# **TYPESCRIPT 3.7 - WHAT'S NEW**

**Let's check out some of the new features available**

TypeScript 3.7 added some amazing new features that really help in simplifying and restructuring existing codebases.

**Optional Chaining**

### ?. Operator

Optional Chaining is a great new feature which allows expressions to only run up to the point until one of the accessed fields is **null** or **undefined**. This is through the use of the **?.** operator, which is interpreted as optional property access. It is expressed as checking if the value on the left of the question mark is **null** or **undefined** then stop execution and return **undefined**, elsecontinue as normal. See the example below:

**let** x = y?.z.execute();

This means that if y is not **null or undefined** then the full expression will be ran, and x will be assigned the result of execute(). However if y is undefined then x will be assigned **undefined**. This is the same as previously writing:

**let** x = (y === null || y === undefined) ? undefined : y.z.execute();

Clearly the latter is more verbose and less readable. However, there is one caveat to this new approach. Given code that may be replaced may look like this:

**if** (x && x.y && x.y.z)

It is important to note that the ?. operator acts differently to the && operator seen above, as this will evaluate as false if any of the values represent one of the known JavaScript falsy values (e.g. 0, NaN, ‘’, etc.).

Optional Chaining can also be used in two further areas – optional **element access** and **optional calls**.

### Optional Element Access

The optional chaining operator can be used when accessing to non-identifier properties. For instance if accessing an index of an array:

**// New operator usage**

**return** arr?.[0];

// Previous

return (arr === null || arr === undefined) ? // undefined : // arr[0];

### Optional Calls

These allow