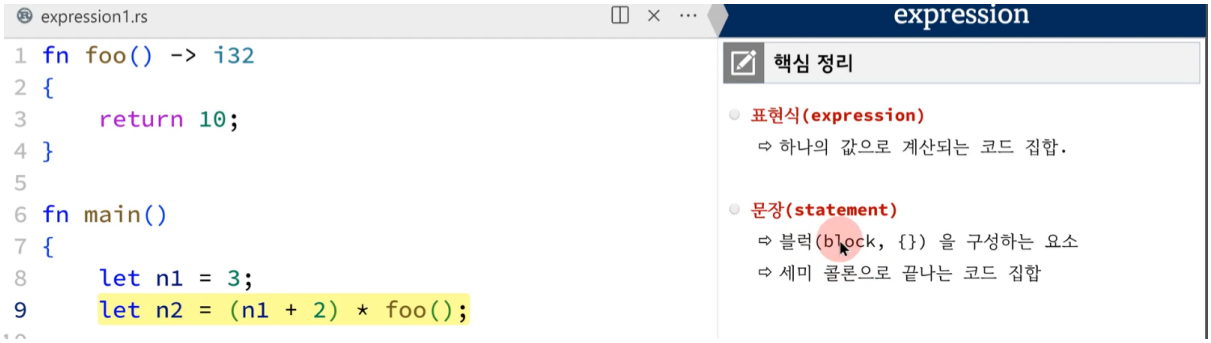
06. expression



표현식

- (n1) ← 표현식
- (n1 + 2) ← 표현식
- (n1 + 2) * foo() ← 표현식

문장

- 선언문 (Declaration Statement)
  - let n1 =3; ← 선언문
  - let n2 = (n1 + 2) * foo(); ← 선언문
- 표현문? (Expression Statement)
  - foo(); ← 표현문
  - let n3 = {3}; 에서 {3} ← 표현식, 표현식이라 3 반환
  - let n3 = {3;}; 에서 {3;} ← 표현문, 표현문이라 Void() 반환

expression1.rs

```rust
fn foo() -> i32
{
    return 10;
}

fn main()
{
    let n1 = 3;
    let n2 = (n1 + 2) * foo();

    foo();

    let n3 = {3};
    let n4 = {3;};

    println!("{:?}", n3);
    println!("{:?}", n4);
}
```

expression

핵심 정리

- Rust 에서 statement 의 종류
  - ⇨ declaration statement
  - ⇨ expression statement

- **Declaration Statement**
  - ⇨ 블럭 안에서 사용할 한 개 이상의 이름을 알리는 문장

- **Expression Statement**
  - ⇨ 표현식 뒤에 ; 을 붙인 것.
  - ⇨ 표현식을 계산하고 그 결과를 무시.

expression1.rs

```rust
fn foo() -> i32
{
    return 10;
}

fn main()
{
    let n1 = 3;
    let n2 = (n1 + 2) * foo();

    foo();

    let n3 = {3};
    let n4 = {3;};

    println!("{:?}", n3); // 3
    println!("{:?}", n4); // Void ()
}
```

expression

핵심 정리

expression	하나의 값을 생성.
expression statement	표현식을 계산하고 그 결과를 무시(Void 값)

블럭을 사용하지 않은 표현식	블럭을 사용하는 표현식
LiteralExpression	**BlockExpression**
PathExpression	UnsafeBlockExpression
OperatorExpression	**LoopExpression**
GroupedExpression	**IfExpression**
ArrayExpression	IfLetExpression
AwaitExpression	**MatchExpression**
IndexExpression	
TupleExpression	
TupleIndexingExpression	
StructExpression	
CallExpression	
MethodCallExpression	
FieldExpression	
ClosureExpression	
AsyncBlockExpression	
ContinueExpression	
BreakExpression	
RangeExpression	
ReturnExpression	
UnderscoreExpression	
MacroInvocation	

expression2.rs

```rust
fn f1() -> i32
{
//    return 10;
      10          // 위와 동일
//    10;
}

fn main()
{
    let score = 30;

    let total = { let report = 30;
                  score + report };

    println!("{:?}", total);

    println!("{:?}", f1() );
}

```

expression

```
rustc expression2.rs -A dead_code -A unused_variables -A unused_assignments -o out.exe
error[E0308]: mismatched types
 --> expression2.rs:1:12
  |
1 | fn f1() -> i32
  |    --      ^^^ expected `i32`, found `()`
  |    |
  |    implicitly returns `()` as its body has no tail or `return` expression
...
5 |     10;
  |       - help: remove this semicolon to return this value

error: aborting due to previous error

For more information about this error, try `rustc --explain E0308`.
[ Error !!, rustc fail to compile. ]

>
```

expression3.rs

```rust
fn main()
{
    let score = 80;

//  let total = if score > 60 { 5 } else { 0 };

//  let total = if score > 60 { 5; } else { 0; }

//  let total = if score > 60 { 5; } else { 0 };

    let total = if score > 60 { 5 };

    println!("{:?}", total);
}

```

expression

```
rustc expression3.rs -A dead_code -A unused_variables -A unused_assignments -o out.exe
error[E0317]: `if` may be missing an `else` clause
  --> expression3.rs:11:14
   |
11 |     let total = if score > 60 { 5 };
   |                 ^^^^^^^^^^^^^^^^^^^
   |                 |              |
   |                 |              found here
   |                 expected integer, found `()`
   |
   = note: `if` expressions without `else` evaluate to `()`
   = help: consider adding an `else` block that evaluates to the expected type

error: aborting due to previous error

For more information about this error, try `rustc --explain E0317`.
[ Error !!, rustc fail to compile. ]

>
```

let total = if score > 60 { 5 }; 의 경우 if 표현식에 else 가 없으면 디폴트로 else 일 경우 Void()를 넣어주는데 타입이 달라서 에러

expression4.rs

```rust
fn main()
{
    let score = 85;

    let total = match score
                {
                    90..=100 => 'A',
                    80..=89 => 'B',
                    70..=79 => 'C',
                    60..=69 => 'D',
                    _ => 'E'
                };

    println!("{:?}", total);
}
```