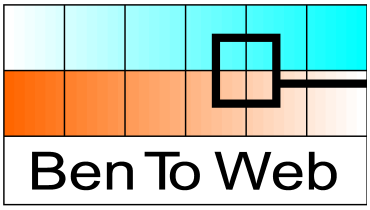
Benchmarking Tools and Methods for the Web (FP6—004275)



Sixth Framework Programme
Information Society Technologies Priority

D4.2 XHTML 1.0 and CSS 2.0 Test Suite

Contractual Date of Delivery to the EC:	30 September 2005 + 45 days
Actual Date of Delivery to the EC:	31 January 2006
Editor:	Christophe Strobbe (KULRD)
Contributors:	Christophe Strobbe (KULRD), Johannes Koch, Carlos A Velasco, Dirk Stegemann (FIT), Sandor Herramhof (i3s3), Kurt Weimann (MMC), Evangelos Vlachogiannis (AEGEAN)
Workpackage:	4
Security:	Public
Nature:	Report
Version:	C
Total number of pages:	37

Keywords: test suite, accessibility, Web Content Accessibility Guidelines (WCAG), HTML, XHTML, CSS, Web Accessibility Initiative (WAI), World Wide Web Consortium (W3C).

DOCUMENT HISTORY			
Version	**Version date**	**Responsible**	**Description**
A	5 January 2006	KULRD	First draft
B	11 January 2006	KULRD	Addition of TCDL mapping; editorial changes; executive summary.
C	31 January 2006	KULRD + FIT	Final version approved by the board

Table of Contents

List of Tables

1 Executive Summary

This report complements the XHTML 1.0 + CSS 2.0 test suite, which is a publicly available deliverable. The test suite consists of more than 480 test cases for the 30 June 2006 Working Draft of WCAG 2.0.[1] Each test case consists of one or more XHTML files and accompanying metadata. The test cases map directly to WCAG 2.0 success criteria instead of tests that sit between the test files and the success criteria. Some success criteria are not covered because they were unfinished at the time when the test suite was developed; some other success criteria were open to interpretation. These issues are documented in Appendix 1 of this report. The test case metadata are written in an XML language developed specially for this purpose: TCDL (Test Case Description Language). TCDL metadata include administrative metadata, technologies used in the test files, test modes and test scenarios, rules and expected results, and namespace mappings. Another language, Rulesets XML, serves as a global map of success criteria and other "rule sets" (Section 508,[2] BITV,[3] ...) and allows TCDL to reference WCAG 2.0 success criteria and other rules indirectly.

[1] http://www.w3.org/TR/2005/WD-WCAG20-20050630/
[2] http://section508.gov/
[3] Barrierefreie Informationstechnik-Verordnung – BITV (Decree on Accessible Information Technology): http://bundesrecht.juris.de/bundesrecht/bitv/index.html

2 The XHTML + CSS Test Suite

2.1 Purpose and General Description

BenToWeb's XHTML + CSS test suite is a suite of test files and accompanying metadata for the Web Content Accessibility Guidelines 2.0 (WCAG 2.0). The test suite serves two purposes:

- a test suite is a way of proving the testability of a specification; testability is a requirement for specifications in the "Recommendation track" of the W3C Process, but it is possible that the W3C decides that complete web sites – not test suites – are need to prove that WCAG 2.0 can be implemented[4];
- a test suite can also serve as a basis for benchmarking Web accessibility evaluation software (also known as ERT): comparing the ERT's evaluation report with the test case metadata shows which WCAG 2.0 success criteria are supported by the tool and where the tool produces false positives or negatives.

The XHTML + CSS test suite contains over 470 test cases and over 500 test files. Each test case covers one success criterion, with at least one test that fails the success criterion and at least one that passes. The test files use XHTML 1.0; CSS 2.0 is also used where this is relevant to the success criterion.

The test cases map to the 30 June 2005 Working Draft of WCAG 2.0, the current draft at the time when the test suite was developed. Some success criteria in that draft were unfinished or open to interpretation; these issues are discussed in "Open Issues", below. Newer drafts may change, move, delete or split success criteria, introduce new success criteria, adopt another numbering system, etc. This is why it was necessary to map the test cases to a specific draft of WCAG 2.0 instead of "WCAG 2.0 generally". This also means that the mapping of the test cases to success criteria in the final WCAG 2.0 recommendation will be different from the current one.[5]

There are several types of test cases:

- simple test cases use one file to test one success criterion;

[4]See "Call for Implementations" in the "World Wide Web Consortium Process Document": http://www.w3.org/2005/10/Process-20051014/tr.html#cfi.
[5]In the new version of the Technical Annex, this deliverable, and in particular the test suites will be updated within the framework of a Task Force in the WCAG Working Group.

- compound test cases use multiple files to test one success criterion;
- complex test cases use one or more files test test multiple success criteria.

The XHTML + CSS test suite does not contain complex test cases, even though a test case that is intended to pass a certain success criterion may fail another success criterion that is not relevant to the test. The test case metadata (TCDL, discussed below) provide a means to explicitly indicate that some other specific success criteria are not relevant to a test case. For example: a test case for blinking content (Guideline 2.2 Level 2 Success Criterion 1 in WCAG 2.0) might use the `<blink>` element, which is not valid XHTML; in this case, the metadata can indicate that validation is not a relevant criterion for the test.

Some success criteria have more than 50 test cases, others have less then 5. The number of test cases for a success criterion can depend on several things, for example:

- the number of XHTML elements and attributes and/or the number of CSS properties that are relevant to a success criterion (e.g., GL 1.3 L1 SC3: "Structures within the content can be programmatically determined" applies to many XHTML elements);
- for Guideline 3.1 ("Make text content readable and understandable") the number of natural languages, the size of vocabularies and other linguistic features make the number of test cases theoretically infinite;
- for Guideline 1.4 ("Make it easy to distinguish foreground and background images or sound"), the number of variations in contrast is almost endless, so finding edge cases is more important than exhaustive coverage.

After test cases are created, they can be evaluated in a number of ways:

- by end users,
- by one or more accessibility experts,
- automatically.

The test case metadata define one or more "test modes" for the test files. The choice of test mode depends on the reliability of code inspection as a method to determine the accessibility issues of the test case.

The BenToWeb XHTML 1 + CSS 2 test suite is publicly available through anonymous CVS access at cvs.fit.fraunhofer.de:2401/bentoweb

(username: anonymous; password: anonymous).[6] A Web interface is also available at: http://bentoweb.org/ts/XHTML1/home.

2.2 Open Issues

2.2.1 Unfinished Success Criteria

In the 30 June 2005 Working Draft of WCAG 2.0, the following success criteria were unfinished:

- GL 1.4 L2 SC1: "Text and diagrams that are presented over a background image, color, or text have a contrast greater than X1 where the whiter element is at least Y1 as measured by _____."
- GL 1.4 L2 SC2: "Text that is presented over a background pattern of lines which are within 500% +/- of the stem width of the characters or their serifs must have a contrast between the characters and the lines that is greater than X2, where the whiter element is at least Y2."
- GL 1.4 L3 SC1: "Text is not presented over any background (image, text, color or pattern), or if any background is present, the contrast between the text and the background is greater than X2."
- 2.3 L1 SC1: "Content that violates Computer adapted ITC Guidance for general flash or Computer adapted ITC Guidance for red flash is marked in a way that the user can avoid its appearance." (The definitions of the general flash threshold and the red flash threshold were incomplete.)
- 2.3 L2 SC1: "Content does not violate Computer adapted ITC Guidance for general flash or Computer adapted ITC Guidance for red flash." (The definitions of the general flash threshold and the red flash threshold were incomplete.)
- 2.3 L3 SC1: "Content does not violate Computer adapted spatial pattern thresholds or red flash." (The definition of "Computer adapted spatial pattern thresholds" was incomplete because the international standards that WCAG wanted to adopt was changing.)
- 4.2 L1 SC2 depended on the text in the Guideline 2.3 success criteria.

In the test case metadata, this is handled by indicating that the expected evaluation result cannot be predicted (expectedResult="cannotTell"). User

[6]Connections need to be made via STUNNEL: http://www.stunnel.org/

testing can reveal whether the relevant test files have accessibility issues, and these issues can be documented in TCDL (and in BenToWeb deliverable 3.7a) and fed back to the WCAG Working Group.

More details on unfinished success criteria are available in Appendix 1.

2.2.2 Interpretation of Success Criteria

In the 30 June 2005 Working Draft of WCAG 2.0, the following success criteria were open to interpretation:

- 3.2 L1 SC1: "Any change of context is implemented in a manner that can be programmatically determined."
- 3.2 L2 SC2: "When any component receives focus, it does not cause a change of context."
- 3.2 L2 SC3: "Changing the setting of any input field does not automatically cause a change of context."
- 3.2 L3 SC1: "Changes of context are initiated only by user action."

The intention of these success criteria is not to confuse the user (e.g., 3.2 L2 SC2-3) or even to give users full control over changes of context (3.2 L3 SC1), but some changes of context are quite harmless while others are not. It is not clear which user action may result in which change of context without violating WCAG 2.0. The table below, which was sent to the mailing list of the WCAG Working Group,[7] tries to define which cases are covered.

[7] See http://lists.w3.org/Archives/Public/w3c-wai-gl/2005JulSep/att-0673/changes_of_context.html.

user actions (events)	**changes of context**							
	open new browser	open another type of UA	close browser	move viewport	change size of viewport	scroll viewport	change focus	load completely new content
tab (keypress, blur, focus)	3.2 L2 SC2	3.2 L2 SC2	3.2 L2 SC2	3.2 L2 SC2	3.2 L2 SC2	3.2 L2 SC2	3.2 L2 SC2	3.2 L2 SC2
activate link (click, keypress)								
change changeable form control (change)	3.2 L2 SC3	3.2 L2 SC3	3.2 L2 SC3	3.2 L2 SC3	3.2 L2 SC3	3.2 L2 SC3	3.2 L2 SC3	3.2 L2 SC3
fill text form control (keypress)								
mark text in text form control (select)								
activate button (click, keypress)								
activate reset button (click, keypress, reset)								
activate submit button (click, keypress, submit)								
hover mouse (mousemove, mouseout, mouseover)								
other (dblclick, load, unload)								

Table 1. User actions and changes of context.

More details on the interpretation of certain success criteria are available in Appendix 1.

2.2.3 Success Criteria Not Covered by the Test Suite

Some success criteria are not covered by the test suite:

- 2.2 L3 SC2: "When an authenticated session has an inactivity time-out, the user can continue the activity without loss of data after re-authenticating." There is currently no technique for this success criterion and it is not obvious how it can be implemented.
- 3.2 L1 SC1: "Any change of context is implemented in a manner that can be programmatically determined." A change of context that cannot be programmatically determined will never happen, because software has no mechanism to detect and execute it. Consequently, this success criterion is always fulfilled. It was removed from the 23 November 2005 Working Draft of WCAG 2.0.
- 4.2 L1 SC1 and 4.2 L3 SC1 depend on the baseline, whereas the test suite is not targeted to any specific baseline. For each individual test case, the combination of technologies used by the test file(s) constitutes an implicit baseline; and the baseline can be inferred from the technology section in the test case metadata.

One success criterion that was not in the 30 June 2005 Working Draft but that was included in previous working drafts, is covered by the test suite because it addresses an issue that was still not closed: 4.1 L3 SC1 from the 11 February Working Draft[8] ("Technologies are used according to specification without exception.") has several test cases for well-formedness and validity of XHTML files.

2.2.4 Other Issues

Like the other guidelines, Guideline 2.3 ("Allow users to avoid content that could cause seizures due to photosensitivity") requires test files that violate the underlying success criteria. For obvious ethical reasons, the evaluation of these test cases must not cause seizures in participants of the user panel, so these tests are set aside for evaluation by accessibility experts.

[8] http://www.w3.org/WAI/GL/WCAG20/WD-WCAG20-20050211/.

2.3 Differences with the Current WCAG 2.0 Test Suite

One of the requirements for WCAG 2.0 is that "things that are normative must be testable."[9] Both the BenToWeb test suite and the current HTML Test Suite for WCAG 2.0[10] aim to prove the testability of the guidelines and success criteria of WCAG 2.0, but they do so in different ways.

In the BenToWeb test suite, test cases (test files with metadata) map directly to success criteria in WCAG 2.0; the current HTML Test Suite for WCAG 2.0 defines a number of tests for each guideline, and these tests are the bridge between the guidelines and the test files. Examples of tests for guideline 1.1 ("Provide text alternatives for all non-text content") include:

- All img elements have an alt attribute.
- Alt text for all img elements is the empty string ("") if the image is decorative.
- Alt text for all img elements that are not used as source anchors conveys the same information as the image.
- All area elements have an alt attribute.

BenToWeb does not define tests, although the description in the test case metadata can often be turned into a test.

The WCAG Working Group also defines techniques for a number of technologies (HTML/XHTML, CSS and client-side scripting); the HTML Techniques document provides links to tests in the HTML test suite if relevant test are available. BenToWeb does not define techniques and tests, although the techniques can be inferred from the description of the test cases (the description element of TCDL) and documented (in the technicalComment element in TCDL). The direct relationship between the WCAG 2.0 success criteria and the test cases makes it easy to check if test cases for each success criterion exist and, consequently, if the success criteria are implementable.

[9] Requirements for WCAG 2.0 (W3C Working Draft 26 April 2002): http://www.w3.org/TR/wcag2-req/#normative.
[10] http://www.w3.org/WAI/GL/WCAG20/tests/.

3 TCDL: Test Case Description Language

3.1 Purpose and General Description

Test Case Description Language (TCDL) is an XML-based language defined by BenToWeb to store metadata for the test cases in the test suites. For each individual test cases, it describes:

- formal metadata, such as author, date, title and a short description;
- technologies (the technologies and technology features used in the test);
- test mode and test scenarios for end user evaluation;
- the "rules" that are tested (WCAG 2.0 success criteria or other rules);
- namespace mappings used in some of the previous sections.

TCDL is defined in W3C XML Schema[11] and uses the namespace `http://bentoweb.org/refs/TCDL1.1`; the XML Schema is available at http://bentoweb.org/refs/schemas/tcdl1.1.xsd and contains detailed documentation. The next section provides a less technical description of TCDL. It also shows which metadata elements TCDL has in common with those suggested in the W3C Working Group Note on Test Metadata[12] by adding "W3CTM" to the description.

3.2 Structure of TCDL

Each test case has a unique identifier (stored as a mandatory ID attribute of the test case description; W3CTM) that uses the format `technology_wcag2_date_x.y_lz_scz_nnn`, where the sections separated by underscores have the following meaning:

- `technology`: the technology for which the test suite is developed: xhtml1, xhtml2, xforms1, etcetera,
- `wcag2`: a constant referring to WCAG 2.0,

[11] XML Schema is defined at http://www.w3.org/TR/2004/REC-xmlschema-1-20041028/ and http://www.w3.org/TR/2004/REC-xmlschema-2-20041028/.
[12] http://www.w3.org/TR/2005/NOTE-test-metadata-20050914/.

- `date`: the date on which the WCAG 2.0 draft was published, in yyyymmdd format,
- `x.y`: the number of WCAG 2.0 guideline, for example 1.1, 1.2, …
- `lz`: the WCAG 2.0 level: l1 | l2 | l3,
- `scz`: the number of the WCAG 2.0 success criterion, for example sc1, sc2, …
- `nnn`: the number of the test case (for this success criterion, not the number in the complete test suite), for example 001, 002, …

3.2.1 Formal Metadata

The formal or administrative metadata that can be stored are:

- description: a detailed description of the test materials and how they behave in user agents (mandatory; W3CTM);
- title (mandatory; W3CTM);
- author (mandatory; W3CTM);
- language (mandatory);
- rights (mandatory; W3CTM) – usually BenToWeb copyright;
- creation date (mandatory);
- status (mandatory; W3CTM): "draft", "pending bugfix", "accepted for end user evaluation", "evaluated by expert", "evaluated by end user" or "accepted";
- version (optional);
- source (optional): if the test file is borrowed from an existing test suite, the test suite and the URL of the original test file can be documented; in addition, it is possible to indicate if the borrowed file was modified and to document these modifications and other comments, and rights (for example copyrights; W3CTM) related to the borrowed file.

3.2.2 Technologies

BenToWeb uses the term "technology" in the same sense as WCAG 2.0: "a data format, programming or markup language, protocol or API." TCDL provides the ability to store the following information about technologies used in a test:

- the formal specification or recommendation that defines the technology (by name and URL; mandatory);

- the technology features (for example, which XHTML elements and/or attributes) that are relevant to the test (optional);
- a reference (URL) to a section in the formal specification or a quote from the formal specification that describes the technology features that are relevant to the test (optional).

3.2.3 Test Case

3.2.3.1 Purpose

This mandatory section provides a brief explanation of the reason why the test was developed: why the test materials pass or fail the success criterion for which the test was developed (W3CTM). It sometimes states explicitly that certain aspects of the test materials are not meant to be tested, for example because they relate to another success criterion. The description (see above), combined with the test materials, and the purpose are the most important information for the evaluators of the test case; test cases are usually reviewed through expert evaluation or end-user evaluation.

3.2.3.2 Preconditions

The W3C Note on Test Metadata defines preconditions as "conditions that must be met before the test can be successfully executed." This section was added to TCDL for other users of the language. It is not used in BenToWeb because test cases map to success criteria instead of tests. However, the current HTML Test Suite for WCAG 2.0 uses "prerequisite tests", and these could be described here.

3.2.3.3 Required Tests

This section describes:

- the test mode or modes (mandatory): the ways in which the test case should be evaluated to advance the test case from "draft" status to "accepted"; test modes are at least one of the following: end user, one accessibility expert, a group of accessibility experts or automatic evaluation;
- one or more test scenarios for end user evalution (strictly optional, but required if the test mode is "end user"): a scenario consists of user guidance (for example, advice on features of a user agent that should be supported by and enabled in a browser), questions that the user is expected to answer (yes/no

question, open question, Likert scale, multiple choice, or a combination of yes/no and open question), the experience that the user should have with certain user agents or assistive technologies to evaluate the test, and, optionally, disabilities to which the test is relevant.

Each scenario has a mandatory identifier that is unique within the test case description. This identifier is necessary to map end user feedback to the scenarios.

3.2.3.4 Test Files

This section contains information about each of the test files that are relevant to the rule or success criterion:

- the URL of the test file (optional);
- the language of the test file (optional);
- data to create an HTTP request (optional): media type, encoding, HTTP headers and HTTP body.

Supporting files such as included images, external style sheets and dummy pages to which forms are submitted (if only the form is relevant to the test) etcetera, are not documented.

3.2.4 Rules

This section contains the "rules" that are tested by a test case. In BenToWeb, rules are generally WCAG 2.0 success criteria, but it is also possible to reference other sets of "rules":

- WCAG 1.0 checkpoints,
- Section 508 provisions,
- BITV provisions.

The rules do not provide a direct reference to the relevant online source because it may not always be possible to reference "rules" directly (for example, because they are not available in HTML). The rules reference a rule ID in a "Rulesets XML" file (see Chapter 4 below), which in turn references the relevant rule in an online source.

For each rule listed in this section, the following information can be provided:

- whether the rule is the "primary" rule of the test case or only listed for informative purposes (for example, some test cases uses invalid markup in tests that are not about validity, so it is

useful to document that checkpoints about validity should not be applied to the test case) (optional);

- the identifier of the rule in "Rulesets XML" (mandatory);
- the locations in the test file that are relevant to the rule, with URL, line number, column number and/or XPath[13] expression (optional);
- whether the test case passes or fails the success criterion or rule;
- the functional outcome of the test: a description of why the test case passes or fails, in terms that relate to a user's experience (as opposed to technical comments about source code) (mandatory);
- technical comment on the test: a technical description of why the test case passes or fails, and other technical information (for example, issues in certain browsers or assistive technologies).

3.2.5 Namespace Mappings

This optional section is important for test cases based on test files that use XML-based technologies. The technology features (see the section on Technologies above) used in a test case exist in an XML namespace[14] and should be referenceable with XPath. The normal mechanism for mapping namespaces to URIs by declaring them with xmlns:prefix= "namespaceURI" cannot be used for this, because it is necessary to be able to define empty prefixes and to extract the namespace mappings with a common XML API.

3.3 TCDL and W3C Test Metadata

The W3C Working Group Note "Test Metadata" (14 September 2005)[15] suggests fourteen metadata elements for test suites. The table below shows how these metadata elements map to TCDL elements. (TCDL elements are identified by means of XPath expressions without a namespace prefix and with the root element – testCaseDescription – omitted, except for the attributes of that element.)

[13] A syntax and semantics for addressing parts of an XML document. XML Path Language (XPath) Version 1.0 – W3C Recommendation: http://www.w3.org/TR/1999/REC-xpath-19991116.
[14] Namespaces in XML – W3C Recommendation: http://www.w3.org/TR/1999/REC-xml-names-19990114/.
[15] http://www.w3.org/TR/2005/NOTE-test-metadata-20050914/.

W3C Metadata	TCDL	Comment
Identifier	`@id`	
Title	`formalMetadata/title`	In TCDL, titles are not necessarily unique; they don't duplicate the identifier.
Purpose	`testCase/purpose`	
Description	`formalMetadata/description`	In TCDL, the description does not require the same level of detail.
Status	`formalMetadata/status`	
SpecRef	`rules/rule` and `technology/recommendation`	In TCDL, two types of specifications need to be referenced: the accessibility guidelines ("rules") and the specifications of the technologies in the test.
Preconditions	`testCase/preconditions`	Not used in BenToWeb, because test cases map directly to WCAG success criteria instead of tests.
Inputs	`testCase/files/file/httpRequest`	Only for the special case where it is necessary to (re)create a specific HTTP request.
ExpectedResults	`rules/rule/locations/@exptectedResult` and `rules/rule/functionalOutcome`	
Version	`formalMetadata/version`	Not used in BenToWeb; a version control system (CVS) is used instead.
Contributor	–	BenToWeb does not have external contributors, but can borrow test files and document them at `formalMetadata/source`.

W3C Metadata	TCDL	Comment
Rights	`formalMetadata/dc:rights` and `formalMetadata/source/dc:rights`	TCDL can document rights related to a test case and rights related to borrowed materials.
Grouping	-	(Because of the strict filename convention in BenToWeb, some types of grouping can be done by means of filename pattern matching.)
SeeAlso	`technology/recommendation` (and `rules/rule/techComment`)	

Table 2. Mapping of W3C Test Metadata to TCDL.

3.4 Differences with IBM's TCDL Submission to W3C QA

BenToWeb's TCDL was developed completely independently from the TCDL proposal submitted by IBM to the W3C Quality Assurance Working Group (QA WG) in 2003.[16] The goal of IBM's TCDL is to catalogue most of the test materials that a W3C Working Group would provide. It is intended to be used in a test lab "to set up and run the test materials against one or more test subjects" and "supports automated setup of the system(s) used for testing, automated running of test cases, automated comparison of results, and automated cleanup of the system(s)". Like BenToWeb's TCDL, IBM's TCDL is an XML vocabulary, so metadata can be tranformed in other formats for human or machine consumption.

An IBM TCDL catalogue contains a list a test cases, whereas a BenToWeb TCDL contains metadata for an individual test case only. BenToWeb's TCDL is also different because each test case references at least two specifications (WCAG 2.0 or another ruleset, and a specification of a technology such as XHTML or CSS), and one of these (WCAG 2.0) cannot be tested fully automatically.

[16] Test Case Description Language 1.0 – Submission to the QA Working Group 12 October 2003:http://www.w3.org/QA/WG/2003/10/tcdl-20031012.html.

Unfortunately, it is not possible to make a detailed comparison of the two languages because the XML Schema for IBM's TCDL submission is not available (the link to the Schema in Section 7.4 does not lead to a file).

4 Rulesets XML

As mentioned in Section 3.2.4 above, the WCAG 2.0 success criteria (or "rules") are not referenced directly in TCDL. BenToWeb has defined an XML vocabulary for "accessibility rulesets" such as WCAG, Section 508 and BITV. A file called "rulesets.xml", available at http://bentoweb.org/refs/rulesets.xml, serves as a global map of guidelines, success criteria, checkpoints etcetera that are relevant to the development of accessibility test suites. XHTML versions of Section 508 and BITV are available at http://bentoweb.org/refs/section508_1194.22.html and http://bentoweb.org/refs/bitv_anlage1.html, respectively.

In `rulesets.xml`, a ruleset consists of guidelines and a guideline may consist of success criteria. Depending on the ruleset, guidelines and success criteria refer to:

- guidelines and success criteria (WCAG 2.0); principles are not referenced;
- guidelines and checkpoints (WCAG 1.0);
- Anforderungen and Bedingungen (BITV);
- rules (Section 508 only knows rules).

The value of the `name` attribute of a `guideline` element corresponds to

- the guideline number in WCAG 1.0 [x] & WCAG 2.0 [x.y],
- the number of an Anforderung in BITV [x],
- the letter identifying a rule in Section 508.

The `id` attribute of the `guideline` element is required; `name` cannot serve this purpose because IDs must not start with a number.

For each guideline, all success criteria or checkpoints are listed as `successCriterion` elements. The value of the `name` attribute of a `successCriterion` element corresponds to

- a checkpoint number in WCAG 1.0 [x.y],
- a success criterion number in WCAG 2.0 [y] — the `level` attribute is required,
- the number of a Bedingung in BITV [x.y],
- nothing in Section 508 because the rules have no subdivisions.

The `level` attribute is optional except for WCAG 2.0.

5 Appendix 1. Interpretation of Success Criteria

This appendix contains further information on how success criteria that were unfinished or open to interpretation were dealt with.

5.1 1.1 L1 SC1-4 Text alternatives for non-text content

5.1.1 Definitions (WCAG 2.0):

Non-text content: Content that is not represented by a Unicode character or sequence of Unicode characters.

> Note: this is interpreted to mean: "Content that is not represented by a character included in the Unicode / ISO/IEC 106464 repertoire or (by) a sequence of characters included in the Unicode / ISO/IEC 106464 repertoire."
>
> Note: in the 23 November 2005 Working Draft, the definition was rewritten as follows: "Content that is not represented by a Unicode character or sequence of Unicode characters when rendered in a user agent according to the formal specification of the content type. This includes ASCII Art, which is a pattern of characters."

Text alternative: Programmatically determined text that is used in place of non-text content or text that is used in addition to non-text content and referred to from the programmatically determined text.

5.1.2 Interpretation

On the WCAG Working Group mailing list, there was a discussion about whether text alternatives should be explicitly associated with the non-text content or not. Examples of explicit association are:

- the alt and title attributes of the img element;
- the longdesc attribute of the img element.

For example, some find it acceptable to leave the alt attribute of an img element empty and place the text alternative next to or immediately

below the image. For the purpose of the test suite, this practice was considered to be acceptable if

- the image is part of a link and the text alternative is used as link text; or
- the text alternative is adjacent to the image.

5.2 1.3 L3 SC1 When content is arranged in a sequence that affects its meaning, that sequence can be determined programmatically.

5.2.1 Definitions (WCAG 2.0):

programmatically determined: programmatically determined means that the specific value can be determined in a standard, machine or software readable form.

5.2.2 Interpretation:

Sequence of content in (X)HTML can be programmatically determined. However the sequence in a presentation can be changed via CSS (positioning) or scripting techniques (DOM manipulation). Whether this changed sequence can be programmatically determined depends on the interpretation of "standard, machine or software readable form".

In Bentoweb, we assume that CSS and JavaScript are machine readable. So the information "can be determined in a standard, machine or software readable form".

5.3 1.4 L1 SC1 Any text that is presented over a background image, color, or text can be programmatically determined.

5.3.1 Definitions (WCAG 2.0):

background image: Images that appear behind or to the back of the visual field.

programmatically determined: programmatically determined means that the specific value can be determined in a standard, machine or software readable form.

5.3.2 Interpretation:

Any text in (X)HTML is part of the DOM and so can be programmatically determined. It does not matter whether the text is presented over a background image, color, or text.

5.4 1.4 L2 SC1 Text and diagrams that are presented over a background image, color, or text have a contrast greater than X1 where the whiter element is at least Y1 as measured by _____.

5.4.1 Definitions (WCAG 2.0):

background image: Images that appear behind or to the back of the visual field.

5.4.2 Interpretation:

As there are no precise values according to some algorithm, we interpret this success criterion according to the algorithm and threshold values used by imergo. The contrast of text over background color can be calculated according to this algorithm.

For images it could be possible to calculate the contrast of each pixel of a screen shot with its surrounding pixels.

But it may not be sufficient to do this calculation on a pixel basis, because humans average colors and brightness over some area. So at least for the combinations of text over image or text, and diagrams over image, color or text it is questionable, if it is really testable.

5.5 1.4 L2 SC2 Text that is presented over a background pattern of lines which are within 500% +/- of the stem width of the characters or their serifs must have a contrast between the characters and the lines that is greater than X2, where the whiter element is at least Y2.

5.5.1 Interpretation:

In addition to what was said about contrast calculation in 1.4 L2 SC1: How to calculate the stem width of a character or its serif?

5.6 1.4 L2 SC3 A mechanism is available to turn off background audio that plays automatically.

5.6.1 Interpretation:

The mechanism must be available in the content, not a feature of the user agent UI. Background audio is interpreted as an audio file that is played by a player without any controls presented to the user. This implies scripting techniques to turn off the audio or remove the element that incorporates the audio from the current document (DOM manipulation).

5.7 1.4 L3 SC1 Text is not presented over any background (image, text, color or pattern), or if any background is present, the contrast between the text and the background is greater than X2.

5.7.1 Interpretation:

Text in (X)HTML is always presented over a background. So this SC is already covered by 1.4 L2 SC1.

5.8 2.4 L1 SC1 Navigational features can be programmatically identified.

5.8.1 Definitions (WCAG 2.0):

navigational features: Something that allows the user to locate and/or move to a different piece of content than the one currently with focus. Examples of navigational features are links and components that are grouped so they can be bypassed.

programmatically identified: Editorial Note: We need to define this term.

5.8.2 Interpretation:

Single elements providing navigation are

- `a/@href`
- `area/@href`
- `img/@longdesc`
- `form/@action`
- `link/@href`
- `object/@data`

They can be identified from markup. However navigational features can be added to elements via scripting (DOM manipulation). But because this is machine readable, it can at least "be programmatically determined".

In (X)HTML there is only one standard way to group collections of single navigational elements: the `map` element. Additionally there may be heuristics to find other types of groups of single navigational elements. These groups need link starting just before the group and pointing to a location just after the group. This may "be programmatically determined".

5.9 3.2 L1 SC1 Any change of context is implemented in a manner that can be programmatically determined.

5.9.1 Definitions (WCAG 2.0):

change of context: A change of user agent, viewport, user interface controls, or focus; or complete change of content.

programmatically determined: programmatically determined means that the specific value can be determined in a standard, machine or software readable form.

5.9.2 Interpretation:

For our interpretation of "change of context" see Section 5.16 below. Whether this change can be programmatically determined depends on the interpretation of "standard, machine or software readable form".

In BenToWeb, we assume that (X)HTML and JavaScript are machine readable. So a change of context "can be determined in a standard, machine or software readable form".

5.10 3.2 L2 SC1 Components that are repeated on multiple delivery units within a set of delivery units occur in the same order each time they are repeated.

5.10.1 Definitions (WCAG 2.0):

delivery unit: A set of material transferred between two cooperating web programs as the response to a single HTTP request. The transfer might, for example, be between an origin server and a user agent.

5.10.2 Interpretation:

The order must be the same in the source code (the resulting DOM) and in the presentation (when changed via CSS or scripting).

5.11 3.2 L2 SC2 When any component receives focus, it does not cause a change of context.

5.11.1 Definitions (WCAG 2.0):

change of context: A change of user agent, viewport, user interface controls, or focus; or complete change of content.

5.11.2 Interpretation:

When a component receives focus, a "change of context" can only appear, when a script is executed `onfocus`. For our interpretation of "change of context" see Section 5.16 below.

5.12 3.2 L2 SC3 Changing the setting of any input field does not automatically cause a change of context.

5.12.1 Definitions (WCAG 2.0):

change of context: A change of user agent, viewport, user interface controls, or focus; or complete change of content.

5.12.2 Interpretation:

The setting of an input field can be changed, when

- a radio button is selected (`input[@type='radio']`)
- a checkbox is checked (`input[@type='checkbox']`)
- a select menu option is selected (`option`)
- text within a text or password field or a text area is changed (`input[not(@type) or @type='text']`, `input[@type='password']`, `textarea`)
- the value of a file input is changed (`input[@type='file']`)

and a script is executed `onchange`.

For our interpretation of "change of context" see Section 5.16 below.

5.13 3.2 L2 SC4 Components that have the same functionality in multiple delivery units within a set of delivery units are labeled consistently.

5.13.1 Definitions (WCAG 2.0):

delivery unit: A set of material transferred between two cooperating web programs as the response to a single HTTP request. The transfer might, for example, be between an origin server and a user agent.

5.13.2 Interpretation:

Functionality is interpreted as with respect to the user, not on a technical level. E.g. a form for submitting a purchase in a shop and another form for sending a mail both are forms on a technical level. But for the user they are different. So the submit buttons in both forms would not have to be labeled consistently.

Functional components are:

Functional component	Label text
`a[@href]`	content + `img/@alt`
`area`	`@alt`
`button`	content + `img/@alt`
`img[@ismap]`	`@alt`
`input[@type='submit']`	`@value`
`input[@type='reset']`	`@value`
`input[@type='button']`	`@value`
`input[@type='image']`	`@alt`
`link`	`@title`
any element with an `onclick` or `onkeypress` event handler	content + `img/@alt`

Table 3 Functional components and label text

5.14 3.2 L3 SC1 Graphical components that appear on multiple pages, including graphical links, are associated with the same text equivalents wherever they appear.

5.14.1 Definitions (WCAG 2.0):

link: Link refers to a hyperlink between the current document and a single destination. (Here, "link" refers to a single "arc" in the XML Linking Language (XLink) Version 1.0 specification.) Only links which are available to be activated by the user need to meet accessibility requirements. This excludes links that are activated automatically or programmatically.

5.14.2 Interpretation:

Because the proper text equivalent of a graphical component depends on its function within the context, this requirement only applies to graphical components that have the same function within the context.

Graphical components:

Graphical component	Text alternative
`area`	`@alt`
`img`	`@alt`
`input[@type='image']`	`@alt`
`object[starts-with(@type, 'image/')]`	content + `img/@alt`

Table 4. Graphical components and text alternatives.

5.15 3.2 L3 SC2 Changes of context are initiated only by user action.

5.15.1 Definitions (WCAG 2.0):

change of context: A change of user agent, viewport, user interface controls, or focus; or complete change of content.

5.15.2 Interpretation:

See the following table for types of "change of context" that are allowed for each type of user action. For our interpretation of "change of context" see Section 5.16 below.

user action	may change				
	user agent	**viewport**	**UI controls**	**focus**	**content**
tab through links				yes	
activate link	yes [17]	yes[18]		yes	yes
tab through form controls				yes	
select option				yes	
fill form control					
submit form				yes	yes
reset form				yes	
hover with mouse					

Table 5. User actions and changes of context.

5.16 What is change of context with respect to (X)HTML, CSS and JavaScript?

5.16.1 user agent:

- open new browser window
    - `@target`

[17]Content that cannot be handled by current UA (external player)

[18]Resource-internal navigation → automatic scrolling

    - `window.open(name, url)`

- use external program (multimedia player, PDF reader, office tool). This depends on media type and whether the user agent can handle content with this media type natively.

  markup indicating new requestable resources:

    - `a/@href`
    - `area/@href`
    - `img/@longdesc`
    - `form/@action`
    - `link/@href`
    - `object/@data`
    - `(q/@cite)`
    - `(blockquote/@cite)`
    - `(del/@cite)`
    - `(ins/@cite)`
    - `(frame/@longdesc)`
    - `(iframe/@longdesc)`

  media types we assume a UA can handle natively:

    - `text/plain`
    - `text/html`
    - `image/gif`
    - `image/jpeg`
    - `image/png`

- close window with JavaScript
    - `window.close()`

5.16.2 viewport:

- move window with JavaScript

    - `window.moveBy(`*`horizontal`*`, `*`vertical`*`)`
    - `window.moveTo(`*`x-coordinate`*`, `*`y-coordinate`*`)`
- change size with JavaScript
    - `window.resizeBy(`*`horizontal`*`, `*`vertical`*`)`
    - `window.resizeTo(`*`outerWidth`*`, `*`outerHeight`*`)`
    - `window.innerHeight`
    - `window.innerWidth`
    - `window.outerHeight`
    - `window.outerWidth`
- scroll with JavaScript
    - `window.scrollBy(`*`horizontal`*`, `*`vertical`*`)`
    - `window.scrollTo(`*`x-coordinate`*`, `*`y-coordinate`*`)`

5.16.3 user interface controls:

- change with JavaScript
    - `window.open(`*`name`*`, `*`url`*`, `*`windowFeatures`*`)` → this results in a change of user agent

5.16.4 focus:

- change focus with JavaScript
    - `otherElementElement.focus()`
    - `this.blur()`
    - `window.location.reload()`
- jump to different location within same document
    - `#somewhere`

5.16.5 complete change of content:

- load new resource

- a/@href
- area/@href
- img/@longdesc
- form/@action
- link/@href
- meta[@http-equiv='refresh'][contains(@content,'url=')]
- object/@data
- (q/@cite)
- (blockquote/@cite)
- (del/@cite)
- (ins/@cite)
- (frame/@longdesc)
- (iframe/@longdesc)

- load new resource with JavaScript
  - window.location.host
  - window.location.hostname
  - window.location.href
  - window.location.pathname
  - window.location.port
  - window.location.protocol
  - window.location.search
  - window.location.assign(*url*)
  - window.location.replace(*url*)
  - window.back()
  - window.forward()
  - window.home()
  - history.back()

- `history.forward()`
- `history.go()`

JavaScript objects, methods and properties according to Netscape's "Client-Side JavaScript Reference"[19], Mozilla's "Gecko DOM Reference"[20] and Microsoft's "HTML and DHTML Reference"[21].

[19]http://devedge-temp.mozilla.org/library/manuals/2000/javascript/1.3/reference/

[20]http://www.mozilla.org/docs/dom/domref/

[21]http://msdn.microsoft.com/library/default.asp?url=/workshop/author/dhtml/reference/dhtml_reference_entry.asp