***The bleeding edge JavaScript toolchain***

* **Style linting** - typically a linter like JSCS is used to ensure the source code is following a certain structure and style
* **Dependency Management** - for JavaScript projects, most people use other packages from npm; some plugins exist for build systems (e.g. Webpack) and compilers (e.g. Babel) that allow automatic installation of packages being imported or require()‘d
* **Transpilation** - a specific sub-genre of compilation, transpilation involves compiling code from one source version to another, only to a similar runtime level (e.g. ES6 to ES5)
* **Compilation** - specifically separate from transpiling ES6 and JSX to ES5, is the act of including assets, processing CSS files as JSON, or other mechanisms that can load and inject external assets and code into a file. In addition, there are all sorts of build steps that can analyze your code and even optimize it for you.
* **Minification and Compression** - typically part of – but not exclusively controlled by – compilation, is the act of minifying and compressing a JS file into fewer and/or smaller files
* **Source-Mapping** - another optional part of compilation is building source maps, which help identify the line in the original source code that corresponds with the line in the output code (i.e. where an error occurred)

*ImmutableJs*

In normal JavaScript, we know two common data types**: Object {} and Array [].**

To translate these into Immutable:

* **Object {} becomes Map Map({ })**
* **Array [] becomes List List([ ])**

To convert normal JavaScript into Immutable, we can use the Map, List, or fromJS functions that Immutable provides:



to convert it back to normal JS use following syntax :

**Advantages of ImmutableJs**

**1. Getting a nested value from an object without checking if it exists**

in normal JavaScript:



in Immutable:



You don’t need to check for undefined values all the way down the nested structure like you would in normal JavaScript:

This simple feature makes your code much more readable, less wordy

**2. Chaining manipulations**

First in normal JavaScript:



in immutable:

Because **List.push() returns the result of the operation,** we can “chain” the next operation right onto it. **In normal JavaScript, the push function returns the length of the new array**

**CONS OF IMMUTABLEJS**

* No Destructuring or Spread Operators for map,Lists and other immutables
* loDash expects js objects so it will also not work with immutables