

# Hypertag

Web templating & document generation  
with clean and powerful syntax

**Marcin Wojnarski**  
mwojnarski@paperity.org



# OUTLINE

[hypertag.io](https://hypertag.io) 

About me

Hypertag - CLEAN code

Hypertag - MODULAR code

Final remarks





# About me

Marcin Wojnarski

- Python programmer: 10 years
- Data Scientist: 20 years
- Web dev:
  - Tunedit (2009)
  - Paperity (2014), [paperity.org](http://paperity.org)
- Designed & implemented other languages before
  - DAST
  - Redex



7,883,292 Papers  
16,285 Journals

Get Paperity Reader app  
for smartphones and tablets:



Tweets by @Paperity



**Paperity**  
@Paperity

Coronaviruses can induce host cell apoptosis. We understand better and better how #SARSCoV2 works. [phys.org/news/2021-06-covid-covid19](https://www.phys.org/news/2021-06-covid-covid19)



**Coronaviruses can induce ho...**  
A large team of researchers wor...  
[phys.org](https://www.phys.org)

Search in full text of 7,883,292 papers:



## Recently added

### A Review of Circular Economy Prospects for Stainless Steelmaking Slags

The world of stainless steel production was 52 Mt in 2019, and the annual amount of slags including electric furnace, AOD converter, ladle, and casting tundish, was estimated at 15–17 Mt. Nowadays, only a minor fraction of slags from stainless steel production is utilized and a major part goes to landfilling. These slags contain high-value elements (Cr, Ni, Mo, Ti, V...) as oxides...

**Journal of Sustainable Metallurgy**, Lauri Holappa, Marko Kekkonen, Ari Jokilaakso, *et al.*

### Full phase diagram of a UV completed $N \mathcal{N} = 1$ Yang-Mills-Chern-Simons matter theory

We study the large  $N$  phase diagram of an asymptotically free UV completion of  $\mathcal{N} = 1$  SU( $N$ ) super-Yang-Mills-Chern-Simons theory coupled to a single massive fundamental scalar multiplet with a quartic superpotential coupling. We compute the effective superpotential at small gauge coupling  $\lambda \equiv N/k$ , and combine this with previous results in the literature to obtain...

**Journal of High Energy Physics**, Adar Sharon, Tal Sheaffer

### Investigation of the long-term stability of various tinctures belonging to the lamiaceae family by HPLC and spectrophotometry method

The aim of the current study was to analyze the stability of rosmarinic acid in ethanolic tinctures of lemon balm (*Melissa officinalis* L.), oregano (*Origanum vulgare* L.), peppermint (*Mentha x piperita*), rosemary (*Rosmarinus officinalis* L.), sage (*Salvia officinalis* L.), and thyme (*Thymus vulgaris* L.). High-performance liquid chromatography with diode-array detection (HPLC-DAD...

**Chemical Papers**, Beatrix Sik, Erika Hanczné Lakatos, Viktória Kapcsándi, *et al.*



# The Goal

Make HTML generation code ...

**CLEAN & MODULAR**





# CLEAN CODE





# Clean syntax? ...

## JINJA2 / DJANGO

```
{% if users %}
<ul>
{% for user in users %}
    <li>{{ user.name|escape }}</li>
{% endfor %}
</ul>
{% endif %}
```

← template  
syntax

← HTML syntax

- **boilerplate** code, only a fraction of code does **anything useful** - *braces everywhere*
- **spaghetti** code, different languages intermixed: **HTML** + **template syntax**
- **PHP legacy**: the idea of injecting template syntax into raw HTML





# Clean syntax? ...

## HYPERTAG

```
if users
  ul
    for user in users
      li | $user.name
```







# Clean syntax? ... Indentation!

## HTML

```
<div>
  <p>
    An example paragraph.
  </p>
</div>
```

## HYPERTAG

```
div
  p | An example paragraph.
```

1. one language for templating & contents
2. indentation instead of closing tags
3. special symbols used in a more efficient way





# Clean syntax? ... Indentation!

## HTML

```
<div>
  <p>
    An example paragraph.
  </p>
</div>
```

- text with occasional tags (html / templ)
- special chars signal tags <>
- blocks defined by closing tags

## HYPERTAG

```
div
→ p | An example paragraph.
```

- tree of tags with occasional text
- special chars signal text /!
- blocks defined by indentation





# Hypertag

➤ inline syntax

```
div : p | An example paragraph.
```

➤ multiline text

```
p  
| An example paragraph  
| with multiple lines.
```

➤ autoescape (or not)

```
p  
| Python & EuroPython  
/ are super <b>fun</b>!
```

➤ attributes of tags  
expressions  
loops

```
p class="main-content"  
| Found { len(items) } search results:  
for item in items:  
    ...
```





# Hypertag

➤ if-elif-else

```
if type == 1 | Car
elif type == 2 | Bus
else | Truck
```

➤ try-else

```
try
    | Price of Tesla is $cars['tesla'].
else
    | Tesla is not available.
```





## HYPERTAG

```
p class="main-content"  
  | Found { len(items) } search results:  
  for item in items:  
    i | $item
```

vs

## DJANGO & JINJA2

```
<p class="main-content">  
  Found {{ items|length }} search results:  
  {% for item in items %}  
    <i>{{ item }}</i>  
  {% endfor %}  
</p>
```





Related:

- **Slim** (ruby) ported to python as **Plim**
- **Haml** (ruby) ported to python as **HamlPy**; also **Shpaml**
- **Pug** (javascript)





# MODULAR CODE





# MODULARITY



## SEPARATION OF CONCERNS

functions  
classes

## CODE REUSE

imports



## ~~CODE DUPLICATION~~

*copy-paste...*







# HTML - code duplication

```
<table>
  <tr>
    <td class="name" >Porsche</td>
    <td class="price">200,000</td>
  </tr>
  <tr>
    <td class="name" >Jaguar</td>
    <td class="price">150,000</td>
  </tr>
  <tr>
    <td class="name" >Cybertruck</td>
    <td class="price">UNKNOWN</td>
  </tr>
</table>
```





# HTML: code duplication

```
<table>
  <tr>
    <td class="name" >Porsche</td>
    <td class="price">200,000</td>
  </tr>
  <tr>
    <td class="name" >Jaguar</td>
    <td class="price">150,000</td>
  </tr>
  <tr>
    <td class="name" >Cybertruck</td>
    <td class="price">UNKNOWN</td>
  </tr>
</table>
```





# Hypertag: concise syntax

```
table
  tr
    td class="name" | Porsche
    td class="price" | 200,000
  tr
    td class="name" | Jaguar
    td class="price" | 150,000
  tr
    td class="name" | Cybertruck
    td class="price" | UNKNOWN
  .....
```





# Hypertag: custom tags

```
table
  tableRow name="Porsche"      price="200,000"
  tableRow name="Jaguar"      price="150,000"
  tableRow name="Cybertruck"  price="UNKNOWN"
```





# Hypertag: custom tags

```
table
  tableRow "Porsche" "200,000"
  tableRow "Jaguar" "150,000"
  tableRow "Cybertruck"
```





# Hypertag: custom tags (*hypertags*)

```
% tableRow name price="UNKNOWN"  
  tr  
    td class="name" | $name  
    td class="price" | $price
```

hypertag  
definition  
block

```
table  
  tableRow "Porsche" "200,000"  
  tableRow "Jaguar" "150,000"  
  tableRow "Cybertruck"
```





# Hypertag: custom tags - body

```
table
  tableRow "Jaguar" "150,000"
    img src="jaguar.jpg"
    b : i | Money may not buy happiness,
      but I'd rather cry in a Jaguar
      than on a bus.
```

actual  
body



# The *body* attribute (@)


```
% tableRow @body name price="UNKNOWN"  
  tr  
    td class="name" | $name  
    td class="price" | $price  
  td  
    @body
```

← DOM tree gets inserted here  
(Document Object Model)

```
table  
  tableRow "Jaguar" "150,000"  
  img src="jaguar.jpg"  
  b : i | Money may not buy happiness,  
    but I'd rather cry in a Jaguar  
    than on a bus.
```







# DOM manipulation - Table of Contents (ToC)

```
%toc @document
  for heading in document['h2']
    $ id = heading.get('id', '')
    li : a href="#{id}"
        @ heading.body
```

- ← find all "h2" headings,
- ← get "id" attr of a heading ...
- ← ... to build a hyperlink,
- ← insert heading's text into ToC (h2 tag replaced with li+a)



# DOM manipulation - Table of Contents (ToC)


```
%toc @document
  for heading in document['h2']
    $ id = heading.get('id', '')
    li : a href="#{id}"
        @ heading.body
```

- ← find all "h2" headings,
- ← get "id" attr of a heading ...
- ← ... to build a hyperlink,
- ← insert heading's text into ToC (h2 tag replaced with li+a)

```
%add_toc @document
  | Table of Contents:
  ol : toc @document
  | The document:
  @document
```

- ← generate the document's ToC,
- ← then append the document itself






# DOM manipulation - Table of Contents (ToC)

Then, to add a ToC to a document ...

```
add_toc
```

```
h2 id="first" | First heading  
p | text...  
h2 id="second" | Second heading  
p | text...
```





# DOM manipulation - Table of Contents (ToC)

... and the HTML output is:

Table of Contents:

```
<ol>
  <li><a href="#first">First heading</a></li>
  <li><a href="#second">Second heading</a></li>
</ol>
```

The document:

```
<h2 id="first">First heading</h2>
<p>text...</p>
<h2 id="second">Second heading</h2>
<p>text...</p>
```





Related:

- **Mako:** %def
- **Jinja2:** macros
- **Pug:** mixins

Mako/Jinja/Pug: access is provided to actual body of a macro / mixin / %def, but manipulating the DOM is not possible and the syntax is more convoluted.

Only Hypertag implements 2-stage rendering with the generation of an intermediate DOM output, which is a precondition for unconstrained DOM manipulation. Mako/Jinja/Pug do not provide a DOM representation.

Only Hypertag provides a unified syntax: custom tags behave the same way as standard (HTML) tags and can be used with the same syntax.





# Imports

```
from my.template.widgets import %add_toc      # my/template/widgets.hy
from ..widgets import %add_toc
from datetime import $datetime as dt

context $width as W      # expected page width [px]
context $height as H    # expected page height [px]
```





# Final remarks

- Backend for **Django**: `hypertag.django.backend.Hypertag`
- Reuse of **Django filters**: `from hypertag.django.filters import $slugify`
- Many more features:
  - **external tags** - custom tags implemented as Python functions
  - **compound expressions**: `. () [] +-*// % ** << >> & ^ | :: is in .....`
  - **pipeline syntax** - any function can be used as a filter
  - **qualifiers ?!** - easily check against errors or emptiness of subexpressions
- **Any target language** can *potentially* be generated, not just HTML



Thank you :)

 [hypertag.io](https://hypertag.io)

