International Journal of Education and Vocational Studies

Vol. 1, No. 2, Month Year, pp. 110-117
ISSN: 3124-9132 (Media Online)
DOI: https://doi.org/10.62712/ijevs.v1i2.12

Scaffolding Novice Web Authoring: A Pedagogical Design Analysis of an Indonesian HTML and CSS Textbook

Dwi Remawati[1*], Yaya Sudarya Triana[2], Ali Muhammad[3], Putra Edi Mujahid[4]

[1]Fakultas Sekolah Vokasi, Teknologi Informasi, Universitas Tiga Serangkai, Surakarta, Indonesia
[2]Fakultas Ilmu Komputer, Magister Sains Data, Universitas Mercu Buana, Jakarta, Indonesia
[3]Fakultas Ilmu Komputer, Teknik Informatika, Universitas Sains Indonesia, Bekasi, Indonesia
[4]Fakultas Ekonomi dan Bisnis Islam, Ekonomi Syariah, UIN Sulthan Thaha Saifuddin Jambi, Jambi, Indonesia
Author's Email: [1*]dwirema@tsu.ac.id, [2]yaya.sudarya@mercubuana.ac.id,
[3]ali.muhammad@lecturer.sains.ac.id, [4]Putraedim@uinjambi.ac.id

Article Info

Article history:

Received: September 5, 2026
Accepted: October 20, 2026
Published: November 1, 2026

Keywords:

HTML and CSS;
Instructional design;
Novice programmers;
Scaffolding;
Vocational education.

ABSTRACT

Web authoring with HTML and CSS is a common entry point to computing education, yet beginner instructional materials are rarely examined for the learning principles they embody. This study analysed the pedagogical design of a 16-chapter Indonesian-language textbook for novice web authors. A design-analysis approach combining qualitative document analysis with frequency counts of learning components was applied to every chapter, using a coding framework built from cognitive load theory, scaffolding, constructive alignment, mastery learning, learning from errors, and project-based learning. The textbook contained 64 stated learning objectives, 45 code listings, 35 rendered-output figures, 23 instructional callouts, 15 practice tasks, and a recurring seven-part chapter structure. Guidance faded from modifying a supplied file to building a locally themed website independently, and a dedicated chapter treated eleven categories of common HTML and CSS errors. All nine requirements of the final project were traceable to earlier chapters, indicating strong constructive alignment. Formative knowledge checks, however, appeared in only three of the fifteen content chapters. The analysis produced a design framework and a revision and validation agenda for beginner web-programming materials in vocational and teacher education.

This work is licensed under a CC BY-SA License.

Corresponding Author:

Dwi Remawati,
Fakultas Sekolah Vokasi, Teknologi Informasi, Universitas Tiga Serangkai
Surakarta, Indonesia
Email: dwirema@tsu.ac.id

1. INTRODUCTION

Web pages are the interface through which most people now study, shop, work, and obtain public services, and the ability to author a simple page has become part of what is often described as digital competence. HyperText Markup Language (HTML) and Cascading Style Sheets (CSS) are usually the first notations a learner meets when moving from using the web to building it. Because HTML and CSS give immediate visual feedback and require no compiler or server, they are widely taught in secondary schools, vocational high schools, and introductory university courses, and they frequently serve as a learner's first contact with formal syntax, debugging, and the separation of structure from presentation (Kharchenko & Semerikov, 2026; Park et al., 2015).

Despite this apparent accessibility, novices struggle. Studies of introductory web development courses show that HTML and CSS syntax errors are frequent in the first weeks of instruction and that the way learners resolve them is shaped by the tools and feedback available, including validation (Park et al., 2015).

Analyses of learners working with CSS layout found that performance dropped when several newly learned properties had to be combined, and that even learners who had mastered syntax continued to make semantic errors when applying unfamiliar properties (Kim et al., 2019). These difficulties echo long-standing findings in programming education, where novices hold fragile knowledge and lack the strategies needed to apply it (Robins et al., 2003; Medeiros et al., 2019). A recent systematic synthesis of research on teaching web design with HTML and CSS in secondary schools located only eight rigorous studies published between 2007 and 2023, identified syntax comprehension, student engagement, and teacher preparation as persistent challenges, and noted that the available evidence was concentrated in Indonesia and the United States (Kharchenko & Semerikov, 2026).

Instructional materials are a central lever in this situation. In many Indonesian vocational and teacher-education settings, a printed or digital textbook is the main structured resource that learners use outside class, and the quality of its design largely determines whether self-study succeeds. Development studies in Indonesian vocational education typically report expert-validity and practicality percentages for new learning media (e.g., Novaliendry et al., 2021), but they seldom make explicit which learning-science principles the product embodies, how those principles are operationalised chapter by chapter, or where the design remains incomplete. Without such an account, other authors cannot reuse the design logic, and later empirical evaluations cannot attribute their results to specific design features.

This article addresses that gap by analysing the pedagogical design of *HTML dan CSS Dasar: Panduan Praktis Membangun Halaman Web untuk Pemula*, a 16-chapter Indonesian-language textbook written for complete beginners and organised around a learning-by-doing approach in which every concept is accompanied by code that can be typed and run in a browser (Supiyandi et al., 2026, p. i). Three research questions guided the study: (RQ1) Which learning-science principles are embodied in the textbook's design? (RQ2) How are learning components distributed and sequenced across chapters? (RQ3) Which design gaps remain to be addressed before the textbook is evaluated empirically? The study contributes (a) a transparent, reusable framework that links six learning-science principles to concrete textbook features for beginner web programming, (b) a chapter-level quantitative profile of a complete beginner textbook, and (c) a revision and validation agenda that can guide authors and teachers of introductory web programming in vocational and teacher education.

2. LITERATURE REVIEW

2.1. Novice Difficulties in Learning HTML and CSS

Research on how beginners learn web development is thinner than research on general-purpose programming languages, but it points to consistent difficulties. Park et al. (2015) analysed the HTML and CSS code written by students in the opening weeks of a web development course and showed that syntax errors were common and that validation played an important role in whether and how they were corrected. Kim et al. (2019) traced learning curves in CSS flexbox and found that learners handled a single new property better than combinations of new and practised properties, which suggests that layout concepts should be introduced incrementally and practised in isolation before being combined. More broadly, reviews of introductory programming report that beginners find it hard to translate conceptual knowledge into working code, to read error feedback, and to sustain motivation (Medeiros et al., 2019; Robins et al., 2003). Error messages themselves are often unhelpful to novices (Becker et al., 2019); this matters particularly for HTML, because browsers silently tolerate malformed markup and display a result rather than an error, leaving the learner to infer the cause from an unexpected appearance.

2.2. Cognitive Load, Worked Examples, and Guidance

Cognitive load theory holds that working memory is severely limited when learners process new information, so instruction should minimise load that does not contribute to learning (Sweller et al., 2019). One of the most robust applications is the worked-example effect: novices learn more from studying fully worked solutions than from solving equivalent problems unaided, provided that examples are well structured and that related representations are integrated (Atkinson et al., 2000). Kirschner et al. (2006) argued that minimally guided discovery, inquiry, and problem-based approaches are ineffective for novices precisely because they overload working memory. For a beginner web textbook, these principles imply pairing every code listing with its rendered output, keeping a predictable chapter structure, and postponing open-ended tasks until the relevant schemas have been built.

2.3. Scaffolding, Fading, and Constructive Alignment

Scaffolding was first described as the support that enables a learner to complete a task that would be beyond unassisted effort (Wood et al., 1976). Reiser (2004) distinguished two mechanisms by which tools and materials scaffold learning: *structuring* the task to reduce complexity, and *problematising* content so that learners attend to important issues. Effective scaffolding is gradually withdrawn as competence grows. Constructive alignment requires that intended learning outcomes, learning activities, and assessment tasks be designed as a coherent system so that what is assessed is what learners have been helped to practise (Biggs,

1996). In a self-study textbook, alignment can be inspected directly by tracing each assessed requirement back to the chapter that teaches and exercises it.

2.4. Mastery Learning, Learning from Errors, and Project-Based Assessment

Mastery learning organises instruction into units, each followed by a formative assessment, corrective activities for learners who have not reached a criterion, and enrichment for those who have (Guskey, 2007). Learning from errors is a related idea: errors that are followed by corrective feedback can enhance rather than impair learning (Metcalfe, 2017), which supports treating debugging as content to be taught rather than as a nuisance. Active learning in science, technology, engineering, and mathematics has been shown to improve performance and reduce failure rates relative to traditional lecturing (Freeman et al., 2014). Project-based learning extends active learning to extended, authentic tasks and is effective when learners receive structured support (Kokotsaki et al., 2016). Analytic scoring rubrics make performance expectations explicit and can improve the reliability of scoring and support learning, especially when criteria are topic-specific (Jonsson & Svingby, 2007).

2.5. Conceptual Framework

Synthesising these strands, this study framed beginner web-programming materials as a chain linking six learning-science principles to observable textbook features and, through them, to intended learning outcomes (**Figure 1**). The framework served both as the coding scheme for the analysis and as the lens for interpreting its results. Because the study analysed a design rather than measured learning, no hypotheses were tested.

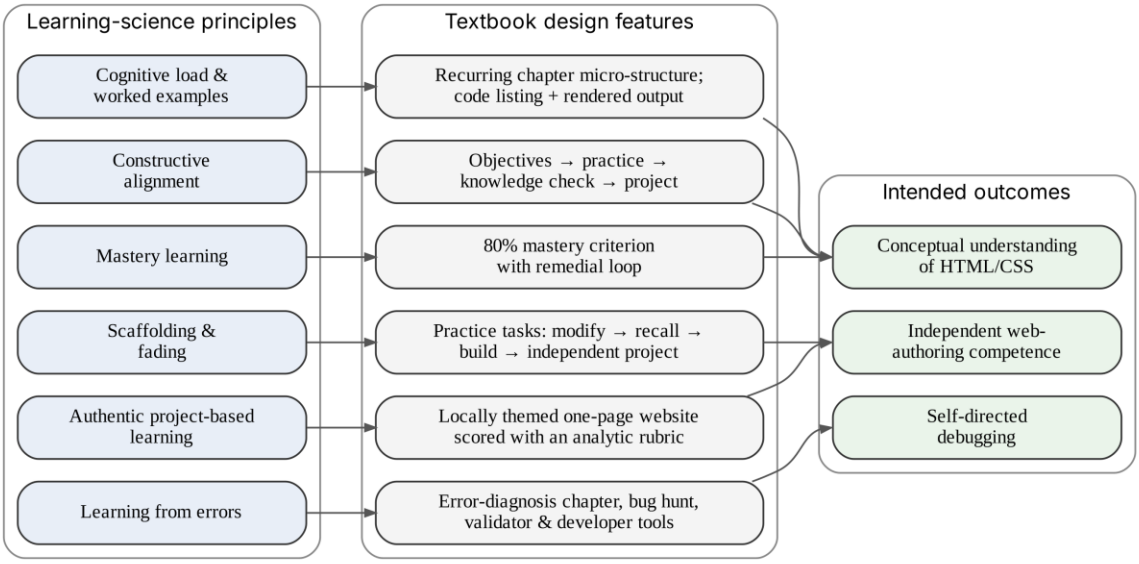


Figure 1. Conceptual framework linking learning-science principles, textbook design features, and intended outcomes

3. RESEARCH METHOD

The study used a design-analysis approach situated within design and development research, in which the object of study is an instructional product and the aim is to make its design rationale explicit and evaluable. Data were generated through qualitative document analysis, a systematic procedure for reviewing and interpreting printed and electronic documents (Bowen, 2009), complemented by frequency counts of learning components.

The analysed document was the second revision of the textbook *HTML dan CSS Dasar* (Supiyandi et al., 2026; ISBN 978-634-7439-55-0), published by CV. Raskha Media Group, Medan, with v + 109 pages in 16 chapters, a glossary (pp. 99–101), a reference list (pp. 102–103), and two appendices summarising HTML tags and CSS properties (pp. 104–106). The book is written in Indonesian for readers with no prior coding experience and is intended for self-study as well as classroom use. It assumes only a text editor and a web browser and requires neither a server nor an internet connection (Supiyandi et al., 2026, pp. 2–5).

The analysis proceeded in five steps. First, a coding framework was derived from the literature reviewed in Section 2 (**Table 1**). Second, the manuscript was converted to plain text and each chapter was segmented into its component types: stated learning objectives, code listings, figures, tables, instructional callouts, practice tasks, knowledge checks, and summaries. Third, component frequencies were counted per

chapter using pattern matching on the numbered captions and section headings, and every count was checked manually against the source. Fourth, each component type and each notable chapter feature was mapped to one or more principles of the framework. Fifth, three targeted analyses were conducted: the sequencing of topics and practice tasks (scaffolding and fading), the error taxonomy presented in the troubleshooting chapter (learning from errors), and the traceability of the final project requirements to earlier chapters (constructive alignment). Because the authors also wrote the textbook, the frequency counts were derived mechanically from the manuscript file so that they could be reproduced, an audit trail of counts and codes was retained, and gaps were reported as fully as strengths.

Table 1. Coding framework used in the design analysis

Principle	Operational indicator in the textbook	Key sources
Cognitive load and worked examples	Predictable chapter structure; each code listing paired with rendered output; concise summaries	Sweller et al. (2019); Atkinson et al. (2000)
Scaffolding and fading	Practice tasks progressing from modifying given code to independent creation	Wood et al. (1976); Reiser (2004)
Constructive alignment	Objectives, practice, and assessment refer to the same skills; project requirements traceable to chapters	Biggs (1996)
Mastery learning	Formative checks with an explicit criterion and a remedial loop	Guskey (2007)
Learning from errors	Explicit treatment of common errors, diagnostic tools, and error-finding exercises	Metcalfe (2017); Park et al. (2015)
Authentic project-based learning	Integrative project with authentic themes scored by an analytic rubric	Kokotsaki et al. (2016); Jonsson & Svingby (2007)

4. RESULTS AND DISCUSSION

4.1. Macro-Structure and Sequencing of Content

The 16 chapters formed four phases. Phase I (Chapters 1–4) established foundations: how the web works, the skeleton of an HTML document, headings and text formatting, and attributes. Phase II (Chapters 5–9) covered content structures: links and images, lists, tables, forms, and block, inline, and semantic elements. Phase III (Chapters 10–13) moved to presentation with CSS: rule syntax, selectors and the cascade, colour and typography, the box model, and flexbox with media queries for responsive design. Phase IV (Chapters 14–16) consolidated learning through a guided project, a troubleshooting chapter, and an independent project with a final knowledge check (Supiyandi et al., 2026, pp. 1–98). **Table 2** shows the distribution of learning components across chapters.

Table 2. Distribution of learning components across the 16 chapters

Ch.	Topic	Obj.	Code	Fig.	Tab.	Callout	Practice	Check
1	HTML and how the web works	4	1	3	2	4	1	–
2	Basic document structure	4	4	4	1	2	1	–
3	Headings, paragraphs, text formatting	4	3	3	1	2	1	–
4	Attributes	4	2	2	1	1	1	–
5	Links and images	4	4	2	3	2	1	–
6	Lists	4	3	3	2	1	1	–
7	Tables	4	2	2	1	1	1	–
8	Forms	4	1	1	1	2	1	✓
9	Block, inline, semantic elements	4	3	4	2	1	1	✓
10	Introduction to CSS	5	6	3	2	1	1	✓
11	Colour, text, and fonts	4	2	1	2	2	1	–
12	Box model	4	1	2	1	1	1	–
13	Flexbox and responsive design	4	2	3	1	1	1	–
14	Project: personal profile page	4	3	2	0	1	1	–
15	Common errors and fixes	4	8	0	0	1	1	–
16	Final check and project task	3	0	0	1	0	Project	✓
	Total	**64**	**45**	**35**	**21**	**23**	**15 + 1**	**4**

Note: Obj. = stated learning objectives; Callout = note, warning, and tip boxes; Check = knowledge check with multiple-choice and essay items. Two further tables appear in the appendices. Source: compiled from Supiyandi et al. (2026).

The sequence moved from structure to presentation and from isolated elements to integrated pages, which is consistent with the recommendation to build schemas for individual elements before asking learners to combine them (Sweller et al., 2019). The ordering within Phase III mirrors the finding of Kim et al. (2019) that combining new layout properties is harder than learning them singly: the box model is taught on its own before flexbox, and flexbox is taught before media queries are layered on top of it (Supiyandi et al., 2026, pp. 71–81). The density of code listings rose at two points, Chapter 10, where CSS syntax and three ways of attaching style sheets are introduced, and Chapter 15, where each error is shown in an incorrect and a corrected version. In both places the larger number of listings reflects contrasting examples rather than additional content.

4.2. A Recurring Chapter Micro-Structure

Every content chapter followed the same seven-part structure: *Konsep Dasar* (basic concept), *Alur Pemahaman Konsep* (learning objectives, four or five per chapter), an exposition built around numbered code listings, callouts labelled *Catatan* (note), *Perhatian* (caution), or *Tips*, a practice task (*Latihan Praktik*), a summary (*Intisari Dasar*), and, in selected chapters, a knowledge check (*Materi Pengetahuan*), as illustrated by Chapter 1 (Supiyandi et al., 2026, pp. 1–7). The 23 callouts comprised 8 notes, 6 cautions, and 9 tips. Cautions were used to forestall misconceptions, for example that HTML is a programming language (Supiyandi et al., 2026, p. 3), while tips pointed to efficient working habits.

From a cognitive load perspective, this regularity reduces extraneous load because learners do not need to work out how each chapter is organised, and it allows them to anticipate where objectives, examples, and summaries will appear (Sweller et al., 2019). Pairing 45 code listings with 35 figures of rendered browser output functions as a worked-example strategy in which the code and its effect are presented together, the condition under which worked examples are most effective (Atkinson et al., 2000). This pairing also compensates for a specific property of web authoring noted earlier: because browsers tolerate faulty markup, novices cannot rely on error messages and must learn to judge code by its visual result (Becker et al., 2019; Park et al., 2015). Listings without a figure were concentrated in the troubleshooting chapter, where the contrast between incorrect and corrected code replaces the rendered output.

4.3. Scaffolding and Fading Across Practice Tasks

The 15 practice tasks showed a clear progression in the amount of support provided. In Chapter 1, learners modified a file they had just typed, replacing a greeting with their own name and adding a paragraph (Supiyandi et al., 2026, p. 6). In Chapter 2, they rewrote the document skeleton from memory, a retrieval task that consolidated the basic structure (Supiyandi et al., 2026, p. 12). Chapters 3 to 13 asked learners to build small, bounded artefacts from a specification, such as a holiday article, a mini profile card, a lesson timetable, a login form, a school page skeleton, and a responsive product display. Chapter 14 then modelled a complete project step by step, from a wireframe sketch and folder structure through HTML construction and CSS styling to saving and testing, before asking learners to build their own profile page (Supiyandi et al., 2026, pp. 82–88). Finally, Chapter 16 removed the step-by-step guidance and specified only the requirements (Supiyandi et al., 2026, pp. 96–97).

This trajectory corresponds closely to the classic notion of scaffolding with fading (Wood et al., 1976). Reiser's (2004) structuring mechanism is visible in the use of wireframes and a prescribed folder structure in Chapter 14, which break a complex task into manageable parts. The decision to postpone the fully independent project until the end of the book is consistent with the argument that open-ended problem solving is unproductive for learners who have not yet acquired the relevant schemas (Kirschner et al., 2006). The independent project offers three themes: a community mini-website, a small family business page, and *Pesona Daerahku* ("the charm of my region"), which features local tourist destinations or cuisine (Supiyandi et al., 2026, p. 96). Anchoring the project in learners' communities increases its authenticity, a quality associated with the effectiveness of project-based learning (Kokotsaki et al., 2016), and suits vocational settings in which graduates are expected to produce work for local clients.

4.4. Errors as Learning Content

Chapter 15 was devoted entirely to common errors. It opened with the observation that browsers are forgiving, which helps beginners but hides mistakes until the page looks wrong (Supiyandi et al., 2026, p. 90), and it then presented 11 error categories, each with its symptom, an incorrect listing, a corrected listing, and an explanation (**Table 3**). The chapter introduced browser developer tools, the HTML validator, and syntax highlighting as systematic diagnostic aids (Supiyandi et al., 2026, p. 93), and it closed with a "bug hunt" exercise in which learners retype a short page containing five planted errors, locate and fix them, and confirm the result with the validator (Supiyandi et al., 2026, p. 94). Passing the validator without errors was also one of the requirements of the final project.

Table 3. Error categories treated in Chapter 15

Layer	Error category
HTML (8)	Unclosed tag; closing tags in the wrong order; unclosed attribute quotation mark; image not displayed because of an incorrect path; file saved as plain text; page content written inside the head element; missing meta charset; label not linked to its field or duplicated id
CSS (3)	Missing semicolon or curly brace; selector that does not match the HTML; external style sheet not linked

Source: Supiyandi et al. (2026, pp. 90–93).

Treating errors as explicit content is supported by evidence that errors followed by corrective feedback can enhance learning (Metcalfe, 2017). The categories selected overlap substantially with the syntax problems that Park et al. (2015) observed among students in an introductory web development course, and the emphasis on the validator reflects their finding that validation shapes how such errors are resolved. Teaching learners to infer causes from symptoms also partly offsets the weakness of automated error feedback for novices (Becker et al., 2019). One limitation is that the chapter addresses mainly syntactic and linkage errors; semantic errors in layout, which persist even among learners who have mastered syntax (Kim et al., 2019), receive less attention and could be added in a later revision.

4.5. Assessment and Constructive Alignment

The final project specified nine mandatory requirements (Supiyandi et al., 2026, pp. 96–97). **Table 4** traces each requirement to the section and page in which the corresponding knowledge was taught and practised. Every requirement was covered by at least one earlier section. The only chapters not tied to a specific requirement were Chapter 1, which is introductory, Chapters 11 and 12, which cover the colour, typography, and spacing that shape the page's visual quality, and Chapter 14, which rehearses the whole set of requirements in a guided project. This pattern indicates a high degree of constructive alignment between objectives, learning activities, and assessment (Biggs, 1996). The project was to be scored with an analytic rubric of five weighted aspects on a 1–4 scale (Supiyandi et al., 2026, p. 97), a format that makes expectations explicit and supports consistent judgement (Jonsson & Svingby, 2007).

Table 4. Traceability of final project requirements to textbook sections

No.	Project requirement	Taught in section (page)
1	Complete HTML skeleton with lang, meta charset, meta viewport, and title	2.4 (p. 9); 13.4 (p. 77)
2	Semantic elements header, nav, main, and footer	9.5 (p. 52)
3	At least two sequential heading levels	3.3 (p. 14)
4	At least one image with descriptive alt text	4.4 (p. 21); 5.5 (p. 27)
5	At least one list, one table with th cells, and one form with three labelled fields	6.3 (p. 32); 7.3 (p. 37); 8.5 (p. 44)
6	At least three links, including an in-page (#id) link and an external link	5.3 (p. 25); 5.6 (p. 29)
7	External style sheet using element, class, and id selectors	4.5 (p. 23); 10.5–10.6 (pp. 58–59)
8	Flexbox layout and at least one media query for mobile screens	13.3–13.5 (pp. 76–78)
9	Passes the HTML validator without errors	15.5 (p. 93)

Source: requirements from Supiyandi et al. (2026, pp. 96–97); section and page numbers refer to the same book.

The textbook also adopted a mastery criterion. Learners who scored below 80% on a knowledge check were instructed to revisit the chapters linked to the items they had missed before moving on (Supiyandi et al., 2026, pp. 48, 56, 64, 96), which corresponds to the corrective loop of mastery learning (Guskey, 2007). The knowledge checks combined five multiple-choice items with two essay items, one of which required learners to design and write code, so that both recognition and production were assessed.

4.6. Design Gaps and a Revision and Validation Agenda

The analysis also identified gaps. First, formative knowledge checks appeared in only three of the 15 content chapters (Chapters 8, 9, and 10), together with the final check in Chapter 16. Because mastery learning depends on a formative assessment and corrective loop after every unit (Guskey, 2007), the mastery criterion could not operate for most of the book. Adding short checks to the remaining chapters is the most consequential revision. Second, several elements referred to in the text, namely the answer key, the formula for calculating mastery, the items of the final knowledge check, and the descriptors of the five-aspect rubric, were not present in the analysed edition (Supiyandi et al., 2026, pp. 96–106). Supplying them, ideally with exemplar projects at each rubric level, would strengthen both self-assessment and scoring reliability (Jonsson & Svingby, 2007). Third, accessibility was addressed implicitly through alt text, labelled form fields, and

semantic elements, but it was not framed explicitly against the Web Content Accessibility Guidelines (W3C, 2023). Making that link explicit would add value at little cost. Fourth, the guidance on layout errors could be extended, as noted above.

Most importantly, the present analysis establishes design quality, not learning effectiveness. The next step is a staged empirical evaluation: expert review by content and instructional-design specialists; one-to-one and small-group formative trials with novice learners; and a field test in vocational high schools and teacher-education programmes using a quasi-experimental design with pre- and post-tests of HTML and CSS knowledge and a rubric-scored performance task. Collecting learners' code and validator output during the trials would also allow the error categories in Chapter 15 to be checked against the errors that Indonesian novices actually make, extending the error-analysis tradition of Park et al. (2015) to a new context. Such evidence would respond directly to the call for more rigorous research on HTML and CSS instruction (Kharchenko & Semerikov, 2026).

For teachers and curriculum developers, the findings suggest a practical checklist for beginner web-programming materials: use a consistent chapter template; pair every listing with its rendered result; sequence layout concepts one at a time; fade support from modification to independent creation; treat errors and diagnostic tools as content; trace every project requirement back to a chapter; and place a formative check with a remedial path after every unit. The framework in Figure 1 can be applied to evaluate other introductory materials, including those for related subjects in vocational informatics.

5. CONCLUSION

This study analysed the pedagogical design of a 16-chapter Indonesian textbook for novice web authors. In answer to RQ1, the textbook embodied all six principles of the framework: it reduced extraneous load through a recurring seven-part chapter structure and code–output pairs, scaffolded practice with gradual fading, aligned objectives with assessment, adopted a mastery criterion, treated errors as learning content, and culminated in an authentic, rubric-scored project. In answer to RQ2, the 64 objectives, 45 code listings, 35 output figures, 23 callouts, and 15 practice tasks were distributed evenly across four phases that moved from structure to presentation and from isolated elements to integrated pages, and all nine project requirements were traceable to earlier chapters. In answer to RQ3, the main gaps were the limited number of formative knowledge checks, missing supporting assessment materials, implicit rather than explicit treatment of accessibility, and limited coverage of semantic layout errors. The study was limited to a single textbook and to design analysis by its own authors, and it did not measure learning. Future research should revise the textbook along the lines identified here and evaluate it empirically with novice learners in vocational and teacher-education settings.

6. ACKNOWLEDGEMENTS

The authors thank [names or institutions] for [support provided]. [Add funding information, if any, or delete this section.]

7. REFERENCES

Atkinson, R. K., Derry, S. J., Renkl, A., & Wortham, D. (2000). Learning from examples: Instructional principles from the worked examples research. *Review of Educational Research*, *70*(2), 181–214. https://doi.org/10.3102/00346543070002181

Becker, B. A., Denny, P., Pettit, R., Bouchard, D., Bouvier, D. J., Harrington, B., Kamil, A., Karkare, A., McDonald, C., Osera, P.-M., Pearce, J. L., & Prather, J. (2019). Compiler error messages considered unhelpful. In *Proceedings of the Working Group Reports on Innovation and Technology in Computer Science Education* (pp. 177–210). ACM. https://doi.org/10.1145/3344429.3372508

Biggs, J. (1996). Enhancing teaching through constructive alignment. *Higher Education*, *32*(3), 347–364. https://doi.org/10.1007/BF00138871

Bowen, G. A. (2009). Document analysis as a qualitative research method. *Qualitative Research Journal*, *9*(2), 27–40. https://doi.org/10.3316/QRJ0902027

Freeman, S., Eddy, S. L., McDonough, M., Smith, M. K., Okoroafor, N., Jordt, H., & Wenderoth, M. P. (2014). Active learning increases student performance in science, engineering, and mathematics. *Proceedings of the National Academy of Sciences*, *111*(23), 8410–8415. https://doi.org/10.1073/pnas.1319030111

Guskey, T. R. (2007). Closing achievement gaps: Revisiting Benjamin S. Bloom's "Learning for Mastery." *Journal of Advanced Academics*, *19*(1), 8–31. https://doi.org/10.4219/jaa-2007-704

Jonsson, A., & Svingby, G. (2007). The use of scoring rubrics: Reliability, validity and educational consequences. *Educational Research Review*, *2*(2), 130–144. https://doi.org/10.1016/j.edurev.2007.05.002

Kharchenko, O. Ye., & Semerikov, S. O. (2026). Teaching web design in secondary schools through HTML and CSS: A systematic synthesis of research evidence and pedagogical implications. *Educational*

Dimension, *14*, 374–388. https://doi.org/10.31812/ed.1061

Kim, M. C., Park, T. H., Liu, R., & Forte, A. (2019). Understanding learning curves and trajectories in CSS layout. In *Proceedings of the 50th ACM Technical Symposium on Computer Science Education* (pp. 504–510). ACM. https://doi.org/10.1145/3287324.3287387

Kirschner, P. A., Sweller, J., & Clark, R. E. (2006). Why minimal guidance during instruction does not work: An analysis of the failure of constructivist, discovery, problem-based, experiential, and inquiry-based teaching. *Educational Psychologist*, *41*(2), 75–86. https://doi.org/10.1207/s15326985ep4102_1

Kokotsaki, D., Menzies, V., & Wiggins, A. (2016). Project-based learning: A review of the literature. *Improving Schools*, *19*(3), 267–277. https://doi.org/10.1177/1365480216659733

Medeiros, R. P., Ramalho, G. L., & Falcão, T. P. (2019). A systematic literature review on teaching and learning introductory programming in higher education. *IEEE Transactions on Education*, *62*(2), 77–90. https://doi.org/10.1109/TE.2018.2864133

Metcalfe, J. (2017). Learning from errors. *Annual Review of Psychology*, *68*, 465–489. https://doi.org/10.1146/annurev-psych-010416-044022

Novaliendry, D., Huda, A., Sanita, D., Putra, D. A., Nasution, M. D. F., Putra, R. S., & Hidayati, R. N. (2021). Android-based network services application learning media for vocational high schools. *International Journal of Interactive Mobile Technologies*, *15*(20), 83. https://doi.org/10.3991/ijim.v15i20.23745

Park, T. H., Dorn, B., & Forte, A. (2015). An analysis of HTML and CSS syntax errors in a web development course. *ACM Transactions on Computing Education*, *15*(1), 1–21. https://doi.org/10.1145/2700514

Reiser, B. J. (2004). Scaffolding complex learning: The mechanisms of structuring and problematizing student work. *Journal of the Learning Sciences*, *13*(3), 273–304. https://doi.org/10.1207/s15327809jls1303_2

Robins, A., Rountree, J., & Rountree, N. (2003). Learning and teaching programming: A review and discussion. *Computer Science Education*, *13*(2), 137–172. https://doi.org/10.1076/csed.13.2.137.14200

Supiyandi, Mailok, R., & Rizal, C. (2026). *HTML dan CSS dasar: Panduan praktis membangun halaman web untuk pemula* [Basic HTML and CSS: A practical guide to building web pages for beginners]. CV. Raskha Media Group.

Sweller, J., van Merriënboer, J. J. G., & Paas, F. (2019). Cognitive architecture and instructional design: 20 years later. *Educational Psychology Review*, *31*(2), 261–292. https://doi.org/10.1007/s10648-019-09465-5

W3C. (2023). *Web content accessibility guidelines (WCAG) 2.2*. World Wide Web Consortium. https://www.w3.org/TR/WCAG22/

Wood, D., Bruner, J. S., & Ross, G. (1976). The role of tutoring in problem solving. *Journal of Child Psychology and Psychiatry*, *17*(2), 89–100. https://doi.org/10.1111/j.1469-7610.1976.tb00381.x