

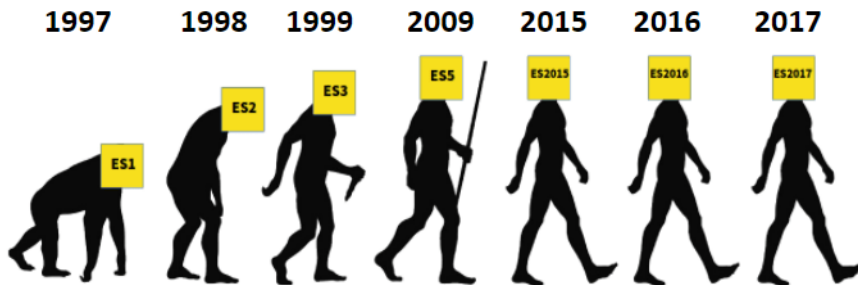
TypeScript

Старков Дима

Mocha / LiveScript / JavaScript



ECMAScript



Эволюция ECMAScript

ES1, ES2 — первые редакции

ES3 (1999) — RegExp, try/catch

ES4 — Темные времена

ES5 (2009) — строгий режим, геттеры и сеттеры, нативная поддержка JSON, новые методы Array и Object

Пример кода на ES5

```
function Student(firstName, lastName) {  
    this.firstName = firstName;  
    this.lastName = lastName;  
}  
  
Object.defineProperty(Student.prototype, 'name', {  
    get: function() {  
        return this.firstName + ' ' + this.lastName;  
    }  
});
```

ES6

~~ES6~~ ECMAScript 2015

Классы

let и const

Стрелочные функции

Шаблонные строки

Деструктуризация

Promise

ECMAScript 2016

Оператор `**` (возведение в степень)

`Array.prototype.includes`

ECMAScript 2017

async / await

Object.values

Object.entries

String.prototype.pad[Start|End]

ECMAScript 2018

Rest и Spread оператор массивов

Rest и Spread оператор объектов

Promise.prototype.finally










Дальнейшее развитие ECMAScript



[tc39/ecma262](https://github.com/tc39/ecma262)

Зачем всё это знать?

Статистика использования браузеров

<input checked="" type="checkbox"/>	 Google Chrome	95 506 767 374	40,51 %
<input checked="" type="checkbox"/>	 Яндекс.Браузер	47 186 148 692	20,02 %
<input checked="" type="checkbox"/>	 Safari	25 134 247 834	10,66 %
<input type="checkbox"/>	 Opera	12 329 612 387	5,23 %
<input type="checkbox"/>	 Firefox	9 985 290 466	4,24 %
<input type="checkbox"/>	 Android Browser	7 088 843 990	3,01 %
<input type="checkbox"/>	 Samsung Internet	6 218 626 156	2,64 %
<input type="checkbox"/>	 Internet Explorer	4 279 508 750	1,82 %
<input type="checkbox"/>	 Edge	3 612 438 677	1,53 %

Данные из отчёта «Яндекс.Радара» на 3 декабря 2019

Таблица совместимости

<https://kangax.github.io/compat-table>

		Compilers/polyfills										Desktop browsers										Servers/runtimes										Mobile	
		78%	6%	56%	52%	55%	10%	0%	52%	58%	78%	78%	78%	100%	100%	83%	90%	2%	19%	77%	100%	8%	94%	83%	90%								
Feature name	Current browser	Traceur	Babel 6+ core-js	Closure 2018.10	Type-Script + core-js	es7-shim	IE 11	Edge 17	Edge 18	FF 60 ESR	FF 62	FF 63	CH 69, OP 56	CH 70, OP 57	SF 11.1	SF 12	PJS	Node >=6.5 <7.12	Node >=8.10 <9.12	Node >=10.13 <11.12	DUK 2.2	GraalVM 1.0.3	iOS 11.3	iOS 12									
2016 features																																	
exponentiation (**) operator		3/3	2/3	3/3	3/3	2/3	0/3	0/3	3/3	3/3	3/3	3/3	3/3	3/3	3/3	3/3	3/3	0/3	0/3	3/3	3/3	2/3	3/3	3/3	3/3								
Array.prototype.includes		3/3	0/3	3/3	2/3	3/3	2/3	0/3	3/3	3/3	3/3	3/3	3/3	3/3	3/3	3/3	3/3	0/3	3/3	3/3	3/3	0/3	3/3	3/3	3/3								
2016 misc																																	
generator functions can't be used with "new"		Yes	No	No	No	No	No	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	No	Yes	Yes	Yes	No	Yes	Yes	Yes								
generator.throw() caught by inner generator		Yes	No	No	Yes	Yes	No	No	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	No	Yes	Yes	Yes	No	Yes	Yes	Yes								
strict fn w/ non-strict non-simple params is error		Yes	No	No	No	No	No	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	No	Yes	Yes	Yes	No	Yes	Yes	Yes								
nested rest destructuring, declarations		Yes	No	Yes	Yes	Yes	No	No	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	No	Yes	Yes	Yes	No	Yes	Yes	Yes								
nested rest destructuring, parameters		Yes	No	Yes	Yes	Yes	No	No	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	No	Yes	Yes	Yes	No	Yes	Yes	Yes								
Proxy, "enumerate" handler removed		Yes	No	No	No	No	No	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	No	Yes	Yes	Yes	Yes	Yes	Yes	Yes								
Proxy internal calls, Array.prototype.includes		Yes	No	No	No	No	No	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	No	Yes	Yes	Yes	No	Yes	Yes	Yes								
2017 features																																	
Object static methods		4/4	0/4	4/4	3/4	4/4	3/4	0/4	4/4	4/4	4/4	4/4	4/4	4/4	4/4	4/4	4/4	0/4	0/4	4/4	4/4	0/4	4/4	4/4	4/4								
String padding		2/2	0/2	2/2	2/2	2/2	2/2	0/2	2/2	2/2	2/2	2/2	2/2	2/2	2/2	2/2	2/2	0/2	0/2	2/2	2/2	0/2	2/2	2/2	2/2								
trailing commas in function syntax		2/2	0/2	2/2	2/2	2/2	0/2	2/2	2/2	2/2	2/2	2/2	2/2	2/2	2/2	2/2	2/2	0/2	0/2	2/2	2/2	0/2	2/2	2/2	2/2								
async functions		15/15	3/15	3/15	9/15	8/15	0/15	0/15	15/15	15/15	15/15	15/15	15/15	15/15	15/15	15/15	15/15	0/15	0/15	15/15	15/15	0/15	13/15	15/15	15/15								
shared memory and atomics		0/17	0/17	0/17	0/17	0/17	0/17	0/17	0/17	0/17	0/17	0/17	17/17	17/17	0/17	0/17	0/17	0/17	17/17	17/17	0/17	17/17	0/17	17/17	0/17								
2017 misc																																	
Proxy "ownKeys" handler, duplicate keys for non-extensible targets (ES 2017 semantics)		No	No	No	No	No	No	Yes	Yes	No	No	No	Yes	Yes	Yes	Yes	Yes	No	Yes	Yes	Yes	Yes	Yes	Yes	Yes								
RegExp "u" flag, case folding		Yes	No	No	No	No	No	No	No	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	No	No	Yes	Yes	No	Yes	Yes	Yes								
arguments.caller removed		Yes	No	No	No	No	No	No	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	No	No	Yes	Yes	No	Yes	Yes	Yes								
2017 annex b																																	
Object.prototype.getter/setter methods		16/16	0/16	16/16	0/16	16/16	0/16	8/16	14/16	14/16	16/16	16/16	16/16	16/16	16/16	16/16	16/16	12/16	10/16	16/16	16/16	16/16	16/16	16/16	16/16								
Proxy internal calls, getter/setter methods		4/4	0/4	0/4	0/4	0/4	0/4	0/4	4/4	4/4	4/4	4/4	4/4	4/4	4/4	4/4	4/4	0/4	0/4	4/4	4/4	0/4	4/4	4/4	4/4								
assignments allowed in for-in head in non-strict mode		Yes	Yes	No	No	No	No	No	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes								

Разный уровень поддержки языка



Транспиляция — это процесс перевода
исходного кода одного языка в другой



BABEL

Babel Playground

Код на ES2015

```
class Student {  
  constructor(firstName, lastName) {  
    this.firstName = firstName;  
    this.lastName = lastName;  
  }  
  
  set skills(skills = []) {  
    this._skills = skills;  
  }  
}
```

Код после транспиляции

```
var Student = function () {  
  function Student(firstName, lastName) {  
    _classCallCheck(this, Student);  
  
    this.firstName = firstName;  
    this.lastName = lastName;  
  }  
  
  _createClass(Student, [{  
    key: 'skills',  
    set: function() {  
      var args = arguments;  
      var skills = args.length > 0 && args[0] !== undefined ? args[0] : [];  
  
      this._skills = skills;  
    }  
  ]]);  
  
  return Student;  
}();
```

Система плагинов

Plugins

arrow-functions

classes

destructuring

duplicate-keys

for-of

С поддержкой браузерами справились
Что дальше?

Что не так?

```
function sum(a, b) {  
    return a + b;  
}
```

```
sum(2, 2); // 4
```

```
sum(2, '2'); // '22'
```

```
sum([], []); // ''
```

```
sum([], {}); // '[object Object]'
```

```
sum({}, []); // 0
```

Как исправить?

```
function sum(a, b) {  
  if (typeof a !== 'number' || typeof b !== 'number') {  
    throw new Error('a and b must be numbers');  
  }  
  
  return a + b;  
}
```

Что в IDE?

```
1 function sum(a, b) {  
2   a.  
3  
4  
5  
6  
7  
8 }  
9
```

constructor (Object, built-in)	Function
hasOwnProperty(v: PropertyKey) (Object, built-in)	boolean
isPrototypeOf(v: Object) (Object, built-in)	boolean
propertyIsEnumerable(v: PropertyKey) (Object, built-in)	boolean
toLocaleString() (Object, built-in)	string
toString() (Object, built-in)	string
valueOf() (Object, built-in)	Object
result (-2018/.../core.js)	
\$current (jquery.textillate.js)	
\$texts	
_ (k, -2018/.../core.js)	string
done (lib/thead.min.js)	number

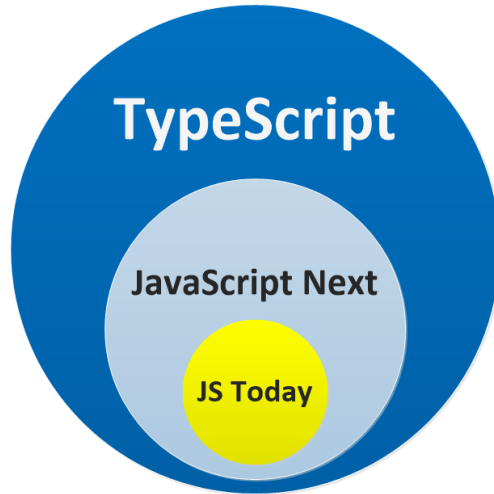
Not all variants are shown, please type more letters to see the rest Next Tip

Слабая динамическая типизация

Неявные преобразования

Типы становятся известны в runtime

TypeScript



TypeScript Playground

Автор



Андерс Хейлсберг

Известен также по Turbo Pascal, Delphi, C#

Возможности

ES2015-ESNext

Транспиляция в ES2015, ES5, ES3 ...

Аннотации типов и их проверка

Вывод типов

Дополнительные фиичи

Примитивные типы данных

```
const isDone: boolean = false;
```

```
const decimal: number = 6 + 0xf00d + 0b1010;
```

```
const sentence: string = 'Hello, my name is Alice.';
```

```
const u: undefined = undefined;
```

```
const n: null = null;
```

Массивы (Array)

```
const list1: number[] = [1, 2, 3];
```

```
const list2: Array<number> = [1, 2, 3];
```

Кортежи (Tuple)

~~Кортежи~~ Tuple

// Объявление кортежа

```
let point: [number, number];
```

// Некорректная инициализация

// Type 'string' is not assignable to type 'number'.

```
point = [10, 'hello'];
```

// Инициализация

```
point = [20, 10]; // OK
```


Перечисления (Enum)

```
enum Color {  
    Red,  
    Green,  
    Blue  
}
```

```
const c: Color = Color.Green;
```

Перечисления (Enum)

```
enum Color {  
    Red = 5,  
    Green = 7,  
    Blue = 9  
}
```

```
const c: Color = Color.Green;
```

Перечисления (Enum)

```
enum Color {  
    Red = '#f00',  
    Green = '#0f0',  
    Blue = '#00f'  
}  
  
const c: Color = Color.Green;
```

Объекты (Object)

```
const colors: object = {  
  red: '#F00',  
  green: '#0F0',  
  blue: '#00F'  
};
```

```
const settings: {  
  color: string;  
  delay: number;  
  retry: boolean;  
} = {  
  color: '#F00',  
  delay: 2000,  
  retry: false  
};
```

Any

```
let notSure: any = 4;  
notSure = "maybe a string instead";  
notSure = false;
```

```
let notSure: any = 4;  
notSure.foo(); // TypeError: notSure.foo is not a function  
notSure.toFixed(); // OK
```

```
let prettySure: object = new Number(4);  
// Property 'toFixed' does not exist on type 'object'.  
prettySure.toFixed();
```

Void и Never

```
function warnUser(): void {  
    console.log("This is my warning message");  
}
```

```
function error(message: string): never {  
    throw new Error(message);  
}
```

```
function infiniteLoop(): never {  
    while (true) {}  
}
```

Приведение типов

```
const someValue: any = 'this is a string';
```

```
const strLength1: number = (<string>someValue).length;
```

```
const strLength2: number = (someValue as string).length;
```

Функции

```
function sum(x: number, y: number): number {  
    return x + y;  
}
```

```
const sum = (x: number, y: number): number => x + y;
```


Интерфейсы

```
function printLogin(user: { login: string }): void {  
    console.log(user.login);  
}
```

```
const user = {  
    login: 'maxie'  
};
```

```
printLogin(user);
```

Интерфейсы

```
interface IUser {  
    login: string;  
}  
  
function printLogin(user: IUser): void {  
    console.log(user.login);  
}  
  
const user = {  
    username: 'maxie'  
};  
  
printLogin(user);
```

Опциональные свойства

```
interface IOptions {  
    color?: string;  
}
```

```
interface ISquare {  
    color: string;  
}
```

```
function create(options: IOptions): ISquare {  
    const square: ISquare = { color: 'white' };
```

```
    // Type 'undefined' is not assignable to type 'string'.  
    square.color = options.color;
```

```
    return square;  
}
```

Опциональные свойства

```
interface IOptions {  
    color?: string;  
}
```

```
interface ISquare {  
    color: string;  
}
```

```
function create(options: IOptions): ISquare {  
    const square: ISquare = { color: 'white' };  
  
    if (options.color) {  
        square.color = options.color;  
    }  
  
    return square;  
}
```

Классы

```
interface IMakesSound {  
    makeSound(): void;  
}  
  
class Python implements IMakesSound {  
    private readonly _length: number;  
  
    constructor(length: number) {  
        this._length = length;  
    }  
  
    public get length(): number {  
        return this._length / 100;  
    }  
  
    protected makeSound() {  
        console.log('Ssssss!');  
    }  
}
```

Классы

```
abstract class Snake {  
    private readonly _length: number;  
  
    public get length(): number {  
        return this._length / 100;  
    }  
  
    constructor(length: number) {  
        this._length = length;  
    }  
  
    protected abstract makeSound(): void;  
}
```

Классы

```
class Python extends Snake {  
    private static population = 10000;  
  
    public static incrementPopulation(): void {  
        Python.population++;  
    }  
  
    constructor(length: number) {  
        super(length);  
        Python.incrementPopulation();  
    }  
  
    protected makeSound(): void {  
        console.log('Ssssss!');  
    }  
}
```

Перерыв?

ВЫВОД ТИПОВ

```
let n: number = 42  
let s: string = 'Hello, world!'  
let a: number[] = [1, 2, 3, 4]
```

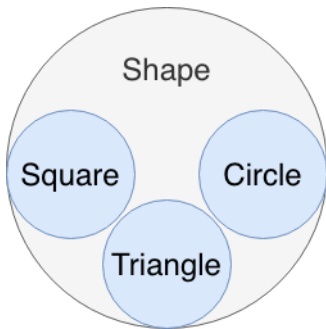
```
let n = 42  
let s = 'Hello, world!'  
let a = [1, 2, 3, 4]
```



Наиболее общий тип

```
let shapes = [new Circle(), new Square()]
```

```
shapes.push(new Triangle())
```



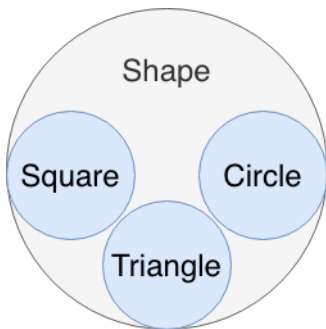
Наиболее общий тип

```
let shapes = [new Circle(), new Square()]
```

```
// Argument of type 'Triangle'
```

```
// is not assignable to parameter of type 'Square | Circle'.
```

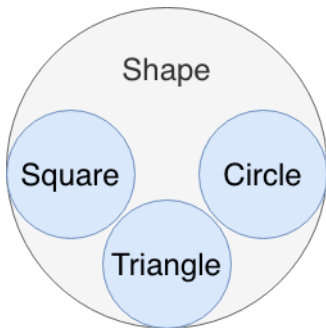
```
shapes.push(new Triangle())
```



Наиболее общий тип

```
let shapes = [new Circle(), new Square()]
```

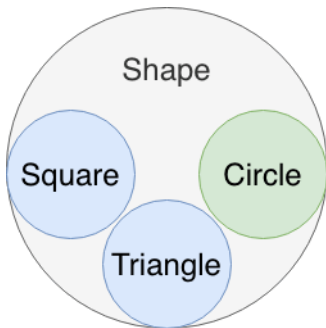
```
shapes.push(new Triangle())
```



Наиболее общий тип

```
let shapes = [new Circle(), new Square()]
```

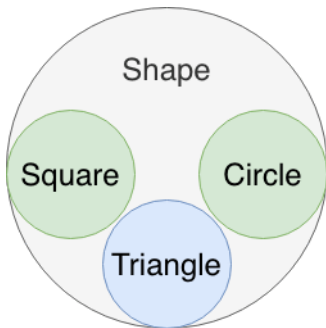
```
shapes.push(new Triangle())
```



Наиболее общий тип

```
let shapes = [new Circle(), new Square()]
```

```
shapes.push(new Triangle())
```



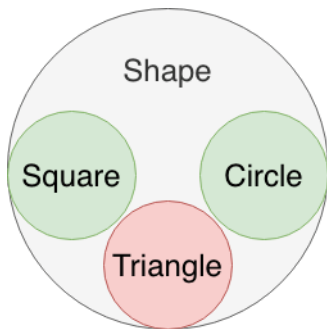
Наиболее общий тип

```
let shapes = [new Circle(), new Square()]
```

```
// Argument of type 'Triangle'
```

```
// is not assignable to parameter of type 'Square | Circle'.
```

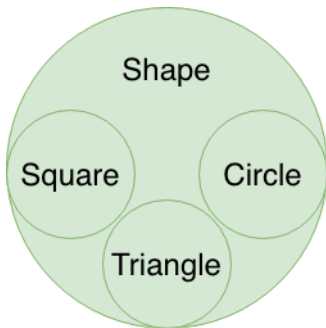
```
shapes.push(new Triangle())
```



Наиболее общий тип

```
let shapes: Shape[] = [new Circle(), new Square()]
```

```
shapes.push(new Triangle())
```



СОВМЕСТИМОСТЬ ТИПОВ

```
class Human {  
    name: string  
}
```

```
class Robot {  
    name: string  
}
```

```
let human: Human = new Robot()
```



Проверка типа в runtime

```
function addShape(shapes: Shape[], obj: object) {  
    if (obj instanceof Shape) {  
        shapes.push(obj as Shape)  
    }  
  
    throw new TypeError('Argument is not instanceof Shape')  
}
```

Проверка типа в runtime

```
function addShape(shapes: Shape[], obj: object) {  
    if (obj instanceof Shape) {  
        shapes.push(obj as Shape)  
    }  
  
    throw new TypeError('Argument is not instanceof Shape')  
}
```

Проверка типа в runtime

```
function addShape(shapes: Shape[], obj: object) {  
  if (obj instanceof Shape) {  
    shapes.push(obj as Shape)  
  }  
  
  throw new TypeError('Argument is not instanceof Shape')  
}
```

Проверка типа в runtime

```
function addShape(shapes: Shape[], obj: object) {  
  if (obj instanceof Shape) {  
    shapes.push(obj as Shape)  
  }  
  
  throw new TypeError('Argument is not instanceof Shape')  
}
```

Проверка типа в runtime

```
function addShape(shapes: Shape[], obj: object) {  
    if (obj instanceof Shape) {  
        shapes.push(obj as Shape)  
    }  
  
    throw new TypeError('Argument is not instanceof Shape')  
}
```

Type Guard

```
function addShape(shapes: Shape[], obj: object) {  
  if (obj instanceof Shape) {  
    shapes.push(obj)  
  }  
  
  throw new TypeError('Argument is not instanceof Shape')  
}
```

Нужно уметь обрабатывать значения
разных типов идентичным образом

Кажется нам нужен...



ану



Это полиморфизм?

Обобщенные типы

```
function identity(arg: any): any {  
    return arg;  
}
```

Обобщенные типы

```
function identity<T>(arg: T): T {  
    return arg;  
}
```

Обобщенные функции

```
function identity<T>(arg: T): T {  
    return arg;  
}
```

```
identity('string') // T is string
```

```
identity(12131415) // T is number
```

```
identity([4, 8, 15, 16, 23, 42]) // T is number[]
```

Встроенные обобщенные типы

```
const fib: Array<number> = [1, 1, 2, 3, 5]
```

```
// Argument of type 'string'
```

```
// is not assignable to parameter of type 'number'.
```

```
fib.push('1')
```

```
const map: Map<number, string> = new Map()
```

```
// Argument of type 'number'
```

```
// is not assignable to parameter of type 'string'.
```

```
map.set(1, 1)
```

Обобщенные интерфейсы

```
interface IStack<TItem> {  
    push(item: TItem)  
    pop(): TItem  
}
```

```
let numStack: IStack<number> = [1, 2, 3]
```

Обобщенные интерфейсы

```
interface IStack<number> {  
    push(item: number)  
    pop(): number  
}
```

```
let numStack: IStack<number> = [1, 2, 3]
```

Обобщенные типы

```
type AsyncResult<TResult> = Promise<TResult> | TResult
```

```
let result: AsyncResult<string> = Promise.resolve('200')
```

```
let result: AsyncResult<string> = '200'
```


Обобщенные типы

```
type AsyncResult<string> = Promise<string> | string
```

```
let result: AsyncResult<string> = Promise.resolve('200')
```

```
let result: AsyncResult<string> = '200'
```

Обобщенные классы

```
class Stack<TItem> implements IStack<TItem> {  
    private state: TItem[]  
  
    constructor() {  
        this.state = []  
    }  
  
    push(item: TItem) {  
        this.state.push(item)  
    }  
  
    pop(): TItem {  
        return this.state.pop()  
    }  
}
```

Обобщенные классы

```
class Stack<TItem> implements IStack<TItem> {  
    private state: TItem[] = []  
  
    push(item: TItem) {  
        this.state.push(item)  
    }  
  
    pop(): TItem {  
        return this.state.pop()  
    }  
}
```

Обобщенные типы

```
interface ISwim {  
    swim()  
}
```

```
class Dog implements ISwim {  
    swim() { ... }  
}
```

```
class Duck implements ISwim {  
    swim() { ... }  
}
```

Ограничения на обобщенные типы

```
function swimTogether<
    T1 implements ISwim,
    T2 implements ISwim
>(firstPal: T1, secondPal: T2) {
    firstPal.swim()
    secondPal.swim()
}
```



Обобщенные типы

```
type TypeName<T> =  
  T extends string ? 'string' :  
  T extends number ? 'number' :  
  T extends boolean ? 'boolean' :  
  T extends undefined ? 'undefined' :  
  T extends Function ? 'function' :  
  'object'
```

Обобщенные типы

```
type TypeName<string> =  
  string extends string ? 'string' :  
  T extends number ? 'number' :  
  T extends boolean ? 'boolean' :  
  T extends undefined ? 'undefined' :  
  T extends Function ? 'function' :  
  'object'
```


Обобщенные типы

```
type TypeName<number> =  
  number extends string ? 'string' :  
  number extends number ? 'number' :  
  T extends boolean ? 'boolean' :  
  T extends undefined ? 'undefined' :  
  T extends Function ? 'function' :  
  'object'
```

```
// TODO:  
// Рассказать больше о TypeScript  
// Увидимся в следующей части
```

Ссылки

[TypeScript Playground](#)

[TypeScript Handbook](#)

[TypeScript Deep Dive](#)

[Андрей Морозов \(Яндекс\) — Типизация](#)

[Андрей Старовойт \(JetBrains\) — Эволюция TypeScript](#)

Спасибо!

Вопросы?