

Building a template engine from scratch



Vipul
@vipulbhj



shorturl.at/iyGTZ

DX and Tooling



NativeBase

Why ??

NRAF(Not Really A Framework)

The screenshot shows the GitHub repository page for `vipulbhj/nraf`. The repository is public and has 28 stars, 6 forks, and 0 watchers. The main content area displays a list of files and folders, including `docs`, `examples`, `packages`, `.gitignore`, `README.md`, `demo.gif`, `package.json`, and `yarn.lock`. The `README.md` file is selected, and its content is displayed in the main area. The repository description is "A lightweight zero dependency NodeJS framework, for build web things." The repository is tagged with `nodejs` and `javascript`. The repository has no releases published.

github.com/vipulbhj/nraf

Search or jump to... / Pull requests Issues Marketplace Explore

vipulbhj / nraf Public

Unpin Watch 0 Fork 6 Starred 28

<> Code Issues Pull requests Actions Projects Wiki Security Insights Settings

master 3 branches 0 tags

Go to file Add file <> Code

About

A lightweight zero dependency NodeJS framework, for build web things.

github.com/vipulbhj/nraf

nodejs javascript

Readme

28 stars

0 watching

6 forks

Releases

No releases published

Create a new release

packages

File	Commit Message	Time Ago
docs	Added link to quick start example	4 days ago
examples	Added support for express like , for writting modular code	17 hours ago
packages	Added support for express like , for writting modular code	17 hours ago
.gitignore	Added TODO app example	14 days ago
README.md	Update README.md	3 days ago
demo.gif	README added	24 days ago
package.json	Project moved to core package with yarn workspaces	24 days ago
yarn.lock	Project moved to core package with yarn workspaces	24 days ago

README.md

https://github.com/vipulbhj/nraf/wiki

What exactly is a template engine

Honestly, I have no clue how to define it in a meaningful way, which won't sound like something I copied from Wikipedia.

So, we will do some supervised learning, for the computer in you called your BRAIN and train it, so it can find the answer, all by itself.

MACHINE LEARNING



This is a template engine

```
<% if (user) { %>
  <h2><%= user.name %></h2>
<% } %>
```

Try EJS online at: <https://ionicabizau.github.io/ejs-playground/>.

Basic usage

```
let template = ejs.compile(str, options);
template(data);
// => Rendered HTML string

ejs.render(str, data, options);
// => Rendered HTML string

ejs.renderFile(filename, data, options, function(err, str){
  // str => Rendered HTML string
});
```



This too is a template engine

Installing

Install and update using [pip](#):

```
$ pip install -U Jinja2
```

In A Nutshell

```
{% extends "base.html" %}
{% block title %}Members{% endblock %}
{% block content %}
    <ul>
    {% for user in users %}
        <li><a href="{{ user.url }}">{{ user.username }}</a></li>
    {% endfor %}
    </ul>
{% endblock %}
```


This, umm.. is also a template engine(sort of)

[LEARN REACT](#) > [DESCRIBING THE UI](#) >

Writing Markup with JSX

JSX is a syntax extension for JavaScript that lets you write HTML-like markup inside a JavaScript file. Although there are other ways to write components, most React developers prefer the conciseness of JSX, and most codebases use it.

You will learn

- Why React mixes markup with rendering logic
- How JSX is different from HTML
- How to display information with JSX

You get the idea.

So, how do I make a template engine ?

The first step is to define constructs for our template language. This is important, we want our template language to be intuitive, provide a good DX and not be a nightmare to parse, conflicting with our host language, which in this case is HTML.

So, let's do that.

Our template lang spec

- Comments -> `{# comment any text #}`
- Value injection -> `{{ <variable_name> }}`
- loop -> `{% for item in items %} ... {% endfor %}`
- conditional ->
 - `{% if predicate_1? %}`
 - `{% elif predicate_2? %}`
 - `{% else %}`
 - `{% endif %}`



```
1 <!DOCTYPE html>
2 <html>
3   <head>
4     <title>Hello, JSWorldConf!</title>
5   </head>
6   <body>
7     <h1>Welcome, {{ user_name }} !! </h1>
8   </body>
9 </html>
```



```
1 <!DOCTYPE html>
2 <html>
3   <head>
4     <title>Hello, JSWorldConf!</title>
5   </head>
6   <body>
7     {% if isAdmin %}
8       <p>You are Admin</p>
9     {% elif isModerator %}
10      <p>You are Moderator</p>
11    {% else %}
12      <p>You are viewer</p>
13    {% endif %}
14  </body>
15 </html>
```




```
1 <!DOCTYPE html>
2 <html>
3   <head>
4     <title>Hello, JSWorldConf!</title>
5   </head>
6   <body>
7     <ul>
8       {% for product in products %}
9         <li>{{product.name}}: {{ product.price }}</li>
10      {% endfor %}
11    </ul>
12  </body>
13 </html>
```

Parsing / Tokenizing

The goal here is to parse our template file and break it down into tokens of different type.

Thanks to the simplicity of our template language, this is trivial.

Yup, that's it.

index.js

```
1  tokenize() {  
2    /*  
3     * {{ <expression> }}  
4     * {% for | if | elif | else | endfor | endif %}  
5     * {%# COMMENT #}  
6     */  
7    const re = new RegExp(/({{.*?}}|{%.*?%}|{%#.*?#})/, "igm");  
8    return this.templateStr.split(re);  
9  }
```

```
<body><h1>Hello {{ user_name }}!!</h1></body>
```

```
1 [  
2     '<body><h1> Hello ',  
3     '{{ user_name }}',  
4     ' !! </h1></body>'  
5 ]
```

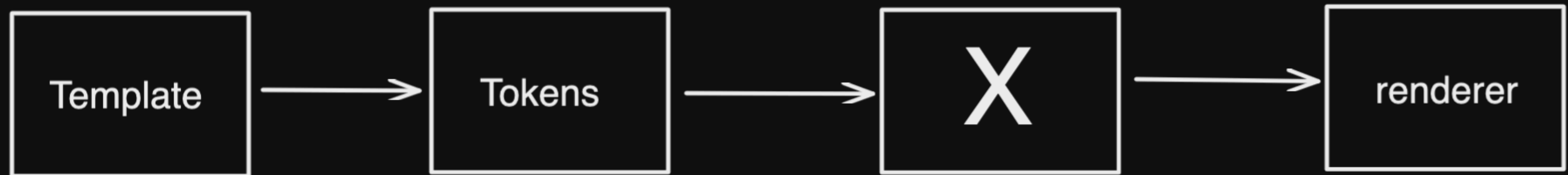


```
1 <p>Welcome, {{user_name}}!</p>
2 {% if isAdmin %}
3   <p>You are Admin.</p>
4 {% elif isModerator %}
5   <p> You are Moderator</p>
6 {% else %}
7   <p>You are not Admin.</p>
8 {% endif %}
9
10 <p>Products:</p>
11 <ul>
12 {% for product in product_list %}
13   <li>{{ product.name }}: {{ product.price }}</li>
14 {% endfor %}
15 </ul>
```



```
1 [  
2     '\n<p>Welcome, ',  
3     '{{user_name}}',  
4     '!</p>\n',  
5     '% if isAdmin %}',  
6     '\n  <p>You are Admin.</p>\n',  
7     '% elif isModerator %}',  
8     '\n  <p> You are Moderator</p>\n',  
9     '% else %}',  
10    '\n  <p>You are not Admin.</p>\n',  
11    '% endif %}',  
12    '\n\n<p>Products:</p>\n<ul>\n',  
13    '% for product in product_list %}',  
14    '\n  <li>',  
15    '{{ product.name }}',  
16    ': ',  
17    '{{ product.price }}',  
18    '</li>\n',  
19    '% endfor %}',  
20    '\n</ul>\n'  
21 ]
```


Black Box of Magic





```
1 const html = template.render({greeting: "Hello World"});
```

Tokens in Javascript

{# <comment> #} => Ignored while transpiling.

{{ <expr> }} => <expr>

{% if predicate? %} => if(predicate?) { ...

{% endif %} => }

{% for item in items %} => for(item of items) {

{% endfor %} => }



```
1 function compile(str) {
2   const tpl = [];
3   const tokens = tokenize(str);
4   for (const token of tokens) {
5     if (token.startsWith("{#")) {
6       // Comment
7       continue;
8     } else if (token.startsWith("{}")) {
9       const expr = token.substring(2, token.length - 2).trim();
10      tpl.push(expr);
11    } else if (token.startsWith("%")) {
12      const expr = token.substring(2, token.length - 2).trim();
13      const keywords = expr.split(" ");
14      const conditionalConstruct = keywords[0];
15
16      switch (conditionalConstruct) {
17        case "if":
18          tpl.push(`if(${keywords[1]}) {`);
19          break;
20        case "for":
21          tpl.push(`for(const ${keywords[1]} of ${keywords[3]}) {`);
22          break;
23        case "endif":
24        case "endfor":
25          tpl.push("}");
26          break;
27      }
28    } else {
29      // Literals
30      tpl.push(token);
31    }
32  }
33 }
```



```
1 function compile(str) {  
2   const tmp1 = [];  
3   const tokens = tokenize(str);  
4   for(const token of tokens) {  
5     if(token.startsWith("{#")) {  
6       // Comment  
7       continue;  
8     } else {  
9       // Literals  
10      tmp1.push(token);  
11    }  
12  }  
13 }
```



1 ...

2 ...

3

4

5 else if(token.startsWith("{}")) {

6 const expr = token.substring(2, token.length - 2).trim();

7 tmpl.push(expr);

8 }

9

10

11 ...

12 ...



```
[
  '\n<p>Welcome, ',
  'user_name',
  '!</p>\n\n',
  'if(isAdmin) {',
  '\n<p>You are Admin</p>\n',
  '}',
  '\n\n<ul>\n',
  'for(const product of products) {',
  '\n<li>',
  'product.name',
  '</li>\n',
  '}',
  '\n</ul>\n'
]
```

The bit that makes it tick

Demo

FIN