

1일 (목)

1. 깃, 깃허브 기초 (생활코딩 github.com, 지옥에서 온 git)

폴더 안에 모든 파일을 add 하고 싶을 때 명령어

```
git add ./June_2021/
```

```
#! [rejected]          master -> master (fetch first)
#error: failed to push some refs to 'https://github.com/kimbigmin/time_operator.git'
```

```
git push --force origin master
```

2. 코드카데미 자바스크립트 Functions

- **Helper Function** ⇒ return 안에 다른 함수를 넣어서 간결하게 표현할 수 있다. 이를 헬퍼 펑션이라고 한다.
- **Function Expressions** ⇒ 함수 표현식(=익명 함수) 는 호이스팅이 안된다. 무조건 함수를 선언하고 함수를 호출해야한다.!!
- **Arrow Functions** 함수명(const로 처리) () ⇒ { } 익명함수에서 좀 더 발전한 형태 사용

```
const rectangleArea = (width, height) => {  
  let area = width * height;  
  return area;  
};
```

- 더 간결한 Arrow Function

ZERO PARAMETERS

```
const functionName = () => {};
```

ONE PARAMETER

```
const functionName = paramOne => {};
```

TWO OR MORE PARAMETERS

```
const functionName = (paramOne, paramTwo) => {};
```

1개의 파라미터는 () 를 없앨 수 있다.

SINGLE-LINE BLOCK

```
const sumNumbers = number => number + number;
```

MULTI-LINE BLOCK

```
const sumNumbers = number => {  
  const sum = number + number;  
  return sum; } — RETURN STATEMENT  
};
```

싱글라인 블록은 { } 과 return 을 없앨 수 있다. 하지만 멀티라인 블록은 { } 을 넣어야한다.

So if we have a function:

```
const squareNum = (num) => {  
  return num * num;  
};
```

We can refactor the function to:

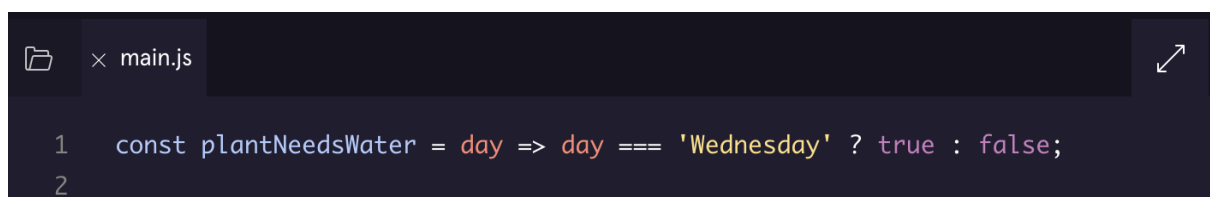
```
const squareNum = num => num * num;
```



A screenshot of a code editor window titled 'main.js'. The code defines a function 'plantNeedsWater' using an arrow function syntax. The function takes 'day' as an argument and returns a boolean value based on whether 'day' is 'Wednesday'. The code is as follows:

```
1 ▼ const plantNeedsWater = (day) => {  
2   return day === 'Wednesday' ? true : false;  
3 };
```

위 arrow 함수를 아래 concise arrow 함수로 리팩토링한 예시



A screenshot of a code editor window titled 'main.js'. The code shows the 'plantNeedsWater' function refactored into a concise arrow function. The code is as follows:

```
1 const plantNeedsWater = day => day === 'Wednesday' ? true : false;  
2
```