HTML

Cast Elements to HTMLElements	`elems as HTMLCollectionOf<HTMLElement>`

Javascript

Find text in strings	`"Hello world".includes("world"); // True`
String to Int	`var a = parseInt("10")`
Int to String	`value.toString()` or `"" + value` or `String(value)`
Switch statement	``` switch(expression) { case n: code block break; default: code block } ```
declare var as function	`var a: Function`
call object method as parameter	``` function bar(obj: Foo, func: string) { obj[func](); } ```
Find string in array of strings	``` var strs = ['abc', 'def', 'ghi']; strs.find(str => { return str === 'abc' }); ```
Creating a Promise	``` myProm = new Promise((res, rej) => { // something asynchronous then call: res(someValue); // or rej("failure reason"); }) ```

Javascript (cont)

Creating an Observable	``` let obs = Observable.create(obs => { setTimeout(() => { obs.next("data to send"); console.log("am done"); obs.complete();// we are done // obs.error(new Error("msg")); }, 2000); }) ```
Loop through object	``` for (const v of Object.keys(vs)) { r.push(vs[v]) } ```
Parse string to object	`JSON.parse()`
Is variable an array	`Array.isArray()`
case statement	``` switch(expression) { case n: code block break; default: code block } ```
Test for empty object	`Object.keys(obj).length === 0`

Angular CLI

Build in test mode	`ng build --environment=test`
Start server	`lite-server`
lite-server configuration	`"files": ["./dist/*.{js}"], "server": { "baseDir": "./dist" }`

Python

Ternary operator	`a if condition else b`

Remove array element	`remove first matching value`
	`[0, 2, 3, 2].remove(2) => [0, 3, 2]`
	`remove a specific index`
	`[3, 2, 2, 1].del a[1] => [3, 2, 1]`
	`return the removed element`
	`[4, 3, 5].pop(1) returns 3 => [4, 5]`

Item in list	`someElement in someList`
Print tuple	`print "{0}, {1}, {2}".format(*[1, 2, 3])`
Flatten tuple	`[el for tup in ((1, 2), (3,)) for el in tup]` `=> [1, 2, 3]`
Index in list	`["foo", "bar", "baz"].index("bar")`
regex replace	`re.sub(pattern, repl, string, count=0, flags=0)`
print array	`print ', '.join(names)`
for loop with index	`for idx, val in enumerate(list):`
for in - val is the value (not index)	`for val in list`
int to string	`str(number)`
find substring	`string.find('substring')`

Angular

```
@NgModule({
    providers
    declarations
    imports


    exports



    entryComponents



    bootstrap


    schemas
    id
})
```

- injectable objects, can appear in the constructor
- list of directives/pipes
- list of modules whose exported directives/pipes should be available to templates in this module
- directives/pipes/modules used in template of components part of module importing this module
- list of components that should be compiled when this module is defined
- components that should be bootstrapped when this module is bootstrapped