Hypertag
Web templating & document generation with clean and powerful syntax

Marcin Wojnarski
mwojnarski@paperture.org
OUTLINE

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
- Designed & implemented other languages before
  - DAST
  - Redex
A Review of Circular Economy Prospects for Stainless Steelmaking Slags

The world of stainless steel production was 52 Mt in 2016, and the annual amount of slags, including electric furnace, AOD converter, ladle, and casting furnace, was estimated at 16–17 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...

Full phase diagram of a UV completed N, N' - 1 Yang-Mills-Chern-Simons matter theory

We study the large N phase diagram of an asymptotically free UV completion of \( \mathcal{N} = 1 \) 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 \( \lambda \ll 1 \), and combine this with previous results in the literature to obtain...

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 rosmarinus acid in ehrlich 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...
The Goal

Make HTML generation code ...
CLEAN CODE
Clean syntax? ...

**JINJA2 / DJANGO**

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

➢ **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

```python
if users
    ul
        for user in users
            li | $user.name
```
Clean syntax? ... Indentation!

**HTML**

```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**

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

**HYPERTAG**

div
  p
  | An example paragraph.

- text with occasional tags (html / templ)
- special chars signal tags `<>
- blocks defined by closing tags
- tree of tags with occasional text
- special chars signal text `/!|
- blocks defined by indentation
Hypertag

- inline syntax
- multiline text
- autoescape (or not)
- attributes of tags
- expressions
- loops

```html
div : p | An example paragraph.
p | An example paragraph
   | with multiple lines.
p | Python & EuroPython
   / are super <b>fun</b>!
p class="main-content"
   | Found { len(items) } search results:
   for item in items:
   ...```
if type == 1 | Car
elif type == 2 | Bus
else | Truck

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

| 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

\neq \neq

CODE DUPLICATION
  copy-paste...

Hypertag - Marcin Wojnarski
<table>
<thead>
<tr>
<th>Name</th>
<th>Price</th>
</tr>
</thead>
<tbody>
<tr>
<td>Porsche</td>
<td>200,000</td>
</tr>
<tr>
<td>Jaguar</td>
<td>150,000</td>
</tr>
<tr>
<td>Cybertruck</td>
<td>UNKNOWN</td>
</tr>
<tr>
<td>Name</td>
<td>Price</td>
</tr>
<tr>
<td>-----------</td>
<td>-----------</td>
</tr>
<tr>
<td>Porsche</td>
<td>200,000</td>
</tr>
<tr>
<td>Jaguar</td>
<td>150,000</td>
</tr>
<tr>
<td>Cybertruck</td>
<td>UNKNOWN</td>
</tr>
<tr>
<td>Name</td>
<td>Price</td>
</tr>
<tr>
<td>----------</td>
<td>--------</td>
</tr>
<tr>
<td>Porsche</td>
<td>200,000</td>
</tr>
<tr>
<td>Jaguar</td>
<td>150,000</td>
</tr>
<tr>
<td>Cybertruck</td>
<td>UNKNOWN</td>
</tr>
<tr>
<td>Name</td>
<td>Price</td>
</tr>
<tr>
<td>------------</td>
<td>-------------</td>
</tr>
<tr>
<td>Porsche</td>
<td>200,000</td>
</tr>
<tr>
<td>Jaguar</td>
<td>150,000</td>
</tr>
<tr>
<td>Cybertruck</td>
<td>unknown</td>
</tr>
</tbody>
</table>
Hypertag: custom tags

<table>
<thead>
<tr>
<th>Car</th>
<th>Price</th>
</tr>
</thead>
<tbody>
<tr>
<td>Porsche</td>
<td>200,000</td>
</tr>
<tr>
<td>Jaguar</td>
<td>150,000</td>
</tr>
<tr>
<td>Cybertruck</td>
<td></td>
</tr>
</tbody>
</table>
Hypertag: custom tags (*hypertags*)

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

<table>
<thead>
<tr>
<th>Car</th>
<th>Price</th>
</tr>
</thead>
<tbody>
<tr>
<td>Porsche</td>
<td>200,000</td>
</tr>
<tr>
<td>Jaguar</td>
<td>150,000</td>
</tr>
<tr>
<td>Cybertruck</td>
<td></td>
</tr>
</tbody>
</table>
```

Hypertag - Marcin Wojnarski
Money may not buy happiness, but I'd rather cry in a Jaguar than on a bus.
The `body` attribute (@)

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

DOM tree gets inserted here (Document Object Model)
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

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

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)

generate the document's ToC,
then append the document itself
DOM manipulation - Table of Contents (ToC)

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

```python
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

```python
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**:

```python
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 :)