Click Here

```
Share copy and redistribute the material in any medium or format for any purpose, even commercially. Adapt remix, transform, and build upon the material for any purpose, even commercially. The licensor cannot revoke these freedoms as long as you follow the license terms. Attribution You must give appropriate credit, provide a link to the license,
and indicate if changes were made . You may do so in any reasonable manner, but not in any way that suggests the licensor endorses you or your use. ShareAlike If you remix, transform, or build upon the material, you must distribute your contributions under the same license as the original. No additional restrictions You may not apply legal terms or
technological measures that legally restrict others from doing anything the license permits. You do not have to comply with the license for elements of the material in the public domain or where your use is permitted by an applicable exception or limitation. No warranties are given. The license may not give you all of the permissions necessary for
your intended use. For example, other rights such as publicity, privacy, or moral rights may limit how you use the material. In this example, we use the line-height property with a value that is equal to the height property with a value that is equal to the height property to center the div element: I am vertically and horizontally centered. 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 which one or more style sheets apply. A source document to which one or more style sheets apply. A source document to which one or more style sheets apply. A source document to which one or more style sheets apply. A source document to which one or more style sheets apply. A source document to which one or more style sheets apply.
documents and associated style sheets. An authoring tool is a User Agent that generates style sheets. User A user agent to view, hear, or otherwise use the 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. 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 editors draft. The transitions Working Draft (LCWD) phase. The CSSWG transitions Working Draft (LCWD) phase 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 Directors 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 to test the specification 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 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 Editors 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 Editors 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 Editors 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 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 CSS24.1,
CSS24.4, and CSS2G, 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 [CSS-CONDITIONAL-3]
Extends and supersedes CSS27.2, updating the definition of @media rules to allow nesting and introducing the @supports rule for feature-support queries. Selectors Level 3 [SELECTORS-3] Replaces CSS25 and CSS26.4.3, 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-3]. Describes how to collate style rules and assign values to all properties on all elements.
CSS Values and Units Level 3 [CSS-VALUES-3] Extends and supersedes CSS21.4.2.1, CSS24.3, and CSS2A.2.13, defining CSSs 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 CSS28.1, 8.2, 8.3 (but not 8.3.1), and 8.4. CSS Color Level 4 [CSS-COLOR-4] Extends and supersedes CSS24.3.6, CSS214.1, and CSS218.2, 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, 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 CSS215 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. Hebrew or Arabic), bidirectional (e.g. Hebrew or Arabic), bidirectional (e.g. mixed Latin and Arabic) and vertical (e.g. Hebrew or Arabic), bidirectional (e.g. mixed Latin and Arabic) and vertical (e.g. Hebrew or Arabic), bidirectional (e.g. mixed Latin and Arabic) and vertical (e.g. Asian scripts).
 Introduces multi-column flows to CSS layout. CSS Flexible Box Module Level 3 [CSS-FLEXBOX-1] Introduces a flexible linear layout model for CSS. CSS Basic User Interface Module Level 3 [CSS-FLEXBOX-1] Introduces a flexible linear layout model for CSS. CSS Basic User Interface Module Level 3 [CSS-FLEXBOX-1] Introduces a flexible linear layout model for CSS. CSS Basic User Interface Module Level 3 [CSS-FLEXBOX-1] Introduces a flexible linear layout model for CSS. CSS Basic User Interface Module Level 3 [CSS-FLEXBOX-1] Introduces a flexible linear layout model for CSS. CSS Basic User Interface Module Level 3 [CSS-FLEXBOX-1] Introduces a flexible linear layout model for CSS. CSS Basic User Interface Module Level 3 [CSS-FLEXBOX-1] Introduces a flexible linear layout model for CSS. CSS Basic User Interface Module Level 3 [CSS-FLEXBOX-1] Introduces a flexible linear layout model for CSS. CSS Basic User Interface Module Level 3 [CSS-FLEXBOX-1] Introduces a flexible linear layout model for CSS. CSS Basic User Interface Module Level 3 [CSS-FLEXBOX-1] Introduces a flexible linear layout model for CSS. CSS Basic User Interface Module Level 3 [CSS-FLEXBOX-1] Introduces a flexible linear layout model for CSS. CSS Basic User Interface Module Level 3 [CSS-FLEXBOX-1] Introduces a flexible linear layout model for CSS. CSS Basic User Interface Module Level 3 [CSS-FLEXBOX-1] Introduces a flexible linear layout model for CSS Basic User Interface Module Level 3 [CSS-FLEXBOX-1] Introduces a flexible linear layout model for CSS Basic User Interface Module Level 3 [CSS-FLEXBOX-1] Introduces a flexible linear layout model for CSS Basic User Interface Module Level 3 [CSS-FLEXBOX-1] Introduces a flexible linear layout model for CSS Basic User Interface Module Level 3 [CSS-FLEXBOX-1] Introduces a flexible linear layout model for CSS Basic User Interface Module Level 3 [CSS-FLEXBOX-1] Introduces a flexible linear layout model for CSS Basic User Interface Module Level 3 [CSS-FLEXBOX-1] Introduces a flexible linear layout model for CSS Basic U
Containment Module Level 1 [CSS-CONTAIN-1] Introduces the contain property, which enforces the independent CSS processing of an elements 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 dont 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], 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 model, the children of a grid container can be positioned into arbitrary slots in a predefined flexible or fixed-size layout model, the children of a grid container can be positioned into arbitrary slots in a predefined flexible or fixed-size layout model, the children of a grid container can be positioned into arbitrary slots in a predefined flexible or fixed-size layout model, the children of a grid container can be positioned into arbitrary slots in a predefined flexible or fixed-size layout model, the children of a grid container can be positioned into arbitrary slots in a predefined flexible or fixed-size layout model, the children of a grid container can be positioned into arbitrary slots in a predefined flexible or fixed-size layout model, the children of a grid container can be positioned into arbitrary slots in a predefined flexible or fixed-size layout model, the children of a grid container can be positioned into arbitrary slots in a predefined flexible or fixed-size layout model, the children of a grid container can be positioned into arbitrary slots in a predefined flexible or fixed-size layout model.
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 CSS29.1.2, 9.2.1 (but not 9.2.2.1), 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 CSS213.3. 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, flex layou
non-rectangular wrapping shapes. CSS Text Module Level 3 [CSS-TEXT-3] Extends and supersedes CSS216 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-3] Extends and supersedes CSS216 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-3] Extends and supersedes CSS216 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-3] Extends and supersedes CSS216 excepting 16.3, defining properties for text manipulation and specifying their processing model. It covers line breaking, justification and specifying their processing model.
TEXT-DECOR-3] Extends and supersedes CSS216.3, 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 CSS211.1.2 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 CSS2A, overhauling the (non-normative) speech rendering chapter. CSS View Transitions Module Level 1 [CSS-SPEECH-1] Replaces CSS2A, overhauling the (non-normative) speech rendering chapter.
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-WILL-CHANGE-1] Introduces a performance hint
   [CSS-SIZING-3] Overlays and extends CSS10., 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 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, absolute positioning, absolute positioning, and fixed positioning, absolute positioning, and fixed positioning, and fixed positioning, and fixed positioning, absolute positioning, and fixed positioning positioning positioning positioning positioning positioning positioning positioning p
 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 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-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 of each 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. 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 CSS2 specification is technically a W3C Recommendation stage.
 instead of expanding an already unwieldy errata list, the CSS Working Group chose to definitive definition. Once CSS2.1 became Candidate Recommendationeffectively though not officially the same level of stability as CSS2obsoleted 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 builds on CSS Level 3 build
CSS modules will not contradict the CSS2.1 specification: only that they will add functionality and refine definitions. As each module is completed 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 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 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 implementations of CSS, even if that subset is an official profile, must follow the forward-compatible parsing rules for partial implementations.
 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, keywords, and other syntactic constructs for which they have no usable level
of support. In particular, user agents must not selectively ignore unsupported values and honor supported 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 hasnt 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 vendors 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, Firefoxs XUL-based UI, Apples iTunes UI, and Microsofts 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 hasnt yet stabilized), but at least three UAs implement the feature(or
a UA has broken the other rules and shipped for broad usean unstable or otherwise non-standard feature in a production release), and the implementations have rough interoperability, and the implementation have rough interoperability and interoperability in the implementation have rough interoperabilities.
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 its 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 its still experimental doesnt 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, 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 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 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 features.
reaches the Candidate Recommendation stage, implementers should release an unprefixed implementation of any CR-level feature they can demonstrate to be correctly implementation stage, 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 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 Groups 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 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 absolute length unit absolutely positioned element abstract dimensions :active active active active active active active duration active duration active duration active duration active duration active duration active alignment container
alignment context alignment subject 'all' media group alphabetic baseline ambiguous image url an+b ancestor anchor unit animation origin anima
atomic inline box atomic inline-level box atomic inline-level box at-rule attr() attribute 'audio' media group auditory icon augmented grid aural box model author origin 
automatic row position auto-placement auto-placement cursor available font faces available grid space avoid break values axis-lock axis value azure background painting area background positioning area background positioning area background background background background image background image background image background image background positioning area background positioning area background background background background image background image background image background image background positioning area background positioning area background background background image background image background image background image background positioning area background positioning area background positioning area background background image background image background image background image background image background image background positioning area background positioning area background positioning area background image background image background image background image background positioning area background positioning area background positioning area background image background image background image background image background positioning area background posit
alignment preference baseline content-alignment baseline self-alignment baseline set baseline se
 orientational transform bisque 'bitmap' media group black blanchedalmond ()-block []-block block in css-display-3 in css21 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 blockify block layout block-level block-level block-level block-level block-level block start block st
a box border radius border style border width box::ontent box::ontent box::content box::content box::content box::ontent box::
 background canvas surface captiontext captures snap positions cascade in css-cascade-4 in css21 cascade-4 in css21 cascade independent keyword cascaded independently cascaded value cascade origin central baseline 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 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 collapsed collapsed collapsed flex item
collapsed grid track collapsed gutter collapsed gutter collapsed track collapsed track collapsed track collapsed track collapsed track collapsed gutter collapsed gutter collapsed grid track collapsed gutter collapsed grid track collapsed grid track collapsed gutter collapsed grid track grid track collapsed grid track grid tr
declaration consume a function consume a list of component values consume an ident-like token consume an ident-lik
consume a style block's contents consume a unicode-range token consume the remnants of a bad declaration consume t
 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 counters, which is constraint crimson cross-axis cross-axis cross-axis baseline set cross dimension cross-end
cross size cross-size cross-size cross size property cross-start css bracketed range notation current input code point current input token
 current transformation matrix current value cursive cursive cursive cursive cursive darksoldenrod darkgreen darkslategrey darkslategrey darkslategrey darkturquoise darkviolet declaration in css-syntax-3, for
CSS in css21 declaration block declared value default sizing algorithm default sizing algorithm definite column position definite column position definite column span definite grid position definite position definite row span
definite size definite span descendant desce
 in css-speech-1 in css-style-attr document language document order 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::following 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 ending gontext establish an independent formatting context establishes an independent
 formatting context establishing an independent formatting context exact matching expanded name explicit grid column explicit grid row explicit grid row explicit grid track explicit grid track explicit grid row explicit grid row explicit grid track explicit grid track explicit grid row explicit grid 
 filter code points filtered code points filter primitive filter primitive attributes filter primitive attributes filter primitive tree filter primitive tree filter primitive attributes filter primitive attributes filter primitive tree filter primitive first baseline self-alignment first baseline self-alignment first baseline set :first child first-baseline first first primitive filter primitive filter primitive attributes filter primitive first f
child first cross-axis baseline set first formatted line :first-letter first-letter first-letter first-letter first-letter first symbol value fixed sizing function flex base size flex basis flex container flex direction flex factor 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 flow-relative flow font feature value declaration font-feature-value-type font
relative lengths font source font specific font-stretch font swap period footnote forced break f
 fragmentation context fragmentation direction fragmentation frost fragmentation root fragmented flow free space fuchsia full-size kana full-width fully inflexible function functional notation generate a counter generate a counter generate a counter generate a counter full-size kana full-width fully inflexible function functional notation gainsboro generate a counter generate a counter generate a counter full-size kana full-width fully inflexible function functional notation function f
 gradient center gradient function gradient function gradient line grapheme cluster gray graytext green greenyellow grey grid area grid column grid layout algorithm grid layout algorithm grid layout algorithm grid layout algorithm grid layout grid item placement algorithm grid layout grid layou
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 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) hyphenation hy
 start code point ignore ignored illegal implicit grid implicit grid column implicit grid lines implicit grid track implicit grid track implicit grid track implicit grid implicit grid implicit grid implicit grid implicit grid track implicit grid track implicit grid imp
 index indianred indigo infinitely growable in flow in-flow in css-display-3 in css21 infobackground infotext inherit 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 css-display-3 in css21 inline box in css-display-3 in css21 inline box in css-display-3 in css21 inline box in line-level content inline-level content inline-level content inline-level inline-level inline-level box in css-display-3 in css21 inline box in css-display-3 in css21 inline box in line-level content inline-level inline-lev
css-display-3 in css21 internal table element in css-display-3 in css21 interpreter in css-namespaces-3 in css21 interpreter in css-namespaces-3 in css-style-attr intrinsic dimensions intrinsic dimposition dimensions intrinsic dimensions intrinsic dimensions in
korean :lang lang (pseudo-class) last-baseline set last coss-axis baseline set last coss-axis baseline set last coss-axis baseline set last main-axis baseline set last main-axis baseline set last main-axis baseline set last coss-axis baseline set last co
 alias legacy shorthand legacy value alias lemonchiffon letter in css-syntax-3 in css-text-3 lightslategrey ligh
 content contribution linear easing function linear timing function line box line break in css-break-3 in css-text-3 line breaking line breaking line-relative direction line-relative direction line-relative direction line-relative direction line-relative line-relative line-relative direction line-relative line
   system local url flag logical neight logical width longhand longhand longhand property lowercase letter magenta main size main-size main-size main-size main size property main-size main size main size main-size main size main size main-size main size main-size main size main-size main size main 
 property max track sizing function may media condition media-dependent import media query media
message entity midnightblue min cross size min cross size min cross size min cross size min main size min track sizing function mistyrose moccasin monolithic monospace multi-column formatting context multi-column formattin
column layout multi-column line multi-column spanning element multi-line flex container multi-line flex container multi-line flex container multi-column spanning element multi-column spanning element multi-column spanning element multi-line flex container multi-line flex contai
nearest neighbor newline next input code point non-ascii ident code point n
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 flow other space separators outer box-shadow outer display type outer edge outline out of flow in css-flexbox-1.
display-3 in css21 out-of-flow output of the cascade output progress value over overflow alignment overflow columns padding box page break page-context paged media 'paged' media group page selector pagination paint containment paint containment box palegoldenrod palegreen
paleturquoise palevioletred papayawhip parse a list of component values parse a comma-separated list of component values parse a list of declarations parse a list of component values parse a list of declarations parse a list of declarations parse a list of component values parse a list of declarations parse a list of declarations parse a list of component values parse a list of component values parse a list of declarations parse a list of declarations parse a list of component values parse a list of component values parse a list of declarations parse a list of component values parse a li
rules parse a rule parse a style block's contents parse a stylesheet p
physical direction physical left physical right physical unit pink pixel unit plum positioned element/box positioning scheme post-multiplied pre-multiplied pre-multiplied pre-multiplied pre-multiplied pre-multiplied pre-multiplied post-multiplied post-multiplied pre-multiplied pre-multiplied pre-multiplied pre-multiplied post-multiplied post-multiplied pre-multiplied pre-multipl
principal box principal writing mode process propagate propagation proper table child pro
 pseudo-class:::left pseudo-class:::right pseudo-elements pseudo-elements:::after pseudo-elements:::first-letter pseudo-eleme
transforms-1 reference pixel region break relative length relative length unit relative positioning relative units remaining fragmentainer extent remaining
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 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 area scroll snap position seagreen seashell segment break selector::match selector::
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 containment box sizing as if empty sizing function skyblue slategray slategray slategray small small kana snow soft wrap break soft wrap
opportunity source source document spaces space-separated matching space to fill span count specified size suggestion in css-grid-1 specified size suggestion in css-grid-1 specified size specified size
sequence starts with a number starts with a number start with a nu
classes strut size stuck on the environment style attribute style change event style change event style sheet in css-speech-1 
to loaded switch the fontfaceset to loading synthesize 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 threeddightshadow
threedshadow timing function tokenization tokenization tokenize tokens token stream tomato top tracking track sizing algorithm track sizing function transfer function transfer function transfer function transfer function transfer function transfer function tokenize tokeniz
element transitionable transition origin transparent trash token triangle trinary turquoise type selector typeset sideways typesetting upright typographic haracter unit typographic character 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 css21 in c
sign valid image validity valid style sheet value definition syntax var() substitution vertical script vertical syntax var() substitution vertical style sheet value definition syntax var() substitution vertical style sheet value value definition syntax var() substitution vertical style sheet value value definition syntax var() substitution vertical style sheet value value value definition syntax var() substitution vertical style sheet value value
'visual' media group volume wheat white space white sp
x-axis x-height y-axis yellow yellowgreen absolute in css-speech-1, for voice-pitch in css-speech-1
in css-text-3, for overflow-wrap arabic-indic armenian in css-backgrounds-3, for break-inside, page-break-inside in css-backgrounds-3, for break-inside in css-backgrounds-3, for break-inside, page-break-inside in css-backgrounds-3, for break-inside in css-backgrounds-3, for break-inside in css-backgrounds-3, for break-inside, page-break-inside in css-backgrounds-3, for break-inside, page-break-inside in css-backgrounds-3, for break-inside in css-backgrounds-3, for break-inside, page-break-inside in css-backgrounds-3, for break-inside in css-backgrounds-3, for break-inside in css-backgrounds-3, for break-inside in css-backgrounds-3, for break-inside in css-backgrounds-3, for break-inside, page-break-inside in css-backgrounds-3, for break-inside in cs
counter-styles-3, for @counter-style/range in css-fonts-4, for font-synthesis in css-fonts-4, for font-synthesis in css-fonts-4, for font-synthesis-position in css-fonts-4, for font-synthesis-positi
small-caps in css-fonts-4, for font-synthesis-style in css-multicol-1, for column-count in css-multicol-1, for column-width in css-multicol-1, for column-width in css-multicol-1, for scroll-padding, scroll-
padding-inline, scroll-padding-inline-end, scroll-padding-inline-end, scroll-padding-block, scroll-padding-blo
underline-position in css-ui-3, for caret-color in css-ui-3, for caret-col
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-lexbox-1, for align-content in css-flexbox-1, for
flexbox-1, for justify-content in css-scroll-snap-1, for scroll-snap-1, for scroll-snap-1, for transform-origin ch character-variant(#) child ch unit circle cjk-decimal cjk-earthly-branch cjk-leavenly-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 contain in css-backgrounds-3, for background-size in css-images-3, for object-fit content in css-backgrounds-3, for background-size in css-images-3, for b
images-3, for object-fit crisp-edges crosshair cursive cyclic dark in css-fonts-4, for base-palette in css-counter-styles-3, for in css21 decimal in css-counter
for in css21 disclosure-closed 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 ex exclude exclusion expanded extends extra-condensed extra-conden
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 in css-flexbox-1, for align-content in css-flexbox-1, for a
flexbox-1, for justify-content in css-flexbox-1, for align-content in css-flexbox-1, for align-content 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(kai) generic(kai) generic(khmer-content flip flow flow-root force-end forwards fr from-image fr unit full-size-kana full-width generic(fangsong) generic(kai) generic(khmer-content flip flow flow-root force-end forwards fr from-image fr unit full-size-kana full-width generic(fangsong) generic(kai) gene
mul) generic(nastaliq) georgian in css-counter-styles-3, for in css-counter-styles-3, for display, in css-grid-1, for display, 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-flex inline-grid in css-display, in css-
isolate isolate-override italic japanese-formal japanese-informal jis04 jis78 jis83 jis90 jump-both jump-end jump-start justify-content, justify-self,
justify-items in css-backgrounds-3, for background-position in css-text-decor-3, for text-emphasis-position in css-text-decor-4, for text-emphasis-position in css-text-decor-4, for text-emphasis-position in css-text-decor-
palette in css-fonts-4, for font-palette lighter linear linearrab [???]+ [/]? line-through lining-nums list-item literal-punctuation local loose loud low in css-speech-1, for voice-range lower-letin in css-counter-styles-3, for in css-speech-1, for voice-range lower-letin in css-counter-styles-3, for in css-speech-1, for voice-pitch in css-speech-1, for voice-p
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-bottom-width, bor
after in css-speech-1, for rest-before, rest
common-ligatures no-contextual no-discretionary-ligatures no-drop no-historical-ligatures none in css-animations-1, for animations-1, for animation-style, border-top-style, b
css-backgrounds-3, for box-shadow in css-contain-1, for contain in css-fonts-4, for font-synthesis-small-caps in css-fonts-4, for font-synthesis-style in css-fonts-4, for font-synthesis-somall-caps in css-fonts-4, for font-synthesis-style in css-fonts-4, for font-synthesis-somall-caps in css-fonts-4, for font-synthesis-style in css-fonts-4, for font-synthesis-synthesis-synthesis-synthesis-synthesis-synthesis-synthesis-synthesis-synthesis-synthesis-synthesis-synthesis-synthesis-synthesis-synthesis-synthesis-synthesis-synthesis-synthesis-synthesis-synthesis-synthesis-synthesis-synthesis-synthesis-synthesis-synthesis-synthesis-synthesis-synthesis-synthesis-synthesis-synthesis-synthesis-synthesis-synthesis-synthesis-synthesis-synthesis-synthesis-synthesis-synthesis-synthesis-synthesis-synthesis-synthesis-synthesis-synthesis-synthesis-synthesis-synthesis-synthesis-synthesis-synthesis-synthesis-synthesis-synthesis-synthesis-synthesis-synthesis-synthesis-synthesis-synthesis-synthesis-synthesis-synthesis-synthesis-synthesis-synthesis-synthesis-synthesis-synthesis-synthesis-synthesis-synthesis-synthesis-synthesis-synthesis-synthesis-synthesis-synthesis-synthesis-synthesis-synthesis-synthesis-synthesis-synthesis-synthesis-synthesis-synthesis-synthesis-synthesis-synthesis-synthesis-synthesis-synthesis-synthesis-synthesis-synthesis-synthesis-synthesis-synthesis-synthesis-synthesis-synthesis-synthesis-synthesis-synthesis-synthesis-synthesis-synthesis-synthesis-synthesis-synthesis-synthesis-synthesis-synthesis-synthesis-synthesis-synthesis-synthesis-synthesis-synthesis-synthesis-synthesis-synthesis-synthesis-synthesis-synthesis-synthesis-synthesis-synthesis-synthesis-synthesis-synthesis-synthesis-synthesis-synthesis-synthesis-synthesis-synthesis-synthesis-synthesis-synthesis-synthesis-synthesis-synthesis-synthesis-synthesis-synthesis-synthesis-synthesis-synthesis-synthesis-synthesis-synthesis-synthesis-synthesis-synthesis-synthesis-synthesis-synthesis-synthesis-synthesis-synthesis-synthesis-synthesis-synthesis-synthesis-synt
4, for font-synthesis-weight in css-fonts-4, for font-variant in css-fonts-4, for font-variant in css-grid-1, for grid-template in c
 scroll-snap-align in css-scroll-snap-1, for scroll-snap-type in css-speech-1, for shape-outside in css-speech-1, for pause-after in css-speech-1, for hanging-punctuation in css-text-3, for hanging-punctuation in css-text-3, for hanging-punctuation in css-text-3, for hanging-punctuation in css-text-3, for text-justify in css-text-3, for text-transform in css-text-3, for hanging-punctuation in css-text-3, for hanging-punctuation in css-text-3, for text-justify in css-text-3, for text-transform in css-text-3, for hanging-punctuation in css-text-4, for hanging-punctuation in css-text-4, for hanging-p
decor-3, for text-decoration-line in css-text-decor-3, for text-emphasis-style in css-transitions-1, for cursor in css-writing-modes-4, for @media/overflow-block in mediaqueries-4, for @media/overflow-inline in mediaqueries-4, for cursor in css-writing-modes-4, for @media/overflow-inline in mediaqueries-4, for @media/overflow-inline in mediaqueries-4, for @media/overflow-inline in css-transitions-1, for text-emphasis-style in css-tr
```

@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 justify-self in css-align-3, for justify-self

```
css-fonts-4, for font-feature-settings in css-fonts-4, for font-variant in css-fonts-4, for font-variant in css-fonts-4, for font-variant-east in css-fonts-4, for font-variant-east in css-fonts-4, for font-variant-east in css-fonts-4, for font-variant in css-fonts-4, for font-variant-east in css-fonts-4, for font-variant in css-
 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-speech-1, for speak-as in css-speech-1, for voice-rate in css-speech-1, for voice-stress in css-speech-1, for speak-as in css-speech-1, for speak-as in css-speech-1, for voice-stress in css-speech-1, for speak-as in css-speech-1, for speak-as in css-speech-1, for voice-stress in css-speech-1, for speak-as in css-speech-1, fo
for line-break in css-text-3, for overflow-wrap in css-text-3, for white-space in css-text-4, for white-space in css-text-4,
oldstyle-nums only open open-quote optional ordinal original origi
region rem rem unit repeat repeat-x repeat-y reverse revert ridge right in css-background-position in css-background-position in css-background-position in css-text-decor-3, for break-after in css-background-position in css-backgro
text-underline-position in css-transforms-1, for transform-origin rightwards round row in css-grid-1, for grid-auto-flow row-resize row-reverse rtl ruby-text ruby-tex
 @media scroll scroll-position self-end self-end self-start semi-condensed semi-expanded in css-speech-1, for voice-rate in mediaqueries-4, for voice-range se-resize serif sesame sideways-right sideways
 for @media/update small-caps small-caps small-caption smooth soft soft-light solid space space-around space-between space-evenly span && [ || ] speech spell-out square in css-counter-styles-3, for in css-text-3, for line-break +
stroke-box strong styleset(#) stylistic() sub subtract super swap swash() sw-resize symbolic system-ui table-column in css-display-3, for display, in css21 table-column in css-display-3, for display in css21 table-column in css-display-3, for display-3, for di
table-footer-group in css-display-3, for display, in css21 table-row in css-display-3, for display in css21 table-row in css-display-3, for display-3, 
chinese-formal trad-chinese-informal trad-chinese-informal traditional triangle tty turn two ui-monospace ui-rounded ui-sans-serif ui-serif ultra-condensed ultra-expanded under underline unicase unicode unsafe unset upper-armenian 
userspaceonuse verso vertical-lr vertical-rl vertical-
 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, and RFC 2119 terminology.
 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 word Note and are set apart from the normative text with class="note", like this: 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. A renderer is conformant to this specification if, in addition to interpreting 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, keywords, and other syntactic constructs for which they have no usable level
of support. In particular, user agents must not selectively ignore unsupported values and honor supported 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 implementation of any CR-level feature they can demonstrate to be correctly implementations, the CSS Working Group requests that non-experimental
CSS renderers submit an implementation report (and, if necessary, the testcases used for that implementation of any CSS features. Testcases submitted to W3C are subject to review and correction by the CSS Working Group. Center elementshorizontally and vertically To horizontally and vertically and vertica
center a block element (like), use margin: auto; Setting the width of the element will prevent it from stretching out to the edges of its container. The element will be split equally between the two margins: This div element is centered. .center (margin: auto; width: 50%; border: 3px solid
green; padding: 10px;}Try it Yourself Note: Center aligning has no effect if the width property is not set (or set to 100%). Center Align TextTo just center; border: 3px solid green;}Try it Yourself Tip: For more examples on how to align text, see the CSS Text chapter. To
center an image, set left and right margin to auto and make it into a block element: img {display: block; margin-left: auto; width: 40%;}Try it Yourself Left and Right Align - Using positionOne method for aligning elements is to use position: absolute;: In my younger and more vulnerable years my father gave me some advice that I've
been turning over in my mind ever since..right{position: absolute; right: 0px; width: 300px; border: 3px solid #73AD21; padding: 10px;} Try it Yourself Note: Absolute positioned elements are removed from the normal flow, and can overlap elements. Left and Right Align - Using floatAnother method for aligning elements is to use the float
property:.right{float: right; width: 300px;border: 3px solid #73AD21;padding: 10px;}Try it Yourself The clearfix hack" to fix this (see example below). Then we can add the clearfix hack to the containing it, and it is floated, it will overflow outside of its containing it, and it is floated, it will overflow outside of its containing it, and it is floated, it will overflow outside of its containing it, and it is floated, it will overflow outside of its containing it, and it is floated, it will overflow outside of its containing it, and it is floated, it will overflow outside of its containing it, and it is floated, it will overflow outside of its containing it, and it is floated, it will overflow outside of its containing it, and it is floated, it will overflow outside of its containing it, and it is floated, it will overflow outside of its containing it, and it is floated, it will overflow outside of its containing it, and it is floated, it will overflow outside of its containing it, and it is floated, it will overflow outside of its containing it, and it is floated, it will overflow outside of its containing it, and it is floated, it will overflow outside of its containing it, and it is floated, it will overflow outside of its containing it, and it is floated, it will overflow outside of its containing it.
element to fix this problem: Center Vertically - Using padding: I am vertically center an element vertically in CSS. A simple solution is to use top and bottom padding: I am vertically and horizontally, use padding and text-align: center: I am
vertically and horizontally centered. .center { padding: 70px 0; border: 3px solid green; text-align: center; }Try it Yourself Center Vertically and horizontally centered. .center { line-height: 200px; height: 200px; border: 3px solid green; text-align: center of the height property with a value that is equal to the height property with a value that is equal to the height property with a value that is equal to the height property with a value that is equal to the height property with a value that is equal to the height property with a value that is equal to the height property with a value that is equal to the height property with a value that is equal to the height property with a value that is equal to the height property with a value that is equal to the height property with a value that is equal to the height property with a value that is equal to the height property with a value that is equal to the height property with a value that is equal to the height property with a value that is equal to the height property with a value that is equal to the height property with a value that is equal to the height property with a value that is equal to the height property with a value that is equal to the height property with a value that is equal to the height property with a value that is equal to the height property with a value that is equal to the height property with a value that is equal to the height property with a value that is equal to the height property with a value that is equal to the height property with a value that is equal to the height property with a value that is equal to the height property with a value that is equal to the height property with a value that is equal to the height property with a value that is equal to the height property with a value that is equal to the height property with a value that is equal to the height property with a value that is equal to the height property with a value that is equal to the height property with a value that is equal to the height property with a value
solid green; text-align: center; }/* If the text has multiple lines, add the following: */.center p {line-height are not options, another solution is to use positioning and the transform property: I am vertically and
horizontally centered. .center { height: 200px;position: relative; border: 3px solid green; }.center p {margin: 0; position: absolute; top: 50%; transform property in our 2D Transforms Chapter. Center Vertically - Using FlexboxYou can also use flexbox to
center things. Just note that flexbox is not supported in IE10 and earlier versions: I am vertically and horizontally centered. .center { display: flex; justify-content: center; align-items: center; height: 200px; border: 3px solid green; }Try it Yourself This blog will discuss 6 techniques (in order of adherence to best practices)that can be used to center
 align an element and when to use each one. Here, center align refers to placing the element at the horizontal and vertical center of its parent. 1. Using Transform When to use: When the width and height of the element are not known Card like modals where
there are multiple child elements with one focussed element at the center. The idea is to use absolute positioning with top and left are resolved based on the dimensions of the element
itself..center { position: absolute; top: 50%; left: 50%; transform: translate(-50%, -50%);} 2. Using Flex When to use: When the care a row of items that need to be centered. When the care are not known. When there is one or more elements to be centered. When the care are not known. When there are a row of items that need to be centered. When the care are not known. When there are a row of items that need to be centered like in a footer. The idea is to make the parent container are not known. When there are not known. When there are not known. When there are not known. When the care are not known are not known. When the care are not known ar
 flexbox and center the element along the horizontal and vertical directions using flex properties as follows..parent { display: flex; justify-content: center; align-items: center; we simply add the following line: 3. Using Negative Margin
 When to use:When the height and width of the element are known. The idea is to again use absolute positioning similar to the transform technique but we apply a negative margin of half the element's width and height instead of translate. $\sigma$ width and height instead
 -50px -100px; // Negative margin of half the height(#{$h} / 2) * -1) calc((#{$h} / 2) * -1); 4. Using Grid I recently discovered this cool technique from css-
tricks.com When to use: When there is only one child element that needs to be centered. The idea is to create a grid container, when margin to auto, margin-top and bottom is assigned the available space evenly, thus
centering the element..parent { display: grid;}.center { margin: auto;} Another way to center using grid, pointed out by Jacob:.parent { display: grid; place-items: center;} 5. Using Padding This technique is not recommended for center-align but it works. When to use:When the height of the parent element is known/fixed. When the height of the
center element is flexible. The idea is to set a fixed vertical padding for the container with fixed height is known and allow the child element to occupy max height? 6. Using Table-cell This
 technique is very rarely used today and may raise eyebrows. However, it does work. The idea is to force the parent to behave like a table cell using display: table-cell; vertical align property for vertical centering and margin auto;} This concludes the 6
 ways to center align elements. Bonus - Horizontal Centering Horizontal centering is often used in title styles and footers in combination with an even padding or margin. Using text-align When to use: When the center element is text content / inline-* type elementInline-* includes inline, inline-block, inline-flex, inline-table etc..parent { text-align:
center;} It can also center block type child elements but it is not a recommended practice. Using margin When to use:When the center element is a block element center for your reference. Please feel free to download and share. Let
me know in the comments if there are any more techniques that you have used/explored. Also, follow me on Twitter for more dev content! Style sheet languageThis article is about the markup styling language. For other uses, see CSS (disambiguation). "Pseudo-element" redirects here. For pseudo-element symbols in chemistry, see Skeletal formula
 Pseudoelement symbols. This article needs to be updated. Please help update this article to reflect recent events or newly available information. (November 2024) Cascading Style Sheets (CSS) Logo by The CSS-Next Community Group [1] Example of CSS source codeFilename extension. cssInternet mediatypetext/cssUniform Type
 Identifier(UTI)public.cssDeveloped by World Wide Web Consortium (W3C)Initial release17December 1996; 28 years ago(1996-12-17)Latest releaseCSS 3 is being developed as multiple separate modules. Regular snapshots summarize their status.7December 2023; 18 months ago(2023-12-07) Type of formatStyle sheet languageContainerforStyle rules
for HTML elementsContainedbyHTML DocumentsOpen format?YesWebsitew3.org/TR/CSS/#cssCascading Style SheetsStyle sheetCSS Zen GardenConceptsAnimationsBox modelFlexboxGridImage replacementPhilosophiesTablelessResponsive"Holy grail"ToolsSassLessStylusComparisonsStylesheet languagesCascading Style SheetsvteHTMLDynamic
HTMLHTML5articleaudiocanvasvideoXHTMLBasicMobile ProfileHTML elementmetadiv and spanblinkmarqueeHTML attributeHTML frameHTML editorCharacter encodingsnamed charactersUnicodeLanguage codeDocument Object ModelStyle sheetsCSSFont familyWeb
 colorsJavaScriptWebCLHTMXWeb3DWebGLWebGPUWebXRW3CValidatorWHATWGQuirks modeWeb storageRendering engineComparisonsDocument markup language used for specifying the presentation and styling of a document written in a markup
language such as HTML or XML (including XML dialects such as SVG, MathML or XHTML).[2] CSS is a cornerstone technology of the World Wide Web, alongside HTML and JavaScript.[3] CSS is designed to enable the separation of content accessibility, since
the content can be written without concern for its presentation; provide more flexibility and control in the specification of presentation of presentation characteristics; enable multiple web pages to share formatting by specifying the relevant CSS in a separate .css file, which reduces complexity and repetition in the structural content; and enable the .css file to be
 cached to improve the page load speed between the page in different styles for different rendering methods, such as on-screen, in print, by voice (via speech-based browser or screen reader), and on Braille-based tactile
devices. CSS also has rules for alternative formatting if the content is accessed on a mobile device.[5]The name cascading priority scheme to determine which declaration applies if more than one declaration of a property match a particular element. This cascading priority scheme is predictable. The CSS specifications are
maintained by the World Wide Web Consortium (W3C). Internet media type (MIME type) text/css is registered for use with CSS by RFC 2318 (March 1998). The W3C operates a free CSS validation service for CSS documents.[6]In addition to HTML, other markup languages support the use of CSS including XHTML, plain XML, SVG, and XUL. CSS is
also used in the GTK widget toolkit.CSS has a simple syntax and uses a number of English keywords to specify the names of various style properties. Main article: Style sheet (web development) a style sheet consists of a list of rules. Each rule or rule-set consists of one or more selectors, and a declaration block. "CSS class" redirects here. For non-CSS class a simple syntax and uses a number of English keywords to specify the names of various style properties.
use of element classes in HTML, see class attribute (HTML). In CSS, selectors declare which part of the markup a style applies to by matching tags and attributes in the markup itself. Selectors may apply to the following: all elements of a specific type, e.g. the second-level headers h2elements specified by attribute, in particular: id: an identifier unique
 within the document, denoted in the selector language by a hash prefix e.g. #idclass: an identifier that can annotate multiple elements in a document, denoted by a dot prefix e.g. .classname (the phrase "CSS class", although sometimes used, is a misnomer, as element classesspecified with the HTML class attribute a markup feature that is distinct
 from browsers' CSS subsystem and the related W3C/WHATWG standards work on document styles; see RDF and microformats for the origins of the "class" system of the web content model)elements depending on how they are placed relative to others in the document tree. Classes and IDs are case-sensitive, start with letters, and can include
 alphanumeric characters, hyphens, and underscores. A class may apply to any number of instances of any element. An ID may only be applied to a single element. Pseudo-classes are used in CSS selectors to permit formatting based on information that is not contained in the document tree. One example of a widely used pseudo-class is :hover, which
 identifies content only when the user "points to" the visible element, usually by holding the mouse cursor over it. It is appended to a selector as in a:hover or #elementid:hover. A pseudo-element makes a selection that may consist of partial elements, such as ::first-line or
 ::first-letter.[7] Note the distinction between the double-colon notation used for pseudo-elements by location, element type, id, class, or any combination thereof.[8] The order of the selectors is important. For example,
div .myClass {color: red;} applies to all elements of class myClass that are inside elements of class myClass that are inside elements of class myClass. This is not to be confused with concatenated identifiers such as div.myClass {color: red;} which applies to all elements of class myClass. The following
 table provides a summary of selector syntax indicating usage and the version of CSS that introduced it.[9] PatternMatchesFirst defined in CSS levelEan element of type E1E:linkan E element that is the source anchor of a hyperlink whose target is either not yet visited (:visited) 1E:visitedE:activean E element during certain user
 actions1E:hover2E:focusE::first-linethe first formatted line of an E element1E::first-letterthe first formatted letter of an E element whose class is "warning" (the document language specifies how class is determined)1E#myidan E element with ID equal to
 "myid"1.c#myidthe element with class="c" and ID equal to "myid"1E Fan F element whose "foo" attribute value is exactly equal to "bar"2E[foo-="bar"]an E element whose "foo" attribute value is a list of whitespace-separated values, one of
 which is exactly equal to "bar"2E[foo|="en"]an E element whose "foo" attribute has a hyphen-separated list of values beginning (from the left) with "en"2E:first-childan E element, first child of its parent2E:lang(fr)an element of type E in language "fr" (the document language specifies how language is determined)2E::beforegenerated content before an
E element's content2E::aftergenerated content after an E element whose "foo" attribute value begins exactly with the string "bar"]an E element whose "foo" attribute value ends exactly with the
string "bar"3E[foo*="bar"]an E element, the n-th child of its parent3E:nth-of-type(n)an E element, the n-th child of its parent, counting from the last one3E:nth-of-type(n)an E element, the n-th sibling of its type3E:nth-last-one3E:nth-of-type(n)an E element, the n-th child of its parent.
of-type(n)an E element, the n-th sibling of its type3E:last-of-typean E element, last child of its parent3E:conly-of-typean E element, last sibling of its type3E:emptyan E element, only sibling of its type3E:emptyan E element, last sibling of its type3
that has no children (including text nodes) 3E:targetan E element E that is checked user interface element E that is checked user interface.
simple selector s3E ~ Fan F element preceded by an E element that contains an element matching simple selector s4A declaration itself consists of a property, a colon (:), and a value. Optional white-space may be
around the declaration block, declaration block, declarations, colons, and semi-colons for readability.[11]Properties can affect any type of element, and others apply only to particular groups of elements. [12][13]Values may be keywords, such as "center" or "inherit", or
numerical values, such as 200px (200 pixels), 50vw (50 percent of the viewport width) or 80% (80 percent of the parent element's width). Color values can be specified with keywords (e.g. #FF0000, also abbreviated as #F00), RGB values on a 0 to 255 scale (e.g. rgb(255, 0, 0)), RGBA values that specify both color and
alpha transparency (e.g. rgba(255, 0, 0, 0.8)), or HSL or HSLA values (e.g. hsl(0 100% 50%), hsl(0 100% 50%)
 (millimetre); pc (pica); and pt (point) are absolute, which means that the rendered dimension does not depend upon the structure of the page; others em (em); ex (ex) and px (pixel)[clarification needed] are relative, which means that factors such as the font size of a parent element can affect the rendered measurement. These eight units were a
 feature of CSS 1[15] and retained in all subsequent revisions. The proposed CSS Values and Units Module Level 3 will, if adopted as a W3C Recommendation, provide seven further length units: ch; Q; rem; vh; vmax; vmin; and vw.[16]Before CSS, nearly all presentational attributes of HTML documents were contained within the HTML markup. All
 font colors, background styles, element alignments, borders, and sizes had to be explicitly described, often repeatedly, within the HTML. CSS lets authors move much of that information to another file, the style sheet, resulting in considerably simpler HTML. And additionally, as more and more devices are able to access responsive web pages,
 different screen sizes and layouts begin to appear. Customizing a website for each device size is costly and increasingly difficult. The modular nature of CSS means that styles can be reused in different parts of a site or even across sites, promoting consistency and efficiency. For example, headings (h1 elements), sub-headings (h2), sub-sub-headings (h2), sub-sub-
(h3), etc., are defined structurally using HTML. In print and on the screen, choice of font, size, color and emphasis for these elements is presentational. Before CSS, document authors who wanted to assign such typographic characteristics to, say, all h2 headings had to repeat HTML presentational markup for each occurrence of that heading type. This
 made documents more complex, larger, and more error-prone and difficult to maintain. CSS allows the separation of presentation from structure. CSS can define color, font, text alignment, size, borders, spacing, layout and many other typographic characteristics, and can do so independently for on-screen and printed views. CSS also defines non
 presentational attributes: Chapter 1. The advantages of this may not be immediately clear but the power of CSS becomes more apparent when the style properties are placed in an internal style element: h1 { color: red; } All h1 elements in the document
 will then automatically become red without requiring any explicit code. If the author later wanted to make h1 elements blue instead, this could be done by changing the color for each individual h1 element. The styles can also be placed in an
external CSS file, as described below, and loaded using syntax similar to: This further decouples the styling from the HTML document and makes it possible to restyle multiple documents by simply editing a shared external CSS file. CSS, or Cascading Style Sheets, offers a flexible way to style web content, with styles originating from browser defaults
user preferences, or web designers. These styles can be applied inline, within an HTML document, or through external .css files for broader consistency. Not only does this simplify web development by promoting reusability and maintainability, it also improves site performance because styles can be offloaded into dedicated .css files that browsers
can cache. Additionally, even if the styles cannot be loaded or are disabled, this separation maintains the accessibility and readabilities. Its multi-faceted approach, including considerations for selector specificity, rule order, and media types, ensures that
 websites are visually coherent and adaptive across different devices and user needs, striking a balance between design intent and user accessibility. Multiple style sheets can be imported. Different styles can be applied depending on the output device being used; for example, the screen version can be quite different from the printed version, so
authors can tailor the presentation appropriately for each medium. The style sheet with the highest priority controls the content display. Declarations not set in the highest priority source are passed on to a source of lower priority source are passed on to a source of lower priority source are passed on to a source of lower priority source are passed on to a source of lower priority source are passed on to a source of lower priority source are passed on to a source of lower priority source are passed on to a source of lower priority source are passed on to a source of lower priority source are passed on to a source of lower priority source are passed on to a source of lower priority source are passed on to a source of lower priority source are passed on to a source of lower priority source are passed on to a source of lower priority source are passed on to a source of lower priority source are passed on to a source of lower priority source are passed on to a source of lower priority source are passed on to a source of lower priority source are passed on to a source of lower priority source are passed on to a source of lower priority source are passed on to a source of lower priority source are passed on to a source of lower priority source are passed on to a source of lower priority source are passed on to a source of lower priority source are passed on to a source of lower priority source are passed on to a source of lower priority source are passed on to a source of lower priority source are passed on to a source of lower priority source are passed on to a source of lower priority source are passed on to a source of lower priority source are passed on to a source of lower priority source are passed on to a source of lower priority source are passed on to a source of lower priority source are passed on to a source of lower priority source are passed on the lower priority source are passed on the lower priority source are passed on the lower passed on the lower passed on the lower passed on the lower passed o
over presentation. Someone who finds red italic headings difficult to read may apply a different style sheet. Depending on the browser and the website, a user may choose from various style sheets provided by the designers, or may remove all added styles, and view the site using the browser and the website, a user may override just the red italic headings
style without altering other attributes. Browser extensions like Stylish and Stylus have been created to facilitate the management of such user style sheets. In the case of large projects, cascading can be used to determine which style has a higher priority when developers do integrate third-party styles that have conflicting priorities, and to further
resolve those conflicts. Additionally, cascading can help create themed designs, which help designers fine-tune aspects of a design without compromising the overall layout.CSS priority scheme (highest to lowest)PriorityCSS source typeDescription1Important annotation overwrites the previous priority types2InlineA style applied to
an HTML element via HTML "style" attribute3Media TypeA property definition applies to all media types unless a media-specific CSS is defined4User definition applies to all media types unless a media-specific CSS is defined4User definition applies to all media types unless a media-specific CSS is defined4User definition applies to all media types unless a media-specific CSS is defined4User definition applies to all media types unless a media-specific CSS is defined4User definition applies to all media types unless a media-specific CSS is defined4User definition applies to all media types unless a media-specific CSS is defined4User defined4User definition applies to all media types unless a media-specific CSS is defined4User defined4User definition applies to all media types unless a media-specific CSS is defined4User defined4User definition applies to all media-specific CSS is defined4User 
a higher priority? Parent inheritance If a property is not specified, it is inherited from a parent element8CSS property definition in HTML documentCSS rule or CSS inline style overwrites a default browser default browser default trouble is determined by W3C initial value specifications Specifications are the relative
 weights of various rules. [18] It determines which styles apply to an element when more than one rule could apply. Based on the specificity of 1,0,0. Because the specificity of
 commas are used to separate the "digits"[19] (a CSS rule having 11 elements and 11 classes would have a specificity of 11,11, not 121). Thus the selectors Specificity of 11,11, not 121). Thus the selectors of the following rule result in the indicated specificity: Selectors Specificity of 11,11, not 121). Thus the selectors of the following rule result in the indicated specificity of 11,11, not 121). Thus the selectors of the following rule result in the indicated specificity of 11,11, not 121). Thus the selectors of the following rule result in the indicated specificity of 11,11, not 121). Thus the selectors of the following rule result in the indicated specificity of 11,11, not 121). Thus the selectors of the following rule result in the indicated specificity of 11,11, not 121). Thus the selectors of the following rule result in the indicated specificity of 11,11, not 121). Thus the selectors of the following rule result in the indicated specificity of 11,11, not 121).
1p.bright em.dark {color: yellow;}0, 0, 2, 2#id218 {color: brown;}0, 1, 0, 0style="11, 0, 0, 0Consider this HTML fragment: #xyz { color: blue;} To demonstrate specificity, and thus, the paragraph appears green: To
demonstrate specificityInheritance is a key feature in CSS; it relies on the ancestor-descendant relationship to operate. Inheritance relies on the document tree, which is the hierarchy of XHTML elements in a page based on
 nesting. Descendant elements may inherit CSS property values from any ancestor element enclosing them. In general, descendant elements inherited are color, font, letter spacing, line-height, list-style, text-align, text-indent, text-transform,
visibility, white-space, and word-spacing. Properties that cannot be inherited are background, border, display, float and clear, height, and width, margin, min- and max-height and -width, outline, overflow, padding, position, text-decoration, vertical-align, and z-index. Inheritance can be used to avoid declaring certain properties over and over again in a
style sheet, allowing for shorter CSS. Inheritance in CSS is not the same as inheritance in class-based programming languages, where it is possible to define a CSS class B like that,
which could then be used to style multiple elements without having to repeat the modifications. Given the following style sheet:p { color: pink;} Suppose there is a p element with an emphasized word "illustrate" inherits the color of the parent
element, p. The style sheet p has the color pink, hence, the em element is likewise pink: This is to illustrate inheritanceThe whitespace between properties and selectors is ignored. This code snippet:body {overflow:hidden;background-position:left top;} is
 functionally equivalent to this one:body { overflow: hidden; background-color: #000000; background-image: url(images/bg.gif); background-repeat: no-repeat; background-position: left top;} Main article: Indentation styleOne common way to formatting CSS for readability is to indent each property and give it its own line. In addition to formatting CSS for
 readability, shorthand properties can be used to write out the code faster, which also gets processed more quickly when being rendered:[21]body { overflow: hidden; background: #000 url(images/bg.gif) no-repeat left top;} Sometimes, multiple property values are indented onto their own line:@font-face { font-family: 'Comic Sans'; font-size: 20px; src
url('first.example.com'), url('second.example.com'), url('second.example.com'), url('second.example.com'), url('fourth.example.com'), url('fourth.example.com'), url('fourth.example.com'), url('second.example.com'), url('second.example.co
 items stack vertically, like paragraphs and like the items in a bulleted list. Normal flow also includes the relative positioning of block or inline items and run-in boxes. Floats floated item is taken out of the normal flow and shifted to the left or right as far as possible in the space available. Other content then flows alongside the floated item. Absolute
positioningAn absolutely positioned item has no place in, and no effect on, the normal flow of other items. It occupies its assigned position in its container independently of other items. [22] There are five position in its container independently of other items.
are used to specify offsets and positions. The element having position static is not affected by the top, bottom, left or right properties. The default value places the item in the normal flow. The item is placed in the normal flow, and then shifted or offset from that position. Subsequent flow items are laid out as if the item had not been moved. Specifies
absolute positioning. The element is positioned in relation to its nearest non-static ancestor. The item is absolutely positioned in fixed positioned in a fixed positioned in a fixed position on the screen even as the rest of the document is scrolled [22]. The float property may have one of three values. Absolutely positioned in relation to its nearest non-static ancestor. The item is absolutely positioned in relation to its nearest non-static ancestor.
floated items, unless they are prevented from doing so by their clear property.leftThe item floats to the line that it would have appeared in; other items may flow around its left side.clearForces the element to appear
underneath ('clear') floated elements to the left (clear:left), right (clear:left), right (clear:right) or both sides (clear:both).[22][23]Hkon Wium Lie on 10 October 1994.[24] At the time, Lie was working with Tim Berners-Lee at
CERN.[25] Several other style sheet languages for the web were proposed around the same time, and discussions on public mailing lists and inside World Wide Web Consortium resulted in the first W3C CSS Recommendation (CSS1)[26] being released in 1996. In particular, a proposal by Bert Bos was influential; he became co-author of CSS1, and is
regarded as co-creator of CSS.[27]Style sheets have existed in one form or another since the beginnings of Standard Generalized Markup Language (SGML) in the 1980s, and CSS was developed to provide style sheets for the web.[28] One requirement for a web style sheet language was for style sheets to come from different sources on the web.
Therefore, existing style sheet languages like DSSSL and FOSI were not suitable. CSS, on the other hand, let a document's style be influenced by multiple style sheets by way of "cascading" styles.[28]As HTML grew, it came to encompass a wider variety of stylistic capabilities to meet the demands of web developers. This evolution gave the designer
 more control over site appearance, at the cost of more complex HTML. Variations in web browser implementations, such as ViolaWWW and WorldWideWeb,[29] made consistent site appearance difficult, and users had less control over how web content was displayed. The browser/editor developed by Tim Berners-Lee had style sheets that were hard-
coded into the program. The style sheets could therefore not be linked to documents on the web.[25] Robert Cailliau, also of CERN, wanted to separate the structure from the presentation so that different style sheets could describe different style sheets could describe different presentation for printing, screen-based presentation so that different style sheets could describe different presentation for printing, screen-based presentation for printing, screen-based presentation for printing web presentation for 
 was a topic of interest to many in the web community and nine different style sheet languages were proposed on the www-style mailing list.[28] Of these nine proposals, two were especially influential on what became CSS: Cascading HTML Style Sheet Proposal (SSP).[27][30] Two browsers served as testbeds for
the initial proposals; Lie worked with Yves Lafon to implement CSS in Dave Raggett's Arena browser.[27] Thereafter, Lie and Bos worked together to develop the CSS standard (the 'H' was removed from the name because these style sheets could also be applied to other
markup languages besides HTML).[25]Lie's proposal was presented at the "Mosaic and the Web" conference (later called WWW2) in Chicago, Illinois in 1994, and again with Bert Bos in 1995.[25] Around this time the W3C was already being established and took an interest in the development of CSS. It organized a workshop toward that end chaired
by Steven Pemberton. This resulted in W3C adding work on CSS to the deliverables of the HTML editorial review board (ERB). Lie and Bos were the primary technical staff on this aspect of the project, with additional members, including Thomas Reardon of Microsoft, participating as well. In August 1996, Netscape Communication Corporation
 presented an alternative style sheet language called JavaScript Style Sheets (JSSS).[25] The spec was never finished, and is deprecated.[34] By the end of 1996, CSS was ready to become official, and the CSS level 1 Recommendation was published in December.Development of HTML, CSS, and the DOM had all been taking place in one group, the
 HTML Editorial Review Board (ERB). Early in 1997, the ERB was split into three working groups: HTML Working Group, chaired by Dan Connolly of W3C; DOM Working Group began tackling issues that had not been addressed
 with CSS level 1, resulting in the creation of CSS level 2 on November 4, 1997. It was published as a W3C Recommendation on May 12, 1998. CSS level 3, which was started in 1998, is still under development as of 2014[update]. In 2005, the CSS Working Groups decided to enforce the requirements for standards more strictly. This meant that already
published standards like CSS2.1, CSS3 Selectors, and CSS3 Text were pulled back from Candidate Recommendation to Working Draft level. The CSS1 specification was completed in 1996. Microsoft's Internet Explorer 3[25] was released that year, featuring some limited support for CSS. IE 4 and Netscape 4.x added more support, but it was typically
 incomplete and had many bugs that prevented CSS from being usefully adopted. It was more than three years before any web browser achieved near-full implementation of the specification. Internet Explorer 5.0 for the Macintosh, shipped in March 2000, was the first browser to have full (better than 99 percent) CSS1 support,[35] surpassing Opera,
 which had been the leader since its introduction of CSS support fifteen months earlier. Other browsers followed soon afterward, and many of them additionally implementation of CSS, they were still incorrect in certain areas. They were fraught
 with inconsistencies, bugs, and other quirks. Microsoft Internet Explorer 5. x for Windows, as opposed to the very different IE for Macintosh, had a flawed implementation in feature support made it difficult for designers to achieve a consistent appearance
 across browsers and platforms without the use of workarounds termed CSS hacks and filters. The IE Windows box model bugs were so serious that, when Internet Explorer 6 was released, Microsoft introduced a backward-compatible mode of CSS interpretation ("quirks mode") alongside an alternative, corrected "standards mode". Other non-
 Microsoft browsers also provided mode-switch capabilities. It, therefore, became necessary for authors of HTML files to ensure they contained SS to be interpreted correctly, in compliance with standards, as opposed to being intended for the now long
obsolete IE5/Windows browser. Without this marker, web browsers with the "quirks mode"-switching capability will size objects in web pages as IE 5 on Windows would, rather than following CSS standards. Problems with the patchy adoption of CSS and errata in the original specification led the W3C to revise the CSS2 standards into CSS2.1, which
moved nearer to a working snapshot of current CSS support in HTML browsers. Some CSS2 properties that no browser successfully implemented were dropped, and in a few cases, defined behaviors were changed to bring the standard into line with the predominant existing implementations. CSS2.1 became a Candidate Recommendation on
February 25, 2004, but CSS2.1 was pulled back to Working Draft status on July 19, 2007.[37]In addition to these problems, the .css extension was used by a software product used to convert PowerPoint files into Compact Slide Show files,[38]so some web servers served all
 .css[39] as MIME type application/x-pointplus[40] rather than text/css.Individual browser vendors occasionally introduced new parameters ahead of standardization and universalization. To prevent interfering with future implementations, vendors prepended unique names to the parameters, such as -moz- for Mozilla Firefox, -webkit- named after the
 browsing engine of Apple Safari, -o- for Opera Browser and -ms- for Microsoft Internet Explorer and early versions of Microsoft Edge that use EdgeHTML.Occasionally, the parameters with vendor prefixes such as -moz-radial-gradient and -webkit-linear-gradient have slightly different syntax as compared to their non-vendor-prefix counterparts.
[41]Prefixed properties are rendered obsolete by the time of standardization. Programs are available to automatically add prefixes for older browsers, removing the prefix allows other browsers to see the functionality. An exception is
certain obsolete -webkit- prefixed properties, which are so common and persistent on the web that other families of browsers have decided to support them for compatibility.[42]CSS Snapshot 2021CSS has various levels and profiles. Each level of CSS builds upon the last, typically adding new features and typically denoted[43] as CSS1, CSS2, CSS3,
and CSS4. Profiles are typically a subset of one or more levels of CSS built for a particular device or user interface. Currently, there are profiles should not be confused with media types, which were added in CSS2. The first CSS specification to become an official W3C Recommendation is CSS
level 1, published on 17 December 1996. Hkon Wium Lie and Bert Bos are credited as the original developers. [44][45] Among its capabilities are support for Font properties such as typeface and emphasis Color of text, backgrounds, and other elements Text attributes such as typeface and emphasis Color of text, backgrounds, and other elements Text attributes such as typeface and emphasis Color of text, backgrounds, and other elements Text attributes such as typeface and emphasis Color of text, backgrounds, and other elements Text attributes such as typeface and emphasis Color of text, backgrounds, and other elements Text attributes such as typeface and emphasis Color of text, backgrounds, and other elements Text attributes such as typeface and emphasis Color of text, backgrounds, and other elements Text attributes such as typeface and emphasis Color of text, backgrounds, and other elements Text attributes such as typeface and emphasis Color of text, backgrounds, and other elements Text attributes such as typeface and emphasis Color of text, backgrounds, and other elements Text attributes such as typeface and emphasis Color of text, backgrounds, and other elements Text attributes are typeface and emphasis Color of text, backgrounds, and other elements Text attributes are typeface and emphasis Color of text, backgrounds, and the text attributes are typeface and emphasis Color of text attributes are typeface attributes attributes are typeface and emphasis Color of text attributes are typeface attributes.
 tables and other elementsMargin, border, padding, and positioning for most elementsUnique identification and generic classification of groups of attributesThe W3C no longer maintains the CSS 1 Recommendation.[46]CSS level 2 specification was developed by the W3C and published as a recommendation in May 1998. A superset of CSS1, CSS2
 includes a number of new capabilities like absolute, relative, and fixed positioning of elements and z-index, the concept of media types, support for aural style sheets (which were later replaced by the CSS 3 speech modules)[47] and bidirectional text, and new font properties such as shadows. The W3C no longer maintains the CSS2 recommendation
[48]CSS level 2 revision 1, often referred to as "CSS 2.1", fixes errors in CSS2, removes poorly supported or not fully interoperable features and adds already implemented browser extensions to the specification. To comply with the W3C Process for standardizing technical specifications, CSS2.1 went back and forth between Working Draft status and
Candidate Recommendation status for many years. CSS 2.1 first became a Candidate Recommendation on 19 July 2007 and then updated twice in 2009. However, because changes and clarifications were made, it
 again went back to Last Call Working Draft on 7 December 2010.CSS 2.1 went to Proposed Recommendation on 12 April 2011.[49] After being reviewed by the W3C Advisory Committee, it was finally published as a W3C Recommendation on 7 June 2011.[50] CSS 2.1 was planned as the first and final revision of level 2but low-priority work on CSS 2.2
 began in 2015. "CSS3" redirects here. For other uses, see CSS3 (disambiguation). Unlike CSS2, which is a large single specification defining various features defined in CSS2, preserving backward compatibility. Work on CSS leve
3 started around the time of publication of the original CSS2 recommendation. The earliest CSS3 drafts were published in June 1999.[51]Due to the modules have Candidate Recommendation (CR) status and are considered moderately stable. At CR stage
 implementations are advised to drop vendor prefixes.[53]Summary of main module-specifications[54]ModuleSpecification titleStatusDatecss3-backgroundCSS Box Model Module Level 3RecommendationApr 2023css-cascade-3CSS Cascading and Inheritance Level
3RecommendationFeb 2021css-color-3CSS Color Module Level 3RecommendationJan 2022css3-content Module Level 3RecommendationSep 2018css3-gcpmCSS Generated Content for Paged Media ModuleWorking DraftMay 2014css3-layoutCSS Template Layout
 ModuleNoteMar 2015css3-mediaqueriesMedia QueriesRecommendationJun 2012mediaqueries-4Media Queries Level 4Candidate Rec. Oct 2021css3-multicolMulti-column Layout Module Level 3Working Draft, and part migrated to css3-breakOct 2018css3-breakCSS Fragmentation
Module Level 3Candidate Rec.Dec 2018selectors-3Selectors Level 3RecommendationNov 2018selectors-4Selectors Level 3Recommendation Jun 2018" redirects here. For other uses, see CSS4 (disambiguation). Jen Simmons discussing the state of CSS in 2019, as
several CSS4 modules were being advancedThere is no single, integrated CSS4 specification, [55] because the specification has been split into many separate modules which level 4 or are already approaching Level 5. Other
modules that define entirely new functionality, such as Flexbox,[56] have been designated as Level 1 and some of them are approaching Level 2. The CSS Working Group sometimes publishes "Snapshots", a collection of whole modules and parts of other drafts that are considered stable enough to be implemented by browser developers. So far, five
such "best current practices" documents have been published as Notes, in 2007,[57] 2010,[58] 2017,[60] and 2018.[61]Since these specification snapshots are primarily intended for developers, there has been a growing demand for a similar versioned reference document targeted at authors, which would present the state of interoperable
implementations as meanwhile documented by sites like Can I Use...[62] and the MDN Web Docs.[63] A W3C Community Group has been established in early 2020 in order to discuss and define such a resource.[64] The actual kind of versioning is also up to debate, which means that the document, once produced, might not be called "CSS4". Each web
 browser uses a layout engine to render web pages, and support for CSS functionality is not consistent between them. Because browsers do not parse CSS perfectly, multiple coding techniques have been developed to target specific browsers do not parse CSS perfectly, multiple coding techniques have been developed to target specific browsers with workarounds (commonly known as CSS hacks or CSS filters). The adoption of new functionality in CSS
 can be hindered by a lack of support in major browsers. For example, Internet Explorer was slow to add support for many CSS 3 features, which slowed the adoption of those features and damaged the browser's reputation among developers. Additionally, a proprietary syntax for the non-vendor-prefixed filter property was used in some versions. [65]
 In order to ensure a consistent experience for their users, web developers often test their sites across multiple operating systems, browsers, and browser versions, increasing development time and complexity. Tools such as BrowserStack have been built to reduce the complexity of maintaining these environments. In addition to these testing tools,
 many sites maintain lists of browser support for specific CSS properties, including CanIUse and the MDN Web Docs. Additionally, CSS 3 defines feature queries, which provide an @supports directive that will allow developers to target browsers with support for certain functionality directly within their CSS.[66] CSS that is not supported by older
browsers can also sometimes be patched in using JavaScript code designed to make browsers behave consistently. These workaroundsand the need to support fallback functionality and complexity to development projects, and consequently, companies frequently define a list of browser versions that they will
and will not support. As websites adopt newer code standards that are incompatible with older browsers can be cut off from accessing many of the most popular sites on the internet are not just visually degraded on older browsers due to poor CSS support but do not
 work at all, in large part due to the evolution of JavaScript and other web technologies. Some noted limitations of the current capabilities of CSS include: Scoping rules for properties such as z-index look for the closest parent element with a position: relative attribute. This odd coupling has undesired effects. For example, it is
impossible to avoid declaring a new scope when one is forced to adjust an element's position, preventing one from using the desired scope of a parent element. SS implements pseudo-class, ":hover", is dynamic (equivalent of JavaScript
 "onmouseover") and has potential for misuse (e.g., implementing cursor-proximity popups),[68] but CSS has no ability for a client to disable "-like property) or limit its effects (no "nochange"-like values for each property) or limit its effects (no "nochange"-like values for each property).
 selector changes.CSS styles often must be duplicated in several rules to achieve the desired effect, causing additional maintenance and requiring more thorough testing. Some new CSS features were proposed to solve this but were abandoned afterward.[69][70] Instead, authors may gain this ability by using more sophisticated stylesheet languages.
which compile to CSS, such as Sass, Less, or Stylus. Besides the ::first-letter pseudo-element, one cannot target specific ranges of text without needing to utilize placeholder elements. Main article: Separation of content and presentation of content and presentation of content and presentation formats by adjusting styles based on various
nominal parameters. These parameters include explicit user preferences (such as themes or font size), compatibility with different web browsers, the type of device used to view the content (e.g., desktop, tablet, or mobile device), screen resolutions, the geographic location of the user, and many other variables. CSS also enables responsive design
ensuring that content dynamically adapts to different screen sizes and orientations, enhancing accessibility and user experience across a wide range of environments. Main article: Style sheet can be used to affect and style elements site-wide
If the situation arises that the styling of the elements should be changed or adjusted, these changes can be made by editing rules in the global style sheet. Before CSS, this sort of maintenance was more difficult, expensive, and time-consuming. A stylesheet, internal or external, specifies the style once for a range of HTML elements selected by class,
type or relationship to others. This is much more efficient than repeating style information inline for each occurrence of the element. An external stylesheet is usually stored in the browser cache, and can therefore be used on multiple pages without being reloaded, further reducing data transfer over a network. Main article: Progressive
enhancementWith a simple change of one line, a different style sheet can be used for the same page. This has advantages for accessibility, as well as providing the ability to tailor a page or site to different target devices. Furthermore, devices not able to understand the styling still display the content. Main article: Tableless web design
AccessibilityWithout CSS, web designers must typically lay out their pages with techniques such as HTML tables that hinder accessibility for vision-impaired users (see Tableless web design Accessibility). Main article: CSS frameworkCSS frameworkCSS frameworks are prepared libraries that are meant to allow for easier, more standards-compliant styling of web
pages using the Cascading Style Sheets language. CSS frameworks include Blueprint, Bootstrap, Foundation and Materialize. Like programming and scripting language libraries, CSS frameworks are usually incorporated as external .css sheets referenced in the HTML. They provide a number of ready-made options for designing and laying out the
 web page. Although many of these frameworks have been published, some authors use them mostly for rapid prototyping, or for learning from, and prefer to 'handcraft' CSS that is appropriate to each published site without the design, maintenance and download overhead of having many unused features in the site's styling.[71]As the size of CSS
resources used in a project increases, a development, ease of development, and performance of the deployed stylesheets in the browser. Popular methodologies include OOCSS (object-oriented CSS), ACSS
(atomic CSS), CSS (organic Cascade Style Sheet), SMACSS (scalable and modular architecture for CSS), and BEM (block, element, modifier).[72]Flash of unstyled contentCSS-in-JS^ "Minutes Telecon 2024-12-11". CSS WG Blog. W3C. 2024-12-12. Archived from the original on 2025-01-16. Retrieved 2025-01-16.^ "CSS developer guide". MDN Web
Docs. Archived from the original on 2015-09-25. Retrieved 2015-09-24. Flanagan, David (18 April 2011). JavaScript is part of the triad of technologies that all Web developers must learn: HTML to specify the content of web pages, CSS to
specify the presentation of web pages, and JavaScript to specify the behavior of web pages. "What is CSS?". World Wide Web Consortium. Archived from the original on 2010-11-29. Retrieved 2010-12-01. "Web-based Mobile Apps of the Future Using HTML 5, CSS and JavaScript". HTML Goodies. HTMLGoodies.
Archived from the original on 2014-10-20. Retrieved 2014-10-21. Retrieved 2011-02-14. Retrieved 2012-06-30.^ "W3C CSS validation service". Archived from the original on 2011-02-14. Retrieved 2012-06-30.^ "W3C CSS validation service". Archived from the original on 2011-02-14. Retrieved 2012-06-30.^ "W3C CSS validation service". Archived from the original on 2011-02-14. Retrieved 2012-06-30.^ "W3C CSS validation service". Archived from the original on 2011-02-14. Retrieved 2012-06-30.^ "W3C CSS validation service". Archived from the original on 2011-02-14. Retrieved 2012-06-30.^ "W3C CSS validation service". Archived from the original on 2011-02-14. Retrieved 2012-06-30.^ "W3C CSS validation service".
 "Selectors". Cascading Style Sheets Level 2 Revision 1 (CSS 2.1) Specification. W3C. Archived from the original on 2014-06-03. Retrieved 2014-05-30.^ "CSS Syntax Module Level 3". W3C. Archived from the original on 1 October 2023. Retrieved 1 October 2023.^ "W3C CSS2.1
 specification for rule sets, declaration blocks, and selectors". World Wide Web Consortium. 7 June 2011. Archived from the original on 28 March 2008. Retrieved 2014-05-30. "Full property table". W3C. Retrieved 2020-08-09. "CSS Color". MDN Web
Docs. 2024-04-05. Archived from the original on 2024-03-27. Retrieved 2024-04-05. *S. Distance Units: the type". CSS Values and Units Module Level 3. 6 June 2019. Archived from the original on 7 June 2019. *S. Distance Units: the type". CSS Values and Units Module Level 3. 6 June 2019. Archived from the original on 7 June 2019. *S. Distance Units: the type". CSS Values and Units Module Level 3. 6 June 2019. Archived from the original on 7 June 2019. *S. Distance Units: the type". CSS Values and Units Module Level 3. 6 June 2019. Archived from the original on 7 June 2019. *S. Distance Units: the type". CSS Values and Units Module Level 3. 6 June 2019. *S. Distance Units: the type". CSS Values and Units Module Level 3. 6 June 2019. *S. Distance Units: the type". CSS Values and Units Module Level 3. 6 June 2019. *S. Distance Units: the type". CSS Values and Units Module Level 3. 6 June 2019. *S. Distance Units: the type". CSS Values and Units Module Level 3. 6 June 2019. *S. Distance Units: the type". CSS Values and Units Module Level 3. 6 June 2019. *S. Distance Units: the type". CSS Values and Units Module Level 3. 6 June 2019. *S. Distance Units: the type". CSS Values and Units Module Level 3. 6 June 2019. *S. Distance Units: the type". CSS Values and Units Module Level 3. 6 June 2019. *S. Distance Units: the type". *S. Distance Units: the type "S. Distance Units" and Units Module Level 3. 6 June 2019. *S. Distance Units Module Level 3. 6 June 2019. *S. Distance Units Module Level 3. 6 June 2019. *S. Distance Units Module Level 3. 6 June 2019. *S. Distance Units Module Level 3. 6 June 2019. *S. Distance Units Module Level 3. 6 June 2019. *S. Distance Units Module Level 3. 6 June 2019. *S. Distance Units Module Level 3. 6 June 2019. *S. Distance Units Module Level 3. 6 June 2019. *S. Distance Units Module Level 3. 6 June 2019. *S. Distance Units Module Level 3. 6 June 2019. *S. Distance Units Module Level 3. 6 June 2019. *S. Distance Units Module Level 3. 6 June 2019. *S. Distance Units Module Level 3. 6 June 20
0. Archived from the original on 2014-02-15. Retrieved 2014-02-16. Retrieved 2014-06-11. Retrieved 2014-06-10. Tean a CSS class inherit one or more other classes?". StackOverflow. Archived from the original on 2017-10-14. Retrieved 2017-09-10. "Shorthand
properties". Tutorial. Mozilla Developers. 2017-12-07. Archived from the original on 2018-01-30. Retrieved 2018-01-30. Retrieved 2018-01-30. Retrieved 2018-01-30. Retrieved 2018-01-30. Retrieved 2018-01-30. Retrieved 2018-01-30.
 Holzschlag, Molly E (2005). Spring into HTML and CSS. Pearson Education, Inc. ISBN0-13-185586-7. a b Lie, Hakon W (10 October 1994). "Cascading HTML style sheets a proposal" (Proposal). CERN. Archived from the original on 4 June 2014. Retrieved 25 May 2014. a b c d e f Lie, Hkon Wium; Bos, Bert (1999). Cascading Style Sheets, designing
for the Web. Addison Wesley. ISBN0-201-59625-3. Retrieved 23 June 2010. "Cascading Style Sheets, level 1". World Wide Web Consortium. Archived from the original on 2014-04-09. Retrieved 23 June 2010. "Simple style Sheets for SGML & HTML on the web". World Wide Web Consortium. Archived from the original
on 23 September 2009. Retrieved 20 June 2010. a b c "Cascading Style Sheets". University of Oslo. Archived from the original on 2006-09-06. Retrieved 3 September 2014. a b Petrie, Charles; Cailliau, Robert (November 1997). "Interview Robert Cailliau on the WWW Proposal: "How It Really Happened."". Institute of Electrical and Electronics
Engineers. Archived from the original on 6 January 2011. Retrieved 18 August 2010. Bos, Bert (31 March 1995). "Stream-based Style sheet Proposal". Archived from the original on 12 October 2014. Retrieved 3 September 2014. Retrieved 3 September 2014. Retrieved 3 September 2014. The original on 12 October 2014. Retrieved 3 September 2014.
2 December 2009. Retrieved 6 June 2010. Tyves Lafon. World Wide Web Consortium. Archived from the original on 24 June 2010. Retrieved 17 June 2010. Retrieved 17 June 2010. Retrieved 22 January 2011. Lou Montulli; Brendan Eich;
Scott Furman; Donna Converse; Troy Chevalier (22 August 1996). "JavaScript-Based Style Sheets". W3C. Archived from the original on 2010-11-25. Retrieved 2011-01-15. Anne van Kesteren. "CSS 2.1 Anne's Weblog". Archived from the original on 2005-12-11-125. Retrieved 2011-01-15.
10. Retrieved 2011-02-16.^ "Archive of W3C News in 2007". World Wide Web Consortium. 21 December 2007. Archived from the original on 2011-05-20. Retrieved 20 June
2010. McBride, Don (27 November 2009). "File Types". Archived from the original on 29 October 2010. Retrieved 20 June 20 June 20 June 20 June 20 June 
Lifewire. Archived from the original on Nov 30, 2024. "CSS Snapshot 2023 2.4. CSS Levels". WHATWG. 24 January 2024. Archived from the original on Feb 8, 2024. "CSS Snapshot 2023 2.4. CSS Levels". WHATWG. 24 January 2024. Archived from the original on Feb 8, 2024. "Bos, Bert; Wium Lie, Hkon (1997). Cascading style sheets: designing for the Web (1st
print.ed.). Harlow, England; Reading, MA.: Addison Wesley Longman. ISBN0-201-41998-X. W3C: Cascading Style Sheets, level 1 specification Archived 2011-02-11 at the Wayback Machine CSS level 1 specification W3C: Cascading Style Sheets". W3C.
Archived from the original on 2014-10-26. Retrieved 2014-10-26. Retrieved 2014-10-26. Retrieved 2011-11-09 at the Wayback Machine CSS 2 specification (W3C Proposed Recommendation) W3C:Cascading Style Sheets, level 2 revision 1 Archived 2011-11-09 at the Wayback Machine CSS 2.1 specification (W3C Proposed Recommendation)
W3C: Cascading Style Sheets Standard Boasts Unprecedented Interoperability Archived 2011. Bescriptions of all CSS specifications. World Wide Web Consortium. Archived from the original on 31 March 2011. Retrieved 3 March 2011. Bos, Bert (26 February 2011). "CSS current
work". World Wide Web Consortium. Archived from the original on 3 March 2011. Retrieved 3 March 2011. 
Archived from the original on 2014-05-30. Retrieved 2014-05-30. Retrieved 2014-05-30. Retrieved 18 October 2012. Retrieved 18 Oct
Sheets (CSS) Snapshot 2007". 12 May 2011. Archived from the original on 8 August 2016. Retrieved 18 July 2016. Retrieved 3 March 2011. Retrieved 3 March 2011. Archived from the original on 27 January 2017.
Retrieved 13 February 2017. "CSS Snapshot 2017". W3C. 31 January 2017. Archived from the original on 13 February 2017. Retrieved 2 January 2017. Archived from the original on 13 February 2017. Archived from the original on 13 February 2017. Test Snapshot 2017. Archived from the original on 13 February 2017. Archived from the original on 14 February 2017. Test Snapshot 2017. Test Snap
the original on 2018-02-19. Retrieved 2019-01-26. Test are represented 2019-01-26. Test are represented 2020-02-27. Lazaris, Louis (2010-04-28). "CSS" MDN Web Docs. 21 July 2023. Archived from the original on Feb 10, 2023. Retrieved 2020-02-27. Lazaris, Louis (2010-04-28). "CSS" Solutions for Internet Explorer".
Smashing Magazine. Archived from the original on 2016-10-12. Retrieved 2016-10-12. Retri
the original on 2016-10-12. Retrieved 2016-10-12. Retrieved 2016-10-12. Retrieved 2016-10-12. Retrieved 2016-02-27. "Pure CSS Popups". meyerweb.com. Archived from the original on 2016-02-22. Retrieved 2016-10-12. Retriev
Handcrafted CSS: More Bulletproof Web Design. New Riders. p.114. ISBN 978-0-321-64338-4. Archived from the original on 2 June 2015. Retrieved 2 June 2015. Meyer, Eric A.;
Weyl, Estelle (2023). Cascading Style Sheets: The Definitive Guide, Fifth Edition. O'Reilly Media, Inc. ISBN 978-1-61729-345-0.MDN CSS reference MDN Getting Started with CSSCSS at Wikipedia's sister projects Definitions from Wiktionary Media from
CommonsTextbooks from WikibooksResources from Wikibook
In web design circles, you hear the term accessibility thrown around quite a bit. I think this is a great thing. When we inspect our sites to see if they meet accessibility criteria, however, we tend to ignore text alignment. Important note: This information does not apply only to websites. Those who create emails, newsletters, flyers, or anything with
paragraphs of information should also take heed. I realize that most people dont give text alignment a second thought, especially on the web, but it makes a difference. As a former educator, I know that the way text appears and other learning disabilities. When I had a job training teachers in
technology, I had training in creating audience-friendly presentation slides. In that class, I learned that text alignment makes a difference. When paragraphs and other long bits of information appears and other long bits of information app
brain work harder to process what its reading. We learn to read left to right and encounter text laid out that way from an early age. When each line of text starts in a consistent spot, our brains dont have to work as hard to process the information. When writers completely center entire paragraphs, however, our eyes have to search for the beginning of
each line, causing the brain to process the information in a more disjointed way. Our brains finds thisunnatural and defiant of howwe are taught as children. To see what I mean, check out this article on Why You Should Never Center Align Paragraph Text. Theres an example of centered text versus left aligned text and which is easier to read. When
considering the text alignment for your web page content (or for email newsletters, flyers, and anything else with paragraphs of information) reserve centered text for certain situations: Headers Centering a topic header or headline will set it apart from the rest of the content. Other options for headers include different font sizes and weights.
Emphasis To emphasize a point, date, deadline, etc., go ahead and center it. As with headers, it setsthe information apart and draws the readers eye. If everything shows centered, nothing stands apartand it all runs together. Quotes If you have a short (about one sentence) quote that doesn't take up too many lines, centering it would work, especially if
you dont overuse it. Having five centered quotes on a page, for example, may go a little to far. Construct longer quotes in paragraph (like these bullet points) so the brain still processes the information easily. To make your website easier for the visually
impaired or for those with reading disabilities, text alignment makes difference. So take a look at your blog, website, or email newsletter and adjust the text alignment. In the meantime, keep swimming along! Click To TweetI see it time and time again with UI and web design people love centering, justifying, and right aligning text. People think this
```

may make their text look pretty or my favourite forbidden word in web design, clean. But in reality, this is another example of people sacrificing user experiencing (UX) for user interface design (UI). Its time to put a stop to this! Does text alignment matter for accessibility? Lets ju	
terms of specifically web design, these types of alignment are used in different ways.Left alignment is by the most popular text alignment is commonly associated with lengthy paragraph text, which enables it to be as easy as possible to reaw with these common 3 or 4 column layouts on homepages. I also often see it at the bottom of pages as call to actions, giving less lengthy content better symmetry. Right alignment is rarely used, if ever. But if it is used, its to align text up to another element for more visual flare. And	
uxurious design styles to give it some visual taste and elegance. Left is the most popular and default text alignment. Its the best for readability and user experience because of the way our eyes read. Right alignment is probably the most uncommon alignment used, and when it is used.	used its typically used in small quantities. A great example of right
alignment being used effectively is in the website navigation. With right alignment, you can include a CTA button In the top right and have the main links align with it nicely making the navigation very scannable. Right-aligned navigation Right alignment should be used sparingly at International Properties of the sense of the se	
right, the right-aligned text on has a maximum of 3 words per line which limits the negative user experience to be insignificant at worst making it a good use case. I should mention that right-aligned text is different than right-aligned elements. Right-aligned elements work well in	most cases, and can actually help make use of otherwise unused space.
In the letterhead picture, its okay that the content area is right-aligned, its the right text-align in that content area where my critiques lay. Center alignment can look great in small doses, but it can lead to problems when people overuse it. The reason why center text alignment is here is a brief moment where the user has to find where the next line begins decreasing the users reading? Hell yes! This is what I think goes through peoples heads when they decide to center a large paragraph. What	
experience. However there are some limits and exceptions of center alignment. Why oh why are we doing this. Oh yes, because its pretty. Centered text has become such a common occurrence, especially in website design. In my opinion centered paragraphs are only acceptable up displeasing to read each line after. Here are some examples: This is an acceptable length of text in a paragraph to center align. No more than three lines of text. This is a much longer paragraph with a length that is displeasing to read because of the number of times you have to find	to a point, 3 lines of text to be specific. Anymore, it becomes too
imited and used sparingly so it doesnt make the user annoyed to read your text on this blog is left aligned? That called user experience, and dont you forget it!You can just see from the second example the paragraph is flat-out annoying to read. Th	
be used 95% of the time. Primary page titles should be okay centered as they tend to not have as many words and therefore lines of text. Most page titles arent long enough that text alignment becomes an issue with usability. However, with secondary titles (h2s) and anything und post to google title alignment is this blog post. At the title section at the top, I use left alignment because some of the titles of some of my articles may get long enough that centering the text would become annoying to read and so I use left alignment. Justified text looks like if center al	
your paragraphs look like blocks and defined have sides. It works by changing the spacing between each word depending on how many words it best sees can fit onto one line. The spacing between the words in the same for each line, but changes for the next. The different spacing	s between each word because of justified-aligned textThe idea behind
using justified text is that it looks more visually appealing than if left-justified alignment was used. Hence its common use in books and written material. What can very easily happen with justified alignment is in narrow columns or lines with a lot of long words, there can be massively splits up words into syllables. This creates for a bad user experience, making the text hard to read. This might happen without you knowing, if text is scaled/resized in such a way to force the text like that. When justified alignment is used in books or magazines, there are people that is scaled and in the control of the control	
any of those awkward spacings and even add hyphens if necessary. This is completely unreasonable to do on a website, there are just too many different screen sizes, screen resolutions, and zoom levels to have justified text on a website without having awkward alignment for at le	
of justified text out there. They love using justified text because it looks more visually appealing, and looks more professional. It can even save on printing costs due to the pages saved in making sure each line of text is used to its full width. Each line of a book or magazine usually happen. Each line of text in a book typically has around 60 characters per line, which is about 10 words per line. This enables books to have the visual appeal of justified text, and the user experience of left-justified text. In the case where justified text doesnt work, left alignment was a support of the contract	
practice for book designers and book formatters to choose what alignment is best for each book or written material. Overall justified text has its uses. It can definitely add to the visual appeal of paragraphs on a page, but if not used with caution it can cause the text to look fragme	
where its at. Its familiar, its fast, and its reliable. Left text alignment should be used in 95% of cases to help your readers read at an optimal, undiminished reading speed. You can use center alignment in small doses like main page headings without detracting from the users exper fewer lines of text, any more then left-aligned should be used as the text becomes too notably annoying to read. Only use justified text for mediums where its commonplace like material books or e-books. Rarely you can get away with justified text on a website, but left alignment	
alignment, make sure there are enough words per line so that the spaces between each word are fairly unnoticeable from one line to the next. Right alignment forces the reader to read in a weird way. Dont alienate your users for the sake of making your design feel unique and spe	
per line, and 3 or less lines of text all together. The only commonplace acceptable use of right alignment is navigations on websites. To horizontally center a block element (like div), use margin: auto; Nearly all browsers nowadays support CSS and many other applications do, too.	
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 lantroduction 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	
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 I see it time and time again with UI and web design people love centering, justifying, and right aligning text. People think this may make	
clean. But in reality, this is another example of people sacrificing user experiencing (UX) for user interface design (UI). Its time to put a stop to this! Does text alignment matter for accessibility? Lets jump in. The 4 types of alignment in graphic, UI, and web design are:In terms of ways. Left alignment is by the most popular text alignment, and also the default one. Left alignment is commonly associated with lengthy paragraph text, which enables it to be as easy as possible to read. Center alignment is used more sparingly and for visual appeal like with these	
pottom of pages as call to actions, giving less lengthy content better symmetry. Right alignment is used, if ever. But if it is used, its to align text up to another element for more visual flare. And lastly justified alignment is used more commonly with minimal or luxurious designment, and default tout alignment, the the heat for more designment, and when it is used, its training ly used its training ly used in small grant grant training ly used in small grant gran	
copular and default text alignment. Its the best for readability and user experience because of the way our eyes read. Right alignment used, and when it is used its typically used in small quantities. A great example of right alignment being can include a CTA button In the top right and have the main links align with it nicely making the navigation very scannable. Right-alignment should be used sparingly at best, as right alignment isnt good for user experience and readability. There are some	
modern business letterhead example. As a rule of thumb, right alignment should not be used if you are aligning text with more than 5 words at a time. This keeps the user experience in check while still adding in some rebellious formatting. In the example to the right, the right-aligned elements work well in most cases, and can actually help make use of otherwise unused space. In the letterhead pic	
align in that content area where my critiques lay. Center alignment can look great in small doses, but it can lead to problems when people overuse it. The reason why center text alignment is horrible for user experience is that with each new line the user reads, there is a brief mon	
the users reading speed. Intentionally handicapping your users ease of reading? Hell yes! This is what I think goes through peoples heads when they decide to center a large paragraph. What youll find is that in most cases centered text makes it worse for user experience. However, we do not think goes through peoples heads when they decide to center a large paragraph. What youll find is that in most cases centered text makes it worse for user experience. However, we do not the provide the provide text has become a through peoples heads when they decide to center a large paragraph. What youll find is that in most cases centered text makes it worse for user experience. However, the provide text has become a through people and the provide text and the provide	
are we doing this. Oh yes, because its pretty. Centered text has become such a common occurrence, especially in website design. In my opinion centered paragraphs are only acceptable up to a point, 3 lines of text to be specific. Anymore, it becomes too displeasing to read each line are only acceptable up to a point, 3 lines of text to be specific. Anymore, it becomes too displeasing to read each line are only acceptable up to a point, 3 lines of text to be specific. Anymore, it becomes too displeasing to read each line are only acceptable up to a point, 3 lines of text to be specific. Anymore, it becomes too displeasing to read each line are only acceptable up to a point, 3 lines of text to be specific. Anymore, it becomes too displeasing to read each line are only acceptable up to a point, 3 lines of text to be specific. Anymore, it becomes too displeasing to read each line are only acceptable up to a point, 3 lines of text to be specific. Anymore, it becomes too displeasing to read each line are only acceptable up to a point, 3 lines of text to be specific. Anymore, it becomes too displeasing to read each line are only acceptable up to a point, 3 lines of text to be specific. Anymore, it becomes too displeasing to read each line are only acceptable up to a point, 3 lines of text to be specific. Anymore, it becomes to a line are only acceptable up to a point are only acceptabl	
content on this blog is left aligned? Thats called user experience, and dont you forget it!You can just see from the second example the paragraph is flat-out annoying to read. This only gets worse on mobile devices. This is why left alignment should be used 95% of the time. Primary words and therefore lines of text. Most page titles arent long enough that text alignment becomes an issue with usability. However, with secondary titles (h2s) and anything under should always be left-aligned to match its paragraph text. An example of google title alignment is this	
some of the titles of some of my articles may get long enough that centering the text would become annoying to read and so I use left alignment. Justified text looks like if center alignment and left alignment had a child together. Justified text makes your paragraphs look like block	s and defined have sides. It works by changing the spacing between each
word depending on how many words it best sees can fit onto one line. The spacing between the words in the same for each line, but changes for the next. The different spacings between each word because of justified text. The idea behind using justified text is that it looks returned to the common use in books and written material. What can very easily happen with justified alignment is in narrow columns or lines with a lot of long words, there can be massive awkward spacings between each word. Sometimes its so bad where it splits up words into syllables. This	
might happen without you knowing, if text is scaled/resized in such a way to force the text like that. When justified alignment is used in books or magazines, there are people that go through each page and manually adjust the spacing to get rid of any of those awkward spacings and manually adjust the spacing to get rid of any of those awkward spacings and manually adjust the spacing to get rid of any of those awkward spacings and manually adjust the spacing to get rid of any of those awkward spacings and manually adjust the spacing to get rid of any of those awkward spacings and manually adjust the spacing to get rid of any of those awkward spacings and manually adjust the spacing to get rid of any of those awkward spacings and manually adjust the spacing to get rid of any of those awkward spacings and manually adjust the spacing to get rid of any of those awkward spacings and manually adjust the spacing to get rid of any of those awkward spacings and manually adjust the spacing to get rid of any of those awkward spacings and manually adjust the spacing to get rid of any of those awkward spacings and manually adjust the spacing to get rid of any of those awkward spacings and manually adjust the spacing to get rid of any of those awkward spacing the spacing that the spacing the spacing that the space th	
on a website, there are just too many different screen sizes, screen resolutions, and zoom levels to have justified text on a website without having awkward alignment for at least someone. Books and magazines are by far the most common source of justified text out there. They love the standard have a standard by the standard have a	
more professional.It can even save on printing costs due to the pages saved in making sure each line of text is used to its full width. Each line of a book or magazine usually have enough words so that the problems of justified alignment dont happen. Each line of text in a book typic line. This enables books to have the visual appeal of justified text, and the user experience of left-justified text. In the case where justified text doesnt work, left alignment would be used to make word spacing less distracting. Its common practice for book designers and book formation.	atters to choose what alignment is best for each book or written
material.Overall justified text has its uses. It can definitely add to the visual appeal of paragraphs on a page, but if not used with caution it can cause the text to look fragmented detracting from a positive user experience. Left justified text is where its at. Its familiar, its fast, and it your readers read at an optimal, undiminished reading speed. You can use center alignment in small doses like main page headings without detracting from the users experience. Paragraph text can get away with being center-aligned if it has 3 or fewer lines of text, any more then	
read.Only use justified text for mediums where its commonplace like material books or e-books.Rarely you can get away with justified text on a website, but left alignment will always be better for usability. If you are going to use justified alignment, make sure there are enough wo	
unnoticeable from one line to the next. Right alignment forces the reader to read in a weird way. Don't alienate your users for the sake of making your design feel unique and special. If you are going to use it, make sure each line has 3 or less words per line, and 3 or less lines of text.	
s navigations on websites. Get ready to learn how to approach the age old question faced by many a CSS practitioner: "How do I center a div?" The holy grail: vertical and horizontal centering along both the x-axis and the y-axis. The most modern and easiest way is display: grid; place-content: center;} We are centered! We are centered! Gotchas Collapse of child grid using auto-fit or auto-fill grid using auto-fit or auto-fill, minmax(10ch, 1fr)); The child grid will collapse in on	itself, in this case down to the min part of minmax, due to the justify-
content set in the place-content shorthand. CSS for "auto-fit fixed".grid-auto-fit, minmax(10ch, 1fr));} The fix is two-fold: switch the grid centering technique to use place-items instead of place-content, and then to specifically of the place are the grid to be a large and the grid to be	
prefer as a width value to create space for the grid columns. CSS for "auto-fit collapse".grid { display: grid; place-items: center;} .grid-autofit { display: grid; minmax(10ch, 1fr)); width: 80%;} Alternatively, you can use the very slightly more align-items: center; justify-content: center;} GotchasFlexbox has a slightly different behavior when a second item is added since flex items default to placement along the x-axis: CSS for "XY Flex Gotcha".flex { display: flex; align-items: center; justify-content: center;} We are center.	
column: CSS for "XY Flex Gotcha fixed".flex { display: flex; flex-direction: column; align-items: center;} Alternatively, wrap the children in a single element, especially if you don't want them to be affected by the outer flexbox alignment. If a child element use the parent container is grid as described previously. margin: auto is unique for flexbox and grid, and in the case you have only one child item, you can do the following for either flex or grid. CSS for "XY Alternative Flexbox Solution". flex; { display: flex; } .grid { display: grid; } .only	
childfren of block elements, can also be applied vertically which allows this solution to work. If you are unable to switch to grid or flexbox layout, here's a modern solution to this classic problem. Ensure the child elements are wrapped in a containing element for the following to we	ork: CSS for "XY Centering for Block Elements".parent { position:
relative; } .child-wrapper { position: absolute; top: 50%; left: 50%; transform: translate(-50%, -50%); } This combo works because when a percentage value is supplied to a translate definition, it based the percentage on the computed width (translateX) or height (translateY). In this cranslate(). Absolute positioning takes an element out of normal document flow after which we can apply precise values (as needed) to control its positioning in the document, or in this case, relative to the parent with the required position: relative on the parent element. After absolute position in the document is position in the document is position in the document.	
of the parent's height and width, respectively, we then use translate(-50%, -50%) to pull the child back up 50% of its own height and back left 50% of its own width. This results in a centered appearance that scales with the content. Gotchas Because we've used absolute posoitioning	ig, there's a chance the content will grow to overflow the parent, even if
ike in the demo the parent has a min-width which typically grows with the content except for absolute children. The fix for this is: use grid or flexbox:) Or prepare to create #allthemediaqueries. Solutions for centering vertically, aka on the y-axis. We only need one property to vertically entering to this is: use grid or flexbox:) Or prepare to create #allthemediaqueries. Solutions for centering vertically, aka on the y-axis. We only need one property to vertically entering to this is: use grid or flexbox:) Or prepare to create #allthemediaqueries. Solutions for centering vertically, aka on the y-axis. We only need one property to vertically entering to the y-axis. We only need one property to vertically entering to the y-axis. We only need one property to vertically entering to the y-axis. We only need one property to vertically.	
vertically aligned with: CSS for "Y Flexbox Solution".flex { display: grid; align-items: center;} I am centered vertically! Me too!GotchasIf you switch the default axis by adding flex-direction: column this solution fails. CSS for "Y Flexbox Gotcha for flex-direction: column".flex { display: grid; align-items: center;}	play: grid; flex-direction: column; align-items: center;} Now I am
centered horizontally? Hmm, me tooA huge cuplprit of issues when dealing with flexbox is missing that flipping the associated properties. For column, or y-axis flex layout, instead of align-items we now need to use: CSS for "Y Flexbox Gotcha fix for flex-directory. I am centered vertically! Me too! If possible, switch the layout model and use flex or grid. Otherwise, much like the XY solution, we'll use absolute positioning and transform, but only apply to translateY to move the child 50% of its height. CSS for "Y Flexbox Gotcha fix for flex-directory."	
wrapper { position: absolute; top: 50%; transform: translateY(-50%);} I am centered vertically! Me too! See XY Centering for Block Elements to learn why this works. Solutions for centering horizontally, aka on the x-axis. The justify- properties are for x-axis alignment: CSS for "X	Grid Solution".grid { display: grid; justify-content: center;} I am
centered horizontally! Me too! Again, this holds up if we switch the default axis with: CSS for "X Grid Solution for columns".grid { display: grid; justify-content: centered horizontally! Me too! To center along the x-axis, which is the default flex! { display: flex; justify-content: center;} I am centered horizontally! Me too! Gotchas this point, you know what's coming - this will fail for flex-direction: column.We'll fix it by using align-items instead of justify-content: CSS for "X Flexbox Solution for flex-direction: column".flex {	
horizontally! Me too! This is the classic solution of using auto margins, although it must be placed on each element you wish to center individually. For the demo, I've also set a width since by default block elements take up the full-width of their container, which visually opposes the	ne centering. CSS for "X Centering for Block Elements".block div {
margin-left: auto; margin-right: auto; width: 60%;} I am centered horizontally! Me too! The use case here is a requirement for centering of items of dynamic/unknown width relative to the associated trigger. We'll use a requirement for centering of items of dynamic/unknown width relative to the associated trigger. We'll use a requirement for centering of items of dynamic/unknown width relative to the associated trigger. We'll use a requirement for centering of items of dynamic/unknown width relative to the associated trigger. We'll use a requirement for centering of items of dynamic/unknown width relative to the associated trigger. We'll use a requirement for centering of items of dynamic/unknown width relative to the associated trigger. We'll use a requirement for centering of items of dynamic/unknown width relative to the associated trigger. We'll use a requirement for centering of items of dynamic/unknown width relative to the associated trigger. We'll use a requirement for centering of items of dynamic/unknown width relative to the associated trigger. We'll use a requirement for centering of items of dynamic/unknown width relative to the associated trigger. We'll use a requirement for centering of items of dynamic/unknown width relative to the associated trigger. We'll use a requirement for centering of items of dynamic/unknown width relative to the associated trigger.	
101 discussion on alignments. Centered alignments are an easy place to go wrong and if you dont know how to wield them properly, the result is a very poorly structured page. Join us as we take a look at why centered alignments tend to be weak, where you should avoid them and	d how you should be using them. Envato gives you unlimited access to
22+ million pro design resources, themes, templates, photos, graphics and more. Everything you'll ever need in your design resource toolkit. See More Centered AlignmentsOne of the first things that youll learn in any basic design layout class is that centered alignments are weal protest, but well explore this more in-depth in a minute. Centered alignments are by no means something only used by beginners, but they do in fact tend to be the go-to option for these groups. The reason for this is complicated. For some reason, we instinctively	
balance and centered alignments are nothing if not balanced, so they make sense. Secretaries, CEOs, teachers and all other manner of other professionals that dip their toe into design almost always run straight for centered alignments on any project. In practice, centered alignments are nothing if not balanced, so they make sense. Secretaries, CEOs, teachers and all other manner of other professionals that dip their toe into design almost always run straight for centered alignments on any project. In practice, centered alignments are nothing if not balanced, so they make sense.	nents are often the source of trouble in a layout. Learning both how to
dentify and how to fix these problems is a major step in your early design education. The Problem with Centered Alignments with theory alone so lets take a look at an example. Lets say youre making little flyers to hand with with will almost always look something like this: This isnt a hideous flyer, in fact Ive seen far worse in the real world. However, for a number of reasons, its definitely not what you would call a strong design. If you handed me the flyer design above and told me to improve it as much	
personality, this is what I would give back to you: This design simply feels better, and not just because I refuse to ever use the Party LET typeface, especially when typing the word party. I also split up the content a little better added some emphasis points and, most importantly, classically when typing the word party.	hanged to a left alignment. Notice how the left alignment feels very
anchored. The text on the page has a clear starting point and you can follow that hard edge all the way from top to bottom. Despite the fact that the first flyer feels balanced, it doesnt feel anywhere near as structured. When To Stay Away from Centered AlignmentsAll right, so you so what? The reality is a little more complicated than that. Theres nothing inherently bad about a centered alignment, you just have to know how to properly yield one if youre going to implement it with any amount of success. The first thing you need to learn is when not to implement in the first flyer feels balanced, it doesn't feel anywhere near as structured. When To Stay Away from Centered Alignments All right, so you need to learn it with any amount of success. The first thing you need to learn is when not to implement it with any amount of success. The first thing you need to learn is when not to implement it with any amount of success.	
have a lot of content. As we saw in the last example, centered alignments are pretty poor for large blocks of text. The lack of a hard edge makes reading difficult and erratic. The same goes for relying on a centered layout for your page as a whole, most of the time, its simply not the	he best idea. The website wireframe below is quite problematic from a
ayout standpoint. Now, dont read that as advising against centering your content on the page. Theres a difference between building a website that uses a center alignment. You can easily have left or right aligned content that is still conversation entirely. Aesthetics Also, dont confuse this as a matter of aesthetics. As the page below shows, you can create beautiful sites that rely heavily on centered alignments. Its very important in any design to analyze your goals. If a significant degree of readability is one of the page below shows, you can create beautiful sites that rely heavily on centered alignments.	
separate or even directly at odds with this goal. The trick is to find the balance between the two. The end result is almost always going to be completely subjective. The designer above decided that this chunk of content was small enough to be center aligned. I may think its pushing the completely subjective.	g the limit a little but that doesnt mean Im right and he/she is wrong, it
ust means we draw the line in different places. How to Use Centered AlignmentsNow, with all of that said, there is no reason that you should abandon using centered alignments altogether. To do so would be to remove a key element from your layout toolbox. Once again, you just million different ways to successfully implement a centered layout so dont imagine that the following examples are exhaustive. However, they can serve as a way for you to start to get a feel for a solid layout versus a weak one. Very Little ContentOne of the first places you can start	
have very little content. Check out the example below. This makes perfect sense, if the main problem with centered alignments is that they don't hold up well with a lot of content, then theyll likely be just fine if you only have a few items. In fact, theyre usually better under these ci	rcumstances. If you implemented a left alignment with the design above,
the page would likely look very empty. However, with the current centered layout, it looks classy and well-spaced. Selective Application Avoiding centered layouts for anything but the simplest designs sounds like an awfully restrictive way to design, and it is. The key is to not avoid design with a stronger alignment. A beginners layout class might teach you to grab one alignment and stick to it, but as you improve youll find that mixing alignments is often a great way to add some variety to a design. As an example, check out the awesome page below from Assignments is often a great way to add some variety to a design.	
eft and right edges. However, there are very specific portions that revert to a centered alignment. The headline is the most popular place to attempt this. Large, centered headlines and subheads look great in justified layouts. Also notice the smaller areas like the alignment to be implemented. Once again, each of these blocks contains a fairly small amount of content so the result isnt overwhelming. Quick Tip: Wrap It UpOne of the simplest and most useful tricks for implementing centered text alignments within a larger layout is shown in	
that all of the have a fairly justified alignment. With the exception of the left-aligned navigation, theres both a hard left and a hard right edge that defines the content. Now, within that framework I have a little more freedom to play around. Notice that Box One and Box Two actual	
ooking perfect within the overall layout. Anytime you have some center-aligned content that just doesnt feel like its working, try wrapping it in a box that conforms to the overall page layout to see if the design improves. Conclusion To sum up, centered alignments are weak, but no	
grasp to be a good designer. Weak implies that they can easily be taken too far and are easy to abuse. However, you can and should be using centered alignments in your designs. Make sure that, if your entire layout is built on a centered alignment, you have a very simple design was a whole, then experiment with selectively dropping in centered alignments in key areas such as headlines. Finally, as a quick trick when youre in	
flow of the rest of the page. In this recipe, you will see how to center one box inside another by using flexbox and grid, centering content both horizontally. To place an item into the center of another box horizontally and vertically. Click "Play" in the code blocks below border: 2px solid rgb(95 97 110); border-radius: 0.5em; padding: 20px; width: 10em; center; border: 2px solid rgb(75 70 74); border-radius: 0.5em; font: 1.2em sans-serif; height: 200px; display: flex; align-items: center; justify-content: center; To center a box within another	
display property to flex. Then set align-items to center for vertical centering (on the block axis) and justify-content to center for horizontal centering (on the inline axis). And that's all it takes to center one box inside another! I am centered!div { border: solid 3px; padding: 1em; may be a center one box inside another! I am centered!div { border: solid 3px; padding: 1em; may be a center one box inside another! I am centered!div { border: solid 3px; padding: 1em; may be a center one box inside another! I am centered!div { border: solid 3px; padding: 1em; may be a center one box inside another! I am centered!div { border: solid 3px; padding: 1em; may be a center one box inside another! I am centered!div { border: solid 3px; padding: 1em; may be a centered!div { border: solid 3	ax-width: 75%;}.item { border: 2px solid rgb(95 97 110); border-radius:
0.5em; padding: 20px; width: 10em; \}.container \{\text{ height: 8em; border: 2px solid rgb(75 70 74); border-radius: 0.5em; font: 1.2em sans-serif; display: flex; align-items: center; \}We set a height for the container to demonstrate that the inner item is indeed ver center; on the container, you can also vertically center the inner item by setting align-self to center on the inner item itself. Another method you can use for centering one box inside another is to first make the containing box a grid container and then set its place-items property to	
centered!div { border: solid 3px; padding: 1em; max-width: 75%;}.item { border: 2px solid rgb(95 97 110); border-radius: 0.5em; padding: 20px; width: 10em;}.container { height: 8em; border: 2px solid rgb(75 70 74); border-radius: 0.5em; font: 1.2em sans-serif; display: grid; play:	ace-items: center;}Instead of applying place-items: center; on the
container, you can achieve the same centering by setting place-content: center; on the container or by applying either place-self: centering things is one of the most difficult aspects of CSS. The methods themselves usually aren't different element you're trying to center, or whether you're trying to center, or whether you're trying to center, or whether you're centering it horizontally. In this tutorial, we'll go over how to center different elements horizontally, vertically, and both vertically and	
centering them vertically. Here are some common elements you may want to center horizontally and different ways to do it. How to Center Text with the CSS Text-Align Center PropertyTo center text or links horizontally, just use the text-align property with the value center: Hello	o, (centered) World!.container { font-family: arial; font-size: 24px;
margin: 25px; width: 350px; height: 200px; outline: dashed 1px black;} p { text-align: center;} Use the shorthand margin property with the value 0 auto to center block-level elements like a div horizontally: .container { font-family: arial; font-size: 24px; margin: 25px; width: 350px; height: 50px; background-color: red; margin: 0 auto;} Flexbox is the most modern way to center things on the page, and makes designing responsive layouts much easier than it used to be. However, it's not fully supported in some legacy browsers like Internet Explorer. To center	
content: center to the parent element: .container { font-family: arial; font-size: 24px; margin: 25px; width: 350px; height: 200px; outline: dashed 1px black; display: flex; justify-content: center;} .child { width: 50px; height: 50px; background-color: red;} Centering elements vertically contents.	ally without modern methods like Flexbox can be a real chore. Here we'll
go over some of the older methods to center things vertically first, then show you how to do it with Flexbox. For a long time this was the go-to way to center things vertically. For this method you must know the height of the element you want to center. First, set the position proper to absolute and top to 50%: .container { font-family: arial; font-size: 24px; margin: 25px; width: 350px; height: 50px; height: 50px; background-color: red; position: absolute; top: 50%; But that red; container and top to 50%: .container are font-family: arial; font-size: 24px; margin: 25px; width: 350px; height: 50px; height: 50px; background-color: red; position: absolute; top: 50%; But that red; font-family: arial; font-size: 24px; margin: 25px; width: 350px; height: 50px; height: 50px; background-color: red; position: absolute; top: 50%; But that red; font-family: arial; font-size: 24px; margin: 25px; width: 350px; height: 50px;	
center the child element, set the margin-top property to -(half the child element's height):.container { font-family: arial; font-size: 24px; margin: 25px; width: 350px; height: 200px; outline: dashed 1px black; position: relative;} .child { width: 50px; height: 50px; background-color:	red; position: absolute; top: 50%; margin-top: -25px; } If you don't know
the height of the element you want to center (or even if you do), this method is a nifty trick. This method is very similar to the negative margins method above. Set the position property of the parent element to relative. For the child element, set the position property to absolute and the child element, just use transform: translate(0, -50%): .container { font-family: arial; font-size: 24px; margin: 25px; width: 350px; height: 200px; outline: dashed 1px black; position: relative;} .child { width: 50px; height: 50px; background-color: red; position: absolute; top: 50%	
shorthand for translateX(0) and translateY(-50%). You could also write transform: translateY(-50%) to center the child element vertically. Like centering things horizontally, Flexbox makes it super easy to center things vertically. To center an element vertically, apply display: flex a	and align-items: center to the parent element: .container { font-family:
arial; font-size: 24px; margin: 25px; width: 350px; height: 200px; outline: dashed 1px black; display: flex; align-items: center; child { width: 50px; background-color: red;} This is very similar to the method above to center an element vertically. Like last time, you make to position property of the parent element to relative. Then set the child's position property to absolute, top to 50%, and left to 50%. This just centers the top left corner of the child element vertically and horizontally. To truly center the child element, apply a negative top margin set to 50%.	
the child element's width: .container { font-family: arial; font-size: 24px; margin: 25px; width: 350px; height: 200px; outline: dashed 1px black; position: relative;} .child { width: 50px; height: 50px; background-color: red; position: absolute; top: 50%; left: 50%; margin: -25px 0 0 -25px	25px; } Use this method to center an element vertically and horizontally
f you don't know its exact dimensions and can't use Flexbox. First, set the position property of the parent element to relative. Next, set the child element's position property to absolute, top to 50%, and left to 50%. Finally, use transform: translate(-50%, -50%) to truly center the child element's position: absolute; top: 50%; height: 200px; outline: dashed 1px black; position: relative; child { width: 50px; height:	nd horizontally. This is really just a combination of the two previous
Flexbox methods. To center the child element(s) horizontally and vertically, apply justify-content: center and align-items: center to the parent element: .container { font-family: arial; font-size: 24px; margin: 25px; width: 350px; height: 200px; outline: dashed 1px black; display: flexbox methods.	x; justify-content: center; align-items: center;} .child { width: 50px;
height: 50px; background-color: red;}That's everything you need to know to center with the best of 'em. Now go forth and center all the things. A combinator is something that explains the relationship between the selectors. A CSS selector can contain more than one simple selector can combinator (*)Next sibling combin	lowing example selects all elements inside elements:div p {background-
color: yellow;}Try it Yourself Child Combinator (>)The child combinator selects all elements that are the children of a specified element. The following example selects all elements that are children of a element:div > p {background-color: yellow;}Try it Yourself The next sibling construction of a specified element. Sibling elements must have the same parent element, and "adjacent" means "immediately following example selects the first element that are placed immediately after elements:div + p { background-color: yellow;}Try it Yourself Subsequent-sibling elements must have the same parent element, and "adjacent" means "immediately following example selects the first element that are placed immediately after elements:div > p { background-color: yellow;}Try it Yourself Subsequent-sibling element. The following example selects all elements that are children of a element:div > p { background-color: yellow;}Try it Yourself The next sibling example selects all element. The following example selects all elements are children of a element element. The following example selects all elements are children of a specified element. The following example selects all elements are children of a element element. The following example selects all elements are children of a element element. The following example selects all elements are children of a element element element element. The following example selects all elements are children of a element ele	
that are next siblings of a specified element. The following example selects all elements that are next siblings of elements: div ~ p { background-color: yellow;} Try it Yourself	g community (), the subsequent similing comminator serects an elements
When to use alignes. When to do alignment. When to do balancing and alignment. When should alignment be done. When is the best time to use a lignment.	lianment alianment center Aliannes
When to use alignas. When to do alignment. When to do balancing and alignment. When should alignment be done. When is the best time to use center alignment. Al When to use centered text.	nymment anymment.center. Anynment at.