

FullStack Developer

Minificación, Empaquetado y despliegue
en JavaScript

Minificación y Empaquetado

Definición: La minificación es el proceso de eliminar todos los caracteres innecesarios de un archivo de código fuente sin cambiar su funcionalidad. Esto incluye la eliminación de espacios en blanco, comentarios, saltos de línea y renombrar variables de manera más corta. El objetivo es reducir el tamaño del archivo y mejorar la velocidad de carga de las páginas web.

Ejemplo de Minificación:

Antes:

```
function sum(a, b) {  
    return a + b;  
}  
console.log(sum(2, 3));
```

Después:

```
function sum(a, b) { return a + b } console.log(sum(2, 3));
```

Empaquetado

Definición: El empaquetado es el proceso de combinar múltiples archivos de JavaScript en uno solo. Esto no solo reduce el número de solicitudes HTTP necesarias para cargar una página web, sino que también permite gestionar mejor las dependencias y módulos.

Ejemplo de Empaquetado:

Imaginemos que tenemos dos archivos JavaScript:

```
// math.js
export function sum(a, b) {
  return a + b;
}

// app.js:
import { sum } from './math.js';

console.log(sum(2, 3));
```

Después de empaquetar, podrías obtener un único archivo bundle.js que contiene el código combinado y optimizado de ambos archivos:

```
!function(e){var n={};function t(o){if(n[o])return n[o].exports;var r=n[o]={i:o,l:!1,exports:{}};return e[o].call(r.exports,r,r.exports,t),r.l=!0,r.exports}t.m=e,t.c=n,t.d=function(e,n,o){t.o(e,n)||Object.defineProperty(e,n,{enumerable:!0,get:o})},t.r=function(e){"undefined"!=typeof Symbol&&Symbol.toStringTag&&Object.defineProperty(e,Symbol.toStringTag,{value:"Module"}),Object.defineProperty(e,"__esModule",{value:!0})},t.t=function(e,n){if(1&n&&(e=t(e)),8&n)return e;if(4&n&&"object"===typeof e&&e.__esModule)return e;var o=Object.create(null);if(t.r(o),Object.defineProperty(o,"default",{enumerable:!0,value:e}),2&n&&"string"!==typeof e)for(var r in e)t.d(o,r,function(n){return e[n]}.bind(null,r));return o},t.n=function(e){var n=e.__esModule?function(){return e.default}:function(){return e};return t.d(n,"a",n),n},t.o=function(e,n){return Object.prototype.hasOwnProperty.call(e,n)},t.p="",t(t.s=0)}([function(e,n,t){"use strict";t.r(n);console.log(5)}}]);
```

Beneficios de la Minificación y Empaquetado

1.Mejora del rendimiento: Menos y más pequeños archivos significan tiempos de carga más rápidos.

2.Reducción del ancho de banda: Archivos más pequeños utilizan menos datos para ser transferidos.

3.Gestión de dependencias: Las herramientas de empaquetado pueden manejar dependencias automáticamente, haciendo el desarrollo más eficiente.

4.Mantenimiento del código: Permite mantener el código fuente organizado en múltiples archivos y módulos mientras que los usuarios finales solo reciben un archivo optimizado.

Uso en el Desarrollo Moderno

En el desarrollo moderno, es común integrar estas herramientas en el flujo de trabajo de desarrollo mediante scripts de construcción automatizados. Por ejemplo, en un proyecto con Node.js, puedes configurar scripts en el archivo package.json para ejecutar tareas de minificación y empaquetado automáticamente con herramientas como Webpack, Terser y Babel.

package.json

Definición: package.json es un archivo fundamental en proyectos que utilizan Node.js y npm (Node Package Manager). Contiene la información necesaria sobre el proyecto y las dependencias necesarias para que funcione. Este archivo se encuentra en la raíz del proyecto y es utilizado por npm para gestionar el paquete.

Componentes Principales de package.json

Metadata del Proyecto:

- **name:** Nombre del paquete.
- **version:** Versión del paquete siguiendo el formato de versionado semántico (semver).
- **description:** Breve descripción del paquete.
- **main:** Punto de entrada del paquete, usualmente un archivo JavaScript.
- **author:** Autor del paquete.
- **license:** Licencia del paquete.

Scripts:

scripts: Permite definir comandos de scripts que se pueden ejecutar con npm run <script-name>.

```
"scripts": {  
  "start": "node index.js",  
  "test": "mocha"  
}
```

En este ejemplo, npm start ejecutará node index.js y npm test ejecutará mocha

Dependencias:

dependencies: Lista de paquetes que necesita el proyecto para funcionar en producción.

```
"dependencies": {  
  "express": "^4.17.1",  
  "mongoose": "^5.10.9"  
}
```

devDependencies: Lista de paquetes necesarios solo para el desarrollo del proyecto.

```
"devDependencies": {  
  "nodemon": "^2.0.4",  
  "eslint": "^7.11.0"  
}
```

Configuración del Módulo:

main:

Define el archivo de entrada principal del proyecto.

```
"main": "index.js"
```

type: "module" en package.json

La configuración type: "module" en el archivo package.json se utiliza para indicar que el código JavaScript en el proyecto utiliza módulos ES (ECMAScript Modules) en lugar de módulos CommonJS. Esto es especialmente relevante a partir de Node.js 12 y versiones posteriores, que han agregado soporte nativo para los módulos ES.

¿Qué son los Módulos ES?

Los módulos ES son una forma estandarizada de importar y exportar funcionalidades entre diferentes archivos JavaScript. Utilizan las palabras clave `import` y `export`, lo que facilita la modularización del código.

Configuración de type: "module"

Para habilitar el uso de módulos ES en tu proyecto Node.js, debes añadir "type": "module" en tu archivo `package.json`. Esto le indica a Node.js que trate todos los archivos `.js` en el proyecto como módulos ES por defecto.

Ejemplo de package.json con type: "module"

```
{  
  "name": "mi-proyecto",  
  "version": "1.0.0",  
  "description": "Un proyecto de ejemplo con Node.js utilizando módulos ES",  
  "main": "index.js",  
  "type": "module",  
}
```

Vite

Definición: Vite es una herramienta de desarrollo de frontend que se centra en proporcionar un entorno de desarrollo rápido y una experiencia de construcción optimizada. Fue creado por Evan You, el creador de Vue.js, y está diseñado para ser rápido, moderno y sencillo de usar.

Características Principales de Vite

1.Desarrollo Rápido:

1. **Servidor de Desarrollo Instantáneo:** Vite proporciona un servidor de desarrollo muy rápido que utiliza el módulo nativo de ES (ES Modules) del navegador para acelerar la recarga.
2. **HMR (Hot Module Replacement):** Vite permite la sustitución en caliente de módulos, lo que significa que puedes ver los cambios en tu código instantáneamente sin recargar la página completa.

3. Construcción Optimizada:

1. **Bundling (Empaquetado):** Vite utiliza esbuild en el proceso de desarrollo para empaquetar archivos rápidamente y Rollup en el proceso de producción para obtener una construcción optimizada y bien configurada.
2. **Minificación y Optimización:** Vite optimiza y minifica automáticamente el código para producción, asegurando que el bundle sea lo más pequeño y eficiente posible.

4. Soporte para Múltiples Marcos de Trabajo:

1. **Integración con Vue, React, Preact, Svelte:** Vite está diseñado para funcionar de manera excelente con estos marcos de trabajo, proporcionando configuraciones y plugins específicos para cada uno.
2. **TypeScript:** Soporte incorporado para TypeScript, lo que facilita el desarrollo con tipado estático.

5. Configuración Sencilla:

1. **Zero Config:** Vite funciona fuera de la caja con una configuración mínima.
2. **Extensible:** Puedes extender la funcionalidad de Vite mediante plugins y configuraciones personalizadas cuando sea necesario.

Ejemplo de Uso de Vite

Instalación

Puedes iniciar un nuevo proyecto con Vite de la siguiente manera:

```
npm create vite@latest
```

Configuración del Proyecto

Durante la configuración, puedes elegir el marco de trabajo que deseas utilizar, como Vue, React, Preact, Svelte, o un proyecto Vanilla.

Desarrollo

Después de la configuración, navega al directorio del proyecto y ejecuta el servidor de desarrollo:

```
npm install  
npm run dev
```

Esto iniciará el servidor de desarrollo de Vite, que proporciona recarga en caliente y una experiencia de desarrollo ultrarrápida.

Beneficios de Usar Vite

1.Velocidad: Vite es extremadamente rápido tanto en el entorno de desarrollo como en el proceso de construcción.

2.Simplicidad: Configuración mínima y fácil de entender.

3.Flexibilidad: Compatible con múltiples marcos de trabajo y extensible mediante plugins.

4.Optimización: Produce bundles optimizados para producción de manera eficiente.

Vite es una excelente opción para desarrolladores que buscan una herramienta moderna y eficiente para desarrollar y construir aplicaciones frontend.

Proyecto con Vite y Vanilla JS

Verifiquemos si tenemos instalado npm y node con `node -v` y `npm -v`

Vamos a abrir una consola (terminal), puedes ser la que tiene Visual Studio Code integrada o la de Window.

En una carpeta vacía vamos a tipear:

```
npm create vite@latest .
```

`npm create vite@latest` Nos va a instalar vite y con el punto estamos diciendo que la instalación se realice en la carpeta que estamos parados

```
PS C:\Users\user> npm create vite@latest .
? Select a framework: » - Use arrow-keys. Return to submit.
> Vanilla
  Vue
  React
  Preact
  Lit
  Svelte
  Solid
  Qwik
  Others
```

Elegimos Vanilla

```
? Select a variant:  
> TypeScript  
   JavaScript
```

Seleccionamos JavaScript

```
Done. Now run:
```

```
npm install  
npm run dev
```

En la consola veremos ese mensaje si todo salió OK

```
> public ●  
  .gitignore U  
  JS counter.js U  
  index.html U  
  javascript.svg U  
  JS main.js U  
  package.json U  
  style.css U
```

Y veremos como se nos crearon todos estos archivos

Tenemos el package.json

```
{  
  "name": "code",  
  "private": true,  
  "version": "0.0.0",  
  "type": "module",  
  "scripts": {  
    "dev": "vite",  
    "build": "vite build",  
    "preview": "vite preview"  
  },  
  "devDependencies": {  
    "vite": "^5.3.1"  
  }  
}
```

Ejecutamos

```
npm install
```

y

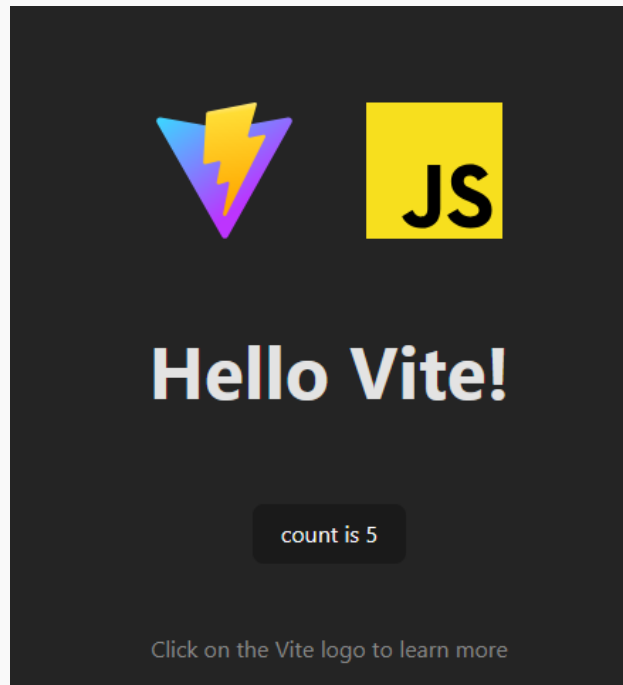
```
npm run dev
```


Veremos en la consola

```
VITE v5.3.3 ready in 1182 ms  
→ Local:   http://localhost:5173/  
→ Network: use --host to expose  
→ press h + enter to show help
```

Y en esa url podemos ver nuestro proyecto

<http://localhost:5173/>



Analizamos main.js

```
document.querySelector('#app').innerHTML = `
  <div>
    <a href="https://vitejs.dev" target="_blank">
      
    </a>
    <a href="https://developer.mozilla.org/en-US/docs/Web/JavaScript" target="_blank">
      
    </a>
    <h1>Hello Vite!</h1>
    <div class="card">
      <button id="counter" type="button"></button>
    </div>
    <p class="read-the-docs">
      Click on the Vite logo to learn more
    </p>
  </div>
`

setupCounter(document.querySelector('#counter'))
```

```
document.querySelector('#app').innerHTML = `
```

Vemos que en el elemento con ID “app”, por medio de innerHTML se introduce un div con varias etiquetas dentro.

```
setupCounter(document.querySelector('#counter'))
```

En este método setupCounter vemos que recibe como parámetro un elemento HTML, un botón.

```
export function setupCounter(element) {  
  let counter = 0  
  const setCounter = (count) => {  
    counter = count  
    element.innerHTML = `count is ${counter}`  
  }  
  element.addEventListener('click', () => setCounter(counter + 1))  
  setCounter(0)  
}
```

Y esta es la función setupCounter que se encarga de modificar el contenido del botón

A partir de ahora no vamos a utilizar CDN's, vamos a instalar las dependencias que vamos a necesitar. Instalemos Bootstrap:



Install via package manager

Install Bootstrap's source Sass and JavaScript files via npm, RubyGems, Composer, or Meteor. Package managed installs don't include documentation or our full build scripts. You can also [use any demo from our Examples repo](#) to quickly jumpstart Bootstrap projects.

```
$ npm install bootstrap@5.3.3
```



Vamos a crear algunos componentes como navbar, footer, about y home

```
const footer = `

Home



Features



Pricing



FAQs



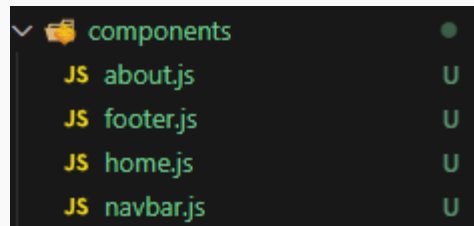
About



© 2024 Company, Inc

`;  
  
export default footer;
```

En home, about y navbar hacemos lo similar



Hacemos una función que se va a encargar de renderizar según el botón que presionemos

```
import about from "../components/about.js";
import home from "../components/home.js";

const app = () => {
  window.addEventListener("DOMContentLoaded", () => {
    const homeBtn = document.querySelector('#home');
    const aboutBtn = document.querySelector('#about');
    const mainContent = document.querySelector('main');

    homeBtn.addEventListener('click', () => {
      mainContent.innerHTML = home;
    });

    aboutBtn.addEventListener('click', () => {
      mainContent.innerHTML = about;
    });
  });
}

export default app
```

Y en main creamos el contenido principal, y luego de renderizarlo, llamamos a la función app para que el contenido vaya cambiando

```
import 'bootstrap/dist/css/bootstrap.min.css';
import 'bootstrap';
import navbar from './components/navbar.js';
import footer from './components/footer.js';
import app from './app.js';
import home from './components/home.js';

document.querySelector('#app').innerHTML = `
  <div class="container-fluid">
    ${navbar}
    <main>
      ${home}
    </main>
    ${footer}
  </div>
`;

app();
```



Vertically centered hero sign-up form

Below is an example form built entirely with Bootstrap's form controls. Each required form group has a validation state that can be triggered by attempting to submit the form without completing it.

Email address

Password

☐ Remember me

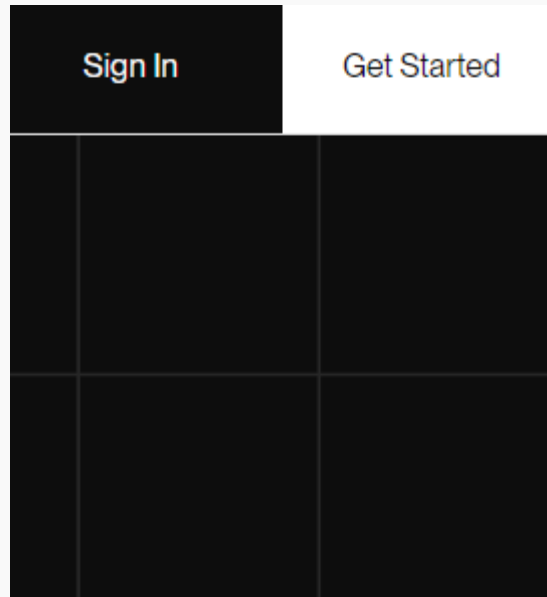
Sign up

By clicking Sign up, you agree to the terms of use.

Despliegue

Vamos a desplegar en render.com un sitio estático

Hacemos click en Sign In



Recomiendo loguearse con GitHub

Sign In to Render



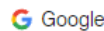
GitHub



GitLab



Bitbucket



Google

or

Email

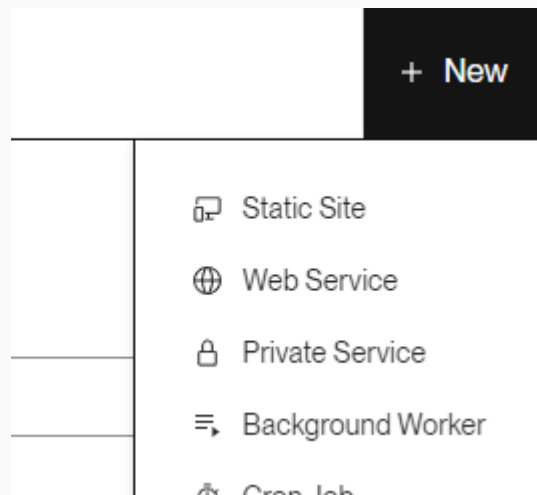
Password

Sign in

Creamos un repositorio y lo subimos a GitHub

Y en render.com en +New

Elegimos Static Site



Buscan el repositorio y presionamos en "Connect"

Connect a repository

🔍 Search



carlos8788 / vite_ejemplo 🏠 • 2 minutes ago

Connect

Si están trabajando en la rama master y el repositorio está en la carpeta raíz, esto lo deben dejar como lo ven, por defecto.

Name

A unique name for your static site.

Branch

The Git branch to build and deploy.

Root Directory Optional

If set, Render runs commands from this directory instead of the repository root. Additionally, code changes outside of this directory do not trigger an auto-deploy. Most commonly used with a [monorepo](#).

Build Command

Render runs this command to build your app before each deploy.

Publish directory

The relative path of the directory containing built assets to publish. Examples: `./`, `./build`, `dist` and `frontend/build`.

Si están trabajando en la rama master y el repositorio está en la carpeta raíz, solo van a agregar `"/dist"` en `"Publish directory"`.

Name

A unique name for your static site.

Branch

The Git branch to build and deploy.

Root Directory Optional

If set, Render runs commands from this directory instead of the repository root. Additionally, code changes outside of this directory do not trigger an auto-deploy. Most commonly used with a [monorepo](#).

Build Command

Render runs this command to build your app before each deploy.


Publish directory

The relative path of the directory containing built assets to publish. Examples: `./`, `./build`, `dist` and `frontend/build`.

Presionan **Deploy Static Site**

Environment Variables

Set environment-specific config and secrets (such as API keys), then read those values from your code. [Learn more](#).

+ Add Environment Variable Add from .env

▼ **Advanced**



Deploy Static Site


Si todo está OK vamos a ver el  Live


Y hacemos click en el link que vemos <https://vite-ejemplo.onrender.com>

STATIC SITE

vite_ejemplo

 carlos8788 / vite_ejemplo  master

<https://vite-ejemplo.onrender.com> 

Manual Deploy 

Events

Environment


Redirects/Rewrites

Headers


Previews


Metrics



Settings


July 8, 2024 at 7:22 AM  Live


[89cd383](#) ejemplo


All logs 


 Search


 Live tail  GMT-3








Jul 8 07:23:01 AM  Round 0 vulnerabilities


Jul 8 07:23:01 AM 


Jul 8 07:23:01 AM  > code@0.0.0 build


Jul 8 07:23:01 AM  > vite build


Jul 8 07:23:01 AM 


Jul 8 07:23:01 AM  vite v5.3.3 building for production...


Jul 8 07:23:01 AM  transforming...


Jul 8 07:23:02 AM  ✓ 67 modules transformed.


Jul 8 07:23:03 AM  rendering chunks...


Jul 8 07:23:03 AM  computing gzip size...



Jul 8 07:23:03 AM  dist/index.html 0.45 kB | gzip: 0.29 kB

Jul 8 07:23:03 AM  dist/assets/index-DUEfxN0n.css 231.03 kB | gzip: 30.70 kB

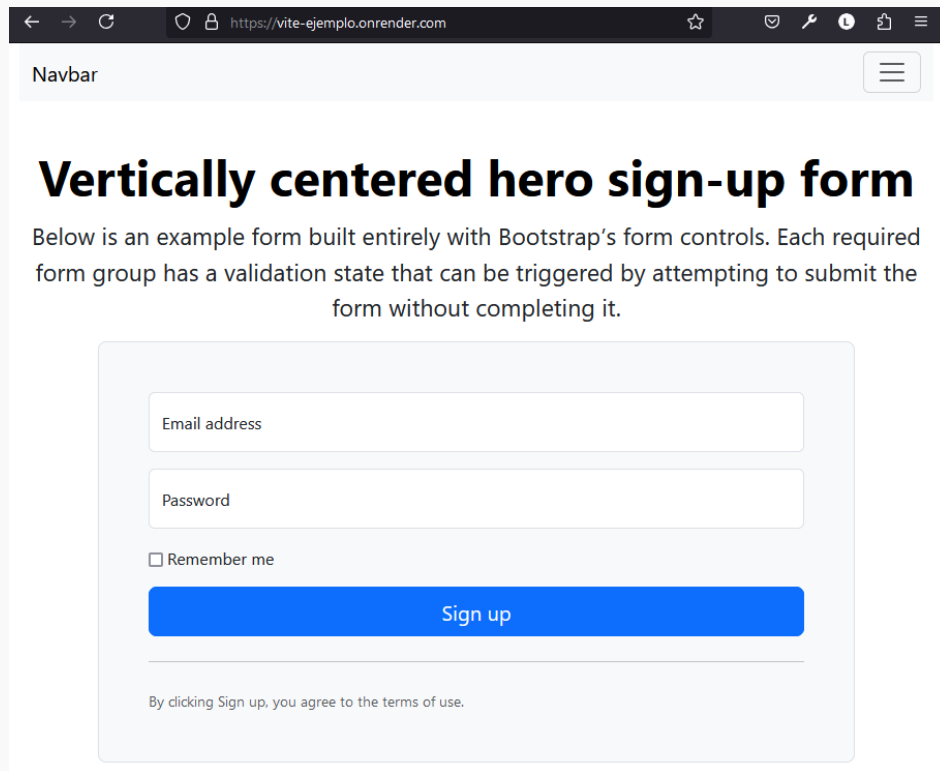
Jul 8 07:23:03 AM  dist/assets/index-C6m5-T6z.js 96.14 kB | gzip: 26.67 kB

Jul 8 07:23:03 AM  ✓ built in 1.23s

Jul 8 07:23:04 AM  ==> Uploading build...

Jul 8 07:23:12 AM  ==> Your site is live 

Ahí vemos como ya nuestro proyecto está desplegado y en “producción”



The screenshot shows a web browser window with the address bar displaying `https://vite-ejemplo.onrender.com`. The page has a light gray background. At the top, there is a "Navbar" section with a hamburger menu icon on the right. Below the navbar, the main heading is "Vertically centered hero sign-up form" in a large, bold, black font. Underneath the heading, there is a paragraph of text: "Below is an example form built entirely with Bootstrap's form controls. Each required form group has a validation state that can be triggered by attempting to submit the form without completing it." Below this text is a light gray rounded rectangle containing the sign-up form. The form consists of two input fields: "Email address" and "Password", both with light gray borders and placeholder text. Below the "Password" field is a checkbox labeled "Remember me". At the bottom of the form is a blue button with the text "Sign up" in white. Below the button, there is a thin horizontal line and a small line of text: "By clicking Sign up, you agree to the terms of use."

Ejercicio

Crear un proyecto Vite Vanilla JS y desplegarlo en render.com