Flow Type Checker

and Topics about Type Systems

Flow Type Checker

Type

Type Systems

Type Systems

Static vs. Dynamic

Strong vs. Weak

Declared vs. Inferred

...

Static Typing

Code	What does the compiler think	
`int a = 42;`	Looks good	
`bool b = true;`	Looks good	
`??? c = a + b;`	BOOM!	

Inferred Static Typing

Code	What does the compiler think	
`var a = 42;`	a is a number	
`var b = true;`	b is a boolean	
`var c = a + b;`	BOOM!	

Dynamic Typing

Code	What does the compiler think	What does the runtime think
`var a = 42;`	???	a is a number
`var b = true;`	???	b is a boolean
`var c = a + b;`	???	BOOM!

JavaScript Typing

Code	What does the compiler think	What does the runtime think
`var a = 42;`	???	Dingdingding
`var b = true;`	???	Wapapapapow
`var c = a + b;`	???	Hateehateeho

Why type checking?

Flow

Created by Facebook

A static type checker for JS

Goals: Precise & Fast

Type Annotation

```
let a: number = 42;
let b: boolean = true;

function square(n: number): number {
    return n * n;
}

square(a); // ok
square(b); // error
```

Type Annotation (Comment)

```
let a/*: number */ = 42;
let b/*: boolean */ = true;

function square(n/*: number */)
    /*: number */ {
    return n * n;
}

square(a); // ok
square(b); // error
```

Type Inference

```
let a = 42;
let b = true;

function square(n) {
    return n * n; // n is number!
}

square(a); // ok
square(b); // error
```

Literal Types

```
let a: number = 42; // number type
let b: 42 = 42; // literal type
let c: Array<number> = [42, 42];
    // array type
let d: [number, number] = [42, 42];
    // tuple type of two numbers
let e: [42, 42] = [42, 42];
    // tuple type of two literals
```

Any Type & Maybe Types

```
let a: any = 42; // ok
a = true; // ok
a = { p: 1, q: ["whatever"] }; // ok

let b: ?number = 42; // ok
b = null; // ok
b = "42"; // error
```

Maybe Types

```
let b: ?number = 42; // ok
b = null; // ok
b = "42"; // error

let c = b + 42; // error
if (typeof b === 'number') {
    let c = b + 42; // ok
}
```

Duck Typing

“If it walks like a duck and it quacks like a duck, then it must be a duck”

```
type A = { a: number, b: boolean }
type B = { a: number, c: string }

let x = { a: 42, b: true } // x is A
let y = { a: 42, c: "meow" } // y is B
```

Duck Typing & Type Algebra

```
type A = { a: number, b: boolean }
type B = { a: number, c: string }

let x = { a: 42, b: true } // x is A
let y = { a: 42, c: "meow" } // y is B

let z: A | B = x; // ok
z = y; // ok
```

Duck Typing & Type Algebra

```
type A = { a: number, b: boolean }
type B = { a: number, c: string }

let x = { a: 42, b: true } // x is A
let y = { a: 42, c: "meow" } // y is B

let z: A & B = x; // error
let w = { a: 42, b: true, c: "meow" };
z = w; // ok
```

Generic

```
function print<T>(value: T): T {
    console.log(value);
    return value;
}

print(42); // ok
print(true); // ok
```

Pattern Matching (JavaScript events)

```
type Event =
    ('connect', () => void)
        => Server
    | ('message', (string) => void)
        => Server;
```

JavaScript Typing

Code	What does the compiler think	What does the runtime think
`var a = 42;`	???	Dingdingding
`var b = true;`	???	Wapapapapow
`var c = a + b;`	???	Hateehateeho

Flow Typing

Code	What does Flow think	What does the runtime think
`var a = 42;`	a is a number	Dingdingding
`var b = true;`	b is a boolean	Wapapapapow
`var c = a + b;`	error!	Hateehateeho

Flow vs. TypeScript

TypeScript =
“new” language + transpiler
Flow =
type annotation + checker