Structs

C Reference

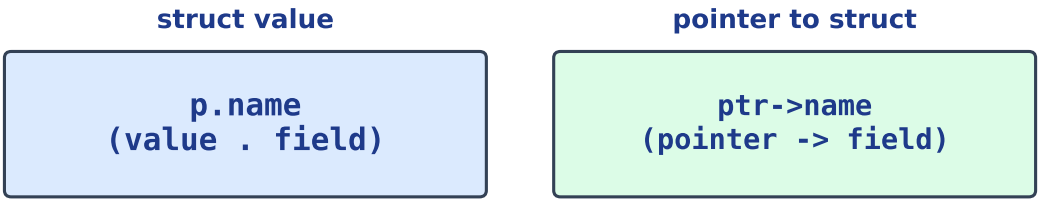
struct, typedef, . vs ->

A struct 结构体 groups related values into one type. Each value is a member 成员. `typedef` gives the struct a short name so you can write `Dog` instead of `struct Dog`. Use `.` on a struct value, but `->` on a pointer to a struct.

```c
#include <stdio.h>

typedef struct {
    char name[20];
    int age;
} Dog;

int main(void) {
    Dog d = {"Rex", 3};
    printf("%s is %d\n", d.name, d.age);   // Rex is 3   (. on a value)

    Dog *p = &d;
    p->age = 4;                            // -> on a pointer
    printf("%s is %d\n", d.name, d.age);   // Rex is 4
    return 0;
}
```

Common mistakes

- Use `.` on a struct value and `->` on a struct pointer.
- `typedef` lets you drop the word `struct` when declaring; without it you write `struct Point p;`.
- Assigning one struct to another copies all its fields.



Exam tip: use . on a struct; use -> on a pointer (same as (*ptr).field)

. for struct values; -> for pointers to structs