



HTML

The Definitive Guide

By Chuck Musciano & Bill Kennedy; ISBN 1-56592-492-4, 576 pages.
Third Edition, August 1998.
(See the [catalog page](#) for this book.)

[Search](#) the text of HTML: The Definitive Guide.

Index

[Symbols](#) | [A](#) | [B](#) | [C](#) | [D](#) | [E](#) | [F](#) | [G](#) | [H](#) | [I](#) | [J](#) | [K](#) | [L](#) | [M](#) | [N](#) | [O](#) | [P](#) | [Q](#) | [R](#) | [S](#) | [T](#) | [U](#) | [V](#) | [W](#) | [X](#) | [Y](#) | [Z](#)

Table of Contents

[Preface](#)

[Chapter 1: HTML and the World Wide Web](#)

[Chapter 2: HTML Quick Start](#)

[Chapter 3: Anatomy of an HTML Document](#)

[Chapter 4: Text Basics](#)

[Chapter 5: Rules, Images, and Multimedia](#)

[Chapter 6: Document Layout](#)

[Chapter 7: Links and Webs](#)

[Chapter 8: Formatted Lists](#)

[Chapter 9: Cascading Style Sheets](#)

[Chapter 10: Forms](#)

[Chapter 11: Tables](#)

[Chapter 12: Frames](#)

[Chapter 13: Executable Content](#)

[Chapter 14: Dynamic Documents](#)

[Chapter 15: Tips, Tricks, and Hacks](#)

[Appendix A: HTML Grammar](#)

[Appendix B: HTML Tag Quick Reference](#)

[Appendix C: Cascading Style Sheet Properties Quick Reference](#)

[Appendix D: The HTML 4.0 DTD](#)

[Appendix E: Character Entities](#)

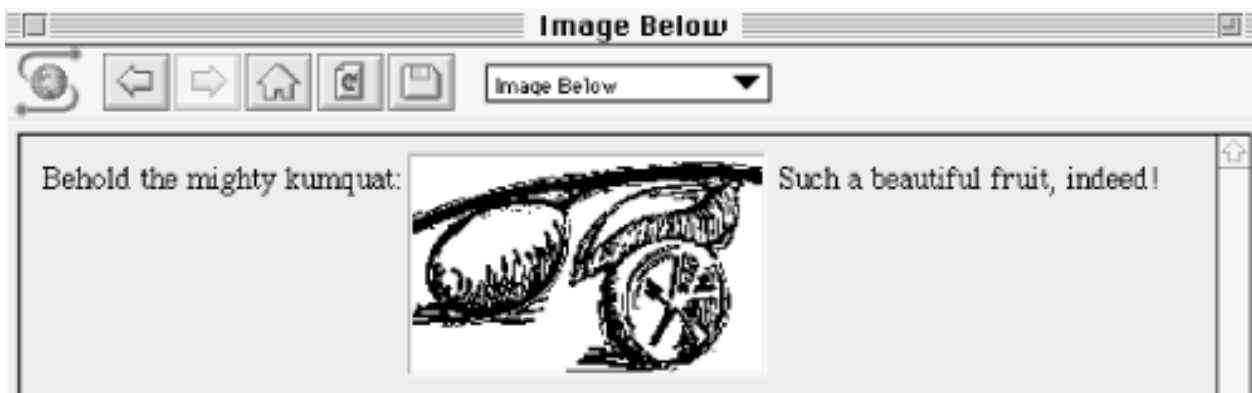
[Appendix F: Color Names and Values](#)



Figure 2.3: An inline image specially aligned with the middle of the text



Figure 2.4: An inline image specially aligned with the top of the text



Experienced HTML authors use images not only as supporting illustrations, but also as quite small inline characters or glyphs, added to aid browsing readers' eyes and to highlight sections of the documents. Veteran HTML authors commonly add custom list bullets or more distinctive section dividers than the conventional horizontal rules. Images, too, may be included in a hyperlink, so that users may select an inline thumbnail sketch to download a full-screen image. The possibilities with inline images are endless.

2.8.2 Image Maps

Image maps are images within an anchor with a special attribute: they may contain more than one hyperlink.

One way to enable an image map is by adding the `ismap` attribute to an `<img>` tag placed inside an

anchor tag (<a>). When the user clicks somewhere in the image, the graphical browser sends the relative x,y coordinates of the mouse position to the server that is also designated in the anchor. A special server program then translates the image coordinates into some special action, such as downloading another HTML document. [\[Server-side considerations , 7.5.1.1\]](#)

A good example of the use of an image map might be to locate a hotel while traveling. The user clicks on a map of the region they intend to visit, for instance, and your image map's server program might return the names, addresses, and phone numbers of local accommodations.

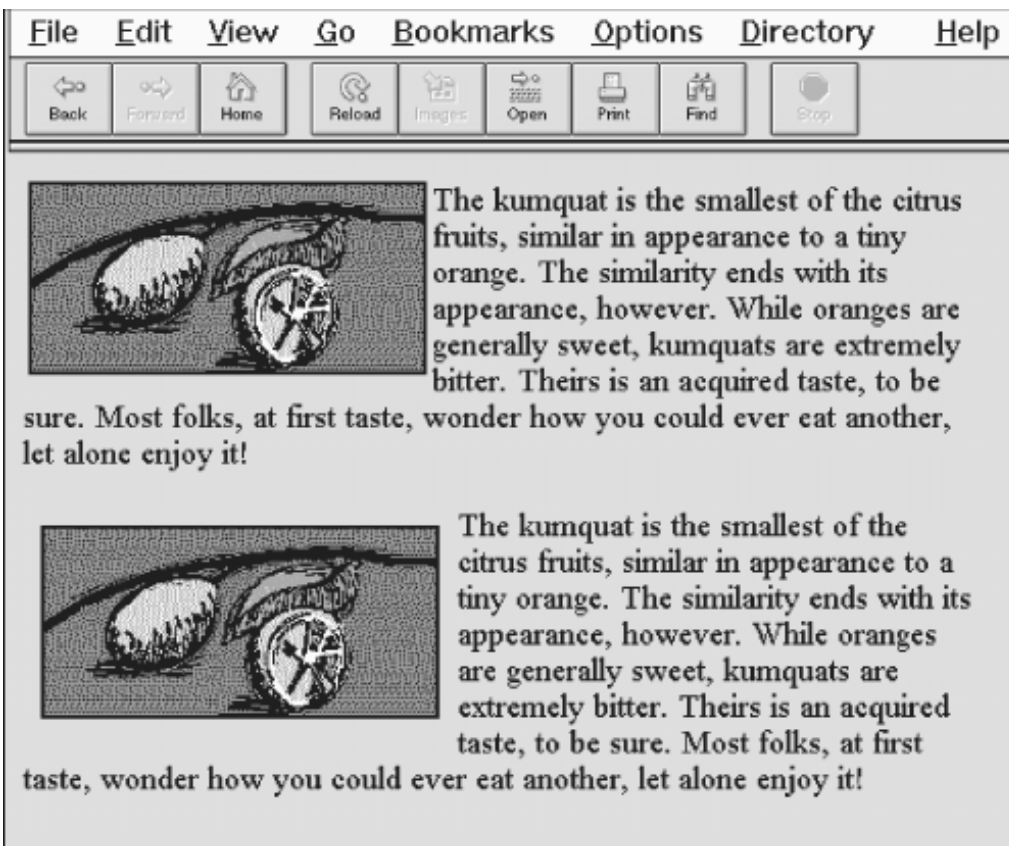
While very powerful and visually appealing, these standard so-called *server-side* image maps mean that HTML authors must have some access to the map's coordinate-processing program on the server. Many authors don't even have access to the server. A better solution is to take advantage of *client-side* image maps.

Rather than depending on a web server, the `usemap` attribute for the `<img>` tag and the `<map>` and `<area>` tags allow HTML authors to embed all the information the browser needs to process an image map in the same document as the image. Because of their reduced network bandwidth and server independence, client-side image maps are becoming increasingly popular among HTML authors. [\[Client-side Image Maps, 7.5.2\]](#)

◀ PREVIOUS
2.7 Hyperlinks

HOME
BOOK INDEX

NEXT ▶
2.9 Lists, Searchable
Documents, and Forms



We're sure you'll agree that the additional space around the image makes the text easier to read and the overall page more attractive.

5.2.6.14 The ismap and usemap attributes

The `ismap` and `usemap` attributes for the `<img>` tag tell the browser that the image is a special mouse-selectable visual map of one or more hyperlinks, commonly known as an *image map*. The `ismap` style of image maps, known as a *server-side* image map, may be specified only within an `<a>` tag hyperlink. [[<a>, 7.3](#)]

For example:

```
<a href="/cgi-bin/images/map2">
  
</a>
```

The browser automatically sends the *x,y* position of the mouse (relative to the upper-left corner of the image) to the server when the user clicks somewhere on the `ismap` image. Special server software (the `/cgi-bin/images/map2` program in the example) may then use those coordinates to determine a response.

The `usemap` attribute provides a *client-side* image map mechanism that effectively eliminates server-side processing of the mouse coordinates and its incumbent network delays and problems. Using special `<map>` and `<area>` tags, HTML authors provide a map of coordinates for the hyperlink-sensitive regions in the `usemap` image along with related hyperlink URLs. The value of the `usemap` attribute is a URL that points to that special `<map>` section. The browser on the user's computer translates the coordinates of a click of the mouse on the image into some action, including loading and displaying another document. [[<map>, 7.5](#)] [[<area>, 7.5](#)]

For example, the following source specially encodes the `100 × 100-pixel map2.gif` image into four segments, each of which, if clicked by the user, links to a different document. Also notice that we've included, validly, the `ismap` image map processing capability in the example `<img>` tag so that users of other, `usemap`-incapable browsers have access to the alternative, server-side mechanism to process the image map:

```
<a href="/cgi-bin/images/map2">
  
</a>
...
```

```
<map name="map2">
  <area coords="0,0,49,49" href="link1.html">
  <area coords="50,0,99,49" href="link2.html">
  <area coords="0,50,49,99" href="link3.html">
  <area coords="50,50,99,99" href="link4.html">
</map>
```

Geographical maps make excellent `ismap` and `usemap` examples: browsing a nationwide company's pages, for instance, the user might click on his or her home town on a map to get the addresses and phone numbers for nearby retail outlets. The advantage of the `usemap` client-side image map processing is that it does not require a server or special server software and so, unlike the `ismap` mechanism, can be used in non-web (networkless) environments, such as local files or CD-ROM.

Please read our more complete discussion of anchors and links, including image maps within links, in section 7.5.

5.2.6.15 The `class`, `dir`, `event`, `id`, `lang`, `style`, and `title` attributes

Several nearly universal attributes give you a common way to identify (`title`) and label (`id`) the image tag's contents for later reference or automated treatment; to change the contents' display characteristics (`class`, `style`); and to reference the language (`lang`) used and related direction the text should flow (`dir`). And, of course, there are all the user events that may happen in and around the tagged contents that the browsers senses and that you may react to via an on-event attribute and some programming. [\[Inline Styles: The style Attribute, 9.1.1\]](#) [\[Style Classes, 9.2.4\]](#)

Of these many HTML 4.0 attributes, `id` is the most important. It lets you label the image for later access by a program or browser operation (see [Chapter 13, Executable Content](#)). [\[The id attribute, 4.1.1.4\]](#)

The remaining attributes have questionable meaning in context with `<img>`. Granted, there are a few style sheet options that may influence an image's display, and a title is good to include, although `alt` is better. And it's hard to imagine what the influence of language (`lang`) or its presentation direction (`dir`) might have on an image. [\[The dir attribute, 3.5.1.1\]](#) [\[The lang attribute, 3.5.1.2\]](#) [\[The title attribute, 4.1.1.5\]](#)

5.2.6.16 The `name`, `onAbort`, `onError`, `onLoad` and other event attributes

There are four `<img>` attributes currently supported by Netscape that enables you to use JavaScript to manipulate the image. The first is the `name` attribute. Now redundant with the HTML 4.0 standard `id` attribute, `name` lets you label the image so that it can be referenced by a JavaScript applet. For example:

```

```

lets you later refer to that picture of a kumquat as simply "kumquat" in a JavaScript applet perhaps to erase or otherwise modify it. You cannot individually manipulate an image with JavaScript if it is not named or doesn't have an associated `id`.

The other three attributes let you provide some special JavaScript event handlers. The value of the attribute is a chunk of JavaScript code, enclosed in quotation marks; it may consist of one or more JavaScript expressions, separated by semicolons.

Netscape invokes the `onAbort` event handler if the user stops loading an image, usually by clicking the browser's "stop" button. You might, for instance, use an `onAbort` message to warn users if they stop loading some essential image, such as an image map (see section 6.5):

```

```

The `onError` attribute is invoked if some error occurs during the loading of the image, but not for a missing image or one that the user chose to stop loading. Presumably, the applet could attempt to recover from the error or load a different image in its place.

Netscape executes the JavaScript code associated with the `<img>` tag's `onLoad` attribute right after the browser successfully loads and displays the image.

See section 13.3.3 for more information about JavaScript and event handlers.

5.2.6.17 Combining `<img>` attributes

You may combine any of the various standard and extension attributes for images where and when they make sense. The order
DataDome Ex. 1019, p. 6
DataDome v. Arkose Labs
IPR2025-00694

7.5 Mouse-Sensitive Images

Normally, an image placed within an anchor simply becomes part of the anchor content. The browser may alter the image in some special way (usually with a special border) to alert the reader that it is a link, but the user clicks the image in the same way they click a textual link.

The HTML standard provides a feature that lets you embed many different links inside the same image. Clicking different areas of the image causes the browser to link to different target documents. Such mouse-sensitive images, known as *image maps*, open up a variety of creative linking styles.

There are two ways to create image maps, known as *server-side* and *client-side* image maps. The former, enabled by the `ismap` attribute for the `<img>` tag, requires access to a server and related image map processing applications. The latter is created with the `usemap` attribute for the `<img>` tag, along with corresponding `<map>` and `<area>` tags. Since translation of the mouse position in the image to a link to another document happens on the user's machine, client-side image maps don't require a special server connection, and even can be implemented in non-web environments, such as on a local hard drive or on a CD-ROM-based document collection. [\[<map>, 7.5\]](#) [\[<area>, 7.5\]](#) [\[The Tag, 5.2.6\]](#)

7.5.1 Server-side Image Maps

You add an image to an anchor simply by placing an `<img>` tag within the body of the `<a>` tag. Make that embedded image into a mouse-sensitive one by adding the `ismap` attribute to the `<img>` tag. This special `<img>` attribute tells the browser that the image is a special map containing more than one link. (The `ismap` attribute is ignored by the browser if the `<img>` tag is not within an `<a>` tag.) [\[The Tag, 5.2.6\]](#)

When the user clicks some place within the image, the browser passes the coordinates of the mouse pointer along with the URL specified in the `<a>` tag to the document server. The server uses the mouse pointer coordinates to determine which document to deliver back to the browser.

When `ismap` is used, the `href` attribute of the containing `<a>` tag must contain the URL of a server application or, for some HTTP servers, a related map file that contains the coordinate and linking information. If the URL is simply that of a conventional document, errors may result and the desired document will most likely not be retrieved.

The coordinates of the mouse position are screen pixels counted from the upper-left corner of the image beginning with (0,0). The coordinates are added to the end of the URL, preceded by a question mark.

For example, if a user clicks 43 pixels over and 15 pixels down from the upper-left corner of the image displayed from the following link:

```
<a href="/cgi-bin/imagemap/toolbar.map">

</a>
```

the browser sends the following search parameters to the HTTP server:

/cgi-bin/imagemap/toolbar.map?43,15

In the example, *toolbar.map* is a special image map file inside the *cgi-bin/imagemap* directory and containing coordinates and links. A special image map process uses that file to match the passed coordinates (43,15 in the example) and return with the selected hyperlink document.

7.5.1.1 Server-side considerations

With mouse-sensitive *ismap*-enabled image maps, the browser is required to pass along only the URL and mouse coordinates to the server. Converting these coordinates into a specific document is handled by the document server. The conversion process differs between servers and is not defined by the HTML standard.

You need to consult with your web server administrators and perhaps even read your server's documentation to determine how to create and program an image map. Most servers come with some software utility, typically located in a *cgi-bin/imagemap* directory, to handle image maps. And most of these use a text file containing the image map regions and related hyperlinks that is referenced by your image map URL to process the image map query.

Here's an example image map file that describes the sensitive regions in our example image:

```
# Imagemap file=toolbar.map

default                dflt.html
circ 100,30,50          link1.html
rect 180,120,290,500    link2.html
poly 80,80,90,72,160,90 link3.html
```

Each sensitive region of the image map is described by a geometric shape and defining coordinates in pixels, such as the circle with its center point and radius, the rectangle's upper-left and lower-right edge coordinates, and the loci of a polygon. All coordinates are relative to the upper-left corner of the image (0,0). Each shape has a related URL.

An image map processing application typically tests each shape in the order it appears in the image file and returns the document specified by the corresponding URL to the browser if the user's mouse x,y coordinates fall within the boundaries of that shape. That means it's okay to overlap shapes; just be aware which takes precedence. Also, the entire image need not be covered with sensitive regions: if the passed coordinates don't fall within a specified shape, the default document gets sent back to the browser.

This is just one example for how an image map may be processed and the accessory files required for that process. Please huddle with your webmaster and server manuals to discover how to implement a server-side image map for your HTML documents and system.

7.5.2 Client-side Image Maps

The obvious downside to server-side image maps is that they require a server. That means you need access to the required HTTP server or its */cgi-bin/* directory, which isn't always available. And server-side image maps limit portability, since not all image map processing applications are the same.

Server-side image maps also mean delays for the user while browsing, since the browser must get the server's attention to process the image coordinates. That's even if there's no action to take, such as a section of the image that isn't hyperlinked and doesn't lead anywhere.

Client-side image maps suffer from none of these difficulties. Enabled by the *usemap* attribute for the `<img>` tag, and defined by special `<map>` and `<area>` extension tags, client-side image maps let HTML authors include in their documents a map of coordinates and links that describe the sensitive regions of an image. The browser on the client computer translates the coordinates of the mouse position within the image into an action,



D. The HTML 4.0 DTD

The HTML 4.0 standard is formally defined as three SGML Document Type Definitions (DTDs): the Strict DTD, the Transitional DTD, and the Frameset DTD. The Strict DTD defines only those elements that are not deprecated in the 4.0 standard. Ideally, everyone would create HTML documents that conform to the Strict DTD. The Transitional DTD includes all those deprecated elements and more accurately reflects the HTML in use today, with many older elements still in common use. The Frameset DTD is identical to the Transitional DTD, with the exception that the document `<body>` is replaced by the `<frameset>` tag.

Since the Transitional DTD provides the broadest coverage of all HTML elements currently in use, it is the DTD upon which this book is based and the one we reproduce here. Note that we have reprinted this DTD verbatim and have not attempted to add extensions to it. Where our description and the DTD deviate, assume the DTD is correct:

```
<!--
```

```
This is the HTML 4.0 Transitional DTD, which includes
presentation attributes and elements that W3C expects to phase out
as support for style sheets matures. Authors should use the Strict
DTD when possible, but may use the Transitional DTD when support
for presentation attribute and elements is required.
```

```
HTML 4.0 includes mechanisms for style sheets, scripting,
embedding objects, improved support for right to left and mixed
direction text, and enhancements to forms for improved
accessibility for people with disabilities.
```

```
Draft: $Date: 2000/02/11 18:30:23 $
```

```
Authors:
```

```
Dave Raggett <dsr@w3.org>
```

```
Arnaud Le Hors <lehors@w3.org>
```

```
Ian Jacobs <ij@w3.org>
```

```
Further information about HTML 4.0 is available at:
```

```
http://www.w3.org/TR/REC-html40
```

```
-->
```

```
<!ENTITY % HTML.Version "-//W3C//DTD HTML 4.0 Transitional//EN"
```

```
-- Typical usage:
```

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN"
"http://www.w3.org/TR/REC-html40/loose.dtd">
```

```
<html>
```

```
<head>
```

```
...
```

```
</head>
```

```
<body>
```

```
...
```

```
</body>
```

```
</html>
```

The URI used as a system identifier with the public identifier allows the user agent to download the DTD and entity sets as needed.

The FPI for the Strict HTML 4.0 DTD is:

```
"//W3C//DTD HTML 4.0//EN"
```

and its URI is:

```
http://www.w3.org/TR/REC-html40/strict.dtd
```

Authors should use the Strict DTD unless they need the

```

>
<!ELEMENT AREA - O EMPTY -- client-side image map area -->
<!ATTLIST AREA
  %attrs; -- %coreattrs, %il8n, %events --
  shape %Shape; rect -- controls interpretation of coords --
  coords %Coords; #IMPLIED -- comma separated list of lengths --
  href %URI; #IMPLIED -- URI for linked resource --
  target %FrameTarget; #IMPLIED -- render in this frame --
  nohref (nohref) #IMPLIED -- this region has no action --
  alt %Text; #REQUIRED -- short description --
  tabindex NUMBER #IMPLIED -- position in tabbing order --
  accesskey %Character; #IMPLIED -- accessibility key character --
  onfocus %Script; #IMPLIED -- the element got the focus --
  onblur %Script; #IMPLIED -- the element lost the focus --
>
<!--===== The LINK Element =====-->
<!--
Relationship values can be used in principle:
a) for document specific toolbars/menus when used
with the LINK element in document head e.g.
start, contents, previous, next, index, end, help
b) to link to a separate style sheet (rel=stylesheet)
c) to make a link to a script (rel=script)
d) by stylesheets to control how collections of
html nodes are rendered into printed documents
e) to make a link to a printable version of this document
e.g. a postscript or pdf version (rel=alternate media=print)
-->
<!ELEMENT LINK - O EMPTY -- a media-independent link -->
<!ATTLIST LINK
  %attrs; -- %coreattrs, %il8n, %events --
  charset %Charset; #IMPLIED -- char encoding of linked resource --
  href %URI; #IMPLIED -- URI for linked resource --
  hreflang %LanguageCode; #IMPLIED -- language code --
  type %ContentType; #IMPLIED -- advisory content type --
  rel %LinkTypes; #IMPLIED -- forward link types --
  rev %LinkTypes; #IMPLIED -- reverse link types --
  media %MediaDesc; #IMPLIED -- for rendering on these media --
  target %FrameTarget; #IMPLIED -- render in this frame --
>
<!--===== Images =====-->
<!-- Length defined in strict DTD for cellpadding/cellspacing -->
<!ENTITY % Length "CDATA" -- nn for pixels or nn% for percentage length -->
<!ENTITY % MultiLength "CDATA" -- pixel, percentage, or relative -->
<!ENTITY % MultiLengths "CDATA" -- comma-separated list of MultiLength -->
<!ENTITY % Pixels "CDATA" -- integer representing length in pixels -->
<!ENTITY % IAlign "(top|middle|bottom|left|right)" -- center? -->
<!-- To avoid problems with text-only UAs as well as
to make image content understandable and navigable
to users of non-visual UAs, you need to provide
a description with ALT, and avoid server-side image maps -->
<!ELEMENT IMG - O EMPTY -- Embedded image -->
<!ATTLIST IMG
  %attrs; -- %coreattrs, %il8n, %events --
  src %URI; #REQUIRED -- URI of image to embed --
  alt %Text; #REQUIRED -- short description --
  longdesc %URI; #IMPLIED -- link to long description
-- (complements alt) --
  height %Length; #IMPLIED -- override height --
  width %Length; #IMPLIED -- override width --
  usemap %URI; #IMPLIED -- use client-side image map

```

```

    ismap          (ismap)          #IMPLIED -- use server-side image map --
    align          %IAAlign;        #IMPLIED -- vertical or horizontal alignment --
    border         %Length;         #IMPLIED -- link border width --
    hspace         %Pixels;         #IMPLIED -- horizontal gutter --
    vspace         %Pixels;         #IMPLIED -- vertical gutter --
>
<!-- USEMAP points to a MAP element which may be in this document
or an external document, although the latter is not widely supported -->
<!--===== OBJECT =====>
<!--
    OBJECT is used to embed objects as part of HTML pages
    PARAM elements should precede other content. SGML mixed content
    model technicality precludes specifying this formally ...
-->
<!ELEMENT OBJECT - - (PARAM | %flow;)*
-- generic embedded object -->
<!ATTLIST OBJECT
    %attrs;                                -- %coreattrs, %il8n, %events --
    declare      (declare)                #IMPLIED -- declare but don't instantiate flag --
    classid      %URI;                    #IMPLIED -- identifies an implementation --
    codebase     %URI;                    #IMPLIED -- base URI for classid, data, archive--
    data         %URI;                    #IMPLIED -- reference to object's data --
    type         %ContentType;            #IMPLIED -- content type for data --
    codetype     %ContentType;            #IMPLIED -- content type for code --
    archive      %URI;                    #IMPLIED -- space separated archive list --
    standby      %Text;                   #IMPLIED -- message to show while loading --
    height       %Length;                 #IMPLIED -- override height --
    width        %Length;                 #IMPLIED -- override width --
    usemap       %URI;                    #IMPLIED -- use client-side image map --
    name         CDATA                    #IMPLIED -- submit as part of form --
    tabindex     NUMBER                   #IMPLIED -- position in tabbing order --
    align        %IAAlign;                #IMPLIED -- vertical or horizontal alignment --
    border       %Length;                 #IMPLIED -- link border width --
    hspace       %Pixels;                 #IMPLIED -- horizontal gutter --
    vspace       %Pixels;                 #IMPLIED -- vertical gutter --
    %reserved;                                -- reserved for possible future use --
>
<!ELEMENT PARAM - O EMPTY                -- named property value -->
<!ATTLIST PARAM
    id          ID                      #IMPLIED -- document-wide unique id --
    name        CDATA                    #REQUIRED -- property name --
    value       CDATA                    #IMPLIED -- property value --
    valuetype   (DATA|REF|OBJECT) DATA  -- How to interpret value --
    type        %ContentType;            #IMPLIED -- content type for value
                                                when valuetype=ref --
>
<!--===== Java APPLET =====>
<!--
    One of code or object attributes must be present.
    Place PARAM elements before other content.
-->
<!ELEMENT APPLET - - (PARAM | %flow;)* -- Java applet -->
<!ATTLIST APPLET
    %coreattrs;                                -- id, class, style, title --
    codebase    %URI;                        #IMPLIED -- optional base URI for applet --
    archive     CDATA                        #IMPLIED -- comma separated archive list --
    code        CDATA                        #IMPLIED -- applet class file --
    object      CDATA                        #IMPLIED -- serialized applet file --
    alt         %Text;                       #IMPLIED -- short description --
    name        CDATA                        #IMPLIED -- allows applets to find each other --
    width       %Length;                     #REQUIRED -- initial width --

```