

# Web Programming

## Week 8

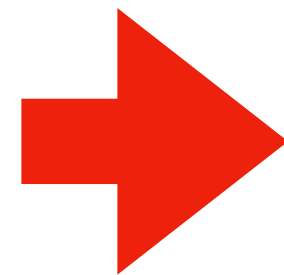
*"They call it optional typing but you still have to type in your code."*

Scott Davis on Groovy

# JS Goodie

# Storybook (initial)

Drehbuch, Intro, Functions  
Scientific foundations  
Algebraic Data Types  
Applied Science, Snake  
Scripting, PWA, Plotter, Excel  
Objects  
Classes  
JS Types, JsDoc  
Async Programming  
Modules  
Data Flow, Excel improved  
Iterator Protocol, Sequences  
Moves, User Interfaces  
Crazy JS



# Today: Types in JS

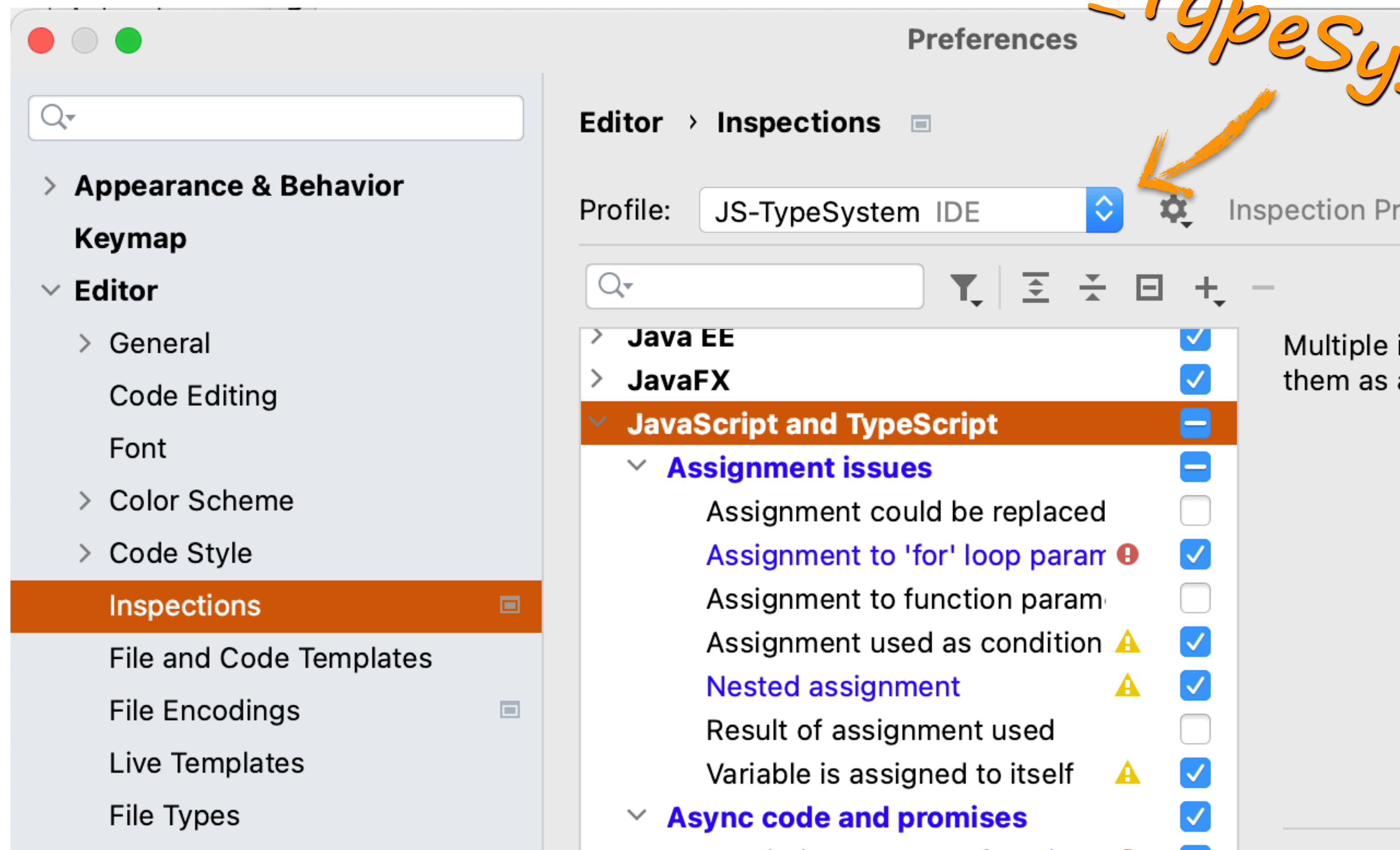
Define types in JsDoc

JsDoc is optional -> typing is optional

Not compiler enforced but IDE support

# IDE Setup

*JS-Typesystem.xml*



# Overview

Untyped: built-in types

Functions, data structures, interfaces

Callbacks, generics, type casts

# Built-in Types

*Literal Constructor  
Operators*

Boolean, Number, String, Object, BigInt,  
Symbol, Null, Undefined, Function  
(typeof)

Object "classes": Array, Date, RegExp,  
Error, Map, Set, JSON, ...

# Functions (classical)

@function, @constructor, @callback

@param

*! non-null    ? optional    ... any number*

@return



# Functions (curried)

@type { (A) => (B) => C }

@type { (a:A) => (b:B) => C }

@type { <T> (a:T) => (b:B) => T }

# Data Types I

Objects:

@property { Number } age - ...

Array:

{ Array<Number> }, { [Number] }

# Data Types II

Union type:  $\{ \text{Number} \mid \text{String} \}$

Intersection type:  $\{ \text{Person} \ \& \ \text{Cool} \}$

String literal type:  $\{ \text{"me"} \mid \text{"you"} \}$

Capability:  $\{ \{a:A, b:\text{Function}\} \}$

# Interfaces

@typedef

even the IDE shows it like an Interface

# Callbacks

@callback

a name for a function type that will be  
used in parameter position

# Generics

@template T      or      <T> notation

similar to Java Generics but the  
processing (unification, reification) and  
the namespace is unclear

# Type Casts

@type

used locally to help the type checker

# Best Practices I

Set up your IDE appropriately.

Follow conventions (type names capitalized, syntax variant, etc.)

Distinguish constructor from constructed object type.



# Best Practices II

@example

{@link xxx}

Validate via test case.

Push only on clean inspection state.

# Putting it all together

Have a look at Kolibri for a fully typed codebase with complex use cases.

# Practical Work

JsDoc your codebase,  
observe the inspection findings



# Ressources

<https://jsdoc.app>

<https://www.typescriptlang.org/docs/handbook/jsdoc-supported-types.html#type>

<https://docs.joshuatz.com/cheatsheets/js/jsdoc/#generics-in-jsdoc>