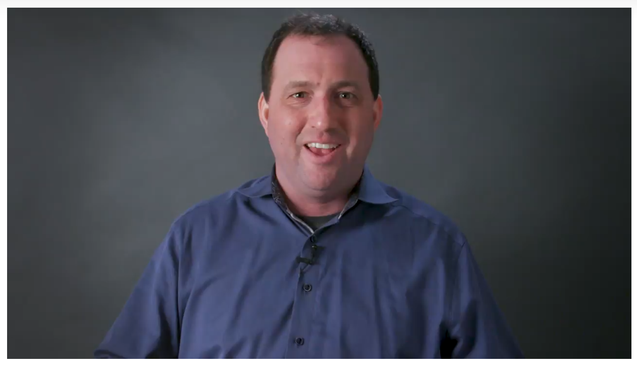

https://youtu.be/zw5fqCmxD8o

Constructors

Constructors are member functions of a class or struct that initialize an object. The Core Guidelines define a constructor) as:

> *constructor*: an operation that initializes ("constructs") an object. Typically a constructor establishes an invariant and often acquires resources needed for an object to be used (which are then typically released by a destructor).

A constructor can take arguments, which can be used to assign values to member variables.

```
class Date {
 public:
  Date(int d, int m, int y) {  // This is a constructor.
    Day(d);
  }
  int Day() { return day; }
  void Day(int d) {
    if (d >= 1 && d <= 31) day = d;
  }
  int Month() { return month; }
  void Month(int m) {
    if (m >= 1 && m <= 12) month = m;
  }
  int Year() { return year_; }
  void Year(int y) { year = y; }

 private:
  int day{1};
  int month{1};
  int year{0};
};
```

As you can see, a constructor is also able to call other member functions of the object it is constructing. In the example above, `Date(int d, int m, int y)` assigns a member variable by calling `Day(int d)`.

Saving Graffiti Recording. Please wait...

↑ Menu Expand

Default Constructor

A class object is always initialized by calling a constructor. That might lead you to wonder how it is possible to initialize a class or structure that does not define any constructor at all.

For example:

```
class Date {
  int day{1};
  int month{1};
  int year{0};
};
```

We can initialize an object of this class, even though this class does not explicitly define a constructor.

This is possible because of the **default constructor**. **The compiler will define a default constructor**, which accepts no arguments, for any class or structure that does not contain an explicitly-defined constructor.