Namespaces

```
/*
      Topics:
            . Creating namespaces
            . Namespaces across multiple files
                  . cylinder, line and point
            . Default global namespace
            . Built in namespaces
            . Using declarations
            . Anonymous namespaces
            . Nested namespaces.
            . Namespace aliases

*/
```

Namespaces: Creating

```
namespace No_Adjust {
    export double add(double x, double y) { return x + y; }
}// namespace No_Adjust

namespace Adjust {
    export double add(double x, double y) { return x + y - adjustment; }
}// namespace Adjust


//Declarations
namespace No_Adjust {
    export double mult(double x, double y);// Declarations
    export double div(double x, double y);

}// namespace No_Adjust

namespace Adjust {
    export double mult(double x, double y);// Declarations
    export double div(double x, double y);
}// namespace Adjust
```

Namespaces: Creating

```
//Definitions
namespace No_Adjust {
    double mult(double x, double y) { return x * y; }
    double div(double x, double y) { return x / y; }

}// namespace No_Adjust

namespace Adjust {
    double mult(double x, double y) { return x * y - adjustment; }
    double div(double x, double y) { return x / y - adjustment; }
}// namespace Adjust
```

Namespaces: Multiple files

```
export class Point
{
  public:
    Point();
    Point(double x, double y);
    void print_info() const { fmt::println("Point [ x: {}, y: {}]", m_x, m_y); }
    ~Point();

  private:
    double m_x;
    double m_y;
};

}// namespace Geom


//Implementations
namespace Geom {

Point::Point(double x, double y) : m_x{ x }, m_y{ y } {}

Point::Point() : Point(0.0, 0.0) {}

Point::~Point() {}

}// namespace Geom
```

Namespaces: Default global namespace

```
// Global namespace
export double add(double a, double b) { return a + b; }

namespace My_Thing {

  export double add(double a, double b) { return a + b - 1; }

  export void do_something()
  {
    double result = ::add(5, 6);
    fmt::println("result : {}", result);
  }

}// namespace My_Thing
```

Namespaces: Built in namespaces

```
// Built in namespaces
std::string
```

Namespaces: Using declarations

```cpp
using namespace Geom;
Point p1(10, 20);
Point p2(3.4, 6.1);
p1.print_info();
p2.print_info();

fmt::println("---");

Cylinder c1(1.4, 10);
fmt::println("c1.volume : {}", c1.volume());

fmt::println("---");

// using std::cout;
// using std::endl;
//using namespace std;// NOT RECOMMENDED!
//string msg("Hello World!");
//fmt::println("Gretting: {}",msg);
```

Anonymous namespaces

```
namespace {
    //Subtract can't be exported and used in external modules because it is constrained
    // to the current translation unit.
    /*
    export double subtract(double a, double b)
    {// Only usable in this translation unit
        return a - b;
    }
    */
}// namespace
```

Nested namespaces

```
namespace Hello {
    unsigned int age{ 23 };

    namespace World {
        int local_var{ 44 };

        export void say_something()
        {
            fmt::println("Hello there ");
            fmt::println("The age is : {}", age);
        }
    }// namespace World

    export void do_something()
    {
        // Here we don't have direct access to local_var, we have to go
        // through the namespace.
        fmt::println("Using local_var : {}", World::local_var);
    }
}// namespace Hello
```

Namespace aliases

```
namespace Level1 {
namespace Level2 {
  namespace Level3 {
    export const double weight{ 33.33 };
  }
}// namespace Level2
}// namespace Level1
```

```
// Namespace aliases
namespace Data = Level1::Level2::Level3;
fmt::println("weight : {}", Data::weight);
```