−Table of Contents

This page has been deprecated and is no longer being maintained.
For up to date information on CSS test format guidelines and test templates, see:
http://testthewebforward.org/docs/test-format-guidelines.html
http://testthewebforward.org/docs/test-templates.html

Test Format

This page describes the standard test format for CSSWG self-describing tests, reftests and script tests. The requirements are explained below; to make things easier we've also provided a template that you can copy.

The recommended structure for CSS tests is a self-describing reftest. If it's possible for a test to be a reftest, it must be a reftest. (We prefer reftests to also be self-describing or otherwise easy for humans to interpret, but this is not a requirement). script tests should be used for scenarios such as testing a JS API or adding automation to a reftest.

Design Requirements

The following are requested of all tests submitted to the CSSWG, both for reftests, script tests, and self-describing tests:

Short
: Tests should be very short and certainly not require scrolling on even the most modest of screens, unless the test is specifically for scrolling or paginating behaviour. This enables them to be run more easily on various testing platforms.

Valid
: Unless specifically testing error-recovery features, the tests should all be valid. Inconsistencies in e.g. HTML parsing shouldn't be affecting CSS test suite pass rates.

Cross-platform
: Tests should be as cross-platform as reasonably possible, working across different devices, screen resolutions, paper sizes, etc. Exceptions should document their assumptions.

Red Means Failure
: Don't use the color red other than to indicate a failure. (Exception: testing for support of red) Since green-with-no-red is often used to indicate success, it's best to also avoid green unless using the presence of red to indicate failures.

Acceptable Test Formats

The preferred submission format for CSSWG tests is either XHTML or HTML5, in UTF-8. HTML < 5 is also acceptable, but it will be processed by an HTML5 compatible parser. SVG is also acceptable where necessary.

Note that in general, the test source will be parsed and re-serialized, even in its source format. The re-serialization will cause minor changes to the test file, notably: attribute values will always be quoted, whitespace between attributes will be collapsed to a single space, duplicate attributes will be removed, optional closing tags will be inserted, and invalid markup will be normalized. If these changes should make the test inoperable, for example if the test is testing markup error recovery, add the flag 'asis' to prevent re-serialization. This flag will also prevent format conversions so it may be necessary to provide alternate versions of the test in other formats (XHTML, HTML, etc.)

Images must be in either PNG or SVG format. (PNG is preferred where raster images can be used.)

A set of scripts will generate the various formats (XHTML, HTML, XHTML for Printers) from this source version.

Tests must be *valid*, so please validate your tests [http://validator.w3.org/] before submission. For tests that use the HTML style header, the validator may report errors on the flags and assert metadata. These specific errors can be ignored - this is a known issue and work is in progress to correct the problem.

When using any characters beyond the ASCII set, in any encoding, the character encoding must be specified properly per the specification of the source format.

If the test uses the Ahem font, make sure its computed font-size is a multiple of 5px, otherwise baseline alignment may be rendered inconsistently (due to rounding errors introduced by certain platforms' font APIs). We suggest to use a minimum computed font-size of 20px.

- Eg. Bad: {font: 1in/1em Ahem;} /* computed font-size is 96px */
- Eg. Bad: {font: 1in Ahem;}
- Eg. Bad: {font: 1em/1em Ahem} /* with computed 1em font-size being 16px */
- Eg. Bad: {font: 1em Ahem;} /* with computed 1em font-size being 16px */
- Eg. Good: {font: 100px/1 Ahem;}
- Eg. Good: {font: 1.25em/1 Ahem;} /* with computed 1.25em font-size being 20px */

If the test uses the Ahem font, make sure the line-height on block elements is specified; avoid 'line-height: normal'. Also, for absolute reliability, the difference between computed line-height and computed font-size should be dividable by 2.

- Eg. Bad: {font: 1.25em Ahem;} /* computed line-height value is 'normal' */
- Eg. Bad: {font: 20px Ahem;} /* computed line-height value is 'normal' */
- Eg. Bad: {font-size: 25px; line-height: 50px;} /* the difference between computed line-height and computed font-size is **not** dividable by 2. */
- Eg. Good: {font-size: 25px; line-height: 51px;} /* the difference between computed line-height and computed font-size is dividable by 2. */

Template for New Tests

A template for new tests follows. Copy and paste the code below into a new file or save this template file and replace the bracketed parts as described below.

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
 <head>
  <title>CSS Test: [Title/Scope of Test]</title>
  <link rel="author" title="[Name of Author]" href="[mailto: or http: contact address]" />
  <link rel="help" href="http://www.w3.org/TR/[direct link to tested spec section]" />
  <meta name="flags" content="[requirement flags]" />
  <meta name="assert" content="Test checks that [explanation of what you're trying to test]." />
  <style type="text/css"><![CDATA[
   [CSS for test]
  ]]></style>
 </head>
 <body>
  <p>Test passes if [description of pass condition].</p>
  [Content of test]
 </body>
</html>
```

Alternatively, you can use this HTML5 template:

```
<!DOCTYPE html>
<title>CSS Test: [Title/Scope of Test]</title>
<link rel="author" title="[Name of Author]" href="[mailto: or http: contact address]">
<link rel="help" href="http://www.w3.org/TR/[direct link to tested section]">
<meta name="flags" content="[requirement flags]">
<meta name="assert" content="Test checks that [explanation of what you're trying to test].">
<style>
   [CSS for test]
</style>
<body>
  <p>Test passes if [description of pass condition].</p>
  [Content of test]
</body>
```

Template Details

Title element

```
<title>CSS Test: SCOPE OF TEST</title>
```

The title appears in the generated index, so make sure it is *concise*, *unique* and *descriptive*. The role of the title is to identify what specific detail of a feature or combination of features is being tested, so that someone looking through an index can see quickly what's tested in which file. In most cases, this description should not require more than 5 or 6 words. There is no need to provide the chapter or section in the title.

Bad example:

```
<title>CSS Test: Border Conflict Resolution</title>
```

We have 100+ tests on “Border Conflict Resolution”.

Good example:

```
<title>CSS Test: Border Conflict Resolution (width) - hidden/double</title>
```

For specifications other than CSS 2.1, you can include the module name somewhere before the colon, like “CSS Selectors Test:” or “CSS Test (Selectors):”. Do not include the module version number, since the test might get reused for the next version.

Credits

```
<link rel="author" title="NAME_OF_AUTHOR" href="mailto:EMAIL OR http://CONTACT_PAGE" />
```

Credits provide a way to identify the person or organization that created the test and/or holds copyright in the test. This is useful for reviewing purposes and for asking questions about the individual test. A test can have multiple author credits if necessary.

Example 1:

```
<link rel="author" title="Boris Zbarsky" href="mailto:bzbarsky@mit.edu" />
```

Example 2:

```
<link rel="author" title="Bert Bos" href="http://www.w3.org/People/Bos/" />
```

Example 3:

```
<link rel="author" title="Microsoft" href="http://microsoft.com/" />
```

Reviewer

```
<link rel="reviewer" title="NAME_OF_REVIEWER" href="mailto:EMAIL OR http://CONTACT_PAGE" /> <!-- YYYY-MM-DD -->
```

If a test has passed review, then the reviewer should note this by adding his or her name as a reviewer, along with the date of the review. A test can have multiple reviewers if necessary. A reviewer must be a person, not an organization.

Example 1:

```
<link rel="reviewer" title="Boris Zbarsky" href="mailto:bzbarsky@mit.edu" /> <!-- 2008-02-19 -->
```

Example 2:

```
<link rel="reviewer" title="Bert Bos" href="http://www.w3.org/People/Bos/" /> <!-- 2005-05-03 -->
```

If a test would pass review with some (non-metadata) changes and the reviewer chooses to make these changes, then the reviewer should add his or her name as a reviewer-author, along with the date of the review, when checking in those changes. This indicates that the reviewer-author approves of the original author's test when taken with these proposed changes, and that someone else (possibly the original author) must review the changes. The test is fully reviewed only when the latest reviewer did not also contribute changes to the test at the time of the review.

Example of a fully-reviewed test:

```
<link rel="author" title="Bert Bos" href="http://www.w3.org/People/Bos/" />
<link rel="reviewer author" title="Boris Zbarsky" href="mailto:bzbarsky@mit.edu" /> <!-- 2008-02-19 -->
<link rel="reviewer" title="Bert Bos" href="http://www.w3.org/People/Bos/" /> <!-- 2008-04-22 -->
```

This test was written by Bert Bos, then reviewed by Boris Zbarsky, who made some corrections before deeming it acceptable. Those corrections were then reviewed and accepted by Bert Bos.

Specification Links

```
<link rel="help" href="RELEVANT_SPEC_SECTION" />
```

The specification link elements provide a way to align the test with information in the specification being tested.

- Links should link to relevant sections within the specification
- Use the anchors from the specification's Table of Contents
- A test can have multiple specification links
  - Always list the primary section that is being tested as the first item in the list of specification links
  - Order the list from the most used/specific to least used/specific
  - There is no need to list common incidental features like the color green if it is being used to validate the test unless the case is specifically testing the color green
- If the test is part of multiple test suites, link to the relevant sections of each spec.

Example 1:

```
<link rel="help" href="http://www.w3.org/TR/CSS21/text.html#alignment-prop" />
```

Example 2:

```
<link rel="help" href="http://www.w3.org/TR/CSS21/text.html#alignment-prop" />
<link rel="help" href="http://www.w3.org/TR/CSS21/visudet.html#q7" />
<link rel="help" href="http://www.w3.org/TR/CSS21/visudet.html#line-height" />
<link rel="help" href="http://www.w3.org/TR/CSS21/colors.html#background-properties" />
```

Reference Links

```
<link rel="match" href="RELATIVE_PATH_TO_REFERENCE_FILE" />
<link rel="mismatch" href="RELATIVE_PATH_TO_REFERENCE_FILE" />
```

The reference link elements are used in reftests and provide the list of reference file(s) that the test should be compared to.

- `match` references must be files that render identically to the test, but use an alternate means to do so
- Multiple `match` references are used when the test can match **any** of the reference files
  - If a test requires multiple `match` references that all need to match (for example, to catch when a reference fails in the same way the test does), then chain the references together, i.e.: place reference links to the additional `match` references in the reference files. It is recommended that the chained reference files form a loop (e.g.: a → b → c → a) so that a test linking to any reference in the chain will find all the references.
- `mismatch` references are files that render differently than the test file. A test may have any number of `mismatch` references. The test is considered to fail if it renders the same as **any** of the `mismatch` references.
  - Note that reference files may themselves have `mismatch` references. In that case the reference file must not render the same as any of its `mismatch` references in order to be considered valid. If a reference is considered invalid (by the fact of not matching any of its `match` references, or matching any of its `mismatch` references), then a test that refers to the reference will be considered to have failed.
- Reference files may be dedicated reference files, images, or other tests

Example 1:

```
<link rel="match" href="green-box-ref.xht" />
```

Example 2:

```
<link rel="match" href="green-box-ref.xht" />
<link rel="match" href="blue-box-ref.xht" />
<link rel="mismatch" href="red-box-notref.xht" />
<link rel="mismatch" href="red-box-notref.xht" />
```

Requirement Flags

```
<meta name="flags" content="TOKENS" />
```

Flags document the test's prerequisites, for example, the Ahem font, or a paged presentation. Multiple tokens are space separated. Flag tokens are case sensitive.

All flags documented elsewhere are deprecated except the following:

Token	Description
ahem	Requires the Ahem font [http://www.w3.org/Style/CSS/Test/Ahem/]
animated	Test is animated in final state. (Cannot be verified using reftests/screenshots.)
asis	The test has particular markup formatting requirements and cannot be re-serialized.
combo	Test, which must have an unsuffixed filename number, is strictly the union of all the suffixed tests with the same name and number. (See File name format, below.)
dom	Requires support for JavaScript and the Document Object Model (DOM)
font	Requires a specific font to be installed. (Details must be provided and/or the font linked to in the test description)
history	User agent session history is required. Testing `:visited` is a good example where this may be used.
HTMLonly	Test case is only valid for HTML
http	Requires HTTP headers
image	Requires support for bitmap graphics and the graphic to load
interact	Requires human interaction (such as for testing scrolling behavior)
invalid	Tests handling of invalid CSS. Note: This case contains CSS properties and syntax that may not validate.
may	Behavior tested is *preferred* but OPTIONAL. [RFC2119 [http://www.ietf.org/rfc/rfc2119.txt]] (These tests must be reviewed by a test suite owner or peer.)
namespace	Requires support for XML Namespaces
nonHTML	Test case is only valid for formats besides HTML (e.g. XHTML or arbitrary XML)
paged	Only valid for paged media
scroll	Only valid for continuous (scrolling) media

Token	Description
should	Behavior tested is RECOMMENDED, but not REQUIRED. [RFC2119 [http://www.ietf.org/rfc/rfc2119.txt]]
speech	Device supports audio output. Text-to-speech (TTS) engine installed
svg	Requires support for vector graphics (SVG)
userstyle	Requires a user style sheet to be set
32bit	Assumes a 32-bit integer as the minimum (-2147483648) or maximum (2147483647) value
96dpi	Assumes 96dpi display

Example 1 (one token applies):

```
<meta name="flags" content="invalid" />
```

Example 2 (multiple tokens apply):

```
<meta name="flags" content="ahem image scroll" />
```

Example 3 (no tokens apply):

```
<meta name="flags" content="" />
```

Test Assertions

```
<meta name="assert" content="TEST ASSERTION" />
```

This element should contain a complete detailed statement expressing what specifically the test is attempting to prove. If the assertion is only valid in certain cases, those conditions should be described in the statement.

The assertion should *not* be:

- A copy of the title text
- A copy of the test verification instructions
- A duplicate of another assertion in the test suite
- A line or reference from the CSS specification unless that line is a complete assertion when taken out of context.

The test assertion is **optional**. It helps the reviewer understand the goal of the test so that he or she can make sure it is being tested correctly. Also, in case a problem is found with the test later, the testing method (e.g. using 'color' to determine pass/fail) can be changed (e.g. to using 'background-color') while preserving the intent of the test (e.g. testing support for ID selectors).

Examples of good test assertions:

- “This test checks that a background image with no intrinsic size covers the entire padding box.”
- “This test checks that 'word-spacing' affects each space (U+0020) and non-breaking space (U+00A0).”
- “This test checks that if 'top' and 'bottom' offsets are specified on an absolutely-positioned replaced element, then any remaining space is split amongst the 'auto' vertical margins.”
- “This test checks that 'text-indent' affects only the first line of a block container if that line is also the first formatted line of an element.”

Style Element (embedded styles)

```
<style type="text/css"><![CDATA[
   CSS FOR TEST
  ]]></style>
```

When creating styles primarily use type, ID or Class selectors. Inline styles should not be used unless the case is specifically testing this scenario. Other selector types should also be avoided unless specifically testing those scenarios.

Script type (embedded script)

```
<script type="text/javascript"><![CDATA[
   ... Javascript code here ...
  ]]></script>
```

Some testcases require support for javascript and the Document Object Model (DOM).

Although type=“application/javascript” and type=“application/ecmascript” are recommended by [RFC4329 [http://www.ietf.org/rfc/rfc4329.txt]], the CSS 2.1 test suite only accepts type=“text/javascript”.

"body" Content

```
<body>
  <p>Test passes if [description of pass condition].</p>
  [Content of test]
</body>
```

- When creating content use : `<div>`, `<span>`, `<p>`, `<img>`
  - Beware! `<p>` has margins by default!
- Use other elements only to test the interaction with those elements' features (or other namespaces' features)
- Test `table` features with both HTML table elements and `<div>` + `display: table-*`
- Do not use the `style` attribute (inline styles) unless specifically testing that attribute
- Avoid making assumptions that are not in the "most of the test suite makes the following assumptions" [http://test.csswg.org/suites/css2.1/latest/#common] list and document any that you do.
- It's helpful to people trying to understand the test if you use meaningful class and ID names and escape invisible characters like no-break-spaces.
- Indent nicely! There's no standard for how many tabs/spaces to use; just be consistent within a file.
- In HTML5 tests, do not omit attribute quotation marks or closing tags, even when it's valid to do so.

Guidelines on designing the content of the test can be found in the CSS2.1 Test Case Authoring Guidelines [http://www.w3.org/Style/CSS/Test/guidelines.html].

File name format

The new file name format is `test-topic-###.ext` where `test-topic` somewhat describes the test and *###* is a zero-filled number used to keep the file names unique.

The file name is no longer restricted to 31 characters, but please try to keep them short.

The file name should not use the underscore ("_") character; please use the hyphen ("-") character instead.

test-topic
A short identifier that describes the test. The `test-topic` should avoid conjunctions, articles, and prepositions. It is a file name, not an English phrase: it should be as concise as possible.

Examples:
`margin-collapsing-###.ext`
`border-solid-###.ext`
`float-clear-###.ext`

###
This is a zero-filled number used to keep the file names unique when files have the same test-topic name.

Note: The number format is limited to 999 cases. If you go over this number it is recommended that you reevaluate your test-topic name.

For example, in the case of margin-collapsing there are multiple cases so each case could have the same test-topic but different numbers.
`margin-collapsing-001.xht`
`margin-collapsing-002.xht`
`margin-collapsing-003.xht`

There may also be a letter affixed after the number, which can be used to indicate variants of a test.

For example, `float-wrap-001l.xht` and `float-wrap-001r.xht` might be left and right variants of a float test.

If tests using both the unsuffixed number and the suffixed number exist, the suffixed tests must be subsets of the unsuffixed test.

For example, if `bidi-004` and `bidi-004a` both exist, `bidi-004a` must be a subset of `bidi-004`.

If the unsuffixed test is strictly the union of the suffixed tests, i.e. covers all aspects of the suffixed tests (such that a user agent passing the unsuffixed test will, by design, pass all the suffixed tests), then the unsuffixed test should be marked with the `combo` flag.

If `bidi-004a` and `bidi-004b` cover all aspects of `bidi-004` (except their interaction), then `bidi-004` should be given the `combo` flag.

ext
The file extension or format of the file, usually `.xht` for test files.

Support files

A number of standard images are provided in the support directory [http://www.w3.org/Style/CSS/Test/CSS2.1/current/xhtml1/support/]. These include

- 1×1 color swatches
- 15×15 color swatches
- 15×15 bordered color swatches
- assorted rulers and red/green grids
- a cat
- a 4-part picture

Additional generic files can be added as necessary, and should have a descriptive file name. Just like other file name, support files' file name should not use the underscore ("_") character; use the hyphen ("-") character instead. Test-specific files should be named after the test (or test-topic if they are shared across several tests within a series). If possible tests should not rely on transparency in images, as they are converted to JPEG (which does not support transparency) for the xhtml1print version.

User style sheets

Some test may require special user style sheets to be applied in order for the case to be verified.

In order for proper indications and prerequisite to be displayed every user style sheet should contain the following rules.

```
#user-stylesheet-indication
{
    /* Used by the harness to display and indication there is a user style sheet applied */
    display: block!important;
}
```

The rule `#user-stylesheet-indication` is to be used by any harness running the test suite.

A harness should identify test that need a user style sheet by looking at their flags meta tag. It then should display appropriate messages indicating if a style sheet is applied or if a style sheet should not be applied.

Harness style sheet rules:

```
#userstyle
{
    color: green;
    display: none;
}
#nouserstyle
{
    color: red;
    display: none;
}
```

Harness `userstyle` flag found:

```
<p id="user-stylesheet-indication" class="userstyle">A user style sheet is applied.</p>
```

Harness `userstyle` flag NOT found:

```
<p id="user-stylesheet-indication" class="nouserstyle">A user style sheet is applied.</p>
```

Within the test case it is recommended that the case itself indicate the necessary user style sheet that is required.

Examples: (code for the cascade.css file)

```
#cascade /* ID name should match user style sheet file name */
{
    /* Used by the test to hide the prerequisite */
    display: none;
}
```

The rule `#cascade` in the example above is used by the test page to hid the prerequisite text. The rule name should match the user style sheet CSS file name in order to keep this orderly.

Examples: (code for the cascade-### XHTML files)

```
<p id="cascade">PREREQUISITE: The <a href="support/cascade.css">"cascade.css"</a> file is enabled as the user agent's user style sheet.</p>
```

The `id` value should match the user style sheet CSS file name and the user style sheet rule that is used to hide this text when the style sheet is properly applied.

Please flag test that require user style sheets with the `userstyle` flag so people running the tests know that a user style sheet is required.

HTTP headers

Some tests may require special HTTP headers. These should be configured in a `.htaccess` file located in the same directory as the relevant file. An example configuration is shown below. Note that multiple file extensions are supported in the configuration so that exported formats are all handled correctly. The build scripts will concatenate all `.htaccess` files in the test sources' parent directories and support directories.

```
<Files ~ "^lang-selector-005\.(xht|xhtml|xml|html|htm)$">
AddLanguage fr .xht .xhtml .xml .html .htm
</Files>
```

Please flag tests that require HTTP interaction with the `http` flag so people running the tests locally know their results will not be valid.

test/format.txt · Last modified: 2014/12/09 15:48 (external edit)