Template functions #1

Richel Bilderbeek

What, why, mastery

- Write generic functions
- Write less functions, less debugging, say more at compile time
- The data type of your function is just a detail

Examples

```
int abs(const int n) noexcept {
   return n < 0 ? -n : n;
}

double abs(const double n) noexcept
{
   return n < 0 ? -n : n;
}
```

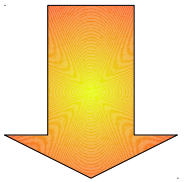
Examples

```
template <typename T>
T abs(const T& n) noexcept
{
  return n < 0 ? -n : n;
}
```

File management

- Function definitions must be in header file

```
//myheader.h
int abs(const int x) noexcept;
int abs(const double x) noexcept;
```



```
//myheader.h
template <typename T>
abs(const T& x) noexcept {
  return x < 0 ? -x : x;
}
```

Note

- Template functions will always be checked for syntax errors
- Template function will be properly compiled when used

```
//myheader.h
template <typename T>
abs(const T& x) noexcept {
  return x.get() < 0 ? -x.get() : x.get();
}
```

Another example

```
template <int n>
void do() noexcept {
  for (int i=0; i!=n; ++i) {
    //Do something
  }
}
```