HTML

Living Standard — Last Updated 23 May 2014

← 12.2 Parsing HTML documents – Table of contents – 12.2.5 Tree construction →

12.2.4 Tokenization

Implementations must act as if they used the following state machine to tokenise HTML. The state machine must start in the data state. Most states consume a single character, which may have various side-effects, and either switches the state machine to a new state to *reconsume* the same character, or switches it to a new state to consume the next character, or stays in the same state to consume the next character. Some states have more complicated behavior and can consume several characters before switching to another state. In some cases, the tokenizer state is also changed by the tree construction stage.

The exact behavior of certain states depends on the insertion mode and the stack of open elements. Certain states also use a ***temporary buffer*** to track progress.

The output of the tokenization step is a series of zero or more of the following tokens: DOCTYPE, start tag, end tag, comment, character, end-of-file. DOCTYPE tokens have a name, a public identifier, a system identifier, and a *force-quirks flag*. When a DOCTYPE token is created, its name, public identifier, and system identifier must be marked as missing (which is a distinct state from the empty string), and the *force-quirks flag* must be set to *off* (its other state is *on*). Start and end tag tokens have a tag name, a *self-closing flag*, and a list of attributes, each of which has a name and a value. When a start or end tag token is created, its *self-closing flag* must be unset (its other state is that it be set), and its attributes list must be empty. Comment and character tokens have data.

When a token is emitted, it must immediately be handled by the tree construction stage. The tree construction stage can affect the state of the tokenization stage, and can insert additional characters into the stream. (For example, the `script` element can result in scripts executing and using the dynamic markup insertion APIs to insert characters into the stream being tokenized.)

Note *Creating a token and emitting it are distinct actions. It is possible for a token to be created but implicitly abandoned (never emitted), e.g. if the file ends unexpectedly while processing the characters that are being parsed into a start tag token.*

When a start tag token is emitted with its *self-closing flag* set, if the flag is not **acknowledged** when it is processed by the tree construction stage, that is a parse error.

When an end tag token is emitted with attributes, that is a parse error.

When an end tag token is emitted with its *self-closing flag* set, that is a parse error.

An **appropriate end tag token** is an end tag token whose tag name matches the tag name of the last start tag to have been emitted from this tokenizer, if any. If no start tag has been emitted from this tokenizer, then no end tag token is appropriate.

Before each step of the tokenizer, the user agent must first check the parser pause flag. If it is true, then the tokenizer

must abort the processing of any nested invocations of the tokenizer, yielding control back to the caller.

The tokenizer state machine consists of the states defined in the following subsections.

12.2.4.1 Data state

Consume the next input character:

↳ **U+0026 AMPERSAND (&)**
Switch to the character reference in data state.

↳ **U+003C LESS-THAN SIGN (<)**
Switch to the tag open state.

↳ **U+0000 NULL**
Parse error. Emit the current input character as a character token.

↳ **EOF**
Emit an end-of-file token.

↳ **Anything else**
Emit the current input character as a character token.

12.2.4.2 Character reference in data state

Switch to the data state.

Attempt to consume a character reference, with no additional allowed character.

If nothing is returned, emit a U+0026 AMPERSAND character (&) token.

Otherwise, emit the character tokens that were returned.

12.2.4.3 RCDATA state

Consume the next input character:

↳ **U+0026 AMPERSAND (&)**
Switch to the character reference in RCDATA state.

↳ **U+003C LESS-THAN SIGN (<)**
Switch to the RCDATA less-than sign state.

↳ **U+0000 NULL**
Parse error. Emit a U+FFFD REPLACEMENT CHARACTER character token.

↳ **EOF**
Emit an end-of-file token.

↳ **Anything else**
Emit the current input character as a character token.

12.2.4.4 Character reference in RCDATA state

Switch to the RCDATA state.

Attempt to consume a character reference, with no additional allowed character.

If nothing is returned, emit a U+0026 AMPERSAND character (&) token.

Otherwise, emit the character tokens that were returned.

12.2.4.5 RAWTEXT state

Consume the next input character:

↪ **U+003C LESS-THAN SIGN (<)**
Switch to the RAWTEXT less-than sign state.

↪ **U+0000 NULL**
Parse error. Emit a U+FFFD REPLACEMENT CHARACTER character token.

↪ **EOF**
Emit an end-of-file token.

↪ **Anything else**
Emit the current input character as a character token.

12.2.4.6 Script data state

Consume the next input character:

↪ **U+003C LESS-THAN SIGN (<)**
Switch to the script data less-than sign state.

↪ **U+0000 NULL**
Parse error. Emit a U+FFFD REPLACEMENT CHARACTER character token.

↪ **EOF**
Emit an end-of-file token.

↪ **Anything else**
Emit the current input character as a character token.

12.2.4.7 PLAINTEXT state

Consume the next input character:

↪ **U+0000 NULL**
Parse error. Emit a U+FFFD REPLACEMENT CHARACTER character token.

↪ **EOF**
Emit an end-of-file token.

↪ **Anything else**
Emit the current input character as a character token.

12.2.4.8 Tag open state

Consume the next input character:

↳ **U+0021 EXCLAMATION MARK (!)**

Switch to the markup declaration open state.

↳ **U+002F SOLIDUS (/)**

Switch to the end tag open state.

↳ **Uppercase ASCII letter**

Create a new start tag token, set its tag name to the lowercase version of the current input character (add 0x0020 to the character's code point), then switch to the tag name state. (Don't emit the token yet; further details will be filled in before it is emitted.)

↳ **Lowercase ASCII letter**

Create a new start tag token, set its tag name to the current input character, then switch to the tag name state. (Don't emit the token yet; further details will be filled in before it is emitted.)

↳ **U+003F QUESTION MARK (?)**

Parse error. Switch to the bogus comment state.

↳ **Anything else**

Parse error. Switch to the data state. Emit a U+003C LESS-THAN SIGN character token. Reconsume the current input character.

12.2.4.9 End tag open state

Consume the next input character:

↳ **Uppercase ASCII letter**

Create a new end tag token, set its tag name to the lowercase version of the current input character (add 0x0020 to the character's code point), then switch to the tag name state. (Don't emit the token yet; further details will be filled in before it is emitted.)

↳ **Lowercase ASCII letter**

Create a new end tag token, set its tag name to the current input character, then switch to the tag name state. (Don't emit the token yet; further details will be filled in before it is emitted.)

↳ **U+003E GREATER-THAN SIGN (>)**

Parse error. Switch to the data state.

↳ **EOF**

Parse error. Switch to the data state. Emit a U+003C LESS-THAN SIGN character token and a U+002F SOLIDUS character token. Reconsume the EOF character.

↳ **Anything else**

Parse error. Switch to the bogus comment state.

12.2.4.10 Tag name state

Consume the next input character:

↳ **U+0009 CHARACTER TABULATION (tab)**
↳ **U+000A LINE FEED (LF)**
↳ **U+000C FORM FEED (FF)**
↳ **U+0020 SPACE**

Switch to the before attribute name state.

↳ **U+002F SOLIDUS (/)**

Switch to the self-closing start tag state.

↳ **U+003E GREATER-THAN SIGN (>)**

Switch to the data state. Emit the current tag token.

↳ **Uppercase ASCII letter**

Append the lowercase version of the current input character (add 0x0020 to the character's code point) to the current tag token's tag name.

↳ **U+0000 NULL**

Parse error. Append a U+FFFD REPLACEMENT CHARACTER character to the current tag token's tag name.

↳ **EOF**

Parse error. Switch to the data state. Reconsume the EOF character.

↳ **Anything else**

Append the current input character to the current tag token's tag name.

12.2.4.11 RCDATA less-than sign state

Consume the next input character:

↳ **U+002F SOLIDUS (/)**

Set the *temporary buffer* to the empty string. Switch to the RCDATA end tag open state.

↳ **Anything else**

Switch to the RCDATA state. Emit a U+003C LESS-THAN SIGN character token. Reconsume the current input character.

12.2.4.12 RCDATA end tag open state

Consume the next input character:

↳ **Uppercase ASCII letter**

Create a new end tag token, and set its tag name to the lowercase version of the current input character (add 0x0020 to the character's code point). Append the current input character to the *temporary buffer*. Finally, switch to the RCDATA end tag name state. (Don't emit the token yet; further details will be filled in before it is emitted.)

↳ **Lowercase ASCII letter**

Create a new end tag token, and set its tag name to the current input character. Append the current input character to the *temporary buffer*. Finally, switch to the RCDATA end tag name state. (Don't emit the token yet; further details will be filled in before it is emitted.)

↳ **Anything else**

Switch to the RCDATA state. Emit a U+003C LESS-THAN SIGN character token and a U+002F SOLIDUS character token. Reconsume the current input character.

12.2.4.13 RCDATA end tag name state

Consume the next input character:

↳ **U+0009 CHARACTER TABULATION (tab)**
↳ **U+000A LINE FEED (LF)**
↳ **U+000C FORM FEED (FF)**
↳ **U+0020 SPACE**

If the current end tag token is an appropriate end tag token, then switch to the before attribute name state. Otherwise, treat it as per the "anything else" entry below.

↪ **U+002F SOLIDUS (/)**

If the current end tag token is an appropriate end tag token, then switch to the self-closing start tag state. Otherwise, treat it as per the "anything else" entry below.

↪ **U+003E GREATER-THAN SIGN (>)**

If the current end tag token is an appropriate end tag token, then switch to the data state and emit the current tag token. Otherwise, treat it as per the "anything else" entry below.

↪ **Uppercase ASCII letter**

Append the lowercase version of the current input character (add 0x0020 to the character's code point) to the current tag token's tag name. Append the current input character to the *temporary buffer*.

↪ **Lowercase ASCII letter**

Append the current input character to the current tag token's tag name. Append the current input character to the *temporary buffer*.

↪ **Anything else**

Switch to the RCDATA state. Emit a U+003C LESS-THAN SIGN character token, a U+002F SOLIDUS character token, and a character token for each of the characters in the *temporary buffer* (in the order they were added to the buffer). Reconsume the current input character.

12.2.4.14 RAWTEXT less-than sign state

Consume the next input character:

↪ **U+002F SOLIDUS (/)**

Set the *temporary buffer* to the empty string. Switch to the RAWTEXT end tag open state.

↪ **Anything else**

Switch to the RAWTEXT state. Emit a U+003C LESS-THAN SIGN character token. Reconsume the current input character.

12.2.4.15 RAWTEXT end tag open state

Consume the next input character:

↪ **Uppercase ASCII letter**

Create a new end tag token, and set its tag name to the lowercase version of the current input character (add 0x0020 to the character's code point). Append the current input character to the *temporary buffer*. Finally, switch to the RAWTEXT end tag name state. (Don't emit the token yet; further details will be filled in before it is emitted.)

↪ **Lowercase ASCII letter**

Create a new end tag token, and set its tag name to the current input character. Append the current input character to the *temporary buffer*. Finally, switch to the RAWTEXT end tag name state. (Don't emit the token yet; further details will be filled in before it is emitted.)

↪ **Anything else**

Switch to the RAWTEXT state. Emit a U+003C LESS-THAN SIGN character token and a U+002F SOLIDUS character token. Reconsume the current input character.

12.2.4.16 RAWTEXT end tag name state

Consume the next input character:

↪ **U+0009 CHARACTER TABULATION (tab)**

↪ **U+000A LINE FEED (LF)**

↪ **U+000C FORM FEED (FF)**

↪ **U+0020 SPACE**

If the current end tag token is an appropriate end tag token, then switch to the before attribute name state. Otherwise, treat it as per the "anything else" entry below.

↪ **U+002F SOLIDUS (/)**

If the current end tag token is an appropriate end tag token, then switch to the self-closing start tag state. Otherwise, treat it as per the "anything else" entry below.

↪ **U+003E GREATER-THAN SIGN (>)**

If the current end tag token is an appropriate end tag token, then switch to the data state and emit the current tag token. Otherwise, treat it as per the "anything else" entry below.

↪ **Uppercase ASCII letter**

Append the lowercase version of the current input character (add 0x0020 to the character's code point) to the current tag token's tag name. Append the current input character to the *temporary buffer*.

↪ **Lowercase ASCII letter**

Append the current input character to the current tag token's tag name. Append the current input character to the *temporary buffer*.

↪ **Anything else**

Switch to the RAWTEXT state. Emit a U+003C LESS-THAN SIGN character token, a U+002F SOLIDUS character token, and a character token for each of the characters in the *temporary buffer* (in the order they were added to the buffer). Reconsume the current input character.

12.2.4.17 Script data less-than sign state

Consume the next input character:

↪ **U+002F SOLIDUS (/)**

Set the *temporary buffer* to the empty string. Switch to the script data end tag open state.

↪ **U+0021 EXCLAMATION MARK (!)**

Switch to the script data escape start state. Emit a U+003C LESS-THAN SIGN character token and a U+0021 EXCLAMATION MARK character token.

↪ **Anything else**

Switch to the script data state. Emit a U+003C LESS-THAN SIGN character token. Reconsume the current input character.

12.2.4.18 Script data end tag open state

Consume the next input character:

↪ **Uppercase ASCII letter**

Create a new end tag token, and set its tag name to the lowercase version of the current input character (add 0x0020 to the character's code point). Append the current input character to the *temporary buffer*. Finally, switch to the script data end tag name state. (Don't emit the token yet; further details will be filled in before it is emitted.)

↪ **Lowercase ASCII letter**

Create a new end tag token, and set its tag name to the current input character. Append the current input character to the *temporary buffer*. Finally, switch to the script data end tag name state. (Don't emit the token yet; further details will be filled in before it is emitted.)

↪ **Anything else**

Switch to the script data state. Emit a U+003C LESS-THAN SIGN character token and a U+002F SOLIDUS character token. Reconsume the current input character.

12.2.4.19 Script data end tag name state

Consume the next input character:

- **U+0009 CHARACTER TABULATION (tab)**
- **U+000A LINE FEED (LF)**
- **U+000C FORM FEED (FF)**
- **U+0020 SPACE**

  If the current end tag token is an appropriate end tag token, then switch to the before attribute name state. Otherwise, treat it as per the "anything else" entry below.

- **U+002F SOLIDUS (/)**

  If the current end tag token is an appropriate end tag token, then switch to the self-closing start tag state. Otherwise, treat it as per the "anything else" entry below.

- **U+003E GREATER-THAN SIGN (>)**

  If the current end tag token is an appropriate end tag token, then switch to the data state and emit the current tag token. Otherwise, treat it as per the "anything else" entry below.

- **Uppercase ASCII letter**

  Append the lowercase version of the current input character (add 0x0020 to the character's code point) to the current tag token's tag name. Append the current input character to the *temporary buffer*.

- **Lowercase ASCII letter**

  Append the current input character to the current tag token's tag name. Append the current input character to the *temporary buffer*.

- **Anything else**

  Switch to the script data state. Emit a U+003C LESS-THAN SIGN character token, a U+002F SOLIDUS character token, and a character token for each of the characters in the *temporary buffer* (in the order they were added to the buffer). Reconsume the current input character.

12.2.4.20 Script data escape start state

Consume the next input character:

- **U+002D HYPHEN-MINUS (-)**

  Switch to the script data escape start dash state. Emit a U+002D HYPHEN-MINUS character token.

- **Anything else**

  Switch to the script data state. Reconsume the current input character.

12.2.4.21 Script data escape start dash state

Consume the next input character:

- **U+002D HYPHEN-MINUS (-)**

  Switch to the script data escaped dash dash state. Emit a U+002D HYPHEN-MINUS character token.

- **Anything else**

  Switch to the script data state. Reconsume the current input character.

12.2.4.22 Script data escaped state

Consume the next input character:

↪ **U+002D HYPHEN-MINUS (-)**
Switch to the script data escaped dash state. Emit a U+002D HYPHEN-MINUS character token.

↪ **U+003C LESS-THAN SIGN (<)**
Switch to the script data escaped less-than sign state.

↪ **U+0000 NULL**
Parse error. Emit a U+FFFD REPLACEMENT CHARACTER character token.

↪ **EOF**
Switch to the data state. Parse error. Reconsume the EOF character.

↪ **Anything else**
Emit the current input character as a character token.

12.2.4.23 Script data escaped dash state

Consume the next input character:

↪ **U+002D HYPHEN-MINUS (-)**
Switch to the script data escaped dash dash state. Emit a U+002D HYPHEN-MINUS character token.

↪ **U+003C LESS-THAN SIGN (<)**
Switch to the script data escaped less-than sign state.

↪ **U+0000 NULL**
Parse error. Switch to the script data escaped state. Emit a U+FFFD REPLACEMENT CHARACTER character token.

↪ **EOF**
Parse error. Switch to the data state. Reconsume the EOF character.

↪ **Anything else**
Switch to the script data escaped state. Emit the current input character as a character token.

12.2.4.24 Script data escaped dash dash state

Consume the next input character:

↪ **U+002D HYPHEN-MINUS (-)**
Emit a U+002D HYPHEN-MINUS character token.

↪ **U+003C LESS-THAN SIGN (<)**
Switch to the script data escaped less-than sign state.

↪ **U+003E GREATER-THAN SIGN (>)**
Switch to the script data state. Emit a U+003E GREATER-THAN SIGN character token.

↪ **U+0000 NULL**
Parse error. Switch to the script data escaped state. Emit a U+FFFD REPLACEMENT CHARACTER character token.

↪ **EOF**
Parse error. Switch to the data state. Reconsume the EOF character.

↪ **Anything else**
Switch to the script data escaped state. Emit the current input character as a character token.

12.2.4.25 Script data escaped less-than sign state

Consume the next input character:

↪ **U+002F SOLIDUS (/)**

Set the *temporary buffer* to the empty string. Switch to the script data escaped end tag open state.

↪ **Uppercase ASCII letter**

Set the *temporary buffer* to the empty string. Append the lowercase version of the current input character (add 0x0020 to the character's code point) to the *temporary buffer*. Switch to the script data double escape start state. Emit a U+003C LESS-THAN SIGN character token and the current input character as a character token.

↪ **Lowercase ASCII letter**

Set the *temporary buffer* to the empty string. Append the current input character to the *temporary buffer*. Switch to the script data double escape start state. Emit a U+003C LESS-THAN SIGN character token and the current input character as a character token.

↪ **Anything else**

Switch to the script data escaped state. Emit a U+003C LESS-THAN SIGN character token. Reconsume the current input character.

12.2.4.26 Script data escaped end tag open state

Consume the next input character:

↪ **Uppercase ASCII letter**

Create a new end tag token, and set its tag name to the lowercase version of the current input character (add 0x0020 to the character's code point). Append the current input character to the *temporary buffer*. Finally, switch to the script data escaped end tag name state. (Don't emit the token yet; further details will be filled in before it is emitted.)

↪ **Lowercase ASCII letter**

Create a new end tag token, and set its tag name to the current input character. Append the current input character to the *temporary buffer*. Finally, switch to the script data escaped end tag name state. (Don't emit the token yet; further details will be filled in before it is emitted.)

↪ **Anything else**

Switch to the script data escaped state. Emit a U+003C LESS-THAN SIGN character token and a U+002F SOLIDUS character token. Reconsume the current input character.

12.2.4.27 Script data escaped end tag name state

Consume the next input character:

↪ **U+0009 CHARACTER TABULATION (tab)**

↪ **U+000A LINE FEED (LF)**

↪ **U+000C FORM FEED (FF)**

↪ **U+0020 SPACE**

If the current end tag token is an appropriate end tag token, then switch to the before attribute name state. Otherwise, treat it as per the "anything else" entry below.

↪ **U+002F SOLIDUS (/)**

If the current end tag token is an appropriate end tag token, then switch to the self-closing start tag state. Otherwise, treat it as per the "anything else" entry below.

↪ **U+003E GREATER-THAN SIGN (>)**

If the current end tag token is an appropriate end tag token, then switch to the data state and emit the current

tag token. Otherwise, treat it as per the "anything else" entry below.

↪ **Uppercase ASCII letter**

Append the lowercase version of the current input character (add 0x0020 to the character's code point) to the current tag token's tag name. Append the current input character to the *temporary buffer*.

↪ **Lowercase ASCII letter**

Append the current input character to the current tag token's tag name. Append the current input character to the *temporary buffer*.

↪ **Anything else**

Switch to the script data escaped state. Emit a U+003C LESS-THAN SIGN character token, a U+002F SOLIDUS character token, and a character token for each of the characters in the *temporary buffer* (in the order they were added to the buffer). Reconsume the current input character.

12.2.4.28 Script data double escape start state

Consume the next input character:

↪ **U+0009 CHARACTER TABULATION (tab)**
↪ **U+000A LINE FEED (LF)**
↪ **U+000C FORM FEED (FF)**
↪ **U+0020 SPACE**
↪ **U+002F SOLIDUS (/)**
↪ **U+003E GREATER-THAN SIGN (>)**

If the *temporary buffer* is the string "`script`", then switch to the script data double escaped state. Otherwise, switch to the script data escaped state. Emit the current input character as a character token.

↪ **Uppercase ASCII letter**

Append the lowercase version of the current input character (add 0x0020 to the character's code point) to the *temporary buffer*. Emit the current input character as a character token.

↪ **Lowercase ASCII letter**

Append the current input character to the *temporary buffer*. Emit the current input character as a character token.

↪ **Anything else**

Switch to the script data escaped state. Reconsume the current input character.

12.2.4.29 Script data double escaped state

Consume the next input character:

↪ **U+002D HYPHEN-MINUS (-)**

Switch to the script data double escaped dash state. Emit a U+002D HYPHEN-MINUS character token.

↪ **U+003C LESS-THAN SIGN (<)**

Switch to the script data double escaped less-than sign state. Emit a U+003C LESS-THAN SIGN character token.

↪ **U+0000 NULL**

Parse error. Emit a U+FFFD REPLACEMENT CHARACTER character token.

↪ **EOF**

Parse error. Switch to the data state. Reconsume the EOF character.

↪ **Anything else**

Emit the current input character as a character token.

12.2.4.30 Script data double escaped dash state

Consume the next input character:

↪ **U+002D HYPHEN-MINUS (-)**
Switch to the script data double escaped dash dash state. Emit a U+002D HYPHEN-MINUS character token.

↪ **U+003C LESS-THAN SIGN (<)**
Switch to the script data double escaped less-than sign state. Emit a U+003C LESS-THAN SIGN character token.

↪ **U+0000 NULL**
Parse error. Switch to the script data double escaped state. Emit a U+FFFD REPLACEMENT CHARACTER character token.

↪ **EOF**
Parse error. Switch to the data state. Reconsume the EOF character.

↪ **Anything else**
Switch to the script data double escaped state. Emit the current input character as a character token.

12.2.4.31 Script data double escaped dash dash state

Consume the next input character:

↪ **U+002D HYPHEN-MINUS (-)**
Emit a U+002D HYPHEN-MINUS character token.

↪ **U+003C LESS-THAN SIGN (<)**
Switch to the script data double escaped less-than sign state. Emit a U+003C LESS-THAN SIGN character token.

↪ **U+003E GREATER-THAN SIGN (>)**
Switch to the script data state. Emit a U+003E GREATER-THAN SIGN character token.

↪ **U+0000 NULL**
Parse error. Switch to the script data double escaped state. Emit a U+FFFD REPLACEMENT CHARACTER character token.

↪ **EOF**
Parse error. Switch to the data state. Reconsume the EOF character.

↪ **Anything else**
Switch to the script data double escaped state. Emit the current input character as a character token.

12.2.4.32 Script data double escaped less-than sign state

Consume the next input character:

↪ **U+002F SOLIDUS (/)**
Set the *temporary buffer* to the empty string. Switch to the script data double escape end state. Emit a U+002F SOLIDUS character token.

↪ **Anything else**
Switch to the script data double escaped state. Reconsume the current input character.

12.2.4.33 Script data double escape end state

Consume the next input character:

↪ **U+0009 CHARACTER TABULATION (tab)**
↪ **U+000A LINE FEED (LF)**
↪ **U+000C FORM FEED (FF)**
↪ **U+0020 SPACE**
↪ **U+002F SOLIDUS (/)**
↪ **U+003E GREATER-THAN SIGN (>)**

If the *temporary buffer* is the string "`script`", then switch to the script data escaped state. Otherwise, switch to the script data double escaped state. Emit the current input character as a character token.

↪ **Uppercase ASCII letter**

Append the lowercase version of the current input character (add 0x0020 to the character's code point) to the *temporary buffer*. Emit the current input character as a character token.

↪ **Lowercase ASCII letter**

Append the current input character to the *temporary buffer*. Emit the current input character as a character token.

↪ **Anything else**

Switch to the script data double escaped state. Reconsume the current input character.

12.2.4.34 Before attribute name state

Consume the next input character:

↪ **U+0009 CHARACTER TABULATION (tab)**
↪ **U+000A LINE FEED (LF)**
↪ **U+000C FORM FEED (FF)**
↪ **U+0020 SPACE**

Ignore the character.

↪ **U+002F SOLIDUS (/)**

Switch to the self-closing start tag state.

↪ **U+003E GREATER-THAN SIGN (>)**

Switch to the data state. Emit the current tag token.

↪ **Uppercase ASCII letter**

Start a new attribute in the current tag token. Set that attribute's name to the lowercase version of the current input character (add 0x0020 to the character's code point), and its value to the empty string. Switch to the attribute name state.

↪ **U+0000 NULL**

Parse error. Start a new attribute in the current tag token. Set that attribute's name to a U+FFFD REPLACEMENT CHARACTER character, and its value to the empty string. Switch to the attribute name state.

↪ **U+0022 QUOTATION MARK (")**
↪ **U+0027 APOSTROPHE (')**
↪ **U+003C LESS-THAN SIGN (<)**
↪ **U+003D EQUALS SIGN (=)**

Parse error. Treat it as per the "anything else" entry below.

↪ **EOF**

Parse error. Switch to the data state. Reconsume the EOF character.

↪ **Anything else**

Start a new attribute in the current tag token. Set that attribute's name to the current input character, and its value to the empty string. Switch to the attribute name state.

12.2.4.35 Attribute name state

Consume the next input character:

↪ **U+0009 CHARACTER TABULATION (tab)**
↪ **U+000A LINE FEED (LF)**
↪ **U+000C FORM FEED (FF)**
↪ **U+0020 SPACE**
Switch to the after attribute name state.

↪ **U+002F SOLIDUS (/)**
Switch to the self-closing start tag state.

↪ **U+003D EQUALS SIGN (=)**
Switch to the before attribute value state.

↪ **U+003E GREATER-THAN SIGN (>)**
Switch to the data state. Emit the current tag token.

↪ **Uppercase ASCII letter**
Append the lowercase version of the current input character (add 0x0020 to the character's code point) to the current attribute's name.

↪ **U+0000 NULL**
Parse error. Append a U+FFFD REPLACEMENT CHARACTER character to the current attribute's name.

↪ **U+0022 QUOTATION MARK (")**
↪ **U+0027 APOSTROPHE (')**
↪ **U+003C LESS-THAN SIGN (<)**
Parse error. Treat it as per the "anything else" entry below.

↪ **EOF**
Parse error. Switch to the data state. Reconsume the EOF character.

↪ **Anything else**
Append the current input character to the current attribute's name.

When the user agent leaves the attribute name state (and before emitting the tag token, if appropriate), the complete attribute's name must be compared to the other attributes on the same token; if there is already an attribute on the token with the exact same name, then this is a parse error and the new attribute must be removed from the token.

Note *If an attribute is so removed from a token, it, and the value that gets associated with it, if any, are never subsequently used by the parser, and are therefore effectively discarded. Removing the attribute in this way does not change its status as the "current attribute" for the purposes of the tokenizer, however.*

12.2.4.36 After attribute name state

Consume the next input character:

↪ **U+0009 CHARACTER TABULATION (tab)**
↪ **U+000A LINE FEED (LF)**
↪ **U+000C FORM FEED (FF)**
↪ **U+0020 SPACE**
Ignore the character.

↳ **U+002F SOLIDUS (/)**

Switch to the self-closing start tag state.

↳ **U+003D EQUALS SIGN (=)**

Switch to the before attribute value state.

↳ **U+003E GREATER-THAN SIGN (>)**

Switch to the data state. Emit the current tag token.

↳ **Uppercase ASCII letter**

Start a new attribute in the current tag token. Set that attribute's name to the lowercase version of the current input character (add 0x0020 to the character's code point), and its value to the empty string. Switch to the attribute name state.

↳ **U+0000 NULL**

Parse error. Start a new attribute in the current tag token. Set that attribute's name to a U+FFFD REPLACEMENT CHARACTER character, and its value to the empty string. Switch to the attribute name state.

↳ **U+0022 QUOTATION MARK (")**

↳ **U+0027 APOSTROPHE (')**

↳ **U+003C LESS-THAN SIGN (<)**

Parse error. Treat it as per the "anything else" entry below.

↳ **EOF**

Parse error. Switch to the data state. Reconsume the EOF character.

↳ **Anything else**

Start a new attribute in the current tag token. Set that attribute's name to the current input character, and its value to the empty string. Switch to the attribute name state.

12.2.4.37 Before attribute value state

Consume the next input character:

↳ **U+0009 CHARACTER TABULATION (tab)**

↳ **U+000A LINE FEED (LF)**

↳ **U+000C FORM FEED (FF)**

↳ **U+0020 SPACE**

Ignore the character.

↳ **U+0022 QUOTATION MARK (")**

Switch to the attribute value (double-quoted) state.

↳ **U+0026 AMPERSAND (&)**

Switch to the attribute value (unquoted) state. Reconsume the current input character.

↳ **U+0027 APOSTROPHE (')**

Switch to the attribute value (single-quoted) state.

↳ **U+0000 NULL**

Parse error. Append a U+FFFD REPLACEMENT CHARACTER character to the current attribute's value. Switch to the attribute value (unquoted) state.

↳ **U+003E GREATER-THAN SIGN (>)**

Parse error. Switch to the data state. Emit the current tag token.

↳ **U+003C LESS-THAN SIGN (<)**

↳ **U+003D EQUALS SIGN (=)**

↳ **U+0060 GRAVE ACCENT (`)**

Parse error. Treat it as per the "anything else" entry below.

↪ **EOF**

Parse error. Switch to the data state. Reconsume the EOF character.

↪ **Anything else**

Append the current input character to the current attribute's value. Switch to the attribute value (unquoted) state.

12.2.4.38 Attribute value (double-quoted) state

Consume the next input character:

↪ **U+0022 QUOTATION MARK (")**

Switch to the after attribute value (quoted) state.

↪ **U+0026 AMPERSAND (&)**

Switch to the character reference in attribute value state, with the additional allowed character being U+0022 QUOTATION MARK (").

↪ **U+0000 NULL**

Parse error. Append a U+FFFD REPLACEMENT CHARACTER character to the current attribute's value.

↪ **EOF**

Parse error. Switch to the data state. Reconsume the EOF character.

↪ **Anything else**

Append the current input character to the current attribute's value.

12.2.4.39 Attribute value (single-quoted) state

Consume the next input character:

↪ **U+0027 APOSTROPHE (')**

Switch to the after attribute value (quoted) state.

↪ **U+0026 AMPERSAND (&)**

Switch to the character reference in attribute value state, with the additional allowed character being U+0027 APOSTROPHE (').

↪ **U+0000 NULL**

Parse error. Append a U+FFFD REPLACEMENT CHARACTER character to the current attribute's value.

↪ **EOF**

Parse error. Switch to the data state. Reconsume the EOF character.

↪ **Anything else**

Append the current input character to the current attribute's value.

12.2.4.40 Attribute value (unquoted) state

Consume the next input character:

↪ **U+0009 CHARACTER TABULATION (tab)**

↪ **U+000A LINE FEED (LF)**

↪ **U+000C FORM FEED (FF)**

↪ **U+0020 SPACE**

Switch to the before attribute name state.

↪ **U+0026 AMPERSAND (&)**

Switch to the character reference in attribute value state, with the additional allowed character being U+003E GREATER-THAN SIGN (>).

↪ **U+003E GREATER-THAN SIGN (>)**

Switch to the data state. Emit the current tag token.

↪ **U+0000 NULL**

Parse error. Append a U+FFFD REPLACEMENT CHARACTER character to the current attribute's value.

↪ **U+0022 QUOTATION MARK (")**

↪ **U+0027 APOSTROPHE (')**

↪ **U+003C LESS-THAN SIGN (<)**

↪ **U+003D EQUALS SIGN (=)**

↪ **U+0060 GRAVE ACCENT (`)**

Parse error. Treat it as per the "anything else" entry below.

↪ **EOF**

Parse error. Switch to the data state. Reconsume the EOF character.

↪ **Anything else**

Append the current input character to the current attribute's value.

12.2.4.41 Character reference in attribute value state

Attempt to consume a character reference.

If nothing is returned, append a U+0026 AMPERSAND character (&) to the current attribute's value.

Otherwise, append the returned character tokens to the current attribute's value.

Finally, switch back to the attribute value state that switched into this state.

12.2.4.42 After attribute value (quoted) state

Consume the next input character:

↪ **U+0009 CHARACTER TABULATION (tab)**

↪ **U+000A LINE FEED (LF)**

↪ **U+000C FORM FEED (FF)**

↪ **U+0020 SPACE**

Switch to the before attribute name state.

↪ **U+002F SOLIDUS (/)**

Switch to the self-closing start tag state.

↪ **U+003E GREATER-THAN SIGN (>)**

Switch to the data state. Emit the current tag token.

↪ **EOF**

Parse error. Switch to the data state. Reconsume the EOF character.

↪ **Anything else**

Parse error. Switch to the before attribute name state. Reconsume the character.

12.2.4.43 Self-closing start tag state

Consume the next input character:

↪ **U+003E GREATER-THAN SIGN (>)**

Set the *self-closing flag* of the current tag token. Switch to the data state. Emit the current tag token.

↪ **EOF**

Parse error. Switch to the data state. Reconsume the EOF character.

↪ **Anything else**

Parse error. Switch to the before attribute name state. Reconsume the character.

12.2.4.44 Bogus comment state

Consume every character up to and including the first U+003E GREATER-THAN SIGN character (>) or the end of the file (EOF), whichever comes first. Emit a comment token whose data is the concatenation of all the characters starting from and including the character that caused the state machine to switch into the bogus comment state, up to and including the character immediately before the last consumed character (i.e. up to the character just before the U+003E or EOF character), but with any U+0000 NULL characters replaced by U+FFFD REPLACEMENT CHARACTER characters. (If the comment was started by the end of the file (EOF), the token is empty. Similarly, the token is empty if it was generated by the string "<!>".)

Switch to the data state.

If the end of the file was reached, reconsume the EOF character.

12.2.4.45 Markup declaration open state

If the next two characters are both U+002D HYPHEN-MINUS characters (-), consume those two characters, create a comment token whose data is the empty string, and switch to the comment start state.

Otherwise, if the next seven characters are an ASCII case-insensitive match for the word "DOCTYPE", then consume those characters and switch to the DOCTYPE state.

Otherwise, if there is an adjusted current node and it is not an element in the HTML namespace and the next seven characters are a case-sensitive match for the string "[CDATA[" (the five uppercase letters "CDATA" with a U+005B LEFT SQUARE BRACKET character before and after), then consume those characters and switch to the CDATA section state.

Otherwise, this is a parse error. Switch to the bogus comment state. The next character that is consumed, if any, is the first character that will be in the comment.

12.2.4.46 Comment start state

Consume the next input character:

↪ **U+002D HYPHEN-MINUS (-)**

Switch to the comment start dash state.

↪ **U+0000 NULL**

Parse error. Append a U+FFFD REPLACEMENT CHARACTER character to the comment token's data. Switch to the comment state.

↪ **U+003E GREATER-THAN SIGN (>)**

Parse error. Switch to the data state. Emit the comment token.

↪ **EOF**

Parse error. Switch to the data state. Emit the comment token. Reconsume the EOF character.

↪ **Anything else**

Append the current input character to the comment token's data. Switch to the comment state.

12.2.4.47 Comment start dash state

Consume the next input character:

↪ **U+002D HYPHEN-MINUS (-)**

Switch to the comment end state

↪ **U+0000 NULL**

Parse error. Append a U+002D HYPHEN-MINUS character (-) and a U+FFFD REPLACEMENT CHARACTER character to the comment token's data. Switch to the comment state.

↪ **U+003E GREATER-THAN SIGN (>)**

Parse error. Switch to the data state. Emit the comment token.

↪ **EOF**

Parse error. Switch to the data state. Emit the comment token. Reconsume the EOF character.

↪ **Anything else**

Append a U+002D HYPHEN-MINUS character (-) and the current input character to the comment token's data. Switch to the comment state.

12.2.4.48 Comment state

Consume the next input character:

↪ **U+002D HYPHEN-MINUS (-)**

Switch to the comment end dash state

↪ **U+0000 NULL**

Parse error. Append a U+FFFD REPLACEMENT CHARACTER character to the comment token's data.

↪ **EOF**

Parse error. Switch to the data state. Emit the comment token. Reconsume the EOF character.

↪ **Anything else**

Append the current input character to the comment token's data.

12.2.4.49 Comment end dash state

Consume the next input character:

↪ **U+002D HYPHEN-MINUS (-)**

Switch to the comment end state

↪ **U+0000 NULL**

Parse error. Append a U+002D HYPHEN-MINUS character (-) and a U+FFFD REPLACEMENT CHARACTER character to the comment token's data. Switch to the comment state.

↪ **EOF**

Parse error. Switch to the data state. Emit the comment token. Reconsume the EOF character.

↪ **Anything else**

Append a U+002D HYPHEN-MINUS character (-) and the current input character to the comment token's data. Switch to the comment state.

12.2.4.50 Comment end state

Consume the next input character:

↪ **U+003E GREATER-THAN SIGN (>)**
Switch to the data state. Emit the comment token.

↪ **U+0000 NULL**
Parse error. Append two U+002D HYPHEN-MINUS characters (-) and a U+FFFD REPLACEMENT CHARACTER character to the comment token's data. Switch to the comment state.

↪ **U+0021 EXCLAMATION MARK (!)**
Parse error. Switch to the comment end bang state.

↪ **U+002D HYPHEN-MINUS (-)**
Parse error. Append a U+002D HYPHEN-MINUS character (-) to the comment token's data.

↪ **EOF**
Parse error. Switch to the data state. Emit the comment token. Reconsume the EOF character.

↪ **Anything else**
Parse error. Append two U+002D HYPHEN-MINUS characters (-) and the current input character to the comment token's data. Switch to the comment state.

12.2.4.51 Comment end bang state

Consume the next input character:

↪ **U+002D HYPHEN-MINUS (-)**
Append two U+002D HYPHEN-MINUS characters (-) and a U+0021 EXCLAMATION MARK character (!) to the comment token's data. Switch to the comment end dash state.

↪ **U+003E GREATER-THAN SIGN (>)**
Switch to the data state. Emit the comment token.

↪ **U+0000 NULL**
Parse error. Append two U+002D HYPHEN-MINUS characters (-), a U+0021 EXCLAMATION MARK character (!), and a U+FFFD REPLACEMENT CHARACTER character to the comment token's data. Switch to the comment state.

↪ **EOF**
Parse error. Switch to the data state. Emit the comment token. Reconsume the EOF character.

↪ **Anything else**
Append two U+002D HYPHEN-MINUS characters (-), a U+0021 EXCLAMATION MARK character (!), and the current input character to the comment token's data. Switch to the comment state.

12.2.4.52 DOCTYPE state

Consume the next input character:

↪ **U+0009 CHARACTER TABULATION (tab)**
↪ **U+000A LINE FEED (LF)**
↪ **U+000C FORM FEED (FF)**
↪ **U+0020 SPACE**
Switch to the before DOCTYPE name state.

↪ **EOF**
Parse error. Switch to the data state. Create a new DOCTYPE token. Set its *force-quirks flag* to *on*. Emit the

token. Reconsume the EOF character.

↳ **Anything else**

Parse error. Switch to the before DOCTYPE name state. Reconsume the character.

12.2.4.53 Before DOCTYPE name state

Consume the next input character:

↳ **U+0009 CHARACTER TABULATION (tab)**

↳ **U+000A LINE FEED (LF)**

↳ **U+000C FORM FEED (FF)**

↳ **U+0020 SPACE**

Ignore the character.

↳ **Uppercase ASCII letter**

Create a new DOCTYPE token. Set the token's name to the lowercase version of the current input character (add 0x0020 to the character's code point). Switch to the DOCTYPE name state.

↳ **U+0000 NULL**

Parse error. Create a new DOCTYPE token. Set the token's name to a U+FFFD REPLACEMENT CHARACTER character. Switch to the DOCTYPE name state.

↳ **U+003E GREATER-THAN SIGN (>)**

Parse error. Create a new DOCTYPE token. Set its *force-quirks flag* to *on*. Switch to the data state. Emit the token.

↳ **EOF**

Parse error. Switch to the data state. Create a new DOCTYPE token. Set its *force-quirks flag* to *on*. Emit the token. Reconsume the EOF character.

↳ **Anything else**

Create a new DOCTYPE token. Set the token's name to the current input character. Switch to the DOCTYPE name state.

12.2.4.54 DOCTYPE name state

Consume the next input character:

↳ **U+0009 CHARACTER TABULATION (tab)**

↳ **U+000A LINE FEED (LF)**

↳ **U+000C FORM FEED (FF)**

↳ **U+0020 SPACE**

Switch to the after DOCTYPE name state.

↳ **U+003E GREATER-THAN SIGN (>)**

Switch to the data state. Emit the current DOCTYPE token.

↳ **Uppercase ASCII letter**

Append the lowercase version of the current input character (add 0x0020 to the character's code point) to the current DOCTYPE token's name.

↳ **U+0000 NULL**

Parse error. Append a U+FFFD REPLACEMENT CHARACTER character to the current DOCTYPE token's name.

↳ **EOF**

Parse error. Switch to the data state. Set the DOCTYPE token's *force-quirks flag* to *on*. Emit that DOCTYPE

token. Reconsume the EOF character.

↪ **Anything else**

Append the current input character to the current DOCTYPE token's name.

12.2.4.55 After DOCTYPE name state

Consume the next input character:

↪ **U+0009 CHARACTER TABULATION (tab)**
↪ **U+000A LINE FEED (LF)**
↪ **U+000C FORM FEED (FF)**
↪ **U+0020 SPACE**

Ignore the character.

↪ **U+003E GREATER-THAN SIGN (>)**

Switch to the data state. Emit the current DOCTYPE token.

↪ **EOF**

Parse error. Switch to the data state. Set the DOCTYPE token's *force-quirks flag* to *on*. Emit that DOCTYPE token. Reconsume the EOF character.

↪ **Anything else**

If the six characters starting from the current input character are an ASCII case-insensitive match for the word "PUBLIC", then consume those characters and switch to the after DOCTYPE public keyword state.

Otherwise, if the six characters starting from the current input character are an ASCII case-insensitive match for the word "SYSTEM", then consume those characters and switch to the after DOCTYPE system keyword state.

Otherwise, this is a parse error. Set the DOCTYPE token's *force-quirks flag* to *on*. Switch to the bogus DOCTYPE state.

12.2.4.56 After DOCTYPE public keyword state

Consume the next input character:

↪ **U+0009 CHARACTER TABULATION (tab)**
↪ **U+000A LINE FEED (LF)**
↪ **U+000C FORM FEED (FF)**
↪ **U+0020 SPACE**

Switch to the before DOCTYPE public identifier state.

↪ **U+0022 QUOTATION MARK (")**

Parse error. Set the DOCTYPE token's public identifier to the empty string (not missing), then switch to the DOCTYPE public identifier (double-quoted) state.

↪ **U+0027 APOSTROPHE (')**

Parse error. Set the DOCTYPE token's public identifier to the empty string (not missing), then switch to the DOCTYPE public identifier (single-quoted) state.

↪ **U+003E GREATER-THAN SIGN (>)**

Parse error. Set the DOCTYPE token's *force-quirks flag* to *on*. Switch to the data state. Emit that DOCTYPE token.

↪ **EOF**

Parse error. Switch to the data state. Set the DOCTYPE token's *force-quirks flag* to *on*. Emit that DOCTYPE token. Reconsume the EOF character.

↳ **Anything else**

Parse error. Set the DOCTYPE token's *force-quirks flag* to *on*. Switch to the bogus DOCTYPE state.

12.2.4.57 Before DOCTYPE public identifier state

Consume the next input character:

↳ **U+0009 CHARACTER TABULATION (tab)**

↳ **U+000A LINE FEED (LF)**

↳ **U+000C FORM FEED (FF)**

↳ **U+0020 SPACE**

Ignore the character.

↳ **U+0022 QUOTATION MARK (")**

Set the DOCTYPE token's public identifier to the empty string (not missing), then switch to the DOCTYPE public identifier (double-quoted) state.

↳ **U+0027 APOSTROPHE (')**

Set the DOCTYPE token's public identifier to the empty string (not missing), then switch to the DOCTYPE public identifier (single-quoted) state.

↳ **U+003E GREATER-THAN SIGN (>)**

Parse error. Set the DOCTYPE token's *force-quirks flag* to *on*. Switch to the data state. Emit that DOCTYPE token.

↳ **EOF**

Parse error. Switch to the data state. Set the DOCTYPE token's *force-quirks flag* to *on*. Emit that DOCTYPE token. Reconsume the EOF character.

↳ **Anything else**

Parse error. Set the DOCTYPE token's *force-quirks flag* to *on*. Switch to the bogus DOCTYPE state.

12.2.4.58 DOCTYPE public identifier (double-quoted) state

Consume the next input character:

↳ **U+0022 QUOTATION MARK (")**

Switch to the after DOCTYPE public identifier state.

↳ **U+0000 NULL**

Parse error. Append a U+FFFD REPLACEMENT CHARACTER character to the current DOCTYPE token's public identifier.

↳ **U+003E GREATER-THAN SIGN (>)**

Parse error. Set the DOCTYPE token's *force-quirks flag* to *on*. Switch to the data state. Emit that DOCTYPE token.

↳ **EOF**

Parse error. Switch to the data state. Set the DOCTYPE token's *force-quirks flag* to *on*. Emit that DOCTYPE token. Reconsume the EOF character.

↳ **Anything else**

Append the current input character to the current DOCTYPE token's public identifier.

12.2.4.59 DOCTYPE public identifier (single-quoted) state

Consume the next input character:

↪ **U+0027 APOSTROPHE (')**

Switch to the after DOCTYPE public identifier state.

↪ **U+0000 NULL**

Parse error. Append a U+FFFD REPLACEMENT CHARACTER character to the current DOCTYPE token's public identifier.

↪ **U+003E GREATER-THAN SIGN (>)**

Parse error. Set the DOCTYPE token's *force-quirks flag* to *on*. Switch to the data state. Emit that DOCTYPE token.

↪ **EOF**

Parse error. Switch to the data state. Set the DOCTYPE token's *force-quirks flag* to *on*. Emit that DOCTYPE token. Reconsume the EOF character.

↪ **Anything else**

Append the current input character to the current DOCTYPE token's public identifier.

12.2.4.60 After DOCTYPE public identifier state

Consume the next input character:

↪ **U+0009 CHARACTER TABULATION (tab)**
↪ **U+000A LINE FEED (LF)**
↪ **U+000C FORM FEED (FF)**
↪ **U+0020 SPACE**

Switch to the between DOCTYPE public and system identifiers state.

↪ **U+003E GREATER-THAN SIGN (>)**

Switch to the data state. Emit the current DOCTYPE token.

↪ **U+0022 QUOTATION MARK (")**

Parse error. Set the DOCTYPE token's system identifier to the empty string (not missing), then switch to the DOCTYPE system identifier (double-quoted) state.

↪ **U+0027 APOSTROPHE (')**

Parse error. Set the DOCTYPE token's system identifier to the empty string (not missing), then switch to the DOCTYPE system identifier (single-quoted) state.

↪ **EOF**

Parse error. Switch to the data state. Set the DOCTYPE token's *force-quirks flag* to *on*. Emit that DOCTYPE token. Reconsume the EOF character.

↪ **Anything else**

Parse error. Set the DOCTYPE token's *force-quirks flag* to *on*. Switch to the bogus DOCTYPE state.

12.2.4.61 Between DOCTYPE public and system identifiers state

Consume the next input character:

↪ **U+0009 CHARACTER TABULATION (tab)**
↪ **U+000A LINE FEED (LF)**
↪ **U+000C FORM FEED (FF)**
↪ **U+0020 SPACE**

Ignore the character.

↳ **U+003E GREATER-THAN SIGN (>)**

Switch to the data state. Emit the current DOCTYPE token.

↳ **U+0022 QUOTATION MARK (")**

Set the DOCTYPE token's system identifier to the empty string (not missing), then switch to the DOCTYPE system identifier (double-quoted) state.

↳ **U+0027 APOSTROPHE (')**

Set the DOCTYPE token's system identifier to the empty string (not missing), then switch to the DOCTYPE system identifier (single-quoted) state.

↳ **EOF**

Parse error. Switch to the data state. Set the DOCTYPE token's *force-quirks flag* to *on*. Emit that DOCTYPE token. Reconsume the EOF character.

↳ **Anything else**

Parse error. Set the DOCTYPE token's *force-quirks flag* to *on*. Switch to the bogus DOCTYPE state.

12.2.4.62 After DOCTYPE system keyword state

Consume the next input character:

↳ **U+0009 CHARACTER TABULATION (tab)**
↳ **U+000A LINE FEED (LF)**
↳ **U+000C FORM FEED (FF)**
↳ **U+0020 SPACE**

Switch to the before DOCTYPE system identifier state.

↳ **U+0022 QUOTATION MARK (")**

Parse error. Set the DOCTYPE token's system identifier to the empty string (not missing), then switch to the DOCTYPE system identifier (double-quoted) state.

↳ **U+0027 APOSTROPHE (')**

Parse error. Set the DOCTYPE token's system identifier to the empty string (not missing), then switch to the DOCTYPE system identifier (single-quoted) state.

↳ **U+003E GREATER-THAN SIGN (>)**

Parse error. Set the DOCTYPE token's *force-quirks flag* to *on*. Switch to the data state. Emit that DOCTYPE token.

↳ **EOF**

Parse error. Switch to the data state. Set the DOCTYPE token's *force-quirks flag* to *on*. Emit that DOCTYPE token. Reconsume the EOF character.

↳ **Anything else**

Parse error. Set the DOCTYPE token's *force-quirks flag* to *on*. Switch to the bogus DOCTYPE state.

12.2.4.63 Before DOCTYPE system identifier state

Consume the next input character:

↳ **U+0009 CHARACTER TABULATION (tab)**
↳ **U+000A LINE FEED (LF)**
↳ **U+000C FORM FEED (FF)**
↳ **U+0020 SPACE**

Ignore the character.

↳ **U+0022 QUOTATION MARK (")**

Set the DOCTYPE token's system identifier to the empty string (not missing), then switch to the DOCTYPE system identifier (double-quoted) state.

↳ **U+0027 APOSTROPHE (')**

Set the DOCTYPE token's system identifier to the empty string (not missing), then switch to the DOCTYPE system identifier (single-quoted) state.

↳ **U+003E GREATER-THAN SIGN (>)**

Parse error. Set the DOCTYPE token's *force-quirks flag* to *on*. Switch to the data state. Emit that DOCTYPE token.

↳ **EOF**

Parse error. Switch to the data state. Set the DOCTYPE token's *force-quirks flag* to *on*. Emit that DOCTYPE token. Reconsume the EOF character.

↳ **Anything else**

Parse error. Set the DOCTYPE token's *force-quirks flag* to *on*. Switch to the bogus DOCTYPE state.

12.2.4.64 DOCTYPE system identifier (double-quoted) state

Consume the next input character:

↳ **U+0022 QUOTATION MARK (")**

Switch to the after DOCTYPE system identifier state.

↳ **U+0000 NULL**

Parse error. Append a U+FFFD REPLACEMENT CHARACTER character to the current DOCTYPE token's system identifier.

↳ **U+003E GREATER-THAN SIGN (>)**

Parse error. Set the DOCTYPE token's *force-quirks flag* to *on*. Switch to the data state. Emit that DOCTYPE token.

↳ **EOF**

Parse error. Switch to the data state. Set the DOCTYPE token's *force-quirks flag* to *on*. Emit that DOCTYPE token. Reconsume the EOF character.

↳ **Anything else**

Append the current input character to the current DOCTYPE token's system identifier.

12.2.4.65 DOCTYPE system identifier (single-quoted) state

Consume the next input character:

↳ **U+0027 APOSTROPHE (')**

Switch to the after DOCTYPE system identifier state.

↳ **U+0000 NULL**

Parse error. Append a U+FFFD REPLACEMENT CHARACTER character to the current DOCTYPE token's system identifier.

↳ **U+003E GREATER-THAN SIGN (>)**

Parse error. Set the DOCTYPE token's *force-quirks flag* to *on*. Switch to the data state. Emit that DOCTYPE token.

↳ **EOF**

Parse error. Switch to the data state. Set the DOCTYPE token's *force-quirks flag* to *on*. Emit that DOCTYPE token. Reconsume the EOF character.

↳ **Anything else**

Append the current input character to the current DOCTYPE token's system identifier.

12.2.4.66 After DOCTYPE system identifier state

Consume the next input character:

↳ **U+0009 CHARACTER TABULATION (tab)**
↳ **U+000A LINE FEED (LF)**
↳ **U+000C FORM FEED (FF)**
↳ **U+0020 SPACE**
Ignore the character.

↳ **U+003E GREATER-THAN SIGN (>)**
Switch to the data state. Emit the current DOCTYPE token.

↳ **EOF**
Parse error. Switch to the data state. Set the DOCTYPE token's *force-quirks flag* to *on*. Emit that DOCTYPE token. Reconsume the EOF character.

↳ **Anything else**
Parse error. Switch to the bogus DOCTYPE state. (This does *not* set the DOCTYPE token's *force-quirks flag* to *on*.)

12.2.4.67 Bogus DOCTYPE state

Consume the next input character:

↳ **U+003E GREATER-THAN SIGN (>)**
Switch to the data state. Emit the DOCTYPE token.

↳ **EOF**
Switch to the data state. Emit the DOCTYPE token. Reconsume the EOF character.

↳ **Anything else**
Ignore the character.

12.2.4.68 CDATA section state

Switch to the data state.

Consume every character up to the next occurrence of the three character sequence U+005D RIGHT SQUARE BRACKET U+005D RIGHT SQUARE BRACKET U+003E GREATER-THAN SIGN (`]]>`), or the end of the file (EOF), whichever comes first. Emit a series of character tokens consisting of all the characters consumed except the matching three character sequence at the end (if one was found before the end of the file).

If the end of the file was reached, reconsume the EOF character.

12.2.4.69 Tokenizing character references

This section defines how to **consume a character reference**, optionally with an **additional allowed character**, which, if specified where the algorithm is invoked, adds a character to the list of characters that cause there to not be a character reference.

This definition is used when parsing character references in text and in attributes.

The behavior depends on the identity of the next character (the one immediately after the U+0026 AMPERSAND character), as follows:

- **U+0009 CHARACTER TABULATION (tab)**
- **U+000A LINE FEED (LF)**
- **U+000C FORM FEED (FF)**
- **U+0020 SPACE**
- **U+003C LESS-THAN SIGN**
- **U+0026 AMPERSAND**
- **EOF**
- **The additional allowed character, if there is one**

  Not a character reference. No characters are consumed, and nothing is returned. (This is not an error, either.)

- **U+0023 NUMBER SIGN (#)**

  Consume the U+0023 NUMBER SIGN.

  The behavior further depends on the character after the U+0023 NUMBER SIGN:

  - **U+0078 LATIN SMALL LETTER X**
  - **U+0058 LATIN CAPITAL LETTER X**

    Consume the X.

    Follow the steps below, but using ASCII hex digits.

    When it comes to interpreting the number, interpret it as a hexadecimal number.

  - **Anything else**

    Follow the steps below, but using ASCII digits.

    When it comes to interpreting the number, interpret it as a decimal number.

  Consume as many characters as match the range of characters given above (ASCII hex digits or ASCII digits).

  If no characters match the range, then don't consume any characters (and unconsume the U+0023 NUMBER SIGN character and, if appropriate, the X character). This is a parse error; nothing is returned.

  Otherwise, if the next character is a U+003B SEMICOLON, consume that too. If it isn't, there is a parse error.

  If one or more characters match the range, then take them all and interpret the string of characters as a number (either hexadecimal or decimal as appropriate).

  If that number is one of the numbers in the first column of the following table, then this is a parse error. Find the row with that number in the first column, and return a character token for the Unicode character given in the second column of that row.

Number	Unicode character	
0x00	U+FFFD	REPLACEMENT CHARACTER
0x80	U+20AC	EURO SIGN (€)
0x82	U+201A	SINGLE LOW-9 QUOTATION MARK (‚)
0x83	U+0192	LATIN SMALL LETTER F WITH HOOK (ƒ)
0x84	U+201E	DOUBLE LOW-9 QUOTATION MARK („)
0x85	U+2026	HORIZONTAL ELLIPSIS (…)
0x86	U+2020	DAGGER (†)
0x87	U+2021	DOUBLE DAGGER (‡)
0x88	U+02C6	MODIFIER LETTER CIRCUMFLEX ACCENT (ˆ)
0x89	U+2030	PER MILLE SIGN (‰)
0x8A	U+0160	LATIN CAPITAL LETTER S WITH CARON (Š)
0x8B	U+2039	SINGLE LEFT-POINTING ANGLE QUOTATION MARK (‹)

0x8C	U+0152	LATIN CAPITAL LIGATURE OE (Œ)
0x8E	U+017D	LATIN CAPITAL LETTER Z WITH CARON (Ž)
0x91	U+2018	LEFT SINGLE QUOTATION MARK (‘)
0x92	U+2019	RIGHT SINGLE QUOTATION MARK (’)
0x93	U+201C	LEFT DOUBLE QUOTATION MARK (“)
0x94	U+201D	RIGHT DOUBLE QUOTATION MARK (”)
0x95	U+2022	BULLET (•)
0x96	U+2013	EN DASH (–)
0x97	U+2014	EM DASH (—)
0x98	U+02DC	SMALL TILDE (˜)
0x99	U+2122	TRADE MARK SIGN (™)
0x9A	U+0161	LATIN SMALL LETTER S WITH CARON (š)
0x9B	U+203A	SINGLE RIGHT-POINTING ANGLE QUOTATION MARK (›)
0x9C	U+0153	LATIN SMALL LIGATURE OE (œ)
0x9E	U+017E	LATIN SMALL LETTER Z WITH CARON (ž)
0x9F	U+0178	LATIN CAPITAL LETTER Y WITH DIAERESIS (Ÿ)

Otherwise, if the number is in the range 0xD800 to 0xDFFF or is greater than 0x10FFFF, then this is a parse error. Return a U+FFFD REPLACEMENT CHARACTER character token.

Otherwise, return a character token for the Unicode character whose code point is that number. Additionally, if the number is in the range 0x0001 to 0x0008, 0x000D to 0x001F, 0x007F to 0x009F, 0xFDD0 to 0xFDEF, or is one of 0x000B, 0xFFFE, 0xFFFF, 0x1FFFE, 0x1FFFF, 0x2FFFE, 0x2FFFF, 0x3FFFE, 0x3FFFF, 0x4FFFE, 0x4FFFF, 0x5FFFE, 0x5FFFF, 0x6FFFE, 0x6FFFF, 0x7FFFE, 0x7FFFF, 0x8FFFE, 0x8FFFF, 0x9FFFE, 0x9FFFF, 0xAFFFE, 0xAFFFF, 0xBFFFE, 0xBFFFF, 0xCFFFE, 0xCFFFF, 0xDFFFE, 0xDFFFF, 0xEFFFE, 0xEFFFF, 0xFFFFE, 0xFFFFF, 0x10FFFE, or 0x10FFFF, then this is a parse error.

↪ **Anything else**

Consume the maximum number of characters possible, with the consumed characters matching one of the identifiers in the first column of the named character references table (in a case-sensitive manner).

If no match can be made, then no characters are consumed, and nothing is returned. In this case, if the characters after the U+0026 AMPERSAND character (&) consist of a sequence of one or more alphanumeric ASCII characters followed by a U+003B SEMICOLON character (;), then this is a parse error.

If the character reference is being consumed as part of an attribute, and the last character matched is not a U+003B SEMICOLON character (;), and the next character is either a U+003D EQUALS SIGN character (=) or an alphanumeric ASCII character, then, for historical reasons, all the characters that were matched after the U+0026 AMPERSAND character (&) must be unconsumed, and nothing is returned. However, if this next character is in fact a U+003D EQUALS SIGN character (=), then this is a parse error, because some legacy user agents will misinterpret the markup in those cases.

Otherwise, a character reference is parsed. If the last character matched is not a U+003B SEMICOLON character (;), there is a parse error.

Return one or two character tokens for the character(s) corresponding to the character reference name (as given by the second column of the named character references table).

Example If the markup contains (not in an attribute) the string `I'm &notit; I tell you`, the character reference is parsed as "not", as in, `I'm ¬it; I tell you` (and this is a parse error). But if the markup was `I'm &notin; I tell you`, the character reference would be parsed as "notin;", resulting in `I'm &notinva; I tell you` (and no parse error).