MCF

Modular Curriculum Format

Version 1.0 Specification

MCF 1.0 - Updated July 2026

Document status

This document defines MCF 1.0, the first public version of the Modular Curriculum Format. MCF is a source-format standard. It defines how a course is authored and organized so that independent tools can read it. It does not prescribe how a compiler, website, mobile application, or other implementation must render or distribute a course.

Normative language

The key words MUST, MUST NOT, REQUIRED, SHALL, SHALL NOT, SHOULD, SHOULD NOT, RECOMMENDED, MAY, and OPTIONAL are to be interpreted as requirement levels. Examples and implementation notes are non-normative unless explicitly stated otherwise.

Contents

1. Purpose and scope

MCF (Modular Curriculum Format) is an open, human-readable format for describing complete courses. It is intended for course authors, visual course-making tools, compilers, learning platforms, and other software that reads or writes structured educational content.

MCF is designed to support broad course creation while remaining independent of any particular implementation. A Node.js compiler may produce an offline static HTML course. A Python compiler may produce HTML, PDF, or another representation. A web application may import the same source into an online learning platform.

1.1 Design principles

- Human-readable: authors can understand and edit course source directly.

- Markdown-based: ordinary instructional content uses Markdown rather than a new prose language.
- Implementation-independent: MCF does not require Node.js, Python, HTML, a browser, a database, or any framework.
- Structured: course, chapter, lesson, activity, and question boundaries are explicit.
- Portable: paths and identifiers are stable and machine-readable.
- Evolvable: later MCF versions may add official activity types, blocks, and question types.

1.2 Out of scope

- Generated HTML, CSS, JavaScript, or directory layouts.
- Runtime behavior, learner accounts, progress storage, analytics, certificates, or badges.
- Themes, visual design, accessibility implementation details, or user-interface layout.
- Compiler architecture or programming language.
- Labs, simulations, interactive diagrams, coding sandboxes, microphone analysis, or music-specific interactions. These may be standardized in later MCF versions.

2. Terminology

Term	Meaning
Course	The complete MCF package rooted at manifest.yaml.
Chapter	An ordered group of lessons stored in its own chapter directory.
Lesson	One .mcf document containing ordered activities.
Activity	A named section of a lesson with an educational purpose: notes, practice, or assessment.
Block	A structured fenced element inside an activity. MCF 1.0 defines mcf-question.
Question type	The response model used by an mcf-question, such as multiple_choice or numeric.
Rich content	Markdown text that may include LaTeX, images, audio, video, links, tables, and code.
Reader	Software that reads MCF course source. A compiler is one kind of reader.
Writer	Software that creates or edits valid MCF source.

3. Course package

A valid MCF 1.0 course MUST use the following package model:

```
my-course/
├── manifest.yaml
├── chapters/
│   ├── motion/
│   │   ├── chapter.yaml
│   │   └── lessons/
│   │       ├── 01-introduction.mcf
│   │       └── 02-velocity.mcf
│   └── forces/
│       ├── chapter.yaml
│       └── lessons/
│           └── 01-newtons-laws.mcf
└── assets/
    ├── images/
    ├── audio/
    ├── video/
    └── files/
```

- The course root MUST contain exactly one manifest.yaml.
- The course root MUST contain a chapters directory.
- Each chapter declared by the manifest MUST identify a chapter directory under chapters/.
- Each chapter directory MUST contain one chapter.yaml and one lessons/ directory.
- The assets/ directory is OPTIONAL. Its subdirectories are conventional and MAY be arranged differently.

- Readers MUST use YAML declarations for ordering. Readers MUST NOT infer order from filenames or directory names.

4. Root manifest

manifest.yaml defines course-level metadata and the ordered list of chapters.

```
mcf: "1.0"
id: intro-physics
title: "Introduction to Physics"
language: en
description: "A first course in motion and forces."
authors:
  - "Jane Doe"
license: CC-BY-4.0
version: "1.0.0"

chapters:
  - source: chapters/motion
  - source: chapters/forces
```

Field	Required	Type	Meaning
mcf	Yes	string	MCF specification version. For this document: 1.0.
id	Yes	identifier	Stable course identifier.
title	Yes	string	Human-readable course title.
language	Yes	string	Primary course language, preferably a BCP 47 tag.
chapters	Yes	list	Ordered chapter source directories.
description	No	string	Course summary.
authors	No	list of strings	Course authors or organizations.
license	No	string	Declared license for the authored course content.
version	No	string	Course revision chosen by the author.
cover	No	path or URL	Course cover image.

4.1 Chapter entries

Each chapters entry MUST contain source. The path MUST identify a chapter directory relative to the course root.

```
chapters:
  - source: chapters/motion
  - source: chapters/forces
```

5. Chapter package

Each chapter directory contains chapter.yaml and a lessons/ directory.

```
id: motion
title: "Motion"
description: "Distance, speed, velocity, and acceleration."

lessons:
  - lessons/01-introduction.mcf
  - lessons/02-distance.mcf
  - lessons/03-speed.mcf
  - lessons/04-velocity.mcf
  - lessons/05-acceleration.mcf
  - lessons/06-review.mcf
```

Field	Required	Type	Meaning
id	Yes	identifier	Stable chapter identifier.
title	Yes	string	Human-readable chapter title.

lessons	Yes	list of paths	Ordered lesson files relative to the chapter directory.
description	No	string	Chapter summary.

A lesson file MAY be referenced by more than one chapter or course, but each reference MUST resolve to a valid file.

6. Lesson files

A .mcf file is a Markdown document with YAML frontmatter followed by one or more activity containers. Content outside an activity container is invalid in MCF 1.0, except for blank lines and comments.

```
---
id: velocity
title: "Velocity"
---

:::mcf-activity
type: notes
id: velocity-notes
title: "Understanding Velocity"
:::

Velocity is displacement over time.

$$
v = \frac{\Delta x}{\Delta t}
$$

:::mcf-end
```

Frontmatter field	Required	Meaning
id	Yes	Stable lesson identifier.
title	Yes	Human-readable lesson title.
description	No	Lesson summary.
authors	No	Lesson-specific attribution.
license	No	Lesson-specific license override.

Chapter membership and lesson ordering are defined by chapter.yaml, not by lesson frontmatter.

7. Activity syntax

A lesson is an ordered sequence of activities. Each activity begins with an mcf-activity directive and ends with mcf-end.

```
:::mcf-activity
type: practice
id: velocity-practice
title: "Velocity Practice"
:::

Activity content goes here.

:::mcf-end
```

Field	Required	Meaning
type	Yes	One of notes, practice, or assessment.
id	Yes	Stable activity identifier unique within the lesson.
title	No	Human-readable activity title.
passing_score	Assessment only, optional	Number from 0 through 1. Meaningful to implementations that score assessments.
randomize	Practice or assessment, optional	Boolean request to randomize question order.
question_pool_size	Practice or assessment, optional	Positive integer limiting questions selected from the activity.

7.1 notes

A notes activity provides instructional content. It MAY contain rich content and MAY contain question blocks when an author intentionally embeds a check within instruction. The presence of a question does not change the activity type.

7.2 practice

A practice activity provides formative work intended for learning, repetition, or feedback. It MAY contain rich content and zero or more question blocks.

7.3 assessment

An assessment activity provides evaluative work. It MAY contain rich content and zero or more question blocks. MCF describes the assessment content; it does not require a particular grading interface, attempt policy, or storage mechanism.

8. Rich content

All prose inside an activity and all fields explicitly defined as rich content use the same content model: Markdown with optional LaTeX and inline media.

8.1 Markdown

- Implementations MUST support CommonMark-compatible paragraphs, headings, emphasis, lists, links, block quotes, fenced code, and images.
- Implementations SHOULD support GitHub-style tables.
- Raw HTML support is implementation-defined and SHOULD be disabled by default for safety.

8.2 LaTeX mathematics

```
Inline mathematics: $v = d/t$

Display mathematics:

$$
F = ma
$$
```

Implementations that cannot render LaTeX MUST preserve the source text or provide a clear fallback.

8.3 Inline media

Images use standard Markdown. Audio and video use MCF media directives.

```
![Force diagram](../../../assets/images/force-diagram.png)

@[audio](../../../assets/audio/pronunciation.mp3 "Pronunciation example")

@[video](../../../assets/video/velocity.mp4 "Velocity demonstration")

@[video](youtube:VIDEO_ID "Online demonstration")
```

Syntax	Purpose
![alt](source)	Image.
@[audio](source "label")	Audio clip.
@[video](source "label")	Video clip or supported remote video reference.

- Media MAY appear anywhere rich content is allowed, including activity prose, question prompts, options, hints, and explanations.
- A source may be a local path, an absolute URL, or a defined provider reference such as youtube:VIDEO_ID.
- Local media is portable with the course. Remote media requires network access.
- Readers MUST preserve the media reference even when they cannot render it.

9. mcf-question block

MCF 1.0 defines one structured block: mcf-question.

````
```mcf-question
id: q1
type: multiple_choice
prompt: |
  Which quantity includes direction?

options:
  - id: speed
    text: "Speed"
  - id: velocity
    text: "Velocity"

answer: velocity
hint: "Think about vectors."
explanation: "Velocity includes magnitude and direction."
points: 1
```
````

Field	Required	Meaning
id	Yes	Question identifier unique within the lesson.
type	Yes	One of the question types in Section 10.
prompt	Yes	Rich content presented to the learner.
hint	No	Rich content that may assist the learner.
explanation	No	Rich content explaining the answer or concept.
points	No	Non-negative number. Default 1.
required	No	Boolean. Default true.
options	Type-specific	Answer choices for choice-based questions.
answer	Type-specific	Machine-readable expected answer where applicable.
tolerance	Numeric only, optional	Allowed absolute numeric difference. Default 0.
minimum_words	Essay only, optional	Positive integer. Minimum number of whitespace-delimited words required for response completion.
minimum_sentences	Essay only, optional	Positive integer. Minimum number of sentences required for response completion.
keywords	Essay only, optional	Non-empty list of expected concept words or phrases used for completion checking.
minimum_keywords	Essay only, optional	Positive integer no greater than the number of keywords. Minimum distinct listed keywords that must appear.

MCF does not prescribe when hints or explanations are displayed. An implementation MAY use activity type as guidance, but presentation behavior is outside the source-format standard.

10. Question types

MCF 1.0 defines the following question types.

Type	Required fields	Answer semantics
multiple_choice	options, answer	answer is one option id.
multiple_select	options, answer	answer is a list of option ids. Order is not significant.
true_false	answer	answer is true or false.
numeric	answer; tolerance optional	Submitted number is correct when absolute difference is within tolerance.
short_answer	answer	Case-insensitive trimmed string comparison.
essay	minimum_words, minimum_sentences, keywords, and minimum_keywords optional	Captures a written response. Optional structural criteria determine completion, not objective correctness.

10.1 Option objects

```
options:
  - id: a
    text: "Speed"
  - id: b
    text: |
      Velocity, measured in $\text{m/s}$
answer: b
```

Option text is rich content. Option identifiers MUST be unique within the question.

10.2 Multiple select

````
```mcf-question
id: q2
type: multiple_select
prompt: "Select all vector quantities."
options:
  - id: displacement
    text: "Displacement"
  - id: speed
    text: "Speed"
  - id: velocity
    text: "Velocity"
answer: [displacement, velocity]
```
````

10.3 Numeric

````
```mcf-question
id: q3
type: numeric
prompt: "A car travels 60 m in 5 s. Find its speed in m/s."
answer: 12
tolerance: 0.1
```
````

10.4 Short answer

A reader performing automatic evaluation SHOULD trim leading and trailing whitespace and compare case-insensitively. More advanced matching is not standardized in MCF 1.0.

10.5 Essay

An essay question records a free-form written response. It MAY define structural completion requirements using minimum_words, minimum_sentences, keywords, and minimum_keywords. These fields determine whether the learner has provided a sufficiently complete response; they MUST NOT be treated as proof that the response is factually correct, high quality, or deserving of a particular grade.

When no essay completion fields are present, a non-empty response satisfies the response-completion requirement. When one or more fields are present, every declared requirement MUST be satisfied before the question is complete.

Word counting MUST split the trimmed response on one or more whitespace characters. Sentence counting MUST count non-empty text segments ending in a period, question mark, or exclamation mark; a final non-empty segment without terminal punctuation counts as one sentence. Keyword matching MUST be case-insensitive and MUST match whole words for single-word keywords. Multi-word keyword phrases MUST match as normalized consecutive text.

keywords declares the expected concepts. minimum_keywords declares how many distinct listed keywords or phrases must appear. If keywords is present and minimum_keywords is omitted, all listed keywords are required. minimum_keywords MUST NOT exceed the number of listed keywords.

````
```mcf-question
id: q4
type: essay
prompt: |
  Explain how velocity differs from speed.
minimum_words: 40
minimum_sentences: 3
keywords:
  - direction
  - displacement
  - velocity
minimum_keywords: 2
```
````

A conforming reader that evaluates essay completion MUST report which declared requirements remain unmet. It MAY save the response continuously. It MUST preserve the distinction between complete and correct.

11. Identifiers and paths

11.1 Identifiers

Identifiers MUST match:

```
[a-z][a-z0-9._-]*
```

- Course IDs are unique within a catalog or implementation-defined scope.
- Chapter IDs are unique within the course.
- Lesson IDs are unique within the course.
- Activity IDs and question IDs are unique within the lesson.
- Option IDs are unique within the question.
- Changing a title SHOULD NOT require changing its identifier.

11.2 Paths

- All local paths are resolved relative to the file containing the reference unless a section explicitly states course-root-relative.
- manifest.yaml chapter sources are course-root-relative.
- chapter.yaml lesson paths are chapter-directory-relative.
- Paths MUST use forward slashes.
- Paths MUST NOT escape the course root.
- A referenced local file that does not exist makes the course invalid.
- Remote URLs are valid but reduce offline portability.

12. Validation and conformance

12.1 Valid MCF course

- The package satisfies the directory requirements in Section 3.
- manifest.yaml and every chapter.yaml contain valid YAML and required fields.
- Every declared chapter, lesson, and local asset exists.
- Every .mcf file has valid frontmatter and correctly nested activity containers.
- Every mcf-question satisfies the schema for its question type.
- Identifiers satisfy syntax and uniqueness rules.

- The declared MCF version is supported by the reader.

12.2 MCF 1.0 reader

- MUST read the package, manifest, chapters, lessons, activities, rich content, and all core question types.
- MUST preserve declared ordering.
- MUST report invalid required structure rather than silently guessing.
- MAY render, transform, import, export, or compile the course in any manner.
- MAY omit capabilities such as grading or media playback, but MUST preserve unsupported valid source content when round-tripping.

12.3 MCF 1.0 writer

- MUST produce valid MCF 1.0 source.
- MUST generate stable, non-conflicting identifiers.
- MUST preserve unknown valid metadata when editing a course where practical.
- SHOULD provide clear authoring errors for invalid paths, nesting, fields, or answer references.

12.4 Compiler conformance

A compiler is an MCF reader that produces another representation. An MCF 1.0 compiler is conforming when it correctly reads valid MCF 1.0 source and clearly reports invalid source. MCF does not standardize the compiler's output.

13. Versioning

- The root manifest MUST declare mcf: "1.0".
- Before publication, all design changes remain work toward MCF 1.0.
- After publication, backward-compatible clarifications or optional additions may be released as 1.x.
- A change that invalidates previously valid required syntax or changes existing semantics requires MCF 2.0.
- New official activities, blocks, or question types may be added in later versions when their syntax and semantics are standardized.
- MCF 1.0 defines no vendor-extension mechanism.

14. Complete example

14.1 Directory

```
intro-physics/
├── manifest.yaml
├── chapters/
│   └── motion/
│       ├── chapter.yaml
│       └── lessons/
│           └── 01-velocity.mcf
└── assets/
    ├── images/
    │   └── displacement.png
    └── audio/
        └── question.mp3
```

14.2 manifest.yaml

```
mcf: "1.0"
id: intro-physics
title: "Introduction to Physics"
language: en
chapters:
  - source: chapters/motion
```

14.3 chapter.yaml

```
id: motion
title: "Motion"
lessons:
  - lessons/01-velocity.mcf
```

14.4 01-velocity.mcf

````
---
id: velocity
title: "Velocity"
---

:::mcf-activity
type: notes
id: velocity-notes
title: "Understanding Velocity"
:::

Velocity is displacement over time:

$$
v = \frac{\Delta x}{\Delta t}
$$

![Displacement diagram](../../../assets/images/displacement.png)

@[video](youtube:VIDEO_ID "Velocity demonstration")

:::mcf-end

:::mcf-activity
type: practice
id: velocity-practice
title: "Practice"
:::

```mcf-question
id: p1
type: numeric
prompt: |
  A cyclist travels $60\,\text{m}$ in $5\,\text{s}$.
  What is the average speed in m/s?
answer: 12
tolerance: 0.1
hint: "Divide distance by time."
explanation: |
  $$
  60 / 5 = 12\,\text{m/s}
  $$
```

:::mcf-end

:::mcf-activity
type: assessment
id: velocity-quiz
title: "Lesson Quiz"
passing_score: 0.7
:::

```mcf-question
id: q1
type: multiple_choice
prompt: |
````
````

````
  Which quantity includes direction?
options:
  - id: speed
    text: "Speed"
  - id: velocity
    text: "Velocity"
answer: velocity
```

```mcf-question
id: q2
type: short_answer
prompt: |
  Listen to the clip:

  @[audio](../../../assets/audio/question.mp3 "Question audio")

  What SI unit is commonly used for velocity?
answer: "m/s"
```

:::mcf-end
````

Appendix A. Non-normative implementation guidance

A static HTML compiler may parse YAML and Markdown, validate course structure, copy assets, render lesson pages, and emit JavaScript and CSS for interactivity. A typical implementation may use modules such as manifest parsing, lesson parsing, rendering, asset handling, and runtime generation. These modules and generated files are implementation choices and are not part of MCF conformance.

```
MCF source
    ↓
reader / compiler
    ↓
any output chosen by the implementation
    ├── static HTML course
    ├── web application records
    ├── PDF or print material
    └── another portable representation
```
```