

MOODLE & LLMS – JOIN THE RESEARCH!

Heikki Wilenius / University of Helsinki

21 Jan 2025

Please suggest co

I understand you wa
treated as Hugo tag
properties work. Th
maintaining the sam
pages.

Let's modify the pa
to:

1. Extract inline t
 2. Add them to the
- with any property t

Here are the necess

Filepath: parser.ct

Replace lines: 33-4

@clojure [<

(defn extract-inlin

[content]

(let [tag-pattern

#word but not ##v

matches (re

(when matches

(map second m

(defn parse-meta-da

[page]

(let [original-na

name)

trim-namesp

namespaces)

:block/original-name)

g/entry :trim-namespaces)

ABOUT ME

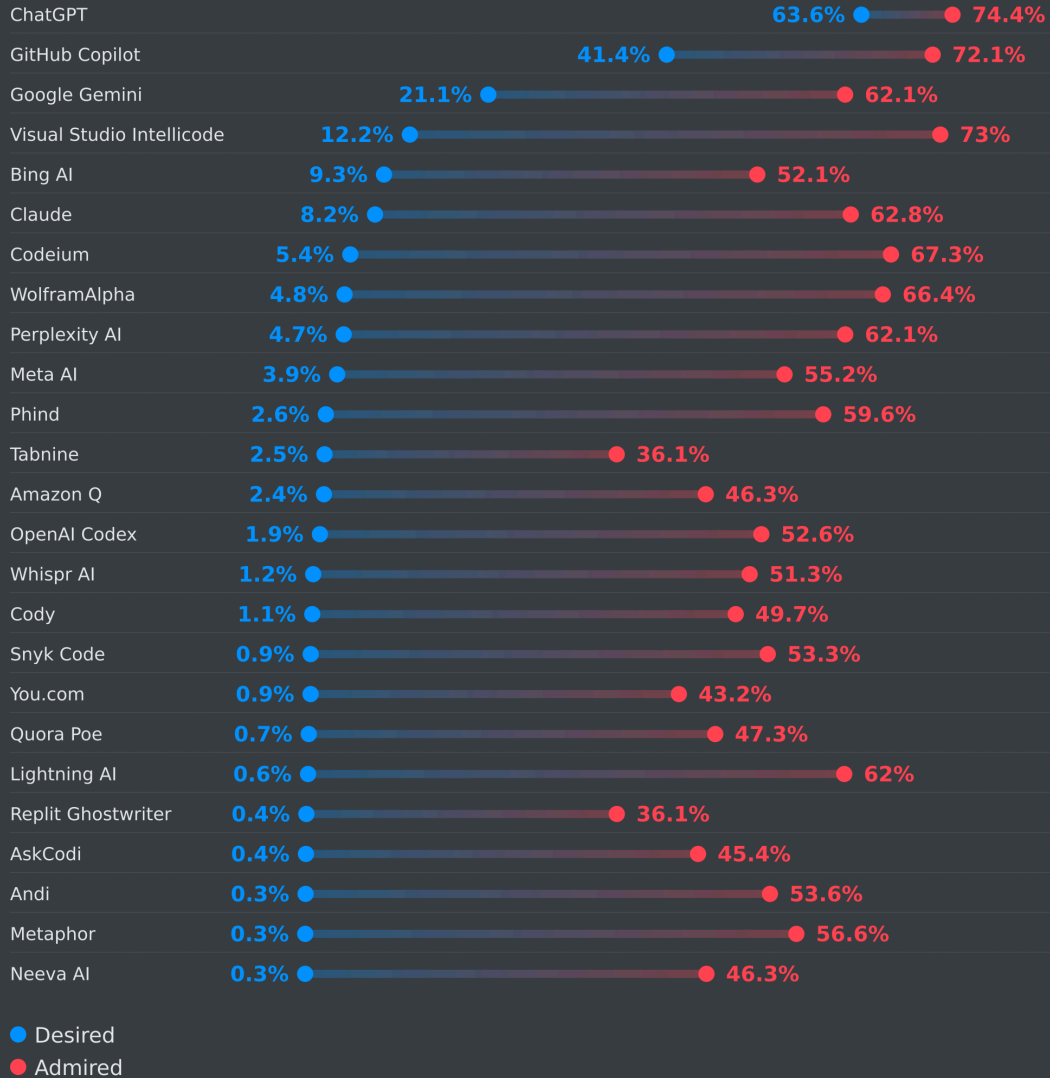
- Postdoctoral researcher
 - Anthropology, University of Helsinki
- Elearning specialist
 - Worked with Moodle since 2007
- Software developer
 - Part-time freelancer

THE RESEARCH

- How the work of software developers changes with LLMs?
- Building upon research done on changes in work organisations and the introduction of new media
- A comparison of open-source project and a small company that uses closed-source products

Admired and Desired / Desired and Admired

AI Search and Developer Tools



THE IMPORTANCE OF MOODLE (YOU!)

- An organizational culture that groks the importance of open source
- What that means for GenAI?

JOIN THE RESEARCH!

- Initial step: fill the survey at <https://redcap.link/0ny5ou41>
- You can participate no matter what is your experience level on LLMs!

PARTICIPATE IN INTERVIEWS

- Both one-on-one and group interviews
- If you're available, please indicate that when filling the survey

COMMUNICATION OF RESULTS

- Sessions at MoodleMoot DACH and MoodleMoot Global next fall
- A research article in 2026
- A monograph in 2027/2028

THANK YOU FOR LISTENING!

- heikki.wilenius@helsinki.fi
- @hwileniu@fediscience.org
- @hwileniu:matrix.org

```
(let [string-value? (string? property-value)
      object-value? (try
                      (and (not string-value?) (> (count property-value) 0))
                      (catch :default _ false))
      iterable-value? (and object-value? (> (count property-value) 1))
      value-lines (or
                   (and
                    iterable-value?
                    (let [property-lines (map (str "\n- " property-value)
                                               (property-values))]
                      (concat (property-values)
                              (concat (property-lines)
                                      (property-values))))
                    (and
                     string-value?
                     (str "\n- " property-value))
                    (and
                     object-value?
                     (str "\n- " (first (property-values))
                               (str property-value))))
                   (str value-lines))]
```

```
+ <<<<<< HEAD (Current changes) [commit 00000000]
```

```
+ =====
+ (defn extract-inline-tags
+   [content]
+   (let [tag-pattern #"(<?!#)#(\w+)" ; Matches
+         matches (re-seq tag-pattern content)]
+     (when matches
+       (map second matches))))
```

```
+ >>>>>> Snippet (Incoming changes)
```

```
(defn parse-meta-data
  [page]
  (let [original-name (get page :block/original-name)
        trim-namespaces? (config/entry :trim-namespaces)]
```

Avante

Please suggest code changes to implement that.

I understand you want inline tags (like #en) to be treated as Hugo tags, similar to how tags in page properties work. This is a good idea for maintaining the same functionality as Logseq's tag pages.

Let's implement this with a parser to handle this. We'll need to:

1. Extract inline tags when parsing links
2. Add them to the frontmatter tags list along with any property tags

Please suggest the necessary changes:

Filepath: parser.cljs
Replace lines: 33-44

```
@closure [<a>: apply this, <A>: apply all]
(defn extract-inline-tags
  [content]
  (let [tag-pattern #"(<?!#)#(\w+)" ; Matches
        matches (re-seq tag-pattern content)]
    (when matches
      (map second matches))))
```

```
(defn parse-meta-data
  [page]
  (let [original-name (get page :block/original-name)
        trim-namespaces? (config/entry :trim-namespaces)]
```

