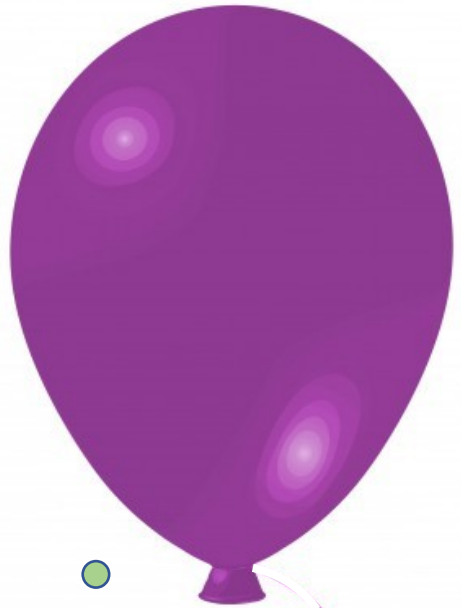
Everything you need to know about pointers you already learned as a kid

What's an Object?

A Balloon is an object, right?

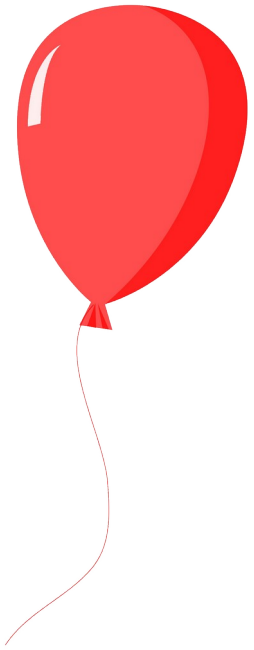


- Something that requires memory and takes up space
- You can touch and feel it
- It has state (color, size, weight, texture, smell, high score)
- It has lifespan (it's born, learns, parties, and dies)
- It has a type (a balloon, automobile)
- It has identity (a name so we can distinguish it from all others)

- Takes up space
- There's stuff inside the balloon's skin
- `Balloon myBalloon (9", purple)`

I ‘tought I saw a balloon

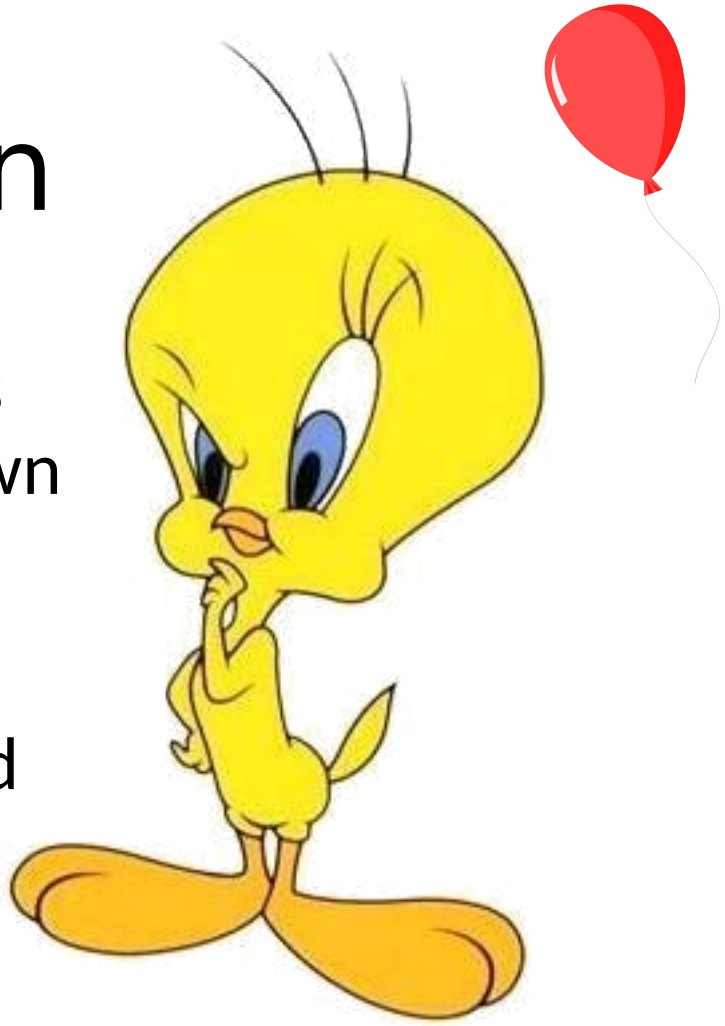
- Let’s fill our balloon with helium
- Ooops, where did it go?
- I know I had one but it must have floated away
- I’m going to blow up another one but this time I’m going to tie a string on it



Rats! It floated away again

- It’s not enough to tie a string to the balloon, I guess I must tie the other end to something to hold it down

- Okay, I get it. This time I’m going to blow up a new balloon, tie a string on it, and then tie the other end of the string to a light-weight ring so the balloon doesn’t float away again.



- Hey Now I have a balloon!!
- Wait, what?
- Okay, I don’t really have a balloon I have a ring with a string tied to it that gets me to the balloon.

A Pointer is an object too!

- It takes up space
- You can touch and feel it
- It has value
- It has lifespan
- It has a type

The metaphor

- The balloon is my dynamically allocated object
- The ring is my pointer
- And the string is the value of my pointer because following the string gets me to my balloon

```
void f() {
        Balloon * myBalloon =
nullptr;
}
```

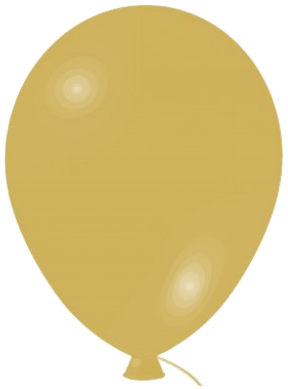
```
void f() {
        Balloon * myBalloon = new
Balloon;
}
```

An unnamed Balloon

The address of the balloon as retuned by operator new

myBalloon

```
void f() {
        Balloon * myBalloon = new
Balloon;
}
```



After the function returns

But the unnamed Balloon still lives
MEMORY Leak

The address is lost

The pointer dies

```
void f() {
        Balloon * myBalloon = new
Balloon;
        delete myBalloon;
}
```

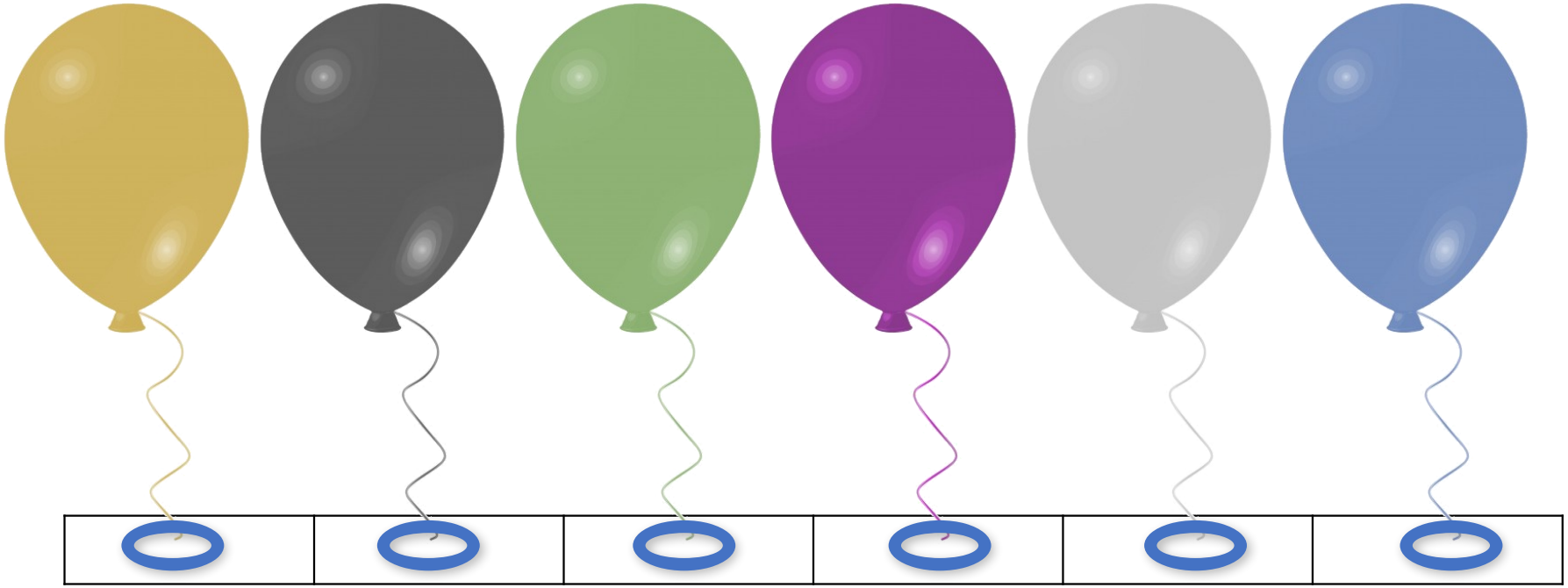
An unnamed Balloon dies

The address of the balloon remains
DANGLING Pointer

myBalloon lives

An array of pointer-to-balloon

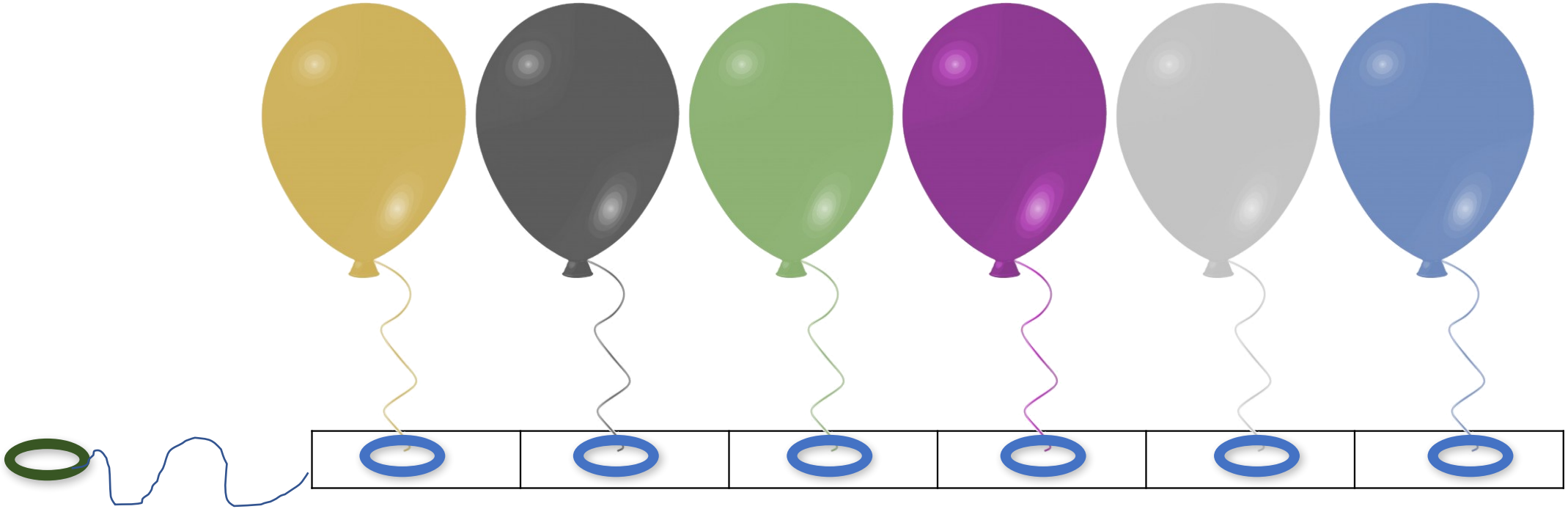
```
Balloon * myCollection[6]; // avoid C-Style arrays
std::array<Balloon*, 6> myCollection
```



myCollection is an array of 6 pointer-to-Balloon

A dynamically allocated array of dynamically allocated balloons

```
Balloon ** myCollection = new Balloon*[6];
std::array<Balloon*, 6> * myCollection =
                          new std::array<Balloon*, 6>
```



myCollection is a pointer to
an array of 6 pointer-to-Balloon

Shorter alternative:

```
auto myCollection = new Balloon*[6];
auto myCollection = new std::array<Balloon*, 6>
```