ВСТРОЕННЫЕ ВСПОМОГАТЕЛЬНЫЕ ТИПЫ

Вспомогательные типы используются для самых разных целей при формировании новых типов. Это раздел **Utility types** в документации с полным их перечнем

Тип Omit необходим для того, чтобы **исключить указанные свойства** из другого типа. Второй аргумент может быть только **string | number | symbol**

```ts
interface Currencies {
    usa: "usd";
    china: "cny";
    ukraine: "uah";
    kz: "tenge";
}

type CurrWithoutUSA = Omit<Currencies, "usa">; // исключение
```

Тип Pick необходим для **фильтрации типа по заданным свойствам.** Остаются только **указанные**. Указывать можно только существующие в целевом типе. Если их несколько - то в union типе:

```ts
type CurrUSAAndUkraine = Pick<Currencies, "usa" | "ukraine">; // фильтрация по свойству
```

Тип Exclude позволяет **убирать из union типа** те, которые соответствуют условию:

```ts
type CountriesWithoutUSA = Exclude<keyof Currencies, "usa">; // "china" | "ukraine" | "kz"

type MyAnimation = "fade" | "swipe";
type FadeType = Exclude<MyAnimation, "swipe">; // удаление из union type
```

Тип Extract **выбирает типы**, которые соответствуют условию. Это Exclude наоборот:

```ts
type MyAnimation = "fade" | "swipe";
type SwipeType = Extract<MyAnimation, "swipe">; // выбор подходящего типа, "swipe"
```

Тип Record позволяет **сконструировать другой тип в формате ключ-значение.** Это удобный способ сказать TS: тут будет объект, ключами которого будет это, значениями – это:

```ts
type PlayersNames = "alex" | "john";
type CustomCurrencies = CreateCustomCurr<Currencies>;
type GameDataCurr = Record<PlayersNames, CustomCurrencies>;

const gameData: GameDataCurr = {
    alex: {
        customChina: "test",
        customKz: "test",
        customUkraine: "test",
        customUsa: "test",
    },
    john: {
        customChina: "test",
        customKz: "test",
        customUkraine: "test",
        customUsa: "test",
    },
};
```