Guía — Componente Avanzado React + TypeScript (local-first i18n, theme dark/light)

----------------------------------------------------------------------------------

Objetivo

========

Crear un componente modular en React + TypeScript que:

- Respete el \*\*tema global\*\* (dark/light).

- Use i18n con \*\*prioridad local-first\*\* (primero el diccionario del componente; si la key no existe, usa el global; de lo contrario, fallback a la key).

- Acepte visibilidad opcional por `index.visibility.ts` (devices, orientación y breakpoints).

- Mantenga estructura clara (types, hook, provider, view, utils) y sin dependencias circulares.

- Sea portable entre proyectos y fácil de documentar.

Estructura final (actualizada)

==============================

[REEMPLAZA\_CON\_NOMBRE]/

├── index.tsx

├── i18n/

│ ├── es.json

│ ├── en.json

│ └── index.ts

├── index.visibility.ts # (OPCIONAL)

├── [REEMPLAZA\_CON\_NOMBRE].module.css

├── [REEMPLAZA\_CON\_NOMBRE].module.ts

├── README.md

├── types/

│ ├── visibility.ts

│ ├── component.ts

│ └── index.ts

├── hook/

│ ├── useThemeSafe.ts

│ ├── useI18nMerge.ts

│ ├── useVisibility.ts

│ └── index.ts

├── provider/

│ └── index.tsx

├── view/

│ ├── index.tsx

│ └── index.ts

├── utils/

│ ├── dom.ts

│ ├── visibility.ts

│ ├── i18n.ts

│ └── index.ts

└── documentation/

├── index.ts

├── preview.tsx

├── react.tsx

├── html.tsx

├── css.tsx

└── usage.tsx

Tipos

=====

types/visibility.ts

-------------------

export type Device = 'mobile' | 'tablet' | 'desktop';

export type Orientation = 'portrait' | 'landscape';

export type VisibilitySetting = 'visible' | 'hidden';

export interface BreakpointRule {

minWidth: number;

maxWidth: number; // Usa Infinity para “sin tope”

visibility: VisibilitySetting;

}

export interface VisibilityConfig {

allowedDevices?: Device[];

allowedOrientations?: Orientation[];

breakpoints?: BreakpointRule[];

defaultVisibility?: VisibilitySetting; // por defecto 'visible'

}

types/component.ts

------------------

import type { CSSProperties, ReactNode } from 'react';

import type { VisibilityConfig, Device, Orientation } from './visibility';

export interface [REEMPLAZA\_CON\_NOMBRE]Props {

id?: string;

className?: string;

style?: CSSProperties;

titleKey?: string;

children?: ReactNode;

config?: VisibilityConfig; // override sobre index.visibility.ts si se pasa

}

export interface [REEMPLAZA\_CON\_NOMBRE]Context {

theme: 'light' | 'dark';

t: (key: string, params?: Record<string, string | number>) => string;

visibilityConfig?: VisibilityConfig;

isVisible: boolean;

device: Device;

orientation: Orientation;

width: number;

}

types/index.ts

--------------

export \* from './visibility';

export \* from './component';

i18n

====

i18n/en.json

------------

{ "title": "Default Title", "cta": "Continue", "empty": "No data available" }

i18n/es.json

------------

{ "title": "Título por defecto", "cta": "Continuar", "empty": "No hay datos disponibles" }

i18n/index.ts

-------------

import es from './es.json';

import en from './en.json';

export const localDictionaries = { es, en } as const;

export const getLocalDict = (lang?: string) => {

const pick = (lang || 'en').toLowerCase().startsWith('es') ? 'es' : 'en';

return localDictionaries[pick];

};

Utils

=====

utils/dom.ts

------------

import type { Orientation } from '../types';

export const detectWidth = (def = 1024) =>

typeof window === 'undefined' ? def : (window.innerWidth || def);

export const detectOrientation = (): Orientation => {

if (typeof window === 'undefined') return 'landscape';

const { innerWidth: w, innerHeight: h } = window;

return h >= w ? 'portrait' : 'landscape';

};

utils/visibility.ts

-------------------

import type { VisibilityConfig, Device, Orientation } from '../types';

export const detectDevice = (w: number): Device =>

w <= 640 ? 'mobile' : w <= 1024 ? 'tablet' : 'desktop';

export const isVisibleByConfig = (

cfg: VisibilityConfig | undefined,

width: number,

device: Device,

orientation: Orientation

): boolean => {

if (!cfg) return true;

const { allowedDevices, allowedOrientations, breakpoints, defaultVisibility = 'visible' } = cfg;

if (allowedDevices?.length && !allowedDevices.includes(device)) return false;

if (allowedOrientations?.length && !allowedOrientations.includes(orientation)) return false;

if (!breakpoints?.length) return defaultVisibility !== 'hidden';

const rule = breakpoints.find(b => width >= b.minWidth && width <= b.maxWidth);

return (rule?.visibility ?? defaultVisibility) !== 'hidden';

};

utils/i18n.ts (Local-first por defecto)

---------------------------------------

export type TranslationOrder = 'global-first' | 'local-first';

export function interpolate(template: string, params?: Record<string, string | number>) {

if (!params) return template;

return template.replace(/\{(\w+)\}/g, (\_, k) => String(params[k] ?? `{${k}}`));

}

export function makeTranslator(

local?: Record<string, string>,

global?: Record<string, string>,

order: TranslationOrder = 'local-first' // ⬅️ ahora local-first por defecto

) {

return (key: string, params?: Record<string, string | number>) => {

let value: string | undefined;

if (order === 'global-first') {

value = global?.[key] ?? local?.[key];

} else {

value = local?.[key] ?? global?.[key];

}

return interpolate(value ?? key, params);

};

}

Hooks

=====

hook/useThemeSafe.ts (tema global dark/light)

---------------------------------------------

type ThemeHook = () => { theme: 'light' | 'dark' };

// @ts-expect-error: el host puede proveerlo

declare const useTheme: ThemeHook;

const FallbackTheme: ThemeHook = () => ({ theme: 'light' });

export function useThemeSafe() {

try { return (useTheme ?? FallbackTheme)(); } catch { return FallbackTheme(); }

}

hook/useI18nMerge.ts (usa local-first por defecto)

--------------------------------------------------

import { getLocalDict } from '../i18n';

import { makeTranslator, type TranslationOrder } from '../utils';

type LangHook = () => { lang: string; dict?: Record<string, string> };

// @ts-expect-error: el host puede proveerlo

declare const useLanguage: LangHook;

export function useI18nMerge(

langOverride?: string,

opts?: { order?: TranslationOrder } // opcional

) {

let lang = 'en';

let globalDict: Record<string, string> | undefined;

try {

const g = (useLanguage as LangHook)?.();

lang = g?.lang ?? 'en';

globalDict = g?.dict;

} catch { /\* fallback seguro \*/ }

if (langOverride) lang = langOverride;

const local = getLocalDict(lang);

// ⬇️ por defecto ahora es local-first

const t = makeTranslator(local as any, globalDict, opts?.order ?? 'local-first');

return { lang, t };

}

hook/useVisibility.ts

---------------------

import { useEffect, useMemo, useState } from 'react';

import type { VisibilityConfig } from '../types';

import { detectWidth, detectOrientation, detectDevice, isVisibleByConfig } from '../utils';

export function useVisibility(configFromProps?: VisibilityConfig) {

const [width, setWidth] = useState(detectWidth());

const [cfg, setCfg] = useState<VisibilityConfig | undefined>(configFromProps);

useEffect(() => {

if (configFromProps) { setCfg(configFromProps); return; }

let mounted = true;

(async () => {

try {

// @ts-ignore archivo opcional

const mod = await import('../index.visibility');

if (mounted) setCfg(mod.default as VisibilityConfig);

} catch { if (mounted) setCfg(undefined); }

})();

return () => { mounted = false; };

}, [configFromProps]);

useEffect(() => {

const onResize = () => setWidth(detectWidth());

if (typeof window !== 'undefined') {

window.addEventListener('resize', onResize);

return () => window.removeEventListener('resize', onResize);

}

}, []);

const device = detectDevice(width);

const orientation = detectOrientation();

const isVisible = useMemo(() => isVisibleByConfig(cfg, width, device, orientation), [cfg, width, device, orientation]);

return { cfg, width, device, orientation, isVisible };

}

Provider y View

===============

provider/index.tsx

------------------

import React, { createContext, useContext, useMemo } from 'react';

import type { ReactNode } from 'react';

import type { [REEMPLAZA\_CON\_NOMBRE]Context, VisibilityConfig } from '../types';

import { useThemeSafe, useI18nMerge, useVisibility } from '../hook';

const Ctx = createContext<[REEMPLAZA\_CON\_NOMBRE]Context | null>(null);

export function use[REEMPLAZA\_CON\_NOMBRE]Context() {

const ctx = useContext(Ctx);

if (!ctx) throw new Error('[REEMPLAZA\_CON\_NOMBRE]Provider no montado');

return ctx;

}

export const [REEMPLAZA\_CON\_NOMBRE]Provider: React.FC<{

children: ReactNode;

config?: VisibilityConfig;

langOverride?: string;

i18nOrder?: 'global-first' | 'local-first';

}> = ({ children, config, langOverride, i18nOrder = 'local-first' }) => {

const { theme } = useThemeSafe();

const { t } = useI18nMerge(langOverride, { order: i18nOrder });

const { cfg: visibilityConfig, width, device, orientation, isVisible } = useVisibility(config);

const value = useMemo<[REEMPLAZA\_CON\_NOMBRE]Context>(() => ({

theme, t, visibilityConfig, isVisible, device, orientation, width

}), [theme, t, visibilityConfig, isVisible, device, orientation, width]);

return <Ctx.Provider value={value}>{children}</Ctx.Provider>;

};

view/index.tsx

--------------

import React from 'react';

import { use[REEMPLAZA\_CON\_NOMBRE]Context } from '../provider';

import styles, { classes } from '../[REEMPLAZA\_CON\_NOMBRE].module';

export const [REEMPLAZA\_CON\_NOMBRE]View: React.FC<{

className?: string;

titleKey?: string;

}> = ({ className, titleKey = 'title', children }) => {

const { theme, t, isVisible } = use[REEMPLAZA\_CON\_NOMBRE]Context();

return (

<div className={classes(theme, isVisible, className)} aria-hidden={!isVisible}>

<h3>{t(titleKey)}</h3>

{children ?? <p>{t('empty')}</p>}

</div>

);

};

Estilos y Config opcional

=========================

[REEMPLAZA\_CON\_NOMBRE].module.css

---------------------------------

.root { display:block; border-radius:12px; padding:12px; transition:background-color .2s,color .2s; }

.light { background:#fff; color:#111827; border:1px solid #e5e7eb; }

.dark { background:#111827; color:#f9fafb; border:1px solid #374151; }

.hidden { display:none !important; }

[REEMPLAZA\_CON\_NOMBRE].module.ts

--------------------------------

import styles from './[REEMPLAZA\_CON\_NOMBRE].module.css';

export function classes(theme: 'light' | 'dark', visible: boolean, extra?: string) {

const parts = [styles.root, theme === 'dark' ? styles.dark : styles.light];

if (!visible) parts.push(styles.hidden);

if (extra) parts.push(extra);

return parts.join(' ');

}

export default styles;

index.visibility.ts (opcional)

------------------------------

import type { VisibilityConfig } from './types';

const config: VisibilityConfig = {

allowedDevices: ['mobile','tablet'],

allowedOrientations: ['portrait'],

breakpoints: [

{ minWidth: 0, maxWidth: 640, visibility: 'visible' },

{ minWidth: 641, maxWidth: 1024, visibility: 'visible' },

{ minWidth: 1025, maxWidth: Infinity, visibility: 'hidden' }

],

defaultVisibility: 'visible'

};

export default config;

Entrada del componente

======================

index.tsx

---------

import React from 'react';

import type { [REEMPLAZA\_CON\_NOMBRE]Props } from './types';

import { [REEMPLAZA\_CON\_NOMBRE]Provider } from './provider';

import { [REEMPLAZA\_CON\_NOMBRE]View } from './view';

const [REEMPLAZA\_CON\_NOMBRE]: React.FC<[REEMPLAZA\_CON\_NOMBRE]Props> = ({

id, className, style, titleKey, children, config

}) => {

return (

<[REEMPLAZA\_CON\_NOMBRE]Provider>

<[REEMPLAZA\_CON\_NOMBRE]View className={className} titleKey={titleKey}>

<div id={id} style={style}>{children}</div>

</[REEMPLAZA\_CON\_NOMBRE]View>

</[REEMPLAZA\_CON\_NOMBRE]Provider>

);

};

export default [REEMPLAZA\_CON\_NOMBRE];

export \* from './types';

Uso rápido

==========

1) Copia la carpeta `[REEMPLAZA\_CON\_NOMBRE]` en tu proyecto.

2) (Opcional) Crea `index.visibility.ts` con tus reglas.

3) Asegura que la App provea `useTheme()` (dark/light) y `useLanguage()` (lang, dict). Si no, el Provider usa fallbacks seguros (tema light y t(key)=key).

4) Renderiza el componente:

import [REEMPLAZA\_CON\_NOMBRE] from './[REEMPLAZA\_CON\_NOMBRE]';

export default function Demo() {

return <[REEMPLAZA\_CON\_NOMBRE] titleKey="title" />;

}

5) Para i18n \*\*local-first\*\* ya no hay que hacer nada extra; si deseas cambiar el orden:

<[REEMPLAZA\_CON\_NOMBRE]Provider i18nOrder="global-first">...</[REEMPLAZA\_CON\_NOMBRE]Provider>

6) Para tema \*\*dark/light\*\*, basta con que `useTheme()` del host cambie su valor; las clases se aplican automáticamente (`.dark`/`.light`).

Notas finales

=============

- i18n: \*\*local-first\*\* por defecto (componente > app > key).

- Theme: responde automáticamente a `useTheme()` (dark/light).

- Visibilidad: si no defines `index.visibility.ts`, el componente es visible en todos los dispositivos.

- No hay dependencias externas. Todo es React + TS + CSS Modules.