13 Объекты, изображения и апплеты

13.1 Введение в объекты, изображения и апплеты

Функции мультимедиа языка HTML позволяют авторам включать в свои страницы изображения, апплеты (программы, которые автоматически загружаются и выполняются на машине пользователя), видеоклипы и другие документы в формате HTML. Например, чтобы включить в документ изображение в формате PNG, можно использовать:

```
<BODY>
<P>Около Большого Каньона:
<OBJECT data="canyon.png" type="image/png">
<EM>Около</EM> Большого Каньона.
</OBJECT>
</BODY>
```

В предыдущих версиях HTML авторы могли включать изображения (с помощью IMG) и апплеты (с помощью APPLET). Эти элементы имеют несколько ограничений:

- Они не могут решить более общей проблемы - включение новых и возможных в будущем типов устройств.
- Элемент APPLET работает только с апплетами языка Java. Этот элемент теперь нежелателен, вместо него используется элемент OBJECT.
- Они вызывают проблемы доступности.

Для решения всех этих вопросов в HTML 4.0 вводится элемент OBJECT, обеспечивающий всестороннее решение для включения объектов. Элемент OBJECT позволяет авторам документов в формате HTML указывать всю информацию, необходимую для представления объекта агентом пользователя: исходный код, начальные значения и рабочие данные. В данной спецификации термин "объект" используется для описания всех объектов, которые Вы захотите включить в HTML-документы; другие термины: апплеты, подключаемые модули (plug-ins), дескрипторы устройств и т.д.

Новый элемент OBJECT, таким образом, subsumes некоторые задачи, выполняемые существующими элементами. Рассмотрим следующую классификацию функций:

Тип включения	Конкретный элемент	Общий элемент
Изображение	IMG	OBJECT
Апплет	APPLET (**Нежелателен.**)	OBJECT
Другой документ HTML	IFRAME	OBJECT

Из таблицы видно, что каждый тип включения имеет конкретное и общее решение. Общий элемент OBJECT служит решением для использования возможных в будущем типов устройств. Для включения изображений авторы могут использовать элемент OBJECT или элемент IMG. Для включения апплетов авторам следует использовать элемент OBJECT, поскольку использование элемента APPLET нежелательно.

Для включения одного документа HTML в другой авторы могут использовать новый элемент IFRAME или элемент OBJECT. В обоих случаях внедренный документ не зависит от основного документа. Визуальные агенты пользователей могут представлять внедренный документ в виде отдельного окна в основном документе. Для сравнения элементов OBJECT и IFRAME обратитесь к замечаниям о внедряемых документов.

С изображениями и другими включаемыми объектами могут быть связаны ссылки, с помощью стандартных механизмов ссылок, а также и с помощью *навигационных карт (image maps)*. На навигационной карте задаются геометрические области включаемого объекта, и каждой из них назначается ссылка. При активизации эти ссылки могут вызывать загрузку документа, запускать программу на сервере и т.д.

В следующих разделах мы обсудим различные механизмы, которые авторы могут использовать для включения мультимедиа и создания навигационных карт для этих

объектов.

13.2 Включение изображения: элемент IMG

```
<!-Во избежание проблем с текстовыми агентами пользователей,а также для того, чтобы
включение изображений было понятно пользователям невизуальных агентов и могло
использоваться ими,Вы должны указывать описания в элементе ALT и избегать
использования
   Серверных навигационных карт-->
<!ELEMENT IMG - O EMPTY                -- Внедренное изображение -->
<!ATTLIST IMG
  %attrs;                              -- %coreattrs, %i18n, %events --
  src          %URI;           #REQUIRED -- URI внедряемого изображения --
  alt          %Text;          #REQUIRED -- краткое описание --
  longdesc     %URI;           #IMPLIED  -- ссылка на длинное описание
                                            (дополняет alt) --
  height       %Length;        #IMPLIED  -- переопределение высоты --
  width        %Length;        #IMPLIED  -- переопределение ширины --
  usemap       %URI;           #IMPLIED  -- использовать клиентскую
                                            навигационную карту --
  ismap        (ismap)         #IMPLIED  -- использовать серверную
                                            навигационную карту --
  >
```

Начальный тег: ***обязателен****, Конечный тег:* ***запрещен***

Определения атрибутов

`src` = *uri* [CT]

Этот атрибут задает местоположение изображения. Примерами широко распознаваемых форматов являются GIF, JPEG и PNG.

`longdesc` = *uri*[CT]

Этот атрибут определяет ссылку на длинное описание изображения. Это описание должно дополнять краткое описание, задаваемое атрибутом `alt`. Если с изображением связана навигационная карта, в этом атрибуте должна приводиться информация о ее содержимом. Это особенно важно для серверных навигационных карт.

Атрибуты, определенные в другом месте

- `id`, `class` (идентификаторы в пределах документа)
- `alt` (альтернативный текст)
- `lang` (информация о языке), `dir` (направление текста)
- `title` (заголовок элемента)
- `style` (встроенная информация о стиле)
- `onclick`, `ondblclick`, `onmousedown`, `onmouseup`, `onmouseover`, `onmousemove`, `onmouseout`, `onkeypress`, `onkeydown`, `onkeyup` (внутренние события)
- `ismap`, `usemap` (клиентская навигационная карта)
- `align`, `width`, `height`, `border`, `hspace`, `vspace` (визуальное представление объектов, изображений и апплетов)

Элемент `IMG` внедряет изображение в текущий документ по адресу из определения элемента. Элемент `IMG` не имеет содержимого; обычно он замещается изображением, назначаемым атрибутом `src`, исключение при этом составляют выровненные влево или вправо изображения, которые "floated" out of line.

В приведенном ранее примере мы определили ссылку на семейную фотографию. Здесь мы вставим фотографию непосредственно в текущий документ:

```
<BODY>
<P>Я только что вернулся из отпуска! Вот фотография моей семьи на озере:
<IMG src="http://www.somecompany.com/People/Ian/vacation/family.png"
     alt="Фотография моей семьи на озере.">
</BODY>
```

Этого же эффекта можно достичь с помощью элемента `OBJECT` следующим образом:

```
<BODY>
```

```
<P>Я только что вернулся из отпуска! Вот фотография моей семьи на озере:
<OBJECT data="http://www.somecompany.com/People/Ian/vacation/family.png"
        type="image/png">
Фотография моей семьи на озере.
</OBJECT>
</BODY>
```

Атрибут alt задает альтернативный текст, который генерируется, если изображение невозможно отобразить (информацию о том, как указать альтернативный текст, см. ниже). Агенты пользователей должны генерировать альтернативный текст, если они не поддерживают изображение, если они не поддерживают определенный тип изображений или если они сконфигурированы так, чтобы не выводить изображений.
В следующем примере показано, как можно использовать атрибут longdesc для ссылки на более подробное описание:

```
<BODY>
<P>
<IMG src="sitemap.gif"
     alt="Карта узла лабораторий HP"
     longdesc="sitemap.html">
</BODY>
```

Атрибут alt задает краткое описание изображения. Его должно быть достаточно для того, чтобы пользователи могли решить, хотят ли они следовать по ссылке, определяемой атрибутом longdesc для более получения подробного описания, здесь это ссылка "sitemap.html".
Информацию о размере изображения, выравнивании и границах см. в разделе о визуальном представлении объектов, изображений и апплетов.

13.3 Общее включение: элемент OBJECT

```
<!ELEMENT OBJECT - - (PARAM | %flow;)*
 -- общий внедренный объект -->
<!ATTLIST OBJECT
  %attrs;                                -- %coreattrs, %i18n, %events --
  declare     (declare)      #IMPLIED  -- объявить, но не instantiate флаг --
  classid     %URI;          #IMPLIED  -- определяет применение --
  codebase    %URI;          #IMPLIED  -- базовый URI для classid, data, archive--
  data        %URI;          #IMPLIED  -- ссылка на данные объекта --
  type        %ContentType;  #IMPLIED  -- тип содержимого для данных --
  codetype    %ContentType;  #IMPLIED  -- тип содержимого для кода --
  archive     %URI;          #IMPLIED  -- разделенный пробелами список архивов --
  standby     %Text;         #IMPLIED  -- сообщение, отображаемое при загрузке --
  height      %Length;       #IMPLIED  -- переопределение высоты --
  width       %Length;       #IMPLIED -- переопределение ширины --
  usemap      %URI;          #IMPLIED -- использовать клиентскую навигационную карту--
  name        CDATA          #IMPLIED -- представить в качестве части формы --
  tabindex    NUMBER         #IMPLIED -- положение в последовательности перехода --
  >
```

Начальный тег: ***обязателен****, Конечный тег:* ***обязателен***
Определения атрибутов
classid = *uri* [CT]
Этот атрибут может использоваться для указания местоположения объекта с помощью URI. Он может использоваться вместе с атрибутом data или как альтернатива ему, в зависимости от типа объекта.
codebase = *uri* [CT]
Этот атрибут определяет базовый путь, используемый для разрешения относительных адресов URI, задаваемых в атрибутах classid, data и archive. Если этот атрибут отсутствует, значением по умолчанию является базовый адрес URI текущего документа.
codetype = *content-type* [CI]
Этот атрибут определяет тип содержимого данных, получения которых следует ожидать

при загрузке объекта, задаваемого атрибутом classid. Этот атрибут не является обязательным, но рекомендуется, если используется атрибут classid, поскольку он позволяет агенту пользователя избежать загрузки информации для типа содержимого, который он не поддерживает. Если этот атрибут отсутствует, по умолчанию используется значение атрибута type.

data = *uri* [CT]

Этот атрибут может использоваться для указания местоположения данных объекта, например, данных изображения для объектов, определяющих изображения, или в более общем случае - serialized формы объекта, который может использоваться для повторного его создания. Если дается относительный адрес URI, он должен интерпретироваться относительно атрибута codebase.

type = *content-type* [CI]

Этот атрибут определяет тип содержимого для данных, задаваемых атрибутом data. Этот атрибут не является обязательным, но рекомендуется, если используется атрибут data, поскольку он позволяет агенту пользователя избежать загрузки информации для типа содержимого, который они не поддерживают.

archive = *uri list* [CT]

Этот атрибут может использоваться для определения разделенного пробелами списка адресов URI архивов, содержащих относящиеся к объекту ресурсы, который может включать ресурсы, задаваемые атрибутами classid и data. Предварительная загрузка архивов приведет к уменьшению времени загрузки объекта. Архивы, указанные в виде относительных адресов URI, должны интерпретироваться относительно атрибута codebase.

declare [CI]

Если этот логический атрибут указан, он делает текущее определение OBJECT только объявлением. Объект должен быть instantiated последующим определением OBJECT, ссылающимся на это объявление.

standby = *text* [CS]

Этот атрибут определяет сообщение, которое агент пользователя может генерировать при загрузке implementation и данных объекта.

Атрибуты, определенные в другом месте

- id, class (идентификаторы в пределах документа)
- lang (информация о языке), dir (направление текста)
- title (заголовок элемента)
- style (встроенная информация о стиле)
- onclick, ondblclick, onmousedown, onmouseup, onmouseover, onmousemove, onmouseout, onkeypress, onkeydown, onkeyup (внутренние события)
- tabindex (переход по клавише tab)
- usemap (клиентские навигационные карты)
- name (предоставление формы)
- align, width, height, border, hspace, vspace (визуальное представление объектов, изображений и апплетов)

В большинстве агентов пользователей имеются встроенные механизмы для генерации основных типов данных, таких как текст, изображения в формате GIF, цвета, шрифты и ряд графических элементов. Для генерации типов данных, которые агенты пользователей не поддерживают по умолчанию, они обычно запускают внешние приложения. Элемент OBJECT позволяет авторам управлять генерацией данных - задавать внешнюю генерацию или использование некоторой определяемой автором программы, генерирующей данные в агенте пользователя.

В более общем случае автор должен будет определить три типа информации:

- Реализация включенного объекта. Например, если включенный объект - апплет, автор должен указать местоположение исполняемого кода апплета.
- Генерируемые данные. Например, если включенный объект является программой, генерирующей данные шрифта, автор должен указать местоположение этих данных.
- Дополнительные значения, необходимые объекту. Например, некоторым апплетам могут быть нужны исходные значения для их параметров.

Элемент OBJECT позволяет авторам указать все три типа данных объекта, но авторы не обязательно должны указывать их все. Например, некоторым объектам не требуются данные (например, апплет, выполняющий анимацию). Другим может быть не нужна инициализация. Другим же может не понадобиться дополнительная информация о реализации, то есть сам агент пользователя может уже знать, как генерировать этот тип данных (например, изображения в формате GIF).

Авторы задают реализацию объекта и местоположение данных, генерируемых с помощью элемента OBJECT. Однако для указания рабочих значений авторы используют элемент PARAM, обсуждаемый в разделе об инициализации объекта.

Элемент OBJECT может также присутствовать внутри элемента HEAD. Поскольку агенты пользователей обычно не генерируют элементы в HEAD, авторам следует убедиться, что во всех элементах OBJECT в HEAD нет содержимого, которое можно генерировать.

Пример включения элемента OBJECT в элемент HEAD см. в разделе о совместном использовании данных кадра.

Информацию об элементе OBJECT в формах см. в разделе об управлении формами.

13.3.1 Правила генерации объектов

Агент пользователя должен интерпретировать элемент OBJECT в соответствии со следующими правилами старшинства:

1. Сначала агент пользователя должен попытаться сгенерировать объект. Он не должен генерировать содержимое элемента, но должен проверить его на случай, если элемент содержит дополнительные дочерние элементы PARAM (см. инициализация объекта) или элементы MAP (см. клиентские навигационные карты).
2. Если агент пользователя по какой-либо причине не может сгенерировать объект (не сконфигурирован для этого, недостаточно ресурсов, ошибочная архитектура и т.д.), он должен попытаться сгенерировать его содержимое.

Авторам не следует включать содержимое в элементы OBJECT, расположенные в элементе HEAD.

В следующем примере мы вставляем в документ апплет, представляющий часы, с помощью элемента OBJECT. Апплету, написанный на языке Python, не нужны дополнительные и рабочие значения. Атрибут classid определяет местоположение апплета:

```
<P><OBJECT classid="http://www.miamachina.it/analogclock.py">
</OBJECT>
```

Обратите внимание на то, что часы будут генерироваться, как только агент пользователя интерпретирует это объявление OBJECT. Можно отложить генерацию объекта, начав с *объявления* объекта (описывается ниже).

Авторы должны выполнять объявление, включая альтернативный текст в качестве содержимого элемента OBJECT в случае, если агент пользователя не может сгенерировать часы.

```
<P><OBJECT classid="http://www.miamachina.it/analogclock.py">
Часы с анимацией.
```

```
</OBJECT>
```

Одним важным последствием создания элемента OBJECT является то, что он предлагает механизм задания альтернативной генерации объектов; в каждом внедренном объявлении OBJECT могут задаваться альтернативные типы содержимого. Если агент пользователя не может сгенерировать outermost OBJECT, он пытается сгенерировать содержимое, которое может быть другим элементом OBJECT и т.д.

В следующем примере мы внедряем несколько объявлений OBJECT для того, чтобы показать работу альтернативной генерации. Агент пользователя попытается сгенерировать первый элемент OBJECT, который он может, в следующем порядке: (1) апплет Earth, написанный на языке Python, (2) клип Земли в формате MPEG, (3) изображение Земли в формате GIF, (4) альтернативный текст.

```
<P>                 <!- Сначала попробовать апплет на языке Python -->
<OBJECT title="Вид Земли из космоса"
        classid="http://www.observer.mars/TheEarth.py">
                    <!- Затем попробовать воспроизвести видеоклип
                              в формате MPEG -->
  <OBJECT data="TheEarth.mpeg" type="application/mpeg">
                    <!- Затем попробовать изображение в формате GIF -->
    <OBJECT data="TheEarth.gif" type="image/gif">
                    <!- Затем сгенерировать текст -->
     Вид <STRONG>Земли</STRONG> из космоса.
    </OBJECT>
  </OBJECT>
</OBJECT>
```

Внешнее объявление определяет апплет, которому не нужны данные или начальные значения. Второе объявление определяет клип в формате MPEG и, поскольку местоположение обработчика формата MPEG не указано, предполагается, что клип будет обрабатываться агентом пользователя. Мы также установили атрибут `type`, так что агент пользователя, который знает, что он не может сгенерировать клип в формате MPEG, не будет загружать файл "TheEarth.mpeg" из сети. В третьем объявлении задается местоположение файла в формате GIF и определяется альтернативный текст на случай, если все прочие механизмы не сработают.

Встроенные и внешние данные. *Данные, которые должны генерироваться, могут указываться двумя способами: в виде встроенного или внешнего ресурса. Последний метод обычно обеспечивает более быструю генерацию, но неудобен при генерации большого объема данных.*

Ниже приводится пример, показывающий, как встроенные данные могут be fed to an OBJECT:

```
<P>
<OBJECT id="clock1"
        classid="clsid:663C8FEF-1EF9-11CF-A3DB-080036F12502"
        data="data:application/x-oleobject;base64, ...данные base64...">
    Часы.
</OBJECT>
```

Информацию о размере, выравнивании и границах объекта см. в разделе визуальное представление объектов, изображений и апплетов.

13.3.2 Инициализация объекта: элемент PARAM

```
<!ELEMENT PARAM - O EMPTY               -- значение именованного свойства -->
<!ATTLIST PARAM
  id          ID              #IMPLIED  -- идентификатор в пределах
                                           документа --
  name        CDATA           #REQUIRED -- имя свойства --
  value       CDATA           #IMPLIED  -- значение свойства --
  valuetype   (DATA|REF|OBJECT) DATA    -- Как интерпретировать значение --
  type        %ContentType;   #IMPLIED  -- тип содержимого для значения,
```

```
                                        если valuetype=ref --
  >
```

Начальный тег: ***обязателен****, Конечный тег:* ***запрещен***

Определения атрибутов

`name` = *cdata*

Этот атрибут определяет имя рабочего параметра, которое должно быть понятно вставляемому объекту. Учитывает ли имя свойства регистр, зависит от конкретной реализации объекта.

`value` = *cdata*

Этот атрибут определяет значение рабочего параметра, задаваемого атрибутом `name`. Значения свойств не имеют значения в HTML; их значение определяется объектом.

`valuetype` = `data|ref|object` [CI]

Этот атрибут определяет тип атрибута `value`. Возможные значения:

- `data`: Это значение используется по умолчанию. Оно означает, что значение, задаваемое атрибутом `value`, будет определяться и передаваться в объект в виде строки.
- `ref`: Значение, задаваемое атрибутом `value`, является адресом URI ресурса, где хранятся рабочие значения. Это позволяет средствам поддержки идентифицировать адреса URI, данные в качестве параметров. Адрес URI должен передаваться в объект **как есть**, то есть неразрешенным.
- `object`: Значение, задаваемое атрибутом `value`, является идентификатором, ссылающимся на объявление `OBJECT` в этом же документе. Идентификатором должно быть значение атрибута `id` для объявленного элемента `OBJECT`.

`type` = *content-type* [CI]

Этот атрибут задает тип содержимого ресурса, назначаемого атрибутом `value` **только** в случае, если значением атрибута `valuetype` является "ref". Таким образом, этот атрибут определяет для агента пользователя тип значений, которые будут находиться по адресу URI, назначенному атрибутом `value`.

Атрибуты, определенные в другом месте

- `id` (идентификаторы в пределах документа)

Элементы `PARAM` определяют набор значений, которые могут понадобиться объекту во время работы. В элементах `OBJECT` или `APPLET` может присутствовать любое число атрибутов `PARAM` в любом порядке, но они должны помещаться в начале тела включающего элемента `OBJECT` или `APPLET`.

Синтаксис имен и значений считается понятным обработчику объекта. Данный документ не указывает, как агенты пользователей должны загружать пары имя/значение, а также того, как они должны интерпретировать повторяющиеся имена параметров.

Вернемся к примеру с часами и покажем использование элемента `PARAM`: предположим, что апплет может принимать два рабочих параметра, определяющих его начальную высоту и ширину. Мы можем установить исходные размеры 40x40 пикселов с помощью двух элементов `PARAM`.

```
<P><OBJECT classid="http://www.miamachina.it/analogclock.py">
<PARAM name="height" value="40" valuetype="data">
<PARAM name="width" value="40" valuetype="data">
Этот агент пользователя не может сгенерировать приложение на языке Python.
</OBJECT>
```

В следующем примере рабочие данные для параметра "Init_values" объекта задаются в виде внешнего ресурса (файл GIF). Таким образом для атрибута `valuetype` устанавливается значение "ref", а атрибутом `value` является адрес URI ресурса.

```
<P><OBJECT classid="http://www.gifstuff.com/gifappli"
       standby="Загрузка Элвиса...">
```

```
<PARAM name="Init_values"
       value="./images/elvis.gif">
       valuetype="ref">
</OBJECT>
```

Обратите внимание, что мы также установили атрибут standby, так что агент пользователя может отобразить сообщение во время загрузки генерирующего механизма. Когда элемент OBJECT сгенерирован, агенты пользователя должны выполнить поиск содержимого только для тех элементов PARAM, которые являются их прямыми дочерними элементами и "feed" их to the OBJECT.
Таким образом, в следующем примере, если сгенерирован "obj1", "param1" применяется к "obj1" (и не применяется к "obj2"). Если "obj1" не сгенерирован, а "obj2" сгенерирован, "param1" игнорируется, а "param2" применяется к "obj2". Если ни один OBJECT не сгенерирован, ни один PARAM не применяется.

```
<P>
<OBJECT id="obj1">
   <PARAM name="param1">
   <OBJECT id="obj2">
      <PARAM name="param2">
   </OBJECT>
</OBJECT>
```

13.3.3 Глобальные схемы именования объектов

Местоположение обработчика объекта задается адресом URI. Как было сказано во введении в URI, первый сегмент абсолютного адреса URI задает схему именования, используемую для передачи данных, назначаемых адресом URI. Для документов в формате HTML этой схемой часто является "http". Некоторые апплеты могут использовать внешние схемы именования. Например, при указании апплета Java авторы могут использовать адреса URI, начинающиеся с "java", я для апплетов ActiveX авторы могут использовать "clsid".
В следующем примере мы вставляем апплет на языке Java в документ в формате HTML.

```
<P><OBJECT classid="java:program.start">
</OBJECT>
```

Установив атрибут codetype, агент пользователя может определить, нужно ли загружать ли приложение Java, в зависимости от своих возможностей.

```
<OBJECT codetype="application/java-archive"
        classid="java:program.start">
</OBJECT>
```

Некоторым схемам генерации для определения обработки необходима дополнительная информация, поэтому им необходимо указать, где находится эта информация. Авторы могут указать путь к обработчику объекта с помощью атрибута codebase.

```
<OBJECT codetype="application/java-archive"
        classid="java:program.start">
        codebase="http://foooo.bar.com/java/myimplementation/"
</OBJECT>
```

В следующем примере (с помощью атрибута classid) задается объект ActiveX в виде адреса URI, начинающегося со схемы именования "clsid". Атрибут data определяет местоположение генерируемых данных (еще одни часы).

```
<P><OBJECT classid="clsid:663C8FEF-1EF9-11CF-A3DB-080036F12502"
        data="http://www.acme.com/ole/clock.stm">
Это приложение не поддерживается.
</OBJECT>
```

13.3.4 Объявление и инициализация объекта

В приведенных выше примерах были показаны отдельные определения объектов. Если в документе должно содержаться несколько экземпляров одного и того же объекта, объявление и инициализацию объекта можно разделить. Такой способ имеет несколько преимуществ:

- Данные могут загружаться агентом пользователя из сети *один раз* (во время объявления) и повторно использоваться в каждой инициализации.
- Инициализировать объект можно из местоположения, отличного от того, в котором объект объявлялся, например, из ссылки.
- Объекты можно определять в качестве рабочих данных для других объектов.

Чтобы объявить объект так, чтобы он не обрабатывался агентом пользователя при чтении, установите логический атрибут `declare` элемента `OBJECT`. В то же время авторы должны идентифицировать объявление, установив уникальное значение для атрибута `id` в элементе `OBJECT`. Инициализация объекта позже будет ссылаться на этот идентификатор. Объявленный `OBJECT` должен присутствовать в документе до первого экземпляра `OBJECT`.

Объект, определенный с атрибутом `declare`, инициализируется каждый раз, когда необходима генерация элемента, ссылающегося на этот объект (например, активизируется ссылка на него, активизируется объект, ссылающийся на него и т.д.).

В следующем примере мы объявляем `OBJECT` и вызываем его инициализацию, указав его в ссылке. Таким образом объект можно активизировать, щелкнув, например, на выделенном тексте.

```
<P><OBJECT declare
        id="earth.declaration"
        data="TheEarth.mpeg"
        type="application/mpeg">
   Вид <STRONG>Земли</STRONG> из космоса.
</OBJECT>
...далее в документе...
<P>Красивое<A href="#earth.declaration"> анимационное изображение Земли!</A>
```

В следующем примере показано, как указать рабочие значения, являющиеся другими объектами. В этом примере мы отправляем текст (стихотворение) гипотетическому механизму для просмотра стихотворений. Объект распознает рабочий параметр с именем "font" (скажем, для генерации текста стихотворения с использованием определенного шрифта). Значение этого параметра само является объектом, вставляющим (но не генерирующим) объект шрифта. Отношение между объектом шрифта и объектом механизма просмотра стихотворений достигается с помощью (1) назначения атрибута `id` в объявлении объекта шрифта и (2) ссылки на него в элементе `PARAM` объекта механизма просмотра стихотворений (с помощью `valuetype` и `value`).

```
<P><OBJECT declare
     id="tribune"
     type="application/x-webfont"
     data="tribune.gif">
</OBJECT>
...просмотр стихотворения из файла KublaKhan.txt...
<P><OBJECT classid="http://foo.bar.com/poem_viewer"
           data="KublaKhan.txt">
<PARAM name="font" valuetype="object" value="#tribune">
<P>У вас нет такой классной программы просмотра стихотворений...
</OBJECT>
```

Агенты пользователей, не поддерживающие атрибут `declare`, должны генерировать содержимое объявления `OBJECT`.

13.4 Включение апплета: элемент APPLET

Элемент APPLET является нежелательным (как и все атрибуты этого элемента), вместо него следует использовать элемент `OBJECT`.

Формальное определение см. в Transitional DTD.

Определения атрибутов

`codebase` = *uri* [CT]

Этот атрибут определяет базовый адрес URI апплета. Если этот атрибут не указан, по умолчанию используется базовый адрес URI, используемый для всего документа. Значениями этого атрибута могут быть только подкаталоги каталога, в котором расположен текущий документ.

`code` = *cdata* [CS]

Этот атрибут определяет имя файла класса, содержащего скомпилированный подкласс апплета или путь, по которому можно получить класс, включая сам файл класса. Он интерпретируется с учетом кода апплета. Для этого должен быть указан один из атрибутов `code` или `object`.

`name` = *cdata* [CS]

Этот атрибут определяет имя экземпляра апплета, что дает возможность апплетам на одной странице находить друг друга и взаимодействовать друг с другом.

`archive` = *uri-list* [CT]

Этот атрибут определяет разделенный запятыми список адресов URI архивов, содержащих классы и другие ресурсы, которые будут "предварительно загружаться". Классы загружаются с помощью экземпляра AppletClassLoader с заданным `codebase`. Относительные адреса URI архивов интерпретируются относительно codebase апплета. Предварительная загрузка ресурсов может существенно увеличить производительность апплетов.

`object` = *cdata* [CS]

Этот атрибут определяет имя ресурса, содержащего serialized представление состояния апплета. Он интерпретируется относительно codebase апплета. serialized данные содержат имя класса апплета, но не обработчика. Имя класса используется для загрузки обработчика из файла класса или архива.

Если апплет "deserialized", метод `start()` вызывается вместо метода `init()`. Атрибуты, допустимые при serialized исходного объекта, **не** восстанавливаются. Атрибуты, переданные в этот экземпляр `APPLET`, будут доступны апплету. Авторам следует очень осторожно использовать это свойство. Перед serialized в апплет должен быть остановлен.

Должен присутствовать один из атрибутов `code` или `object`. Если даны оба атрибута `code` и `object`, и в них указаны разные имена классов, это является ошибкой.

`width` = *длина* [CI]

Этот атрибут определяет начальную ширину области отображения апплета (не включая окна и диалоги, создаваемые апплетом).

`height` = *длина* [CI]

Этот атрибут определяет начальную высоту области отображения апплета (не включая окна и диалоги, создаваемые апплетом).

Атрибуты, определенные в другом месте

- `id`, `class` (идентификаторы в пределах документа)
- `title` (заголовок элемента)
- `style` (встроенная информация о стиле)
- `alt` (альтернативный текст)
- `align`, `hspace`, `vspace` (визуальное представление объектов, изображений и апплетов)

Этот элемент, поддерживаемый всеми программами просмотра с поддержкой Java, позволяет дизайнерам внедрять апплеты Java в документы HTML. Он является нежелательным, и вместо него следует использовать элемент `OBJECT`.

Содержимое элемента `APPLET` служит альтернативной информацией для агентов

пользователей, не поддерживающих этот элемент или не сконфигурированных для поддержки апплетов. В противном случае агенты пользователей должны игнорировать содержимое.

ПРИМЕР НЕЖЕЛАТЕЛЬНОГО ИСПОЛЬЗОВАНИЯ:

В следующем примере элемент APPLET включает в документ апплет на языке Java. Поскольку атрибут codebase не установлен, предполагается, что апплет находится в том же каталоге, что и сам документ.

```
<APPLET code="Bubbles.class" width="500" height="500">
Java-апплет, рисующий движущиеся пузыри.
</APPLET>
```

Этот пример можно переписать с использованием элемента OBJECT следующим образом:

```
<P><OBJECT codetype="application/java"
        classid="java:Bubbles.class"
        width="500" height="500">
Java-апплет, рисующий движущиеся пузыри.
</OBJECT>
```

Задать для апплета исходные значения можно с помощью элемента PARAM.

ПРИМЕР НЕЖЕЛАТЕЛЬНОГО ИСПОЛЬЗОВАНИЯ:

Следующий апплет на языке Java:

```
<APPLET code="AudioItem" width="15" height="15">
<PARAM name="snd" value="Hello.au|Welcome.au">
Java-апплет, воспроизводящий звуковой файл приветствия.
</APPLET>
```

можно определить с использованием элемента OBJECT следующим образом:

```
<OBJECT codetype="application/java"
        classid="AudioItem"
        width="15" height="15">
<PARAM name="snd" value="Hello.au|Welcome.au">
Java-апплет, воспроизводящий звуковой файл приветствия.
</OBJECT>
```

13.5 Замечания о внедренных документах

Иногда вместо ссылки на документ автору нужно внедрить его непосредственно в первичный документ HTML. Авторы могут использовать для этого элемент IFRAME или OBJECT, но эти элементы несколько различны. Эти два элемента не только имеют различные модели содержимого, но также элемент IFRAME может быть целевым кадром (подробнее см. раздел об указании информации о целевом кадре) и может быть "выделен" агентом пользователя для печати, просмотра кода HTML и т.д. Агенты пользователя могут генерировать выделенные кадры способом, отличным от генерации невыделенных кадров (например, отображать границу вокруг выделенного кадра).

Внедренный документ полностью независим от документа, в который он внедрен. Например, относительные адреса URI во внедренном документе разрешаются в соответствии с базовым адресом URI, указанным во внедренном документе, а не в основном документе. Внедренный документ только генерируется в другом документе (например, во вложенном окне); it во всех остальных отношениях он остается независимым.

Например, следующая строка внедряет содержимое файла `embed_me.html` в то место документа, в котором встречено определение OBJECT.

```
...предшествующий текст...
<OBJECT data="embed_me.html">
Внимание: невозможно внедрить файл embed_me.html.
</OBJECT>
...последующий текст...
```

Вспомните, что содержимое элемента OBJECT должно генерироваться, только если файл, задаваемый атрибутом data, невозможно загрузить.
Поведение агента пользователя в случаях, когда файл включает сам себя, не определено.

13.6 Навигационные карты

Навигационные карты позволяют авторам определять области изображения или объекта и назначать каждой области определенное действие (например, загрузку документа, запуск программы и т.д.) Когда область активизируется пользователем, выполняется действие. Навигационная карта создается путем назначения объекта с указанием соответствующих геометрических областей.
Имеется два типа навигационных карт:

- *Клиентская.* Когда пользователь активизирует область клиентской навигационной карты с помощью мыши, координаты точки интерпретируются агентом пользователя. Агент пользователя выбирает ссылку, указанную для активизированной области, и выполняет ее.
- *Серверная.* Когда пользователь активизирует область серверной навигационной карты с помощью мыши, координаты точки щелчка передаются агенту на сервере, определенному с помощью атрибута href элемента A. Агент на сервере интерпретирует координаты и выполняет соответствующие действия.

Клиентские навигационные карты предпочтительны по отношению к серверным по крайней мере по двум причинам: они доступны пользователям неграфических агентов и позволяют незамедлительно определить, находится ли указатель в активной области.

13.6.1 Клиентские навигационные карты: элементы MAP и AREA

```
<!ELEMENT MAP - - ((%block;)+ | AREA+) - клиентская навигационная карта -->
<!ATTLIST MAP
  %attrs;                               -- %coreattrs, %i18n, %events --
  name        CDATA           #REQUIRED - для ссылки usemap --
  >
```

Начальный тег: ***обязателен****, Конечный тег:* ***обязателен***

```
<!ELEMENT AREA - O EMPTY                -- область клиентской навигационной карты -->
<!ATTLIST AREA
  %attrs;                               -- %coreattrs, %i18n, %events --
  shape       %Shape;         rect      -- управляет интерпретацией координат --
  coords      %Coords;        #IMPLIED  -- разделенный запятыми список длин --
  href        %URI;           #IMPLIED  -- адрес URI связанного ресурса --
  nohref      (nohref)        #IMPLIED  -- для этого региона действие не назначено --
  alt         %Text;          #REQUIRED -- краткое описание --
  tabindex    NUMBER          #IMPLIED  -- положение в последовательности перехода --
  accesskey   %Character;     #IMPLIED  -- символ доступа --
  onfocus     %Script;        #IMPLIED  -- элемент получил фокус --
  onblur      %Script;        #IMPLIED  -- элемент потерял фокус --
  >
```

Начальный тег: ***обязателен****, Конечный тег:* ***запрещен***

Определения атрибутов элемента MAP

`name` = *cdata* [CI]

Этот атрибут назначает имя навигационной карты, определяемой элементом MAP.

Определения атрибутов элемента AREA

`shape` = `default|rect|circle|poly` [CI]

Этот атрибут определяет форму области. Возможные значения:

- `default`: Задает всю область.
- `rect`: Определяет прямоугольную область.
- `circle`: Определяет круглую область.
- `poly`: Определяет многоугольную область.

`coords` = *координаты* [CN]

Этот атрибут определяет положение формы на экране. Число и порядок значений зависят от определенной формы. Возможные комбинации:

- `rect:` x левой границы, y верхней границы, x правой границы, y нижней границы.
- `circle:` x центра, y центра, радиус. **Примечание.** Если радиус указан в процентах, агенты пользователя должны вычислять окончательное значение радиуса в зависимости от назначенных объекту ширины и высоты. Радиус должен быть наименьшим из этих двух значений.
- `poly:` x1, y1, x2, y2, ..., xN, yN.

Координаты задаются относительно верхнего левого угла объекта. Все значения являются длинами. Все значения отделяются друг от друга запятыми.

`nohref` [CI]

Если этот логический атрибут установлен, он указывает, что с этой областью ссылка не связана.

Атрибут для установления связи навигационной карты с элементом

`usemap` = *uri* [CT]

Этот атрибут связывает навигационную карту с элементом. Навигационная карта определяется с помощью элемента `MAP`. Значение атрибута `usemap` должно совпадать со значением атрибута `name` связанного элемента `MAP`.

Атрибуты, определенные в другом месте

- `id`, `class` (идентификаторы в пределах документа)
- `lang` (информация о языке), `dir` (направление текста)
- `title` (заголовок элемента)
- `style` (встроенная информация о стиле)
- `name` (предоставление агентов с помощью форм)
- `alt` (альтернативный текст)
- `href` (ссылка anchor) `target` (информация о целевом кадре)
- `tabindex` (переход по клавише Tab)
- `accesskey` (клавиши доступа)
- `shape` (навигационные карты)
- `onclick`, `ondblclick`, `onmousedown`, `onmouseup`, `onmouseover`, `onmousemove`, `onmouseout`, `onkeypress`, `onkeydown`, `onkeyup`, `onfocus`, `onblur` (внутренние события)

Элемент `MAP` определяет клиентскую навигационную карту, которая может быть связана с одним или несколькими элементами (`IMG`, `OBJECT` или `INPUT`). Навигационная карта связывается с элементом с помощью атрибута `usemap` этого элемента.

Наличие атрибута `usemap` в элементе `OBJECT` подразумевает, что объект включается в виде изображения. Более того, если с элементом `OBJECT` связана клиентская навигационная карта, агенты пользователей могут интерпретировать взаимодействие пользователя с элементом `OBJECT` исключительно в терминах клиентской навигационной карты. Это позволяет агентам пользователей (например, звуковым браузерам или роботам) взаимодействовать с элементом `OBJECT`, не обрабатывая его; агент пользователя может даже не загружать (или не обрабатывать) объект. Если с элементом `OBJECT` связана навигационная карта, авторы не могут быть уверены, что этот объект будет загружаться и обрабатываться всеми агентами пользователей.

Каждый элемент `MAP` может содержать следующее:

1. Один или несколько элементов `AREA`. Эти элементы не имеют содержимого, но определяют геометрические области навигационной карты и ссылки, связанные с каждой областью. Обратите внимание, что при использовании этого метода элемент `MAP` не имеет генерируемого содержимого. Таким образом, авторы должны изменять текст для каждого элемента `AREA` с помощью атрибута `alt` (см. далее информацию о том, как указать

альтернативный текст).

2. Содержимое уровня блока. Сюда должны включаться элементы A, определяющие геометрические области навигационной карты, и ссылки, связанные с каждой областью. Обратите внимание, что при использовании этого метода содержимое элемента MAP может генерироваться агентом пользователя. Для создания более доступных документов авторам следует использовать этот способ.
Если несколько определенных регионов перекрываются, приоритет имеет элемент, указанный в документе первым (т.е. на ввод пользователя отвечает именно этот элемент). Агенты пользователей и авторы должны обеспечивать текстовые альтернативы графических навигационных карт на случай, если графика недоступна или пользователь не может получить к ней доступ. Например, агенты пользователей могут использовать текст атрибута alt для создания текстовых ссылок вместо графической навигационной карты. Такие ссылки могут активизироваться различными способами (клавиатура, голос и т.д.).
Примечание. *Элемент MAP не совместим с агентами пользователей для языка HTML версии 2.0.*

Примеры клиентских навигационных карт

В следующем примере мы создаем клиентскую навигационную карту для элемента OBJECT. Мы не хотим генерировать содержимое карты при генерации элемента OBJECT, поэтому мы "скроем" элемент MAP в содержимом элемента OBJECT. Затем содержимое элемента MAP будет генерироваться, только если нельзя сгенерировать содержимое элемента OBJECT.

```
<HTML>
   <HEAD>
      <TITLE>Крутая страница!</TITLE>
   </HEAD>
   <BODY>
     <P><OBJECT data="navbar1.gif" type="image/gif" usemap="#map1">
     <MAP name="map1">
       <P>Перемещение по узлу:
       <A href="guide.html" shape="rect" coords="0,0,118,28">Руководство по
доступу</a> |
       <A href="shortcut.html" shape="rect" coords="118,0,184,28">Переход</A> |
       <A href="search.html" shape="circle" coords="184,200,60">Поиск</A> |
       <A href="top10.html" shape="poly" coords="276,0,373,28,50,50,276,0">Первые
десять</A><
     </MAP>
     </OBJECT>
   </BODY>
</HTML>
```

Нам может понадобиться генерация содержимого карты, даже если агент пользователя может сгенерировать элемент OBJECT. Например, мы хотим связать навигационную карту с элементом OBJECT *и* включить текстовую навигационную панель внизу страницы. Чтобы это сделать, определим элемент MAP вне элемента OBJECT:

```
<HTML>
   <HEAD>
      <TITLE>Крутая страница!</TITLE>
   </HEAD>
   <BODY>
     <P><OBJECT data="navbar1.gif" type="image/gif" usemap="#map1">
     </OBJECT>

     ...продолжение страницы...

     <MAP name="map1">
       <P>Перемещение по узлу:
       <A href="guide.html" shape="rect" coords="0,0,118,28">Руководство по
доступу</a> |
       <A href="shortcut.html" shape="rect" coords="118,0,184,28">Переход</A> |
```

```
       <A href="search.html" shape="circle" coords="184,200,60">Поиск</A> |
       <A href="top10.html" shape="poly" coords="276,0,373,28,50,50,276,0">Первые
десять</A>
     </MAP>
   </BODY>
</HTML>
```

В следующем примере мы создаем сходную навигационную карту, на этот раз используя элемент AREA. Обратите внимание на использование текста alt:

```
<P><OBJECT data="navbar1.gif" type="image/gif" usemap="#map1">
   <P>Это навигационная панель.
   </OBJECT>

<MAP name="map1">
 <AREA href="guide.html"
          alt="Руководство по доступу"
          shape="rect"
          coords="0,0,118,28">
 <AREA href="search.html"
          alt="Поиск"
          shape="rect"
          coords="184,0,276,28">
 <AREA href="shortcut.html"
          alt="Переход"
          shape="circle"
          coords="184,200,60">
 <AREA href="top10.html"
          alt="Первые десять"
          shape="poly"
          coords="276,0,373,28,50,50,276,0">
</MAP>
```

Вот версия с использованием элемента IMG вместо элемента OBJECT (с тем же самым объявлением MAP):

```
<P><IMG src="navbar1.gif" usemap="#map1" alt="навигационная панель">
```

В следующем примере показано, как элементы могут совместно использовать навигационные карты.

Вложенные элементы OBJECT полезны для обеспечения fallbacks в случае, если агент пользователя не поддерживает определенные форматы. Например:

```
<P>
<OBJECT data="navbar.png" type="image/png">
  <OBJECT data="navbar.gif" type="image/gif">
    текст с описанием изображения...
  </OBJECT>
</OBJECT>
```

Если агент пользователя не поддерживает формат PNG, он пытается сгенерировать изображение в формате GIF. Если он не поддерживает GIF (например, это речевой агент пользователя), он воспроизводит текстовое описание, указанное в содержимом внутреннего элемента OBJECT. Если элементы OBJECT вложены таким образом, авторы могут обеспечивать совместное использование этими элементами навигационных карт:

```
<P>
<OBJECT data="navbar.png" type="image/png" usemap="#map1">
  <OBJECT data="navbar.gif" type="image/gif" usemap="#map1">
     <MAP name="map1">
     <P>Перемещение по узлу:
      <A href="guide.html" shape="rect" coords="0,0,118,28">Руководство по доступу</a>
|
      <A href="shortcut.html" shape="rect" coords="118,0,184,28">Переход</A> |
      <A href="search.html" shape="circle" coords="184,200,60">Поиск</A> |
      <A href="top10.html" shape="poly" coords="276,0,373,28,50,50,276,0">Первые
десять</A>
     </MAP>
```

```
  </OBJECT>
</OBJECT>
```

В следующем примере показано, как можно указать anchors для создания неактивных зон в навигационной карте. Первый anchor определяет небольшую круглую область, с которой не связана ссылка. Второй anchor определяет круглую область большего размера с той же координатой центра. Обе они вместе образуют кольцо, центр которого неактивен, а внешняя часть активна. Порядок определения anchor важен, поскольку меньший круг должен иметь приоритет над большим.

```
<MAP name="map1">
<P>
<A shape="circle" coords="100,200,50">Я неактивная.</A>
<A href="outer-ring-link.html" shape="circle" coords="100,200,250">Я активная.</A>
</MAP>
```

Точно так же атрибут `nohref` элемента `AREA` объявляет, что с геометрической областью не связана ссылка.

13.6.2 Серверные навигационные карты

Серверные навигационные карты представляют интерес в случаях, когда карта слишком сложна.

Определить серверную навигационную карту можно только для элементов `IMG` и `INPUT`. В случае элемента `IMG` этот элемент должен быть включен в элемент `A`. В случае элемента `INPUT` он должен иметь тип "image". В обоих случаях для элемента должен быть установлен логический атрибут `ismap` [CI].

Когда пользователь активизирует ссылку, щелкнув на изображении, экранные координаты отправляются непосредственно на сервер, на котором располагается документ. Экранные координаты выражаются в виде пикселов относительно изображения. Нормативную информацию об определении пикселов и об их масштабировании см. в [CSS1].

В следующем примере активная область определяет серверную ссылку. Таким образом щелчок в любой точке изображения вызовет передачу координат на сервер.

```
<P><A href="http://www.acme.com/cgi-bin/competition">
        <IMG src="game.gif" ismap alt="target"></A>
```

Координаты области, в которой произошел щелчок, передаются на сервер следующим образом. Агент пользователя получает новый адрес URI из адреса URI, указанного в атрибуте `href` элемента `A`, добавляя `?', за которым следуют координаты x и y, разделенные запятой. Затем происходит переход по ссылке с использованием нового адреса URI. Например, в данном примере, если пользователь щелкает в точке с координатами x=10, y=27, то новый адрес URI - "http://www.acme.com/cgi-bin/competition?10,27".

Агенты пользователей, не предлагающие пользователям средств выбора определенных координат (например, неграфические агенты пользователей, зависящие от ввода с клавиатуры, речевые агенты пользователей и т.д.) должны при активизации ссылки передавать на сервер координаты "0,0".

13.7 Визуальное представление изображений, объектов и апплетов

Все атрибуты элементов `IMG` и `OBJECT`, относящиеся к визуальному выравниванию и представлению, являются устаревшими, вместо них следует использовать таблицы стилей.

13.7.1 Ширина и высота

Определения атрибутов

`width` = *длина* [CN]

Переопределение ширины изображения и объекта.
`height` = *длина* [CN]
Переопределение для изображения и объекта.
Если указаны атрибуты `width` и `height`, они сообщают агентам пользователя о необходимости переопределения исходного размера изображения или объекта этими значениями.
Если объектом является изображение, оно масштабируется. Агенты пользователей должны наилучшим образом масштабировать объект или изображение, чтобы они соответствовали ширине и высоте, определенным автором. Обратите внимание, что длины, выраженные в процентах, зависят от доступного горизонтального или вертикального пространства, а не от исходного размера изображения, объекта или апплета.
Атрибуты `height` и `width` дают агентам пользователей представление о размере изображения или объекта, чтобы они могли зарезервировать соответствующее пространство и продолжать генерацию документа, ожидая данных об изображении.

13.7.2 Пространство вокруг изображений и объектов

Атрибуты `vspace` и `hspace` определяют свободное пространство слева и справа (hspace) и над и под (vspace) `IMG`, `APPLET`, `OBJECT`. По умолчанию значение этого атрибута не определено, но обычно это небольшое ненулевое значение. Оба атрибута имеют значение типа *длина*.

13.7.3 Границы

Изображение или объект может окружать граница (например, если она указана пользователем или изображение имеет содержимое элемента `A`).
Определения атрибутов
`border` = *пикселы*
Нежелателен. Атрибут `border` определяет ширину границы в пикселах. Значение этого атрибута, используемое по умолчанию, зависит от агента пользователя.

13.7.4 Выравнивание

Атрибут `align` определяет положение `IMG`, `OBJECT` или `APPLET` относительно его содержимого.
Следующие значения атрибута `align` относятся к положению объекта относительно окружающего текста:

- `bottom`: означает, что окно объекта должно быть вертикально выровнено относительно текущей базовой линии. Это значение используется по умолчанию.
- `middle`: означает, что центр объекта должен быть выровнен вертикально относительно текущей базовой линии.
- `top`: означает, что верх объекта должен быть вертикально выровнен относительно верха текущей текстовой строки.

Два других значения, `left` и `right`, приводят к перемещению изображения к текущему левому или правому полю. Они обсуждаются в разделе о плавающих объектах.
Различие интерпретаций атрибута align. *Агенты пользователей по-разному интерпретируют атрибут `align`. Некоторые принимают в расчет только текстовую строку, находящуюся перед элементом, некоторые учитывают текст по обеим сторонам элемента.*

13.8 Как указать альтернативный текст

Определения атрибутов
`alt` = *текст* [CS]
Для агентов пользователей, не имеющих возможности вывести изображения, формы или апплеты, этот атрибут определяет альтернативный текст. Язык альтернативного текста определяется атрибутом `lang`.

Для некоторых нетекстовых элементов (IMG, AREA, APPLET и INPUT) авторы должны указывать альтернативный текст, служащий содержимым, если элемент нельзя нормально сгенерировать. Задание альтернативного текста помогает пользователям, не имеющих графических дисплеев, пользователям, браузеры которых не поддерживают формы, visually impaired users, пользователям синтезаторов речи, пользователям, графический агент которых сконфигурирован так, чтобы не выводить изображения и т.д.
Атрибут alt должен быть указан для элементов IMG и AREA. Он не является обязательным для элементов INPUT и APPLET.
В то время как альтернативный текст может быть очень полезным, использовать его нужно осторожно. Авторы должны иметь в виду следующее:

- Не указывайте несоответствующий альтернативный текст, включая изображения, предназначенные для *форматирования* страницы, например, `alt="red ball"` не соответствует изображению, определяющему красный круг для отметки заголовка абзаца. В этих случаях в качестве альтернативного текста следует указать пустую строку (""). Авторам в любом случае рекомендуется не использовать изображения для форматирования страниц; вместо этого следует использовать таблицы стилей.
- Не указывайте бессмысленный альтернативный текст (например, "dummy text"). Он не только будет frustrate пользователей, но также будет замедлять агенты пользователей, преобразующие текст в речь или текст Бройля.
Информацию об обработке случаев, когда альтернативный текст отсутствует, разработчики могут найти в разделе о генерации альтернативного текста.