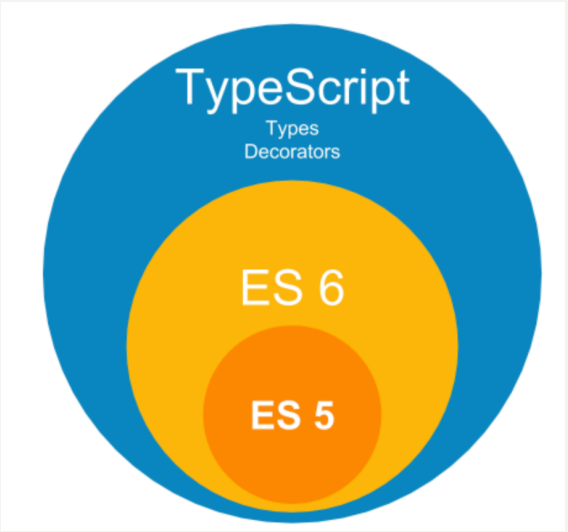
TYPESCRIPT

¿QUÉ ES TYPESCRIPT

- Es un superset de Javascript. Esto significa que los programas de JavaScript son programas válidos de TypeScript, a pesar de que TypeScript sea otro lenguaje de programación.
- Fue creado por Microsfot.
- Typescript es la solución a muchos de los problemas de JavaScript, está pensado para el desarrollo de aplicaciones robustas, implementando características en el lenguaje que nos permitan desarrollar herramientas más avanzadas para el desarrollo de aplicaciones.



¿QUÉ PROBLEMAS OCURRÍAN EN JAVASCRIPT QUE RESUELVE TYPESCRIPT?

- Errores porque una variable no está definida
- Errores porque el objeto no tiene la propiedad esperada
- Errores porque no se sabe como trabajan las funciones de otros compañer@s
- Errores porque se sobrescriben variables, clases, funciones, etc.
- Errores porque el caché del navegador mantiene los archivos de Javascript viejos.
- Errores porque colocamos las mayúsculas o minúsculas en lugares incorrectos.
- Errores porque el IDE no me ha avisado de que algo no se podía hacer.

LO PEOR DE TODO ES QUE NO NOS DAMOS CUENTA HASTA QUE NUESTRO CÓDIGO SE EJECUTA

¿CÓMO FUNCIONA TYPESCRIPT?

- Los archivos de TypeScript tiene extensión **.ts**
- Estos archivos no los interpreta el navegador. Deben ser compilados y traducidos a archivos .js
- Si conocemos Javascript se puede decir que conocemos el 80% de Typescript
- ¿Cómo compilo mi archivo .ts? Tengo que escribir en el terminal lo siguiente:
    - **tsc nombreapplicacion.ts** con esto nos generará nuestro archivo .js
    - **tsc nombreaplicacion.ts –w** con esto si queremos ponerlo en modo observador para no tener que estar siempre mandando a compilar
    - **tsc –init** con esto si no queremos estar siempre tecleando el nombre del archivo me compila todos los .ts

TIPOS BÁSICOS DE DATOS

- String
- Boolean
- Numérico
- Enum
- Any
- Never
- Null
- Indefinido
- Array y Tupla

Ejemplo

```
let e: string = "building";
let f: number = 300;
```

Mirar el manual de Typescript para principiantes

PARÁMETROS OPCIONES Y PARÁMETROS POR DEFECTO

```
let funcionAlCuadradoFlecha =
(numero:number, parametroxdefecto:string="prueba", parametroopcional?:string) => {
    return `El Cuadrado del número es ${ numero * numero } y ${ parametroxdefecto }`;
    }

    console.log(funcionAlCuadradoFlecha(3));
```

No es necesario mandar el parámetro opcional en la llamada a la función si no lo quiero utilizar. Sin embargo si quiero utilizarlo tendré también que poner en la llamada el parámetro que hay declarado anteriormente en este caso el **parámetroxdefecto.**

INTERFACES DE TYPESCRIPT

- Las interfaces son nombres dados a tipos de objetos. No solamente podemos declarar una interfaz sino que también podemos usarla como anotación de tipo.

```
interface Shape {
      name: string;
      width: number;
      height: number;
      color?: string;
}

function area(shape : Shape) {
      var area = shape.width * shape.height;
      return "I'm " + shape.name + " with area " + area + " cm squared";
}

console.log( area( {name: "rectangle", width: 30, height: 15} ) );
console.log( area( {name: "square", width: 30, height: 30, color: "blue"} ) );
```

¿qué pasaría si ejecuto console.log(area({width: 30, height: 15}));?

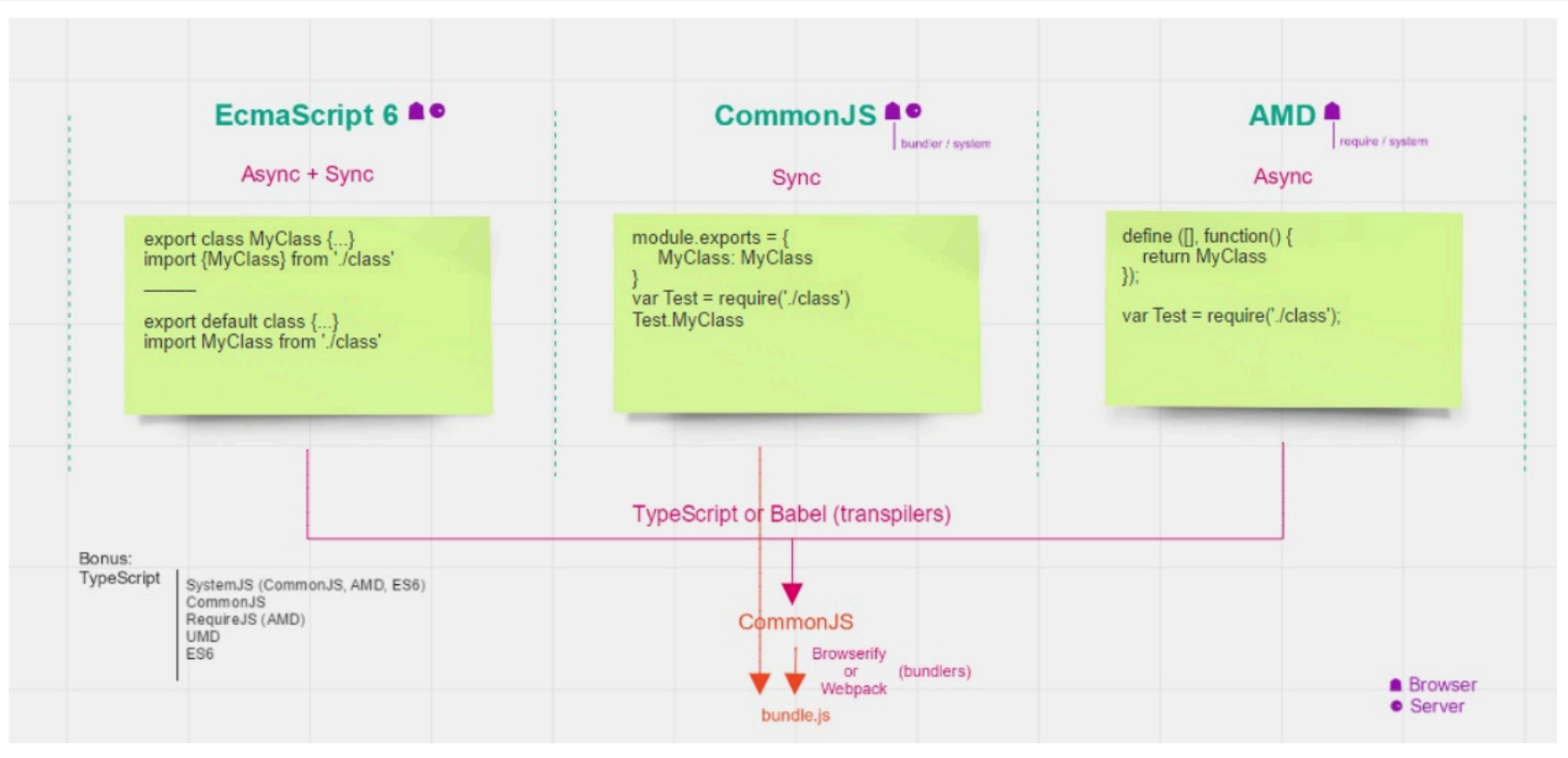
CLASES EN TYPESCRIPT

- TypeScript soporta clases y su implementación está cerca de la propuesta en ECMAScript 6

```
class Avenger {

  nombre:string = "Sin nombre";
  equipo:string = undefined;
  nombreReal:string = undefined;

  puedePelear:boolean = false;
  peleasGanadas:number = 0;

  constructor(  nombre:string, equipo:string, nombreReal:string  ){
    this.nombre = nombre;
    this.equipo = equipo;
    this.nombreReal = nombreReal;
  }

}

let antman:Avenger = new Avenger( "Antman","cap", "Scott Lang" );

console.log(antman);
```

MÓDULOS DE TYPESCRIPT

- En la actualidad existen tres tipos de cargas de módulos, los módulos son una manera de llamar a las clases o métodos que deseamos exportar para que otras clases puedan utilizarlas mediante importaciones.



MÓDULOS DE TYPESCRIPT. EJEMPLO

- Archivo xmen.class.js

```
export class Xmen{
  nombre:string;
  clave:string;

  constructor(nombre:string, clave:string){
    this.nombre = nombre;
    this.clave = clave;
  }

  imprimir(){
    console.log( `${ this.nombre } - ${ this.clave }` );
  };
}
```

- Archivo app.ts

```
import { Xmen } from "./clases/xmen.class"



let wolverine = new Xmen("Logan","Wolverine");



wolverine.imprimir();
```

DECORADORES DE TYPESCRIPT

- Un decorador es una función que, dependiendo de que cosa queramos *decorar*, sus argumentos serán diferentes. Usan la forma @expression donde expression evaluará la función que será llamada

```
function consola(  constructor:Function  ){
  console.log( constructor );
}



@consola
class Villano{

  constructor( public nombre:string ){

  }


}
```

Los decoradores en las clases lo que hacen es mandar directamente el constructor a la función

Para poder utilizar decoradores tenemos que
Modificar el archivo tsconfig.json e incluir los siguiente:

"experimentalDecorators": true