Introducción a Node

Devolviendo un sitio web de contenido dinámico

{desafío} latam_

Implementar un servidor web de contenidos estáticos y dinámicos utilizando motores de plantillas acorde al entorno Node Express para dar solución a un problema.

- Unidad 1:
  Introducción a Node
- Unidad 2:
  Node y el gestor de paquetes
- Unidad 3:
  Persistencia

Te encuentras aquí

{desafío}
latam_

¿Qué aprenderás en esta sesión?

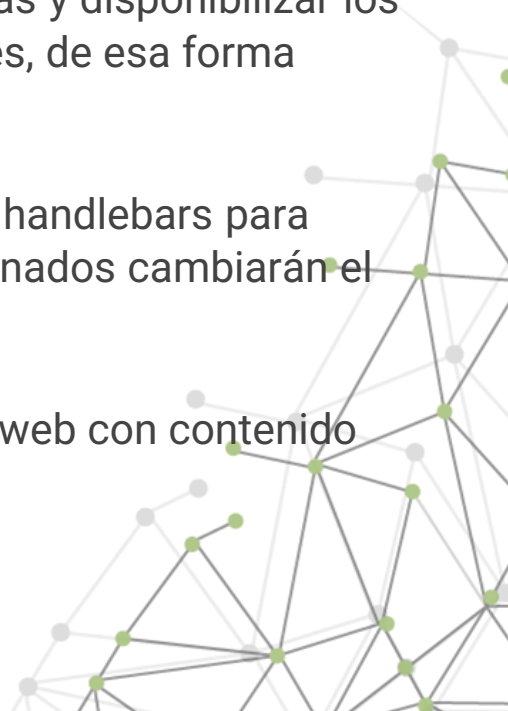
- *Construir un servidor utilizando Express para servir un sitio web estático con Bootstrap y jQuery integrados.*

Introducción

En este capítulo aprenderás a instalar y utilizar jQuery y Bootstrap de forma local a través de NPM y los middlewares de Express para definir rutas personalizadas y disponibilizar los códigos fuentes que se encuentran dentro de la carpeta node_modules, de esa forma importarlos en el index.html que devolvemos en la ruta raíz.

Se realizará un ejercicio en el que se creará un servidor con express y handlebars para mostrar una vista con 8 botones de bootstrap, los cuales al ser presionados cambiarán el color de un spinner de forma dinámica.

Al finalizar este ejercicio estarás listo para empezar a construir sitios web con contenido dinámico utilizando los parciales y helpers de Handlebars.



{desafío} latam_

/* Integrando jQuery y Bootstrap*/

Importación de jQuery y Bootstrap

En el desarrollo a nivel de cliente, la importación de jQuery y Bootstrap se maneja por cdn o descargando su código fuente. Para el caso del backend, tenemos a disposición los siguientes comandos:

jQuery

```
npm install --save
jquery
```

Bootstrap

```
npm install --save bootstrap
```

¿Cuál es la diferencia de instalarlos de esta manera y no importarlos por cdn?

Demostración "Importando Bootstrap y jQuery desde node_modules"

Ejercicio guiado

Importando Bootstrap y jQuery desde node_modules

Servir un sitio web estático que importe jQuery y Bootstrap, ambos instalados con NPM. El objetivo de este ejercicio será aprender a usar los middlewares y el método “static” para definir 2 rutas que permita importar los códigos fuente de nuestras dependencias desde la carpeta node_modules.

Ejercicio guiado

Importando Bootstrap y jQuery desde node_modules

Consideraciones previas

- Ambas tecnologías deben ser importadas desde un Index.html con las etiquetas correspondientes, sus rutas (atributos href y src) y deben apuntar a sus códigos fuentes.

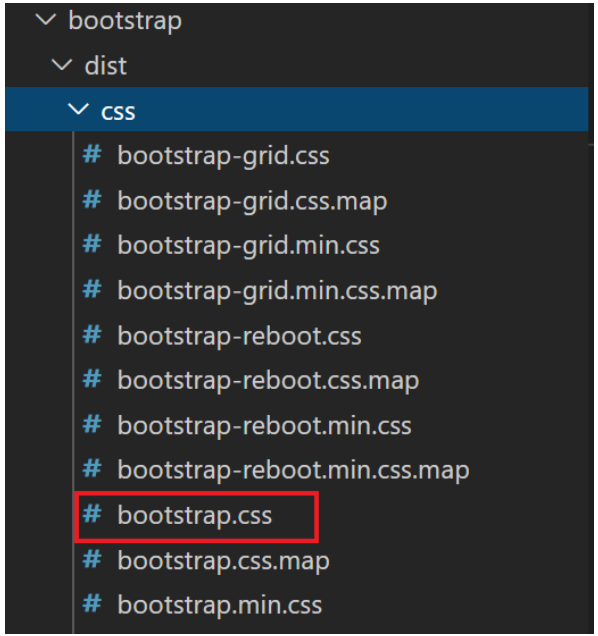
- **¿Cómo importar un archivo ubicado dentro de los node_modules a mi index.html?** Esto no será necesario ni posible, recordemos que los servidores con Express, disponibilizan contenido a través de rutas o directorios "liberados" con el método "static" de la instancia de Express.

- **¿Cómo haré para importar jQuery y Bootstrap desde mi sitio web HTML devuelto por Express, teniendo sus códigos fuentes dentro de una carpeta node_modules?** La respuesta y la solución la encontramos con los "middlewares", puesto que podremos especificar que una ruta "X" será la referencia para acceder a un archivo o carpeta dentro del profundo árbol de archivos de node_modules.

Ejercicio guiado

Importando Bootstrap y jQuery desde node_modules

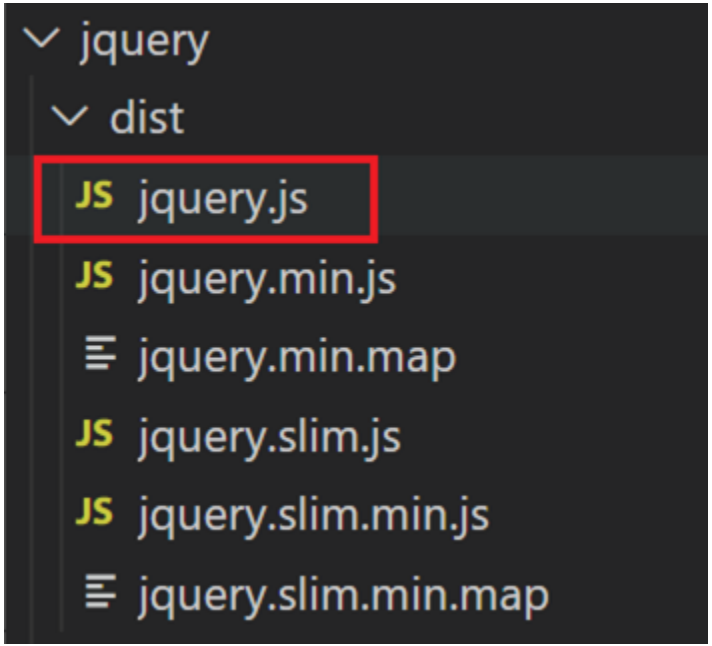
Debemos estar conscientes de la ubicación de estos archivos fuentes en esta carpeta, para eso despliega los node_modules y busca las carpetas correspondientes a jQuery y Bootstrap, donde encontrarás que ambos tienen una subcarpeta llamada “dist” y dentro de esta, se encuentran los tan queridos archivos fuentes.



Ejercicio guiado

Importando Bootstrap y jQuery desde node_modules

Ahora que sabemos dónde está nuestro framework de CSS, en la siguiente imagen veremos la ubicación del jquery.js:



Ejercicio guiado

Importando Bootstrap y jQuery desde node_modules

Teniendo clara de la ubicación de ambos archivos, es necesario seguir los siguientes pasos para resolver este ejercicio:

- **Paso 1:** Crear un servidor con Express que escuche el puerto 3000.
- **Paso 2:** Crear un middleware que define una ruta /bootstrap y libere el contenido de la carpeta “css”, de la dependencia de Bootstrap en el node_modules.
- **Paso 3:** Crear un middleware que define una ruta /jquery y libere el contenido de la carpeta “dist”, de la dependencia de jQuery en el node_modules.
- **Paso 4:** Crear una ruta GET / que devuelva un archivo index.html.

```
// Paso 1
const express = require("express");
const app = express();
app.listen(3000, () => {
  console.log("El servidor está inicializado en el puerto 3000");
});

// Paso 2
app.use('/bootstrap', express.static(__dirname +
'/node_modules/bootstrap/dist/css'))

// Paso 3
app.use('/jquery', express.static(__dirname + '/node_modules/jquery/dist'))

// Paso 4
app.get("/", (req, res) => {
  res.sendFile(__dirname + '/index.html')
})
```

Ejercicio guiado

Importando Bootstrap y jQuery desde node_modules

Con la lógica escrita en el servidor, solo falta probar la efectividad de nuestros middlewares, por ende necesitaremos crear nuestro index.html y escribir dentro de este algún componente de Bootstrap, junto con la instrucción con jQuery

- **Paso 1:** Importar Bootstrap.css con una etiqueta “link”.
- **Paso 2:** Importar un Jumbotron para demostrar que estamos importando correctamente Bootstrap. Te dejo el link oficial para que tomes este componente
- **Paso 3:** Importar jQuery.js usando la etiqueta “script”.
- **Paso 4:** Ocupa el document ready de jQuery para cambiarle el color de fondo al Jumbotron de Bootstrap a negro y cambiarle el color de texto a blanco.

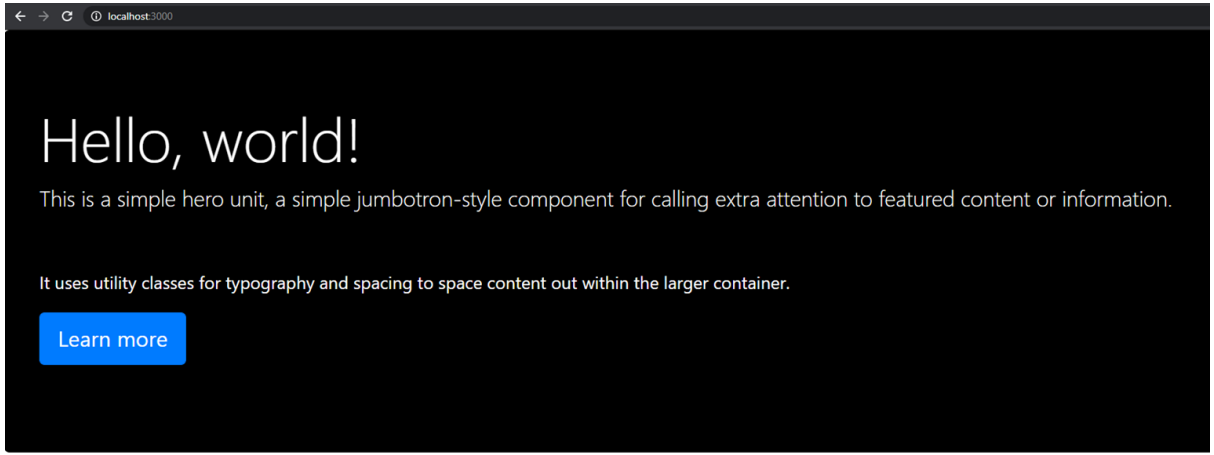
```
<!-- Paso 1 -->
<link rel="stylesheet" href="/bootstrap/bootstrap.css" />
<!-- Paso 2 -->
<div class="jumbotron">
  <h1 class="display-4">Hello, world!</h1>
  <p class="lead">
    This is a simple hero unit, a simple jumbotron-style component for calling
    extra attention to featured content or information.
  </p>
  <hr class="my-4" />
  <p>
    It uses utility classes for typography and spacing to space content out
    within the larger container.
  </p>
  <p class="lead">
    <a class="btn btn-primary btn-lg" href="#" role="button">Learn more</a>
  </p>
</div>

<!-- Paso 3 -->
<script src="/jquery/jquery.js"></script>
<!-- Paso 4 -->
<script>
  $(document).ready(() => {
    $(".jumbotron").css("background", "black").css("color", "white");
  });
</script>
```

Ejercicio guiado

Importando Bootstrap y jQuery desde node_modules

- Visita la ruta raíz de tu servidor (http://localhost:3000/) con tu navegador preferido y deberás recibir lo que se muestra en la siguiente imagen:



/* Devolviendo un sitio web de contenido dinámico */

Demostración "Spinner de colores"

Ejercicio guiado

Spinner de colores

Desarrollar un servidor que reciba en su ruta raíz un parámetro con el nombre de un color según las variantes de colores de bootstrap. Este parámetro será enviado a través del render a una vista “Inicio”, la cual importará 2 parciales:

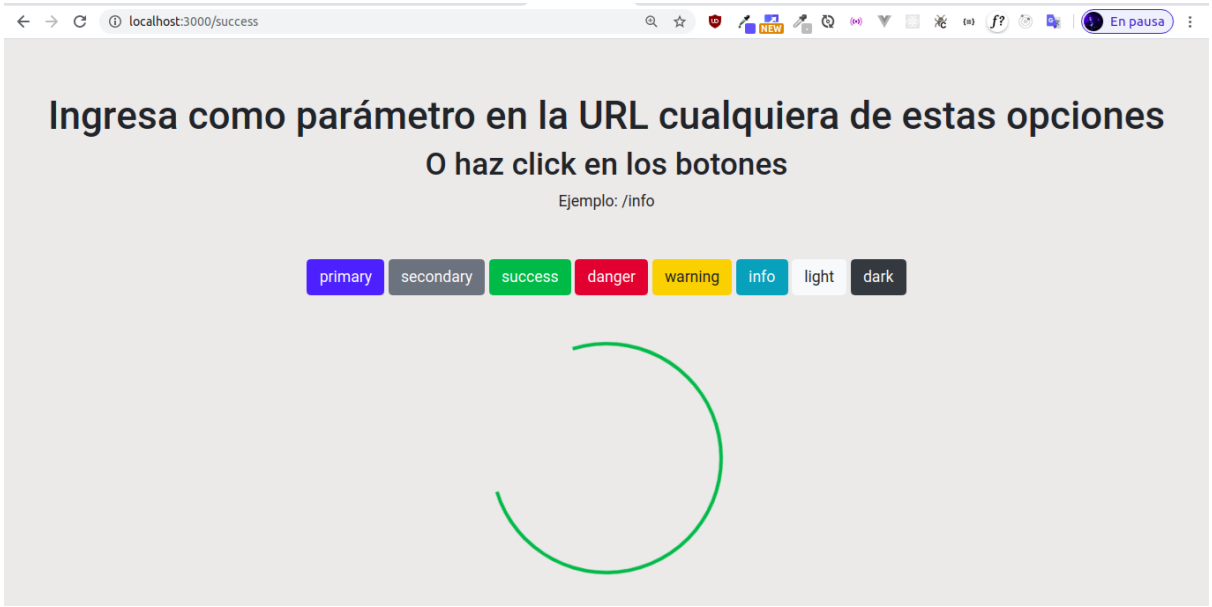
- **Botones**: Contendrá el helper “each” para iterar un arreglo de colores.
- **Spinner:** Contendrá el componente spinner de Bootstrap, en donde su variante de color será igual a un parámetro “color” enviado desde el render.

El objetivo será programar que al ser presionado un botón de color “success”, por ejemplo, se consulte al servidor pasando como parámetro de la URL esta variante de color y ¿Cómo redireccionaremos al usuario luego del click en el botón? Usaremos las etiquetas de ancla, concatenando de forma dinámica el color en el atributo “href”.

Ejercicio guiado

Spinner de colores

La siguiente imagen es una referencia de lo que se debe lograr al finalizar este ejercicio:



{desafío} latam_

Ejercicio guiado

Spinner de colores: Parte I - Plantillas

- **Paso 1:** Crear una plantilla dentro de la carpeta “views” llamado “Inicio.handlebars” que contenga el siguiente código:

- En esta plantilla lo que se hace es:
  - Importar Bootstrap.
  - Diseñar la maqueta con código HTML y clases de Bootstrap.
  - Importar 2 parciales: Botones y Spinner.

```
<link rel="stylesheet" href="/css/bootstrap.min.css">

<div class="contain d-flex flex-column text-center justify-content-center align-items-center">
  <h1>Ingresa como parámetro en la URL cualquiera de estas opciones</h1>
  <h2>O haz clic en los botones</h2>
  <span>Ejemplo: /info</span>
  <div class="m-5">
    {{> Botones }}
  </div>

  <div>
    {{> Spinner}}
  </div>
</div>
<style>
  * {
    margin: 0;
  }

  .contain {
    background: #ece9e9;
    height: 100vh;
  }
</style>
```

Ejercicio guiado

Spinner de colores: Parte I - Plantillas

- **Paso 2:** Crear una plantilla dentro de la carpeta “componentes” llamada “Botones” y escribe el siguiente código.

```
{{#each colores}}
<a href="/{{this}}"> <button class="btn
btn-{{this}}">{{this}}</button> </a>
{{/each}}
```

En esta plantilla estamos iterando un arreglo llamado “colores” que será pasado como parámetro dentro del método render, es importante que sepas que aunque este parámetro se está pasando a la vista “Inicio”, todos los parciales hijos también podrán acceder a este dato.

Ejercicio guiado

Spinner de colores: Parte I - Plantillas

- **Paso 3:** Crear una plantilla dentro de la carpeta “componentes” llamada “Spinner” y escribe dentro el siguiente código:

```
<div class="spinner">
  <div class="spinner-border text-{{color}}"
style="width: 15rem; height: 15rem;">
  </div>
</div>
```

Esta plantilla estará mostrando el componente spinner de Bootstrap, pero la variante de color será un parámetro “color” enviado desde el render en el servidor.

Ejercicio guiado

Spinner de colores: Parte II - Servidor

- **Paso 1:** Crear un servidor con Express e integra handlebars definiendo en su configuración las rutas para el consumo de vistas y componentes.
- **Paso 2:** Crear un middleware que publique la carpeta “css” de Bootstrap localizada dentro de los node_modules.
- **Paso 3:** Crear una ruta raíz que reciba un parámetro “color”.
- **Paso 4:** Utilizar destructuring para extraer la propiedad color de los parámetros de la consulta con el objeto request.
- **Paso 5:** Ocupar el método render para renderizar la plantilla Inicio y pasar como parámetros un arreglo con todas las variantes de colores de Bootstrap y una propiedad “color” cuyo valor será el recibido como parámetro en la ruta.

```
// Paso 1
const express = require("express");
const app = express();
const exphbs = require("express-handlebars");

app.listen(3000, () => {
  console.log("El servidor está inicializado en el puerto 3000");
});

app.engine(
  "handlebars",
  exphbs({
    layoutsDir: __dirname + "/views",
    partialsDir: __dirname + "/views/componentes/",
  })
);

app.set("view engine", "handlebars");

// Paso 2
app.use("/css", express.static(__dirname + "/node_modules/bootstrap/dist/css"));

// Paso 3
app.get("/:color", function (req, res) {
  // Paso 4
  const { color } = req.params;
  // Paso 5
  res.render("Inicio", {
    layout: "Inicio",
    colores: [
      "primary",
      "secondary",
      "success",
      "danger",
      "warning",
      "info",
      "light",
      "dark",
    ],
    color: color,
  });
});
```

{desafío} latam_

Ejercicio guiado

Spinner de colores: Parte II - Servidor

- Ha llegado el momento de la verdad, levanta el servidor y consulta la siguiente ruta: http://localhost:3000/danger, deberás ver lo que se muestra en la siguiente imagen:

{desafío} latam_

Ejercicio guiado

Spinner de colores: Parte II - Servidor

Muy bien, se está renderizando todo sin problema, sin embargo, falta probar lo más importante, que es el cambio dinámico del color del spinner, para esto haz click en el botón “primary” y deberás recibir lo que se muestra en la siguiente imagen:

¿Hay algún concepto o paso con el que tengas dudas?

Próxima sesión...

- *Desafío evaluado - Mercado web*

{desafío}
latam_

{desafío}
latam_

Academia de
talentos digitales