# Typing, Variables and Functions

Dan Wahlin

Twitter: @danwahlin

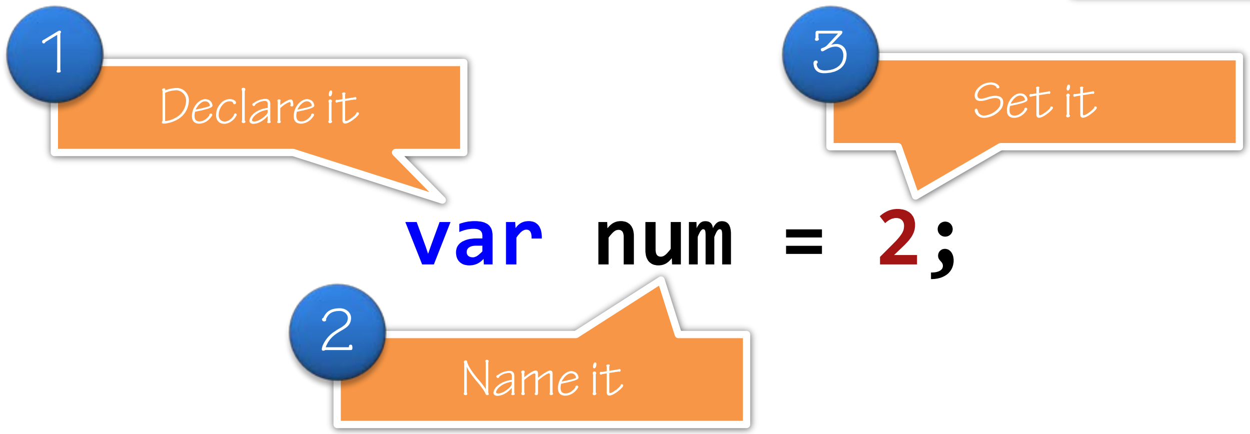
John Papa

Twitter: @john_papa

## Grammar, Declarations and Annotations

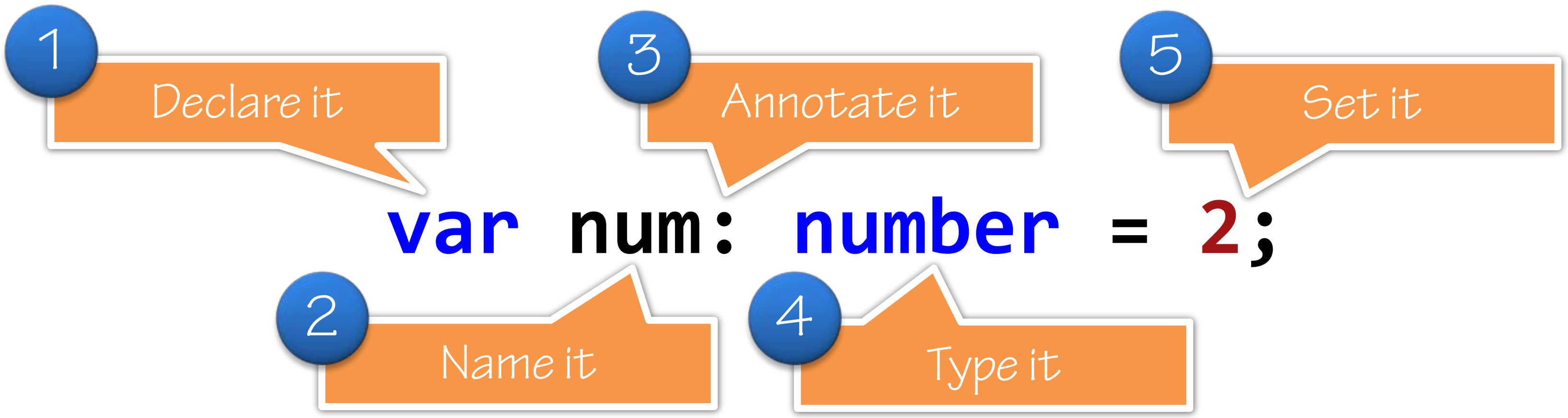
## Grammar: Type Inference

This TypeScript Example is JavaScript



```
var num = 2;
```

# Grammar: Type Annotations



```
var num: number = 2;
```

# Annotations and Inferences

| | |
|---|---|
| `var any1;` | Type could be any type (any) |
| `var num1: number;` | Type Annotation |
| `var num2: number = 2;` | Type Annotation Setting the Value |
| `var num3 = 3;` | Type Inference (number) |
| `var num4 = num3 + 100;` | Type Inference (number) |
| `var str1 = num1 + 'some string';` | Type Inference (string) |
| `var nothappy : number = num1 + 'some string';` | Error! |

## Typing and Ambient Declarations

pluralsight
hardcore developer training

## Dynamic and Static

| TypeScript | JavaScript |
|---|---|
| Static typing (optional) | Dynamic typing |
| Type safety is a compile-time feature | Type safety happens at run-time debugging |

# JavaScript’s Dynamic Types

Could be any type

```
var person;
person = 'John Papa';
person.substring(1, 4);

person = 1;
person.substring(1, 4);
```

Uncaught TypeError: Object 1 has no method 'substring'

## Ambient Declarations

TypeScript

```
declare var document;

document.title = "Hello";
```

JavaScript

```
document.title = "Hello";
```

lib.d.ts is referenced by default and contains references for the DOM and JavaScript

Ambient Declarations do not appear anywhere in the JavaScript

# Type Definition Files (aka Declaration Source Files)

TypeScript

```typescript
/// <reference path="jquery.d.ts" />

declare var $;

var data = "Hello John";

$("div").text(data);
```

Helps provide types for jquery

JavaScript

```javascript
var data = "Hello John";

$("div").text(data);
```

Ambient Declarations do not appear anywhere in the JavaScript

## Any and Primitive Types

## Any

- Represents any JavaScript value

```
var data: any;
var info;
```

any

No static type checking on “any"

## Primitive Types

```
var age: number = 2;
var score: number = 98.25;
var rating = 98.25;
```

number

```
var hasData: bool = true;
var isReady = true;
```

boolean

```
var firstName: string = 'John';
var lastName = 'Papa';
```

string

## Arrays and Indexers

```
var names: string[] = ['John', 'Dan', 'Aaron', 'Fritz'];

var firstPerson: string;
firstPerson = names[0];
```

indexer

# Primitive Types - Null

```
var num: number = null;
var str: string = null;
var isHappy: bool = null;
var customer: {} = null;
```

null

```
var age: number;
var customer = undefined;
```

undefined

Null type is a subtype of all primitives
(except void and undefined)

# Primitive Types - Undefined

```
var quantity: number;
var company = undefined;
```

undefined

undefined type is a subtype of all types

## Object Types

## Object Types

- **Examples**
  - Functions, class, module, interface, and literal types

- **May contain**
  - Properties
    - public or private
    - required or optional
  - Call signatures
  - Construct signatures
  - Index signatures

## Object Types

### Object literals

```
var square = { h: 10, w: 20 };

var points: Object = { x: 10, y: 20 };
```

### Functions

```
var multiply = function (x: number) {
    return x * x;
};

var multiplyMore: Function;
multiplyMore = function (x: number) {
    return x * x;
};
```

## Functions

pluralsight
hardcore developer training

# Functions

- **Parameter types (required and optional)**
- **Arrow function expressions**
  - Compact form of function expressions
  - Omit the function keyword
  - Have scope of “this”
- **Void**
  - Used as the return type for functions that return no value

## Arrow Function Expressions

TypeScript

```
var myFunc = function (h: number, w: number) {
    return h * w;
};
```

Omit the function keyword

Compact return statement

```
var myFunc = (h: number, w: number) => h * w;
```

Emit the same JavaScript

```
var myFunc = function (h, w) {
    return h * w;
};
```

## Void

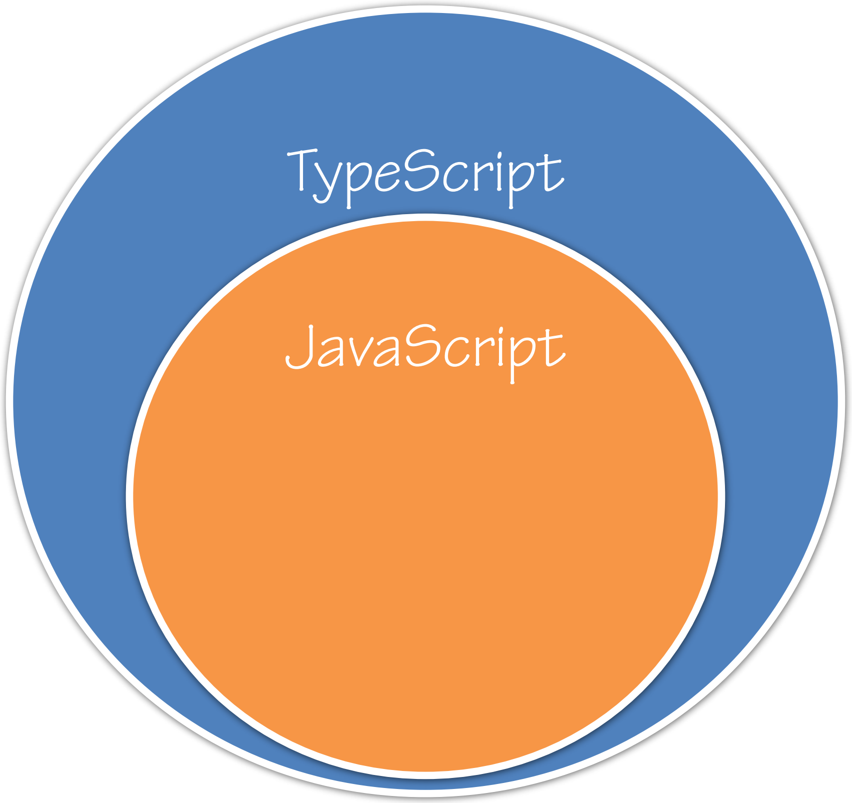
Used as the return type for functions that return no value

```
var greetMe : (msg: string) => void;

greetMe = function (msg) {
    console.log(msg);
}

greetMe('Hello!');
```

No return value

## Summary

## All JavaScript is Valid TypeScript



## Typings, Variables and Functions

- **Emits JavaScript**
- **Optional static typing**
  - Various types
- **Compile time checking**
- **Ambient Declarations for external references**
  - Use with typings (*.d.ts files)
- **Objects and functions**
  - Parameter types (required and optional)
  - Arrow function expressions
- **Interfaces**