

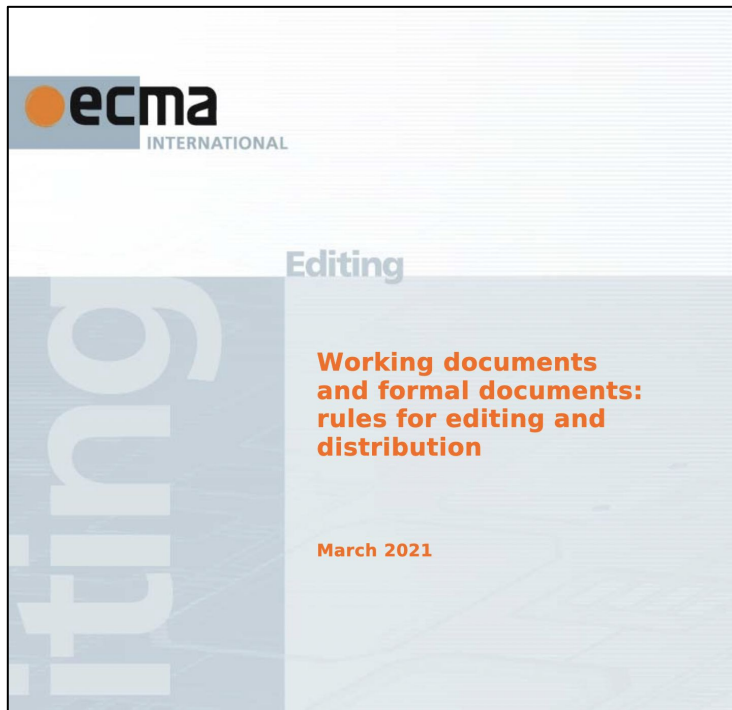
# Well-formatted PDFs for TC39 Standards in 2022 and Beyond

Allen Wirfs-Brock

Ecma Fellow

July 2022

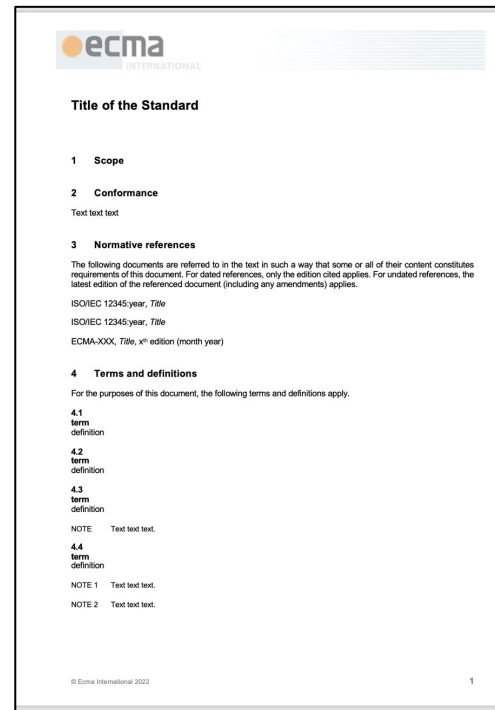
# Ecma has a “standard” for formatting standards as a “book”



TOOLS-011.docx

In [members.ecma-international.org](https://members.ecma-international.org)

TOOLS directory

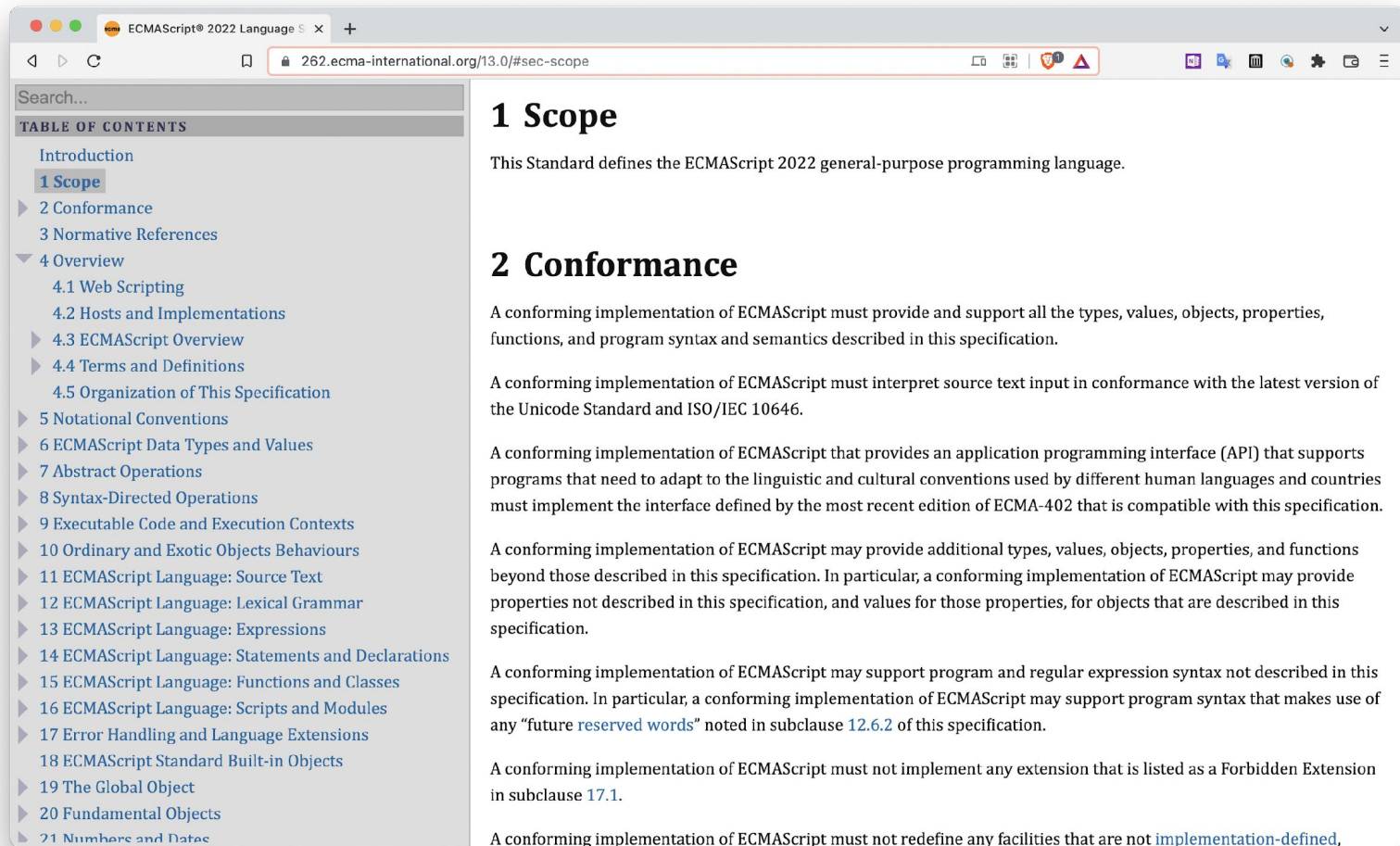


TOOLS-017.dotx

In [members.ecma-international.org](https://members.ecma-international.org)

TOOLS directory

# But, how do you turn this into book?



The image shows a web browser window displaying the ECMAScript 2022 Language Specification page. The browser's address bar shows the URL `262.ecma-international.org/13.0/#sec-scope`. The page has a sidebar on the left with a search bar and a table of contents. The main content area on the right displays the text for sections 1 and 2 of the specification.

**Search...**

**TABLE OF CONTENTS**

- Introduction
- 1 Scope**
- 2 Conformance
- 3 Normative References
- 4 Overview
  - 4.1 Web Scripting
  - 4.2 Hosts and Implementations
  - 4.3 ECMAScript Overview
  - 4.4 Terms and Definitions
  - 4.5 Organization of This Specification
- 5 Notational Conventions
- 6 ECMAScript Data Types and Values
- 7 Abstract Operations
- 8 Syntax-Directed Operations
- 9 Executable Code and Execution Contexts
- 10 Ordinary and Exotic Objects Behaviours
- 11 ECMAScript Language: Source Text
- 12 ECMAScript Language: Lexical Grammar
- 13 ECMAScript Language: Expressions
- 14 ECMAScript Language: Statements and Declarations
- 15 ECMAScript Language: Functions and Classes
- 16 ECMAScript Language: Scripts and Modules
- 17 Error Handling and Language Extensions
- 18 ECMAScript Standard Built-in Objects
- 19 The Global Object
- 20 Fundamental Objects
- 21 Numbers and Dates

## 1 Scope

This Standard defines the ECMAScript 2022 general-purpose programming language.

## 2 Conformance

A conforming implementation of ECMAScript must provide and support all the types, values, objects, properties, functions, and program syntax and semantics described in this specification.

A conforming implementation of ECMAScript must interpret source text input in conformance with the latest version of the Unicode Standard and ISO/IEC 10646.

A conforming implementation of ECMAScript that provides an application programming interface (API) that supports programs that need to adapt to the linguistic and cultural conventions used by different human languages and countries must implement the interface defined by the most recent edition of ECMA-402 that is compatible with this specification.

A conforming implementation of ECMAScript may provide additional types, values, objects, properties, and functions beyond those described in this specification. In particular, a conforming implementation of ECMAScript may provide properties not described in this specification, and values for those properties, for objects that are described in this specification.

A conforming implementation of ECMAScript may support program and regular expression syntax not described in this specification. In particular, a conforming implementation of ECMAScript may support program syntax that makes use of any “future [reserved words](#)” noted in subclause [12.6.2](#) of this specification.

A conforming implementation of ECMAScript must not implement any extension that is listed as a Forbidden Extension in subclause [17.1](#).

A conforming implementation of ECMAScript must not redefine any facilities that are not [implementation-defined](#),

# TC39 PDFs Don't Meet Ecma's Expectations

## Table of Contents

Introduction	
1 Scope	
2 Conformance	
2.1 Example Clause Heading	
3 Normative References	
4 Overview	
4.1 Web Scripting	
4.2 Hosts and Implementations	
4.3 ECMAScript Overview	
4.3.1 Objects	
4.3.2 The Strict Variant of ECMAScript	
4.4 Terms and Definitions	
4.4.1 implementation-approximated	
4.4.2 implementation-defined	
4.4.3 host-defined	
4.4.4 type	
4.4.5 primitive value	
4.4.6 object	
4.4.7 constructor	
4.4.8 prototype	
4.4.9 ordinary object	
4.4.10 exotic object	
4.4.11 standard object	
4.4.12 built-in object	
4.4.13 undefined value	
4.4.14 Undefined type	
4.4.15 null value	
4.4.16 Null type	
4.4.17 Boolean value	
4.4.18 Boolean type	
4.4.19 Boolean object	
4.4.20 String value	
4.4.21 String type	
4.4.22 String object	
4.4.23 Number value	
4.4.24 Number type	
4.4.25 Number object	
4.4.26 Infinity	
4.4.27 NaN	
4.4.28 BigInt value	
4.4.29 BigInt type	
4.4.30 BigInt object	
4.4.31 Symbol value	



Contents	Page
1 Scope	1
2 Conformance	1
3 Normative References	2
4 Overview	2
4.1 Web Scripting	3
4.2 Hosts and Implementations	3
4.3 ECMAScript Overview	3
4.4 Terms and Definitions	6
4.5 Organization of This Specification	11
5 Notational Conventions	12
5.1 Syntactic and Lexical Grammar	12
5.2 Algorithm Conventions	19
6 ECMAScript Data Types and Values	25
6.1 ECMAScript Language Types	25
6.2 ECMAScript Specification Types	51
7 Abstract Operations	64
7.1 Type Conversion	64
7.2 Testing and Comparison Operations	75
7.3 Operations on Objects	81
7.4 Operations on Iterator Objects	91
8 Syntax-Directed Operation	95
8.1 Scope Analysis	95
8.2 Labels	115
8.3 Function Name Inference	124
8.4 Contains	129
8.5 Miscellaneous	132
9 Executable Code and Execution Contexts	141
9.1 Environment Records	141
9.2 PrivateEnvironment Records	160
9.3 Realms	161
9.4 Execution Contexts	163
9.5 Jobs and Host Operations to Enqueue Jobs	166
9.6 InitializeHostDefinedRealm()	168
9.7 Agents	169
10 Ordinary and Exotic Objects Behaviours	175
10.1 Ordinary Object Internal Methods and Internal Slots	175
10.2 ECMAScript Function Objects	183
10.3 Built-in Function Objects	194
10.4 Built-in Exotic Object Internal Methods and Slots	196
10.5 Proxy Object Internal Methods and Internal Slots	214
11 ECMAScript Language: Source Text	224
11.1 ECMAScript Language: Source TextSource Text	224
11.2 Types of Source Code	227
12 ECMAScript Language: Lexical Grammar	229
12.1 Unicode Format-Control Characters	230
12.2 White Space	230
12.3 Line Terminators	231
12.4 Comments	232
12.5 Tokens	233
12.6 Names and Keywords	233
12.7 Punctuators	237
12.8 Literals	237
12.9 Automatic Semicolon Insertion	249

# CSS Paged Media to the rescue?

[W3C Specifications:](#)

[CSS Paged Media Module Level 3](#)

[CSS Generated Content for Paged Media  
Module](#)

[Requirements for Latin Text Layout and  
Pagination](#)

Etc.

But browsers don't support most CSS Paged Media features

# There are non-browser renderers that do

<https://print-css.rocks/lessons>

Lesson	PDFreactor	Prince	Antenna-house	Weasyprint	PagedJS	Typeset.sh	Vivliostyle
Basic	✓	✓	✓	✓	✓	✓	✓
Fonts	✓	✓	✓	✓	✓	✓	✓
Images	✓	✓	✓	✓	✓	✓	✓
Footnotes	✓	✓	✓	✓	!	✓	✓
Hyphenation	✓	✓	✓	✓	✗	✗	✗
Named pages	✓	✓	✓	✓	!	✓	!
Page areas	✓	✓	✓	✓	✓	✓	✓
Page numbers	✓	✓	✓	✓	✓	✓	✓
Pagination	✓	✓	✓	✓	✓	✓	✓
Positioning	✓	✓	✓	✓	✓	✓	✓
Right to left	✓	✓	✓	✓	✓	!	✓
Running elements	✓	✓	✓	✓	✓	✗	✗
Tables	✓	✓	✓	✓	!	!	✓
Chapter numbering	✓	✓	✓	✓	✓	✓	✓
Cross references	✓	✓	✓	✓	!	✓	✓
Grid	✓	✗	✗	✗	!	✓	✗
Flex	✓	✓	✗	✓	✓	✓	✓
Grid	✓	✗	✗	✗	!	✓	✗
Flex	✓	✓	✗	✓	✓	✓	✓

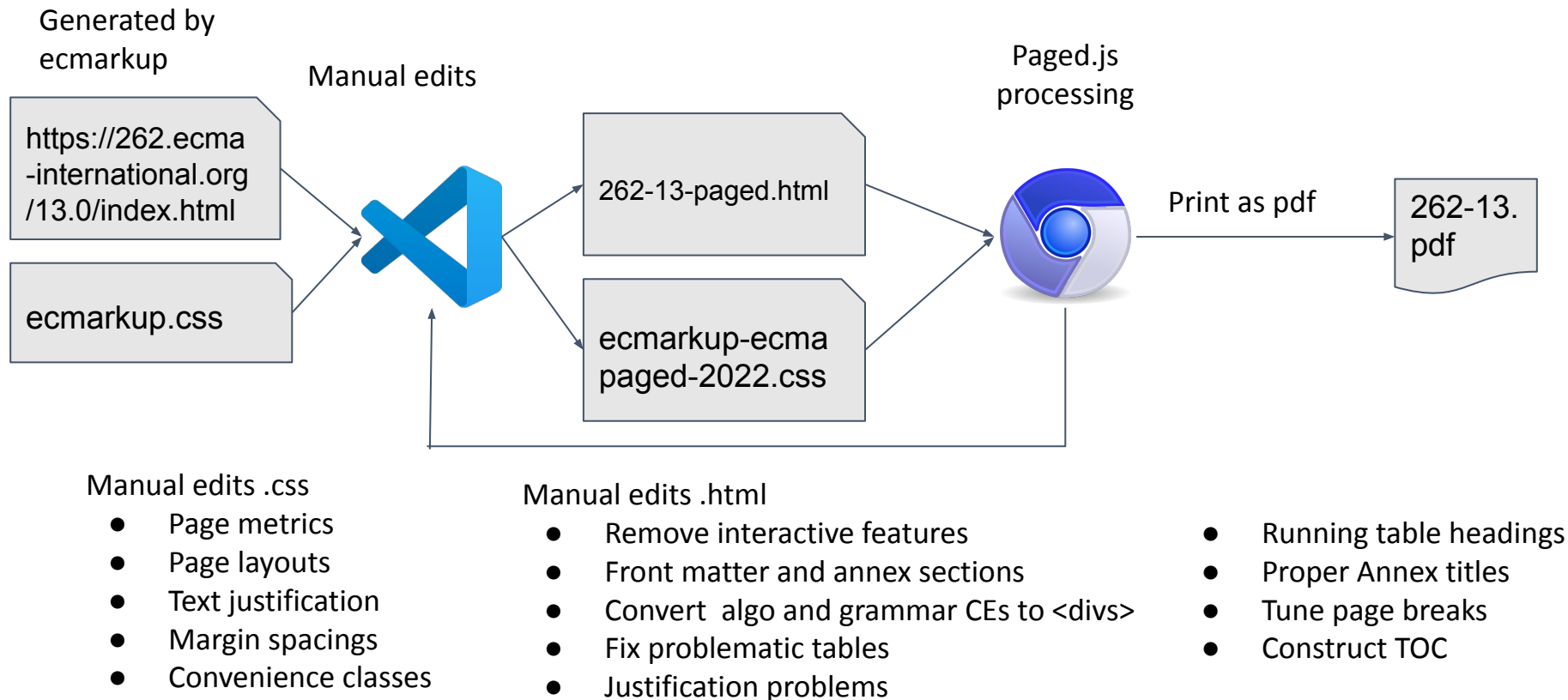
# About Paged.js?

Paged.js is a free and open source JavaScript library that paginates content in the browser to create PDF output from any HTML content. This means you can design works for print (eg. books) using HTML and CSS!

Paged.js follows the Paged Media standards published by the W3C (ie the Paged Media Module, and the Generated Content for Paged Media Module). In effect Paged.js acts as a polyfill for the CSS modules to print content using features that are not yet natively supported by browsers.

<https://pagedjs.org/>

# Workflow to create 2022 TC39 PDFs





# What about 2023 and beyond

## Options

1. Continue with Paged.js or switch to a commercial product
2. Repeat 2022 process (AWB is documenting this process)
3. Start with 2022 artifacts and apply 2023 changes
4. Automate as much of the PDF generation process as possible

## Considerations

- Good pagination requires esthetic decisions
- TC39 probably need a “PDF editor” to be responsible for this process

# A More Automated Workflow?

