

DIPLOMADO

Front End

INTRODUCCIÓN

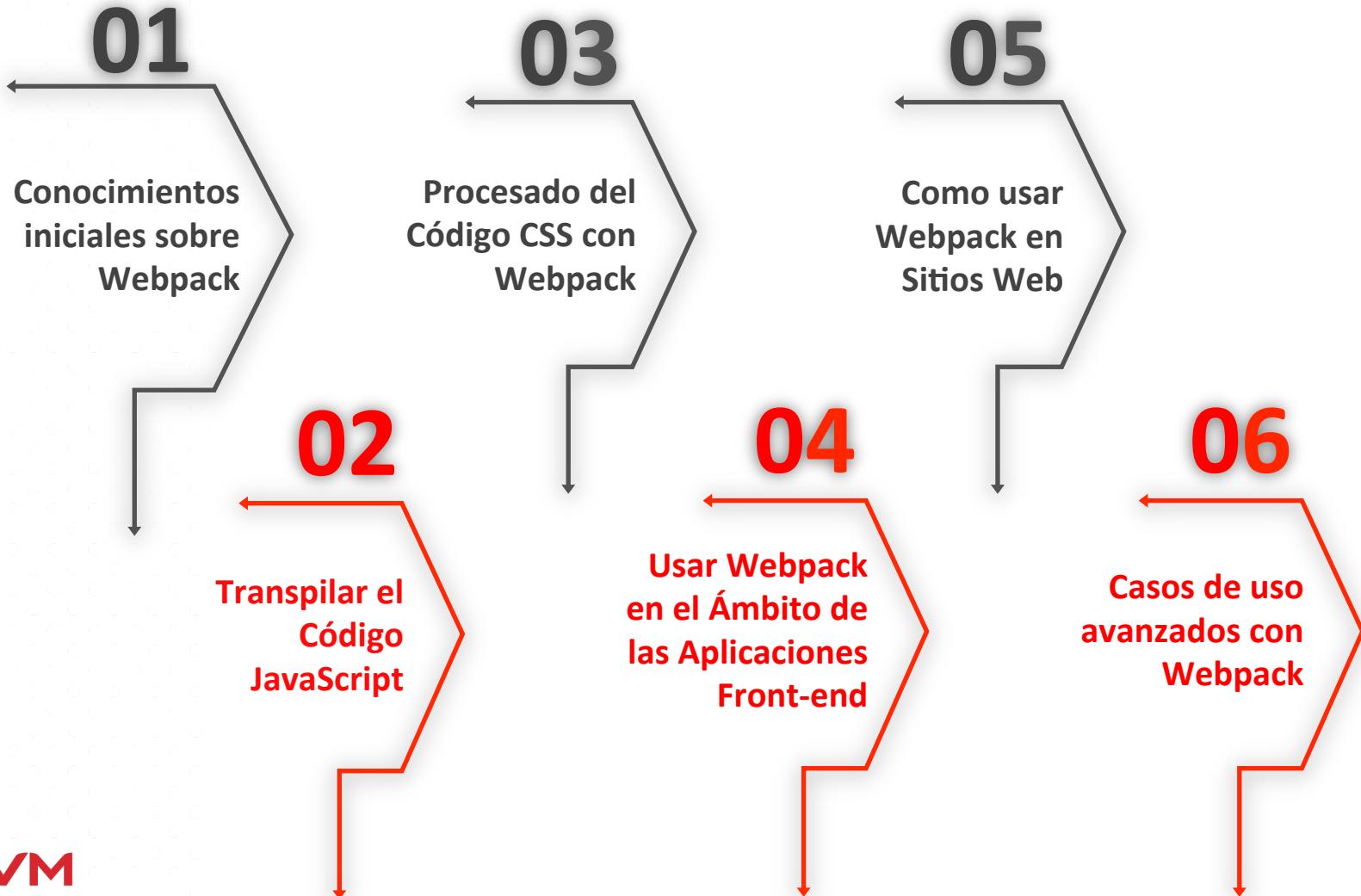
Build Tools,
Webpack, Service Workers

Objetivo

Desarrollarás una comprensión de cómo usar las herramientas de empaquetamiento, como **Webpack**, para automatizar las tareas de compilación. Crearás variables **CSS** y configurarás **Webpack** para utilizar hojas de estilo. Finalmente, aprenderás a utilizar **Webpack** para el desarrollo de proyectos y casos de uso avanzados.



Temario



DIPLOMADO

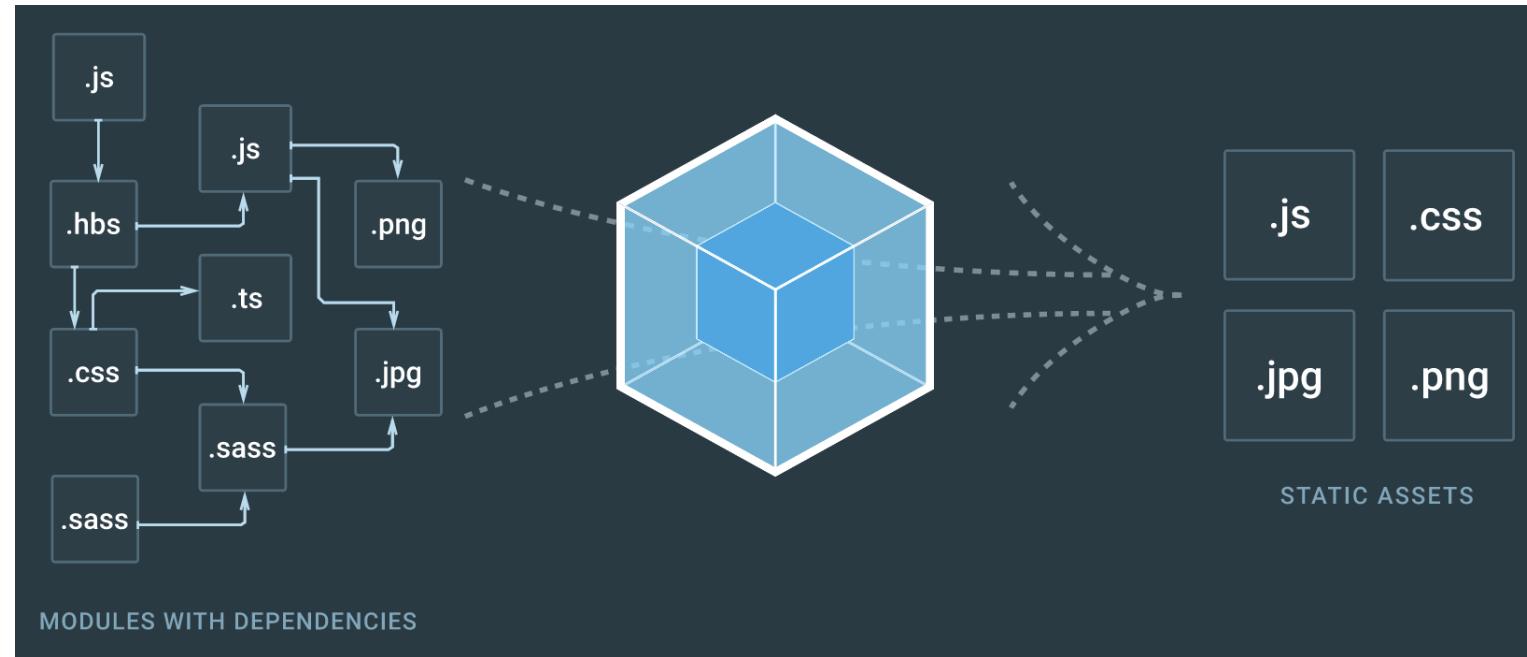
Front End

Conocimientos
Iniciales sobre Webpack

TEMA 1

Webpack

Webpack es un empaquetador de módulos estáticos para aplicaciones modernas en Javascript.



Al procesar una aplicación, Webpack crea internamente un grafo de dependencia a partir de uno o más puntos de entrada y luego combina cada módulo que tu proyecto necesita en uno o más paquetes, que se convierten en activos estáticos consumibles por la aplicación

Entrada (entry)

Un punto de entrada le indica a Webpack que módulo debería utilizar para empezar a construir el grafo de dependencias.

Por defecto el punto de entrada esta en:

```
module.exports = {  
  entry: './path/to/my/entry/file.js',  
};
```

Webpack resolverá cuales otros módulos y bibliotecas necesita el punto de entrada directa o indirectamente para funcionar.

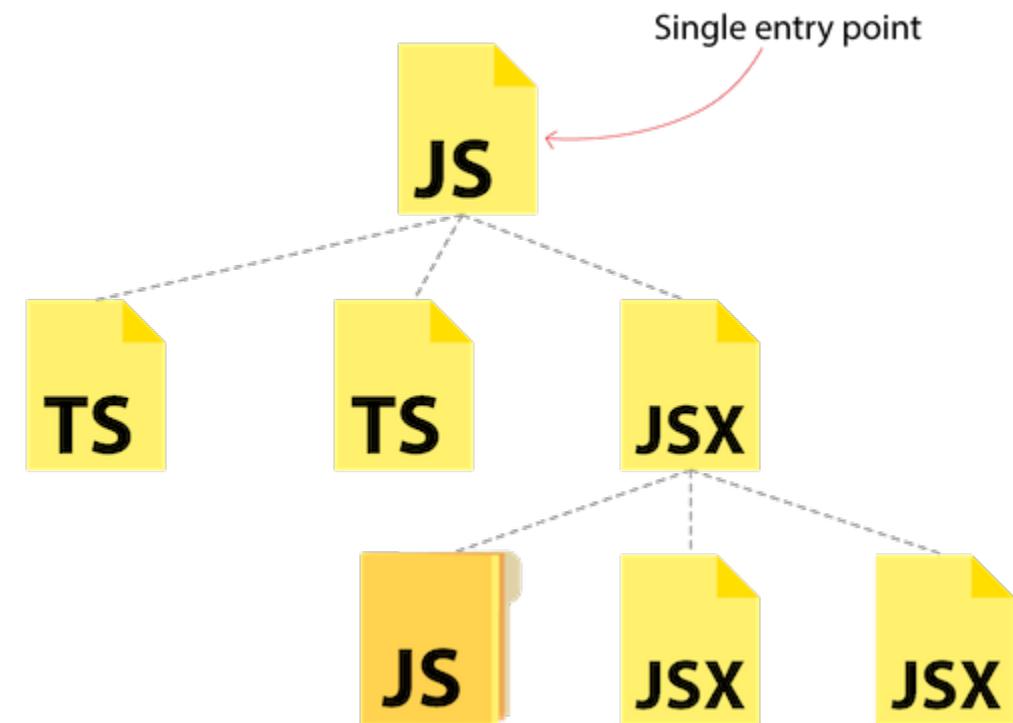
Salida (output)

La salida le indica a Webpack donde emitir los paquetes que generará y como nombrarlos. Por defecto el archivo de salida es ./dist/main.js y ./dist la carpeta para cualquier otro archivo generado.

```
const path = require('path');
module.exports = {
  entry: './src/file.js',
  output: {
    path: path.resolve(__dirname, 'dist'),
    filename: 'main.bundle.js',
  },
};
```

Grafo de Dependencia

Es una representación de como los archivos y dependencias son ordenados y enlazados dentro de una aplicación, además muestra como los archivos interactúan entre sí.



Otros Conceptos Importantes 1

- **Activos (assets):** Se refiere principalmente a los archivos que son imágenes, incluyendo también archivos de datos y scripts recopilados por Webpack al producir la aplicación empaquetada.
- **Paquete (bundle):** Es la aplicación generada como salida de Webpack una vez que compila la aplicación de entrada. Cabe mencionar que es una versión optimizada de la aplicación de entrada.
- **Transpilar:** Es el proceso donde Webpack toma un conjunto de código fuente de entrada y lo cambia a un código de distribución de salida más optimizado quitando código no utilizado y/o duplicado.

Otros Conceptos Importantes 2

- **CSS:** Las hojas de estilo en cascada o Cascading Style Sheets describen la presentación de documentos HTML especificando como deben ser renderizados los elementos al presentarse en pantalla.
- **SASS:** Syntactically Awesome Style Sheet es un preprocesador de código que agrega características mejoradas a CSS.
- **SCSS:** Sassy CSS es un preprocesador de código que agrega características mejoradas a CSS y tiene una sintaxis más parecida.
- **CDN:** Es una red de distribución de contenido o Content Delivery Network que proporciona alta disponibilidad de recursos y alto rendimiento.

¿Por qué Webpack?

- Genera código optimizado que ayuda a mejorar el desempeño de los navegadores
- Elimina problemas como código sin utilizar, código repetido y la necesidad de reescribir código.
- En su versión más reciente necesita una mínima configuración.
- Tiene opciones de configuración flexibles y adaptables a un gran número de casos de uso.
- Permite utilizar plugins para extender su funcionamiento.

DIPLOMADO

Front End

```
<div style="background-image:url('/pix/samples/bg1.gif');background . text-todoitem ;  
height . text : 200px; ><p>The image can be tiled across the background,  
while the text runs across the top.</p> </div>  
  
/* Logo */  
<body style="background-color:yellowgreen;color:white;">  
<html> <.todolistid = data.todoIdb;  
  
/* Header */  
<header { background:#eee; }  
#header-inner { margin:0 auto; padding:10px; width:970px;background:#fff; }  
  
/* Feature */  
<div style="background-image:url('/pix/samples/bg1.gif');background . text-todoitem ;  
height . text : 200px;"><p>The image can be tiled across the background,  
while the text runs across the top.</p> </div>  
  
/* Menu */  
<p>You can make <span style="font-style:italic">some</span> the HTML 'span' tag.  
<p>You can bold <span style="">parts</span> of your text using the HTML tag.</p>  
<div style="background-image:url('/pix/samples/bg1.gif');background . text-todoitem ;  
height . text : 200px;"><p>The image can be tiled across the background,  
while the text runs across the top.</p> </div>  
<body style="background-color:yellowgreen;color:white;">  
<p>You can make <span style="font-style:italic">some</span> the HTML 'span' tag.  
<p>You can bold <span style="">parts</span> of your text using the HTML tag.</p>  
<div style="background-image:url('/pix/samples/bg1.gif');background . text-todoitem ;  
height . text : 200px;"><p>The image can be tiled across the background,  
while the text runs across the top.</p> </div>  
#content #sidebar .widget ul { margin:0; padding:0; list-style:none; color:#999999; }  
#content #sidebar .widget ul li { margin:0; }  
#content #sidebar .widget ul li { padding:4px 0; width:185px; }  
#content #sidebar .widget ul li a { color:blue; text-decoration:none; margin-left:-16px; padding:4px 0px 4px 16px; }  
while the text runs across the top.</p> </div>  
  
/* Footer */  
<footer { background:#eee; }  
#footer-inner { margin:auto; text-align:center; padding:12px; width:970px; }  
<p style="font-weight:bold;"><p>code is done using CSS.</p>  
<p> <.todolistid = data.todoIdb;
```

UVM

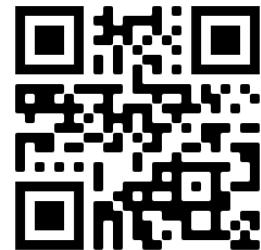
TEMA 2

Transpilar el
Código JavaScript

Herramientas Iniciales

Para que puedas utilizar Webpack es necesario que tengas instaladas las siguiente dos herramientas y verificar que estén funcionando desde la línea de comandos (CMD o PowerShell en Windows o Terminal en Mac y Linux):

- NodeJS (versión LTS recomendada <https://nodejs.org/en/>)
- NPM (se incluye dentro de la instalación de NodeJS)



Los ejemplos se realizarán utilizando el siguiente editor:

- Visual Studio Code (<https://code.visualstudio.com/>)



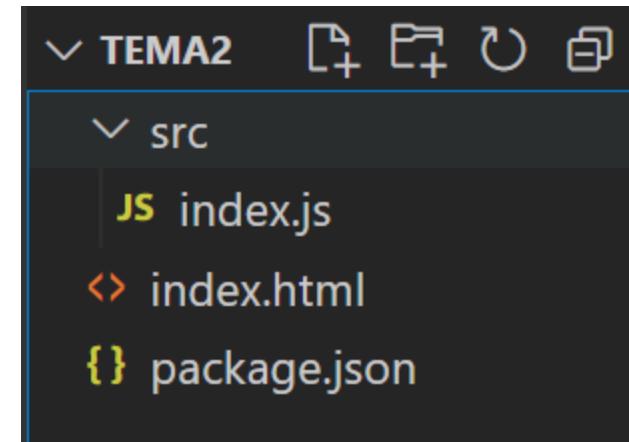
Configuración Inicial

Como primer paso debes crear un directorio de trabajo e inicializarlo con npm, posteriormente deberás instalar los módulos de **webpack** y **webpack-cli** utilizados para ejecutar webpack en la línea de comandos.

```
mkdir modulo5  
cd modulo5  
npm init -y  
npm install webpack webpack-cli -D  
code .
```

Directorio Inicial

- Te pedimos que generes la siguiente estructura de directorio dentro de tu proyecto utilizando Visual Studio Code.



¡Vamos a Codificar!

A continuación empezaremos a trabajar con los archivos que has generado para mostrarte el funcionamiento básico de Webpack usando los siguientes pasos:

1. Generar una función de Javascript en un archivo separado (`./src/index.js`)
2. Utilizar una biblioteca externa a través de un CDN en el archivo (`index.html`) que necesite la función de Javascript en el archivo separado (`./src/index.js`) para trabajar.
3. Ejecutar ambos scripts dentro de la página web (`index.html`)
4. Adaptar el código para que funcione con Webpack
5. Generar un paquete utilizando Webpack
6. Estructurar un archivo de configuración de Webpack y el archivo de configuración del proyecto activo (`package.json`)

DIPLOMADO

Front End

UVM

TEMA 2

Práctica

Generando Nuestro Primer Paquete con Webpack

DIPLOMADO

Front End

TEMA 3

Procesado del
Código CSS con Webpack

Cargador (loader)

Por defecto, Webpack únicamente entiende archivos Javascript y JSON. Los cargadores permiten a Webpack procesar otro tipo de archivos (por ejemplo, CSS) y los pueda convertir en módulos válidos agregándolos al grafo de dependencia para que la aplicación final pueda consumirlos.

Los cargadores tienen dos propiedades asociadas:

- La propiedad de test que identifica qué archivo o archivos deben transformarse.
- La propiedad de use que indica qué cargador se debe usar para realizar la transformación.

Trabajando con Cargadores, Activos y Datos

Seguiremos trabajando dentro del mismo proyecto por lo que te pedimos que instales los siguientes elementos de forma local:

- CSS: `npm install -D style-loader css-loader`
- CSV: `npm install -D csv-loader`
- JSON, YAML: `npm install -D yamljs json5`

Los paquetes que instalaste nos ayudarán a que Webpack pueda procesar diferentes tipos de archivos útiles para la construcción de proyectos.

¡Vamos a Codificar!

- Los pasos que seguiremos durante este tema para que comprendas la integración de archivos utilizando Webpack son:
 1. Cargar CSS a nuestra aplicación usando style-loader y css-loader .
 2. Integrar imágenes y fuentes a tu aplicación usando type: 'asset/resource'.
 3. Cargar datos de archivos CSV con csv-loader.
 4. Procesar información en archivos YAML y JSON.

TEMA 3

Práctica

Integrando CSS, CSV, YAML y JSON5 a un Proyecto usando Webpack

DIPLOMADO

Front End



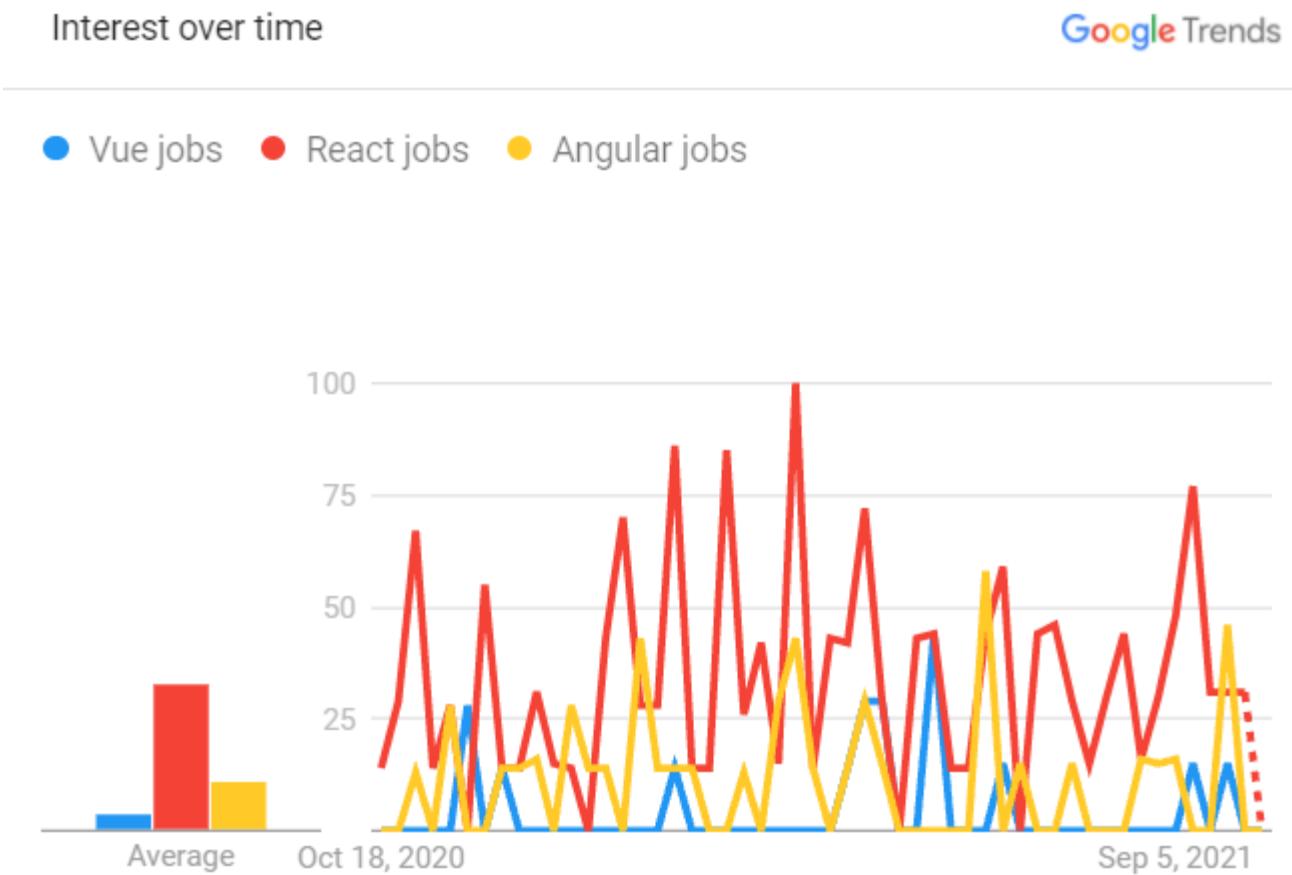
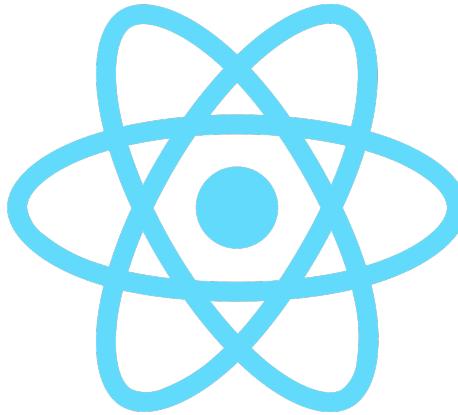
DIPLOMADO

Front End

TEMA 4

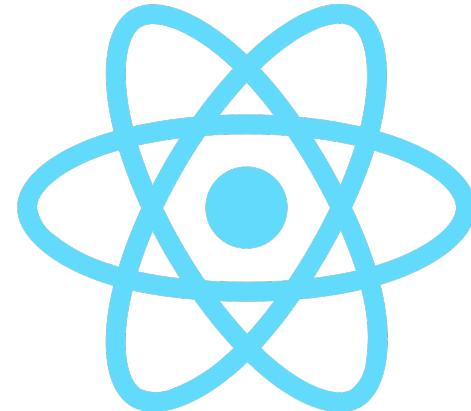
Usar Webpack
en el Ámbito de las
Aplicaciones Front-end

Frameworks Front-end ¿Cuál Usar?



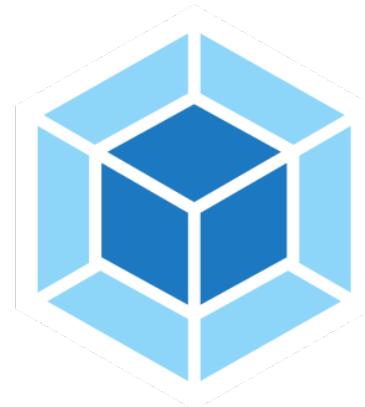
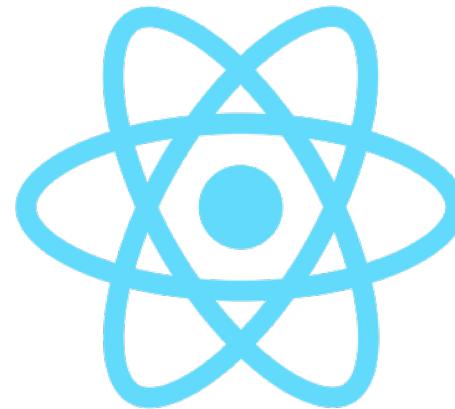
React

React es una biblioteca Javascript de código abierto diseñada para crear interfaces de usuario (front-end) facilitando el desarrollo de aplicaciones SPA o de una sola página. Es mantenido por Facebook y una comunidad de software libre.



React

Babel es un transcompilador de JavaScript gratuito y de código abierto que se utiliza principalmente para convertir el código ECMAScript 2015+ en una versión de JavaScript compatible con versiones anteriores que puede ser ejecutada por motores Javascript más antiguos.



Trabajando con React y Babel

Te recomendamos generar un proyecto nuevo para utilizar React y Babel y además de Webpack instalaras los siguientes elemento:

- React: `npm install react react-dom`
- Babel: `npm install -D @babel/core @babel/preset-env @babel/preset-react @babel/cli babel-loader`

Los paquetes que instalaste nos ayudarán a que Webpack pueda trabajar junto React y Babel

¡Vamos a Codificar!

- Los pasos que seguiremos durante este tema para que comprendas el uso de Webpack con frameworks SPA son:
 1. Generar un nuevo proyecto.
 2. Instalar los módulos necesarios de trabajo.
 3. Configurar Webpack a través de webpack.config.js para poder utilizar React.
 4. Generar el paquete de la aplicación
 5. Probar la aplicación generada

DIPLOMADO

Front End

TEMA 4

Práctica

Integrando React y
Babel con Webpack

DIPLOMADO

Front End

TEMA 5

Como usar
Webpack en Sitios Web

Transpilar Código SASS y TypeScrip

Existen más tecnologías avanzadas que normalmente se incluyen en el diseño de sitios y aplicaciones web. Entre ellas algunas de las más útiles son:

SASS: integra características adicionales a CSS como anidamiento, mixins, herencia y otras ingeniosas ventajas que lo ayudan a escribir CSS robusto y fácil de mantener.

TypeScript: TypeScript es un lenguaje de programación fuertemente tipado que se basa en Javascript y brinda mejores herramientas a cualquier escala de proyecto.

Trabajando con Sitios Web Avanzados

Para utilizar TypeScript y SASS, además de Webpack instalaras los siguientes elementos:

- TypeScript: `npm install -D typescript ts-loader`
- SASS: `npm install sass-loader sass webpack -D`

Los paquetes que instalaste nos ayudarán a generar aplicación con herramientas útiles para cualquier caso en desarrollo web

¡Vamos a Codificar!

Los pasos que seguiremos durante este tema para que generes una aplicación con SASS y TypeScript son:

1. Instalar los módulos necesarios de trabajo.
2. Generar un archivo SCSS para integrarlo al proyecto de trabajo.
3. Configurar SASS dentro del archivo webpack.config.js
4. Generar un archivo tsconfig.json
5. Adaptar código y bibliotecas existentes
6. Probar la aplicación funcionando

DIPLOMADO

Front End

TEMA 5

Práctica

Integrando SASS, SCSS y TypeScript con Webpack

DIPLOMADO

Front End

TEMA 5

Práctica

TypeScript con Webpack

A black and white photograph of a person's hands typing on a laptop keyboard. The laptop is open, and the hands are positioned on the keys. The background is a light-colored surface with some faint markings.

BUILD TOOLS, WEBPACK, SERVICE WORKERS

UVM

TEMA 6

DIPLOMADO

Front End

Casos de uso
Avanzados con Webpack

Aplicaciones Progresivas Web

Las aplicaciones web progresivas (o Progressive Web Applications - PWA) brindan una experiencia similar a las aplicaciones nativas. Entre las diversas funcionalidades que pueden tener las PWAs es la capacidad de funcionar sin conexión. Esto se logra mediante el uso de una tecnología web llamada Service Workers.



Workbox

Workbox es un conjunto de bibliotecas para escribir y administrar Service Workers y su almacenamiento en caché con la API CacheStorage. En conjunto estas herramientas permiten controlar activos (HTML, CSS, JS, imágenes, etc.) que se usan en un sitio web y se solicitan desde la red o la caché, incluso permitiéndole devolver el contenido en caché cuando está fuera de línea.



Workbox

Trabajando con Service Workers

Para utilizar Service Workers, además de Webpack instalaras los siguientes elementos:

- Workbox: `npm install workbox-webpack-plugin -D`

El paquete que instalaste nos ayudará a utilizar nuestra aplicación offline.

¡Vamos a Codificar!

Los pasos que seguiremos durante este tema para que generes una aplicación que se pueda utilizar offline con Webpack son:

1. Instalar el módulo necesario de trabajo.
2. Probar la aplicación usando un servidor web básico y visualizar que pasa si no esta funcionando.
3. Configurar Workbox dentro del archivo webpack.config.js
4. Registrar el Service Worker.
5. Probar nuevamente la aplicación desconectando el servidor básico.

Conclusiones

- Webpack es una herramienta que empaqueta diversos tipos de archivos asociados a una aplicación para mejorar el proceso de compilación dando grandes ventajas al desarrollo del front-end.
- Webpack puede utilizarse con una gran cantidad de tecnologías como React, Service Workers, Typescript entre otras.

DIPLOMADO

Front End

TEMA 6

Práctica

Service Workers con Webpack y Workbox