Vite

Que resuelve y por qué debes usarlo

Qué es Vite

Vite

Next Generation ¿Frontend Tooling?

Get ready for a development environment that can finally catch up with you.

Get Started | Why Vite? | View on GitHub | 🎉 ViteConf 23!

Instant Server Start

On demand file serving over native ESM, no bundling required!

Lightning Fast HMR

Hot Module Replacement (HMR) that stays fast regardless of app size.

Rich Features

Out-of-the-box support for TypeScript, JSX, CSS and more.

Optimized Build

Pre-configured Rollup build with multi-page and library mode support.

Universal Plugins

Rollup-superset plugin interface shared between dev and build.

Fully Typed APIs

Flexible programmatic APIs with full TypeScript typing.

Vite es

- Lanzado el 2020 por Evan You
- Creado inicialmente como herramienta de desarrollo de Vue
- Usa principalmente Rollup, creado por Rich Harris (Svelte)
- Una herramienta que combina:
  - Bundlers
  - Transpiladores
  - Dev Server con HMR
- Es un **ecosistema** de herramientas

https://vitejs.dev/guide/features.html
https://youtu.be/fxQXUqdzzFw

Vite

Next Generation Frontend Tooling

Get ready for a development environment that can finally catch up with you.

Get Started | Why Vite? | View on GitHub | 🎉 ViteConf 23!

💡 Dev Server

Instant Server Start

On demand file serving over native ESM, no bundling required!

⚡ con HMR

Lightning Fast HMR

Hot Module Replacement (HMR) that stays fast regardless of app size.

🛠️ Transpilar

Rich Features

Out-of-the-box support for TypeScript, JSX, CSS and more.

📦 Bundlear

Optimized Build

Pre-configured Rollup build with multi-page and library mode support.

🔩 Ecosistema

Universal Plugins

Rollup-superset plugin interface shared between dev and build.

🔑 Ecosistema

Fully Typed APIs

Flexible programmatic APIs with full TypeScript typing.

Los problemas que resuelve Vite

1. Dev Server con HMR

- HMR: Hot Module Replacement
- Generalización: Hot Swapping, Hot Code Replacement
- ¿Cómo actualizamos cambios de código?
  - Live Reloading: Stop & Start
  - Hot Reloading: Mantener estado
- Vite cambia módulos cuando estos fueron actualizados
- Gracias a su API, permite la creación de **plugins** de HMR

https://bjornlu.com/blog/hot-module-replacement-is-easy

2. Transpilar

Transformar un código de un lenguaje a otro lenguaje similar

2. Transpilar .tsx

```tsx
type CounterProps = { initialCount?: number };

function Counter({ initialCount = 0 }: CounterProps) {
  const [count, setCount] = React.useState(initialCount);

  return (
    <div>
      <p>Count: {count}</p>
      <button onClick={() => setCount(count + 1)}>Increment</button>
    </div>
  );
}

const root = createRoot(document.getElementById("counter"));
root.render(<Counter />);
```

```
Uncaught SyntaxError: Unexpected identifier 'CounterProps' (at v1.tsx:4:6)   v1.tsx:4
```

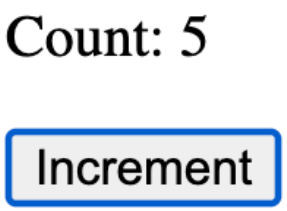
2. Transpilar .tsx → .jsx

```
function Counter({ initialCount = 0 }) {
  const [count, setCount] = React.useState(initialCount);

  return (
    <div>
      <p>Count: {count}</p>
      <button onClick={() => setCount(count + 1)}>Increment</button>
    </div>
  );
}

const root = createRoot(document.getElementById("counter"));
root.render(<Counter />);
```

```
Uncaught SyntaxError: Unexpected token '<' (at v2.jsx:8:5)    v2.jsx:8
```

2. Transpilar .tsx → .jsx → .js

```
const jsx = React.createElement;

function Counter({ initialCount = 0 }) {
  const [count, setCount] = React.useState(initialCount);

  return (
    jsx("div", null,
      jsx("p", null, "Count: ", count),
      jsx("button", { onClick: () => setCount(count + 1) }, "Increment")
    )
  );
}

const root = createRoot(document.getElementById("counter"));
root.render(jsx(Counter, null));
```



2. Transpilar

Vite posee con muy buenos defectos y **plugins** para no preocuparse del pipeline de transpilación.

```ts
import { defineConfig } from "vite";
import react from "@vitejs/plugin-react-swc";

export default defineConfig({
  plugins: [react()],
});
```

Webpack

```ts
const path = require('path');

module.exports = {
  entry: './src/index.ts',
  module: {
    rules: [
      {
        test: /\.tsx?$/,
        use: 'ts-loader',
        exclude: /node_modules/,
      },
    ],
  },
  resolve: {
```

3. Bundlear

- Traducción literal: Empaquetar
- Se necesita **mandar** JavaScript de una manera que el navegador entienda **“como ejecutarlo en conjunto”**
- Cómo y qué se manda podría ser optimizado

Pero, ¿qué problema resuelve bundlear?

¿código que se ejecute en conjunto el navegador?

Como correr *bien* JS en conjunto en el navegador

Y un poco de historia de JavaScript

Contexto y fechas importantes

- Contexto e hitos importantes:
  - JavaScript inventado en 1995
  - ECMAScript 1997, v3 en 1999, v5 con `"strict mode"` en 2009
  - Termino Asynchronous JavaScript And XML (AJAX) en 2005
  - Lanzamiento de Firefox el 2002, Chrome el 2008, Node el 2009
- CommonJS (Modules/1.0, 2009~2010)
  - Estándar que trae la función `require`
- AMD: Asynchronous Module Definition (2011)
  - Estándar que permite módulos ***en el navegador***

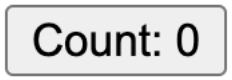
https://requirejs.org/docs/why.html
https://requirejs.org/docs/whyamd.html

JS Originalmente

```html
<!DOCTYPE html>
<html>
  <head>
    <script src="./other.js"></script>
    <script src="./index.js"></script>
  </head>
  <body>
    <button onclick="updateCounter()" id="counter">
      Count: 0
    </button>
  </body>
</html>
```

```js
// index.js
function updateCounter() {
  var count = parseInt(counter.textContent.split(" ")[1]);
  console.log(count);
  counter.textContent = `Count: ${count + 1}`;
}
```

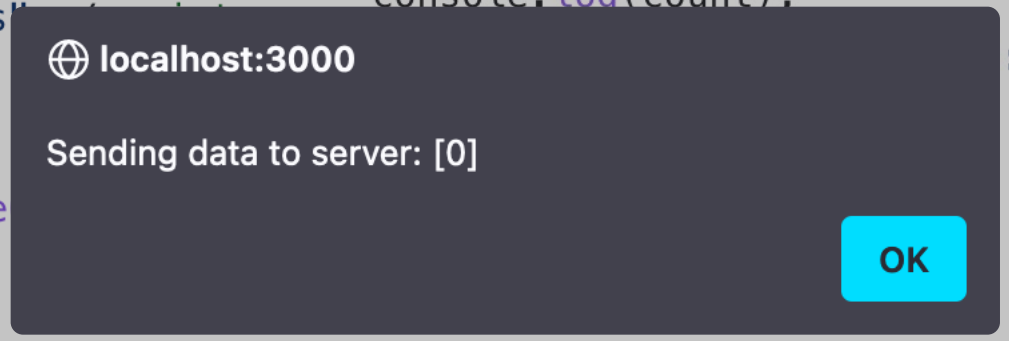
¿Qué hace este código?



JS Originalmente

```html
<!DOCTYPE html>
<html>
  <head>
    <script src="./other.js"></script>
    <script src="./index.js
  </head>
  <body>
    <button onclick="update
      Count: 0
    </button>
  </body>
</html>
```

```js
// index.js
function updateCounter() {
  var count = parseInt(counter.textContent.split(" ")[1]);
  console.log(count);
                                        : ${count + 1}`;
```



¿Qué hace este código?

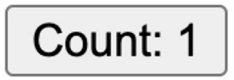
Count: 0

JS Originalmente

```html
<!DOCTYPE html>
<html>
  <head>
    <script src="./other.js"></script>
    <script src="./index.js"></script>
  </head>
  <body>
    <button onclick="updateCounter()" id="counter">
      Count: 0
    </button>
  </body>
</html>
```

```js
// index.js
function updateCounter() {
  var count = parseInt(counter.textContent.split(" ")[1]);
  console.log(count);
  counter.textContent = `Count: ${count + 1}`;
}
```

```js
// other.js
var console = {
  log: function (...data) {
    alert("Sending data to server: " + JSON.stringify(data));
  },
};
```



¿Cómo evitamos este problema?

Soluciones: IIFE

```
;(function () {
  // Código se ejecuta en funciones
  // para no encuciar el scope global
})();
```

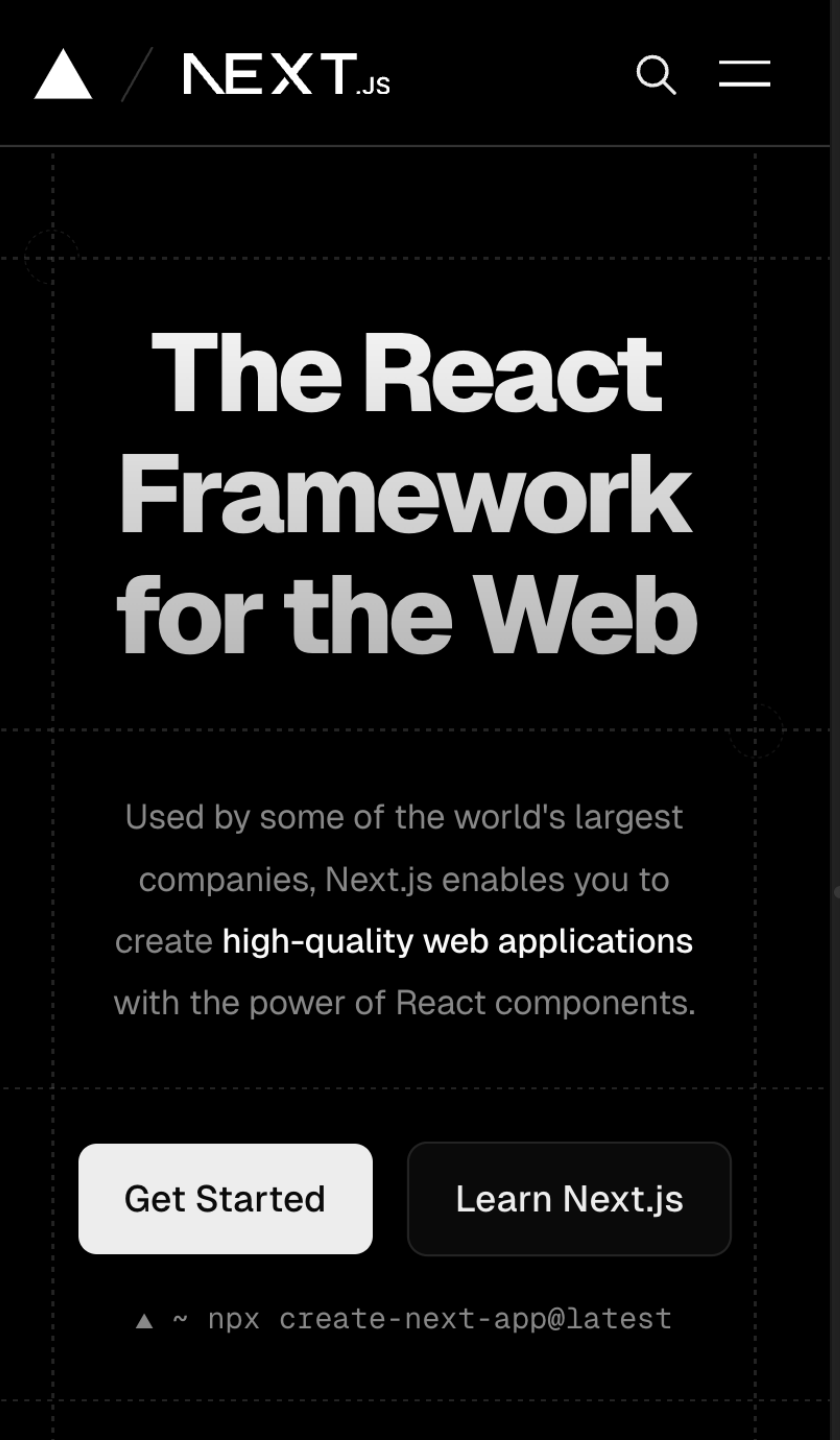
¿Cómo se manejan dependencias?

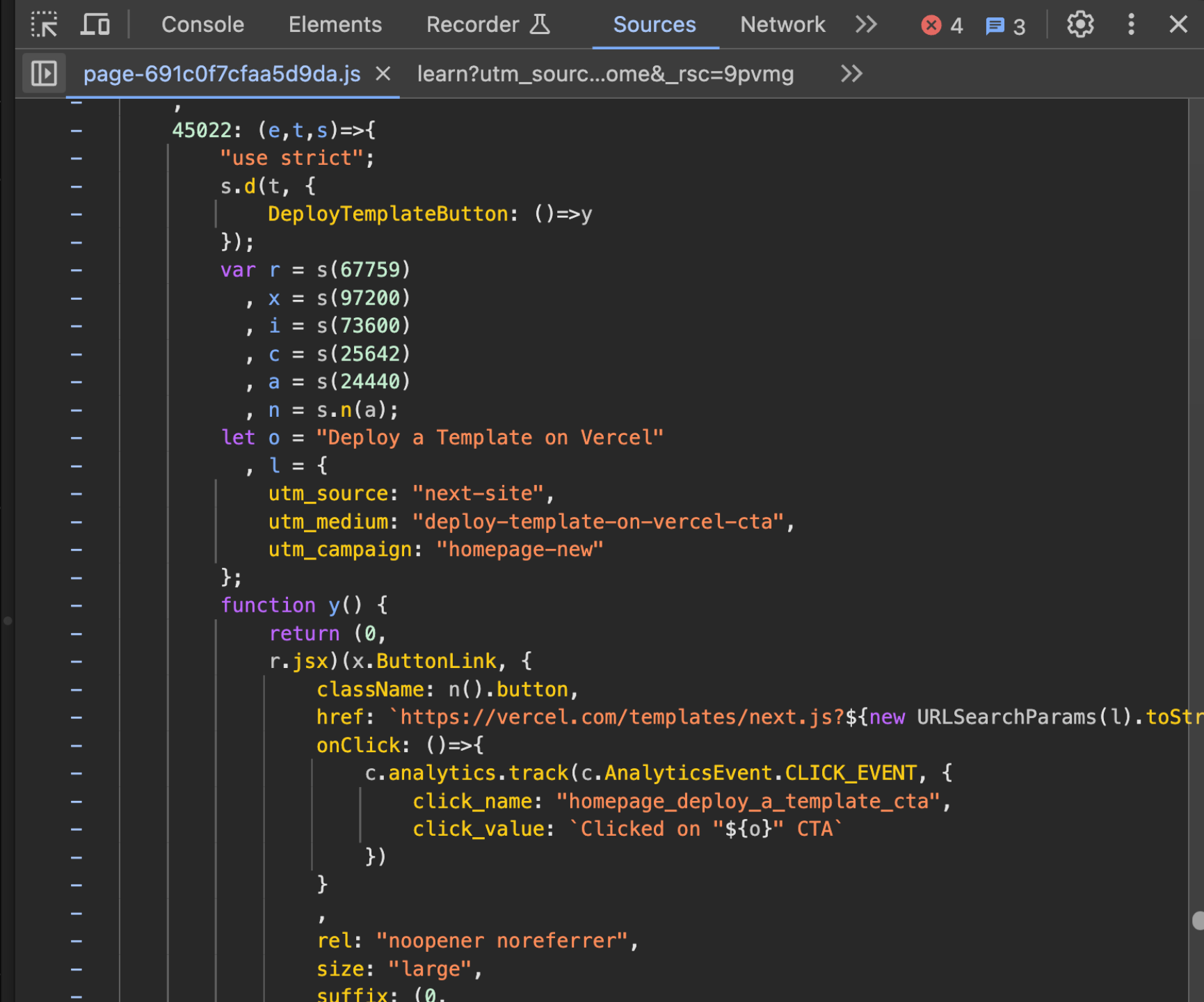
Soluciones: AMD y similares

```
define('this-module-id', function (module, exports, require) {
    var dep1 = require('dependency-1');
    var dep2 = require('dependency-2');

    class Foo() {
      // ...
    }

    function util() {
        // ...
    }

    module.default = Foo;
    exports.util = util;
});
```

Un bundler junta el código

webpack


NEXT.JS
The React Framework for the Web
Used by some of the world's largest companies, Next.js enables you to create high-quality web applications with the power of React components.
Get Started
Learn Next.js
▲ ~ npx create-next-app@latest


Console
Elements
Recorder
Sources
Network
4
3
page-691c0f7cfaa5d9da.js
learn?utm_sourc...ome&_rsc=9pvmg
45022: (e,t,s)=>{
"use strict";
s.d(t, {
DeployTemplateButton: ()=>y
});
var r = s(67759)
, x = s(97200)
, i = s(73600)
, c = s(25642)
, a = s(24440)
, n = s.n(a);
let o = "Deploy a Template on Vercel"
, l = {
utm_source: "next-site",
utm_medium: "deploy-template-on-vercel-cta",
utm_campaign: "homepage-new"
};
function y() {
return (0,
r.jsx)(x.ButtonLink, {
className: n().button,
href: `https://vercel.com/templates/next.js?${new URLSearchParams(l).toStr
onClick: ()=>{
c.analytics.track(c.AnalyticsEvent.CLICK_EVENT, {
click_name: "homepage_deploy_a_template_cta",
click_value: `Clicked on "${o}" CTA`
})
}
,
rel: "noopener noreferrer",
size: "large",
suffix: (0,

ECMAScript 6 - 2015

- 20 años de la creación de JS, 6 de la última de ECMAScript
- Soluciona grandes dolores que han atormentado JavaScript
  - `const` y `let` como mejor alternativa a `var`
  - Clases, `for...of` para iteradores, maps, sets, template literals
  - Introducción de Promesas, destructuring, arrow functions
  - **Modulos** con:
    - `import` y `export`
    - Con scope local en vez de global
    - Estricto por defecto
    - Carga y ejecución asíncrona

https://exploringjs.com/es6/ch_overviews.html

ECMAScript 6 - 2015

- 2015 se crea Rollup, que luego permitiría generar ES6
- En 2017
  - Gana soporte en los navegadores más usados
  - Se empieza a integrar a node (estable en 2020)
- 2018 se añade soporte de ES6 a vue-cli (precesor de vite)
- 2020 se crea Vite

Resumiendo

- La historia es:
  - 1995 a 2000 se establece JS y ECMAScript
  - 2007 a 2011 node, Chrome, CommonJS, AMD, strict mode
  - 2015 a 2020 se crea Rollup y surge ES6 que revoluciona JS
- Bundlers para llevar código en conjunto al navegador
- ES6 integra una **forma nativa** de cargar módulos

Fin mini charla de como correr JS en conjunto en el navegador

3. Bundlear

- Vite trata de **no bundlear**
  - El servidor entrega código fuente ESM, transpilado si es necesario
  - Se pre-bundlea solo las dependencias y se cachean
  - Al hacer build, se junta el código, pero con output ESModule
- El poder y velocidad de Vite se debe a ser lazy
  - Solo es necesario pre-procesar dependencias y servidor
  - Cada módulo se carga con una request HTTP al servidor
  - El servidor transpila el código y lo entrega
  - Si no ha cambiado código, retorna 304 Not Modified

3. Bundlear

Vite aprovecha como bundle de código...

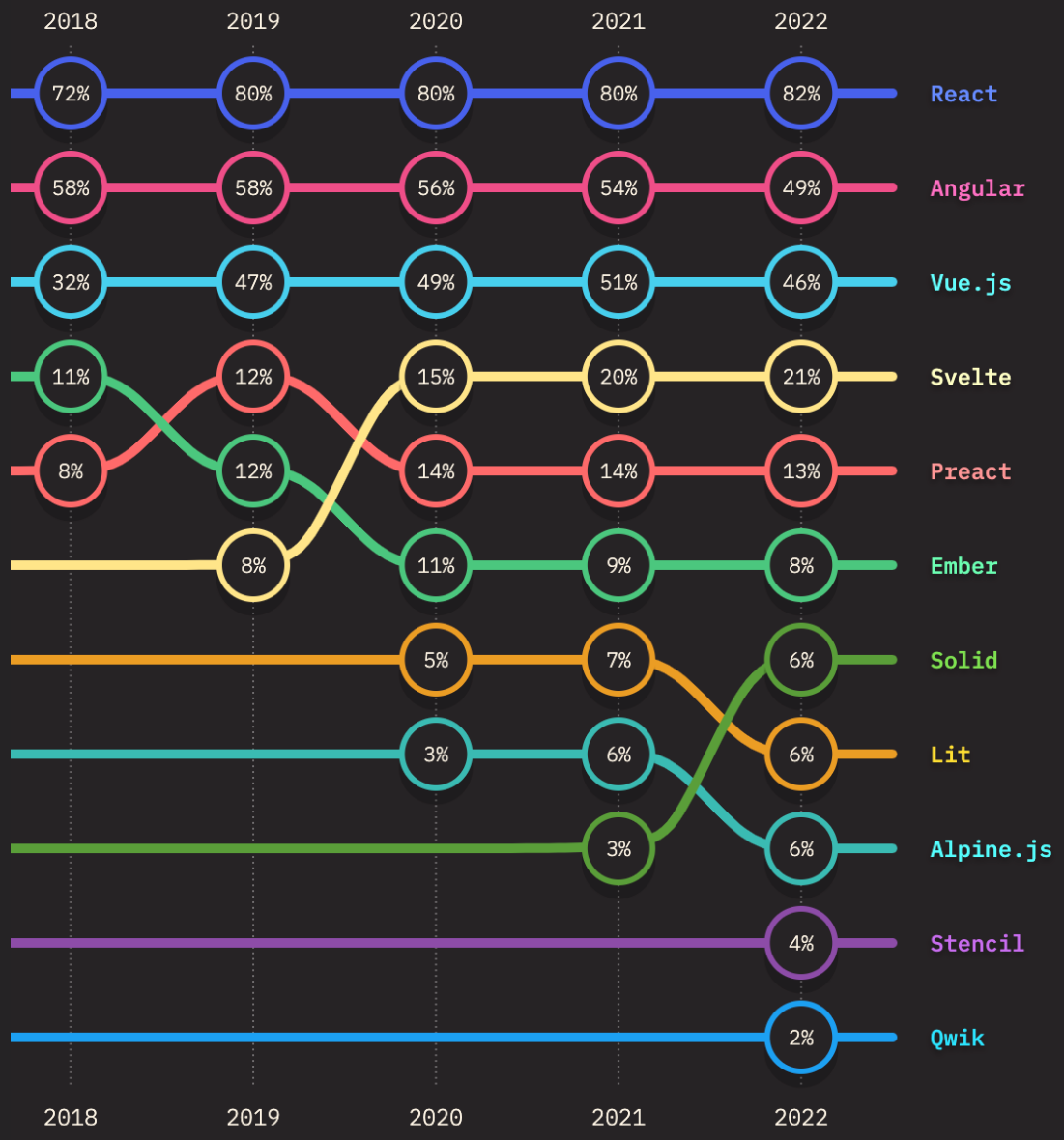
- Tree-shaking: eliminar código que no se necesita
- Common chunk splitting: no repetir código
- Hash en nombres de archivos: cache infinito
- Variables de entorno públicas

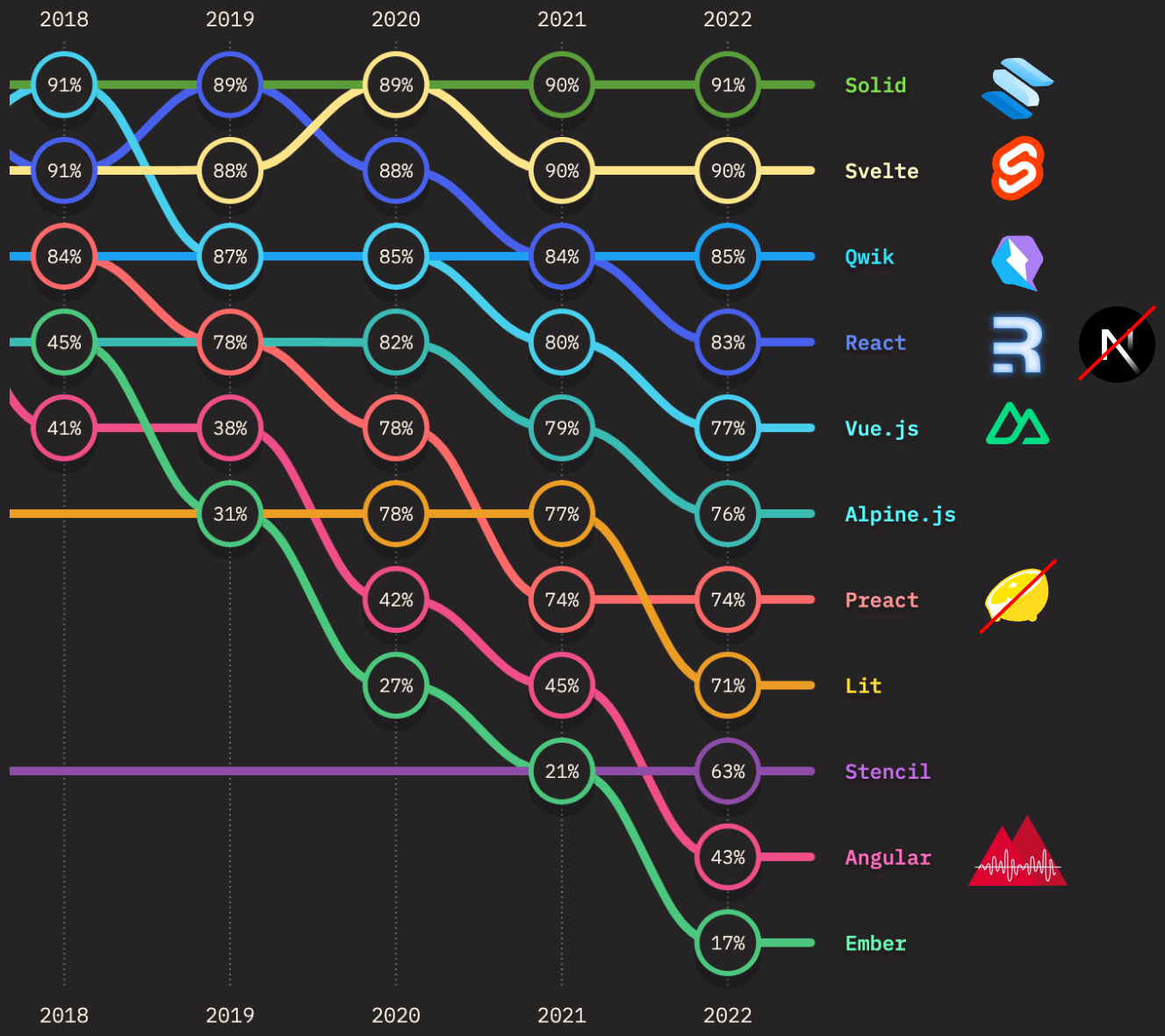
El Ecosistema

Librerias

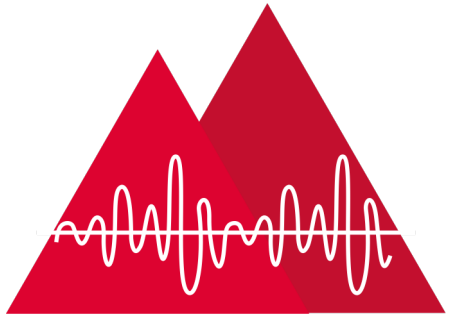
Frameworks

Ranking de uso y retención 2022





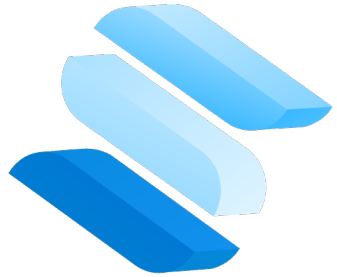
https://2022.stateofjs.com/en-US/libraries/front-end-frameworks/



+

Nitro =

+Vinxi



Framework Remix como plugin

```
import { vitePlugin as remix } from "@remix-run/dev";
import { installGlobals } from "@remix-run/node";
import { defineConfig } from "vite";
import tsconfigPaths from "vite-tsconfig-paths";

installGlobals();

export default defineConfig({
  plugins: [remix(), tsconfigPaths()],
});
```

Framework Solid Start (con Vinxi)

```
import { defineConfig } from "@solidjs/start/config";

export default defineConfig({
  vite: { /* configuración expuesta de vite */ },
  server: { /* configuración expuesta de nitro (expuesta por vinxi) */ },
});
```

Optimizar imágenes y HTML

```
import { defineConfig } from "vite";
import { ViteImageOptimizer } from "vite-plugin-image-optimizer";
import { createHtmlPlugin } from "vite-plugin-html";

export default defineConfig(() => {
    return {
        build: { assetsInlineLimit: 15360 },
        plugins: [ViteImageOptimizer({}), createHtmlPlugin({ minify: true })],
    };
});
```

Cerrando

Vite es

- Una herramienta que combina herramientas
  - Dev Server rápido con HMR
  - Transpiladores
  - Bundlers
- Usado como base para **todo un ecosistema**
- Pionero en JS moderno

```
npm create vite@latest
```