std::bind and std::function

Programmers can use this pair of utilities to create and bind functions to variables.

The two functions `std::bind` and `std::functions` fit very well together. While `std::bind` enables you to create new function objects on the fly, `std::function` takes these temporary function objects and binds them to a variable. Both functions are powerful tools from functional programming and need the header `<functional>`.

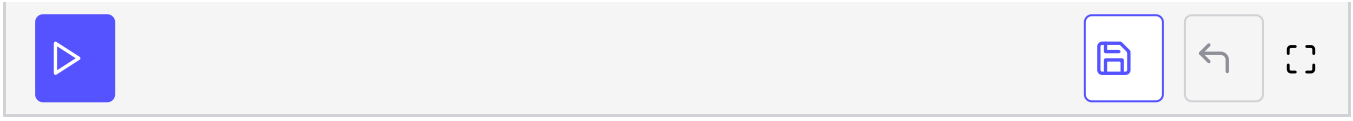
```
#include <iostream>
#include <functional>

// for placehoder _1 and _2
using namespace std::placeholders;

using std::bind;
using std::function;

double divMe(double a, double b){ return a/b; };

int main(){
  function < double(double, double) > myDiv1= bind(divMe, _1, _2);
  function < double(double) > myDiv2= bind(divMe, 2000, _1);
  std::cout << (divMe(2000, 10) == myDiv1(2000, 10) == myDiv2(10));
}
```



Creating and binding function objects

> **std::bind and std::function are mostly superfluous**
> `std::bind` and `std::function`, which where part of TR1, are mostly not necessary any more with C++11. You can use lambda functions instead of `std::bind` and most often you can use the automatic type deduction instead of `std::function`.

Now, let's discuss in detail the behavior of `std::bind` and `std::function`.