Lobechat

内容手册

目录

[App 3](#_Toc197886147)

[(backend) 3](#_Toc197886148)

[[variants] 3](#_Toc197886149)

[layout 3](#_Toc197886150)

[(auth) 6](#_Toc197886151)

[(main) 6](#_Toc197886152)

[@model 6](#_Toc197886153)

[loading 6](#_Toc197886154)

[oauth 6](#_Toc197886155)

[actions 6](#_Toc197886156)

[desktop 6](#_Toc197886157)

[manifest 6](#_Toc197886158)

[robots 6](#_Toc197886159)

[sitemap 6](#_Toc197886160)

[sw 6](#_Toc197886161)

[Config 6](#_Toc197886162)

[auth 6](#_Toc197886163)

[Const 8](#_Toc197886164)

[auth 8](#_Toc197886165)

[version 9](#_Toc197886166)

[branding 9](#_Toc197886167)

[layoutTokens 10](#_Toc197886168)

[Components 11](#_Toc197886169)

[CircleLoader 11](#_Toc197886170)

[useStyles 11](#_Toc197886171)

[CircleLoader 11](#_Toc197886172)

[Loading 11](#_Toc197886173)

[BrandTextLoading 11](#_Toc197886174)

[CircleLoading 11](#_Toc197886175)

[FullscreenLoading 11](#_Toc197886176)

[SkeletonLoading 11](#_Toc197886177)

[UpdateLoading 11](#_Toc197886178)

[Branding 11](#_Toc197886179)

[OrgBrand 11](#_Toc197886180)

[ProjectLogo 11](#_Toc197886181)

[CustomLogo 11](#_Toc197886182)

[InitProgress 11](#_Toc197886183)

[BubblesLoading 12](#_Toc197886184)

[SafeSpacing 13](#_Toc197886185)

[Layout 14](#_Toc197886186)

[GlobalProvider 14](#_Toc197886187)

[AuthProvider 15](#_Toc197886188)

[Feature 16](#_Toc197886189)

[PWAInstall 16](#_Toc197886190)

[Install 17](#_Toc197886191)

[Modules 19](#_Toc197886192)

[react 19](#_Toc197886193)

[function memo(function) 19](#_Toc197886194)

[any forwardRef(any) 19](#_Toc197886195)

[antd-style 19](#_Toc197886196)

[string cx(ClassNamesArgs[]) 19](#_Toc197886197)

# [App](#_top)

## [(backend)](#_App)

## [[variants]](#_App)

### [layout](#_[variants])

const inVercel = process.env.VERCEL === '1';

interface RootLayoutProps extends DynamicLayoutProps {

  children: ReactNode;

  modal: ReactNode;

}

const RootLayout = async ({ children, params, modal }: RootLayoutProps) => {

  const { variants } = await params;

  const { locale, isMobile, theme, primaryColor, neutralColor } =

    RouteVariants.deserializeVariants(variants);

  const direction = isRtlLang(locale) ? 'rtl' : 'ltr';

  return (

    <html dir={direction} lang={locale} suppressHydrationWarning>

      <body>

        <NuqsAdapter>

          <[GlobalProvider](#_GlobalLayout)

            appearance={theme}

            isMobile={isMobile}

            locale={locale}

            neutralColor={neutralColor}

            primaryColor={primaryColor}

          >

            <[AuthProvider](#_AuthProvider)>

              {children}

              {!isMobile && modal}

            </[AuthProvider](#_AuthProvider)>

            <[PWAInstall](#_PWAInstall) />

          </[GlobalProvider](#_GlobalProvider)>

        </NuqsAdapter>

        <Analytics />

        {inVercel && <SpeedInsights />}

      </body>

    </html>

  );

};

export default RootLayout;

export { generateMetadata } from './metadata';

export const generateViewport = async (props: DynamicLayoutProps): ResolvingViewport => {

  const isMobile = await RouteVariants.getIsMobile(props);

  const dynamicScale = isMobile ? { maximumScale: 1, userScalable: false } : {};

  return {

    ...dynamicScale,

    initialScale: 1,

    minimumScale: 1,

    themeColor: [

      { color: '#f8f8f8', media: '(prefers-color-scheme: light)' },

      { color: '#000', media: '(prefers-color-scheme: dark)' },

    ],

    viewportFit: 'cover',

    width: 'device-width',

  };

};

export const generateStaticParams = () => {

  const themes: ThemeAppearance[] = ['dark', 'light'];

  const mobileOptions = isDesktop ? [false] : [true, false];

  // only static for serveral page, other go to dynamtic

  const staticLocales: Locales[] = [DEFAULT\_LANG, 'zh-CN'];

  const variants: { variants: string }[] = [];

  for (const locale of staticLocales) {

    for (const theme of themes) {

      for (const isMobile of mobileOptions) {

        variants.push({

          variants: RouteVariants.serializeVariants({ isMobile, locale, theme }),

        });

      }

    }

  }

  return variants;

};

### [(auth)](#_[variants])

### [(main)](#_[variants])

### [@model](#_[variants])

### [loading](#_[variants])

### [oauth](#_[variants])

## [actions](#_App)

## [desktop](#_App)

## [manifest](#_App)

## [robots](#_App)

## [sitemap](#_App)

## [sw](#_App)

# [Config](#_top)

## [auth](#_Config)

* 认证配置文件，主要用于管理各种认证提供商的环境变量和配置。
* **支持的认证提供商**：

- Clerk

- NextAuth

- Auth0

- GitHub

- Azure AD

- Authentik

- Authelia

- Cloudflare Zero Trust

- Generic OIDC

- Zitadel

- Logto

- Casdoor

* 环境变量配置：

client: {

  NEXT\_PUBLIC\_CLERK\_PUBLISHABLE\_KEY: z.string().optional(),

  NEXT\_PUBLIC\_ENABLE\_CLERK\_AUTH: z.boolean().optional(),

  NEXT\_PUBLIC\_ENABLE\_NEXT\_AUTH: z.boolean().optional(),

}

// 服务器端环境变量

server: {

  // 各种认证提供商的配置

  CLERK\_SECRET\_KEY: z.string().optional(),

  NEXT\_AUTH\_SECRET: z.string().optional(),

  // ... 其他配置

}

* 组织结构：

// 全局类型声明

declare global {

  namespace NodeJS {

    interface ProcessEnv {

      // 各种环境变量定义

    }

  }

}

// 配置获取函数

export const getAuthConfig = () => {

  // 配置验证和返回

}

// 导出配置

export const authEnv = getAuthConfig();

# [Const](#_top)

## [auth](#_Const)

import { [authEnv](#_auth) } from '[@/config/auth](#_auth)';

export const enableClerk = authEnv.NEXT\_PUBLIC\_ENABLE\_CLERK\_AUTH;

export const enableNextAuth = authEnv.NEXT\_PUBLIC\_ENABLE\_NEXT\_AUTH;

export const enableAuth = enableClerk || enableNextAuth || false;

export const LOBE\_CHAT\_AUTH\_HEADER = …;

export const OAUTH\_AUTHORIZED = …;

export const JWT\_SECRET\_KEY = …;

export const NON\_HTTP\_PREFIX = 'http\_nosafe';

export interface JWTPayload {

  /\*\*

   \* password

   \*/

  accessCode?: string;

  /\*\*

   \* Represents the user's API key

   \*

   \* If provider need multi keys like bedrock,

   \* this will be used as the checker whether to use frontend key

   \*/

  apiKey?: string;

  /\*\*

   \* Represents the endpoint of provider

   \*/

  baseURL?: string;

  azureApiVersion?: string;

  awsAccessKeyId?: string;

  awsRegion?: string;

  awsSecretAccessKey?: string;

  awsSessionToken?: string;

  cloudflareBaseURLOrAccountID?: string;

  /\*\*

   \* user id

   \* in client db mode it's a uuid

   \* in server db mode it's a user id

   \*/

  userId?: string;

}

## [version](#_Const)

import pkg from '@/../package.json';

import { BRANDING\_NAME, ORG\_NAME } from './branding';

export const CURRENT\_VERSION = pkg.version;

export const isServerMode = process.env.NEXT\_PUBLIC\_SERVICE\_MODE === 'server';

export const isUsePgliteDB = process.env.NEXT\_PUBLIC\_CLIENT\_DB === 'pglite';

export const isDesktop = process.env.NEXT\_PUBLIC\_IS\_DESKTOP\_APP === '1';

export const isDeprecatedEdition = !isServerMode && !isUsePgliteDB;

export const isCustomBranding = BRANDING\_NAME !== 'LobeChat';

export const isCustomORG = ORG\_NAME !== 'LobeHub';

## [branding](#_Const)

export const LOBE\_CHAT\_CLOUD = 'LobeChat Cloud';

export const BRANDING\_NAME = …;

export const BRANDING\_LOGO\_URL =…;

export const ORG\_NAME = …;

export const BRANDING\_URL = {

  help: …,

  privacy: …,

  terms: …,

};

## [layoutTokens](#_Const_1)

export const HEADER\_HEIGHT = …;

export const MOBILE\_NABBAR\_HEIGHT = …;

export const MOBILE\_TABBAR\_HEIGHT = …;

export const CHAT\_TEXTAREA\_MAX\_HEIGHT = …;

export const CHAT\_TEXTAREA\_HEIGHT = …;

export const CHAT\_TEXTAREA\_HEIGHT\_MOBILE = …;

export const CHAT\_SIDEBAR\_WIDTH = …;

export const CHAT\_PORTAL\_WIDTH = …;

export const CHAT\_PORTAL\_MAX\_WIDTH = …;

export const CHAT\_PORTAL\_TOOL\_UI\_WIDTH = …;

export const MARKET\_SIDEBAR\_WIDTH = …;

export const FOLDER\_WIDTH = …;

export const MAX\_WIDTH = …;

export const FORM\_STYLE: FormProps = {

  itemMinWidth: 'max(…%,…px)',

  style: { maxWidth: MAX\_WIDTH, width: '100%' },

};

export const MOBILE\_HEADER\_ICON\_SIZE: ActionIconProps['size'] = { blockSize: …, size: … };

export const DESKTOP\_HEADER\_ICON\_SIZE: ActionIconProps['size'] = { blockSize: …, size: … };

export const HEADER\_ICON\_SIZE = (mobile?: boolean) =>

  mobile ? MOBILE\_HEADER\_ICON\_SIZE : DESKTOP\_HEADER\_ICON\_SIZE;

export const PWA\_INSTALL\_ID = 'pwa-install';

# [Components](#_top)

## [CircleLoader](#_Components)

### [useStyles](#_CircleLoader)

export const useStyles = createStyles(({ css, token }, borderWidth: number = 2.5) => ({

  background: css…,

  container: css…,

  loader: css…,

}));

### [CircleLoader](#_CircleLoader)

const CircleLoader = memo(() => {

  const { styles } = useStyles();

  return (

    <div className={styles.container}>

      <div className={styles.loader} />

      <div className={styles.background} />

    </div>

  );

});

export default CircleLoader;

## [Loading](#_Components)

### [BrandTextLoading](#_Loading)

export default () => {

  if ([isCustomBranding](#_version)) return <[CircleLoading](#_CircleLoader) />;

  return (

    <Center height={'100%'} width={'100%'}>

      <BrandLoading size={40} style={{ opacity: 0.6 }} text={LobeHubText} />

    </Center>

  );

};

### [CircleLoading](#_Loading)

export default () => {

  const { t } = useTranslation('common');

  return (

    <Center height={'100%'} width={'100%'}>

      <Flexbox align={'center'} gap={8}>

        <div>

          <Icon icon={LoaderCircle} size={'large'} spin />

        </div>

        <Typography.Text style={{ letterSpacing: '0.1em' }} type={'secondary'}>

          {t('loading')}

        </Typography.Text>

      </Flexbox>

    </Center>

  );

};

垂直水平居中

图标和文字之间有 8px 间距

文字使用次要颜色

文字有 0.1em 的字母间距

### [FullscreenLoading](#_Loading)

interface FullscreenLoadingProps {

  activeStage: number;

  contentRender?: ReactNode;

  stages: StageItem[];

}

const FullscreenLoading = [memo](#_function_memo(function))<FullscreenLoadingProps>(({ activeStage, stages, contentRender }) => {

  return (

    <Flexbox height={'100%'} style={{ position: 'relative', userSelect: 'none' }} width={'100%'}>

      <Center flex={1} gap={16} width={'100%'}>

        <[ProductLogo](#_ProjectLogo) size={48} type={'combine'} />

        {contentRender ? contentRender : <[InitProgress](#_InitProgress) activeStage={activeStage} stages={stages} />}

      </Center>

    </Flexbox>

  );

});

export default FullscreenLoading;

### [SkeletonLoading](#_Loading)

const useStyles = createStyles(({ css, responsive }) => css…);

const SkeletonLoading = [memo](#_function_memo(function))<SkeletonProps>(({ className, ...rest }) => {

  const { cx, styles } = useStyles();

  return <Skeleton active className={cx(styles, className)} paragraph={{ rows: 8 }} {...rest} />;

});

export default SkeletonLoading;

显示占位骨架内容(默认八行)

### [UpdateLoading](#_Loading)

interface UpdateLoadingProps {

  size?: IconSizeType;

  style?: CSSProperties;

}

const UpdateLoading = memo<UpdateLoadingProps>(({ size, style }) => {

  return (

    <div style={style}>

      <Icon icon={Loader2} size={size} spin />

    </div>

  );

});

export default UpdateLoading;

旋转的加载图标

## [Branding](#_Components)

export { OrgBrand } from './OrgBrand';

export { ProductLogo } from './ProductLogo';

### [OrgBrand](#_Branding)

export const OrgBrand = [memo](#_function_memo(function))<LobeHubProps>((props) => {

  if ([isCustomORG](#_version)) {

    return <span>{[ORG\_NAME](#_branding_1)}</span>;

  }

  return <LobeHub {...props} />;

});

### [ProjectLogo](#_Branding)

interface ProductLogoProps extends LobeHubProps {

  height?: number;

  width?: number;

}

export const ProductLogo = memo<ProductLogoProps>((props) => {

  if ([isCustomBranding](#_version)) {

    return <[CustomLogo](#_CustomLogo) {...props} />;

  }

  return <LobeHub {...props} />;

});

### [CustomLogo](#_Branding)

加载[BRANDING\_NAME](#_branding_1)的文本与位于[BRANDING\_LOGO\_URL](#_branding_1)的图片

## [InitProgress](#_Components)

export interface StageObjectItem {

  icon?: ReactNode;

  text: string;

}

export type StageItem = string | StageObjectItem;

interface InitingProps {

  activeStage: number;

  stages: StageItem[];

}

const InitProgress = [memo](#_function_memo(function))<InitingProps>(({ activeStage, stages }) => {

  const theme = useTheme();

  const outStage = stages[activeStage];

  const percent = (activeStage / (stages.length - 1)) \* 100;

  const stage = typeof outStage === 'string' ? { text: outStage } : outStage;

  return (

    <Center gap={8} width={180}>

      <Progress

        percent={parseInt(percent.toFixed(0))}

        showInfo={false}

        strokeColor={theme.colorPrimary}

      />

      <Flexbox align={'center'} gap={4} horizontal>

        {stage?.icon ? stage?.icon : <Icon icon={Loader2} spin />}

        <Typography.Text type={'secondary'}>{stage?.text}</Typography.Text>

      </Flexbox>

    </Center>

  );

});

export default InitProgress;

## [BubblesLoading](#_Components)

const container = css…;

const BubblesLoadingIcon: IconType = forwardRef(

  ({ size = '1em', style, className, ...rest }, ref) => {

    return (

      <svg

        className={cx(container, className)}

        fill="currentColor"

        fillRule="evenodd"

        height={size}

        ref={ref}

        style={{ flex: 'none', lineHeight: 1, ...style }}

        viewBox="0 0 60 32"

        xmlns="http://www.w3.org/2000/svg"

        {...rest}

      >

        <circle cx="7" cy="16" r="6" />

        <circle cx="30" cy="16" r="6" />

        <circle cx="53" cy="16" r="6" />

      </svg>

    );

  },

);

const BubblesLoading = memo(() => {

  const theme = useTheme();

  return (

    <Center style={{ fill: theme.colorTextSecondary, height: 24, width: 32 }}>

      <BubblesLoadingIcon size={14} />

    </Center>

  );

});

export default BubblesLoading;

由缓动曲线控制的渐变动画效果

## [SafeSpacing](#_Components)

interface SafeSpacingProps {

  height?: number;

  mobile?: boolean;

  position?: SafeAreaProps['position'];

}

const SafeSpacing = memo<SafeSpacingProps>(({ height, position = 'top', mobile }) => {

  let h;

  if (mobile) {

    h = position === 'top' ? [MOBILE\_NABBAR\_HEIGHT](#_layoutTokens) : [MOBILE\_TABBAR\_HEIGHT](#_layoutTokens);

  } else {

    h = [HEADER\_HEIGHT](#_layoutTokens);

  }

  return (

    <>

      <div style={{ flex: 'none', height: height || h }} />

      {mobile && <SafeArea position={position} />}

    </>

  );

});

export default SafeSpacing;

# [Layout](#_top)

## [GlobalProvider](#_Layout)

interface GlobalLayoutProps {

  appearance: string;

  children: ReactNode;

  isMobile: boolean;

  locale: string;

  neutralColor?: string;

  primaryColor?: string;

}

const GlobalLayout = async ({

  children,

  neutralColor,

  primaryColor,

  locale: userLocale,

  appearance,

  isMobile,

}: GlobalLayoutProps) => {

  const antdLocale = await getAntdLocale(userLocale);

  // get default feature flags to use with ssr

  const serverFeatureFlags = getServerFeatureFlagsValue();

  const serverConfig = await getServerGlobalConfig();

  return (

    <StyleRegistry>

      <AppTheme

        customFontFamily={appEnv.CUSTOM\_FONT\_FAMILY}

        customFontURL={appEnv.CUSTOM\_FONT\_URL}

        defaultAppearance={appearance}

        defaultNeutralColor={neutralColor as any}

        defaultPrimaryColor={primaryColor as any}

        globalCDN={appEnv.CDN\_USE\_GLOBAL}

      >

        <Locale antdLocale={antdLocale} defaultLang={userLocale}>

          <ServerConfigStoreProvider

            featureFlags={serverFeatureFlags}

            isMobile={isMobile}

            serverConfig={serverConfig}

          >

            <QueryProvider>{children}</QueryProvider>

            <StoreInitialization />

            <Suspense>

              <ImportSettings />

              <ReactScan />

              {process.env.NODE\_ENV === 'development' && <DevPanel />}

            </Suspense>

          </ServerConfigStoreProvider>

        </Locale>

      </AppTheme>

      <AntdV5MonkeyPatch />

    </StyleRegistry>

  );

};

export default GlobalLayout;

## [AuthProvider](#_Layout)

const AuthProvider = ({ children }: PropsWithChildren) => {

  if (authEnv.NEXT\_PUBLIC\_ENABLE\_CLERK\_AUTH) return <Clerk>{children}</Clerk>;

  if (authEnv.NEXT\_PUBLIC\_ENABLE\_NEXT\_AUTH) return <NextAuth>{children}</NextAuth>;

  return <NoAuth>{children}</NoAuth>;

};

export default AuthProvider;

# [Feature](#_top)

## [PWAInstall](#_Feature)

const Install: any = dynamic(() => import('./Install'), {

  ssr: false,

});

const PWAInstall = memo(() => {

  const { isPWA, isSupportInstallPWA } = usePlatform();

  const isShowPWAGuide = useUserStore((s) => s.isShowPWAGuide);

  const hidePWAInstaller = useGlobalStore((s) => systemStatusSelectors.hidePWAInstaller(s));

  const [canInstallFromPWAInstallHandler, setCanInstallFromPWAInstallHandler] = useState<

    boolean | undefined

  >();

  useEffect(() => {

    pwaInstallHandler.addListener((canInstall) => {

      setCanInstallFromPWAInstallHandler(canInstall);

    });

    return () => {

      pwaInstallHandler.removeListener(setCanInstallFromPWAInstallHandler);

    };

  }, []);

  if (

    isPWA ||

    !isShowPWAGuide ||

    !isSupportInstallPWA ||

    hidePWAInstaller ||

    canInstallFromPWAInstallHandler === false

  )

    return null;

  // only when the user is suitable for the pwa install and not install the pwa

  // then show the installation guide

  return <Install />;

});

export default PWAInstall;

### [Install](#_PWAInstall)

const PWAInstall = memo(() => {

  const { t } = useTranslation('metadata');

  const { install, canInstall } = usePWAInstall();

  const isShowPWAGuide = useUserStore((s) => s.isShowPWAGuide);

  const [hidePWAInstaller, updateSystemStatus] = useGlobalStore((s) => [

    systemStatusSelectors.hidePWAInstaller(s),

    s.updateSystemStatus,

  ]);

  // we need to make the pwa installer hidden by default

  useLayoutEffect(() => {

    sessionStorage.setItem('pwa-hide-install', 'true');

  }, []);

  const pwaInstall =

    // eslint-disable-next-line unicorn/prefer-query-selector

    typeof window === 'undefined' ? undefined : document.getElementById(PWA\_INSTALL\_ID);

  // add an event listener to control the user close installer action

  useEffect(() => {

    if (!pwaInstall) return;

    const handler = (e: Event) => {

      const event = e as CustomEvent;

      // it means user hide installer

      if (event.detail.message === 'dismissed') {

        updateSystemStatus({ hidePWAInstaller: true });

      }

    };

    pwaInstall.addEventListener('pwa-user-choice-result-event', handler);

    return () => {

      pwaInstall.removeEventListener('pwa-user-choice-result-event', handler);

    };

  }, [pwaInstall]);

  // trigger the PWA guide on demand

  useEffect(() => {

    if (!canInstall || hidePWAInstaller) return;

    // trigger the pwa installer and register the service worker

    if (isShowPWAGuide) {

      install();

      if ('serviceWorker' in navigator && window.serwist !== undefined) {

        window.serwist.register();

      }

    }

  }, [canInstall, hidePWAInstaller, isShowPWAGuide]);

  return (

    <PWA

      description={t('chat.description', { appName: BRANDING\_NAME })}

      id={PWA\_INSTALL\_ID}

      manifest-url={'/manifest.webmanifest'}

    />

  );

});

export default PWAInstall;

# [Modules](#_top)

## [react](#_Modules)

### [function memo(function)](#_react)

实参记忆化, 减少在相同参数时的渲染

### [any forwardRef(any)](#_react)

前向引用(实例)

## [antd-style](#_Modules)

### [string cx(ClassNamesArgs[])](#_antd-style)

输入是字符串/对象/数组/条件表达式等类型的数组

export type ClassNamesArg = undefined | null | string | boolean | {

    [className: string]: boolean | null | undefined;

} | ArrayClassNamesArg | SerializedStyles;

返回一个可用类名的字符串, 自动过滤掉null、undefined、false等无效值