Cheatsheets / **Learn TypeScript**

Functions

TypeScript Parameter Type Annotations

Function parameters may be given type annotations with the same syntax as variable declarations – a colon next to the name. The type annotations ensure that the parameters are of the correct type.

```
function greet(noun: string) {
  console.log(`Hello, ${noun}!`);
}

greet('World'); // Prints: Hello, World

greet(2020); // Argument of type
'number' is not assignable to parameter
of type 'string'.
```

TypeScript Optional Parameter

Sometimes we would like to skip providing values to function calls. We can declare some parameters in a function to be optional if a value is not provided for all arguments in a function. We do this by adding a question mark symbol, `?`, after the parameter name before the colon, `:`.

```
function greet(name?: string) {
  console.log(`Hello, ${ name ||
'stranger' }!`);
}

greet(); // Prints: Hello, stranger!
```

TypeScript Default Parameters

If we assign a function parameter to have a default value, TypeScript will infer the parameter type to be the same as the default value's type.

```
function exponentiation(power = 1) {
  console.log(4 ** power);
}

exponentiation(); // Prints: 4

exponentiation(4); // Prints: 256

exponentiation(true); // Error:
Argument of type 'true' is not
```

```
assignable to parameter of type 'number
| undefined'.
```

TypeScript Inferring Return Types

By looking at the types of the values in a function's `return` statement, TypeScript can infer the `return` type of a function.

```
function factOrFiction() {
  return Math.random() >= .5 ? 'true' :
'false';
}

const myAnswer : boolean =
factOrFiction(); // Type 'string' is
not assignable to type 'boolean'
```

TypeScript Void Return Type

If a function does not return any value, then you can specify `void` as a return type using type annotation.

```
function sayHello(): void {
    console.log('Hello!')
}
```

TypeScript Explicit Return Types

We can be explicit about what type a function returns by adding type annotation (`:` followed by the type) after a function's closing parenthesis, `)` .

```
function trueOrFalse(value: boolean):
boolean {
  if (value) {
    return true;
  }

  return 'false'; // Typescript Error:
Type 'string' is not assignable to type
'boolean'.
}
```