TypeScript

coding JavaScript without the pain

@Sander_Mak

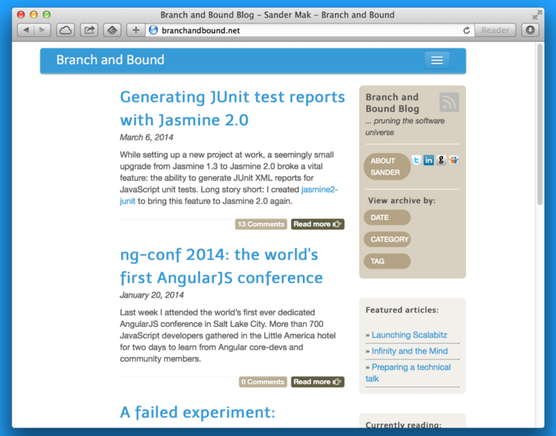
Luminis Technologies

INTRO

@Sander_Mak: Senior Software Engineer at Luminis

Author:

Dutch Java Magazine



blog @ branchandbound.net

Speaker:

JavaOne™ QCon geecon JavaLand JEE Conf Ts

AGENDA

- [ ] Why TypeScript?
- [ ] Language introduction / live-coding
- [ ] TypeScript and Angular
- [ ] Comparison with TS alternatives
- [ ] Conclusion

WHAT'S WRONG WITH JAVASCRIPT?

- Dynamic typing
- Lack of modularity
- Verbose patterns (IIFE)

WHAT'S WRONG WITH JAVASCRIPT?

- Dynamic typing
- Lack of modularity
- Verbose patterns (IIFE)

In short: JavaScript development scales badly

WHAT'S GOOD ABOUT JAVASCRIPT?

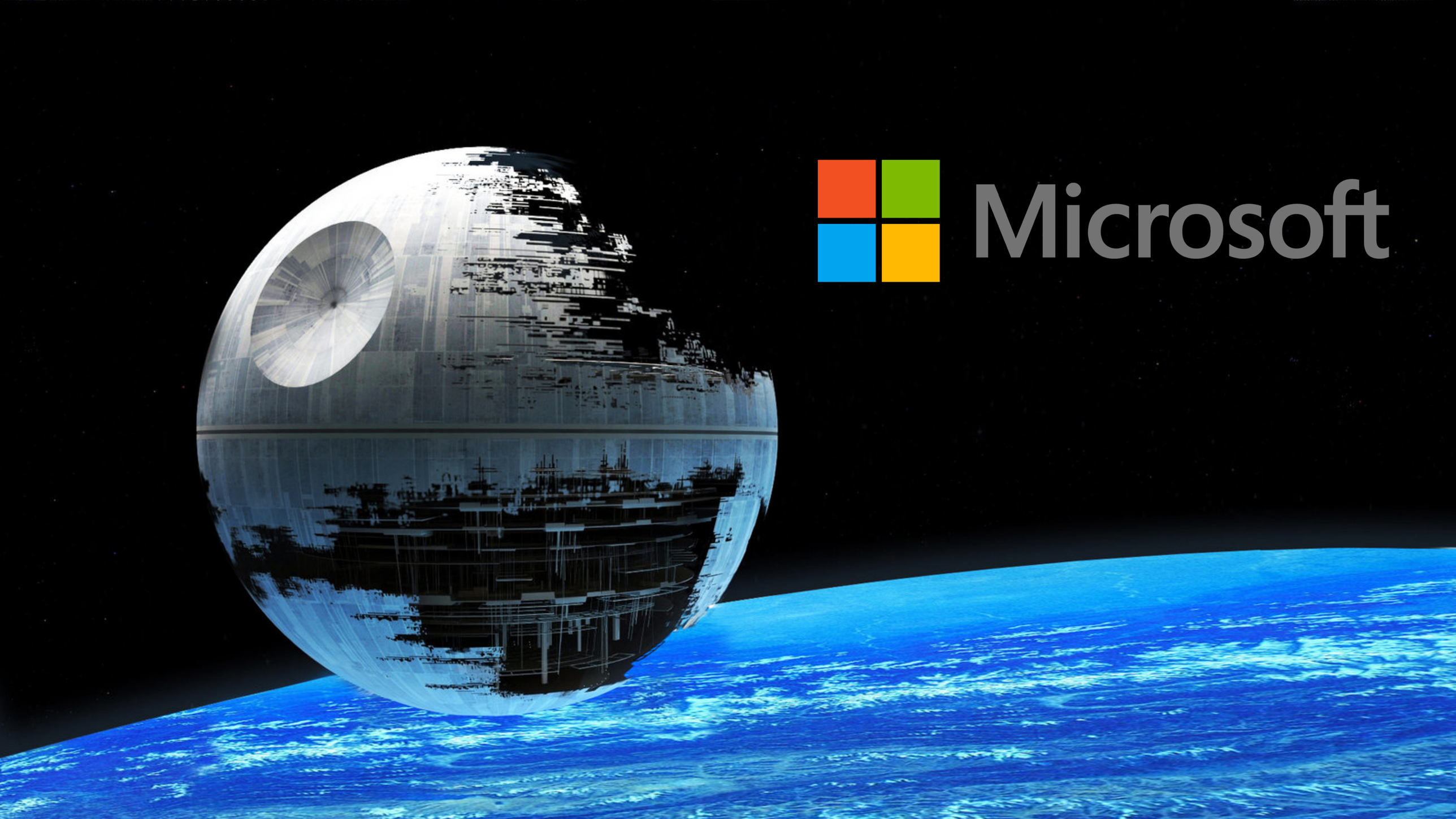
- [ ] It's everywhere
- [ ] Huge amount of libraries
- [ ] Flexible

WISHLIST

- Scalable HTML5 clientside development
- Modular development
- Easily learnable for Java developers
- Non-invasive (existing libs, browser support)
- Long-term vision
- Clean JS output (exit strategy)

WISHLIST

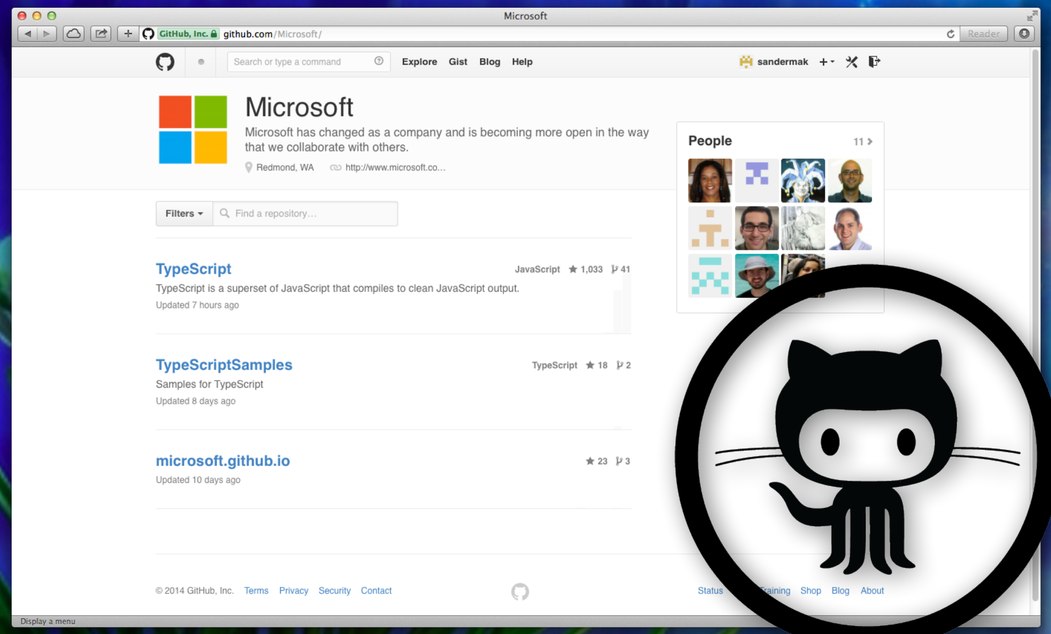
- [x] Scalable HTML5 clientside development
- [x] Modular development
- [x] Easily learnable for Java developers
- [x] Non-invasive (existing libs, browser support)
- [x] Long-term vision
- [x] Clean JS output (exit strategy)


Microsoft


Apache 2.0 licensed
node
JS
Microsoft
GitHub, Inc. github.com/Microsoft/
Search or type a command
Explore Gist Blog Help
sandermak
Microsoft
Microsoft has changed as a company and is becoming more open in the way that we collaborate with others.
Redmond, WA
http://www.microsoft.co...
People 11
Filters
Find a repository...
TypeScript
JavaScript 1,033 41
TypeScript is a superset of JavaScript that compiles to clean JavaScript output.
Updated 7 hours ago
TypeScriptSamples
TypeScript 18 2
Samples for TypeScript
Updated 8 days ago
microsoft.github.io
23 3
Updated 10 days ago
© 2014 GitHub, Inc. Terms Privacy Security Contact
Status Shop Blog About
Display a menu

Apache 2.0 licensed

node.js™



TYPESCRIPT

- Superset of JavaScript
- Optionally typed
- Compiles to ES3/ES5
- No special runtime
- 1.0 in April 2014, future ES6 alignment

TYPESCRIPT

- Superset of JavaScript
- Optionally typed
- Compiles to ES3/ES5
- No special runtime
- 1.0 in April 2014, future ES6 alignment

In short:	Lightweight productivity booster

GETTING STARTED

```
$ npm install -g typescript
$ mv mycode.js mycode.ts
$ tsc mycode.ts
```

GETTING STARTED

```
$ npm install -g typescript
$ mv mycode.js mycode.ts
$ tsc mycode.ts
```

May even find problems in existing JS!

OPTIONAL TYPES

Type annotations

JS

```
> var a = 123
> a.trim()

TypeError: undefined is
not a function
```

runtime

TS

```
> var a: string = 123
> a.trim()

Cannot convert 'number'
to 'string'.
```

compile-time

OPTIONAL TYPES

Type annotations

```
> var a = 123
> a.trim()

TypeError: undefined is
not a function
```

JS

```
> var a: string = 123
> a.trim()

Cannot convert 'number'
to 'string'.
```

TS

Type inference

```
> var a = 123
> a.trim()

The property 'trim' does
not exist on value of
type 'number'.
```

Types dissapear at runtime

OPTIONAL TYPES

Java	TS
Object	any
void	void
boolean	boolean
integer long short ...	number
String char	string
Type[]	type[]

OPTIONAL TYPES

Types are **structural** rather than **nominal**

TypeScript has **function types:**

```
var find: (elem: string, elems: string[]) => string =
    function(elem, elems) {
      ..
    }
```

OPTIONAL TYPES

Types are **structural** rather than **nominal**

TypeScript has **function types:**

```
var find: (elem: string, elems: string[]) => string =
    function(elem, elems) {
      ..
    }
```

DEMO: OPTIONAL TYPES

code 

Code: http://bit.ly/tscode

INTERFACES

```
interface MyInterface {
  // Call signature
  (param: number): string

  member: number
  optionalMember?: number

  myMethod(param: string): void
}

var instance: MyInterface = ...
instance(1)
```

INTERFACES

Use them to describe data returned in REST calls

```
$.getJSON('user/123').then((user: User) => {
   showProfile(user.details)
}
```

INTERFACES

TS interfaces are open-ended:

```
interface JQuery {
    appendTo(..): ..
    ..
}
```

jquery.d.ts

```
interface JQuery {
    draggable(..): ..
    ..
}
```

jquery.ui.d.ts

OPTIONAL TYPES: ENUMS

```
enum Language { TypeScript, Java, JavaScript }

var lang = Language.TypeScript
var ts = Language[0]
ts === "TypeScript"
```

```
enum Language { TypeScript = 1, Java, JavaScript }

var ts = Language[1]
```

GOING ALL THE WAY

Force explicit typing with noImplicitAny

```
var ambiguousType;

ambiguousType = 1
ambiguousType = "text"
```

noimplicitany.ts

```
$ tsc --noImplicitAny noimplicitany.ts
```

GOING ALL THE WAY

Force explicit typing with noImplicitAny

```
var ambiguousType;

ambiguousType = 1
ambiguousType = "text"
```

noimplicitany.ts

```
$ tsc --noImplicitAny noimplicitany.ts
```

```
error TS7005: Variable 'ambiguousType' implicitly
has an 'any' type.
```

TYPESCRIPT CLASSES

- Can implement interfaces
- Inheritance
- Instance methods/members
- Static methods/members

similar

Java

- Single constructor
- Default/optional parameters
- ES6 class syntax

different

DEMO: TYPESCRIPT CLASSES

code 

Code: http://bit.ly/tscode

ARROW FUNCTIONS

- Implicit return
- No braces for single expression
- Part of ES6

ARROW FUNCTIONS

- Implicit return
- No braces for single expression
- Part of ES6

```
function(arg1) {
    return arg1.toLowerCase();
}
```

ARROW FUNCTIONS

- Implicit return
- No braces for single expression
- Part of ES6

```
function(arg1) {
    return arg1.toLowerCase();
}
```

```
(arg1) => arg1.toLowerCase();
```

ARROW FUNCTIONS

- Implicit return
- No braces for single expression
- Part of ES6

```
function(arg1) {
    return arg1.toLowerCase();
}
```

```
(arg1) => arg1.toLowerCase();
```

Lexically-scoped this (no more 'var that = this')

DEMO: ARROW FUNCTIONS

code ~~Pics~~

Code: http://bit.ly/tscode

TYPE DEFINITIONS

How to integrate existing JS code?

- Ambient declarations
- Any-type :(
- Type definitions
- lib.d.ts
- Separate compilation:

```
tsc --declaration file.ts
```

TYPE DEFINITIONS

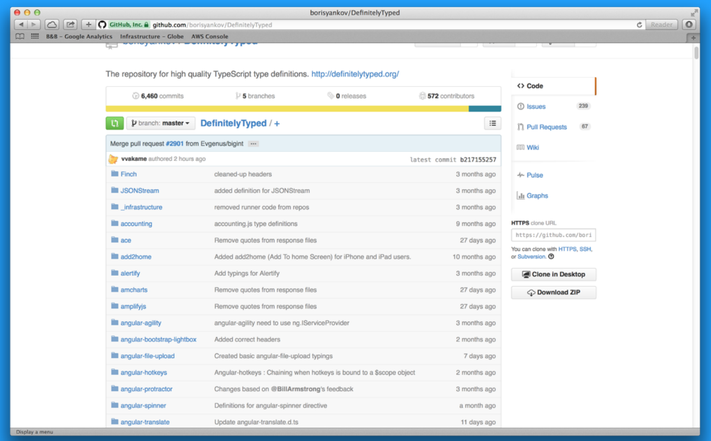
How to integrate existing JS code?

- Ambient declarations
- Any-type :(
- Type definitions
- lib.d.ts
- Separate compilation:

```
tsc --declaration file.ts
```

DefinitelyTyped.org

Community provided .d.ts files for popular JS libs



INTERNAL MODULES

```
module StorageModule {
   export interface Storage { store(content: string): void }

   var privateKey = 'storageKey';

   export class LocalStorage implements Storage {
      store(content: string): void {
         localStorage.setItem(privateKey, content);
      }
   }

   export class DevNullStorage implements Storage {
      store(content: string): void { }
   }
}

var storage: StorageModule.Storage = new StorageModule.LocalStorage();
storage.store('testing');
```

INTERNAL MODULES

```
module StorageModule {
   export interface Storage { store(content: string): void }

   var privateKey = 'storageKey';

   export class LocalStorage implements Storage {
      store(content: string): void {
         localStorage.setItem(privateKey, content);
      }
   }

   export class DevNullStorage implements Storage {
      store(content: string): void { }
   }
}

var storage: StorageModule.Storage = new StorageModule.LocalStorage();
storage.store('testing');
```

INTERNAL MODULES

```
module StorageModule {
   export interface Storage { store(content: string): void }

   var privateKey = 'storageKey';

   export class LocalStorage implements Storage {
      store(content: string): void {
         localStorage.setItem(privateKey, content);
      }
   }

   export class DevNullStorage implements Storage {
      store(content: string): void { }
   }
}

var storage: StorageModule.Storage = new StorageModule.LocalStorage();
storage.store('testing');
```

INTERNAL MODULES

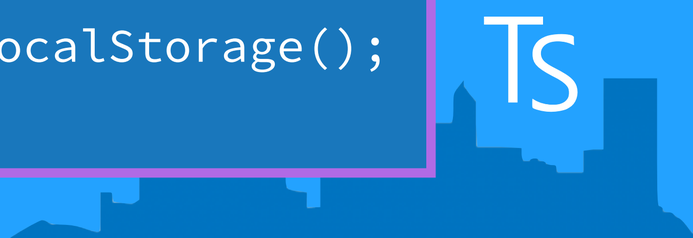
```
module StorageModule {
   export interface Storage { store(content: string): void }

   var privateKey = 'storageKey';

   export class LocalStorage implements Storage {
      store(content: string): void {
         localStorage.setItem(privateKey, content);
      }
   }

   export class DevNullStorage implements Storage {
      store(content: string): void { }
   }
}
```

```
var storage: StorageModule.Storage = new StorageModule.LocalStorage();
storage.store('testing');
```



INTERNAL MODULES

TS internal modules are open-ended:

```
module Webshop {
  export class Cart { .. }
}
```

cart.ts

```
/// <reference path="cart.ts" />
module Webshop {
 export class Catalog { .. }
}
```

main.ts

INTERNAL MODULES

TS internal modules are open-ended:

```
module Webshop {
  export class Cart { .. }
}
```

cart.ts

```
/// <reference path="cart.ts" />
module Webshop {
 export class Catalog { .. }
}
```

main.ts

Can be hierarchical:

```
module Webshop.Cart.Backend {
  ...
}
```

INTERNAL MODULES

TS internal modules are open-ended:

```
module Webshop {
  export class Cart { .. }
}
```

cart.ts

```
/// <reference path="cart.ts" />
module Webshop {
 export class Catalog { .. }
}
```

main.ts

Can be hierarchical:

```
module Webshop.Cart.Backend {
  ...
}
```

Combine modules:

```
$ tsc --out main.js main.ts
```

DEMO: PUTTING IT ALL TOGETHER

code ~~Pics~~

Code: http://bit.ly/tscode

EXTERNAL MODULES

CommonJS

```
$ tsc --module common main.ts
```

Asynchronous Module Definitions

```
$ tsc --module amd main.ts
```

Combine with module loader

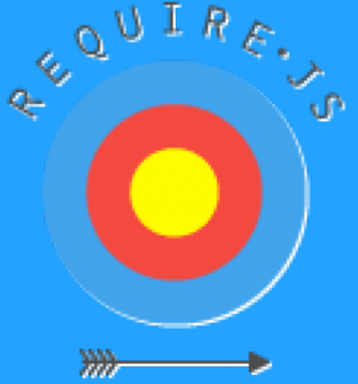
EXTERNAL MODULES

- 'Standards-based', use existing external modules
- Automatic dependency management
- Lazy loading
- AMD verbose without TypeScript
- Currently not ES6-compatible

DEMO

+

+



=

DEMO: TYPESCRIPT AND ANGULAR

code

Code: http://bit.ly/tscode

BUILDING TYPESCRIPT

```
$ tsc -watch main.ts
```

gulp-type (incremental)

gulp-tsc

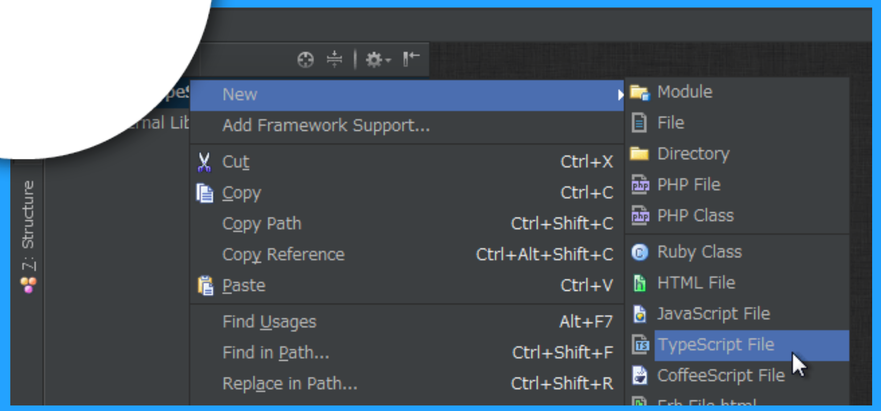
grunt-typescript

grunt-ts

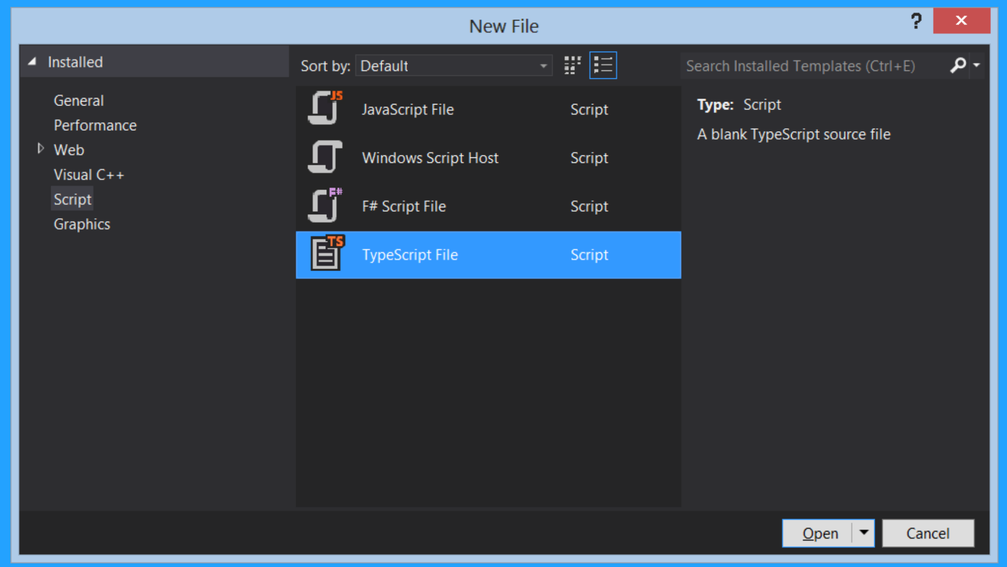
TOOLING

JetBRAINS

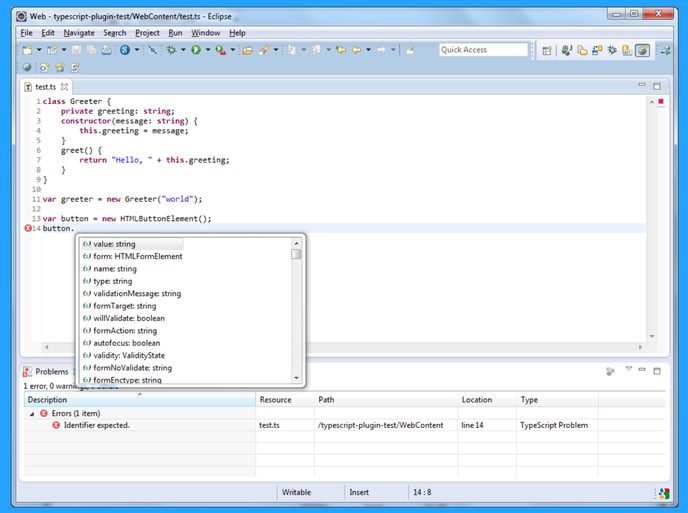
IntelliJ IDEA
WebStorm



Visual Studio



eclipse
plugin



Ts

TYPESCRIPT vs ES6 HARMONY

ES6

- Complete language + runtime overhaul
- More features: generators, comprehensions, object literals
- Will take years before widely deployed
- No typing (possible ES7)

TYPESCRIPT vs COFFEESCRIPT

- Also a compile-to-JS language
- More syntactic sugar, still dynamically typed
- JS is not valid CoffeeScript
- No spec, definitely no Anders Hejlsberg...
- Future: CS doesn't track ECMAScript 6

TYPESCRIPT vs DART

Dart

- Dart VM + stdlib (also compile-to-JS)
- Optionally typed
- Completely different syntax & semantics than JS
- JS interop through dart:js library
- ECMA Dart spec

TYPESCRIPT vs CLOSURE COMPILER

Google Closure Compiler

- Pure JS
- Types in JsDoc comments
- Less expressive
- Focus on optimization, dead-code removal

WHO USES TYPESCRIPT?

Adobe

Microsoft (duh)

Walmart

redhat

Qhawtio

Palantir

LEAP MOTION

CONCLUSION

Internal modules	External modules	Generics
Classes/Interfaces	Type defs	Type defs
Some typing	More typing	-noImplicitAny

CONCLUSION

TypeScript allows for gradual adoption

Internal modules	External modules	Generics
Classes/Interfaces	Type defs	Type defs
Some typing	More typing	-noImplicitAny

CONCLUSION

Some downsides:

- ☐ Still need to know some JS quirks
- ☐ Current compiler slowish (faster one in the works)
- ☐ External module syntax not ES6-compatible (yet)
- ☐ Non-MS tooling lagging a bit

CONCLUSION

- High value, low cost improvement over JavaScript
- Safer and more modular
- Solid path to ES6

MORE TALKS

TypeScript:
Wednesday, 11:30 AM, same room

Akka & Event-sourcing
Wednesday, 8:30 AM, same room

@Sander_Mak
Luminis Technologies

RESOURCES

Code: http://bit.ly/tscode

Learn: www.typescriptlang.org/Handbook

@Sander_Mak
Luminis Technologies