Rest Parameters

Rest parameters (denoted by ...argumentName for the last argument) allow you to quickly accept multiple arguments in your function and get them as an array. This is demonstrated in the below example.

function iTakeItAll(first, second, ...allOthers) {

console.log(allOthers);

}

iTakeItAll('foo', 'bar'); // []

iTakeItAll('foo', 'bar', 'bas', 'qux'); // ['bas','qux']

Rest parameters can be used in any function be it function/()=>/class member

Required, optional, and default parameters all have one thing in common: they talk about one parameter at a time. Sometimes, you want to work with multiple parameters as a group, or you may not know how many parameters a function will ultimately take. In JavaScript, you can work with the arguments directly using the arguments variable that is visible inside every function body.

In TypeScript, you can gather these arguments together into a variable:

function buildName(firstName: string, ...restOfName: string[]) {

return firstName + " " + restOfName.join(" ");

}

let employeeName = buildName("Joseph", "Samuel", "Lucas", "MacKinzie");

*Rest parameters* are treated as a boundless number of optional parameters. When passing arguments for a rest parameter, you can use as many as you want; you can even pass none. The compiler will build an array of the arguments passed in with the name given after the ellipsis (...), allowing you to use it in your function.

The ellipsis is also used in the type of the function with rest parameters:

function buildName(firstName: string, ...restOfName: string[]) {

return firstName + " " + restOfName.join(" ");

}

let buildNameFun: (fname: string, ...rest: string[]) => string = buildName;