DigitalHouse >
Coding School

PLANTILLAS
PERSONALIZADAS

“

Existen maneras de **modularizar** nuestra **estructura** html logrando así simplificar mucho el **desarrollo** y **ensamblado** final de nuestro proyecto.

CÓMO MODULARIZAR

Lo primero a tener en cuenta es detectar aquellas porciones de código que vemos repetidas en todos los archivos del proyecto.

En una estructura corriente de html podemos encontrar que todos los archivos fácilmente van a contar con: un head, una barra de navegación y un footer, *por mencionar algunas*.

Cada una de estas partes, se convertirá en un **archivo nuevo** que contendrá **únicamente** esa **porción** de código.

Si estamos trabajando con un **template engine** deberemos almacenar estos archivos dentro de la carpeta **views** *-o la que hayamos configurado por defecto-* para no cortar el flujo de trabajo.

PARTIENDO EL CÓDIGO

Tendremos que **cortar** el código que queramos modularizar y **pegarlo** en un **nuevo archivo** al cual le asignaremos un nombre que represente la parte que contiene, *aclarando siempre la extensión del motor de plantillas que usemos*, por ejemplo: `head.ejs`.

html

```
<head>
      <meta charset="UTF-8">
      <meta name="viewport" content="width=device-width, initial-scale=1.0">
      <link rel="stylesheet" href="/css/master.css">
      <title>Mi web</title>
</head>
```

Como buena práctica se sugiere almacenar estos archivos dentro de una carpeta llamada **partials.**

REUTILIZANDO EL CÓDIGO

Para disponer del código que modularizamos, necesitamos hacer uso de la función `include()` que nos provee *ejs*. La misma recibe como parámetro un **string** que será la **ruta** hacia el **archivo** que queremos incluir.

html
```
include('./partials/head')
```

Por último, deberemos escribir la etiqueta **<%-**, que imprimirá en el documento el contenido exacto que retorne el `include()` .

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

¿POR QUÉ MODULARIZAR?

Porque nos permite **ordenar** nuestro código, ser menos **repetitivos**, y conseguir un mejor y más simple **mantenimiento** del código.