Item 24) Distinguish universal references from rvalue references

---

```
void f(Widget&& param); // rvalue reference
Widget&& var1 = Widget(); // rvalue reference
auto&& var2 = var1; // not rvalue reference

template<typename T>
void f(std::vector<T>&& param); // rvalue reference

template<typename T>
void f(T&& param); // not rvalue reference (universal reference)
```

`T&&` 라는 코드는 항상 rvalue 참조를 의미하지 않는다.

그렇다면 언제 유니버설 참조가 되는건가요??

1. **타입 추론이 일어나야 한다** : 템플릿 함수의 매개변수 또는 `auto&&` 와 같이 타입이 추론되어야 한다.
2. **정확한 형태여야 함** : 참조 선언이 정확이 `T&&` 의 형태여야 한다. 만약 `const T&&` 또는 `std::vector<T>&&` 와 같은 형태가 되면 유니버설 참조가 되지 않는다.

유니버설 참조가 사용되는 대표적인 예시 : `std::vector` 의 `emplace_back`

```
template<class... Args>
void emplace_back(Args&&... args);
```

- `emplace_back` 의 `args` 는 유니버설 참조이다. `args` 가 호출될 때마다 `Args` 가 추론되며, `args` 는 lvalue와 rvalue 모두에 바인딩될 수 있다.
- 반면, `push_back` 은 유니버설 참조가 아니며, 명확하게 rvalue 참조로 정의되어 있다.