Angular Version Differences and Features (8 to 20)

This document compiles the key differences, features, and example snippets for Angular versions from 8 up to the experimental features proposed for Angular 20. It includes stable release highlights as well as preview and experimental APIs for future versions.

Chapter 1 : Angular 8 to Angular 9

- Angular 9 introduced the Ivy compiler and runtime as the default, bringing smaller bundle sizes and faster compilation.
- Improved type checking and build performance.
- Ivy enables better debugging and improved build errors.
- Angular 9 supports TypeScript 3.7 and Angular Package Format v9.
- Code snippet showing Ivy compilation enabling is internal, no direct code change needed.

Chapter 2 : Angular 9 to Angular 10

Key Features:
- Optional Stricter Settings on New Projects
- Updated TypeScript Support to 3.9
- Warnings about CommonJS Imports
- New Default Browser Configuration

Snippets illustrating Optional Strict Mode setup:

tsconfig.json snippet:
```
{
  "compilerOptions": {
    "strict": true
  }
}
```

angular.json snippet to enable strict type checking:
```
"strict": true
```

Chapter 3 : Angular 10 to Angular 11

Key Features:
- Faster Builds with Webpack 5 Support
- Updated Language Service
- Improved Hot Module Replacement (HMR)
- Automatic Inlining of Fonts

Example snippet for enabling HMR in Angular 11:

```
// main.ts
if (module['hot']) {
  module['hot'].accept();
  module['hot'].dispose(() => {
    // Cleanup before module replacement
  });
}
```

Chapter 4 : Angular 11 to Angular 12

Key Features:
- Ivy Everywhere (View Engine deprecated)
- Nullish Coalescing in Templates
- Inline Sass in Components
- Strict Mode enabled by default

Example snippet for Nullish Coalescing:

```
{{ user?.name ?? 'Guest' }}
```

Chapter 5 : Angular 12 to Angular 13

Key Features:
- Removal of View Engine
- TypeScript 4.4 Support
- Simplified Angular Package Format
- Support for ES2020

No specific code snippet needed; mainly internal build changes.

Chapter 6 : Angular 13 to Angular 14

Key Features:
- Typed Reactive Forms

- Standalone Components (experimental)
- Extended Template Diagnostics

Snippet: Typed Reactive Forms

```
import { FormControl } from '@angular/forms';

const nameControl = new FormControl<string>('');
```

Chapter 7 : Angular 14 to Angular 15

Key Features:
- Standalone Components stabilized
- Directive Composition API
- Functional Router Guards

Snippet: Standalone component usage

```
import { Component } from '@angular/core';

@Component({
  selector: 'app-hello',
  standalone: true,
  template: `<h1>Hello Angular 15!</h1>`
})
export class HelloComponent {}
```

Chapter 8 : Angular 15 to Angular 16

Key Features:
- Signals introduced (reactive primitives)
- Functional Component APIs
- New Reactive Forms API

Snippet: Using Signals

```
import { signal } from '@angular/core';
```

```
const count = signal(0);

function increment() {
  count.update(c => c + 1);
}
```

Chapter 9 : Angular 16 to Angular 17

Key Features:
- Control flow syntax (@if, @for)
- Deferrable views with @defer
- Native View Transitions support
- Model-based forms stabilized

Snippets:

Control flow syntax:

```
@if (user.loggedIn) {
  <div>Welcome, {{ user.name }}</div>
} @else {
  <div>Please log in.</div>
}

@for (item of items; track item.id) {
  <p>{{ item.name }}</p>
}
```

Deferrable view:

```
@defer (when userLoaded) {
  <user-profile [user]="user"></user-profile>
} @placeholder {
  <p>Loading...</p>
}
```

View Transitions setup:

```
import { provideAnimations } from '@angular/platform-browser/animations';
```

```
bootstrapApplication(AppComponent, {
  providers: [provideAnimations()],
});
```

Chapter 10 : Angular 17 to Angular 18 (Experimental)

Expected features:
- Extended Signals API with batch() and toObservable()
- Typed templates
- Incremental hydration for SSR
- Reactive router APIs
- Form API stabilization

Snippets:

Signals batch and toObservable:

```
batch(() => {
  count.set(5);
  count.update(v => v + 10);
});

const count$ = count.toObservable();
count$.subscribe(value => console.log(value));
```

Typed template example:

```
@for (item of items as ItemType) {
  <p>{{ item.name }}</p>
}
```

Incremental hydration setup:

```
import { provideServerRendering, provideIncrementalHydration } from '@angular/platform-server';

bootstrapApplication(AppComponent, {
  providers: [
    provideServerRendering(),
    provideIncrementalHydration()
  ]
});
```

Chapter 11 : Angular 18 to Angular 19 (Experimental)

Expected features:
- Async signals (asyncSignal)
- Declarative suspense (@suspense)
- DI integration with signals
- Optimized hydration and streaming SSR
- Native Web Components integration

Snippets:

AsyncSignal example:

```
const userData = asyncSignal(fetchUserData());
```

@suspense example:

```
@suspense (pendingTemplate: loadingTpl) {
  <user-profile [user]="userData()"></user-profile>
}
<ng-template #loadingTpl>
  <p>Loading user profile...</p>
</ng-template>
```

Web Components example:

```
@Component({
  selector: 'my-counter',
  standalone: true,
  template: `<button (click)="increment()">Count: {{ count() }}</button>`,
})
export class MyCounter {
  count = signal(0);
  increment() {
    this.count.update(c => c + 1);
  }
}
```

```
customElements.define('my-counter', defineCustomElement(MyCounter));
```

Chapter 12 : Angular 19 to Angular 20 (Speculative / Not Released)

As of now, Angular 20 is not officially released and no concrete features or code snippets are available.

Typically, future Angular versions build upon previous innovations with focus on performance, developer ergonomics, and more reactive programming features.

Stay tuned for official announcements from the Angular team.