import type { Placement, ModifierPhases } from "./enums";

import type { PopperOffsetsModifier } from "./modifiers/popperOffsets";

import type { FlipModifier } from "./modifiers/flip";

import type { HideModifier } from "./modifiers/hide";

import type { OffsetModifier } from "./modifiers/offset";

import type { EventListenersModifier } from "./modifiers/eventListeners";

import type { ComputeStylesModifier } from "./modifiers/computeStyles";

import type { ArrowModifier } from "./modifiers/arrow";

import type { PreventOverflowModifier } from "./modifiers/preventOverflow";

import type { ApplyStylesModifier } from "./modifiers/applyStyles";

export declare type Obj = {

[key: string]: any;

};

export declare type VisualViewport = EventTarget & {

width: number;

height: number;

offsetLeft: number;

offsetTop: number;

scale: number;

};

export declare type Window = {

innerHeight: number;

offsetHeight: number;

innerWidth: number;

offsetWidth: number;

pageXOffset: number;

pageYOffset: number;

getComputedStyle: typeof getComputedStyle;

addEventListener(type: any, listener: any, optionsOrUseCapture?: any): void;

removeEventListener(type: any, listener: any, optionsOrUseCapture?: any): void;

Element: Element;

HTMLElement: HTMLElement;

Node: Node;

toString(): "[object Window]";

devicePixelRatio: number;

visualViewport?: VisualViewport;

ShadowRoot: ShadowRoot;

};

export declare type Rect = {

width: number;

height: number;

x: number;

y: number;

};

export declare type Offsets = {

y: number;

x: number;

};

export declare type PositioningStrategy = "absolute" | "fixed";

export declare type StateRects = {

reference: Rect;

popper: Rect;

};

export declare type StateOffsets = {

popper: Offsets;

arrow?: Offsets;

};

declare type OffsetData = {

[key in Placement]?: Offsets;

};

export declare type State = {

elements: {

reference: Element | VirtualElement;

popper: HTMLElement;

arrow?: HTMLElement;

};

options: OptionsGeneric<any>;

placement: Placement;

strategy: PositioningStrategy;

orderedModifiers: Array<Modifier<any, any>>;

rects: StateRects;

scrollParents: {

reference: Array<Element | Window | VisualViewport>;

popper: Array<Element | Window | VisualViewport>;

};

styles: {

[key: string]: Partial<CSSStyleDeclaration>;

};

attributes: {

[key: string]: {

[key: string]: string | boolean;

};

};

modifiersData: {

arrow?: {

x?: number;

y?: number;

centerOffset: number;

};

hide?: {

isReferenceHidden: boolean;

hasPopperEscaped: boolean;

referenceClippingOffsets: SideObject;

popperEscapeOffsets: SideObject;

};

offset?: OffsetData;

preventOverflow?: Offsets;

popperOffsets?: Offsets;

[key: string]: any;

};

reset: boolean;

};

declare type SetAction<S> = S | ((prev: S) => S);

export declare type Instance = {

state: State;

destroy: () => void;

forceUpdate: () => void;

update: () => Promise<Partial<State>>;

setOptions: (setOptionsAction: SetAction<Partial<OptionsGeneric<any>>>) => Promise<Partial<State>>;

};

export declare type ModifierArguments<Options extends Obj> = {

state: State;

instance: Instance;

options: Partial<Options>;

name: string;

};

export declare type Modifier<Name, Options extends Obj> = {

name: Name;

enabled: boolean;

phase: ModifierPhases;

requires?: Array<string>;

requiresIfExists?: Array<string>;

fn: (arg0: ModifierArguments<Options>) => State | void;

effect?: (arg0: ModifierArguments<Options>) => (() => void) | void;

options?: Partial<Options>;

data?: Obj;

};

export declare type StrictModifiers = Partial<OffsetModifier> | Partial<ApplyStylesModifier> | Partial<ArrowModifier> | Partial<HideModifier> | Partial<ComputeStylesModifier> | Partial<EventListenersModifier> | Partial<FlipModifier> | Partial<PreventOverflowModifier> | Partial<PopperOffsetsModifier>;

export declare type EventListeners = {

scroll: boolean;

resize: boolean;

};

export declare type Options = {

placement: Placement;

modifiers: Array<Partial<Modifier<any, any>>>;

strategy: PositioningStrategy;

onFirstUpdate?: (arg0: Partial<State>) => void;

};

export declare type OptionsGeneric<TModifier> = {

placement: Placement;

modifiers: Array<TModifier>;

strategy: PositioningStrategy;

onFirstUpdate?: (arg0: Partial<State>) => void;

};

export declare type UpdateCallback = (arg0: State) => void;

export declare type ClientRectObject = {

x: number;

y: number;

top: number;

left: number;

right: number;

bottom: number;

width: number;

height: number;

};

export declare type SideObject = {

top: number;

left: number;

right: number;

bottom: number;

};

export declare type Padding = number | Partial<SideObject>;

export declare type VirtualElement = {

getBoundingClientRect: () => ClientRect | DOMRect;

contextElement?: Element;

};

export {};