles-3, for in css21 disclosure-closed disclosure-open discretionary-ligatures distribute dot dotted double double-circle dpcm dpi dppx each-line ease ease-in ease-in-out ease-out ellipse ellipsis em embed embossed emoji em unit end e-resize ethiopic-numeric evenodd ew-resize ex exclude exclusion expanded extends extra-condensed extra-expanded ex unit fallback fantasy farthest-corner farthest-side fast in css-speech-1, for voice-rate in mediaqueries-4, for @media/update female fill fill-box filled fine first first baseline fit-content() fixed flex flex-end in css-align-3, for , , justify-self, align-self, justify-content, align-content in css-flexbox-1, for align-content in css-flexbox-1, for align-items, align-self in css-flexbox-1, for justify-content flex-start in css-align-3, for , , justify-self, align-self, justify-content, align-content in css-flexbox-1, for align-content in css-flexbox-1, for align-items, align-self in css-flexbox-1, for justify-content flip flow flow-root force-end forwards fr from-image fr unit full-size-kana full-width generic(fangsong) generic(kai) generic(khmer-mul) generic(nastaliq) georgian in css-counter-styles-3, for in css21 grab grabbing grad grid in css-display-3, for display, in css-grid-1, for display / [auto-flow && dense?] ? / groove gujarati gurmukhi handheld hanging hard-light hebrew help hidden high in css-speech-1, for voice-pitch in css-speech-1, for voice-range high-quality hiragana hiragana-iroha historical-forms historical-ligatures horizontal-tb hover hue hz icon in infinite inherit initial inline inline-block inline-flex inline-grid in css-display-3, for display, in css-grid-1, for display inline-table in css-display-3, for display, in css21 inset [|] && ? && ? inter-character interlace intersect inter-word invert in css-ui-3, for outline-color in css21 isolate isolate-override italic japanese-formal japanese-informal jis04 jis78 jis83 jis90 jump-both jump-end jump-none jump-start justify justify-all kannada katakana katakana-iroha keep-all khmer khz korean-hangul-formal korean-hanja-formal korean-hanja-informal landscape lao last last baseline layout left in css-align-3, for justify-content, justify-self, justify-items in css-backgrounds-3, for background-position in css-break-3, for break-before, break-after in css-speech-1, for voice-balance in css-text-3, for text-align in css-text-decor-3, for text-emphasis-position in css-text-decor-3, for text-underline-position in css-transforms-1, for transform-origin leftwards legacy {2} light in css-fonts-4, for base-palette in css-fonts-4, for font-palette lighten lighter linear lineargrb [? ? ?]+ [/]? line-through lining-nums list-item literal-punctuation local loose loud low in css-speech-1, for voice-pitch in css-speech-1, for voice-range lower-alpha lower-armenian lowercase lower-greek in css-counter-styles-3, for in css21 lower-latin in css-counter-styles-3, for in css21 lower-roman in css-counter-styles-3, for in css21 ltr luminance luminosity malayalam male mandatory manual margin-box match-parent match-source math max-content medium in css-backgrounds-3, for , border-width, border-top-width, border-left-width, border-bottom-width, border-right-width, border in css-speech-1, for pause-before, pause-after in css-speech-1, for rest-before, rest-after in css-speech-1, for voice-pitch in css-speech-1, for voice-range in css-speech-1, for voice-rate in css-speech-1, for voice-volume menu message-box min-content minmax() mixed mm moderate mongolian monospace move ms multiply myanmar ne-resize nesw-resize neutral never no-clip no-close-quote no-common-ligatures no-contextual no-discretionary-ligatures no-drop no-historical-ligatures none in css-animations-1, for animation-fill-mode in css-animations-1, for animation-name in css-backgrounds-3, for , border-style, border-top-style, border-left-style, border-bottom-style, border-right-style, border in css-backgrounds-3, for background-image in css-backgrounds-3, for box-shadow in css-contain-1, for contain in css-display-3, for display, in css-flexbox-1, for flex in css-fonts-4, for font-kerning in css-fonts-4, for font-optical-sizing in css-fonts-4, for font-size-adjust in css-fonts-4, for font-synthesis-position in css-fonts-4, for font-synthesis-small-caps in css-fonts-4, for font-synthesis-style in css-fonts-4, for font-synthesis-weight in css-fonts-4, for font-variant in css-fonts-4, for font-variant-ligatures in css-grid-1, for grid-template in css-grid-1, for grid-template-areas in css-grid-1, for grid-template-rows, grid-template-columns in css-images-3, for image-orientation in css-images-3, for object-fit in css-multicol-1, for column-span in css-scroll-snap-1, for scroll-snap-align in css-scroll-snap-1, for scroll-snap-type in css-shapes-1, for shape-outside in css-speech-1, for pause-before, pause-after in css-speech-1, for rest-before, rest-after in css-speech-1, for voice-stress in css-text-3, for hanging-punctuation in css-text-3, for hyphens in css-text-3, for text-justify in css-text-3, for text-transform in css-text-decor-3, for text-decoration-line in css-text-decor-3, for text-emphasis-style in css-transitions-1, for transition-property in css-ui-3, for cursor in css-writing-modes-4, for text-combine-upright in mediaqueries-4, for @media/hover in mediaqueries-4, for @media/overflow-block in mediaqueries-4, for @media/overflow-inline in mediaqueries-4, for @media/pointer in mediaqueries-4, for @media/update 'none'::as border style nonzero no-open-quote no-punctuation no-repeat normal in compositing-1, for in css-align-3, for align-self in css-align-3, for justify-content, align-content in css-align-3, for justify-self in css-align-3, for row-gap, column-gap, gap in css-animations-1, for animation-direction in css-fonts-4, for font-feature-settings in css-fonts-4, for font-kerning in css-fonts-4, for font-language override in css-fonts-4, for font-palette in css-fonts-4, for font-style in css-fonts-4, for font-variant in css-fonts-4, for font-variant-alternates in css-fonts-4, for font-variant-caps in css-fonts-4, for font-variant-east-asian in css-fonts-4, for font-variant-emoji in css-fonts-4, for font-variant-ligatures in css-fonts-4, for font-variant-numeric in css-fonts-4, for font-variant-position in css-fonts-4, for font-weight in css-fonts-4, for font-width in css-scroll-snap-1, for scroll-snap-stop in css-speech-1, for speak-as in css-speech-1, for voice-rate in css-speech-1, for voice-stress in css-text-3, for letter-spacing in css-text-3, for line-break in css-text-3, for overflow-wrap in css-text-3, for white-space in css-text-3, for word-break in css-text-3, for word-spacing in css-writing-modes-4, for unicode-bidi not not-allowed nowrap in css-flexbox-1, for flex-wrap in css-text-3, for white-space n-resize ns-resize numbers numeric nw-resize nwse-resize objectboundingbox oblique ? old oldstyle-nums only open open-quote optional ordinal oriya ornaments() outset over overlay overline p3 padding-box page paged paint paused pc persian petite-caps pixelated plaintext pointer portrait pre pre-line preserve pre-wrap print progress progressive projection proportional-nums proportional-width proximity pt px q rad rec2020 recto reduced region rem rem unit repeat repeat-x repeat-y reverse revert ridge right in css-align-3, for justify-content, justify-self, justify-items in css-backgrounds-3, for background-position in css-break-3, for break-before, break-after in css-speech-1, for voice-balance in css-text-3, for text-align in css-text-decor-3, for text-emphasis-position in css-text-decor-3, for text-underline-position in css-transforms-1, for transform-origin rightwards round row in css-flexbox-1, for flex-direction in css-grid-1, for grid-auto-flow row-resize row-reverse rtl ruby ruby-base ruby-base-container ruby-text ruby-text-container run-in running s safe sans-serif saturation scale-down screen in compositing-1, for in mediaqueries-4, for @media scroll scroll-position self-end self-start semi-condensed semi-expanded in css-speech-1, for voice-pitch in css-speech-1, for voice-range se-resize serif sesame sideways sideways-lr sideways-right sideways-rl silent simp-chinese-formal simp-chinese-informal simplified size slashed-zero slice slow in css-speech-1, for voice-rate in mediaqueries-4, for @media/update small-caps small-caption smooth soft soft-light solid space space-around space-between space-evenly span && [||] span && [||] speech spell-out square in css-counter-styles-3, for in css21 s-resize srgb stacked-fractions start status-bar step-end step-start stretch strict in css-contain-1, for contain in css-text-3, for line-break + stroke-box strong styleset(#) stylistic() sub subtract super swap swash() sw-resize symbolic system-ui table in css-display-3, for display, in css21 table-caption in css-display-3, for display, in css21 table-cell in css-display-3, for display, in css21 table-column in css-display-3, for display, in css21 table-column-group in css-display-3, for display, in css21 table-footer-group in css-display-3, for display, in css21 table-header-group in css-display-3, for display, in css21 table-row in css-display-3, for display, in css21 table-row-group in css-display-3, for display, in css21 tabular-nums tamil telugu text in css-fonts-4, for font-variant-emoji in css-ui-3, for cursor thai thick thin tibetan titling-caps top | trad-chinese-formal trad-chinese-informal traditional triangle tty turn tv ui-monospace ui-rounded ui-sans-serif ui-serif ultra-condensed ultra-expanded under underline unicase unicode unsafe unset upper-alpha upper-armenian uppercase upper-latin in css-counter-styles-3, for in css21 upper-roman in css-counter-styles-3, for in css21 upright userspaceonuse verso vertical-lr vertical-rl vertical-text vh view-box visible vmax vmin vw wait weak words wrap wrap-reverse w-resize x x-fast x-high in css-speech-1, for voice-pitch in css-speech-1, for voice-range x-loud x-low in css-speech-1, for voice-pitch in css-speech-1, for voice-range x-slow x-soft x-strong x-weak y young zoom-in zoom-out Special thanks to Florian Rivoal for creating the initial draft of the § 3.2.1 Experimentation and Unstable Features recommendations.

Conformance requirements are expressed with a combination of descriptive assertions and RFC 2119 terminology. The key words “MUST”, “MUST NOT”, “REQUIRED”, “SHALL”, “SHALL NOT”, “SHOULD”, “SHOULD NOT”, “RECOMMENDED”, “MAY”, and “OPTIONAL” in the normative parts of this document are to be interpreted as described in RFC 2119. However, for readability, these words do not appear in all uppercase letters in this specification. All of the text of this specification is normative except sections explicitly marked as non-normative, examples, and notes. [RFC2119]Examples in this specification are introduced with the words “for example” or are set apart from the normative text with class="example", like this: Informative notes begin with the word “Note” and are set apart from the normative text with class="note", like this: Note, this is an informative note.Advisements are normative sections styled to evoke special attention and are set apart from other normative text with , like this: UAs MUST provide an accessible alternative. A style sheet is conformant to this specification if all of its statements that use syntax defined in this module are valid according to the generic CSS grammar and the individual grammars of each feature defined in this module. A renderer is conformant to this specification if, in addition to interpreting the style sheet as defined by the appropriate specifications, it supports all the features defined by this specification by parsing them correctly and rendering the document accordingly. However, the inability of a UA to correctly render a document due to limitations of the device does not make the UA non-conformant. (For example, a UA is not required to render color on a monochrome monitor.) An authoring tool is conformant to this specification if it writes style sheets that are syntactically correct according to the generic CSS grammar and the individual grammars of each feature in this module, and meet all other conformance requirements of style sheets as described in this module. So that authors can exploit the forward-compatible parsing rules to assign fallback values, CSS renderers must treat as invalid (and ignore as appropriate) any at-rules, properties, property values, keywords, and other syntactic constructs for which they have no usable level of support. In particular, user agents must not selectively ignore unsupported component values and honor supported values in a single multi-value property declaration: if any value is considered invalid (as unsupported values must be), CSS requires that the entire declaration be ignored.Once a specification reaches the Candidate Recommendation stage, non-experimental implementations are possible, and implementors should release an unprefixed implementation of any CR-level feature they can demonstrate to be correctly implemented according to spec. To establish and maintain the interoperability of CSS across implementations, the CSS Working Group requests that non-experimental CSS renderers submit an implementation report (and, if necessary, the testcases used for that implementation report) to the W3C before releasing an unprefixed implementation of any CSS features. Testcases submitted to W3C are subject to review and correction by the CSS Working Group. The look of an HTML form can be greatly improved with CSS: Use the width property to determine the width of the input field: First Name Try it Yourself » The example above applies to all elements. If you only want to style a specific input type, you can use attribute selectors: input[type=text] - will only select text fields input[type=password] - will only select password fields input[type=number] - will only select number fields etc.. Use the padding property to add space inside the text field. Tip: When you have many inputs after each other, you might also want to add some margin, to add more space outside of them: First Name Last Name input[type=text] { width: 100%; padding: 12px 20px; margin: 8px 0; box-sizing: border-box; } Try it Yourself » Note that we have set the box-sizing property to border-box. This makes sure that the padding and eventually borders are included in the total width and height of the elements. Read more about the box-sizing property in our CSS Box Sizing chapter. Bordered Inputs Use the border property to change the border size and color, and use the border-radius property to add rounded corners: First Name If you only want a bottom border, use the border-bottom property: First Name Colored Inputs Use the background-color property to add a background color to the input, and the color property to change the text color: Focused Inputs By default, some browsers will add a blue outline around the input when it gets focus (clicked on). You can remove this behavior by adding outline: none; to the input. Use the :focus selector to do something with the input field when it gets focus: Input with icon/image If you want an icon inside the input, use the background-image property and position it with the background-position property. Also notice that we add a large left padding to reserve the space of the icon: input[type=text] { background-color: white; background-image: url('searchicon.png'); background-position: 10px 10px; background-repeat: no-repeat; padding-left: 40px; } Try it Yourself » Animated Search Input In this example we use the CSS transition property to animate the width of the search input when it gets focus. You will learn more about the transition property later, in our CSS Transitions chapter. input[type=text] { transition: width 0.4s ease-in-out;}input[type=text]:focus { width: 100%;} Try it Yourself » Styling Textareas Tip: Use the resize property to prevent textareas from being resized (disable the "grabber" in the bottom right corner): Some text... textarea { width: 100%; height: 150px; padding: 12px 20px; box-sizing: border-box; border: 2px solid #ccc; border-radius: 4px; background-color: #f8f8f8; resize: none; } Try it Yourself » Styling Select Menus select { width: 100%; padding: 16px 20px; border: none; border-radius: 4px; background-color: #f1f1f1; } Try it Yourself » Styling Input Buttons input[type=button], input[type=submit], input[type=reset] { background-color: #04AA6D; border: none; color: white; padding: 16px 32px; text-decoration: none; margin: 4px 2px; cursor: pointer; }/* Tip: use width: 100% for full-width buttons */ Try it Yourself » For more information about how to style buttons with CSS, read our CSS Buttons Tutorial. Responsive Form Resize the browser window to see the effect. When the screen is less than 600px wide, make the two columns stack on top of each other instead of next to each other. Advanced: The following example uses media queries to create a responsive form. You will learn more about this in a later chapter. Try it Yourself »