Hypertag

Web templating & document generation with clean and powerful syntax

Marcin Wojnarski
mwojnarski@paperity.org

paperity
open science aggregated

EuroPython 2021

OUTLINE

hypertag.io

About me

Hypertag - CLEAN code

Hypertag - MODULAR code

Final remarks

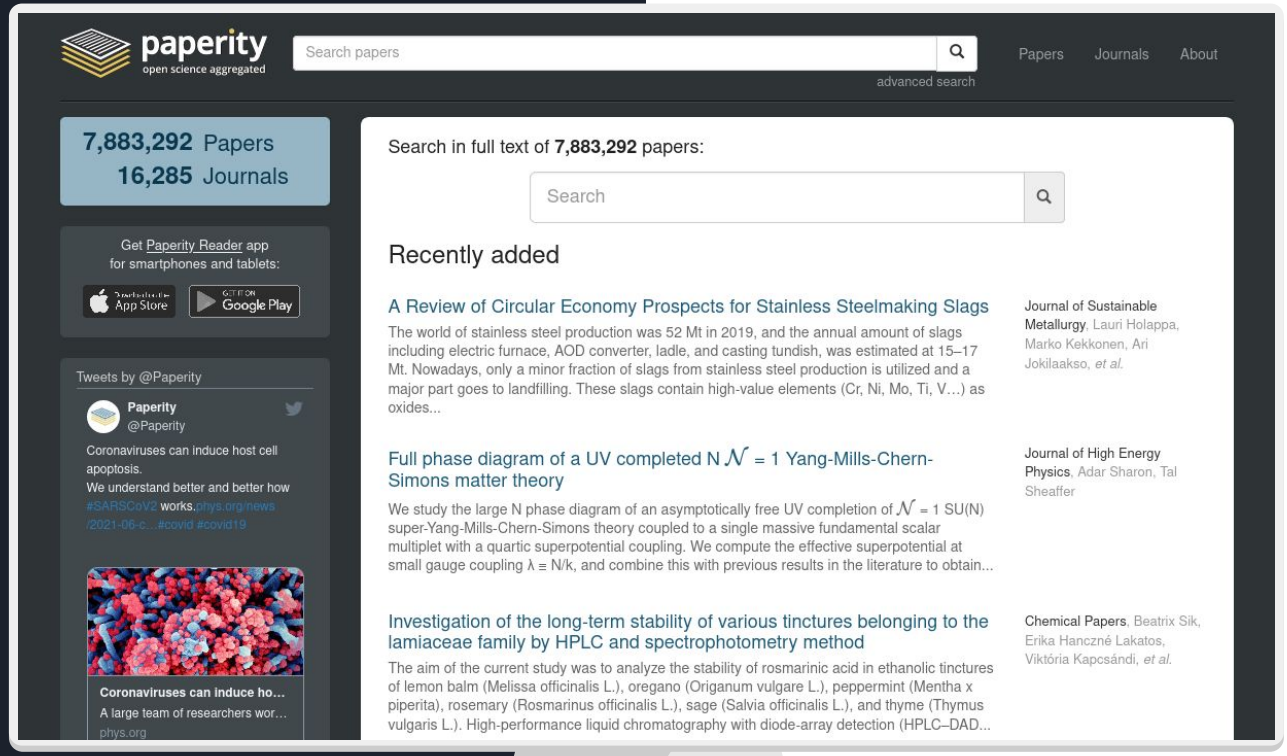
paperity

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

paperity



paperity.org

OPEN ACCESS

The Goal

Make HTML generation code ...

CLEAN & MODULAR

paperity

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

paperity

Clean syntax? ...

HYPERTAG

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

paperity

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

paperity

Clean syntax? ... Indentation!

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

paperity

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:
        ...
```

paperity

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.
```

paperity

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>
```

paperity

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...

paperity

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
```

paperity

Hypertag: custom tags

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

paperity

Hypertag: custom tags

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

paperity

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"
```

paperity

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

paperity

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.
```

paperity

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)

paperity

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

paperity

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...
```

paperity

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]
```

paperity

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

paperity

Thank you :)

hypertag.io