



Clase 11. Programación Backend

Motores de Plantillas (Parte 2)



OBJETIVOS DE LA CLASE

- Conocer el motor de plantillas Pug: sintaxis y uso.
- Integrar Pug a express.
- Conocer el motor de plantillas Ejs: sintaxis y uso.
- Integrar Ejs a express.

CRONOGRAMA DEL CURSO

Clase 10



Motores de Plantillas -
Parte 1

Clase 11



Motores de Plantillas -
Parte 2

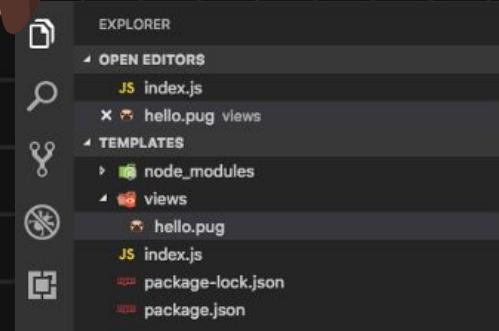
Clase 12



Websockets

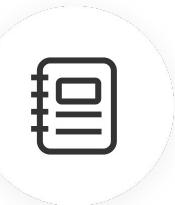
```
JS index.js          hello.pug — templates
1 html
2   head
3     title='Mi primera plantilla Pug JS'
4   body
5     h1=mensaje
```

Pug





¿Qué es Pug?



- Pug JS es un **motor de plantillas** que nos permite **utilizar archivos estáticos** como plantillas, enviar **valores** para **reemplazar variables** dentro de las mismas y **transformar** estos archivos en **páginas HTML** que se envían al cliente.
- Express permite trabajar con muchos motores de plantillas, entre los que se encuentra Pug JS.
- Pug es muy **fácil de implementar**, solo bastará un par de líneas de código para indicarle a express que use Pug JS como motor de plantillas.

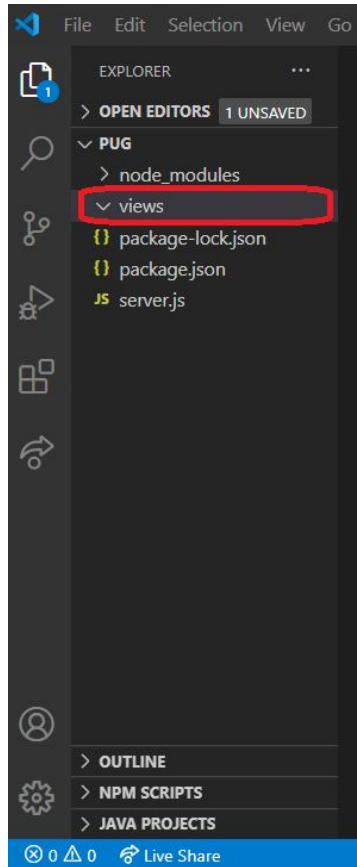


Configuración



- El primer paso es crear un directorio en carpeta en la raíz de nuestro proyecto para guardar las plantillas que se utilizarán en la aplicación. En la imagen (siguiente slide) se puede apreciar el nuevo directorio creado “views” (*panel lateral izquierdo*).
- Ahora necesitamos indicarle a express que “views” será nuestro directorio de plantillas. Y también indicar cuál será el motor de plantillas que se utilizará (en este caso Pug JS). Lo configuramos con:
 - `app.set('views', './views');`
 - `app.set('view engine', 'pug');`

Implementación de pug en express



```
const express = require('express');
const app = express();

app.use('/static', express.static(__dirname + '/public'));
app.use(express.urlencoded({ extended: false }));
app.use(express.json());

// Se indica el directorio donde se almacenarán las plantillas
app.set('views', './views');
// Se indica el motor del plantillas a utilizar
app.set('view engine', 'pug');

app.get('/hello', (req, res) => {
  res.send('Hola mundo');
});

app.get('/urlparam', (req, res) => {
  res.send(req.query);
});

app.post('/urljson', (req, res) => {
  res.send(req.body);
});

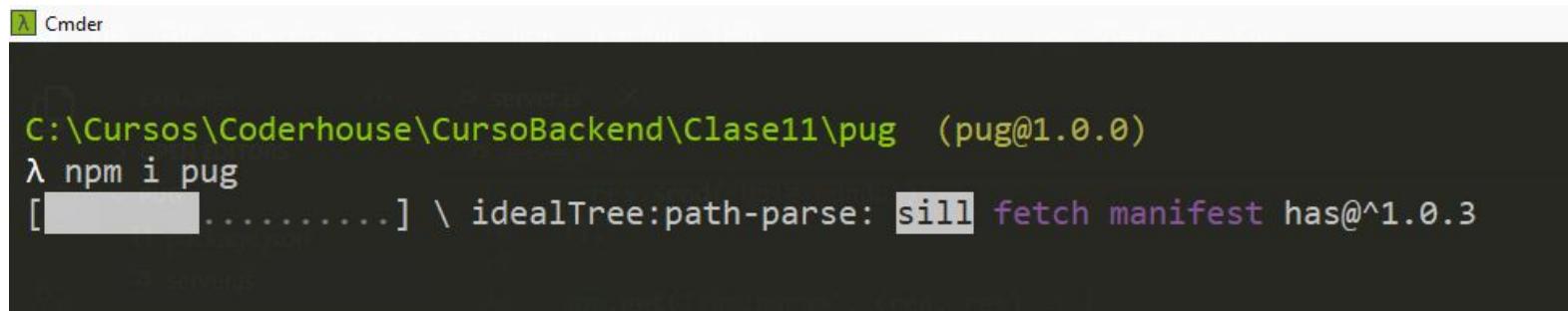
const PORT = 8080;
app.listen(PORT, () => console.log(`Servidor iniciado en el puerto ${PORT}`));
```



Instalar Pug JS

Para esto abriremos la terminal, nos posicionamos en la ruta de nuestra aplicación y lo instalaremos con ayuda de npm con el siguiente comando

```
npm install pug
```

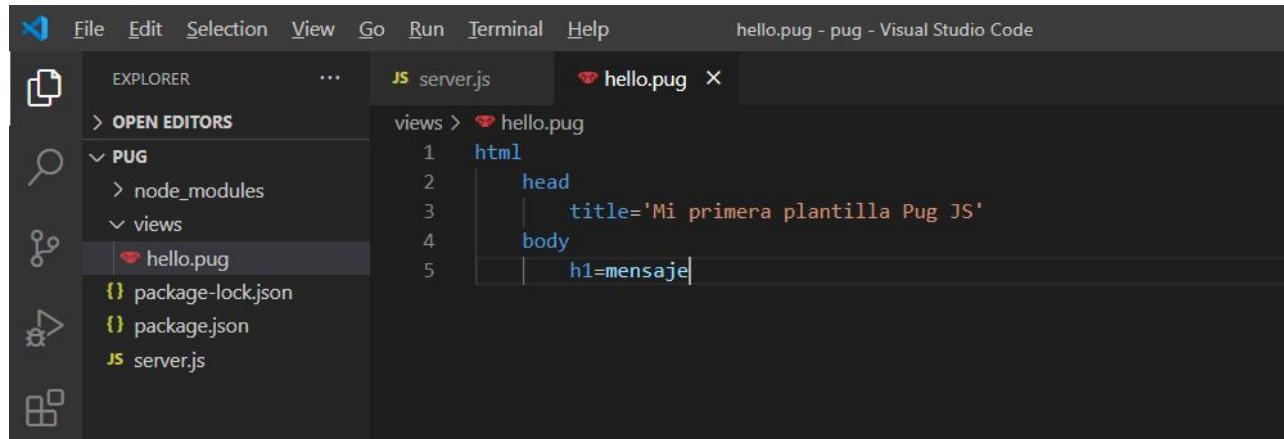


The screenshot shows a terminal window titled 'Cmder'. The command 'npm install pug' is being run from the directory 'C:\Cursos\Coderhouse\CursoBackend\Clase11\pug'. The output shows the command being typed and the progress of the installation process.

```
C:\Cursos\Coderhouse\CursoBackend\Clase11\pug (pug@1.0.0)
λ npm i pug
[redacted] \ idealTree:path-parse: sill fetch manifest has@^1.0.3
```

Crear nuestra primera plantilla

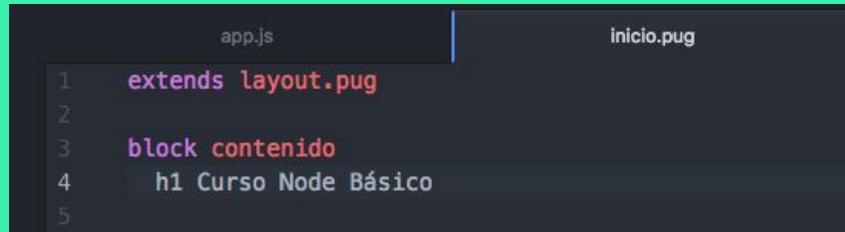
Una vez que se ha configurado e instalado correctamente Pug js solo queda **crear** nuestra **primera plantilla** y mostrarla al cliente. Para ello crearemos el archivo hello.pug (.pug es la extensión de las plantillas) y la mostraremos al ingresar en la url: **localhost:8080/hello**



The screenshot shows the Visual Studio Code interface with a dark theme. The left sidebar has icons for Explorer, Search, Problems, and Terminal. The Explorer view shows a folder named 'PUG' containing 'node_modules' and 'views'. Inside 'views', there is a file named 'hello.pug'. The main editor area shows the contents of 'hello.pug':

```
1 html
2   head
3     title='Mi primera plantilla Pug JS'
4   body
5     h1=mensaje
```

Pug: sintaxis



```
app.js | inicio.pug
1 extends layout.pug
2
3   block contenido
4     h1 Curso Node Básico
5
```

- Pug JS utiliza su **propia sintaxis** para declarar atributos html sin necesidad de abrir y cerrar etiquetas. En cambio se usa la **tabulación** para indicar que una etiqueta pertenece o está dentro de otra.

*Para mayor información visitar el sitio web oficial de pugjs:
<https://pugjs.org/api/getting-started.html>*

- En la línea 5 de **hello.pug** se está declarando una **variable** con nombre “**mensaje**”. Esta será **reemplazada** con el valor que se enviará al momento de transformar de formato .pug a HTML

Siguiente paso

- El siguiente paso será **modificar la ruta “/hello”** que actualmente está mostrando al usuario el mensaje “Hola mundo”. Lo reemplazamos **con la plantilla que creamos**.
- Para ello usaremos la función “**render**” que está disponible en el objeto res (response). Esta función recibe **dos parámetros**: el primero es el **nombre** de la plantilla a mostrar y el segundo un **objeto** con los valores a reemplazar.

```
res.render(view: string, options?: Object)
```

A continuación se puede ver el código final:

File Edit Selection View Go Run Terminal Help

server.js - pug - Visual Studio Code

EXPLORER ... JS server.js X hello.pug

> OPEN EDITORS JS server.js > app.get('/urlparam') callback

✓ PUG 9 app.set('views', './views');

node_modules 10 // Se indica el motor de plantillas a utilizar

views 11 app.set('view engine', 'pug');

hello.pug 12

package-lock.json 13 // Se muestra la plantilla hello.pug

package.json 14 app.get('/hello', (req, res) => {

server.js 15 res.render('hello.pug', { mensaje: 'Usando Pug JS en Express' });

16});

17

18 app.get('/urlparam', (req, res) => [

19 res.send(req.query);

20]);

21

localhost:8080/hello

Usando Pug JS en Express

CODER HOUSE



PUG

Vamos a practicar lo aprendido hasta ahora

Tiempo: 10 minutos



- 1) Realizar un servidor que reciba por query params (mediante la ruta get '/datos') el valor que debe representar una barra de medición (meter).
- 2) Asimismo recibirá además los valores mínimos y máximos permitidos y el título que se pondrá por arriba de la barra, en un elemento h1 en color azul (debe permitir formato HTML).

Ejemplo de request:

```
http://localhost:8080/datos?min=10&nivel=15&max=20&titulo=<i>Medidor</i>
```

- 3) Como respuesta a este request, el servidor devolverá al frontend una plantilla armada con los datos recibidos.
- 4) Utilizar pug integrado a express, manejando una plantilla común y una particular con la representación requerida.



Mi plantilla de medidor en Pug JS X +

← → C ⓘ localhost:8080/datos?min=5&nivel=15&max=20&titulo=<i>Medidor</i>

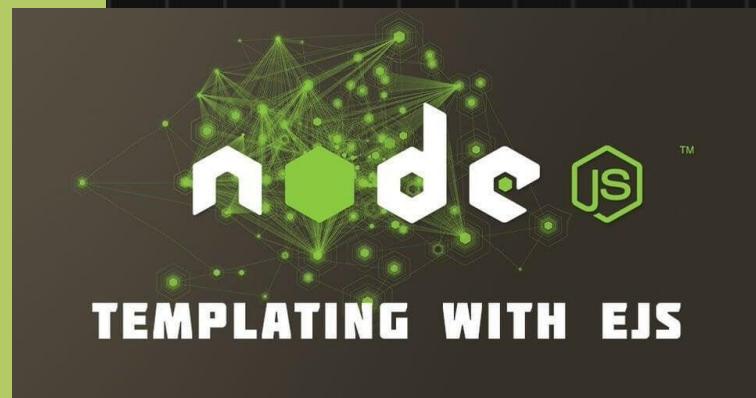
Medidor

5  20

EJS

<%= EJS %>

Effective JavaScript templating.



CODER HOUSE

<%= EJS %>

¿Qué es EJS?



- EJS se encuentra entre los **motores de visualización temáticos más populares para node.js y express** con 5k estrellas en github y más de 8 millones de descargas por semana en npm.
- EJS significa **plantillas de JavaScript incrustadas** y podemos usarlo tanto en el **lado del servidor** como en el **del cliente**. En esta presentación, nos centraremos en el lado del servidor.
- EJS es **fácil de configurar** y podemos incluir las partes repetibles de nuestro sitio (parciales) y pasar los datos a nuestras vistas.



Sintaxis básica (etiquetas)



- **<%** Etiqueta 'Scriptlet' para control de flujo sin salida
- **<%=** Envía el valor a la plantilla (HTML escapado)
- **<%:** Muestra el valor sin escape en la plantilla

Ejemplo

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



Configuración

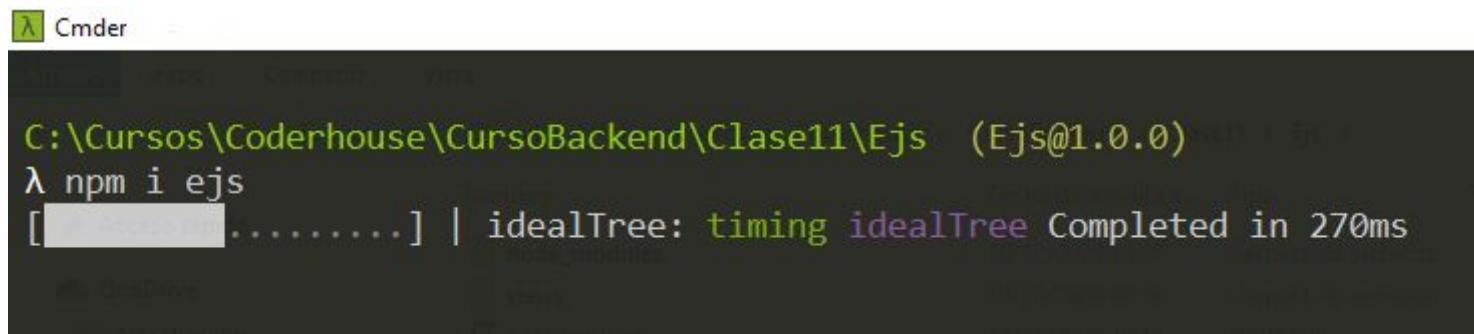


- Configuramos EJS como el motor de visualización de nuestra aplicación Express usando `app.set('view engine', 'ejs')`
- Creamos una carpeta de vistas: **views**
- EJS enviará una vista al usuario usando `res.render()`. Es importante tener en cuenta que `res.render()` buscará la vista en una carpeta **views**.
- Por ejemplo, si definimos `pages/index` dentro de `views`,
`res.render('pages/index')` buscará en `views/pages/index`.

Instalar EJS

Para esto abriremos la terminal, nos posicionamos en la ruta de nuestra aplicación y lo instalaremos con ayuda de npm con el siguiente comando

```
npm install ejs
```



The screenshot shows a terminal window titled 'Cmder'. The command 'npm install ejs' is being run from the directory 'C:\Cursos\Coderhouse\CursoBackend\Clase11\Ejs'. The output shows the progress of the installation, including a progress bar and the message 'idealTree: timing idealTree Completed in 270ms'.

```
C:\Cursos\Coderhouse\CursoBackend\Clase11\Ejs  (Ejs@1.0.0)
λ npm i ejs
[██████████] | idealTree: timing idealTree Completed in 270ms
```

Implementación de EJS en express

```
var express = require('express');
var app = express();

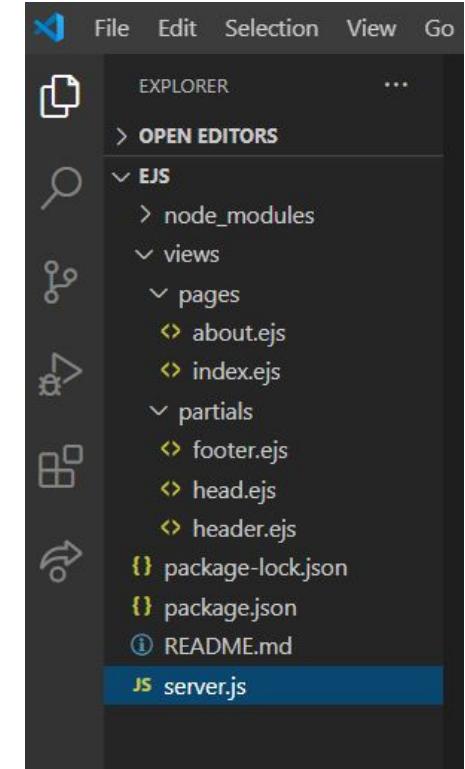
// configuramos EJS como motor de plantillas
app.set('view engine', 'ejs');

// index page
app.get('/', function(req, res) {
  var mascots = [
    { name: 'Sammy', organization: "DigitalOcean", birth_year: 2012},
    { name: 'Tux', organization: "Linux", birth_year: 1996},
    { name: 'Moby Dock', organization: "Docker", birth_year: 2013}
  ];
  var tagline = "No programming concept is complete without a cute animal mascot.";

  res.render('pages/index', {
    mascots: mascots,
    tagline: tagline
  });
});

// about page
app.get('/about', function(req, res) {
  res.render('pages/about');
});

app.listen(8080);
console.log('8080 is the magic port');
```





EJS

Realizar el mismo ejercicio que en el desafío anterior, utilizando ejs.

Tiempo: 10 minutos

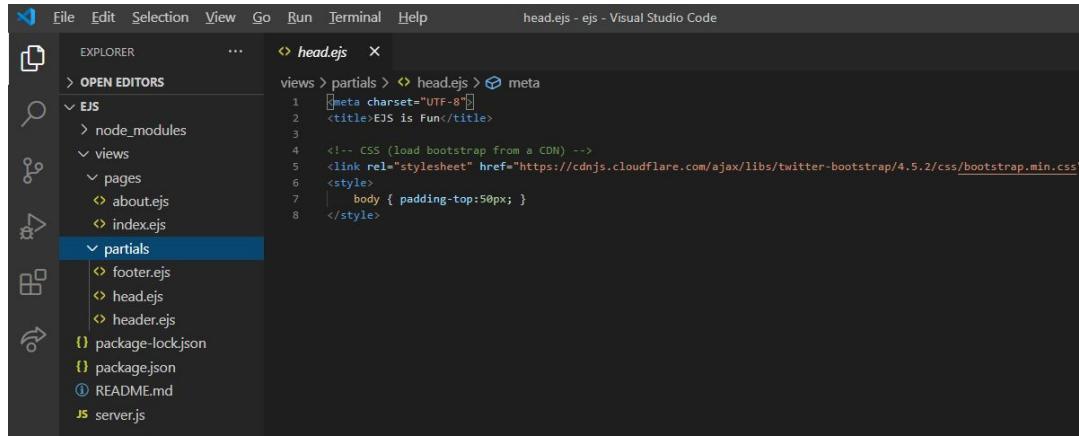


BREAK

¡5/10 MINUTOS Y VOLVEMOS!

Creando nuestras plantillas Parciales

- Al igual que muchas aplicaciones que creamos, hay mucho **código que se reutiliza**. En EJS llamamos a estos códigos **'parciales'**
- En el ejemplo que mostramos a continuación, los definimos dentro de la **carpeta 'parciales'**



The screenshot shows the Visual Studio Code interface with the following details:

- File Explorer:** Shows the project structure:
 - EJS folder
 - views
 - pages
 - about.ejs
 - index.ejs
 - partials folder
 - footer.ejs
 - head.ejs
 - header.ejs
- Editor:** The file "head.ejs" is open, showing its content:

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="UTF-8">
    <title>EJS Is Fun</title>
    <!-- CSS (load bootstrap from a CDN) -->
    <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/twitter-bootstrap/4.5.2/css/bootstrap.min.css">
    <style>
      body { padding-top:50px; }
    </style>
  </head>
  <body>
    <h1>EJS Is Fun</h1>
    <p>This is a simple example of using EJS partials.</p>
  </body>
</html>
```

Ejemplo: Plantillas Parciales

```
views/partials/head.ejs
```

```
<meta charset="UTF-8">
<title>EJS Is Fun</title>

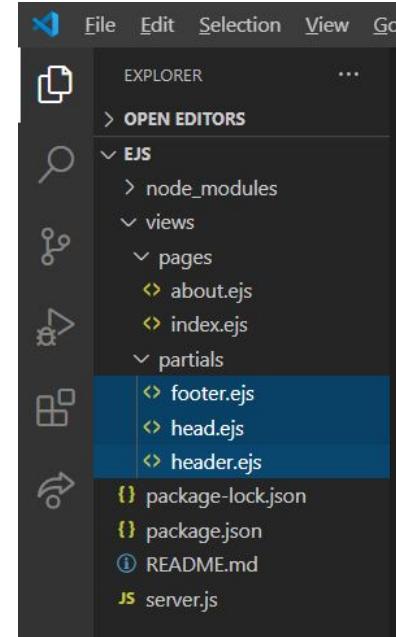
<!-- CSS (load bootstrap from a CDN) --&gt;
&lt;link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/twitter-bootstrap/4.5.2/css/bootstrap.min.css" /&gt;
&lt;style&gt;
    body { padding-top:50px; }
&lt;/style&gt;</pre>
```

```
views/partials/header.ejs
```

```
<nav class="navbar navbar-expand-lg navbar-light bg-light">
  <a class="navbar-brand" href="/">EJS Is Fun</a>
  <ul class="navbar-nav mr-auto">
    <li class="nav-item">
      <a class="nav-link" href="/">Home</a>
    </li>
    <li class="nav-item">
      <a class="nav-link" href="/about">About</a>
    </li>
  </ul>
</nav>
```

```
views/partials/footer.ejs
```

```
<p class="text-center text-muted">© Copyright 2020 The Awesome People</p>
```



Añadiendo los parciales de EJS a Vistas

Ahora tenemos nuestros parciales definidos. Lo único que debemos hacer es **incluirlos en nuestras vistas**.

- Utilizamos `<%- include('RELATIVE/PATH/TO/FILE') %>` para integrar un parcial de EJS en otro archivo.
- El guión `<%-` en lugar de solo `<%` es para indicar a EJS que **renderice HTML sin formato**.
- La ruta al parcial es **relativa** al archivo actual

Ejemplo: Plantillas Parciales en Vista

The image shows a comparison between the source code of a view file and its rendered output in a web browser.

views/pages/index.ejs

```
<!DOCTYPE html>
<html lang="en">
<head>
    <%- include('../partials/head'); %>
</head>
<body class="container">

<header>
    <%- include('../partials/header'); %>
</header>

<main>
    <div class="jumbotron">
        <h1>This is great</h1>
        <p>Welcome to templating using EJS</p>
    </div>
</main>

<footer>
    <%- include('../partials/footer'); %>
</footer>

</body>
</html>
```

EJS is Fun | localhost:8080

This is great

Welcome to templating using EJS

© Copyright 2020 The Awesome People

Pasando datos a Vistas y Parciales

Vamos a definir algunas variables básicas y una lista para pasar a nuestra página de inicio. Volvamos al archivo server.js e incorporemos lo siguiente dentro de la ruta **app.get('/')**

server.js

```
// index page
app.get('/', function(req, res) {
  var mascots = [
    { name: 'Sammy', organization: "DigitalOcean", birth_year: 2012},
    { name: 'Tux', organization: "Linux", birth_year: 1996},
    { name: 'Moby Dock', organization: "Docker", birth_year: 2013}
  ];
  var tagline = "No programming concept is complete without a cute animal mascot.";

  res.render('pages/index', {
    mascots: mascots,
    tagline: tagline
  });
});
```

EJS: Incorporando Datos en Vistas

Ejemplo: datos en Vistas

Renderizar una variable única en EJS

- Para hacer eco de una variable, acabamos usamos **<%= tagline %>**

```
views/pages/index.ejs
```

```
...
<h2>Variable</h2>
<p><%= tagline %></p>
...
```

Ejemplo: datos en Vistas

Hacer bucle sobre los datos en EJS

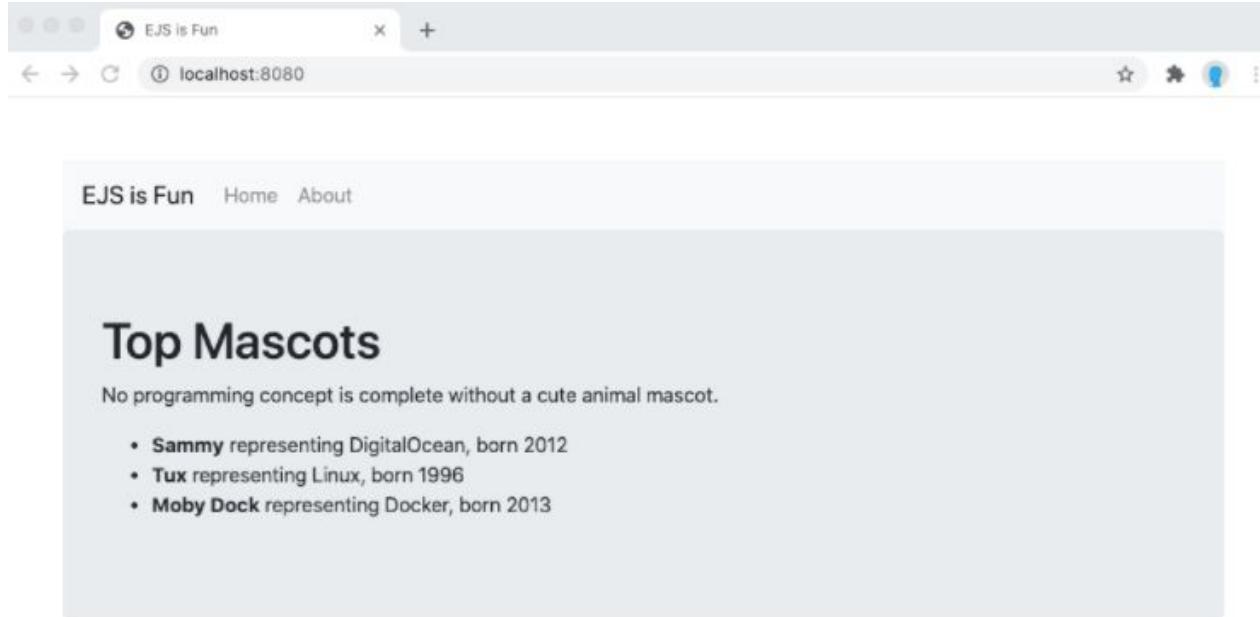
- Para hacer bucle sobre nuestros datos usaremos `.forEach`

views/pages/index.ejs

```
...
<ul>
  <% mascots.forEach(function(mascot) { %>
    <li>
      <strong><%= mascot.name %></strong>
      representing <%= mascot.organization %>, born <%= mascot.birth_year %>
    </li>
  <% }); %>
</ul>
...
```

Ejemplo: datos en Vistas

Podemos ver en nuestro navegador la información que hemos añadido.



EJS: Incorporando Datos en Parciales

Pasando datos a un parcial en EJS

El parcial EJS tiene acceso a todos los datos que la vista principal.

Aclaración: si hacemos referencia a una variable en un parcial, debe definirse en cada vista que utilice el parcial o arrojará un error.

- También podemos definir y pasar variables a un parcial EJS en la **sintaxis include** de esta forma

views/pages/about.ejs

```
...
<header>
  <%- include('../partials/header', {variant:'compact'}); %>
</header>
...
```

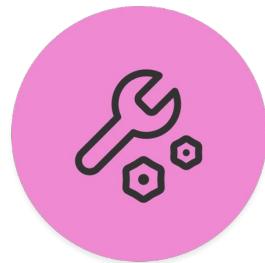
Pasando datos a un parcial en EJS

Si deseamos referenciar una variable en un parcial que puede no definirse siempre y darle un valor predeterminado, podemos hacerlo de esta forma:

views/partials/header.ejs

```
...
<em>Variant: <%= typeof variant != 'undefined' ? variant : 'default' %></em>
...
```

*En la línea anterior, el código EJS se renderiza el valor de **variant** si está definido y de **default** si no lo está.*



FORMULARIO + HISTORIAL

Tiempo: 15 minutos



- 1) Desarrollar un servidor basado en node.js, express y ejs que disponga de un **formulario** en su ruta raíz (creado con una plantilla de ejs) para ingresar los siguientes datos de una persona: nombre, apellido y edad.
- 2) La información será enviada mediante el **método post** al endpoint '/datos'
- 3) Representar por debajo del mismo formulario los **datos históricos** ingresados más el actual en forma de tabla. En el caso de no encontrarse información mostrar el mensaje '**'No se encontraron datos'** en lugar de la tabla.

Se recomienda el uso de bootstrap en los estilos de las plantillas.



Mi plantilla de Ingreso de Datos x +

localhost:8080

Ingrese datos

Nombre

Apellido

Edad

Enviar

Historial

Nombre	Apellido	Edad
Daniel	Sánchez	52
Ana	Mei	23
Diego	Suarez	34

Mi plantilla de Ingreso de Datos x +

localhost:8080

Ingrese datos

Nombre

Apellido

Edad

Enviar

Historial

no se encontraron datos



PUG y EJS

PUG y EJS

Formato: link a un repositorio en Github con el proyecto cargado.

Sugerencia: no incluir los node_modules

Desafío
entregable



>> Consigna: Sobre el proyecto entregable de la clase anterior, realizar las siguientes adaptaciones

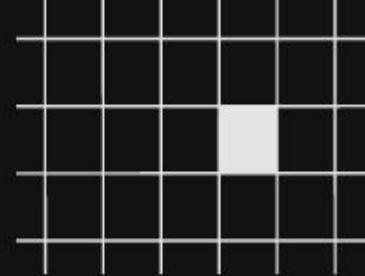
- 1) Manteniendo la misma funcionalidad reemplazar el motor de plantillas handlebars por **pug**.
- 2) Manteniendo la misma funcionalidad reemplazar el motor de plantillas handlebars por **ejs**.
- 3) Por escrito, indicar cuál de los tres motores de plantillas prefieres para tu proyecto. Justificar tu respuesta.

>> Aspectos a incluir en el entregable:

- Puedes utilizar branches para las consignas 1) y 2) (Optativo) o simplemente hacer dos commits.
- La justificación puede ir escrita al subir la entrega. Si utilizaste branches, puedes enviar al máster la opción seleccionada.

*¿*PREGUNTAS?

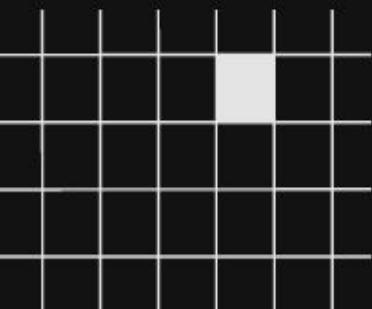




¡MUCHAS GRACIAS!

Resumen de lo visto en clase hoy:

- Motor de plantillas Pug
- Motor de plantillas EJS





OPINA Y VALORA ESTA CLASE

#DEMOCRATIZANDOLAEDUCACIÓN