]>

# Extensible 3D (X3D)
# Part 1: Architecture and base components

# Annex L

# HTML authoring guidelines

## (informative)

## L.1 Introduction and table of contents

This annex describes basic X3D scene integration within an HTML page and provides guidelines for X3D browser implementers when targeting HTML environments [W3C-HTML5]. These recommended practices are intended to encourage use of X3D models compatibly with HTML pages when used in a single Web browser.

Editors NOTE  this draft Annex gives general guidelines for how X3D scenes can be displayed and interactive as part of HTML/DOM Web-page presentations. Current activity by X3D practitioners is focused on implementation and evaluation using the open-source X3DOM and X_ITE players. Further improvements are expected.

Table L.1 lists the major topics in this annex.

**Table L.1 — Topics**

## L.2 X3D implementations in HTML browsers

The Hypertext Markup Language (HTML) is the World Wide Web's core markup language. HTML is a general description language for documents and online applications [W3C-HTML5]. As part of HTML, the Document Object Model (DOM) defines a string-based node tree for representing HTML or XML documents, and also defines event-based processing methods that can be implemented by multiple programming languages [W3C-DOM] [W3C-HTML5].

This document specifies a general-purpose architecture for publishing and sharing 3D graphics models. Since X3D utilizes Web standards and includes an XML file encoding, it is well suited for integration in HTML pages over the World Wide Web.

The X3D design supports multiple goals related to use with HTML/DOM:

- X3D models are specified as a "first-class media type" similar to other image, audio and video formats.
- Event models are harmonized so that DOM events such as user interaction within the HTML page can also affect the X3D model, and vice versa.
- Standards compliance is encouraged wherever possible while also enabling ongoing interaction and interoperability improvements.
- Usage of multiple file encodings is allowed. For example, a url address for an X3D model might point to XML, ClassicVRML, JSON, or another file encoding.

- One or more distinct X3D scenes are allowed to be be loaded at one time within an HTML page.
- HTML page-integration patterns specified for Scalable Vector Graphics (SVG) [W3C-SVG] are consistently followed.

These general guidelines for X3D model display and interaction as part of Web-page presentations allow continuing innovation and adaptability.

# L.3 Page integration

## L.3.1 Content definition and page presentation

There are two basic approaches to loading X3D models within HTML pages:

- Including X3D model elements within an HTML page, or else
- Referencing an external X3D model in an HTML page media element.

For X3D elements appearing within HTML document source, the HTML/DOM `id` attribute can be used for selecting X3D model elements using HTML-page Script nodes (as can any other HTML selector such as CSS). Such node identification is similar to the use of `DEF` when X3D events are passed by a ROUTE connection, allowing disambiguation of specific node instances when needed.

There are also two basic approaches for the visual presentation of X3D models: either

- Reserve a dedicated rectangular canvas (similar to historic EMBED element), or else
- Float above/below other layers of content on the HTML page.

X3D Background *transparency* field allows HTML layer(s) beneath an X3D model to be visible (whether floating or fixed). This capability can support special layouts for page composition.

Each approach is expected to follow [W3C-HTML5] and [W3C-DOM] rules for page composition, and X3D rules for ~~model definition~~ scene presentation.

~~Expected capabilities within an HTML page include compatible interoperation with Scalable Vector Graphics (SVG) content, for example Roy Walmsley's Rosetta Stone demo for X3DOM and Cobweb (X_ITE) (demonstration video).~~

## L.3.2 Syntax, capitalization and validation

HTML has two ~~encoding~~ syntax definitions: HTML ~~encoding~~ syntax (~~lower~~ mixed case, ~~no~~ some restrictions on self-closing tags) and XHTML ~~encoding~~ syntax (CamelCase capitalization, well-formed XML elements).

The ~~naming~~ capitalization of X3D elements and attributes shall match the allowed syntax of HTML where they appear, ~~i.e.~~ matching case conventions for a given HTML or XHTML ~~encoding~~ syntax.

X3D model validation according to schema, DOCTYPE or other mechanisms typically requires strict compliance to upper/lower case name definitions in this specification.

## ~~L.3.3 X3D statement attributes~~

~~Attributes of X3D or X3DCanvas element: url/src, width, height, others... Do we need to decide on url, or src, or either, or no fixed requirement? Current approaches remain far apart.~~

~~X3DOM Configuration lists many candidate attributes of interest for x3d element.~~

~~X_ITE Attributes of the X3DCanvas Tag lists many candidate attributes of interest for x3d element.~~

~~Editorial assessment: this topic is not yet sufficiently stable for guidelines, further implementation and evaluation work is needed.~~

## L.3.3 Cascading Style Sheets (CSS) considerations

Cascading Style Sheets (CSS) is a language for describing the rendering of structured documents (such as HTML and XML) on various display surfaces including screen and paper. CSS is a core language for the World Wide Web that is commonly used for separation of content and presentation, improving layout flexibility.

Design goals for using CSS with X3D include achieving levels of functionality similar to that which exists for using CSS with HTML.

CSS comprises multiple specifications listed in the latest version of CSS Snapshot Recommendation [W3C-CSS-Snapshot]. CSS can be applied and used for Web pages using HTML syntax or XHTML syntax [W3C-HTML5], Scalable Vector Graphics (SVG) [W3C-SVG], and Extensible Markup Language (XML) [W3C-XML] documents, such as models defined using the X3D XML encoding in ISO/IEC 19776-1.

The reserved *class* attribute on each X3D node can provide a space-separated list of classes that pertain from associated stylesheets.

The reserved *style* attribute on each X3D node permits direct definition of style information, rather than referring to styles defined in a separate document [W3C-CSS-Style] [W3C-CSS-Snapshot].

CSS styling of X3D models follows the specific styling of the parent media element on the HTML page.

The CSS *style* attribute is not used to override the FontStyle/ScreenFontStyle *style* field, which is reserved for native X3D style definitions. Instead the parent Text node's CSS *style* attribute can be used to override FontStyle/ScreenFontStyle styling, if appropriate and supported.

~~TODO. Further documentation needed, probably as usage guidelines. XML3D references can provide excellent examples of what is possible.~~

### L.3.4 Cross-Origin Resource Sharing (CORS) considerations

Cross-origin resource sharing (CORS) "is a mechanism that allows restricted resources on a web page to be requested from another domain, outside the domain from which the first resource was served."

"CORS defines a way in which a browser and server can interact to determine whether it is safe to allow the cross-origin request." HTML Web browser limitations may similarly restrict loading remote content (such as an JavaScript X3D browser) from a remote system. Restrictions apply for any combination of locally served content and remotely served content, including local file system.

Details regarding CORS operation by a Web browser are specified in [WHATWG-Fetch].
~~See X3D minutes 24 April 2020: Cross-origin resource sharing (CORS) for additional information and links.~~

## L.4 Event handling, user focus and interaction

### L.4.1 HTML and X3D synchronization

An example approach to HTML X3D Event-Passing Connections is shown in Figure L.1, Example HTML X3D Rendering Synchronization and Event-Passing Connections. This diagram illustrates potential timing and synchronization for rendering of an HTML page and an X3D model, with possibility of event passing between them. It is important to achieve repeatable behavior and efficient rendering with interactive response to user focus.

These relationships are conceptual and not necessarily indicative of any individual implementation, in a manner similar to Figure 4.1 — X3D architecture and Figure 4.3 — Conceptual execution model. Multiple design variations are permitted for HTML-X3D implementations as long as the functional requirements of relevant HTML and X3D specifications are met.

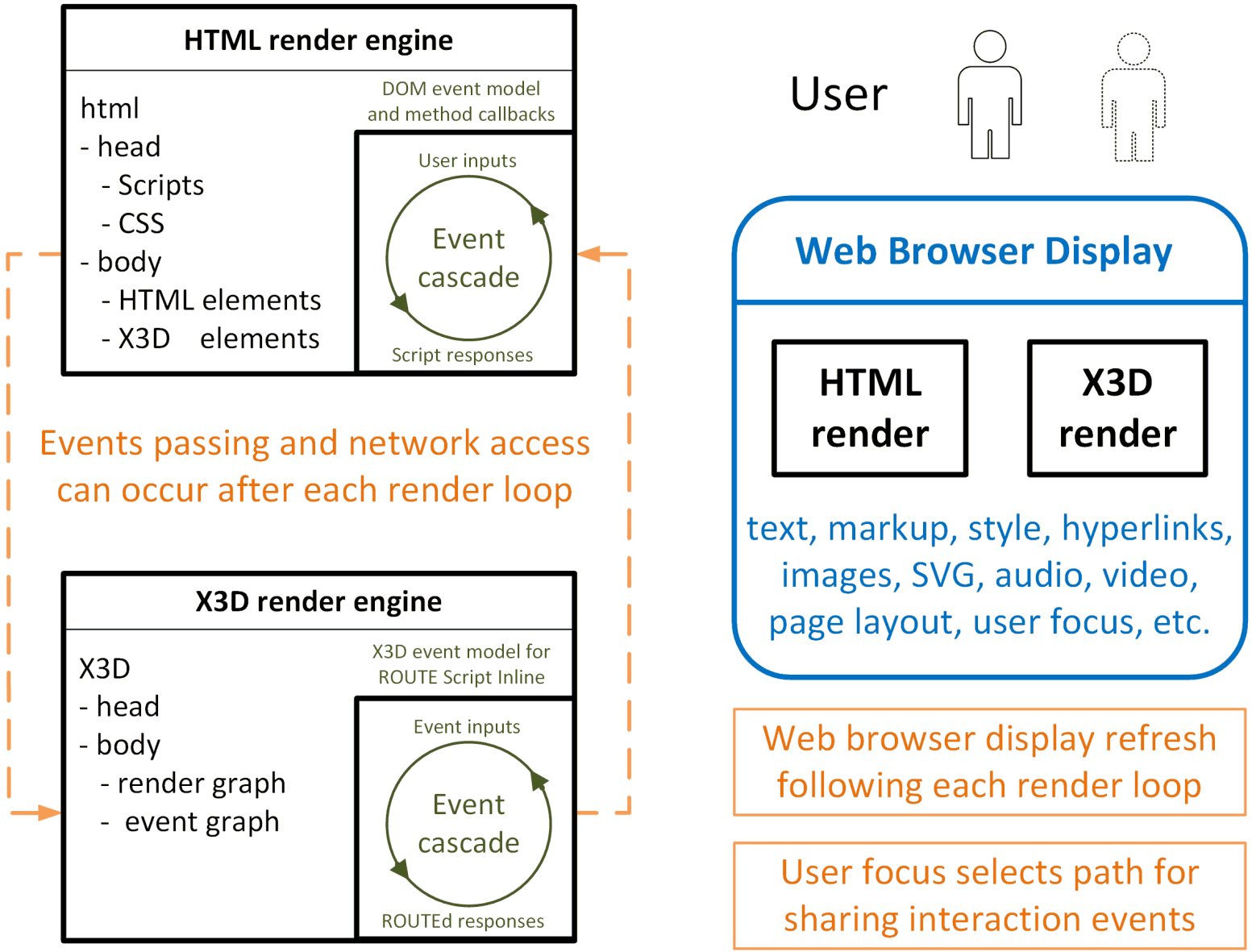


**Figure L.1 — Example HTML X3D Rendering Synchronization and Event-Passing Connections**

Shared HTML events and X3D events (if available) are sent and received at end of each respective render cycle. Once initiated, an X3D event cascade proceeds to completion before any further external events are exchanged (see 4.4.8.3 Execution model). Available X3D events are shared with the HTML engine via mechanisms such as dispatching a custom HTML event, or invoking an author-defined callback.

Note that both HTML and X3D specifications include definitions for a Script node, so disambiguation is necessary. One possible approach is to require that ~~only~~ X3D Script nodes can only appear between X3D elements within an HTML page, and HTML Script nodes are forbidden from appearing between X3D elements. An alternative approach is that any X3D Script node located within HTML page might be renamed X3DScript to avoid any ~~name~~ node naming collisions.

~~When an external HTML environment exists for an X3D model,~~

- ~~TODO. Abstract definition of when events are be exchanged between external environment and scene graph.~~
- ~~Essentially the external presentation event loop must complete each render/interaction cycle before passing events to a contained X3D scene, and~~
- ~~Event loop for a contained X3D scene must complete each render/interaction cycle before passing events to an external presentation.~~

~~Describe how, when an external environment exists,~~

- ~~Abstract definition of how events can be exchanged between external environment and scene graph.~~
- ~~Syntax for multiple encoding/language bindings may be defined in related specifications, e.g. updates to 19777-1 JavaScript, 19776-1 XML Encoding, and (eventually) 19776-5 JSON.~~
- ~~Editors discussion: examples should not go into an annex, will need to go into other file encodings and language bindings.~~
- ~~Pending eventual ISO submission and review of those specifications, we will need example usage and some specification details publicly available to support implementation efforts.~~

## L.4.2 JavaScript/ECMAScript considerations

JavaScript is a programming language that conforms to the ECMAScript specification. The names are often used interchangeably, with ECMAScript indicating strictly specified formal definitions (see ISO/IEC 16262 Information technology — ECMAScript language specification).

Specified ECMAScript Application Programming Interface (API) capabilities for X3D Script node are defined functionally and syntactically in ~~[I19775-2]~~ISO/IEC 19775-2 X3D Scene Authoring Interface (SAI) and ~~[I19776]~~ISO/IEC 19776-1 — X3D ECMAScript encoding, respectively.

Browser implementations and language versions for JavaScript/ECMAScript engines can vary. Since X3D SAI functional requirements are carefully scoped to match the essential capabilities of this core Web programming language, a single JavaScript/ECMAScript engine can typically be used for both HTML and X3D event handling.

Within a Web browser, implementations for HTML and X3D may share a single JavaScript/ECMAScript engine. Such integration is often important for both performance and synchronization issues. This consideration is especially important when considering the demanding response-time requirements of immersive interfaces and spatial body-tracking devices. To aid portability and avoid unintended overloading of variable references, it is good practice for X3D Script authors to avoid the use of variables with global scope.

## L.4.3 User focus considerations

User focus of attention refers to which part of an HTML page is receiving user-directed events such as selection, dragging, and text input. Such capturing of events for callback handling may seem simple but nevertheless can become quite sophisticated, and multiple approaches may be necessary since both displays and device interfaces can vary widely.

X3D provides a consistent selection and navigation interface across a wide range of devices that is based on platform-neutral definitions for model interaction. Suggested device mappings providing consistent user semantics are defined in Annex G Recommended navigation behaviours. Such flexibility is especially important when designing model displays and interactions that support user accessibility.

Web browsers implementing HTML pay close attention to user focus according to [W3C-HTML5] and correspondingly direct user-driven events to appropriate sections of the document graph through the DOM [W3C-DOM]. These are the governing references regarding disposition of events according to user focus prior to receipt by the X3D render engine.

Of interest is that X3D scenes can include Background *transparency* field that may reveal underlying HTML content unobscured by geometry. X3D browsers may optionally pass user-driven events back to the underlying page layers for further HTML/DOM processing.