GOALS OF THIS TALK

- Annoy you
- Teach you something
- But mostly annoy you

AGGREGATE TYPES

- A (built-in) *array type*
- A *class type* without any:
  - user-declared or inherited constructors
  - non-`public` data members or base classes
  - `virtual` base classes or member functions

```cpp
struct person
{
    std::string name;
    int age;
};
```

```cpp
struct couple
{
    person a;
    person b;
};
```

```cpp
person p0{"Alice", 35};
person p1{"Bob", 33};
couple c0{p0, p1};
```

```cpp
couple c1{{"Alice", 35}, {"Bob", 33}};
```

```cpp
couple c2{"Alice", 35, "Bob", 33}; // brace elision
```

```cpp
person p2{.name{"Eve"}, .age{30}}; // designated initializers
                                   // (since C++20)
```

```cpp
person p3("Carl", 60); // round-parentheses syntax
                       // (since C++20)
```

P0960R3

- *“Allow aggregate-init from a parenthesized list of values”*
  - no brace elision
  - no lifetime extension
  - no order of evaluation

- Why?
  - `emplace_back` did not work with aggregates

```
template <typename T>
template <typename... Args>
void std::vector<T>::emplace_back(Args&&... args)
{
    ::new(endptr) T(std::forward<Args>(args)...)
//                  ^                          ^
//                   \__________________________/
//                              round parentheses
}
```

```cpp
struct person
{
    std::string name;
    int age;
};
```

```cpp
#include <vector>



std::vector<person> v0;
```

```cpp
v0.push_back(person{"Alice", 35}); // ✔ OK
v0.push_back(       {"Alice", 35}); // ✔ OK
v0.push_back(        "Alice", 35 ); // ✘ compile-time error
```

```cpp
v0.emplace_back(person{"Alice", 35}); // ✔ OK, but bad idea
v0.emplace_back(       {"Alice", 35}); // ✘ compile-time error
v0.emplace_back(        "Alice", 35 ); // ✔ OK (since C++20)
```

```cpp
std::vector<std::array<int, 3>> v1;
```

```cpp
   v1.push_back(std::array{0, 1, 2}); // ✔ OK
   v1.push_back(           {0, 1, 2}); // ✔ OK
   v1.push_back(            0, 1, 2 ); // ✘ compile-time error
```

```cpp
v1.emplace_back(std::array{0, 1, 2}); // ✔ OK
v1.emplace_back(           {0, 1, 2}); // ✘ compile-time error
v1.emplace_back(            0, 1, 2 ); // ✘ compile-time error
```

```cpp
template <typename T, std::size_t N>
struct std::array
{
    T __data[N];
};
```

- Brace elision:

```cpp
std::array<int, 3>{0, 1, 2}
// ...is syntactic sugar for...
std::array<int, 3>{{0, 1, 2}}
```

- Doesn’t apply to C++20’s parens syntax
- There’s no way to emplace a `std::array`!

- Takeways:
  - **groan**
  - aggregate types
  - aggregate initialization
  - c++20 designated initializers
  - c++20 parentheses syntax
  - `emplace_back`
  - brace elision
- Thanks!
  - github.com/vittorioromeo/accu2023
  - mail@vittorioromeo.com | @supahvee1234
  - emcpps.com - Embracing Modern C++ Safely