Fixes

Let's get started with the major fixes and improvements in C++17.

WE'LL COVER THE FOLLOWING

We can argue what is a fix in a language standard and what is not.

Ahead, there are three things that might look like a fix for something that was missing or not working in the previous rules.

New auto Rules For Direct-list-initialisation

Since `C++11`, there's been a strange problem where

```
auto x = { 1 };
```

is deduced as `std::initializer_list<int>`. Such behaviour is not intuitive as in most cases you should expect it to work like `int x { 1 };`.

Brace initialization is the preferred pattern in modern C++, but such exceptions make the feature weaker.

With the new Standard, we can fix this so that it will deduce `int`. To make this happen, we need to understand two ways of initialization - **copy** and **direct**:

```
// foo() is a function that returns some Type by value
auto x = foo();   // copy-initialisation
auto x{foo()};    // direct-initialisation, initializes an
                  // initializer_list (until C++17)

int x = foo();    // copy-initialisation
int x{foo()};     // direct-initialisation
```

For the direct initialization, C++17 introduces new rules:

- For a braced-init-list with a single element, auto deduction will deduce from that entry.
- For a braced-init-list with more than one element, auto deduction will be ill-formed.

For example, lines 7 and 8 will cause errors in the code below. Highlighting them out will allow the code to compile successfully.

```
#include <iostream>
using namespace std;

int main(){

  auto x1 = { 1, 2 }; // decltype(x1) is std::initializer_list<int>
  auto x2 = { 1, 2.0 }; // error: cannot deduce element type
  auto x3{ 1, 2 }; // error: not a single element
  auto x4 = { 3 }; // decltype(x4) is std::initializer_list<int>
  auto x5{ 3 }; // decltype(x5) is int

}
```

> *Extra Info:* The change was proposed in: N3922 and N3681. The compilers fixed this issue quite early, as the improvement is available in GCC 5.0 (Mid 2015), Clang 3.8 (Early 2016) and MSVC 2015 (Mid 2015). Much earlier than C++17 was approved.

Another improvement in the language is related to the `static_assert` method. Find out more in the next lesson.