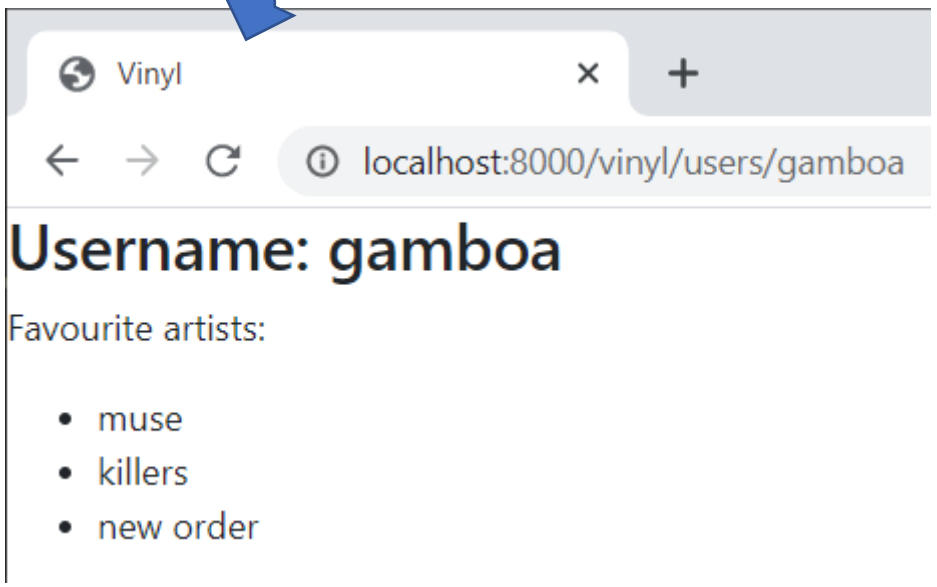
Programação na Internet

Turma i52d

Lesson 36, 39
Web APP, HTML, Forms, View and Template Engines

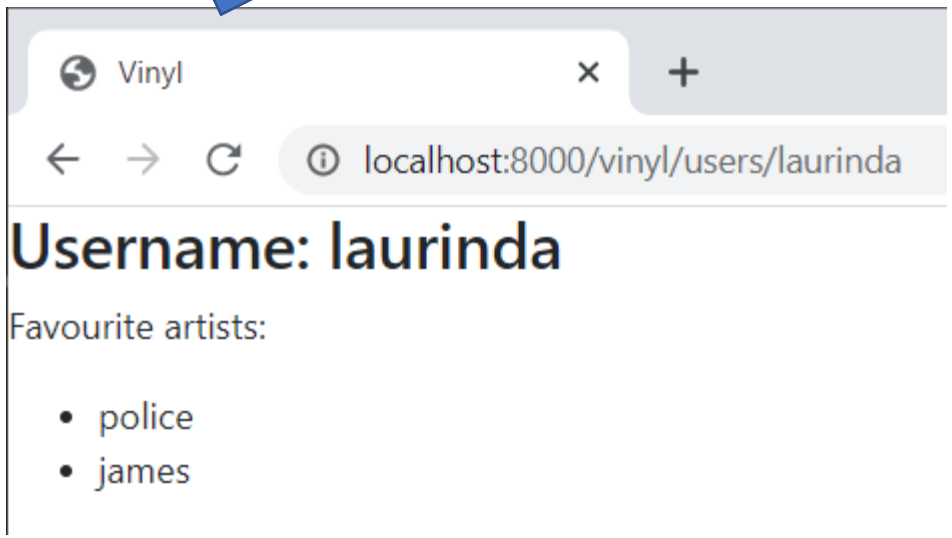
HTML e.g. gamboa

```
<html>
    <head>
        <title>Vinyl</title>
    </head>
    <body>
        <h3>Username: gamboa</h3>
        <p>Favourite artists:</p>
        <ul>
            <li>muse</li>
            <li>killers</li>
            <li>new order</li>
        </ul>
    </body>
</html>
```



HTML e.g. laurinda

```
<html>
    <head>
        <title>Vinyl</title>
    </head>
    <body>
        <h3>Username: laurinda</h3>
        <p>Favourite artists:</p>
        <ul>
            <li>police</li>
            <li>james</li>
        </ul>
    </body>
</html>
```



HTML with a Template e.g. handlebars

```
<html>
    <body>
        <h3>Username: {{username}}</h3>
        <p>Favourite artists:</p>
        <ul>
            {{#each artists}}  <!-- artists is an Array of Strings -->
                <li>{{.}}</li> <!-- current element of iteration -->
            {{/each}}
        </ul>
    </body>
</html>
```

Express render()

user is an Object with properties: *username* and *artists*

```
resp.render('userDetails', user)
```

```
<html>
    <body>
        <h3>Username: {{username}}</h3>
        <p>Favourite artists:</p>
        <ul>
            {{#each artists}}  <!-- artists is an Array of Strings -->
                <li>{{.}}</li> <!-- current element of iteration -->
            {{/each}}
        </ul>
    </body>
</html>
```

Template Engine

Template + Model => HTML

- Model also known as *context object*
- Template or View:
  - *Model binding*
  - Control flow e.g., foreach, if, etc
  - *Layouts* reused in different views

E.g. Pug, Mustache, ***Handlebars*** (few features)*, Thymleaf* (full suite of features)*, etc*

MPA versus SPA

- Multiple Page Application (since the beginning of the Web)
  - UI Web (server)
  - Web API (server)
  - !!!! Approach described in Node.js the right way. (1nd edition)
  - Use in Pi

    !=
- Single Page Application
  - UI Web (browser) => Web API (server)
  - !!!! Approach described in Node.js 8 the right way. (2nd edition)
  - Used in DAW

HTML and CSS

- HTML => document content structure
- CSS => document presentation

Structure != Presentation

CSS

Cada **regra** de estilo é constituída por duas partes:

Selector

É a ligação entre o documento HTML e o estilo a ser definido. Identifica o(s) elemento(s) a que a regra se aplica (normalmente o nome de um elemento HTML, ex: `body`, `p`, `h1`, etc.).

Bloco de Declarações

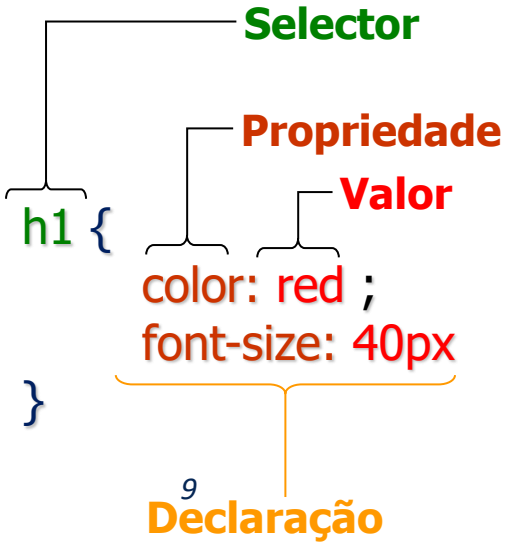
Um bloco de declarações é delimitado por '{ }' e é constituído por declarações, separadas por ';'.

Declaração

Define a propriedade de estilo a aplicar aos elementos identificados pelo selector. Uma declaração é constituída por duas partes, separadas por ':' :

Propriedade – nome do efeito/característica a aplicar.

Valor – Valor a aplicar à propriedade.



CSS external

- Uma *Style Sheet* externa, pode ser associada a uma página HTML através do elemento **<link>**.
- A utilização de *Style Sheets* externas é recomendada quando o mesmo estilo é aplicado a uma grande quantidade de páginas (ex: um *site*).
- Utilizando esta técnica, a alteração do aspecto de um *site* inteiro, consegue-se modificando apenas um único ficheiro.

```
<head>
      <link rel="stylesheet" href="style.css" type="text/css" />
</head>
```

Browser -----> Server

1. Anchor (links) e.g. <a> --- only limited to GET requests
2. Forms --- limited to GET and POST requests
3. Javascript (XMLHttpRequest and fetch)

Forms

- <form>... </form>
  - Attributes:
    - action=url
    - method=(get|post)
    - target=
      _self (default)same browsing
      _blank load into a new unnamed browsing context.
    - enctype=
      application/x-www-form-urlencoded
      multipart/form-data when it includes <input type="file">)