- Exercise

In this exercise, you will make a variadic template from the perfect factory method.

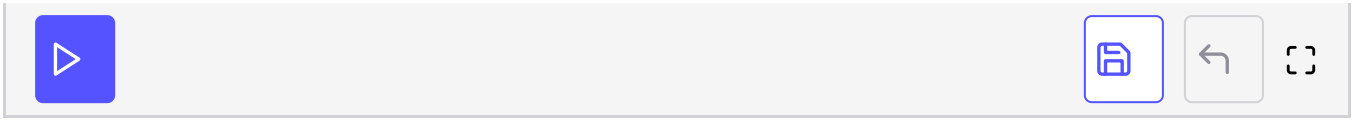
WE'LL COVER THE FOLLOWING

Exercise

Use the perfect factory method in the code below and make a variadic template to use it with different types.

```
#include <iostream>
#include <string>
#include <utility>

template <typename T, typename T1>
T create(T1&& t1){
  return T(std::forward<T1>(t1));
}

int main(){

  std::cout << std::endl;

  // Lvalues
  int five=5;
  int myFive= create<int>(five);
  std::cout << "myFive: "  << myFive << std::endl;

  std::string str{"Lvalue"};
  std::string str2= create<std::string>(str);
  std::cout << "str2: " << str2 << std::endl;

  // Rvalues
  int myFive2= create<int>(5);
  std::cout << "myFive2: " << myFive2 << std::endl;

  std::string str3= create<std::string>(std::string("Rvalue"));
  std::cout << "str3: " << str3 << std::endl;

  std::string str4= create<std::string>(std::move(str3));
  std::cout << "str4: " << str4 << std::endl;

  std::cout << std::endl;
```

```
};
```



The solution to this exercise is in the next lesson.