Table of Contents

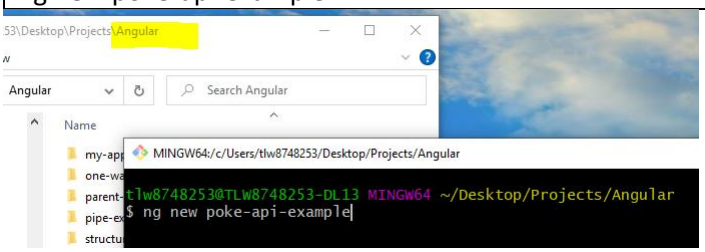
Lecture and project for HTTP request processing

Create new project poke-api-example

Open Git Bash window in your Angular folder

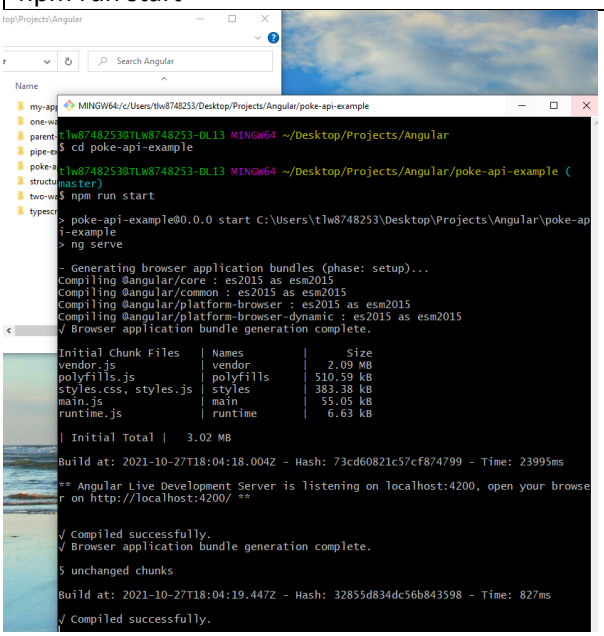
Create the poke-api-example project

```
ng new poke-api-example
```



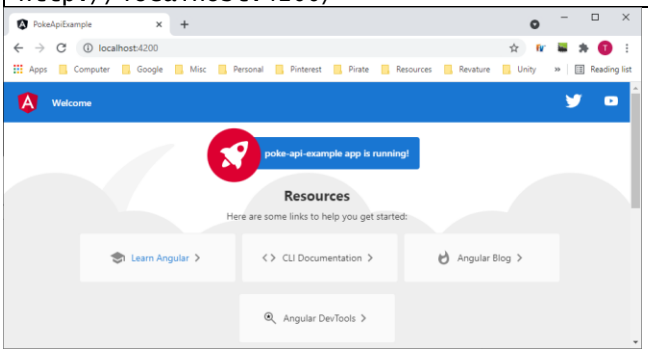
Start the application

```
cd poke-api-example
npm run start
```

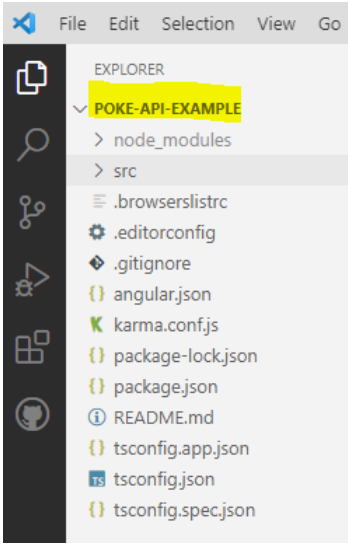


Open the webpage

```
http://localhost:4200/
```



Open the project in VS Code



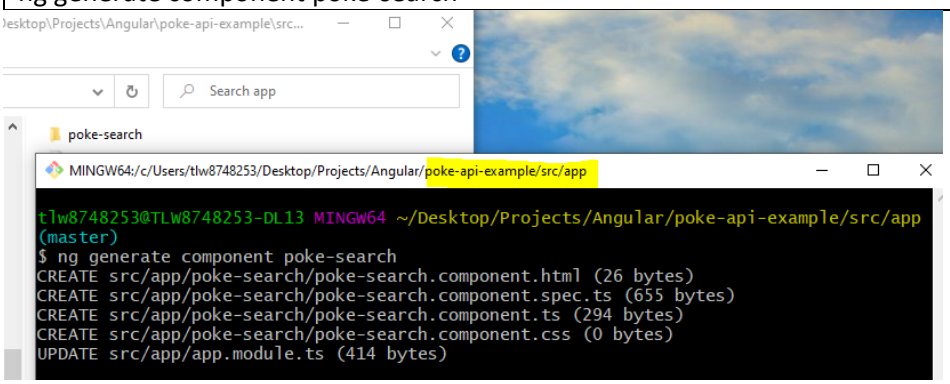
Modify the poke-api-example project

Create new project components

In a Git Bash window in the directory poke-api-example\src\app

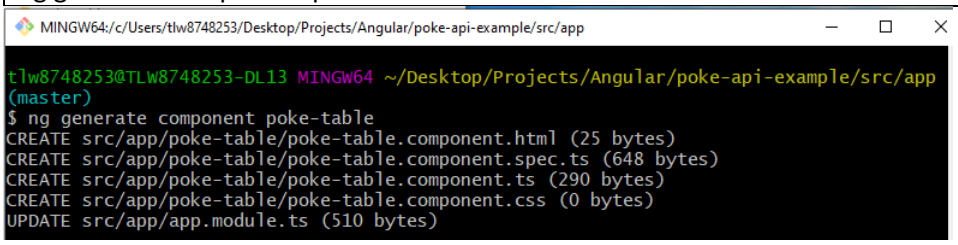
Create the new component poke-search

```
ng generate component poke-search
```



Create a new component poke-table

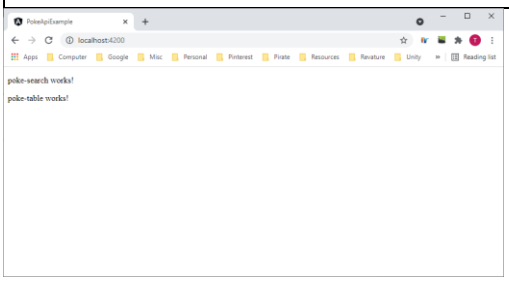
```
ng generate component poke-table
```



Update project files

Replace all code in app.component.html

```
<app-poke-search></app-poke-search>
<app-poke-table></app-poke-table>
```



Replace default code in poke-search.component.html

```
<input [(ngModel)]="id" type="number" />

<button (click)="emitSearch()">Get Pokemon</button>
```

Update code in poke-search.component.ts

Change:

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

To:

```
import { Component, EventEmitter, OnInit, Output } from '@angular/core';
```

Add:

```
@Output()
search: EventEmitter<number> = new EventEmitter();
id: number = 0;

emitSearch() {
  this.search.emit(this.id);
}
```

app.component.html M × | TS poke-search.component.ts U ×

src > app > poke-search > TS poke-search.component.ts > ...

```
1  import { Component, EventEmitter, OnInit, Output } from '@angular/core';
2
3  @Component({
4    selector: 'app-poke-search',
5    templateUrl: './poke-search.component.html',
6    styleUrls: ['./poke-search.component.css']
7  })
8  export class PokeSearchComponent implements OnInit {
9
10   @Output()
11   search: EventEmitter<number> = new EventEmitter();
12   id: number = 0;
13
14   constructor() { }
15
16   ngOnInit(): void {
17   }
18
19   emitSearch() {
20     this.search.emit(this.id);
21   }
22
23 }
```

Update code in app.module.ts

Add:

```
import { FormsModule } from '@angular/forms';
import { HttpClientModule } from '@angular/common/http';
  FormsModule,
  HttpClientModule
```

app.component.html M app.module.ts M

src > app > app.module.ts > AppModule

```
1  import { NgModule } from '@angular/core';
2  import { FormsModule } from '@angular/forms';
3  import { BrowserModule } from '@angular/platform-browser';
4
5  import { HttpClientModule } from '@angular/common/http';
6
7  import { AppComponent } from './app.component';
8  import { PokeSearchComponent } from './poke-search/poke-sear
9  import { PokeTableComponent } from './poke-table/poke-table.
10
11 @NgModule({
12   declarations: [
13     AppComponent,
14     PokeSearchComponent,
15     PokeTableComponent
16   ],
17   imports: [
18     BrowserModule,
19     FormsModule,
20     HttpClientModule
21   ],
22   providers: [],
23   bootstrap: [AppComponent]
24 })
25 export class AppModule { }
```

Update the new code in poke-search.component.html

Change:

```
<app-poke-search></app-poke-search>
```

To:

```
<app-poke-search (search)="onSearch($event)"></app-poke-search>
```

app.component.html M app.module.ts M

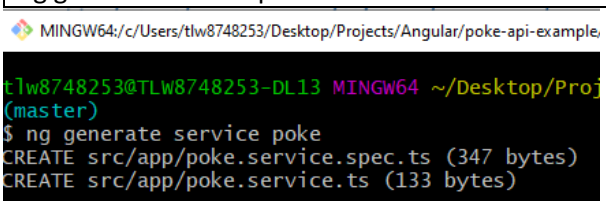
src > app > app.component.html > ...

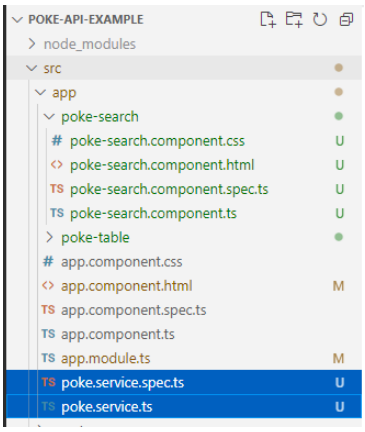
```
1 <app-poke-search (search)="onSearch($event)"></app-poke-search>
2 <app-poke-table></app-poke-table>
```

Generate a project service

In the Git Bash window

```
ng generate service poke
```

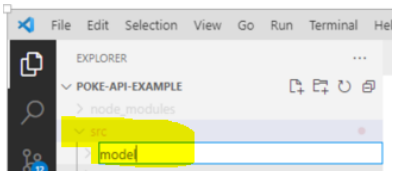




Create a pokemon model

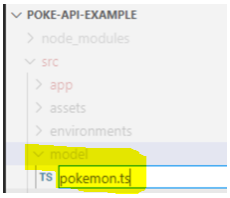
Create a folder called model under src folder

Right mouse click over src folder and select “New Folder”, enter folder name “model”



Create the pokemon.ts file

Right mouse click over model folder, select “New File”, enter file name “pokemon.ts”



Add model code in pokemon.ts

Enter the attributes we want from the https://pokeapi.co/ site

```
export interface Pokemon {
    id: number,
    name: string,
    weight: number,
    forms: { 'name': string, 'url': string }[]
  }
```

```
src > model > TS pokemon.ts > ...
1  export interface Pokemon {
2      id: number,
3      name: string,
4      weight: number,
5      forms: { 'name': string, 'url': string }[]
6    }
7
```

Update the service code in poke.service.ts

Add:

```
import { HttpClient } from '@angular/common/http';
import { Observable } from 'rxjs';
import { Pokemon } from 'src/model/pokemon';
```

Change:

```
constructor()
```

To:

```
constructor(private http: HttpClient)
```

Add:

```
  getPokemonById(id: number): Observable<Pokemon> {
    return this.http.get<Pokemon>(`https://pokeapi.co/api/v2/pokemon/${id}`);
    // this.http.get<...>(...) returns an "Observable"
    // Observables are pretty similar to Promises
  }
```

```
app.component.html M    TS poke.service.ts U ×    TS pokemon.ts U
src > app > TS poke.service.ts > ...
1  import { Injectable } from '@angular/core';
2  import { HttpClient } from '@angular/common/http';
3  import { Observable } from 'rxjs';
4  import { Pokemon } from 'src/model/pokemon';
5
6  @Injectable({
7    providedIn: 'root'
8  })
9  export class PokeService {
10
11   constructor(private http: HttpClient) {
12   }
13
14   getPokemonById(id: number): Observable<Pokemon> {
15     return this.http.get<Pokemon>(`https://pokeapi.co/api/v2/pokemon/${id}`);
16     // this.http.get<...>(...) returns an "Observable"
17     // Observables are pretty similar to Promises
18   }
19
20 }
```

Update code in poke-table.component.ts

Change:

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

To:

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

Add:

```
import { Pokemon } from 'src/model/pokemon';

@Input()
pokemon: Pokemon[] = [];

ngOnChanges() {
    console.log('pokemon input property array changed in PokeTableComponent');
}

console.log('PokeTableComponent has been initialized');

ngOnDestroy(): void {
    console.log('PokeTableComponent has been destroyed');
}
```

app.component.html M | app.component.ts 1, M | poke-table.component.ts U

src > app > poke-table > poke-table.component.ts > PokeTableComponent

```
1  import { Component, Input, OnInit } from '@angular/core';
2  import { Pokemon } from 'src/model/pokemon';
3
4  @Component({
5    selector: 'app-poke-table',
6    templateUrl: './poke-table.component.html',
7    styleUrls: ['./poke-table.component.css']
8  })
9  export class PokeTableComponent implements OnInit {
10
11   @Input()
12   pokemon: Pokemon[] = [];
13
14   constructor() { }
15
16   ngOnChanges() {
17     console.log('pokemon input property array changed in PokeTableComponent');
18   }
19
20
21   ngOnInit(): void {
22     console.log('PokeTableComponent has been initialized');
23   }
24
25   ngOnDestroy(): void {
26     console.log('PokeTableComponent has been destroyed');
27   }
28
29 }
```

Replace the default code in poke-table.component.html

```
<table>
    <thead>
      <tr>
        <th>id</th>
        <th>name</th>
        <th>weight</th>
        <th>forms</th>
      </tr>
    </thead>
    <tbody *ngFor="let poke of pokemon">
      <tr>
        <td>{{ poke.id }}</td>
        <td>{{ poke.name }}</td>
        <td>{{ poke.weight }}</td>
        <td>{{ poke.forms | json }}</td>
      </tr>
    </tbody>
  </table>
```

Update code in app.component.html

From:

```
<app-poke-search></app-poke-search>
<app-poke-table></app-poke-table>
```

To:

```
<app-poke-search (search)="onSearch($event)"></app-poke-search>
<app-poke-table *ngIf="showPokeTableComponent" [pokemon]="pokemonArr"></app-poke-table>
```

Add:

```
<button (click)="showPokeTableComponent = !showPokeTableComponent">Toggle
PokeTableComponent</button>
```

app.component.html M × | poke-table.component.html U | app.component.ts 1, M | poke-table.component.ts U

src > app > app.component.html > ...

```
<app-poke-search (search)="onSearch($event)"></app-poke-search>
<app-poke-table *ngIf="showPokeTableComponent" [pokemon]="pokemonArr"></app-poke-table>

<button (click)="showPokeTableComponent = !showPokeTableComponent">Toggle PokeTableComponent</button>
```

Update code in app.component.ts

Add:

```
import { Pokemon } from 'src/model/pokemon';
import { PokeService } from './poke.service';

showPokeTableComponent: boolean = true;

arrowFunction = (data: Pokemon) => {
    this.pokemonArr.push(data);
};

pokemonArr: Pokemon[] = [];

constructor(private pokeService: PokeService) {}
```

TS app.component.ts M ×

src > app > TS app.component.ts > AppComponent > onSearch

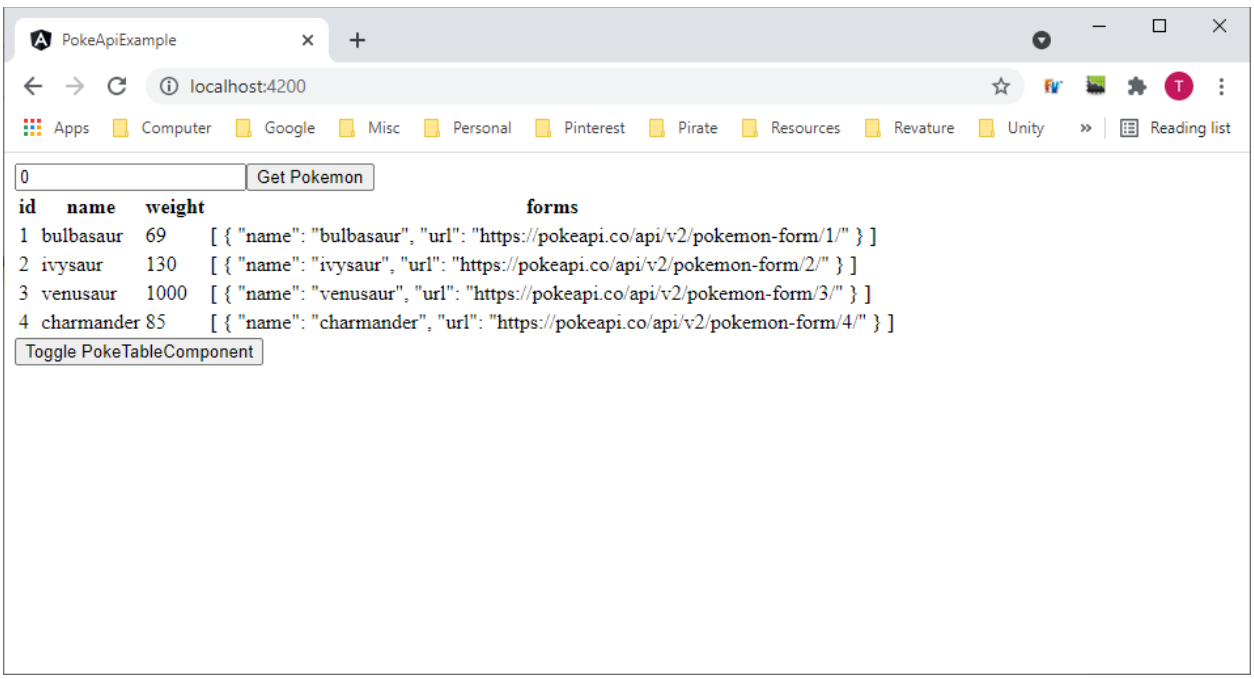
```
1  import { Component } from '@angular/core';
2  import { Pokemon } from 'src/model/pokemon';
3  import { PokeService } from './poke.service';
4
5  @Component({
6    selector: 'app-root',
7    templateUrl: './app.component.html',
8    styleUrls: ['./app.component.css']
9  })
10 export class AppComponent {
11   title = 'poke-api-example';
12
13   showPokeTableComponent: boolean = true;
14
15   arrowFunction = (data: Pokemon) => {
16     this.pokemonArr.push(data);
17   };
18
19   pokemonArr: Pokemon[] = [];
20
21   // Angular automatically figures out when it needs to instantiate the AppComponent,
22   // that it also needs to provide a PokeService object
23   // To provide this object, Angular also first needs to instantiate a PokeService
24   // object and then keep track of that object
25   constructor(private pokeService: PokeService) {}
```

Add:

```
ngOnInit(): void {
    this.pokeService.getPokemonById(1).subscribe(this.arrowFunction);
    this.pokeService.getPokemonById(2).subscribe(this.arrowFunction);
    this.pokeService.getPokemonById(3).subscribe(this.arrowFunction);

    this.pokeService.getPokemonById(4).toPromise().then((data) => {
      this.pokemonArr.push(data);
    });
}

onSearch(event: number) {
     this.pokeService.getPokemonById(event).subscribe(this.arrowFunction);
}
```

```
constructor(private pokeService: PokeService) {}

ngOnInit(): void {
  this.pokeService.getPokemonById(1).subscribe(this.arrowFunction);
  this.pokeService.getPokemonById(2).subscribe(this.arrowFunction);
  this.pokeService.getPokemonById(3).subscribe(this.arrowFunction);

  this.pokeService.getPokemonById(4).toPromise().then((data) => {
    this.pokemonArr.push(data);
  });
}


onSearch(event: number) {
  // getPokemonById(...) is returning the observable that was returned from the get(...)
  // method.

  // Whenever a response is received, the Observable will "publish" the response
  // to the subscriber

  // const arrowFunction = (data: Pokemon) => {
  //   this.pokemonArr.push(data);
  // };
  // This arrow function is the subscriber, that will received the published
  // data, and then perform some operations on that data

  this.pokeService.getPokemonById(event).subscribe(this.arrowFunction);
}
```

Webpage final results



Aug 27th Lecture and Project Updates

This project was revisited during the Aug 27th lecture. All coding changes were incorporated during the Aug 26th lecture. Notes from the Aug 27th lecture are found in the appendix.

[Appendix: Aug 27th PowerPoint Slides]

[Appendix: Aug 27th Component Lifecycle notes add to angular-components.md]

[Appendix: Aug 27th dependency-injection-diagram]

Aug 27th Lifecycle hooks Already in the Code

Changes for lifecycle hooks in poke-table.components.ts

```
constructor() { }
  ngOnChanges() {
    console.log('pokemon input property array changed in PokeTableComponent');
  }
  ngOnInit(): void {
    console.log('PokeTableComponent has been initialized');
  }
  ngOnDestroy(): void {
    console.log('PokeTableComponent has been destroyed');
  }
```

TS poke-table.component.ts U ×

src > app > poke-table > TS poke-table.component.ts > ...

```
1  import { Component, Input, OnInit } from '@angular/core';
2  import { Pokemon } from 'src/model/pokemon';
3
4  @Component({
5    selector: 'app-poke-table',
6    templateUrl: './poke-table.component.html',
7    styleUrls: ['./poke-table.component.css']
8  })
9  export class PokeTableComponent implements OnInit {
10
11   @Input()
12   pokemon: Pokemon[] = [];
13
14   constructor() { }
15
16   ngOnChanges() {
17     console.log('pokemon input property array changed in PokeTableComponent');
18   }
19
20   ngOnInit(): void {
21     console.log('PokeTableComponent has been initialized');
22   }
23
24   ngOnDestroy(): void {
25     console.log('PokeTableComponent has been destroyed');
26   }
27
28 }
```

Aug 27th button was added to app.component.html to demonstrate ngOnDestroy() in the above code.

```
<button (click)="showPokeTableComponent = !showPokeTableComponent">Toggle PokeTableComponent</button>
```

app.component.html M ×

src > app > app.component.html > ...

```
1  <app-poke-search (search)="onSearch($event)"></app-poke-search>
2  <app-poke-table *ngIf="showPokeTableComponent" [pokemon]="pokemonArr"></app-poke-table>
3
4  <button (click)="showPokeTableComponent = !showPokeTableComponent">Toggle PokeTableComponent</button>
5
```

Asynchronous requests processing in app.component.ts

Aug 27th changes to initialize pokemon table data with three requests

```
ngOnInit(): void {
  this.pokeService.getPokemonById(1).subscribe(this.arrowFunction);
  this.pokeService.getPokemonById(2).subscribe(this.arrowFunction);
  this.pokeService.getPokemonById(3).subscribe(this.arrowFunction);
}
```

0 Get Pokemon

id	name	weight	forms
2	ivysaur	130	[{ "name": "ivysaur", "url": "https://pokeapi.co/api/v2/pokemon-form/2/" }]
1	bulbasaur	69	[{ "name": "bulbasaur", "url": "https://pokeapi.co/api/v2/pokemon-form/1/" }]
3	venusaur	1000	[{ "name": "venusaur", "url": "https://pokeapi.co/api/v2/pokemon-form/3/" }]

Although the ngOnInit() requests asked for ids 1,2,3 the requests finished in the shortest time to complete and was received and displayed in order of 2,1,3. Code was added to insure the request to display order.

```
ngOnInit(): void {
  this.pokeService.getPokemonById(1).subscribe(this.arrowFunction);
  this.pokeService.getPokemonById(2).subscribe(this.arrowFunction);
  this.pokeService.getPokemonById(3).subscribe(this.arrowFunction);

  this.pokeService.getPokemonById(4).toPromise().then((data) => {
    this.pokemonArr.push(data);
  });
}
```

```
ngOnInit(): void {
  this.pokeService.getPokemonById(1).subscribe(this.arrowFunction);
  this.pokeService.getPokemonById(2).subscribe(this.arrowFunction);
  this.pokeService.getPokemonById(3).subscribe(this.arrowFunction);

  this.pokeService.getPokemonById(4).toPromise().then((data) => {
    this.pokemonArr.push(data);
  });
}
```

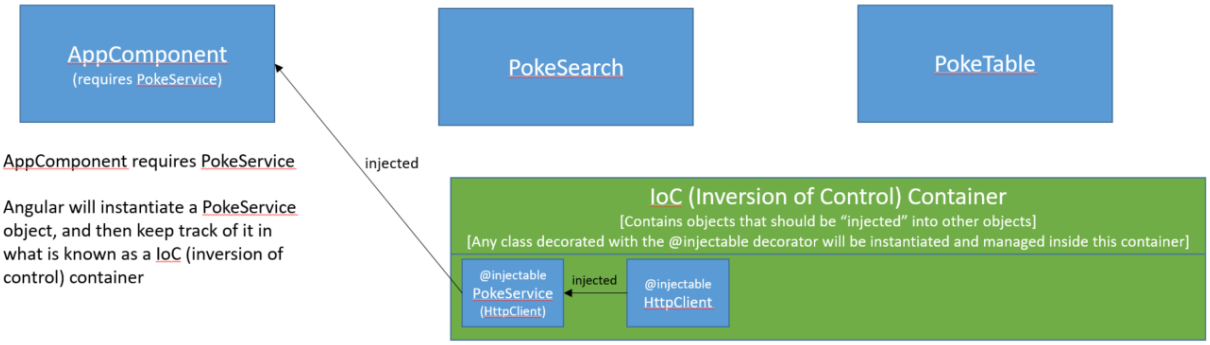
0 Get Pokemon

id	name	weight	forms
1	bulbasaur	69	[{ "name": "bulbasaur", "url": "https://pokeapi.co/api/v2/pokemon-form/1/" }]
2	ivysaur	130	[{ "name": "ivysaur", "url": "https://pokeapi.co/api/v2/pokemon-form/2/" }]
3	venusaur	1000	[{ "name": "venusaur", "url": "https://pokeapi.co/api/v2/pokemon-form/3/" }]
4	charmander	85	[{ "name": "charmander", "url": "https://pokeapi.co/api/v2/pokemon-form/4/" }]

Appendix: Aug 27th PowerPoint Slides

Poke-api-example

When our Angular application starts up, it will initialize our 3 components:



AppComponent requires PokeService

Angular will instantiate a PokeService object, and then keep track of it in what is known as a IoC (inversion of control) container

Sending HTTP requests in Angular

- Step 1: Include the HttpClientModule as an import in your AppModule

```
14      PokeSearchComponent,
15      PokeTableComponent
16    ],
17    imports: [
18      BrowserModule,
19      FormsModule,
20      HttpClientModule
21    ],
22    providers: [],
23    bootstrap: [AppComponent]
24  })
25  export class AppModule { }
```

- Step 2: Create a service using `ng generate service <service-name>`
- Step 3: Specify the HttpClient dependency as a parameter in the constructor of the newly created service

```
11  constructor(private http: HttpClient) {
12  }
```

- Step 4: Specify the service dependency as a parameter in the constructor of the components that require the data from this http request

```
21  // Angular automatically figures out when it needs to instantiate the AppComponent,
22  // that it also needs to provide a PokeService object
23  // To provide this object, Angular also first needs to instantiate a PokeService
24  // object and then keep track of that object
25  constructor(private pokeService: PokeService) {}
```

Appendix: Aug 27th Component Lifecycle notes add to angular-components.md

Component Lifecycle (Lifecycle Hooks)
Whenever components are created and during the time of their operation, they go through various different phases. We have function that are known as `lifecycle hooks` that will execute whenever certain conditions are met. We can utilize this lifecycle hooks to potentially perform useful actions with our components.

These are the following lifecycle hooks to be aware of:
- constructor: Actually instantiates and populates the initial dependencies (through dependency injection, in the case of Angular)
- ngOnChanges(): whenever the input properties of a component change (properties decorated with the @Input() decorator), this method is called. Therefore, this method could be called multiple times during the lifetime of a component
- ngOnInit(): called ONE TIME when the component is first initialized (when it actually populates the DOM with that component)
- ngDoCheck(): called immediately after ngOnChanges() and ngOnInit() so that we can implement our own custom actions for change detection
- ngOnDestroy(): called before Angular destroys a component

Appendix: Aug 27th dependency-injection-diagram