Introduction to TypeScript in Web Design

Introduction to TypeScript

What is TypeScript?

- A superset of JavaScript that introduces static typing, making code safer and more predictable
- Developed by Microsoft, transpiles to JavaScript, ensuring full compatibility across all browsers
- Popular in web design and front-end frameworks (like Angular, React)
- Used by Google, Slack, Airbnb, etc.

Benefits of TypeScript in Web Design

Error detection: Detects errors at compile time, preventing runtime crashes

Better tooling: Auto-completion, type-checking, and better code navigation (IntelliSense in VSCode)

Scalability: Easier to maintain larger projects

TypeScript Setup

Environment Setup:

`Node.js` and **`npm`** installation are prerequisites

Install TypeScript globally:

```
npm install -g typescript
```

TypeScript Setup

Environment Setup:

- Node.js and npm installation are prerequisites
  - On lab machines, we may have to (like Sass) install TS on a per-project basis

```
npm install -g typescript
```


Integrating TypeScript in a Web Project

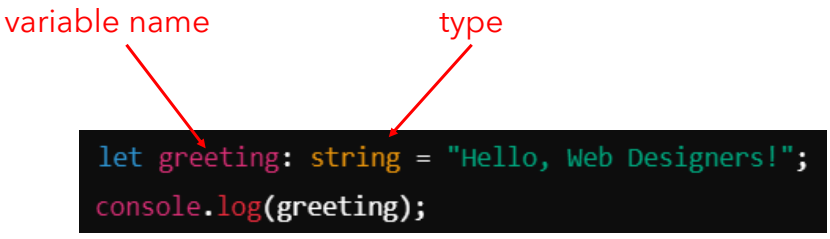
In a web design workflow, TypeScript can be used alongside popular frameworks like React or Angular

Set up TypeScript in a project:

```
tsc --init
```

Basic TypeScript file

Create a .ts file, then compile to .js:



TypeScript and JavaScript: Key Differences

JavaScript: Dynamically typed, types are determined at runtime

TypeScript: Statically typed, types are determined at compile-time

```
let number: number = 42;  // Type explicitly declared
```

Type Inference

TypeScript can infer types even without explicit annotations

```
let message = "Hello";  // inferred as string
```


Typescript - Advantages

Error Detection:

TypeScript catches errors early, making the code more reliable

Typescript - Advantages

Use of ES6 and Beyond:

TypeScript supports modern JavaScript features (ES6 and beyond), ensuring compatibility with all browsers by compiling to ECMAScript 5 or 6

Typescript - Advantages

Better OOP Support:

TypeScript provides more robust Object-Oriented Programming features like classes, interfaces, and inheritance

Core TypeScript Features for Web Development

13

Type Annotations

Explicitly declare variable types for better control

```
let isCompleted: boolean = true;
let count: number = 5;
let username: string = "John";
```

Interfaces

Define the shape of objects and ensure consistent data structures

```typescript
interface User {
  name: string;
  age: number;
}

let user: User = { name: "Jane", age: 28 };
```


Classes and Inheritance

Object-Oriented Programming in TypeScript, similar to JavaScript but with better type

```typescript
class Person {
  name: string;
  constructor(name: string) {
    this.name = name;
  }

  greet(): string {
    return `Hello, my name is ${this.name}.`;
  }
}

class Employee extends Person {
  position: string;

  constructor(name: string, position: string) {
    super(name);
    this.position = position;
  }

  greet(): string {
    return `${super.greet()} I am a ${this.position}.`;
  }
}
```

Generics in TypeScript

Create reusable components that work with any data type

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

let result = identity<number>(42);
```


TypeScript in Web Design Projects

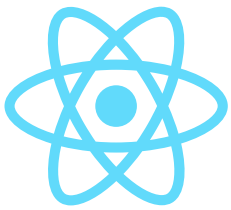
Using TypeScript with HTML and CSS

Example: Using TypeScript to manipulate the DOM

```typescript
const button = document.querySelector('button') as HTMLButtonElement;
button.addEventListener('click', () => {
  console.log('Button clicked!');
});
```


Frameworks

Typescript also integrates well with frameworks like React and Angular



Configuring TypeScript with tsconfig.json

Why tsconfig.json?

A configuration file that defines how TypeScript compiles your code

Key configurations:

Strict Mode:

Always use "strict": true for better error checking and safer code

```
{
  "compilerOptions": {
    "target": "ES6",
    "module": "commonjs",
    "strict": true,
    "outDir": "./dist"
  },
  "include": ["src/**/*"],
  "exclude": ["node_modules"]
}
```


Best Practices for TypeScript in Web Design

Use Type Inference:

Let TypeScript infer types when possible to keep the code clean

Modular Code:

Split your code into multiple files and modules to keep it manageable and organized

Strong Typing for Props and State (for React):

Always define types for component props and state to prevent common bugs

Best Practices for TypeScript in Web Design

Gradual Adoption:

TypeScript can be adopted gradually, especially for existing JavaScript projects

Leverage Tooling:

Use VSCode's TypeScript integration for better auto-completion, refactoring, and navigation

Summary

TypeScript adds value to web design by providing static typing, better tooling, and enhanced code scalability

Integrates seamlessly with modern frameworks like React and Angular

25