Independent Submission
Internet-Draft
Intended status: Standards Track
Expires: 16 July 2026

W. Hackett
January 2026

Unified Rendering of Email Standard (URES)
draft-hackett-ures-00

Abstract

This document specifies requirements for the uniform rendering of HTML email content across mail user agents (MUAs). It addresses critical inconsistencies in how email clients interpret HTML and CSS, including dark mode adaptation, embedded asset handling, positioning constraints, and secure SVG rendering. The specification aims to eliminate the current fragmentation in email rendering while maintaining backwards compatibility with existing MIME standards (RFCs 2045-2049) and the Internet Message Format (RFC 5322).

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at https://datatracker.ietf.org/drafts/current/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on 5 July 2026.

Copyright Notice

Table of Contents

1. Introduction

1.1. Background

Email remains one of the most critical communication channels for both personal and business communications. Despite decades of development, the rendering of HTML email content remains frustratingly inconsistent across mail user agents (MUAs).

The current standards landscape is fragmented:

* RFC 5322 [RFC5322] defines the Internet Message Format, specifying header fields and basic body structure.

* RFCs 2045-2049 [RFC2045] define MIME, enabling multi-part messages with various content types including text/html.

* The W3C maintains HTML and CSS specifications, but these are designed for web browsers, not email clients.

* No RFC or standard body governs how email clients SHOULD or MUST render HTML and CSS content.

This gap has resulted in a situation where email developers must maintain extensive compatibility matrices, use archaic techniques such as table-based layouts, and accept that their carefully crafted emails will appear differently—or broken—across different clients.

1.2. Problem Statement

The following critical issues plague current email rendering:

1. *CSS Support Inconsistency:* Properties universally supported in browsers (flexbox, grid, CSS variables) have zero or partial support in major email clients. Gmail strips style tags in non-AMP contexts. Outlook uses Microsoft Word's rendering engine.

2. *Dark Mode Chaos:* Email clients implement dark mode adaptation differently. Some invert all colours, some honour author preferences, some do nothing. Text becomes illegible when foreground colours are inverted but backgrounds are not.

3. *Image Handling:* Base64 inline images are blocked by many clients citing size concerns. CID attachments render inconsistently. Remote images enable tracking and are blocked by default.

4. *SVG Non-Support:* Most email clients strip or fail to render SVG entirely, despite it being a W3C standard since 2001. Executives receive emails with missing logos and charts.

5. *Positioning Attacks:* In web-based clients (Gmail, Outlook.com), malicious use of position:fixed or high z-index values could overlay phishing content atop the webmail UI.

6. *Layout Primitives:* Modern CSS layout (flexbox, grid) is unsupported in Outlook desktop, forcing continued use of nested HTML tables—a technique deprecated in web development since 2010.

7. *Font Rendering:* @font-face and web fonts work in approximately 40% of email clients, with no fallback standardisation.

8. *Media Queries:* Support for responsive design via @media queries is inconsistent, with Gmail notably stripping them entirely.

1.3. Requirements Language

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

2. Terminology

Mail User Agent (MUA) An application that accesses email on behalf of a user. This includes desktop applications (Apple Mail, Outlook, Thunderbird), web-based clients (Gmail, Outlook.com, Yahoo Mail), and mobile applications (iOS Mail, Gmail app).

Rendering Engine The component of an MUA responsible for interpreting and displaying HTML/CSS content. Some MUAs use web browser engines (WebKit, Blink), others use proprietary engines (Microsoft Word in Outlook desktop).

Dark Mode A display preference where the UI uses a dark background with light text. Email clients may automatically adapt email content to match this preference.

Colour Scheme The light or dark preference indicated by the email author or inferred by the email client.

Inline Image An image embedded directly in the HTML using a data: URI with base64 encoding, or referenced via Content-ID (CID).

Executive-Safe Content that renders correctly in email clients commonly used by business executives, particularly Outlook desktop, Apple Mail, and Gmail.

Stacking Context A three-dimensional conceptualisation of HTML elements along an imaginary z-axis relative to the user. Elements with z-index values form stacking contexts.

Content Isolation The practice of preventing email content from affecting or overlaying the host application’s user interface.

3. Document Structure Requirements

3.1. HTML Version

Conforming MUAs MUST support HTML5 document structure as defined in [HTML5]. The following DOCTYPE declaration MUST be recognised:

```
<!DOCTYPE html>
```

MUAs MUST also accept and render documents using XHTML and HTML4 doctypes for backwards compatibility. However, email authors SHOULD use HTML5.

MUAs MUST NOT require an XML declaration for HTML email content.

3.2. Required Document Structure

MUAs MUST correctly parse and render the following minimal document structure:

```
<!DOCTYPE html>
<html lang="en" xmlns="http://www.w3.org/1999/xhtml"
      xmlns:o="urn:schemas-microsoft-com:office:office">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width,initial-scale=1">
  <meta name="color-scheme" content="light dark">
  <meta name="supported-color-schemes" content="light dark">
  <title>Email Subject</title>
  <!--[if mso]>
  <noscript>
    <xml>
      <o:OfficeDocumentSettings>
        <o:PixelsPerInch>96</o:PixelsPerInch>
      </o:OfficeDocumentSettings>
    </xml>
  </noscript>
  <![endif]-->
  <style>
    /* Author styles */
  </style>
</head>
<body>
  <!-- Content -->
</body>
</html>
```

MUAs MUST preserve the html element’s lang attribute and make it available to accessibility tools.

MUAs MUST parse and apply the viewport meta tag for responsive rendering on mobile devices.

MUAs supporting Microsoft Office conditional comments SHOULD interpret <!--[if mso]> blocks appropriately.

3.3. Style Placement

MUAs MUST support CSS styles in ALL of the following locations:

1. Inline styles via the style attribute on any HTML element

2. Embedded styles via style elements in the head

3. Embedded styles via style elements in the body

MUAs MUST NOT strip or ignore style elements based on their placement in the document.

MUAs MUST process multiple style elements in document order, applying standard CSS cascade rules.

MUAs SHOULD support link rel="stylesheet" elements referencing external stylesheets, subject to security controls defined in Section 8.

When an MUA chooses not to load external stylesheets for security reasons, it MUST still apply all inline and embedded styles.

4. CSS Support Requirements

4.1. MUST Support Properties

Conforming MUAs MUST support the following CSS properties with full specification compliance:

4.1.1. Box Model Properties

margin, margin-top, margin-right, margin-bottom, margin-left, padding, padding-top, padding-right, padding-bottom, padding-left, border, border-width, border-style, border-color, border-top, border-right, border-bottom, border-left, border-radius, border-top-left-radius, border-top-right-radius, border-bottom-left-radius, border-bottom-right-radius, width, min-width, max-width, height, min-height, max-height, box-sizing

4.1.2. Typography Properties

font-family, font-size, font-weight, font-style, line-height, text-align, text-decoration, text-decoration-line, text-decoration-color, text-decoration-style, text-transform, letter-spacing, word-spacing, white-space, vertical-align

4.1.3. Colour and Background Properties

color, background-color, background-image (for gradients and data: URIs), background-position, background-repeat, background-size, background (shorthand), opacity

4.1.4. Layout Properties

display (block, inline, inline-block, none, table, table-row, table-cell, flex, inline-flex), position (static, relative), float, clear, overflow, overflow-x, overflow-y

4.1.5. CSS Selectors

MUAs MUST support the following CSS selectors:

Type selectors (p, div, table), Class selectors (.classname), ID selectors (#idname), Descendant combinator (div p), Child combinator (div > p), Adjacent sibling combinator (h1 + p), Attribute selectors ([attr], [attr=value], [attr~=value], [attr|=value], [attr^=value], [attr$=value], [attr*=value]), Pseudo-classes (:first-child, :last-child, :nth-child(), :hover, :active, :focus, :visited, :link), Pseudo-elements (::before, ::after, ::first-line, ::first-letter)

4.2. Positioning Constraints

For security reasons detailed in Section 8.2, MUAs MUST constrain the position property as follows:

4.2.1. Permitted Values

MUAs MUST support:

* position: static (default, normal flow)

* position: relative (offset from normal position)

4.2.2. Prohibited Values in Web Clients

Web-based MUAs (those running within a web browser context) MUST treat the following values as position: static:

* position: fixed

* position: absolute

* position: sticky

When an email contains these prohibited position values, the MUA MUST either:

a. Render the element as position: static, OR

b. Render the element as position: relative with top, left, right, and bottom values clamped to values that cannot cause the element to escape its container.

4.3. z-index Handling

The z-index property controls stacking order within a stacking context. MUAs MUST implement z-index handling that prevents malicious overlay attacks.

4.3.1. Stacking Context Isolation

MUAs MUST create a new stacking context for email content that is isolated from the host application. This means:

* z-index values within email content MUST NOT affect elements outside the email content container.

* The email content container MUST have a z-index lower than any application UI that could be targeted for spoofing.

4.3.2. Maximum z-index

MUAs SHOULD normalise z-index values within email content. If an email specifies z-index: 2147483647 (the maximum 32-bit signed integer), this MUST still be rendered below application UI.

Implementation approaches:

a. Contain email content in an iframe with srcdoc

b. Apply CSS containment (contain: strict)

c. Clamp z-index values to a maximum safe value

d. Render email in a separate compositing layer

4.4. Flexbox

MUAs MUST support CSS Flexible Box Layout [CSS-FLEXBOX-1] including:

display: flex, display: inline-flex, flex-direction, flex-wrap, flex-flow, justify-content, align-items, align-content, order, flex-grow, flex-shrink, flex-basis, flex, align-self

4.5. CSS Grid (SHOULD Support)

MUAs SHOULD support CSS Grid Layout [CSS-GRID-1]. Where supported, MUAs MUST implement:

display: grid, display: inline-grid, grid-template-columns, grid-template-rows, grid-template-areas, grid-gap, gap, row-gap, column-gap, grid-column, grid-row, grid-area

5. Dark Mode Adaptation

Dark mode support is critical for email legibility. This section defines requirements for consistent dark mode behaviour.

5.1. Colour Scheme Meta Tag

MUAs MUST recognise and honour the color-scheme meta tag:

```
<meta name="color-scheme" content="light dark">
```

Valid values:

* "light" - Email is designed for light mode only
* "dark" - Email is designed for dark mode only
* "light dark" - Email supports both modes
* "only light" - Email MUST NOT be adapted for dark mode
* "only dark" - Email MUST NOT be adapted for light mode

When color-scheme is set to "only light", MUAs MUST NOT invert, adjust, or otherwise modify the colours specified by the email author, even when the MUA or operating system is in dark mode.

5.2. Media Query Support

MUAs MUST support the prefers-color-scheme media query:

```
@media (prefers-color-scheme: dark) {
  body {
    background-color: #1a1a1a;
    color: #ffffff;
  }
}

@media (prefers-color-scheme: light) {
  body {
    background-color: #ffffff;
    color: #1a1a1a;
  }
}
```

When the user has selected dark mode in their MUA or operating system, prefers-color-scheme: dark MUST match.

MUAs MUST NOT strip or ignore @media rules, including @media (prefers-color-scheme: ...) rules.

5.3. Colour Inversion Rules

When an MUA applies automatic dark mode adaptation to an email that has not opted out, the following rules MUST be followed:

5.3.1. Inversion Pairing

MUAs MUST NOT invert foreground colours without also inverting the corresponding background colour. Specifically:

* If text colour is inverted from dark to light, the background behind that text MUST also be inverted from light to dark.

* If a background is inverted, all foreground elements (text, borders) on that background MUST be correspondingly adjusted.

5.3.2. Contrast Preservation

When inverting colours, MUAs MUST maintain a minimum contrast ratio of 4.5:1 for normal text and 3:1 for large text, as defined in WCAG 2.1 [WCAG21].

6. Image and Asset Handling

6.1. Inline Base64 Images

Inline images using data: URIs provide a tracking-free method for including images directly in email content. MUAs MUST support this mechanism.

6.1.1. Syntax

```
<img src="
AAAFCAYAAACNbyblAAAAHElEQVQI12P4//8/w38GIAXDIBKE0DHxgljNBAAO
9TXL0Y4OHwAAAABJRU5ErkJggg==" alt="Red dot">
```

6.1.2. Required Support

MUAs MUST render inline base64 images for the following MIME types:

* image/png

* image/jpeg (including image/jpg as an alias)

* image/gif

* image/webp

* image/svg+xml (subject to security restrictions in Section 7)

6.1.3. Size Limits

MUAs MUST support inline images with encoded sizes of at least:

* Individual image: 1 MB (after base64 encoding)

* Total per email: 10 MB (cumulative across all inline images)

MUAs MAY support larger sizes but MUST NOT fail to render images below these thresholds.

6.1.4. Rendering Behaviour

MUAs MUST render inline base64 images:

* Without requiring user interaction to "load images"

* Without displaying tracking/privacy warnings for these images

* Immediately upon email open (no lazy-load deferral)

This behaviour is REQUIRED because inline images:

* Cannot be used for tracking (no external request is made)

* Are already fully downloaded as part of the email content

* Provide a privacy-preserving alternative to remote images

6.2. CID (Content-ID) Images

Content-ID references allow images attached to the email to be displayed inline. MUAs MUST support this mechanism as defined in RFC 2392 [RFC2392].

6.2.1. Syntax

In the HTML body:

```
<img src="cid:image001@example.com" alt="Inline image">
```

6.2.2. Required Behaviour

MUAs MUST:

* Resolve cid: URLs to the corresponding MIME part
* Render the image inline at the point of reference
* Support CID references in both img elements and CSS background-image properties
* Render CID images without requiring "load images" interaction

7. SVG Rendering

Scalable Vector Graphics provide resolution-independent images essential for logos, icons, charts and diagrams. This section defines an "Executive-Safe" SVG profile that balances functionality with security.

7.1. Executive-Safe SVG Profile

The Executive-Safe SVG Profile defines a restricted subset of SVG that MUST be supported by conforming MUAs. This profile is named to emphasise that SVG content—particularly logos and charts—must render correctly in email clients commonly used by business executives.

The Executive-Safe profile includes:

* All SVG 1.1 shape elements
* Text elements
* Gradients and patterns
* Basic filters (subset)
* Symbols and use references (internal only)
* Styling via presentation attributes and internal stylesheets

The Executive-Safe profile excludes:

* Script elements and event handlers
* External references
* Animation elements (use CSS animation instead)

* ForeignObject element

* Certain filter primitives

7.2. Prohibited SVG Elements

MUAs MUST strip or neutralise the following SVG elements:

7.2.1. Script and Event Handlers

The script element and any element with on* attributes (onclick, onload, onerror, etc.) MUST be stripped.

7.2.2. External References

The following MUST NOT be resolved:

* use href="external.svg#symbol"

* image href="external.png"

* a href="javascript:..."

* feImage xlink:href="external.svg"

Note: Internal references within the same SVG document ARE permitted:

```
<use href="#internal-symbol">   <!-- OK -->
```

8. Security Considerations

Email is a primary vector for phishing, malware distribution and social engineering attacks. MUAs MUST implement robust security measures while rendering HTML content.

8.1. Script Execution

MUAs MUST NOT execute JavaScript or any other scripting language in email content.

8.1.1. Prohibited Elements

The following elements MUST be stripped or rendered inert:

* script element

* noscript element (SHOULD be displayed)

8.1.2. Prohibited Attributes

The following attributes MUST be stripped from all elements:

All on* event handlers: onclick, onload, onerror, onmouseover, onfocus, onblur, onsubmit, etc.

8.1.3. JavaScript URLs

The following URL schemes MUST NOT be executable:

* javascript:
* vbscript:
* data:text/html (when containing script)

MUAs MUST either:

a. Strip href/src attributes containing these schemes, OR

b. Replace them with safe values (e.g., href="#")

8.2. Positioning Attack Prevention

In web-based MUAs, malicious use of CSS positioning could overlay deceptive content atop the webmail UI.

8.2.1. Fixed Position Blocking

MUAs MUST NOT render position: fixed as specified. Such elements:

* MUST be rendered as position: static or position: relative
* MUST remain within the email content area
* MUST NOT overlay any part of the MUA interface

8.2.2. z-index Sandboxing

MUAs MUST ensure that email content z-index values cannot cause elements to render above the MUA interface.

Implementation: Create a new stacking context for email content that is positioned below all MUA UI elements.

8.3. Tracking Prevention

8.3.1. Pixel Tracking

Remote images are commonly used as tracking pixels. MUAs SHOULD:

* Block remote images by default
* Provide clear UI indication when images are blocked
* Allow users to load images per-email or per-sender

8.3.2. Inline Content as Alternative

The mandatory support for base64 inline images (Section 6.1) and CID images (Section 6.2) provides senders with tracking-free alternatives for embedding images.

9. Accessibility

Email content MUST be accessible to users with disabilities. MUAs MUST NOT strip or alter HTML that supports accessibility.

9.1. Semantic HTML

MUAs MUST correctly interpret and render semantic HTML elements:

header, footer, nav, main, article, section, aside, h1 through h6, p, ul, ol, li, dl, dt, dd, table, th, td (with scope attributes), figure, figcaption, blockquote, cite, strong, em

MUAs MUST NOT flatten semantic structure (e.g., converting all elements to div or span).

9.2. Alternative Text

MUAs MUST:

* Display alt attribute text when images are not loaded
* Expose alt text to assistive technologies
* Support role and aria-* attributes

MUAs MUST NOT strip:

alt, title, role, aria-label, aria-labelledby, aria-describedby, aria-hidden, aria-live, aria-atomic, aria-relevant

9.3. Colour Contrast

When MUAs apply automatic dark mode adaptation (Section 5.3), the resulting colour combinations MUST maintain WCAG 2.1 AA contrast ratios:

* Normal text: 4.5:1
* Large text: 3:1
* UI components and graphics: 3:1

10. Conformance Levels

To facilitate gradual adoption, this specification defines three conformance levels.

10.1. URES Level 1 (Baseline)

An MUA conforming to URES Level 1 MUST:

* Support all properties in Section 4.1
* Support inline base64 images (Section 6.1)
* Support CID images (Section 6.2)
* Strip all JavaScript (Section 8.1)
* Implement positioning constraints (Section 4.2)
* Support prefers-color-scheme media query (Section 5.2)
* Preserve semantic HTML (Section 9.1)

10.2. URES Level 2 (Enhanced)

An MUA conforming to URES Level 2 MUST:

* Meet all Level 1 requirements
* Support the Executive-Safe SVG Profile (Section 7)
* Support CSS Flexbox (Section 4.4)
* Implement dark mode adaptation rules (Section 5.3)
* Support picture element

* Support CSS custom properties (variables)

10.3. URES Level 3 (Full)

An MUA conforming to URES Level 3 MUST:

* Meet all Level 2 requirements

* Support CSS Grid (Section 4.5)

* Support @font-face

* Support CSS animations (with user preference for reduced motion)

* Implement full CSP

* Provide tracking prevention (Section 8.3)

11. Adoption Considerations

This section addresses practical considerations for adoption of this specification by email client vendors.

11.1. Open Source Reference Implementations

The authors recognise that some email client vendors have historically been reluctant to invest in standards-compliant rendering engines, preferring instead to repurpose existing technologies (such as word processing engines) that were never designed for standards-compliant HTML/CSS rendering.

To address this, the community intends to develop and maintain open source reference implementations of URES-compliant rendering components. These implementations will be:

* Licensed under permissive terms (MIT, Apache 2.0, or similar)

* Designed for embedding in both native and web-based clients

* Maintained by the community with regular security updates

* Tested against the URES conformance test suite

Vendors who have historically chosen not to invest in standards-compliant rendering may simply adopt these community-maintained implementations when ready, rather than continuing to maintain proprietary solutions that diverge from industry consensus.

11.2. Precedent for Community-Driven Adoption

This approach mirrors successful adoption patterns across the software industry, where vendors have benefited from community-developed standards and implementations rather than maintaining proprietary alternatives:

* React (Meta): Now embedded across enterprise applications, including those from vendors who initially resisted component-based UI architectures.

* GraphQL (Meta): Adopted by major cloud platforms after community momentum demonstrated clear benefits over proprietary query languages.

* OpenTelemetry (CNCF): Observability standard now integrated across all major cloud providers, replacing fragmented proprietary instrumentation.

* Chromium (Google): The rendering engine now powers multiple browsers from vendors who previously maintained independent engines, including Microsoft Edge.

* VS Code / Monaco Editor (Microsoft): An acknowledgement that community-driven tooling often surpasses proprietary alternatives.

Vendors currently maintaining non-compliant rendering engines face an ongoing maintenance burden for technology that provides no competitive advantage. The availability of URES-compliant open source alternatives offers a path to reduced maintenance costs while improving interoperability for their users.

11.3. Incremental Adoption Path

Vendors need not adopt all conformance levels simultaneously. The three-tier conformance structure (Section 10) enables:

1. Level 1 adoption with minimal investment, addressing the most critical rendering inconsistencies.

2. Level 2 adoption as community implementations mature, adding modern layout and dark mode support.

3. Level 3 adoption for full feature parity with web browsers, enabled by proven, stable open source components.

This graduated approach ensures that even conservative vendors can adopt URES without significant risk, leveraging battle-tested implementations developed and maintained by the broader community.

12. IANA Considerations

This document registers a new media feature for the prefers-color-scheme-only media query:

Name: prefers-color-scheme-only

Values: light | dark

Applies to: visual media

Description: Indicates that the email author has requested no automatic colour scheme adaptation.

This document requests registration of the following HTTP header for use in email content negotiation:

Header field name: X-Email-Renderer-Level

Applicable protocol: HTTP (for web-based MUAs)

Status: Informational

Author/Change controller: IETF

Specification document(s): This document

Values: 1 | 2 | 3

Example: X-Email-Renderer-Level: 2

13. References

13.1. Normative References

[CSS-FLEXBOX-1]
Atkins Jr., T., Etemad, E., and R. Atanassov, "CSS Flexible Box Layout Module Level 1", W3C Candidate Recommendation, November 2018, <https://www.w3.org/TR/css-flexbox-1/>.

[CSS-GRID-1]
Atkins Jr., T., Etemad, E., and R. Atanassov, "CSS Grid Layout Module Level 1", W3C Candidate Recommendation, December 2020, <https://www.w3.org/TR/css-grid-1/>.

[HTML5] WHATWG, "HTML Living Standard", <https://html.spec.whatwg.org/>.

[RFC2045] Freed, N. and N. Borenstein, "Multipurpose Internet Mail Extensions (MIME) Part One: Format of Internet Message Bodies", RFC 2045, November 1996, <https://www.rfc-editor.org/info/rfc2045>.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997, <https://www.rfc-editor.org/info/rfc2119>.

[RFC2392] Levinson, E., "Content-ID and Message-ID Uniform Resource Locators", RFC 2392, August 1998, <https://www.rfc-editor.org/info/rfc2392>.

[RFC5322] Resnick, P., Ed., "Internet Message Format", RFC 5322, October 2008, <https://www.rfc-editor.org/info/rfc5322>.

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, May 2017, <https://www.rfc-editor.org/info/rfc8174>.

13.2. Informative References

[CANIEMAIL]
Community project, "Can I email... Support tables for HTML and CSS in emails", <https://www.caniemail.com/>.

[WCAG21] W3C, "Web Content Accessibility Guidelines (WCAG) 2.1", W3C Recommendation, June 2018, <https://www.w3.org/TR/WCAG21/>.

Appendix A. Known Client Inconsistencies

This appendix documents specific inconsistencies in current email clients that this RFC aims to address. Data sourced from caniemail project [CANIEMAIL] and direct testing as of January 2026.

A.1. CSS Support Matrix (Selected Properties)

Property	Gmail	O.com	O.app	Apple	Yahoo	T-bird
style in head	Partial	Yes	Yes	Yes	Yes	Yes
display: flex	Yes	Yes	No	Yes	Yes	Yes
display: grid	Yes	Yes	No	Yes	Yes	Yes
position: fixed	No	No	No	No	No	Yes
position: absolute	No	No	Yes	Yes	No	Yes
background-image	Yes	Yes	No	Yes	Yes	Yes
border-radius	Yes	Yes	Partial	Yes	Yes	Yes
@media queries	No	Yes	Yes	Yes	Yes	Yes
CSS variables	Yes	Yes	No	Yes	Yes	Yes
@font-face	No	Yes	No	Yes	No	Yes
::before/::after	No	Yes	No	Yes	Yes	Yes

Table 1

Legend: Gmail (web), O.com (Outlook.com), O.app (Outlook desktop), Apple (Apple Mail), Yahoo (Yahoo Mail), T-bird (Thunderbird)

Acknowledgements

The author wishes to acknowledge the work of the Email Standards Project (2007-2011), the W3C HTML for Email Community Group (2014-2023), and the caniemail.com project for documenting the inconsistencies that this specification aims to address.

Author’s Address

Will Hackett
London
United Kingdom
Email: will@willhackett.uk
URI: https://willhackett.uk