Const

This lesson will explain what the const keyword does.

WE'LL COVER THE FOLLOWING

Definition

The `const` keyword affects the behavior of a variable. Any variable initialized with the `const` keyword cannot be modified later on. It is *constant*.

```
const int i = 10;
i = 20; // "Error: Cannot modify a constant variable"
```

Variables or attributes of a class that have been made using `const` must always be initialized.

Class methods can also be `const` but they can only be invoked by `const` instances of the class.

Pointers

Both the pointer and the data being pointed to can be `const`.

type const* / const type*

This declaration implies that the value being pointed to is `const`. It should not be altered. However, the pointer itself is not `const`:

```
int i = 2011;
int const* ip = &i;
*ip = 2012; // ERROR

int j = 2012;
ip = &j; // ERROR: pointer being non-const
```

type* const

In this case, the pointer is constant. It cannot point to a different pointer throughout its lifetime.

```
int i = 2011;
int j = 2012;
int* const p = &i;
p = &j; // ERROR

*p = 2015; // ERROR: pointer being non-const
```

const type* const

Now, both the pointer and the value are constant.

```
int i = 100, int j = 200;
const int* const p = &i;
*p = i; // ERROR
p = &j; // ERROR
```

The line, `const int* const p`, should be read from right to left. `p` is a constant pointer, `* const` points to an `int` that is `const`. We can modify `i` directly, but we can't modify it through `p`.

In the next lesson, we will study **constant expressions**.