ТИНЬКОФФ

Angular Platforms

Как запускать приложения где угодно?

HolyJS Spring 2024

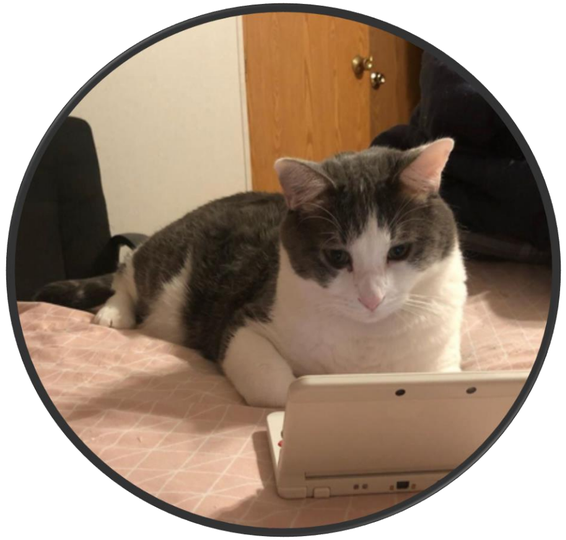
Соловьев Олег | Frontend-разработчик

- В IT с 2019 года, работал в крупных IT компаниях
- Сейчас пишу фронт в Тинькофф
- И учу людей программировать
- Большой фанат Ангуляра

О чем сегодня поговорим

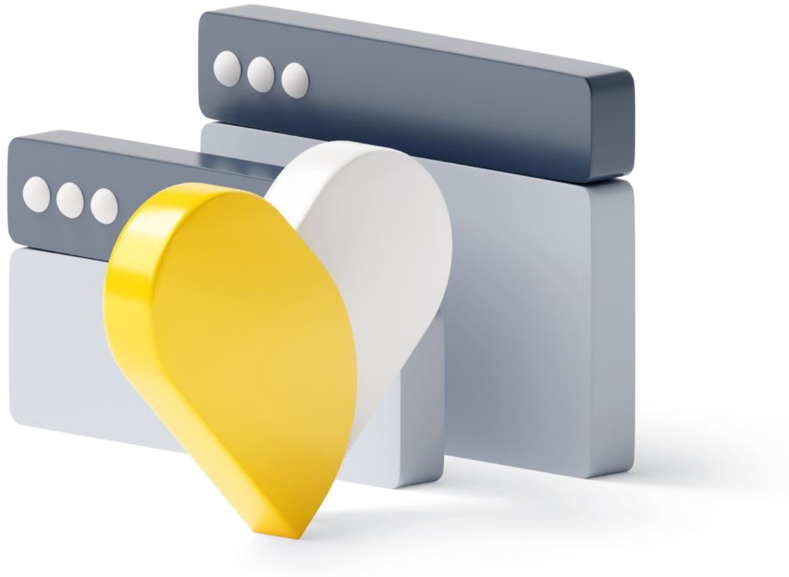
- Проведем обзор существующих кросс-платформенных технологий на Angular
- Узнаем о процессе запуска Angular приложения
- Познакомимся с механизмом платформ
- Напишем свою собственную платформу!

Программист Вася



Angular

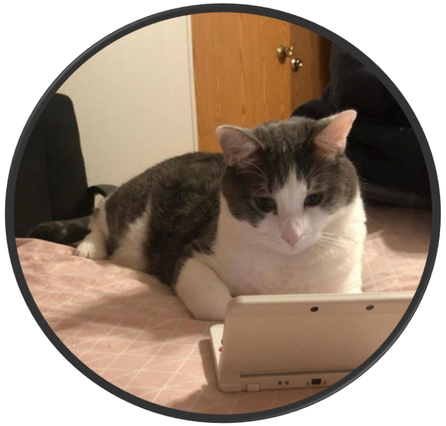
За что Вася любит Angular?



- Удобная модель компонентов
- Встроенные пайпы и директивы
- Change Detection
- RxJS
- Куча библиотек

Программист Вася написал уже кучу фронтендов на Angular. Он в совершенстве освоил реактивное программирование
и познал все тонкости разработки под веб

Вася выгорает и хочет написать что-то новое. При этом ему не хочется расставаться
со своим любимым фреймворком



Всё надоело. Выгораю...

Не бросай меня!

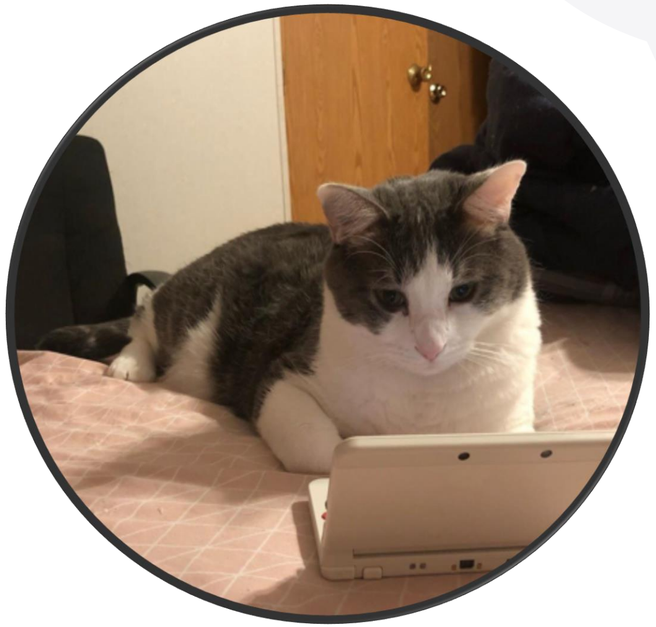
А где ещё работает Angular?

На сервере

Universal

Компилируем приложение в статические
HTML-файлы и отдаем клиентам.
Круто для быстрой начальной загрузки
и поисковой оптимизации

Тот же веб, только на сервере...

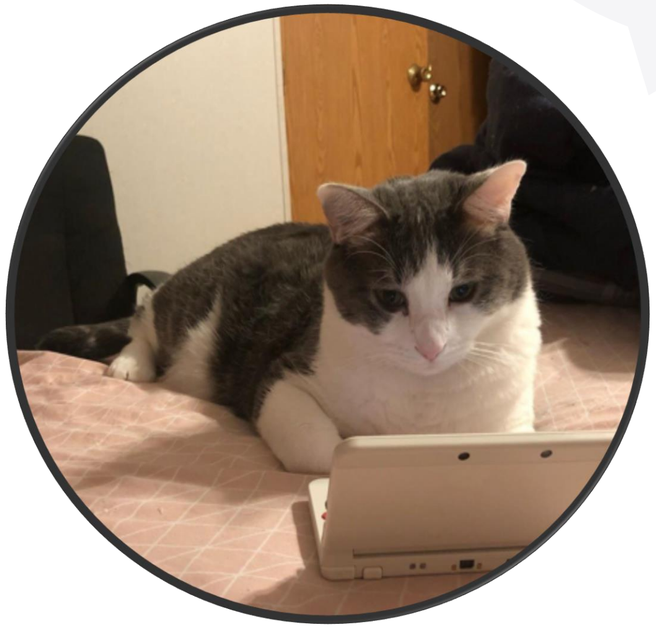


На мобильных устройствах

NativeScript

Позволяет компилировать кросс-платформенные мобилки из одной кодовой базы.
Запускает два процесса – джаваскриптовый
для выполнения логики и нативный для рендера родных контролов устройства.

Это уже интереснее!




JS Process
Native Process
Bridge

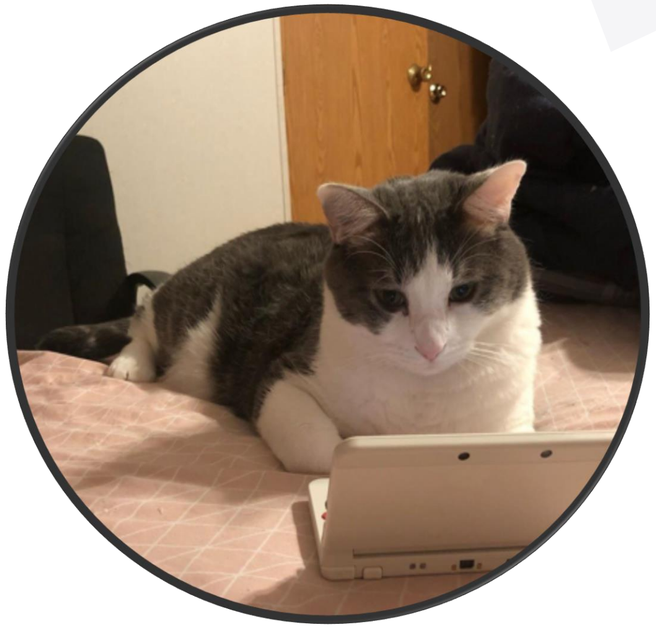
Есть ли что-то для десктопа?

На десктопе – Electron

Открываем экземпляр браузера и даем ему доступ к некоторым API операционки, типа доступа к файлам и нотификациям.

По сути – обертка вокруг веб-приложения

Звучит не очень производительно



В чем минусы Электрона?

- Не задействуем нативные возможности
- Медленный
- Кушает много памяти
- IPC

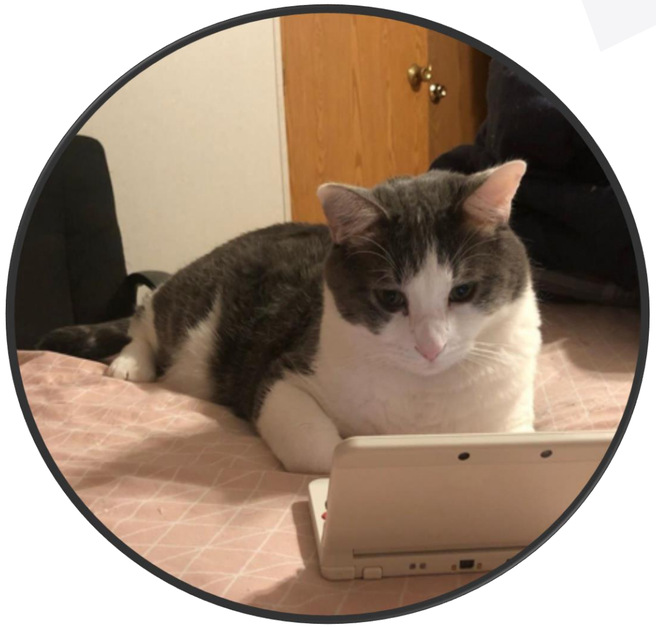
Может, есть ещё технологии?

Angular Node GUI

В отличие от Электрона, не оборачиваем приложение в инстанс браузера. Вместо этого дергаем нативные API операционной системы.

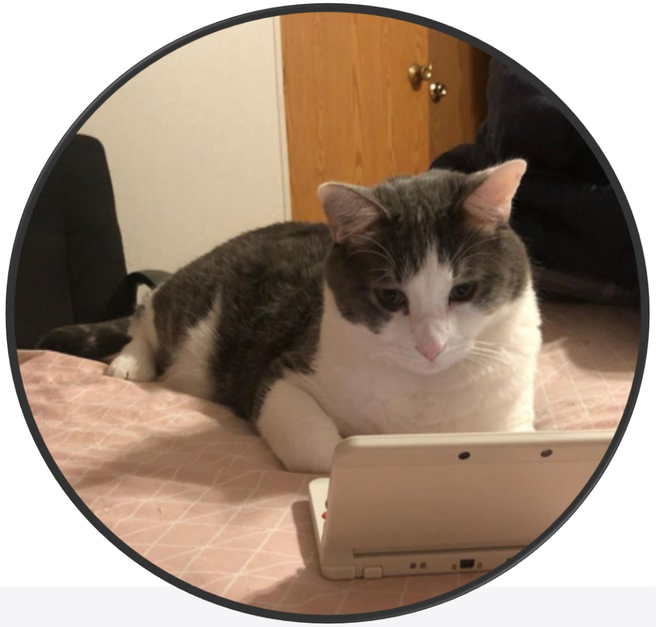
Под капотом – библиотека QT. У которой биндинги вообще на C++.

Звучит гораздо круче Электрона!

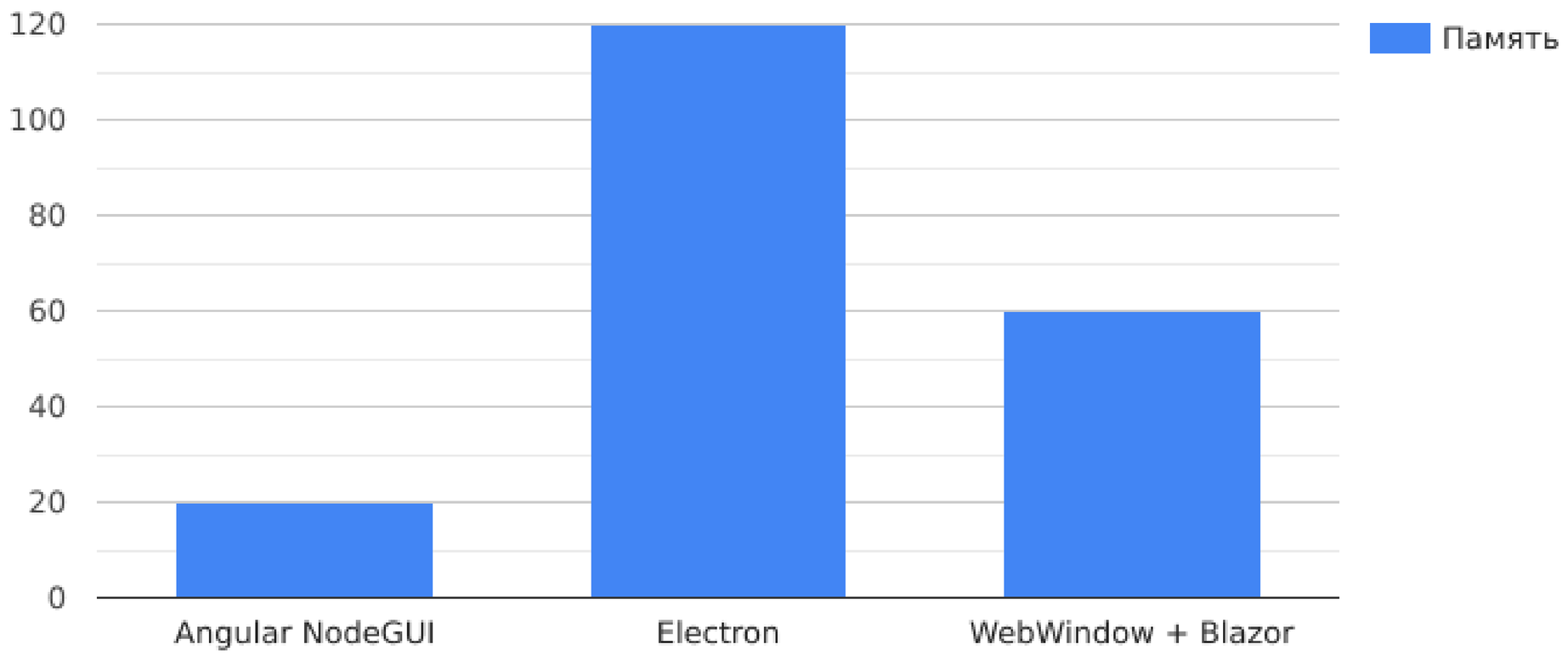


Angular Node GUI

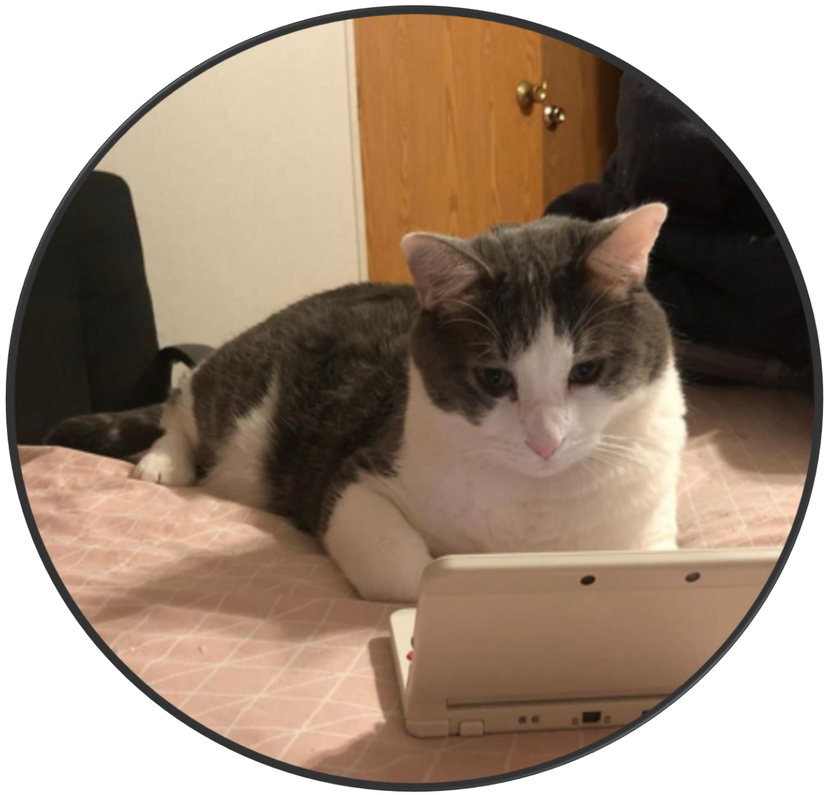
```
<window #window title="Weather" id="win">
  <view id="container">
    <image id="image" [src]="'./projects/weather-demo/src/assets/09d.png'" [aspectRatioMode]="aspectRatioMode"></image>
    <view id="details">
      <app-summary id="summary" [title]="'Rain'" [description]="'shower rain'"></app-summary>
      <view id="tempbox">
        <app-temerature [now]="3" [max]="5" [min]="1"></app-temerature>
        <app-placedate [place]="'Snt.Petersburg'" [date]="date"></app-placedate>
      </view>
      <view id="buttonbox">
        <button (clicked)="onRefresh()">Refresh</button>
        <button (clicked)="onClose()">Close</button>
      </view>
    </view>
  </view>
</window>
```



Сравним решения

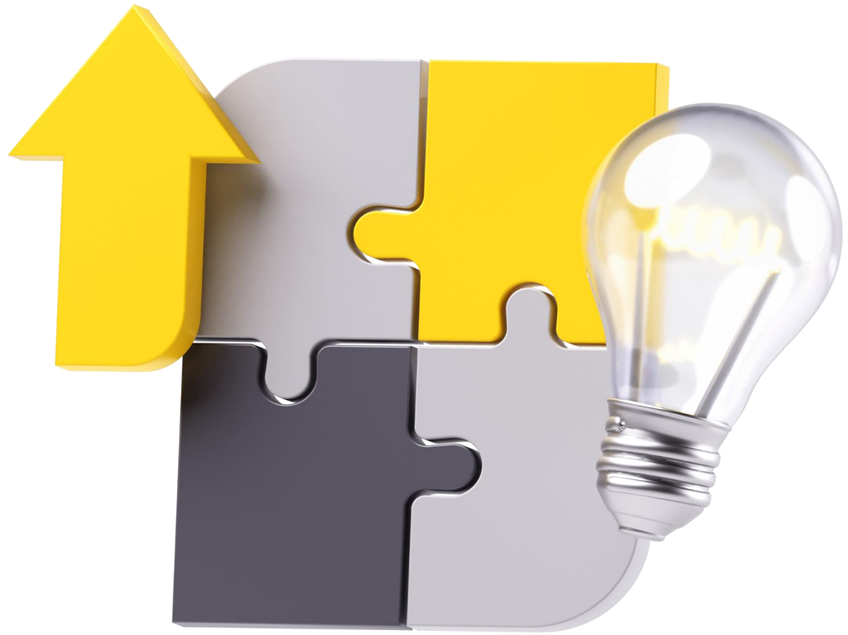


Но как оно работает?



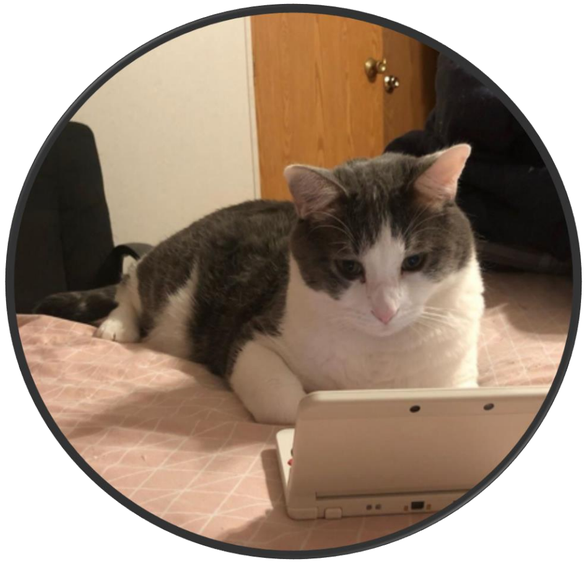
Прочитал что-то про платформы и подмену рендера. Ничего не понял ☹

Давайте напишем свою платформу!




./demo.js
Total Transactions
80
60
40
20
0
USA
Europe
00:00 01:20 03:00
Server Utilization (%)
0 0 6 7
US1 US2 EU1 AU1
Total Transactions
80
60
40
20
0
USA
Europe
00:00 00:30 01:20 02:10 03:00 03:50
Active Processes
Process Cpu (%) Memory
watchdog 2 72
gulp 1 44
awk 1 99
node 0 25
netns 3 37
java 4 64
Servers Location
Throughput (bits/sec)
Server1:
Server2:
Server3:
Server4:
Server5:
Server6:

Хочу…



Время для теории!

Что требуется для запуска Angular?

Какой механизм позволяет достичь кросс-платформенности?

Angular Platforms

main.ts

```
import { platformBrowserDynamic } from '@angular/platform-browser-dynamic';

platformBrowserDynamic().bootstrapModule(AppModule);
```

main.ts

```
import { platformBrowserDynamic } from '@angular/platform-browser-dynamic';
import { PlatformRef } from '@angular/core';


const platformRef: PlatformRef = platformBrowserDynamic();
platformRef.bootstrapModule(AppModule);
```

PlatformRef – умеет запускать приложение,
Используя модуль или компонент

platformBrowserDynamic

```
export const platformBrowserDynamic = createPlatformFactory(
    platformCoreDynamic,
    'browserDynamic',
    INTERNAL_BROWSER_DYNAMIC_PLATFORM_PROVIDERS,
);
```

Платформа создается через фабрику платформ

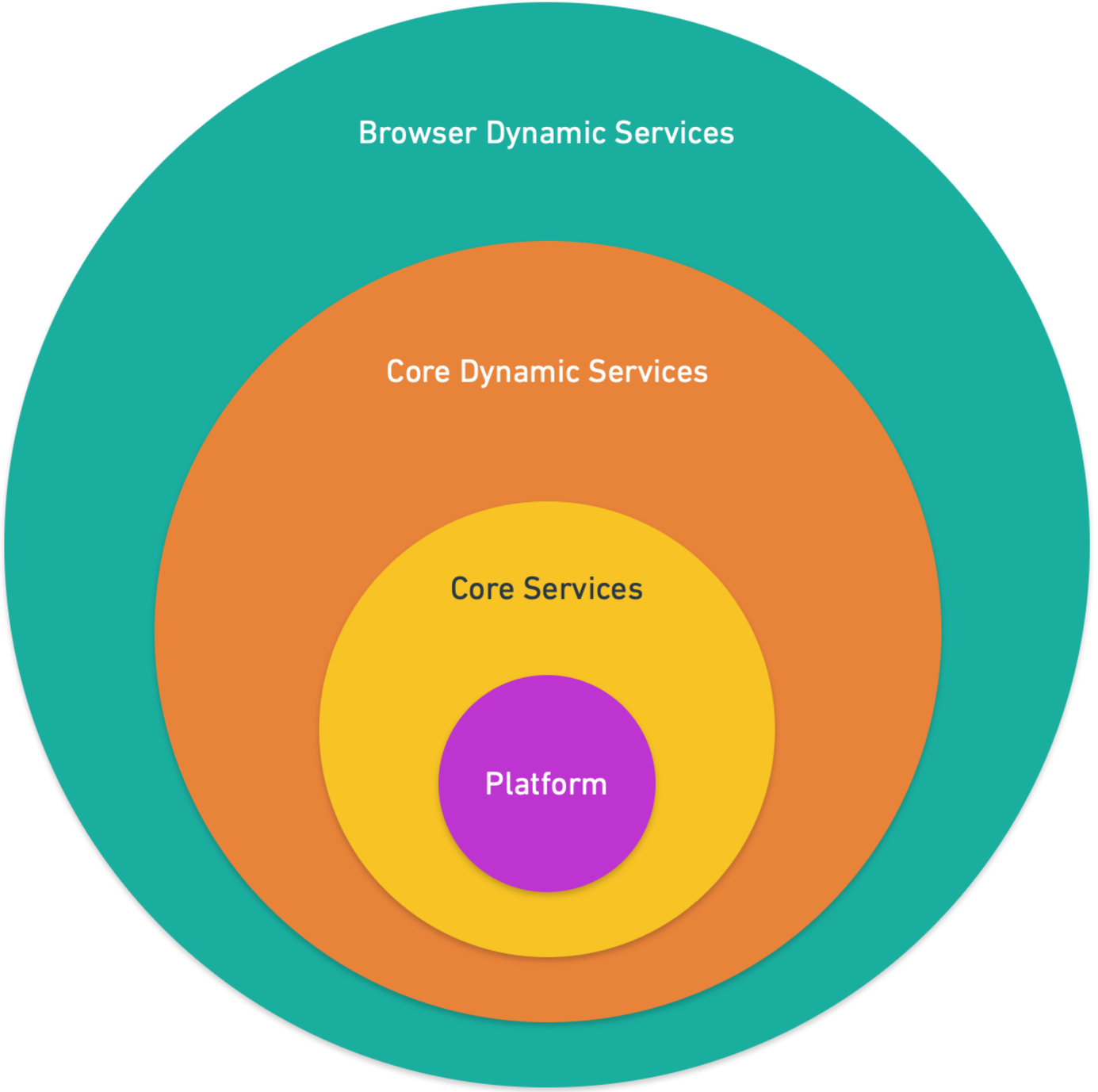
platformCoreDynamic

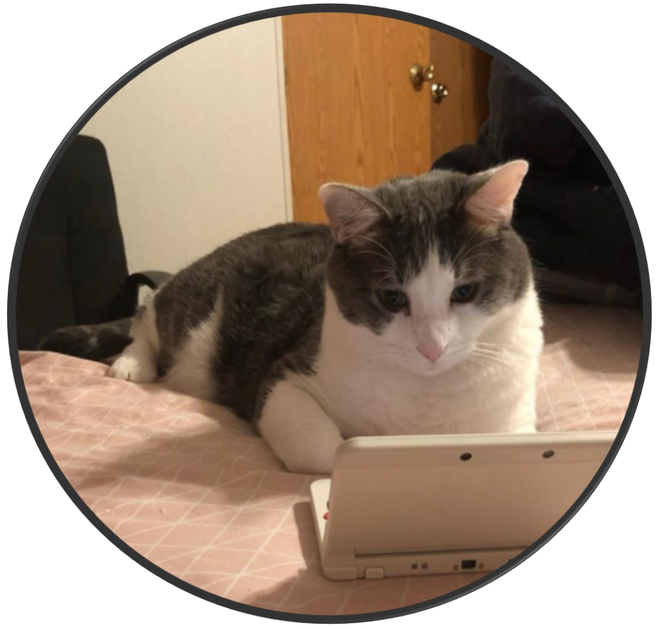
```
export const platformCoreDynamic = createPlatformFactory(
  platformCore,
  'coreDynamic',
  CORE_DYNAMIC_PROVIDERS,
);
```

platformCore

```
export const platformCore = createPlatformFactory(
  null,
  'core',
  CORE_PLATFORM_PROVIDERS,
);
```

Иерархия платформ





Вижу паттерн Декоратор!

Фабрика платформ

```
type PlatformFactory = (extraProviders?: StaticProvider[]) => PlatformRef;

export function createPlatformFactory(
  parentPlatformFactory: PlatformFactory,
  name: string,
  providers: StaticProvider[],
): PlatformFactory {
  return (extraProviders: StaticProvider[]) => {
    const injectedProviders: StaticProvider[] = providers.concat(extraProviders);
    if (parentPlatformFactory) {
      return parentPlatformFactory(injectedProviders);
    } else {
      return createPlatform(Injector.create({ providers: injectedProviders }));
    }
  };
}
```

Рутовая платформа

```
function createPlatform(injector: Injector): PlatformRef {
  return injector.get(PlatformRef);
}
```

Так вот он откуда берется...

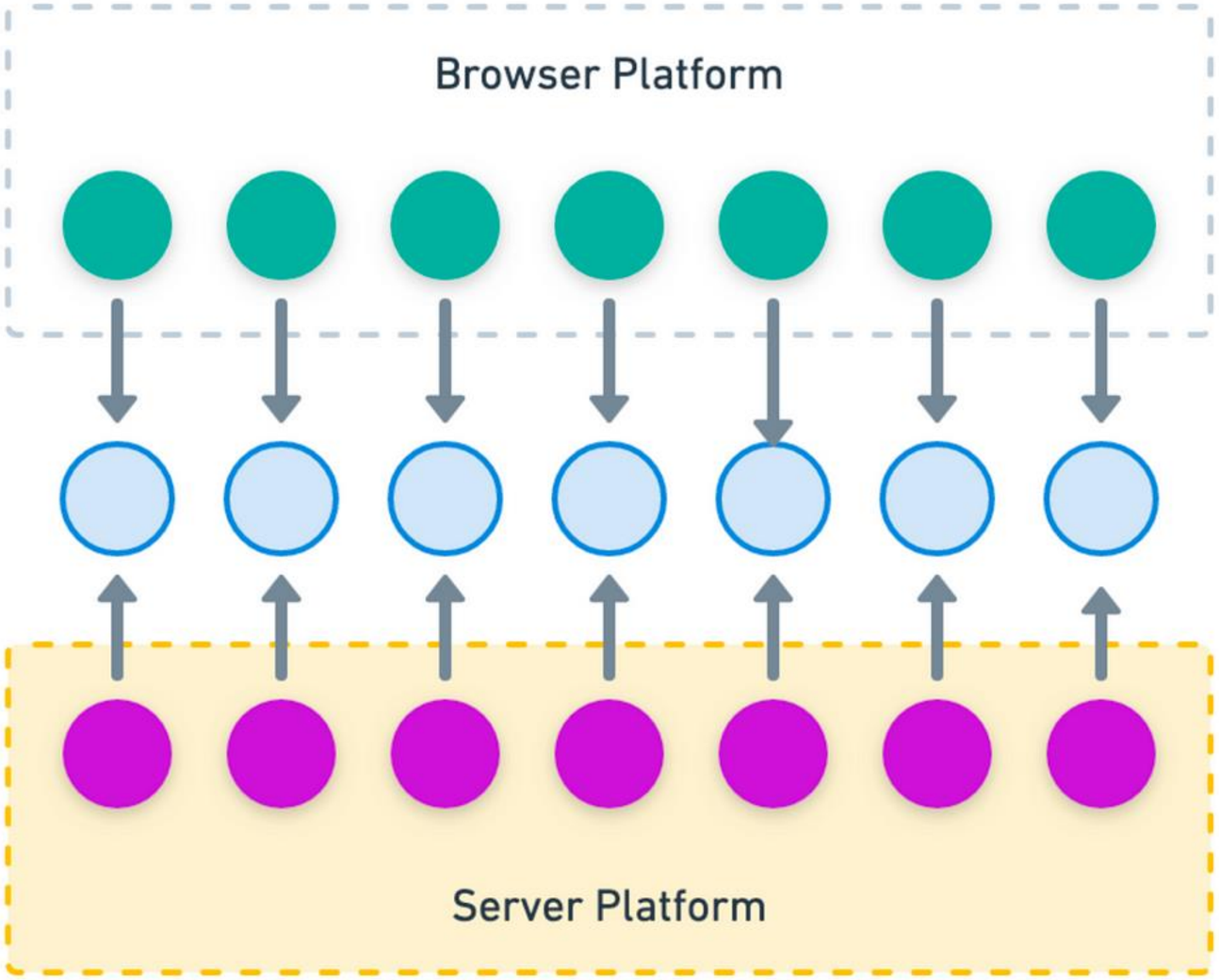
Я понял, как работает механизм платформ. Но как создать свою платформу?

Платформа должна предоставлять необходимые сервисы

Что за сервисы?

- Renderer
- RendererFactory
- ElementsRegistry
- SchemaRegistry
- Sanitizer

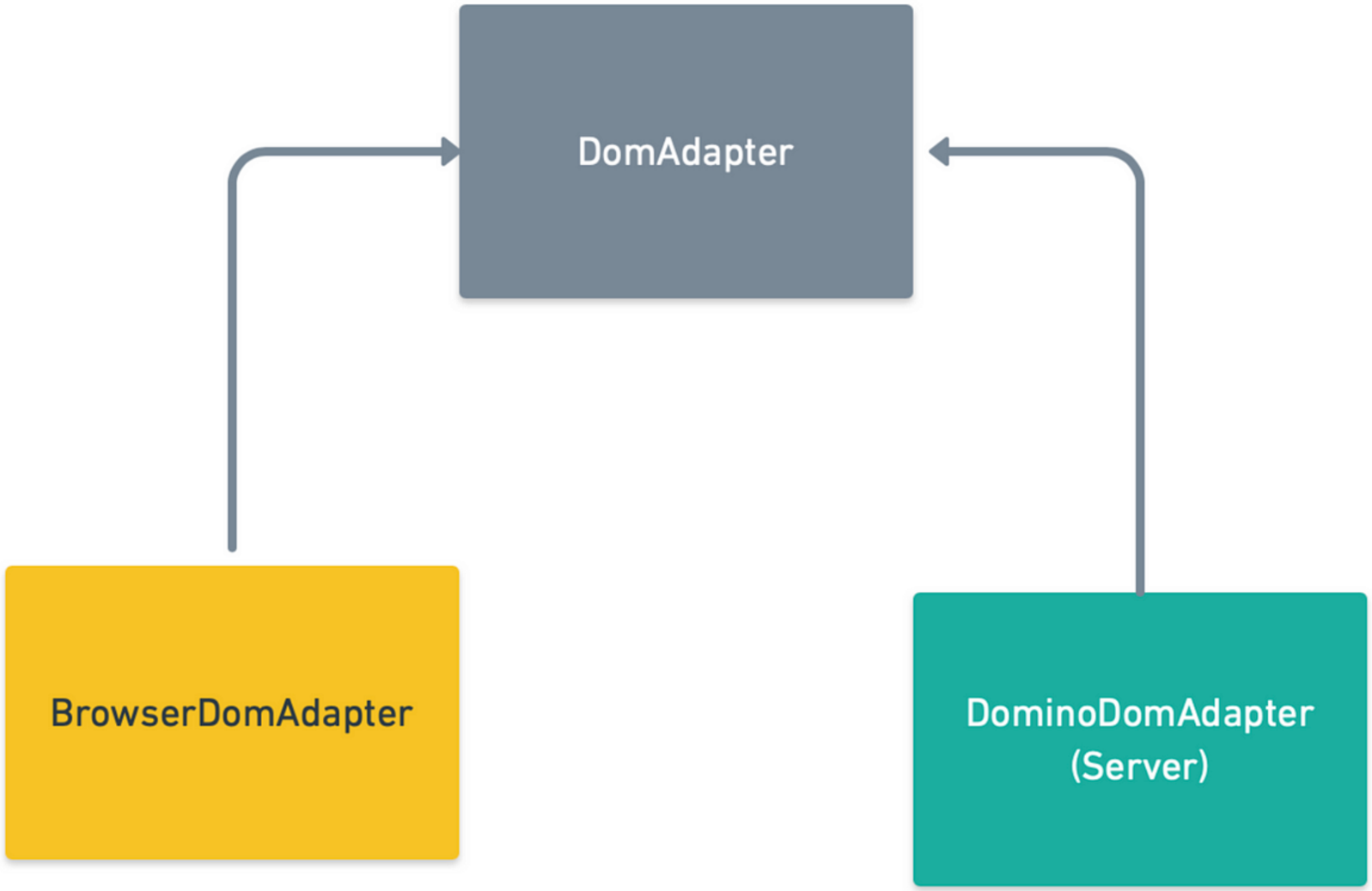
Платформенные сервисы



Платформенные сервисы

```
export abstract class DomAdapter {
  abstract setProperty(el, name, value);
  abstract getProperty(el, name);
  abstract querySelector(el, selector);
  abstract querySelectorAll(el, selector);
  abstract appendChild(el, node);
  abstract removeChild(el, node);
}
```

Платформенные сервисы



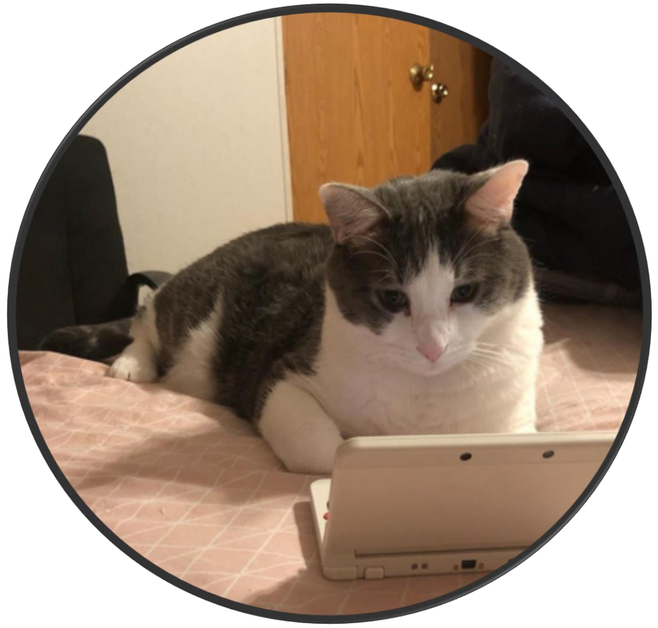
Время
для разработки!

Выбираем Библиотеку Для рисования В терминале

blessed

A curses-like library with a high level terminal interface API for node.js.

```
var screen = blessed.screen({
  smartCSR: true
});

screen.title = 'my window title';

var box = blessed.box({
  top: 'center',
  left: 'center',
  width: '50%',
  height: '50%',
  content: 'Hello {bold}world{/bold}!',
})
```

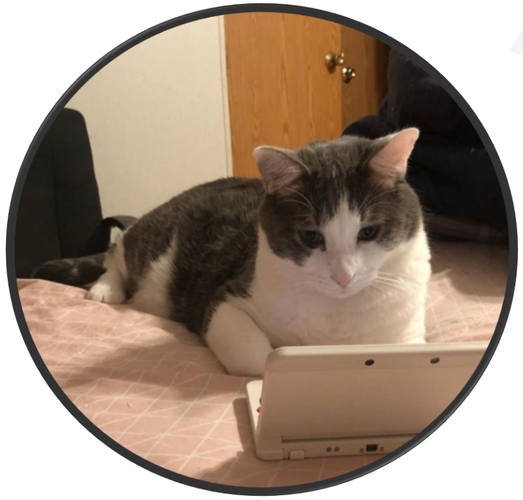


Мне нужно подружить blessed с Angular...


blessed
A curses-like library with a high level terminal interface API for node.js.

Elements Registry

Сервис, отвечающий за реестр атомарных элементов пользовательского интерфейса.
В вебе это DOM-элементы



Подменим браузерный ElementsRegistry на самописный терминальный!

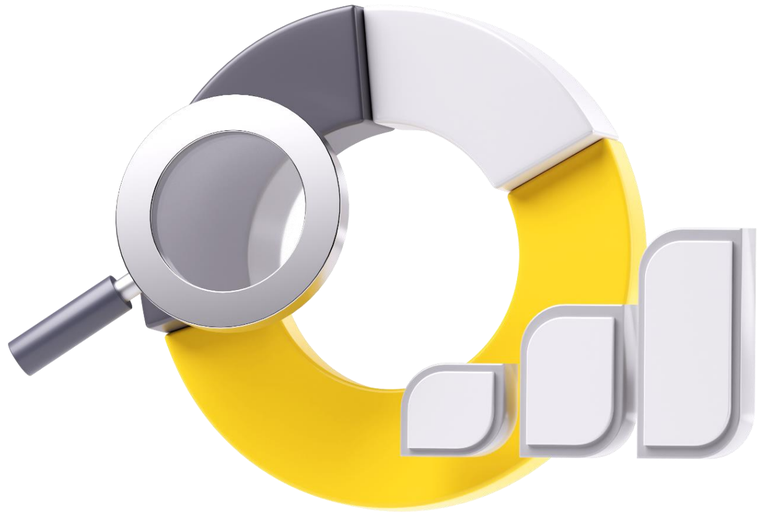
ElementsRegistry

```
export const elementsFactory: Map<string, ElementFactory> = new Map()
  .set('text', blessed.text)
  .set('box', blessed.box)
  .set('table', blessed.table)
  .set(line, customBlessedLine)

function customBlessedLine(): Blessed.Element {
    // Любая логика по отрисовке через blessed
}
```

ElementsRegistry

```
@Injectable()
export class ElementsRegistry {

  createElement(name: string, options: any = {}): Widgets.BoxElement {
    let elementFactory: ElementFactory = elementsFactory.get(name);

    if (!elementFactory) {
      elementFactory = elementsFactory.get('box');
    }

    return elementFactory({ ...options, screen: this.screen });
  }
}
```

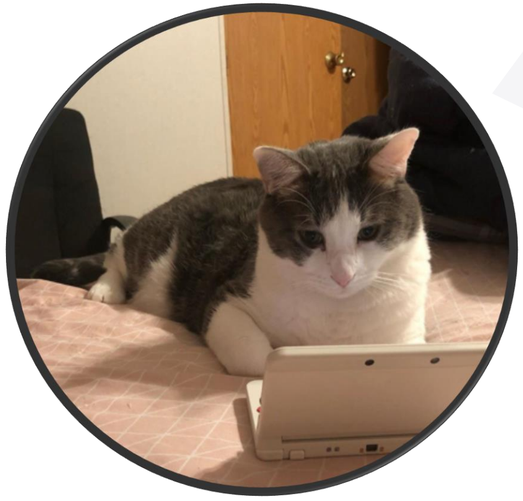
Root Screen Element



Screen

Renderer

Сервис, отвечающий за создание элементов из реестра и за манипуляции с ними – изменение атрибутов, удаление и т.д.



Наконец-то буду рендерить не в браузер!

Renderer

```
export class TerminalRenderer implements Renderer2 {
  constructor(private screen: Screen, private elementsRegistry: ElementsRegistry) {}

  createElement(name: string, namespace?: string | null): any {
    return this.elementsRegistry.createElement(name);
  }

  createText(value: string): any {
    return this.elementsRegistry.createElement('text', { content: value });
  }

  selectRootElement(): Widgets.Screen {
    return this.screen.selectRootElement();
  }

  appendChild(parent: Widgets.BlessedElement, newChild: Widgets.BlessedElement): void {
    parent.append(newChild);
  }
```

Renderer

```
setAttribute(el: Widgets.BlessedElement, name: string, value: string): void {
  el[name] = value;
}

setValue(node: Widgets.BlessedElement, value: string): void {
  node.setContent(value);
}

listen(target: Widgets.BlessedElement, eventName: string, callback: Function) {
  target.on('click', callback);
}
```

Renderer Factory

```
@Injectable()
export class TerminalRendererFactory implements RendererFactory2 {

  constructor(private screen: Screen, private elementsRegistry: ElementsRegistry)

  createRenderer(): Renderer2 {
    return new TerminalRenderer(this.screen, this.elementsRegistry);
  }
}
```

Sanitizer

Сервис, экранирующий инпут от пользователя. В вебе предотвращает XSS атаки и инъекции.

В терминале таких проблем нет, поэтому делаем заглушку.

Sanitizer

```ts
import { Sanitizer, SecurityContext } from '@angular/core';

export class TerminalSanitizer extends Sanitizer {
  sanitize(context: SecurityContext, value: string): string {
    return value;
  }
}
```

Error Handler

Сервис, который обрабатывает все unhandled exceptions.

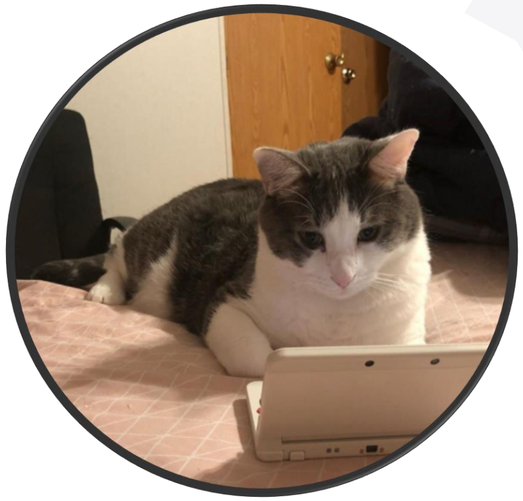
В консоли нет кнопки refresh – поэтому будем убивать процесс.

ErrorHandler

```
import { ErrorHandler, Injectable } from '@angular/core';

@Injectable()
export class TerminalErrorHandler implements ErrorHandler {

  handleError(error: Error): void {
    console.error(error.message, error.stack);
    process.exit(1);
  }
}
```

Terminal Module & Platform

Наконец, необходимо предоставить приложению необходимые провайдеры и запуститься с новой терминальной платформы



Пора собирать всё вместе!

Terminal Module

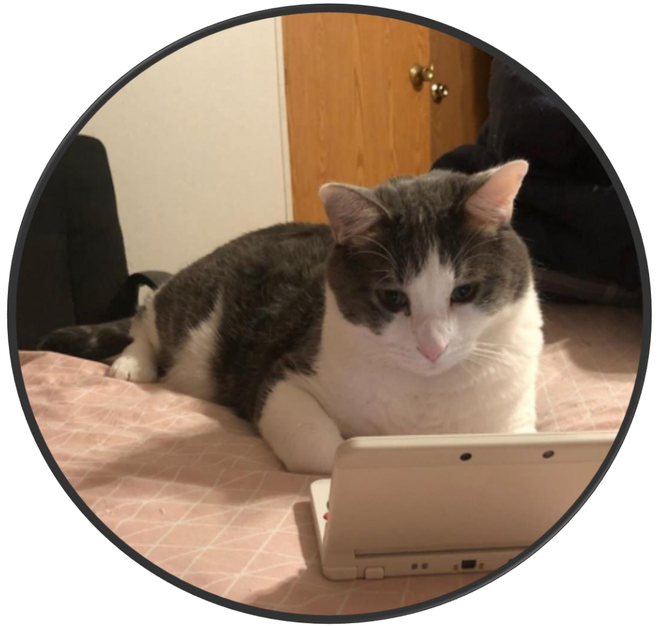
```
@NgModule({
  exports: [CommonModule, ApplicationModule],
  providers: [
    Screen,
    ElementsRegistry,
    { provide: RendererFactory2, useClass: TerminalRendererFactory },
    { provide: ErrorHandler, useClass: TerminalErrorHandler },
  ],
})
export class TerminalModule {
}
```

Terminal Platform

```
const TERMINAL_PROVIDERS = [
  { provide: DOCUMENT, useValue: {} },
  { provide: Sanitizer, useClass: TerminalSanitizer, deps: [] },
];

export const platformTerminalDynamic = createPlatformFactory(platformCoreDynamic,
  'terminalDynamic', TERMINAL_PROVIDERS);
```

main.ts

```
platformTerminalDynamic().bootstrapModule(AppModule)
  .catch(err => console.error(err));
```



Вроде всё готово. Хочу написать свой первый терминальный компонент!

app.component.ts

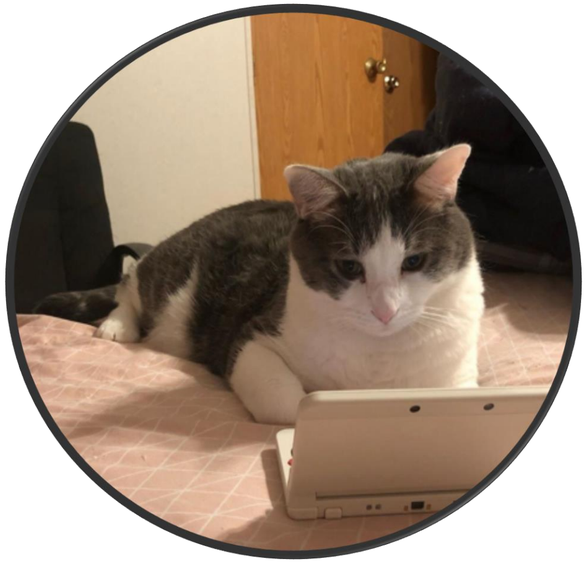
```
export class AppComponent {
  transactions$ = this.transactionsService.transactions$;
  sparkline$ = this.sparklineService.sparkline$;
  serversUtilization$ = this.serversUtilization.serversUtilization$;
  process$ = this.processManager.process$;

  constructor(private transactionsService: TransactionsService,
              private sparklineService: SparklineService,
              private serversUtilization: ServerUtilizationService,
              private processManager: ProcessManagerService) {
  }
}
```

app.component.html

```
<grid rows="12" cols="12">
    <line
      [row]="0" [col]="0" [rowSpan]="3" [colSpan]="3"
      label="Total Transactions"
      [data]="transactions$ | async">
    </line>
    <bar
      [row]="0" [col]="3" [rowSpan]="3" [colSpan]="3" label="Server Utilization (%)"
      [data]="serversUtilization$ | async">
    </bar>
    <table
      [row]="3" [col]="0" [rowSpan]="3" [colSpan]="6"
      fg="green"
      label="Active Processes"
      [data]="process$ |async">
    </table>
</grid>
```


./demo.js
Total Transactions
80
60
40
20
0
USA
Europe
00:00 01:20 03:00
Server Utilization (%)
0 0 6 7
US1 US2 EU1 AU1
Total Transactions
80
60
40
20
0
USA
Europe
00:00 00:30 01:20 02:10 03:00 03:50
Active Processes
Process Cpu (%) Memory
watchdog 2 72
gulp 1 44
awk 1 99
node 0 25
netns 3 37
java 4 64
Servers Location
Throughput (bits/sec)
Server1:
Server2:
Server3:
Server4:
Server5:
Server6:

Работает



Время
для демонстрации!

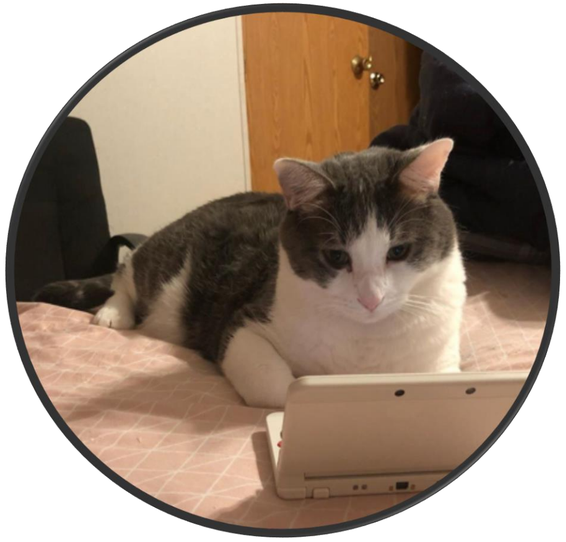
А как оно устроено у остальных проектов, которые я изучал ранее?

NativeScript Elements Registry

```
export function registerNativeScriptViewComponents() {
    registerElement('AbsoluteLayout', () => AbsoluteLayout);
    registerElement('ActivityIndicator', () => ActivityIndicator);
    registerElement('Button', () => Button, textBaseMeta);
    registerElement('ContentView', () => ContentView);
    registerElement('DatePicker', () => DatePicker);
    registerElement('DockLayout', () => DockLayout);
    registerElement('Frame', () => Frame, frameMeta);
    registerElement('GridLayout', () => GridLayout);
    registerElement('HtmlView', () => HtmlView);
}
```

NativeScript Platform Providers

```
export const NATIVESCRIPT_MODULE_STATIC_PROVIDERS: StaticProvider[] = [
  { provide: APP_ROOT_VIEW, useFactory: generateFallbackRootView, deps: [[new Optional(), new SkipSelf(), APP_ROOT_VIEW]] },
  { provide: ErrorHandler, useFactory: errorHandler, deps: [] },
  { provide: ViewUtil, useClass: ViewUtil, deps: [NAMESPACE_FILTERS, [new Optional(), ENABLE_REUSABE_VIEWS]] },
  {
    provide: NativeScriptRendererFactory,
    useClass: NativeScriptRendererFactory,
    deps: [APP_ROOT_VIEW, NAMESPACE_FILTERS, NATIVESCRIPT_ROOT_MODULE_ID, [new Optional(), ENABLE_REUSABE_VIEWS]],
  },
  { provide: RendererFactory2, useExisting: NativeScriptRendererFactory },
  { provide: XhrFactory, useClass: NativescriptXhrFactory, deps: [] },
];'
```



Web Worker

Client

Desktop

Mobile

Server

Плюсы использования платформ

- Применяем экспертизу во фронте
- Имеем доступ ко всем возможностям Ангуляра
- А также к экосистеме библиотек Node.js
- Не тратим месяцы и годы на изучение Swift, C++ и Java

Минусы использования платформ

- Все еще медленнее, чем нативные приложения
- Дополнительный слой абстракции может вызвать сложности при отладке
- Не ко всем возможностям нативной платформы может быть доступ
- Дополнительные накладки на IPC

Что в итоге?

Что мы узнали?

Angular приложения можно запустить где угодно. Главное, чтобы поддерживался движок JS.

Что мы узнали?

За кроссплатформенность фреймворка отвечает механизм Angular Platforms.

75

Что мы узнали?

Платформа – это набор критически важных провайдеров для отображения элементов.

76

Что мы узнали?

Платформы образуют иерархию наследования, расширяя возможности родительских платформ.

Что мы узнали?

Чтобы создать свою платформу, необходимо предоставить собственную реализацию всех платформенных сервисов.

78

@CODEGUIDES

79

ТИНЬКОФФ

Спасибо!