Typescript

استاد درس: دکتر آرش شفیعی

پدید آورندگان: محمدامین صابری_۹۹۳۶۲۳۰۲۶، ابوالفضل شیشه‌گر_۹۹۳۶۲۳۰۲۵

دانشکده مهندسی کامپیوتر

دانشگاه اصفهان

پاییز ۱۴۰۳

فهرست مطالب

مقدمه:

سرعت و سادگی زبان برنامه نویسی جاوا اسکریپت از مزیت‌های اصلی این زبان به حساب می‌آید، اما بیشتر کتاب‌ها و مستندات علمی روی این قضیه تأکید دارند که جاوا اسکریپت زبانی کامل نیست. ولی با این حال امروزه جاوا اسکریپت چه در برنامه نویسی سمت سرور و چه در برنامه نویسی سمت کلاینت کاربردهای خاص خودش را دارد. امروزه تایپ اسکریپت به عنوان راه‌حلی برای رفع محدودیت‌های جاوا اسکریپت توسعه یافته است تا کم و کاستی‌های جاوا اسکریپت را به عنوان زبانی «بدون نیاز به تعیین صریح نوع» (Loosely Typed) پشت سر بگذارد.

تایپ اسکریپت جزو زبان‌های برنامه نویسی متن باز محسوب می‌شود. TypeScript یک ابرمجموعه نحوی از جاوا اسکریپت است که تایپ ایستا را اضافه می‌کند. این زبان توسط شرکت مایکروسافت گسترش یافت. TypeScript یک "Syntactic Superset" است؛ به این معنی که همان نحو پایه جاوا اسکریپت را به اشتراک می‌گذارد، اما چیزی را به آن اضافه می کند.

تایپ‌اسکریپت زبان برنامه‌نویسی‌ای است که توسط مایکروسافت توسعه‌داده می‌شود. تایپ‌اسکریپت پیشرفت‌داده‌شده جاوااسکریپت است و بررسی نوع قوی و شئ‌گرایی واقعی را ذاتاً به آن زبان می‌افزاید. آندرس هجلزبرگ پدر سی‌شارپ در توسعه تایپ‌اسکریپت فعالیت داشته‌است.

پارادایم برنامه‌نویسی	**Multi-paradigme** **اسکریپت نویسی، شئ گرا، ساخت یافته، دستوری، دستوری، تابعی، جنریک**
طراحی شده توسط	Microsoft
توسعه‌دهنده	Microsoft
ظهوریافته در	۱ اکتبر ۲۰۱۲؛ ۱۱ سال پیش
انتشار پایدار	۲/۳ ۲۷ آوریل ۲۰۱۷؛ ۶ سال پیش
پروانه	پروانه آپاچی
پسوند(های) نام پرونده	.ts, .tsx
وبگاه	www.typescriptlang.org
متأثر از	جاوااسکریپت، جاوا، سی شارپ

تاریخچه:

TypeScript تاریخچه‌ای جالب و مهم در دنیای برنامه‌نویسی دارد. TypeScript یک زبان برنامه‌نویسی تایپ‌سیستم دار است که توسط Microsoft توسعه داده شده و اولین بار در سال ۲۰۱۲ منتشر شد.

۱. **پیش‌نمونه‌ها و تاثیرهای اولیه:**

قبل از ایجاد TypeScript، مایکروسافت تلاش‌های زیادی در زمینه توسعه و بهبود JavaScript داشت. اولین تلاش‌هایی که به TypeScript انجام شد، از جمله CScript و JScript بودند. اما این تلاش‌ها موفقیت زیادی نداشتند.

۲. **پروژه Volta**:

توسعه TypeScript با پروژه با نام "Volta" شروع شد. این پروژه متمرکز بر ایجاد یک زبان برنامه‌نویسی بر پایه JavaScript با قابلیت تایپ‌سیستم بود. تیم توسعه TypeScript به رهبری "آندرس هایلسبرگ" (Anders Hejlsberg)، که قبلاً در توسعه زبان‌های برنامه‌نویسی معروفی نظیر C# و Delphi نقش داشته بود، به این پروژه پیوست.

۳. **پیش‌نمونه‌های ابتدایی:**

ابتدا، پیش‌نمونه‌های ابتدایی از TypeScript توسط تیم توسعه ساخته شدند و نسخه‌های اولیه تست شدند. این پیش‌نمونه‌ها نشان دادند که توسعه‌دهندگان به راحتی می‌توانند کد JavaScript خود را به TypeScript تبدیل کنند و از تایپ‌سیستم قدرتمند آن بهره‌برند.

۴. **آغاز عمومی:**

در سال ۲۰۱۲، TypeScript به صورت عمومی منتشر شد. این منتشر برای اولین بار به توسعه‌دهندگان اجازه داد که از این زبان تایپ‌سیستم دار برای توسعه برنامه‌های وب استفاده کنند.

۵. **گسترش و جامعه TypeScript**:

از آن زمان، TypeScript به سرعت جلب توجه توسعه‌دهندگان و شرکت‌های بزرگ شده است. این زبان به طور فزاینده در پروژه‌های متعدد وب و برنامه‌نویسی از جمله Angular، React، و Node.js مورد استفاده قرار گرفته است. همچنین، جامعه TypeScript بزرگ و فعالی دارد که باعث توسعه و بهبود مداوم این زبان شده است.

۶. **تکامل TypeScript**:

از انتشار اولیه تا به امروز، TypeScript تغییرات و بهبودهای متعددی داشته است. ویژگی‌های جدید اضافه شده و تایپ‌سیستم TypeScript بهبود یافته است.

TypeScript به توسعه‌دهندگان این امکان را می‌دهد که برنامه‌های جاوااسکریپتی پیچیده‌تر و ایمن‌تری توسعه دهند و از امکاناتی مانند تایپ‌سیستم و IntelliSense بهره‌برند. این زبان بهبود قابلیت‌های توسعه و نگهداری کد وب را فراهم کرده و به توسعه‌دهندگان امکان تشخیص اشکالات کد در مراحل ابتدایی توسعه را می‌دهد.

خوانایی :

۱. **سادگی:**

- **استفاده آسان از تایپ‌ها:** TypeScript به برنامه‌نویسان امکان می‌دهد تا تایپ‌های متغیرها و پارامترها را به صورت صریح تعریف کنند. این ویژگی باعث می‌شود که کد بهبود یابد و خوانایی بالاتری داشته باشد.
- **مدیریت ورودی و خروجی توابع:** TypeScript با تایپ‌های ورودی و خروجی توابع کمک می‌کند تا برنامه‌نویسان بهتر درک کنند که یک تابع چه نوع اطلاعات را می‌پذیرد و چه نوع مقادیر را برمی‌گرداند.
- **مثال:**

توضیحات نوع ساده: خوب//

```
let age: number;
```

توضیحات نوع پیچیده: بد//

```
let x: Map<string, Array<{ id: number, name: string }>>;
```

۲. **تعامد:**

- **استفاده از استانداردهای نام‌گذاری:** TypeScript از استانداردهای نام‌گذاری منطقی برای متغیرها، توابع و کلاس‌ها استفاده می‌کند. این استانداردها باعث می‌شود که تعامد کد افزایش یابد.
- **استفاده از ویژگی‌های ES6 :** TypeScript به برنامه‌نویسان امکان می‌دهد از ویژگی‌های جدید ES6 و بالاتر استفاده کنند که به تعامد کد کمک می‌کند.
- **مثال ۱:**

استفاده از camelCase: خوب//

```
let myVariableName: string;
```

ترکیب بی ترتیب حروف بزرگ و کوچک: بد//

```
let MyVaRiAbLeNAME: string;
```

- **مثال ۲:** یکی از مثال‌های معروف ES6 که می‌توانید در TypeScript نیز استفاده کنید، استفاده از توابع Arrow (Arrow Functions) است. توابع Arrow یک ویژگی از ES6 هستند که از نحوه‌ی تعریف توابع در JavaScript ساده‌تر می‌کنند. در TypeScript، می‌توانید از این ویژگی استفاده کنید.

```
//تعریف یک تابع Arrow
const add = (a: number, b: number): number => {
 return a + b;
};

//فراخوانی تابع
const result = add(3, 5);
console.log(result); // خروجی: ۸
```

در این مثال، ما یک تابع Arrow به نام "add" تعریف کرده‌ایم که دو عدد را جمع می‌کند و نتیجه را برمی‌گرداند. توابع Arrow با استفاده از عملگر => تعریف می‌شوند و می‌توانند توضیحات نوع داده‌ای را نیز داشته باشند. این ویژگی از ES6 باعث می‌شود تعریف توابع در TypeScript واضحتر و ساده‌تر شود.

۳. نوع‌های داده‌ای:

- **سیستم تایپ قوی:** TypeScript دارای یک سیستم تایپ قوی است که به برنامه‌نویسان اجازه می‌دهد تا انواع داده را به صورت صریح تعریف کنند. این سیستم باعث می‌شود که کدها برخورد خوبی با خطاهای نوعی داشته باشند و کد نوعاً تمیزتری داشته باشد.
- **تایپ‌های سفارشی:** شما می‌توانید تایپ‌های سفارشی تعریف کنید که باعث افزایش انعطاف‌پذیری و نوعی در کد می‌شود.
- **مثال:**

```
//استفاده از توضیحات نوع: خوب
function calculateArea(width: number, height: number): number {
 return width * height;
}

//عدم استفاده از توضیحات نوع: بد
function calculateArea(width, height){
 return width * height;
}
```

۴. **طراحی نحوی:**

- **استفاده از کلاس‌ها و اینترفیس‌ها:** TypeScript از کلاس‌ها و اینترفیس‌ها به عنوان ابزارهایی برای طراحی کد استفاده می‌کند. این ویژگی‌ها باعث انتزاع درست و سازماندهی کد می‌شوند.
- **پیگیری اصول SOLID:** TypeScript برنامه‌نویسان امکان می‌دهد اصول SOLID را به راحتی پیاده کنند و کد قابل توسعه و نگهداری ایجاد کنند.
- **مثال:**

```
ساختار قابل فهم: خوب//
if (condition) {
عملیات ۱//
عملیات ۲//
}

ساختار نامنطقی: بد//
if (condition)
عملیات ۱//
عملیات ۲//
```

قابلیت اطمینان:

برای ارزیابی قابلیت اطمینان (Reliability) زبان TypeScript با چهار معیار مختلف، می‌توانید به موارد زیر توجه کنید:

۱. **Type Checking (بررسی نوع):**

TypeScript به عنوان یک زبان برنامه‌نویسی تایپ‌سیستم معروف است و به برنامه‌نویسان اجازه می‌دهد تا نوع داده‌های متغیرها و پارامترهای توابع را تعیین کنند. این ویژگی می‌تواند به تشخیص و جلوگیری از بسیاری از خطاهای نوعی در زمان اجرا کمک کند. به عبارت دیگر، TypeScript از این نظر معیار خوبی برای افزایش قابلیت اطمینان کد است.

مثال:

```
function multiply(a: number, b: number): number {
 return a * b;
}

const result = multiply(5, "2"); // خطا: نوع داده ورودی نادرست است
```

در این مثال، تابع multiply انتظار دارد که دو عدد به عنوان ورودی دریافت کند. اما اگر شما یکی از ورودی‌ها را به عنوان یک رشته ارسال کنید، TypeScript از بررسی نوع به عنوان یک معیار برای اطمینان از درستی کد استفاده می‌کند و یک خطای کامپایل ایجاد می‌شود.

۲. **Exception Handling (مدیریت استثناء):**

TypeScript همچنین از نظر مدیریت استثناءها (exceptions) توانمندی‌های خوبی دارد. شما می‌توانید استثناءها را با استفاده از try-catch بلاک‌ها مدیریت کنید. این ویژگی می‌تواند به کدهایی که با مشکلات در زمان اجرا مواجه می‌شوند، قابلیت اطمینان بیشتری بدهد.

مثال:

```
try {
کدی که ممکن است استثناء پرتاب کند//
 throw new Error("یک استثناء اتفاق افتاد!");
} catch (error) {
 console.error("خطا رخ داد: " + error.message);
}
```

در این مثال، یک تست استثناء پرتاب می‌شود و سپس با استفاده از بلاک catch، خطای مدیریت می‌شود. این مثال نشان می‌دهد که TypeScript به شما امکان مدیریت استثناءها و افزایش قابلیت اطمینان از طریق مدیریت خطاها می‌دهد.

۳. **Aliasing (تبدیل نوعی نامگذاری):**

TypeScript اجازه می‌دهد تا نوع‌ها را با استفاده از Type Aliasing تعریف کنید. این ویژگی به شما کمک می‌کند که کدها را خواناتر کنید و نوع‌ها را با نام‌های معنادار توصیف کنید. این می‌تواند به خوانایی و قابلیت اطمینان کد کمک کند.

مثال:

```
type Point = { x: number; y: number };

function distance(p1: Point, p2: Point): number {
 const dx = p1.x - p2.x;
 const dy = p1.y - p2.y;
 return Math.sqrt(dx * dx + dy * dy);
}

const pointA: Point = { x: 0, y: 0 };
const pointB: Point = { x: 3, y: 4 };

const dist = distance(pointA, pointB);
console.log(`فاصله بین دو نقطه: ${dist}`);
```

در این مثال، یک تعریف نوع با نام "Point" ایجاد می‌شود که شامل دو ویژگی عددی x و y است. این نوع تعریف نوعی نامگذاری است که می‌تواند کد را خواناتر کند و قابلیت اطمینان را افزایش دهد.

۴. **Readability (خوانایی):**

خوانایی کد یکی از جنبه‌های مهم قابلیت اطمینان است. TypeScript با امکانات مانند Type Aliasing و توجه به نوع‌ها به خوانایی کد کمک می‌کند. همچنین ویژگی‌های تایپ‌سیستم به اصطلاح کد را "خواناتر" می‌کنند، چون توضیح می‌دهند که هر متغیر یا پارامتر چه نوع داده‌ای دارد.

در کل، TypeScript به خوبی می‌تواند توسعه دهندگان در افزایش قابلیت اطمینان کدها کمک کند، از طریق بررسی نوع‌ها، مدیریت استثناءها، تبدیل نوعی نامگذاری و افزایش خوانایی کد.

مثال:

```
function calculateCircleArea(radius: number): number}
 return Math.PI * radius ** 2;
{

const r = 5;
const area = calculateCircleArea(r);

console.log(`مساحت دایره با شعاع ${r} واحد: ${area}`);
```

این مثال نشان می‌دهد که TypeScript با استفاده از نوع‌ها و توضیحات خواناتری ارائه می‌دهد. از این روش‌ها می‌توان برای افزایش خوانایی کد و افزایش قابلیت اطمینان استفاده کرد.

این مثال‌ها نمایشی ساده از چگونگی استفاده از مفاهیم Type Checking، Exception Handling، Aliasing، و Readability در TypeScript هستند. این مفاهیم به افزایش قابلیت اطمینان کد شما کمک می‌کنند.

هزینه:

TypeScript یک زبان برنامه‌نویسی سطح بالا (High-Level) است. یعنی این زبان از انتزاعات و توضیحات بالا برای توسعه برنامه‌ها استفاده می‌کند و تمرکز اصلی آن بر راحتی توسعه برنامه‌ها و قابلیت‌هایی برای مدیریت پیچیدگی و توسعه کدها است.

در واقع، TypeScript یک زبان فرعی از جاوااسکریپت (JavaScript) است و توسط کامپایلر به کد جاوااسکریپت ترجمه می‌شود. این به برنامه‌نویسان اجازه می‌دهد که از ویژگی‌های سطح بالا مانند تایپ‌ها، کلاس‌ها، ماژول‌ها و انواع داده سفارشی برای توسعه برنامه‌ها بهره ببرند و به کدهای JavaScript که در محیط مرورگرها اجرا می‌شوند ترجمه شوند.

با این حال، باید توجه داشته باشید که ترجمه TypeScript به JavaScript به منزله نزدیک‌تر شدن به سطح پایین‌تر نیست. TypeScript همچنان از ویژگی‌ها و انتزاعات سطح بالا برای توسعه استفاده می‌کند.

قابلیت جابجایی:

TypeScript به عنوان یک زبان برنامه‌نویسی که به کد جاوااسکریپت ترجمه می‌شود، دارای برخی ویژگی‌هایی است که قابلیت انتقال و استفاده در سیستم‌های مختلف را بهبود می‌بخشد:

۱. مستقل از سیستم عامل (Platform-Independent): کدهای TypeScript بعد از ترجمه به کد جاوااسکریپت، در هر محیط و سیستم عاملی که قادر به اجرا کردن JavaScript باشد، قابل اجرا است. این امر به برنامه‌های TypeScript امکان انتقال به محیط‌های مختلف را می‌دهد.

۲. پشتیبانی از محیط‌های مختلف: TypeScript توسط مرورگرها، محیط‌های اجرایی Node.js و سایر محیط‌ها پشتیبانی می‌شود. این به توسعه‌دهندگان امکان انتقال کدهای خود بین محیط‌های مختلف را می‌دهد.

۳. مدیریت وابستگی‌ها TypeScript :(Dependency Management) با استفاده از مدیریت کننده وابستگی‌هایی مانند npm یا yarn از توزیع و مدیریت وابستگی‌های پروژه‌ها پشتیبانی می‌کند. این به شما امکان مدیریت و انتقال پروژه‌های خود به سیستم‌های مختلف را می‌دهد.

۴. امکان تولید فایل‌های ترجمه TypeScript :(Output Files) به شما امکان تولید فایل‌های JavaScript با پسوند .js از کدهای خود می‌دهد. این فایل‌ها می‌توانند به سادگی به هر مکان منتقل شوند و در هر محیطی اجرا شوند.

۵. پشتیبانی از محیط‌های ابری TypeScript :(Cloud Environments) به خوبی در محیط‌های ابری مانند AWS Lambda، Google Cloud Functions و Azure Functions استفاده می‌شود. این به توسعه‌دهندگان اجازه می‌دهد کدهای خود را در محیط‌های ابری اجرا کنند.

با توجه به این ویژگی‌ها، TypeScript به صورت گسترده در پروژه‌ها و محیط‌های مختلف مورد استفاده قرار می‌گیرد و امکان انتقال و پرتابلیتی بالایی دارد. این امر به توسعه‌دهندگان امکان می‌دهد کدهای خود را به طریقی منعطف در سیستم‌ها و محیط‌های مختلف اجرا کنند.

عمومی یا خصوصی ؟:

TypeScript یک زبان برنامه‌نویسی عمومی (General-Purpose) است. این به این معناست که TypeScript برای توسعه برنامه‌های متنوع در زمینه‌های مختلف قابل استفاده است و به عنوان یک زبان تخصصی معین محدود نیست.

ویژگی‌ها و امکانات TypeScript، از جمله تایپ‌ها، کلاس‌ها، ماژول‌ها، توابع و ویژگی‌های متنوع دیگر، امکان توسعه برنامه‌ها در زمینه‌های مختلف از وب و موبایل تا برنامه‌های سروری را فراهم می‌کنند.

از جمله زمینه‌های مورد استفاده اصلی TypeScript می‌توان به موارد زیر اشاره کرد:

- توسعه وب: TypeScript برای توسعه وبسایت‌ها و برنامه‌های تحت وب به کار می‌رود.
- توسعه اپلیکیشن‌های موبایل: با استفاده از فریمورک‌ها مانند React Native، توسعه‌دهندگان می‌توانند از TypeScript برای ساخت اپلیکیشن‌های موبایل استفاده کنند.
- توسعه برنامه‌های سمت سرور: TypeScript به صورت گسترده در توسعه برنامه‌های سمت سرور با استفاده از Node.js مورد استفاده قرار می‌گیرد.
- توسعه ابزارها و کتابخانه‌ها: برنامه‌نویسان می‌توانند ابزارها و کتابخانه‌های TypeScript را برای سایر توسعه‌دهندگان ایجاد کنند.
- توسعه بازی: TypeScript می‌تواند در توسعه بازی‌های ویدئویی و بازی‌های وب مورد استفاده قرار گیرد.

با توجه به این موارد، TypeScript به عنوان یک زبان برنامه‌نویسی عمومی، در انواع مختلف پروژه‌ها و زمینه‌های توسعه به کار می‌رود و از پرتابلیتی و انعطاف بالایی برخوردار است.

دستوری یا اعلانی ؟:

زبان TypeScript یک زبان برنامه‌نویسی **اعلانی (declarative)** است. زبان‌های اعلانی تمرکز بر توصیف اهداف و نتایج مورد نظر برنامه‌نویسی دارند، به جای توصیف مراحل اجرا یا دستورات به صورت دقیق.

در TypeScript، شما توصیف نوع داده‌ها، ساختارهای داده، توابع، و روابط بین اجزاء مختلف برنامه را با استفاده از توضیحات نوع‌ها و تایپ‌ها انجام می‌دهید. این توضیحات نوع‌ها از قوانین و اصول خاصی پیروی می‌کنند که به کامپایلر TypeScript اطلاع می‌دهند که چگونه تایپ‌ها و ارتباطات بین اجزاء برنامه باید باشند.

به عبارت دیگر، شما تعریف می‌کنید که "چه چیزی" باید انجام شود (توصیف اعلانی) به جای اینکه بگویید "چگونه انجام دهید" (برنامه‌نویسی دستوری). این اعلان‌گری باعث افزایش خوانایی کد و افزایش قابلیت اطمینان آن می‌شود، زیرا کامپایلر توسط تایپ‌ها و توضیحات نوع‌ها اشکالات نوعی و مشکلات احتمالی در کد را به شما اطلاع می‌دهد.

پارادایم‌ها:

TypeScript در اصل یک زبان برنامه‌نویسی چند منظوره (multi-paradigm) است و به شما امکان استفاده از چندین پارادایم برنامه‌نویسی را می‌دهد. از جمله پارادایم‌هایی که در TypeScript می‌توان به آنها اشاره کرد، عبارتند از:

۱. **پارادایم شیءگرایی (Object-Oriented Programming - OOP)**: TypeScript از اصول شیءگرایی مانند کلاس‌ها، وراثت، انکپسولیشن و پلی‌مورفیسم پشتیبانی می‌کند. شما می‌توانید کلاس‌ها و اشیاء ایجاد کنید و از مفاهیم شیءگرایی برای ساختاردهی کدها و داده‌ها استفاده کنید.

۲. **پارادایم تابعی (Functional Programming - FP)**: TypeScript نیز از ویژگی‌ها و مفاهیم تابعی پشتیبانی می‌کند. این شامل توابع بالقوه (نقشه‌برداری)، توابع بالقوه‌ای (کاهشی)، برنامه‌نویسی تابعی و تجزیه و تحلیل داده‌ها با توابع می‌شود. TypeScript به شما امکان تعریف توابع بالقوه و تجربه برنامه‌نویسی تابعی می‌دهد.

۳. **پارادایم مختلط (Mixed Paradigms)**: TypeScriptبه شما اجازه می‌دهد تا از هر دو پارادایم شیءگرایی و تابعی به صورت ترکیبی استفاده کنید. این به شما امکان می‌دهد کدهایی با ترکیب ویژگی‌های هر دو پارادایم ایجاد کنید.

۴. **پارادایم کامپایلر (Compiler Paradigm)**: TypeScriptبه عنوان یک زبان کامپایل شده نیز مطابق با اصول پارادایم کامپایلر عمل می‌کند.

به طور خلاصه، TypeScript یک زبان چند منظوره است که از چندین پارادایم برنامه‌نویسی پشتیبانی می‌کند و به شما امکان می‌دهد از آن‌ها به ترتیب مناسب برای پروژه‌های خود استفاده کنید.

روش پیاده سازی:

TypeScript با روش "Compilation" پیاده‌سازی شده است. این به این معناست که کدهای TypeScript به کدهای جاوااسکریپت ترجمه می‌شوند و سپس اجرا می‌شوند. ترجمه کد TypeScript به کد جاوااسکریپت توسط کامپایلر TypeScript انجام می‌شود. این کامپایلر TypeScript به شما امکان می‌دهد که کدهای TypeScript را به کد معادل JavaScript ترجمه کرده و سپس این کد ترجمه شده را در محیط‌های مختلف اجرا کنید.

در واقع، کامپایلر TypeScript ترجمه کد TypeScript به جاوااسکریپت انجام می‌دهد تا برنامه‌های شما بتوانند در محیط‌هایی که JavaScript قابل اجرا است (مانند مرورگرها و محیط‌های اجرایی Node.js) به درستی اجرا شوند. این مکانیزم ترجمه از TypeScript به JavaScript امکان افزایش قابلیت اطمینان و کارایی کدهای تایپ‌دار TypeScript را فراهم می‌کند.