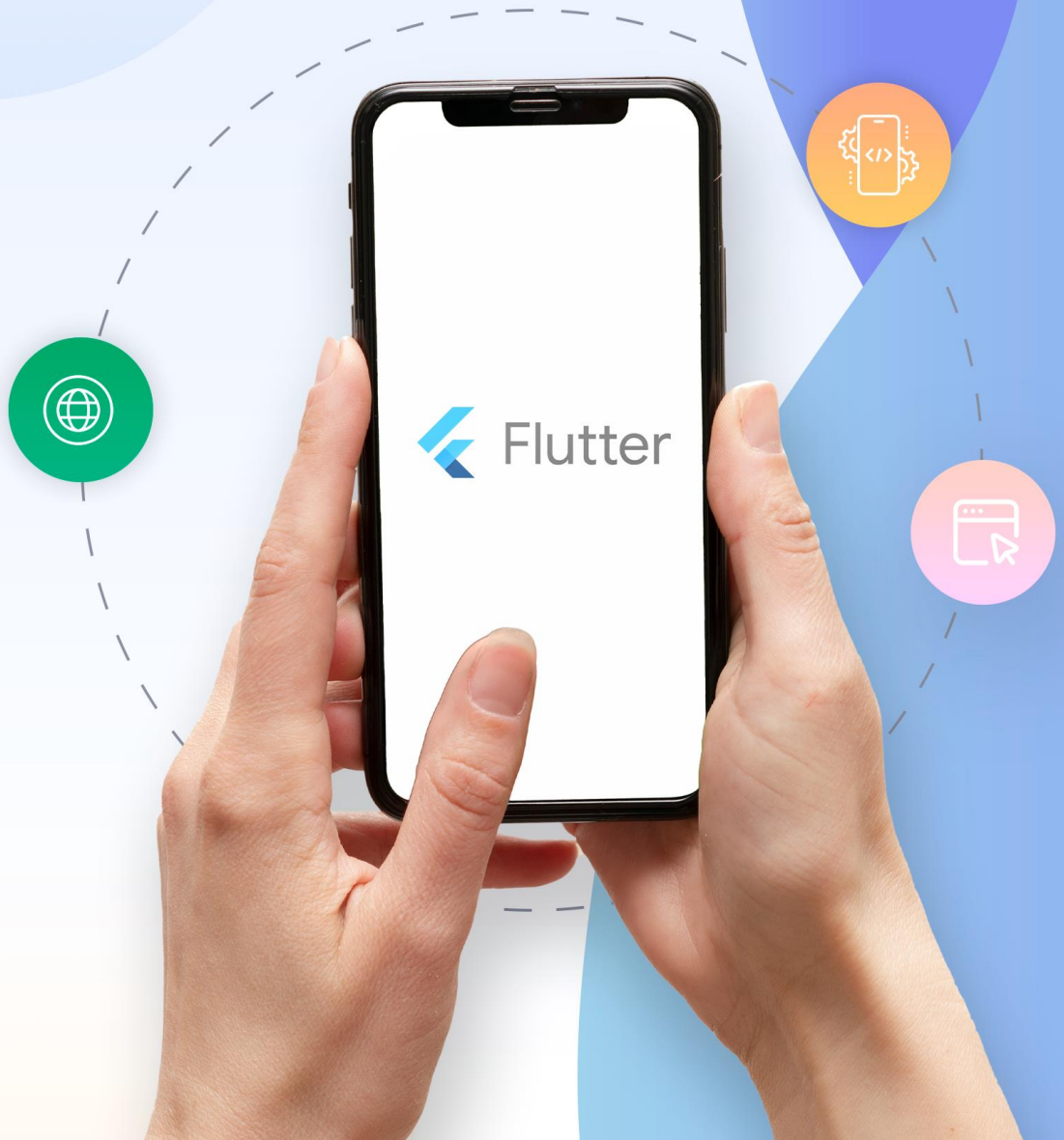
Ostad

FLUTTER

INTERVIEW QUESTIONS



১ আপনি ফ্লাটার বলতে কী বুঝেন?

ফ্লাটার হচ্ছে গুগলের তৈরিকৃত এক ধরনের সিঙ্গেল কোডবেসের ওপেন- সোর্স ফ্রেমওয়ার্ক যেটি দ্বারা মাল্টি-প্লাটফর্ম অ্যাপ্লিকেশন বিল্ড করা হয়। ফ্লাটার কোন ল্যাংগুয়েজ নয়, এটি হচ্ছে একটি SDK (Software Development Kit) যেটি ডার্ট প্রোগ্রামিং ল্যাংগুয়েজ ইউজ করে অ্যাপ ক্রিয়েট করে।

২

ডার্ট কী? ডার্ট (Dart) কী? এর প্রয়োজনীয়তা বলুন।

ডার্ট হচ্ছে একটি ক্লায়েন্ট-অপ্টিমাইজড প্রোগ্রামিং ল্যাংগুয়েজ এবং এটি দিয়ে ওয়েব এবং মোবাইল অ্যাপস এর জন্য ফ্রন্ট-এন্ড ইউজার ইন্টারফেস ক্রিয়েট করা হয়। শিখতে এবং ইউজ করতে সহজ, সময়ের সাথে ইভল্ভ ও ইম্প্রুভ হওয়ার কারণে সফটওয়্যার ডেভেলপমেন্ট সেক্টরে ডার্ট এর প্রয়োজনয়ীতা অনেক।

৩ ফ্লাটার উইজেটস (Flutter widgets) কী?

REACT থেকে ইন্সপায়ারড একটি মডার্ন ফ্রেমওয়ার্ক ইউজ করে ফ্লাটার উইজেটস বিল্ড করা হয়। ফ্লাটার উইজেটস হচ্ছে ফ্লাটার অ্যাপ্লিকেশন এর ইউজার ইন্টারফেস। এই ফ্লাটার উইজেটস কাস্টমাইজেবল এবং ভিন্ন ভিন্ন উইজেটস কম্বাইন করে আরোও কমপ্লেক্স উইজেট ক্রিয়েট করা যায়।

Ostad

৪ ফ্লাটারে স্টেটলেস (Stateless) এবং স্টেটফুল (Stateful) উইজেটসের মধ্যে পার্থক্য কী? বুঝিয়ে বলুন তো।

স্টেটফুল উইজেট হচ্ছে একটি মিউটেবল উইজেট কারণ একে তার লাইফটাইমে অনেকবার ড্র করা যায় এবং এটি তার ইনার ডাটাও পরিবর্তন করে পারে। স্টেটফুল উইজেট ইউজারকে স্ক্রিন রিফ্রেশ করতে অ্যালাও করে। এর কিছু উদাহরণ হলো চেকবক্স (Checkbox), রেডিও (Radio), স্লাইডার, ইঙ্কওয়েল, ফরম এবং টেক্সটফিল্ড।

একটি স্টেটলেস উইজেট ইমিউটেবল এবং এক্সটারনাল ইভেন্ট ব্যতীত নিজেকে কখনো রিবিল্ড করতে পারে না। এটি তার পুরো লাইফসাইকেলে স্ট্যাটিক থাকে। যেই UI এলিমেন্ট গুলো পরিবর্তন হয় না যেমন টেক্সট লেবেল অথবা আইকন, এইসব এলিমেন্ট রিপ্রেজেন্ট করার জন্য স্টেটলেস উইজেট ব্যবহার করা হয়।

৫ স্টেটফুল (Stateful) উইজেট লাইফসাইকেল কী? ব্যাখ্যা করুন।

স্টেটফুল উইজেট লাইফসাইকেলের কিছু স্পেসিফিক মেথড রয়েছে। সেগুলো হলোঃ

১) ক্রিয়েটস্টেট (createstate): এই মেথড উইজেট প্রথম ক্রিয়েট করার সময় ব্যবহার করা হয় এবং এর ওভাররাইট মেথড অবশ্যই এক্সিস্ট করতে হবে।

২) ইনিস্টেট (initState): ক্রিয়েটস্টেট দ্বারা স্টেট অবজেক্ট ক্রিয়েট করার পরে এই মেথড ব্যবহারকর হয়। এটি স্টেট এর উইজেট ইনিশিয়ালাইজ করতে ব্যবহার করা হয়।

৩) ডিডচেঞ্জডিপেন্ডেন্সিস (didchangedependencies): এই মেথডের প্রয়োজন পরে যখন উইজেট এর ডিপেন্ডেন্সিস চেঞ্জ হয়।

৪) বিল্ড (build): এই মেথডকে কল করা হয় উইজেট ট্রি বানানোর জন্য় এবং সব GUI রেন্ডার করা হয়। প্রত্যেকবার UI রেন্ডার করার সময় বিল্ড কল করতে হবে।

৫) ডিডআপডেটউইজেট (didUpdateWidge): এই মেথড তখনই কল করা হয় যখনি উইজেট আপডেট দেয়া হয়। এটি আগের উইজেটও প্রোভাইড করে এবং নতুন উইজেটের সাথেও তুলনা করা করতে দেয়।

৬) ডিএক্টিভেট (Deactivate): যখনই উইজেট ট্রি থেকে এই অবজেক্ট ট্রি রিমুভ করা হয় তখন ফ্রেমওয়ার্ক এই মেথড কল করে।

৭) ডিস্পোজ (Dispose): একে কল করা হয় যখন অবজেক্ট এবং এর স্টেট ট্রি থেকে পারমানেন্টলি রিমুভ করে দেয়া হয়।

৬ AppLifecycleState কী?

অ্যাপলাইফসাইকেলস্টেট মূলত চার ভাগে বিভক্ত।

১) ইনেক্টিভঃ এটি দিয়ে ইন্ডিকেট করে যে অ্যাপ্লিকেশন ইনেক্টীভ স্টেট এ এবং উইজার ইনপুট রিসিভ করছে না। (শুধু IOS এর জন্য)

২) পোজডঃ এটি দ্বারা বোঝায় যে অ্যাপ্লিকেশন উইজারের কাছে ভিজিবল নয়, কিন্তু ব্যাকগ্রাউন্ডে রান করছে।

৩) রিজিউমডঃ এর মানে অ্যাপ্লিকেশন রান করছে এবং ইউজার ইনপুটে রেস্পন্ড করছে।

৪) সাস্পেন্ডিংঃ অর্থাৎ অ্যাপ্লিকেশন কিছুক্ষণের জন্য সাস্পেন্ডেড আছে। (শুধু android এর জন্য)

Ostad

৭

আচ্ছা বলুন তো pubspec.yaml ফাইল কী?

আপনি আপনার প্রজেক্টে ইনক্লুড করতে চান, এমন ছবি/ফন্ট/থার্ড-পার্টি প্যাকেজ ইম্পোর্ট করার জন্য pubspec. yaml file দায়ী। pubspec. yaml file আপনার ফ্লাটার প্রজেক্টের ডিপেন্ডেন্সি নির্ধারণ করতে ব্যবহৃত হয়। এই মেটাডেটা ইনফরমেশন YAML ল্যাঙ্গুয়েজে লেখা হয়। এই ফাইলটিতে fields name, version, description, homepage, repository, documentation, dependencies, environment- সহ pubspec.yaml file এর আরও অনেক কিছু থাকতে পারে।

৮ একটি ফ্লাটার প্যাকেজ এবং ফ্লাটার প্লাগইনের মধ্যে পার্থক্য কী? বুঝিয়ে বলুন।

একটি "প্যাকেজে" শুধুমাত্র ডার্ট কোড থাকে।

একটি "প্লাগইন"-এ ডার্ট এবং নেটিভ কোড উভয়ই থাকে (kotlin/js/swift/...)

Flutter plugins: (নেটিভ রিলেটেড ডেভেলপমেন্টস)

ফ্লাটার প্লাগইন হচ্ছে অ্যান্ড্রয়েড (কোটলিন বা জাভা) এবং iOS (সুইফ্ট বা অবজেক্টিভ সি) এর মতো নেটিভ কোডের একটি র‍্যাপার। ফ্লাটার প্ল্যাটফর্ম চ্যানেল এবং ম্যাসেজ পাসিংয়ের মাধ্যমে একটি নেটিভ অ্যাপ্লিকেশন যা করতে পারে, এর সবটুকুই করতে সক্ষম। Flutter নেটিভ iOS/Android কোডকে কাজ করার ইন্সট্রাকশন দিলে এর রেজাল্ট আবার ডার্টে ফেরত আসে।

Flutter packages or modules: (ইউটিল লাইব্রেরি থেকে কোড ব্যবহারের মাধ্যমে ডেভেলপমেন্ট আরও দ্রুত করে)

ফ্লাটার এবং ডার্ট ইকোসিস্টেম উভয়ই অন্যান্য ডেভেলপারদের দ্বারা তৈরি শেয়ার্ড প্যাকেজ ইউজ করার পারমিশন দেয়। তাই এটি স্ক্র্যাচ থেকে শুরু না করেই দ্রুত অ্যাপ ডেভেলপ করার কাজ সম্ভব করে তোলে।

Ostad

৯ WidgetsApp এবং MaterialApp এর মধ্যে পার্থক্য কী?

WidgetsApp	MaterialApp
WidgetsApp হলো একটি বেইসিক ফ্লাটার উইজেট যা ফ্লাটার অ্যাপ্লিকেশন তৈরির জন্য Core Functionality প্রদান করে।	এটি WidgetsApp-এর একটি সুনির্দিষ্ট ইমপ্লিমেন্টেশন যা Google-এর তৈরি মেটেরিয়াল ডিজাইন গাইডলাইন ফলো করে।
এটি একটি versatile widget যা আপনাকে কাস্টম থিম, লেআউট এবং রাউটিং সহ অ্যাপ তৈরি করতে সাহায্য করে।	মেটেরিয়াল ডিজাইন হলো একটি ডিজাইন ল্যাঙ্গুয়েজ যা Android, iOS এবং ওয়েব সহ বিভিন্ন প্ল্যাটফর্ম জুড়ে একটি সামঞ্জস্যপূর্ণ এবং ভিজ্যুয়ালি এট্রাক্টিভ UI প্রদান করে।
WidgetsApp কোনো নির্দিষ্ট ভিজ্যুয়াল ডিজাইন বা স্টাইলিং প্রয়োগ না করে সম্পূর্ণ কাস্টম লুক এবং ফিল সহ অ্যাপ তৈরি করার জন্য উপযুক্ত হিসেবে কাজ করে।	MaterialApp অ্যাপটিকে একটি প্রিডিফাইন্ড ভিজ্যুয়াল স্টাইল এবং ডিফল্ট উইজেটগুলির সাথে সেট আপ করে যা মেটেরিয়াল ডিজাইন প্রিন্সিপ্যালগুলো মেনে চলে।

১০ Hot Reload এবং Hot Restart এর মধ্যে পার্থক্য কী? আইডিয়া আছে?

Ostad

ফ্লাটারে, "হট রিলোড" এবং "হট রিস্টার্ট" হলো সম্পূর্ণ অ্যাপ্লিকেশন রিস্টার্ট না করেই ডেভেলপমেন্টের সময় আপনার অ্যাপ আপডেট এবং পরীক্ষা করার দুটি ভিন্ন উপায়।

Hot Reload	Hot Restart
হট রিলোড আপনাকে আপনার ফ্লাটার কোডে পরিবর্তন করতে দেওয়ার পাশাপাশি, অ্যাপে পরিবর্তিত আপডেটগুলি প্রায় সাথে সাথেই দেখতে সুযোগ দেয়।	হট রিস্টার্ট ফ্লাটার অ্যাপটিকে রিস্টার্ট করে। অর্থাৎ, রানিং অ্যাপটি বন্ধ করে, কোডটি ইনজেক্ট করার পর আবার প্রথম থেকে শুরু করে। তাই যখন আপনি একটি হট রিস্টার্ট করেন, তখন অ্যাপটি তার অবস্থা হারায় এবং টেম্পোরারি ডেটা বা ইনফরমেশন রিসেট হয়।
আপনি আপনার অ্যাপের বর্তমান অবস্থা না হারিয়ে বা স্ক্র্যাচ থেকে অ্যাপ রিস্টার্ট না করেই পরিবর্তনগুলো দেখতে পারবেন।	আপনার অ্যাপে যদি কোনো লগইন স্ক্রিন থাকে, তাহলে হট রিস্টার্টের পর আপনাকে আবার প্রাথমিক লগইন স্ক্রিনে ফেরত নিয়ে যাওয়া হবে। এজন্য হট রিস্টার্ট হট রিলোডের তুলনায় বেশি সময় নেয়।

১১

আপনি কী একটি scaffold nest করতে পারবেন? পারলে কেনো পারবেন অথবা না পারলে কেন পারবেন না? বুঝিয়ে বলুন।

হ্যাঁ, আপনি একটি scaffold nest করতে পারবেন। ফ্লাটারের অন্যতম সুবিধা হলো এখানে আপনি পুরো UI কন্ট্রোল করতে পারবেন। Scaffold হলো একটি উইজেট। তাই একটি উইজেট যেখানে যেখানে যাওয়ার ক্ষমতা রাখে, Scaffold ও সেখানে যেতে পারে। আপনি Scaffolds নেস্টিং- এর মাধ্যমে drawers, snack bars, এবং bottom sheets ও লেয়ার করতে পারবেন।

১২ ফ্লাটারের কিছু প্রধান বৈশিষ্ট্য বর্ণনা করুন। যাতে ক্লাস ফাইভের স্টুডেন্ট বুঝতে পারে।

Fast Development: কমপ্লিটলি কাস্টমাইজযোগ্য উইজেটগুলির একটি সেট ব্যবহারের মাধ্যমে আপনি ফ্লাটারে মিনিটের মধ্যেই নেটিভ ইন্টারফেস তৈরি করতে পারবেন।

Expressive and Flexible UI: ফ্লাটারের লেয়ারড আর্কিটেকচার আপনাকে আপনার UI সম্পূর্ণরূপে কাস্টমাইজ করতে সাহায্য করে। এর ফলে দ্রুত রেন্ডারিং এবং এক্সপ্রেসিভ ডিজাইন করতে সুবিধা হয়।

Native Performance: ফ্লাটারে উপস্থিত উইজেটগুলি সব গুরুত্বপূর্ণ প্ল্যাটফর্মের পার্থক্য যেমনঃ স্ক্রলিং, নেভিগেশন, আইকন এবং আরও অনেক কিছুকে অন্তর্ভুক্ত করে। এটি সব প্ল্যাটফর্মে একটি সম্পূর্ণ নেটিভ পারফরম্যান্স দেয়।

Ostad

১৩ কীভাবে আপনি widget rebuild রিডিউস করবেন?

উইজেট রিবিল্ড রিডিউস করতে কিছু সহজ টিপস ফলো করা যেতে পারে। যেমন-

১। অ্যাপের ইউজার ইন্টারফেসকে ছোট উইজেটে ভাগ করা। এভাবে, কোনো চেঞ্জ আসলে পুরো স্ক্রিন রিবিল্ড হওয়ার পরিবর্তে শুধুমাত্র ইফেক্টেড উইজেটগুলো রিবিল্ড হয়।

২। উইজেটগুলির ডেটা এবং স্টেট সাবধানে ম্যানেজ করুন। প্রয়োজন ছাড়া স্টেটফুল উইজেটগুলো ব্যবহার না করাই ভালো। রিক্রিয়েটের বদলে উইজেট প্রোপার্টিগুলো আপডেট করা ভালো। এটি অপ্রয়োজনীয় রিবিল্ড এড়ায়।

৩। নতুন উইজেট ক্রিয়েট করার পরিবর্তে চেঞ্জ রিফ্লেক্ট করার জন্যে এক্সিস্টিং উইজেটগুলোই আপডেট করে ব্যবহার করুন। উইজেটের রিইউজের ফলে উইজেট ট্রি রিবিল্ডের প্রয়োজনীয়তা কমে আসে।

৪। সম্ভব হলে স্টেটলেস উইজেটের জন্য const constructors ব্যবহার করুন। এটি কম্পাইল-টাইম অপ্টিমাইজেশান পসিবল করে এবং রিবিল্ড কমায়।

৫। অপ্রয়োজনীয় রিবিল্ড ট্রিগার করতে পারে এমন একশনগুলো নিয়ে সতর্ক থাকুন। যেমন- সেটস্টেটকে শুধু শুধু কল করা বা UI কে ইফেক্ট করেনা এমন এমন ইভেন্টগুলিতে রেসপন্ড করা।

১৪ আচ্ছা Buildcontext কী এবং এটি কীভাবে কার্যকরী? বুঝিয়ে বলতে পারবেন?

Buildcontext হচ্ছে Element tree-তে উইজেটের এলিমেন্ট। তাই প্রত্যেক উইেটেরই নিজস্ব Buildcontext আছে। এটি উইজেটের পজিশন এবং কনফিগারেশন সম্পর্কিত বিভিন্ন সার্ভিস এবং তথ্যে এক্সেস দেয়। সাধারণত থিম বা অন্য উইজেটের রেফারেন্স পেতে BuildContext ব্যবহার করা হয়। যেমন- আপনি যদি material dialog দেখাতে চান, তাহলে আপনার scaffold এর একটি রেফারেন্স প্রয়োজন। আপনি এটি Scaffold.of (context) দিয়ে পেতে পারেন, যেখানে Context হলো Buildcontext। এখানে, of() সবথেকে কাছের scaffold না পাওয়া পর্যন্ত ট্রি এর সন্ধান চালাতে থাকে।

১৫

কীভাবে আপনি ফ্লাটার অ্যাপের মধ্যেকার নেটিভ কোডের সাথে যোগাযোগ করবেন? করা সম্ভব?

সাধারণত আপনাকে নেটিভ কোডের সাথে যোগাযোগের দরকার নেই কারণ ফ্লাটার ফ্রেমওয়ার্ক বা থার্ড-পার্টি প্লাগইনগুলি এটি ম্যানেজ করে। তবুও, স্পেশাল এক্সেসের প্রয়োজনীয়তাবোধ করলে প্ল্যাটফর্ম চ্যানেলগুলো ব্যবহার করা যেতে পারে।

একধরনের প্ল্যাটফর্ম চ্যানেল হলো method channel। ডার্ট সাইডে ডেটা সিরিয়ালাইজ করা হয় এবং পরে নেটিভ সাইডে পাঠানো হয়। এ সিরিয়ালাইজড ম্যাসেজ ফেরত পাঠানোর আগে আপনি প্ল্যাটফর্মের সাথে ইন্টারঅ্যাক্ট করার জন্য নেটিভ কোড লিখতে পারেন।

দ্বিতীয় ধরনের প্ল্যাটফর্ম চ্যানেল হলো ইভেন্ট চ্যানেল, যা আপনি নেটিভ প্ল্যাটফর্ম থেকে ফ্লাটারে ডেটার একটি স্ট্রিম পাঠাতে ব্যবহার করতে পারেন। এটি সেন্সর ডেটা মনিটর করার জন্য প্রয়োজনীয়।

১৬ এটি দিয়ে আপনি কি কি ধরনের টেস্ট পারফর্ম করতে পারবেন?

সফ্টওয়্যার ডেভেলপমেন্টে, একটি সফ্টওয়্যার অ্যাপ্লিকেশনের কোয়ালিটি, ফাংশনালিটি এবং নির্ভরযোগ্যতা নিশ্চিত করতে বিভিন্ন ধরনের পরীক্ষা করা যেতে পারে। কিছু সাধারণ ধরনের পরীক্ষাগুলো হলো:

Unit Tests: ইউনিট টেস্ট আলাদা কম্পো্নেন্ট বা কোডের ইউনিট আলাদাভাবে পরীক্ষা করার উপর ফোকাস করে। এরা ছোট এবং self-contained ইউনিটের ফাংশনালিটি এবং সঠিকতা যাচাই করতে ব্যবহৃত হয়। যেমন- functions, methods, বা classes।

Integration Tests: ইন্টিগ্রেশন টেস্টগুলি একটি সিস্টেমের বিভিন্ন কম্পোনেন্ট বা মডিউলের মধ্যের interaction এবং integration ভেরিফাই করে। এর কাজ হলো এটি নিশ্চিত করা যে, একাধিক ইউনিট কম্বাইন্ড হওয়ার পরেও সঠিকভাবে কাজ করছে।

Functional Tests: ফাংশনাল টেস্ট শেষ ব্যবহারকারীর দৃষ্টিকোণ থেকে সফ্টওয়্যারটির বিহেভিয়ার এবং ফাংশনালিটি ভেরিফাই করে। এই পরীক্ষাগুলি যাচাই করে যে অ্যাপ্লিকেশনটি তার উদ্দেশ্য অনুযায়ী কাজ করছে কিনা।

১৭

বিভিন্ন স্টেট ম্যানেজমেন্ট সমাধানের সুবিধা এবং অসুবিধাগুলো কী কী?

কিছু জনপ্রিয় স্টেট ম্যানেজমেন্ট সমাধানগুলির মধ্যে রয়েছে BLoC, ChangeNotifierProvider, Redux, MobX, RxDart ইত্যাদি। এগুলি মাঝারি থেকে বড় আকারের অ্যাপগুলির জন্য উপযুক্ত। যদি আপনি শুধুমাত্র একটি দ্রুত ডেমো অ্যাপ তৈরি করতে আগ্রহী হোন তাহলে একটি স্টেটফুল উইজেটই যথেষ্ট। যদি আপনার হাতে অসংখ্য অপশন থাকে, আপনার অবশ্যই চেষ্টা করা উচিত এমন একটি সল্যুশন চুজ করা যেটি মেন্টালি এক্সসেপ্ট করে নিতে কম সময় নিবে। এমন অবস্থায়, ChangeNotifier with Provider বা MobX একটি ভালো চয়েজ।

আপনি যদি ফায়ারবেস এপিআই-এর মতো স্ট্রিমগুলির উপর খুব বেশি নির্ভরশীল হোন, তাহলে BLoC বা RxDart-এর মতো একটি স্ট্রিম-ভিত্তিক সমাধান বেছে নেওয়া বেটার। এবং আপনার যদি redo/undo এর মতো ফাংশনালিটিগুলো প্রয়োজন হয় তাহলে BLoC or Redux এর মতো সমাধান চুজ করা উচিত যা অপরিবর্তনীয় অবস্থাগুলোকে ভালোভাবে পরিচালনা করে।

১৮

ফ্লাটারে Stream কী? বলতে পারবেন?

একটি স্ট্রিম হলো অ্যাসিঙ্ক্রোনাস ইভেন্টের একটি সিকোয়েন্স। এটি ডেটার একটি অ্যাসিঙ্ক্রোনাস সিকোয়েন্স প্রোভাইড করে। এটি একটি পাইপের মতো

যেখানে যদি আমরা এক প্রান্তে কিছু ভ্যালু রাখি এবং অন্য প্রান্তে যদি আমাদের একজন শ্রোতা থাকে তবে তিনি এই মানটি গ্রহণ করবে।

১৯

আচ্ছা বলুন তো কীভাবে ডার্টে private variables তৈরি করবেন?

ডার্টে ভ্যারিয়েবলের নামের আগে, এটিকে প্রাইভেট হিসেবে ডিক্লেয়ার করতে '_' (underscore) ব্যবহার করা হয়। এখানে private variables

বোঝায় যে, এটি যে ফাইলটিতে রয়েছে সেখানে এটি অ্যাক্সেসযোগ্য এবং অন্য ফাইলগুলিতে অ্যাক্সেসযোগ্য নয়।

Ostad

২০

Event loop কী? এবং আইসোলেট-এর সাথে এর সম্পর্ক কী? বুঝিয়ে বলুন।

ফ্লাটারে ইভেন্ট লুপ হলো একটি সেন্ট্রাল আইডিয়া যা একটি অ্যাপের মধ্যে ফ্লো অব কন্ট্রোল ম্যানেজ করতে সাহায্য করে। এটি ইভেন্টগুলোর প্রসেসিং এবং সেই ইভেন্টগুলির রেসপন্স অনুযায়ী অ্যাপের অবস্থা আপডেট করার জন্য দায়ী।

আইসোলেট হলো এক্সিকিউশনের একটি আলাদা থ্রেড যা অ্যাপের মূল থ্রেড থেকে আইসোলেটেড। ফ্লাটারে আইসোলেট ব্যবহার করা হয় কোডের সমসাময়িক এক্সিকিউশনের জন্য। তাই যেসব টাস্ক শেষ করতে অনেক সময় লাগে, এমন কাজগুলোর জন্য এটি হেল্পফুল। যেমন- নেটওয়ার্ক রিকোয়েস্ট বা computationally intensive operations।

২১

ফ্লাটারে tree shaking কী?

ট্রি শেকিং ডেড কোড এলিমিনেশন নামেও পরিচিত। এটি একটি টেকনিক যা ফ্লাটার (এবং অন্যান্য প্রোগ্রামিংএনভায়রনমেন্টে) বিল্ড প্রসেস চলাকালীন অব্যবহৃত কোড রিমুভ করতে ব্যবহৃত হয়। আসল অ্যাপ্লিকেশন দ্বারা ব্যবহৃত হয় না এমন কোডগুলো বাদ দিয়ে ফাইনাল অ্যাপ্লিকেশন এর সাইজ ছোট করার জন্য এটি ব্যবহৃত হয়।

Ostad

২২

ফ্লাটারে DevTools কী?

ফ্লাটারে DevTools হলো পারফরম্যান্স ম্যানেজমেন্ট এবং ডিবাগিংয়ের জন্য ব্যবহৃত টুলের একটি সেট। এই টুলগুলির সাহায্যে UI layout মনিটর করতে পারবেন, UI performance issues গুলো নির্ণয় করতে পারবেন সোর্স-লেভেল ডিবাগিং করার পাশাপাশি সাধারণ লগ এবং ডায়াগনস্টিক তথ্যও দেখতে পারেন। এই টুলটি এখনও প্রিভিউ রিলিজে থাকলেও আপনি DevTools-এর উপরের-ডান কোনায় “beaker” আইকনে ক্লিক করে এই টুলের আলফা সংস্করণ পরীক্ষা করতে পারেন।

২৩

ফ্লাটারে Flex উইজেট কী?

ফ্লাটারে ফ্লেক্স উইজেট হলো একটি ফ্লেক্সিবল লেআউট উইজেট যা রো এবং কলাম উইজেটের কম্বিনেশনে ব্যবহৃত হয়। এটি ফ্লেক্স ফ্যাক্টরগুলির উপর ভিত্তি করে তার চিলড্রেন এর মধ্যে অ্যাভেইলেবল স্পেস ডিস্ট্রিবিউট করে। এটি ডেটারমাইন করে যে প্রতিটি চিলড্রেন কতটুকু জায়গা পাবে। ফ্লেক্স উইজেটগুলি ডেভেলপারদের রেসপন্সিভ লেআউটগুলি তৈরিতে সাহায্য করে যা ফ্লেক্স ফ্যাক্টর এবং অ্যালাইনমেন্ট প্রোপার্টিগুলো এডজাস্ট করে বিভিন্ন স্ক্রিন সাইজের সাথে সহজেই খাপ খায়। টি ফ্লাটারে ডায়নামিক ইউজার ইন্টারফেস তৈরি করার জন্য একটি শক্তিশালী টুল যা ফ্লেক্সিবল এবং এডাপ্টেবল লেআউট তৈরি করতে সাহায্য করে।

Ostad

২৪

Expanded এবং flexible উইজেটগুলোর মধ্যে পার্থক্য কী?

Flutter-এ Expanded এবং Flexible উইজেট, দুটিই ব্যবহার করা হয় ফ্লেক্সিবল লেআউট তৈরি করার ক্ষেত্রে। কিন্তু এরা এদের অ্যাভেইলেবল স্পেস যেভাবে তাদের চিলড্রেনদের মধ্যে ডিস্ট্রিবিউট করে, এক্ষেত্রে এদের মধ্যে কিছু পার্থক্য আছে। যেমন-

রো এবং কলামে উইজেটগুলির সাইজ পরিবর্তন করতে ফ্লেক্সিবল ব্যবহার করা হয়। এটি মূলত ব্যবহার করা হয় প্যারেন্ট উইজেটের সাথে সম্পর্ক রাখার পাশাপাশি ভিন্ন ভিন্ন চাইল্ড উইজেটের স্পেস এডজাস্ট করার জন্য।

এদিকে Expanded উইজেট রো এবং কলামের চিলড্রেনদের সীমাবদ্ধতাগুলো পরিবর্তন করে যার ফলে খালি স্পেসগুলো ফিল আপ করা সম্ভব হয়।

তাই, আপনার চাইল্ডকে আপনি যদি এক্সপান্ডেড উইজেটে র‍্যাপ করেন, তাহলে এটি খালি স্পেসগুলো ফিল আপ করবে।

২৫

Material বনাম Cupertino উইজেট কী?

ম্যাটেরিয়াল এবং কিউপারটিনো হলো ফ্লাটারে ব্যবহৃত দুটি ডিজাইনের ল্যাঙ্গুয়েজ। এটি ইন্টারফেস তৈরির জন্য ব্যবহৃত হয় যা Android (ম্যাটেরিয়াল ডিজাইন) এবং iOS (Cupertino) এর ডিজাইন গাইডলাইন মেনে চলে। প্রতিটি ডিজাইন ল্যাঙ্গুয়েজ এরই নিজস্ব উইজেট সেটসহ নির্দিষ্ট ভিজ্যুয়াল স্টাইল এবং বিহেভিয়ার রয়েছে।

Material Widgets:

1. ম্যাটেরিয়াল উইজেটগুলি Google-এর মেটেরিয়াল ডিজাইন গাইডলাইন এবং এস্থেটিক ফলো করে, যা Bold Color, Shadow এবং Depth Effect দ্বারা চিহ্নিত করা হয়।

2. ম্যাটেরিয়াল উইজেটগুলো অ্যান্ড্রয়েড এবং iOS উভয় প্ল্যাটফর্মের সাথে সামঞ্জস্যপূর্ণ, যা তাদেরকে ক্রস-প্ল্যাটফর্ম ডেভেলপমেন্টের জন্য উপযোগী করে তোলে।

3. ম্যাটেরিয়াল উইজেটগুলি কম্পোনেন্টস এর একটি ওয়াইড রেঞ্জ অফার করে। যেমন- buttons, cards, app bars, dialogs, এবং navigation elements যা ডেভেলপারদের স্ট্যান্ডার্ড ইন্টারেক্টিভ ইন্টারফেস তৈরি করতে সাহায্য করে।

Cupertino Widgets:

1. কিউপারটিনো উইজেটগুলি অ্যাপলের iOS ডিজাইন গাইডলাইন ফলো করে। এটি নেটিভ iOS লুক এবং মিনিমালিস্ট এস্থেটিক ফিল প্রদান করে।

2. Cupertino উইজেটগুলি iOS এর নির্দিষ্ট ফিচারগুলি অফার করে যেমন- swipe gestures, navigation bars, segmented controls, এবং iOS-style picker। যা ডেভেলপারদের iOS স্ট্যান্ডার্ড ইন্টারেক্টিভ ইন্টারফেস তৈরি করতে সাহায্য করে।

3. Cupertino উইজেটগুলি iOS-নির্দিষ্ট আচরণ এবং ইন্টারেকশন অনুকরণ করে যা iOS ব্যবহারকারীদের ডিভাইসে একটি অথেনটিক এক্সপেরিয়েন্স প্রদান করে।

২৬

Null Aware Operator কী?

ডার্টে Null Aware Operator আপনাকে একটি মান নাল কিনা তার উপর ভিত্তি করে গণনা করতে দেয়। এটি একটি longer expression এর সংক্ষিপ্ত মেথড। ডার্টে একটি null-aware operator একটি nullable মানকে আবার ইউজেবল করে তোলার ক্ষেত্রে একটি হেল্পফুল টুল।

Null Aware Operator এর সবচেয়ে সাধারণ ব্যবহার হলো যখন একজন ডেভেলপার সার্ভার থেকে JSON ডেটা পার্স করতে চায়। JSON পার্স করার পরে ইউজার JSON খালি কিনা তা IF-Else কন্ডিশন ব্যবহার করে খুব সহজেই পরীক্ষা করতে পারেন।

২৭

Form, textfield এবং textFormField কী?

আপনি যদি একটি ফর্ম তৈরি করেন যেখানে আপনাকে save, reset বা অপারেশন ভেলিডেট এর মতো কাজগুলো করতে হবে, এক্ষেত্রে TextFormField ব্যবহার হেল্পফুল। অন্যথায় সাধারণ ব্যবহারকারীর ইনপুট ক্যাপচারের জন্য টেক্সটফিল্ড যথেষ্ট। TextFormField ফর্ম উইজেটের সাথে একত্রিত হয়। এটি একটি সুবিধাজনক উইজেট যা একটি FormField এ একটি TextField উইজেটকে র‍্যাপ করে। এটি ফর্মটি সহজভাবে সেইভ করা, পুনরায় সেট করা বা একাধিক ক্ষেত্র ভেলিডেট করা সহজ করে তোলে। এক্ষেত্রে, ফর্ম ছাড়া ব্যবহার করতে, কনস্ট্রাক্টরকে একটি GlobalKey প্রোভাইড করুন এবং এবং ফর্ম ফিল্ড সেইভ বা রিসেট করতে GlobalKey.currentState ব্যবহার করুন।

২৮

StreamBuilder এবং FutureBuilder এর মধ্যে পার্থক্য কী?

StreamBuilder এবং FutureBuilder উভয়ই ফ্লাটারের উইজেট যা অ্যাসিঙ্ক্রোনাস অপারেশন কন্ডাক্ট করতে এবং অপারেশনের অবস্থার উপর ভিত্তি করে UI আপডেট করতে ব্যবহৃত হয়। তবে তাদের অ্যাসিঙ্ক্রোনাস অপারেশন হ্যান্ডেল করা এবং আপডেটগুলো কন্ডাক্ট করার ধরনের উপর ভিত্তি করে এদের কিছু পার্থক্য রয়েছে।

StreamBuilder:

1. স্ট্রিমবিল্ডার ব্যবহার করা হয় যখন অ্যাসিঙ্ক্রোনাস অপারেশনগুলির সাথে কাজ করা হয় যা ডেটার একটি প্রবাহ ফেরত দেয়। একটি স্ট্রিম সময়ের সাথে ঘটে যাওয়া ইভেন্টগুলির একটি সিকোয়েন্স রিপ্রেজেন্ট করে।

2. StreamBuilder ইনপুট হিসেবে একটি স্ট্রিম নেয় এবং যখনই স্ট্রিমে নতুন ডেটা বিল্ড হয় তখনই UI পুনরায় তৈরি করে।

3. StreamBuilder-এর বিল্ডার কলব্যাক একটি BuildContext এবং একটি AsyncSnapshot প্যারামিটার হিসেবে পায়। এখানে AsyncSnapshot-এ স্ট্রিম থেকে প্রাপ্ত ডেটাগুলো থাকে।

4. StreamBuilder কন্টিনিউয়াস ডেটার স্ট্রিম হ্যান্ডেল করতে পারে। যেমন- chat message stream

FutureBuilder:

1. FutureBuilder ব্যবহার করা হয় যখন এমন অ্যাসিঙ্ক্রোনাস অপারেশনগুলির সাথে কাজ করা হয়, যেটি একটি সিঙ্গেল ফিউচার রেজাল্ট রিটার্ন করে। ফিউচার এমন একটি ভ্যালু রিপ্রেজেন্ট করে যা ফিউচারে কোনো একটি সময় অ্যাভেইলেবল হবে।

2. FutureBuilder ইনপুট হিসেবে একটি ফিউচার নেয় এবং ফিউচারের স্টেটের উপর ভিত্তি করে UI রিবিল্ড করে। এখানে ফিউচারের স্টেটগুলো হলোঃ

ConnectionState.none, ConnectionState.waiting, ConnectionState.done

3. FutureBuilder-এর বিল্ডার কলব্যাক একটি BuildContext এবং একটি AsyncSnapshot প্যারামিটার হিসেবে পায়। এখানে AsyncSnapshot-এ ফিউচার এর রেজাল্ট বা কোনো এরর হলে তার রেজাল্ট থাকে।

4. যখন আপনাকে একবার ডেটা আনতে হবে এবং সাথে সাথে এর রেজাল্ট ডিসপ্লে করতে হবে, এমন সব সিচুয়েশনের জন্য FutureBuilder উপযোগী। যেমন- একটি API কল করা এবং এর রেজাল্ট ডিসপ্লে করা।

২৯

Future কী?

একটি ফিউচার একটি সম্ভাব্য মান, বা একটি এরর রিপ্রেজেন্ট করতে ব্যবহৃত হয়, যা ভবিষ্যতে কোনো একটি সময় এভাইলেবল হবে। ডার্টে অ্যাসিঙ্ক্রোনাস অপারেশন হ্যান্ডেল করতে, আপনি ফিউচার ক্লাস এবং অ্যাসিঙ্ক এবং ওয়েট কি-ওয়ার্ড ব্যবহার করতে পারেন। একটি ফিউচার (লোয়ার কেইস "f") হলো ফিউচার (ক্যাপিটালাইজড "F") ক্লাসের একটি উদাহরণ। একটি ফিউচার একটি অ্যাসিঙ্ক্রোনাস অপারেশনের রেজাল্ট রিপ্রেজেন্ট করে এবং এর দুটি অবস্থা থাকতে পারে: uncompleted or completed

Ostad

৩০

Synchronous operation এবং synchronous function এর মধ্যে পার্থক্য কী? Asynchronous operation এবং asynchronous function এর মধ্যে পার্থক্য কী?

Synchronous operation: একটি সিঙ্ক্রোনাস অপারেশন অন্যান্য অপারেশনগুলি সম্পূর্ণ না হওয়া পর্যন্ত অন্য আরেকটি অপারেশন এক্সেকিউশন করা থেকে বাধা দেয়।

Synchronous function: একটি সিঙ্ক্রোনাস ফাংশন শুধুমাত্র সিঙ্ক্রোনাস অপারেশন পারফর্ম করে।

Asynchronous operation: একবার শুরু হলে, একটি অ্যাসিঙ্ক্রোনাস অপারেশন একটি সম্পূর্ণ হওয়ার আগেই অন্যান্য অপারেশনগুলো চালানো এলাও করে।

Asynchronous function: একটি অ্যাসিঙ্ক্রোনাস ফাংশন কমপক্ষে একটি অ্যাসিঙ্ক্রোনাস অপারেশন এক্সেকিউট করে এবং এর পাশাপাশি সিঙ্ক্রোনাস অপারেশনও হ্যান্ডেল করতে পারে।

০১ Method এবং function এর মধ্যে পার্থক্য কী?

ফ্লাটারে, মেথড এবং ফাংশনের মধ্যে পার্থক্যগুলো সাধারণ প্রোগ্রামিংয়ের মতোই সিমিলার কন্সেপ্টগুলো ফলো করে। যেমন-

Function	Method
১। একটি ফাংশন হলো কোডের একটি self-contained ব্লক যা একটি স্পেসিফিক টাস্ক পারফর্ম করে এবং প্রোগ্রামের যেকোনো জায়গা থেকে কল করা যেতে পারে।	১। একটি মেথড হলো একটি ফাংশন যা একটি অবজেক্ট বা একটি ক্লাসের সাথে এসোসিয়েটেড এবং তাকে শুধুমাত্র সেই ক্লাসের ইন্সট্যান্সেই কল করা যেতে পারে।
২। ফাংশনগুলোকে ইন্ডিপেন্ডেন্টলি ইনভোক করা যেতে পারে। কোনো নির্দিষ্ট ক্লাস বা অবজেক্টের সাথে আবদ্ধ হওয়ার প্রয়োজনীয়তা নেই।	২। মেথডগুলো শুধু একটি ক্লাসের objects বা instances এ invoked হতে পারে।
৩। ফাংশনগুলি public বা private হতে পারে এবং প্রোগ্রামের অন্য পার্ট থেকে এর visibility এবং accessibility কন্ট্রোল করতে পারে।	৩। মেথডগুলোর একটি class hierarchy এর মধ্যে নিজেদের accessibility ডিফাইন করতে বিভিন্ন এক্সেস মডিফায়ার থাকতে পারে যেমন- public, private, protected।

০২

Responsive এবং adaptive অ্যাপগুলো দ্বারা কী বোঝানো হয়?

Responsive: রেসপন্সিভ অ্যাপ্লিকেশনগুলি একটি ফ্লেক্সিবল লেআউট এবং ডিজাইনিং টেকনিকগুলো ব্যবহার করে রেসপন্স জানায় এবং বিভিন্ন স্ক্রিনের সাইজের সাথে মানিয়ে নেয়। এরা ডাইনামিকালি UI এলিমেন্ট এবং কন্টেন্টগুলো অ্যাভেইলেবল স্পেস এর উপর ভিত্তি করে মানিয়ে নেয়। রেসপন্সিভ ডিজাইনগুলো সাধারণত fluid grids, flexible images, এবং CSS media queries এর উপর নির্ভর করে এটি নিশ্চিত করতে যে অ্যাপটি ঠিকমতো ফাংশন করছে। এই অ্যাপের লেআউট এবং কন্টেন্টগুলো রিএরেঞ্জ বা রিসাইজও করা যায়। রেসপন্সিভ অ্যাপগুলো সাধারণত HTML, CSS, এবং JavaScript এর মতো ওয়েব টেকনোলজি ব্যবহার করে তৈরি করা হয়।

Adaptive: অন্যদিকে Adaptive অ্যাপগুলো যে ডিভাইস বা প্ল্যাটফর্মে ব্যবহার করা হচ্ছে তার উপর ভিত্তি করে তাদের appearance এবং ফাংশনালিটি পরিবর্তন এবং মানিয়ে নেওয়ার জন্য ডিজাইন করা হয়েছে। এই অ্যাপগুলো নির্দিষ্ট ডিভাইসের ক্যারেক্টারিস্টিকগুলো বোঝে। যেমন- স্ক্রিন সাইজ, রেজুলেশন এবং ক্যাপিবিলিটি ইত্যাদি এবং এসকল বিষয় অনুযায়ী নিজেদের এডজাস্ট করে নেয় যা ইউজারদেরও বেস্ট এক্সপেরিয়েন্স দিয়ে থাকে।

৩৩ LayoutBuilder কী?

LayoutBuilder হলো একটি ফ্লাটার উইজেট যা এর প্যারেন্ট উইজেটের সীমাবদ্ধতাগুলো নিয়ে ইনফরমেশন প্রোভাইড করে। এটি অ্যাভেইলেবল স্পেসের মধ্যে একটি ডাইনামিকালি উইজেট সাব-ট্রি বিল্ড করতেও সহায়তা করে। এটি একটি কলব্যাক ফাংশন নেয় যেটি context এবং constraints parameter রিসিভ করে।

Ostad

০৪ MediaQuery এবং LayoutBuilder এর মধ্যে পার্থক্য কী?

MediaQuery কারেন্ট অ্যাপের স্ক্রিন সাইজের একটি হায়ার লেভেল ভিউ প্রদান করে এবং ডিভাইস এবং এর লেআউট পছন্দ সম্পর্কে আরও ডিটেইল ইনফরমেশন দিতে পারে।

LayoutBuilder উইজেট ফ্লাটারে একটি উইজেট ট্রি তৈরি করতে সাহায্য করে যা মূল উইজেটের সাইজের উপর নির্ভর করতে পারে।

সহজ কথায়, MediaQuery ডিভাইসের একচ্যুয়াল সাইজ দেয়, যেখানে LayoutBuilder এনক্লোজিং প্যারেন্ট উইজেটের সাইজ ঠিক করে।

৩৪ Double.infinity বনাম MediaQuery এর পার্থক্য কী?

Double.infinity এবং MediaQuery উভয়ই ফ্লাটারের কন্সেপ্ট যা একটি উইজেটের লেআউটে সাইজ বা ডাইমেনশন নির্ধারণ করতে ব্যবহার করা যেতে পারে। কিন্তু এদের বিহেভিয়ারের ক্ষেত্রে কিছু পার্থক্য রয়েছে।

Feature	Double.infinity	MediaQuery
Meaning	Double.infinity এর উইজেটটি প্যারেন্ট উইজেটটি যত বড় এলাও করে এর সমান বড় হতে পারে।	MediaQuery এর উইজেটটি স্ক্রিন যত বড়, এর সমান হতে পারে। এখানে প্যারেন্ট উইজেটের সাইজ ম্যাটার করেনা।
Use cases	যখন মূল উদ্দেশ্য হলো প্যারেন্ট উইজেটের এর কন্টেইনার ফিল করা।	যখন মূল উদ্দেশ্য হলো স্ক্রিন সাইজ ফিল করা।
Performance	প্যারেন্ট উইজেট অতিরিক্ত বড় হলে অকার্যকর হতে পারে।	Double.infinity থেকে বেশি কাজ করার ক্ষমতা রাখে।
Best practices	যদি উইজেটটি স্ক্রিন এর সমান বড় প্রয়োজন না হয় তাহল double.infinity ব্যবহার করা যেতে পারে।	যদি উইজেটটি এক্সাক্টলি স্ক্রিন এর সমান বড় প্রয়োজন হয় তাহলে অবশ্যই MediaQuery ব্যবহার করতে হবে।

৩৬ AspectRatio এর ব্যবহার কী?

Flutter-এ AspectRatio উইজেট একটি উইজেটের মধ্যে দৈর্ঘ্য ও প্রস্থের একটি নির্দিষ্ট অনুপাত বজায় রাখতে ব্যবহৃত হয়। এটি রেসপন্সিভ ডিজাইন ক্রিয়েট করার ক্ষেত্রে ইউজফুল যা আলাদা আলাদা স্ক্রিন সাইজে ভালো দেখায়। AspectRatio এর কিছু ব্যবহার হলো-

১। ইমেজ বা ভিডিও এর ক্ষেত্রে একটি স্পেসিফিক অ্যাসপেক্ট রেশিও বজায় রাখা।

২। এমন উইজেট ক্রিয়েট করা যা সবসময় স্পেসিফিক অ্যাসপেক্ট রেশিও এর মধ্যে থেকে প্যারেন্ট উইজেটের কন্টেইনার ফিল করবে।

৩। এমন উইজেট ক্রিয়েট করা যেগুলো স্ক্রিন সাইজ চেঞ্জের ক্ষেত্রে রেসপন্সিভ।

৩৭ Dynamic, var এবং final কী?

Dynamic: ডাইনামিক হচ্ছে সবচেয়ে কমন টাইপের ভ্যারিয়েবল। ডায়নামিক হিসেবে ডিক্লেয়ার করা একটি ভ্যারিয়েবল যেকোনো টাইপের ভ্যালু হোল্ড করতে পারে এবং রানটাইমে এর ধরন পরিবর্তিত হতে পারে। এটি ডাইনামিককে ফ্লেক্সিবল টাইপ করে তোলে কিন্তু কম্পাইলের সময় ভেরিয়েবলের ধরন জানা না থাকলে এটি এরর এর কারণ হতে পারে।

Var: Var হলো একটি টাইপ ইনফারেন্স কিওয়ার্ড। একটি var হিসেবে ডিক্লেয়ার করা ভ্যারিয়েবলকে অটোমেটিক্যালি এই টাইপের ভ্যালু অ্যাসাইন করা হয়। এটি var-কে ভেরিয়েবল ডিক্লেয়ার করার একটি সুবিধাজনক উপায় করে তোলে, তবে ভেরিয়েবলের জন্য যে মান নির্ধারণ করা হচ্ছে তার ধরণ সম্পর্কে সচেতন হওয়া গুরুত্বপূর্ণ।

Final: Final হলো একটি কিওয়ার্ড যা একটি কন্সট্যান্ট ভ্যারিয়েবল ডিক্লেয়ার করতে ব্যবহৃত হয়। একটি ফাইনাল হিসেবে ডিক্লেয়ার করা ভ্যারিয়েবলকে কেবলমাত্র একবার ভ্যালু অ্যাসাইন করা যায় যা পরিবর্তন করা সম্ভব না।

০৮

Final vs const vs static এর মধ্যে পার্থক্য কী?

Final	ফাইনাল ভ্যারিয়েবলে শুধুমাত্র একবার একটি ভ্যালু ডিক্লেয়ার করা যেতে পারে যা পরবর্তীতে পরিবর্তন করা যায় না।
Const	Const ভ্যারিয়েবলগুলো ফাইনাল এর মতই তবে এগুলি কম্পাইলের সময় পরিচিত কন্সট্যান্টগুলি ডিক্লেয়ার করতেও ব্যবহার করা যেতে পারে। এর মানে হলো যে কনস্ট ভেরিয়েবলগুলি কম্পাইলের সময় শুরু করা হয় এবং রানটাইমে তাদের মান পরিবর্তন করা যায় না।
Static	স্ট্যাটিক ভেরিয়েবল হলো এমন ভেরিয়েবল যা একটি ক্লাসের সমস্ত instance গুলো দ্বারা শেয়ার করা হয়। এর মানে হলো যে আপনি যদি একটি ক্লাসের একটি ইন্সট্যান্সে একটি স্ট্যাটিক ভেরিয়েবলের মান পরিবর্তন করেন, তবে স্ট্যাটিক ভেরিয়েবলের মানটি ক্লাসের অন্য সব ক্ষেত্রেও পরিবর্তিত হবে।

৩৯ Getter এবং setter পদ্ধতি কী?

গেটার এবং সেটার মেথড হলো ক্লাস মেথড যা ক্লাস ফিল্ডের ডেটা ম্যানিপুলেট করতে ব্যবহৃত হয়। গেটার ক্লাস ফিল্ডের ডেটা পড়তে বা পেতে ব্যবহার করা হয় যেখানে সেটার ব্যবহার করা হয় ক্লাস ফিল্ডের ডেটা কিছু ভেরিয়েবলে সেট করতে।

Ostad

৪০ Factory constructors কী?

ডার্টে, ফ্যাক্টরি কনস্ট্রাক্টর হলো একটি বিশেষ ধরনের কনস্ট্রাক্টর যা ক্লাসের কোনো কনস্ট্রাক্টর না থাকলেও ক্লাসের একটি instance তৈরি করতে ব্যবহার করা যেতে পারে। ফ্যাক্টরি কনস্ট্রাক্টরগুলি প্রায়শই একটি ক্যাশে থেকে একটি ক্লাসের instance তৈরি করতে বা একটি সাবক্লাসের instance তৈরি করতে ব্যবহৃত হয়।

Ostad

৪১

State কী?

প্রোগ্রামিং-এ স্টেট বলতে সেই ডেটা বোঝায় যা কোনো বস্তু বা সিস্টেমের অবস্থা নির্ধারণ করে। এটি একটি ভেরিয়েবলের বর্তমান মান থেকে একটি স্ক্রিনে একটি বস্তুর অবস্থান পর্যন্ত যেকোনো কিছু হতে পারে। স্টেট গুরুত্বপূর্ণ কারণ, এটি প্রোগ্রামগুলিকে ওয়ার্ল্ডের চেঞ্জগুলোকে ট্র্যাক করতে এবং সেই অনুযায়ী রেসপন্স করতে এলাও করে।

স্টেটের কিছু এক্সাম্পল হলো-

১। একটি ভেরিয়েবলের কারেন্ট ভ্যালু

২। স্ক্রিনে একটি বস্তুর অবস্থান, যেমন একটি খেলার অবস্থা (স্কোর,রিমেইনিং লাইফের সংখ্যা)

৪২

Ephemeral state বনাম App state এর মধ্যে পার্থক্য কী?

Ephemeral State: Ephemeral State লোকাল স্টেট বা কম্পোনেন্ট স্টেট হিসেবেও পরিচিত। এটি টেম্পোরারি বা লোকালাইজড ডেটাকে বোঝায় যা সাধারণত একটি নির্দিষ্ট কম্পোনেন্ট বা ফাংশনের মধ্যে সীমাবদ্ধ থাকে। Ephemeral State সাধারণত একটি কম্পোনেন্টের মধ্যে UI- সম্পর্কিত ইন্টারেকশন এবং লোকাল ক্যালকুলেশনের জন্য ব্যবহৃত হয়।

App state: অ্যাপ স্টেট, যা গ্লোবাল স্টেট বা অ্যাপ্লিকেশন স্টেট নামেও পরিচিত, একটি অ্যাপ্লিকেশন বা সিস্টেমের অভার অল অবস্থা রিপ্রেজেন্ট করে। এটিতে যে ডেটা বা ইনফরমেশন রয়েছে তা অ্যাপ্লিকেশনের মধ্যে একাধিক কম্পোনেন্ট বা মডুলের মধ্যে শেয়ার করতে হয়। অ্যাপ স্টেট সাধারণত একটি সেন্ট্রালাইজড লোকেশনে স্টোর থাকে। যেমন- global object, store, বা database এবং এটি এটি অ্যাপ্লিকেশনের লাইফটাইম জুড়ে উপস্থিত থাকে।

৪৩

Dependencies বনাম dev_dwpendencies প্যাকেজের মধ্যে পার্থক্য কী?

Dependencies: সহজ কথায়, চালু অবস্থায় আপনার অ্যাপ্লিকেশনটি সঠিকভাবে কাজ করার জন্য প্রয়োজনীয় উপাদান হিসেবে Dependencies কন্সিডার করা যেতে পারে। এই প্যাকেজগুলি প্রয়োজনীয় টুলস এবং ফাংশনালিটি প্রোভাইড করে যার উপর আপনার অ্যাপ্লিকেশনটি রিয়াল ওয়ার্ল্ড কাজগুলো করার জন্য নির্ভর করে। এগুলি আপনার অ্যাপ্লিকেশনের কোর কম্পোনেন্টের মতো যা সফলভাবে অ্যাপ্লিকেশনটি রান করানোর জন্য প্রয়োজনীয়।

devDependencies: অন্যদিকে, devDependencies হলো অতিরিক্ত টুলস এবং হেল্পারগুলির মতো যা আপনি ডেভেলপমেন্ট প্রক্রিয়া চলাকালীন ব্যবহার করেন। কিন্তু যখন আপনার অ্যাপ্লিকেশানটি চালু করা হয় বা এন্ড-ইউজারদের দ্বারা ব্যবহার করা হয় তখন তাদের প্রয়োজন হয় না। এরা আপনাকে টেস্টিং, লিন্টিং, কোড ট্রান্সপিলিং বা বান্ডলিং এর মতো কাজে সহায়তা করে। DevDependencies হলো ইউজফুল টুলের মতো যা আপনাকে স্ক্রিনের পিছনে আপনার অ্যাপ্লিকেশন তৈরি ও রিফাইন করতে সাহায্য করে।

৪৪ Flutter কী প্রযুক্তি দ্বারা নির্মিত?

ফ্লাটার ক্রস-প্ল্যাটফর্ম মোবাইল অ্যাপ্লিকেশন তৈরির জন্য একটি জনপ্রিয় ফ্রেমওয়ার্ক। কিন্তু এই ফ্লাটার তৈরিতেও একাধিক টেকনোলজির ব্যবহার করা হয়েছে। যেমনঃ Dart, Skia, C++, Platform-specific APIs (Java/Kotlin for Android, Objective-C/Swift for iOS), SkiaSharp, Dart UI

৪৫

Deactivate এবং dispose এর মধ্যে পার্থক্য কী?

Deactivate: Deactivate বলতে সাধারণত এমন একটি অবস্থাকে বোঝায় যেখানে একটি অবজেক্ট বা কম্পোনেন্ট সাময়িকভাবে ইনএক্টিভ বা পজ করা হয়। ডিয়েক্টিভেট করা হলে ঐ অবজেক্ট বা কম্পোনেন্ট আবার প্রয়োজন পরলেই এক্টিভ করা যায়। এক্ষেত্রে কোনো ডেটা হারানোর ভয় থাকেনা কেননা এক্টিভ করার পর তা আবার আগের অবস্থায় ফেরত পাওয়া পসিবল হয়।

Dispose: Dispose বলতে বোঝায় কোনো অবজেক্টের ক্লিন আপ প্রসেস। যখন একটি অবজেক্ট আর দরকার পড়ে না তখন এটি পারমানেন্টলি বন্ধ বা ক্লিন আপ করার জন্য ডিসপোজ করা হয়। এই ডেটাগুলো আর ফেরত পাওয়া সম্ভব হয়না। এই কারণে ডিসপোজ করার পর মেমোরিতে স্পেস সেইভ হয়।

৪৬

MVC, MVP এবং MVVM Architecture Pattern এর মধ্যে পার্থক্য কী?

MVC (Model-View-Controller):

Model: অ্যাপ্লিকেশনটির ডেটা এবং বিজনেস লজিক রিপ্রেজেন্ট করে।

View: প্রেজেন্টেশন লেআউট হ্যান্ডেল করে, ইউজারের কাছে ইউজারের ইন্টারফেস ডিসপ্লে করে এবং ইউজারের ইনপুট ক্যাপচার করে।

Controller: মডেল এবং ভিউয়ের মধ্যে মধ্যস্থতাকারী হিসেবে কাজ করে, ইউজারের ইন্টারেকশন হ্যান্ডেল করে, মডেল আপডেট করে এবং মডেলের পরিবর্তনের উপর ভিত্তি করে ভিউ আপডেট করে।

MVP (Model-View-Presenter):

Model: অ্যাপ্লিকেশনটির ডেটা এবং বিজনেস লজিক রিপ্রেজেন্ট করে।

View: প্রেজেন্টেশন লেআউট হ্যান্ডেল করে, ইউজারের কাছে ইউজারের ইন্টারফেস ডিসপ্লে করে এবং ইউজারের ইনপুট ক্যাপচার করে।

Presenter: মডেল এবং ভিউয়ের মধ্যে মধ্যস্থতাকারী হিসেবে কাজ করে। এটি মডেল থেকে ডেটা পুনরুদ্ধার করে, ভিউ থেকে ইউজারের অ্যাকশন প্রসেস করে এবং মডেলের অবস্থার উপর ভিত্তি করে ভিউ আপডেট করে।

MVVM (Model-View-ViewModel):

Model: অ্যাপ্লিকেশনটির ডেটা এবং বিজনেস লজিক রিপ্রেজেন্ট করে।

View: প্রেজেন্টেশন লেআউট হ্যান্ডেল করে, ইউজারের কাছে ইউজারের ইন্টারফেস ডিসপ্লে করে এবং ইউজারের ইনপুট ক্যাপচার করে।

ViewModel: মডেল এবং ভিউ এর মধ্যে মধ্যস্থতাকারী হিসেবে কাজ করে। এটি ভিউতে ডেটা এবং কমান্ডগুলি প্রকাশ করার পাশাপাশি ডেটা ম্যানিপুলেশন বা ট্রান্সফরমেশনও পারফর্ম করে।

সহজভাবে, MVC, MVP, এবং MVVM-এর মধ্যে মূল পার্থক্য হলো তারা যেভাবে মডেল, ভিউ এবং তাদের মধ্যে মধ্যস্থতাকারী কম্পোনেন্টদের মধ্যে রেসপন্সিবিলিটি এবং ইন্টারেকশনগুলো হ্যান্ডেল করে, এসবের ধরনের মধ্যে।

Ostad

৪৭ Debug mode এবং profile mode এর মধ্যে পার্থক্য কী?

সহজ কথায়, যখন আপনি হট রিলোড ব্যবহার করতে চান তখন আমরা ডেভেলপমেন্টের সময় ডিবাগ মোড ব্যবহার করি। এবং আপনি যখন পারফরমেন্স এনালাইজ করতে চান তখন আমরা প্রোফাইল মোড ব্যবহার করি। ডিবাগের মধ্যে কম্পাইল করা ফাইলগুলিতে ডিবাগিং ইনফরমেশনগুলো থাকে। যেখানে রিলিজে সাধারণত অপ্টিমাইজেশান এক্টিভ থাকে।

সফটওয়্যার ডেভেলপমেন্টের ক্ষেত্রে দুটি টুলই আলাদাভাবে গুরুত্বপূর্ণ পারপাস সার্ভ করে। ডিবাগ মোড একুরেসি নিশ্চিত করতে সাহায্য করে, যেখানে প্রোফাইল মোড পারফরমেন্স বাড়াতে সাহায্য করে।

৪৮

“??” এবং “?” অপারেটরের মধ্যে পার্থক্য কী?

?? (null-coalescing operator): যখন একটি নির্দিষ্ট মান নাল বা আনডিফাইনড হয় তখন ?? অপারেটর একটি ফলব্যাক বা ডিফল্ট মান প্রদান করে। অর্থাৎ ঐ মানের বদলে অন্য একটি সিমিলার ভ্যালু ইউজ করতে সাজেস্ট করে।

? (optional chaining operator): ? অপারেটর, অপশনাল চেইনিং অপারেটর হিসেবেও পরিচিত। এটি প্রপার্টি চেইনের কোনো অংশ null বা undefined থাকলে এরর না করেই আপনাকে সেইফ ভাবে প্রপার্টি বা মেথড অ্যাক্সেস করতে দেয়।

৪৯ ডার্টের required বনাম optional বনাম named parameters এর মধ্যে পার্থক্য ব্যাখ্যা করুন।

Required Parameters:

1. Required Parameters হলো সবথেকে বেইসিক টাইপের প্যারামিটার এবং যখন একটি নির্দিষ্ট ভ্যালু এক্সপেক্ট করে, তখন এটি ব্যবহার করা হয়।

2. কোনো স্কয়ার ব্র্যাকেট ছাড়াই এদের ডিক্লেয়ার করা হয়।

Optional Parameters:

1. অপশনাল প্যারামিটার ব্যবহার করা হয় যখন আপনি একটি ডিফল্ট ভ্যালু প্রদান করতে চান বা কলারদের নির্দিষ্ট আর্গুমেন্ট বাদ দিতে চান।

2. এগুলো স্কয়ার ব্র্যাকেট এর মধ্যে ডিক্লেয়ার করা হয়।

Named Parameters:

1. Named optional parameters গুলোকে তাদের নাম দ্বারা স্পেসিফাই করা হয়। এগুলো আপনাকে যেকোনো অর্ডারে আর্গুমেন্ট পাস করতে দেয়। Named parameters এর কাজও Named optional parameters এর সিমিলার।

2. আপনি যখন ক্ল্যারিটি প্রদান করতে চান এবং Name সহ নির্দিষ্ট আর্গুমেন্ট পাস করার জন্য কলারদের এনফোর্স করতে চান তখন Named parameters ইউজফুল।

৫০ Streams এর বিভিন্ন প্রকার ব্যাখ্যা করুন।

ডার্টে, বিভিন্ন ধরনের স্ট্রিম রয়েছে যেগুলির সাথে আপনি কাজ করতে পারেন। যেমন- Single-Subscription Stream, Broadcast Stream, Asynchronous Stream, Synchronous Stream, Single-Subscription Broadcast Stream। এদের প্রতিটি একটি নির্দিষ্ট পারপাস সার্ভ করে। এদের মধ্যে Single-Subscription Stream এবং Broadcast Stream মোস্ট কমন।

Single-Subscription Stream:

১। একটি Single-Subscription Stream একক সময়ে কেবলমাত্র একটি লিসেনারকে সাবস্ক্রাইবের জন্য এলাও করে।

২। একবার একজন লিসেনার সাবস্ক্রাইব করলে সে স্ট্রিমের ইভেন্টগুলো রিসিভ করে।

৩। সবগুলো ইভেন্ট রিসিভ করা হয়ে গেলে বা লিসেনার সাবস্ক্রিপশন ক্যান্সেল করে দিলে স্ট্রিমটি বন্ধ বলে কন্সিডার করা হয়।

৪। ডার্ট-এ Single-Subscription Stream এর মধ্যে রয়েছে StreamController, File, এবং HttpClient।

Broadcast Stream:

১। একটি Broadcast Stream একক সময়ে একাধিক লিসেনারকে সাবস্ক্রাইবের জন্য এলাও করে।

২। প্রতিটি লিসেনার ইভেন্টগুলির একটি আলাদা কপি রিসিভ করে।

৩। যখন আপনার স্ট্রিমটি একাধিক মানুষের সাথে শেয়ার করতে হবে, এমন সিচুয়েশনগুলির জন্য Broadcast Stream উপযোগী।

৪। ডার্ট-এ Broadcast Stream এর মধ্যে রয়েছে StreamController, StreamTransformer।

৫১ ফ্লাটারে keys কী এবং এটি কখন ব্যবহার করতে হয়?

ফ্লাটারে, key গুলি উইজেটগুলির মধ্যে identify এবং differentiate করতে ব্যবহৃত হয়। এগুলি এমন অবজেক্ট যা ফ্লাটারের উইজেট ফ্রেমওয়ার্ককে বুঝতে সাহায্য করে কীভাবে উইজেট রিবিল্ড জুড়ে উইজেটগুলি একটি অন্যটির সাথে যুক্ত। key গুলি উইজেট পরিচালনায় একটি গুরুত্বপূর্ণ ভূমিকা পালন করে এবং বিভিন্ন সিনারিওতে ইউজফুল হতে পারে।

GlobalKey: যখন আপনাকে অ্যাপের একটি ভিন্ন এরিয়া থেকে একটি উইজেট identify করতে হবে, আপনি একটি GlobalKey ব্যবহার করতে পারেন।

UniqueKey: যখন আপনাকে উইজেটের একটি লিস্টের মধ্যে কেবল একটি উইজেট শনাক্ত করতে হবে তখন UniqueKey ইউজফুল।

ValueKey: একটি ValueKey একটি ভ্যালুর উপর ভিত্তি করে তৈরি করা হয়, যেমন- একটি স্ট্রিং বা একটি পূর্ণসংখ্যা। যখন আপনাকে একটি উইজেটকে তার ভ্যালুর উপর ভিত্তি করে একটি উইজেটের লিস্টের মধ্যে থেকে identify করতে হবে, তখন ValueKey ইউজফুল।

৫২

ফ্লাটারের async, await ব্যাখ্যা করুন।

ফ্লাটারে, অ্যাসিঙ্ক্রোনাস অপারেশনগুলো হ্যান্ডেল করতে async এবং await ব্যবহার করা হয়। একটি ফাংশনকে async হিসেবে মার্ক করে, এক্সেকিউশন পজ করতে আপনি await ইউজ করতে পারেন। এটি আপনাকে আরও সিঙ্ক্রোনাস এবং রিডেবল ম্যানারে অ্যাসিঙ্ক্রোনাস কোড তৈরি করতে এলাও করে। await একটি ফিউচারের জন্য অপেক্ষা করে এবং রেজাল্টটি কমপ্লিট করে। এটি এপিআই, ফাইল অপারেশন এবং টাইম-কনজ্যুমিং কাজগুলি করা সহজ করার পাশাপাশি কোড লেখা এবং বোঝাও সহজ করে তোলে। async এবং await একসাথে ফ্লাটারে অ্যাসিঙ্ক্রোনাস প্রোগ্রামিং হ্যান্ডেল করার একটি হেল্পফুল উপায় প্রদান করে।

৪৩ Dart AOT কীভাবে কাজ করে?

Dart AOT (Ahead-of-Time) হলো একটি প্রসেস যা অ্যাপ্লিকেশনটি এক্সেকিউট করার আগে ডার্ট কোডকে মেশিন কোডে রূপান্তর করে। এর মানে হলো, কোডটি রানটাইমে interpret করা হয় না এবং এটির জন্য পারফরম্যান্স আরও বেটার হয়। AOT compilation সাধারণত প্রোডাকশন অ্যাপ্লিকেশনের জন্য ব্যবহার করা হয়ে থাকে।

৫৪

ফ্লাটারে Future এবং Stream এর মধ্যে সাদৃশ্য এবং পার্থক্যগুলো কী?

সাদৃশ্যঃ

১। ফিউচার এবং স্ট্রিম উভয়ই ডার্টে অ্যাসিঙ্ক্রোনাস কনস্ট্রাক্ট।

২। এদের দুটিই ভবিষ্যতে ঘটে যাওয়া ঘটনাগুলি ম্যানেজ করতে ব্যবহৃত হয়।

৩। এদের দুটিকেই একটি ইভেন্ট শুনতে এবং এতে রিয়েক্ট করার জন্য ব্যবহার করা যেতে পারে।

পার্থক্যঃ

১। ফিউচার একটি ওয়ান-টাইম ইভেন্ট। একবার হওয়ার পরেই এটি কমপ্লিট হয়ে যাবে।

২। স্ট্রিম হচ্ছে অনেকগুলো এভেন্টের সিকুয়েন্স। এটি যেকোনো সময় এবং বারবার ইভেন্টগুলো এমিট করার ক্ষমতা রাখে।

৩। একটি ফিউচার কেবলমাত্র একবারই শুনা যাবে। একবার শেষ হয়ে গেলে এটি আর শুনা যায় না।

৪। একটি স্ট্রিম একাধিকবার শোনা যাবে। এমনকি ইভেন্টটি এমিট হওয়ার সময় যদি শুনতে সমস্যা হয়, এটি পরেও যেকোনো সময় শুনা যাবে।

৫৫

ডার্টে async এবং async*এর মধ্যে পার্থক্যগুলো কী?

ডার্টে অ্যাসিঙ্ক এবং অ্যাসিঙ্ক* কি-ওয়ার্ডগুলি একটি ফাংশন অ্যাসিঙ্ক্রোনাস হিসেবে চিহ্নিত করতে ব্যবহৃত হয়। এদের মধ্যের মূল পার্থক্যটি হলোঃ

async ফাংশন একটি ফিউচার অবজেক্ট রিটার্ন করে। এর মানে হলো, ইমিডিয়েটলি না হলেও ফাংশনটি কমপ্লিট হওয়ার পর ভ্যালু রিটার্ন করবে। একটি async ফাংশনের রেজাল্ট অ্যাভেইলেবল হওয়ার জন্য অপেক্ষা করতে await keyword-টি ব্যবহার করা যেতে পারে।

এদিকে, async* ফাংশন একটি স্ট্রিম অবজেক্ট রিটার্ন করে। এর মানে হলো, ফাংশনটি রিপিটেডলি ভ্যালু এমিট করতে থাকবে। একটি async* ফাংশন থেকে একটি মান এমিট করতে yield কি-ওয়ার্ড ব্যবহার করা হয়।

৫৬

প্রথমবার ফ্লাটার অ্যাপ তৈরি করতে এতো সময় লাগে কেনো?

প্রথমবার ফ্লাটার অ্যাপ তৈরি করতে সময় লাগে কারণ প্রথমবার অনেক কাজ করতে হয় যা পরবর্তী বিল্ডে আর প্রয়োজন হয়না। যেমন-

মেশিন কোডে ডার্ট কোড কম্পাইল করা, অ্যাপের জন্য এসেট তৈরি করা (ইমেজ,ফন্ট), অ্যাপটিকে একটি ফরম্যাটে প্যাকেজ করা যা

একটি ডিভাইসে রান করতে পারে ইত্যাদি।

এই কাজগুলি দীর্ঘ সময় নিতে পারে, বিশেষ করে অ্যাপটি যদি বড় হয় এবং এতে কমপ্লেক্স এসেট ব্যবহার করা হয়। তাই ফার্স্ট বিল্ড

কমপ্লিট হয়ে গেলে পরবর্তী বিল্ডগুলো দ্রুত হয় কেননা এখানে শুধুমাত্র কোডে করা পরিবর্তনগুলি আপডেট করতে হয়।

৫৭ ফ্লাটারে "main()" এবং "runApp()" ফাংশনের মধ্যে পার্থক্য আছে কী? কেমন পার্থক্য?

Ostad

main() function প্রোগ্রাম শুরু করার জন্য রেসপন্সিবল। main() ফাংশন ছাড়া আমরা Flutter-এ কোনো প্রোগ্রাম লিখতে পারি না।

runApp() ফাংশন উইজেট ট্রির রুট হিসেবে স্ক্রিনের সাথে এটাচ করা উইজেটগুলি রিটার্নের জন্য রেসপন্সিবল।

main() function হলো ফ্লাটার অ্যাপ সহ সকল ডার্ট প্রোগ্রামের এন্ট্রি পয়েন্ট। এখানেই একটি অ্যাপ initialize করা এবং ফ্লাটার ফ্রেমওয়ার্ক শুরু করা হয়।

runApp() ফাংশন হলো একটি Flutter ফাংশন যা Flutter অ্যাপ শুরু করতে ব্যবহৃত হয়। এটি ইনপুট হিসেবে একটি উইজেট নেয় এবং স্ক্রিনে এটাচ করে।

main() এবং runApp() এর মধ্যে প্রধান পার্থক্য হল main() একটি জেনেরিক ডার্ট ফাংশন, যেখানে runApp() একটি ফ্লাটার-স্পেসিফিক ফাংশন। main() যেকোনো ডার্ট প্রোগ্রাম শুরু করতে ব্যবহার করা যেতে পারে, যখন runApp() শুধুমাত্র একটি Flutter অ্যাপ শুরু করতে ব্যবহার করা যেতে পারে।

৫৮ আপনার কখন mainAxisAlignment এবং crossAxisAlignment ব্যবহার করা উচিত?

Ostad

Flutter-এ mainAxisAlignment এবং crossAxisAlignment প্রোপার্টিগুলো ব্যবহার করে প্যারেন্ট কন্টেইনারে চাইল্ড এলিমেন্টের পজিশনিং এবং অ্যালাইনমেন্ট কন্ট্রোল করা যায়। এই প্রোপার্টিগুলি ব্যবহার করা হয় পজিশনিং ডিরেকশন দিতে যখন আপনি অনেকগুলি চাইল্ড এলিমেন্টকে প্যারেন্ট কন্টেইনারের মধ্যে কন্ট্রোল করতে চান।

MainAxisAlignment নিয়ন্ত্রণ করে কীভাবে একটি রো বা কলাম উইজেটের চিল্ড্রেনরা মেইন এক্সিস বরাবর সারিবদ্ধ হয়। যেখানে, CrossAxisAlignment নিয়ন্ত্রণ করে কীভাবে একটি রো বা কলাম উইজেটের চিলড্রেনরা ক্রস এক্সিস বরাবর সারিবদ্ধ করা হয়।

তাই, একটি রো উইজেটের চিলড্রেনদের সেন্টারে রাখতে mainAxisAlignment বা পুরো স্ক্রিনে stretch করতে crossAxisAlignment ব্যবহার করা যেতে পারে।

৫৯ SizedBox vsContainer এর মধ্যে পার্থক্য কী?

Ostad

SizedBox	Container
১। তুলনামূলক একটি সিম্পল উইজেট এবং এর প্রোপার্টির সংখ্যাও কম।	১। তুলনামূলক একটি কমপ্লেক্স উইজেট এবং এর প্রোপার্টির সংখ্যাও বেশি।
২। শুধুমাত্র দুটি প্রোপার্টি রয়েছে। এগুলো হলো width এবং height।	২। এর প্রোপার্টির মধ্যে রয়েছে- color, border, padding, এবং margin।
৩। এটি একটি lightweight widget	৩। SizedBox এর তুলনায় এটি একটি heavier widget
৪। একটি const হিসেবে ডিক্লেয়ার করা যাবে	৪। একটি const হিসেবে ডিক্লেয়ার সম্ভব নয়
৫। যখন একটি উইজেটে একটি নির্দিষ্ট সাইজ প্রয়োগ করতে হবে তখন ব্যবহার করা হয়।	৫। একটি উইজেটের লুক এবং লেআউটের উপরআরও কন্ট্রোল প্রয়োজন হলে ব্যবহার করা হয়।

৬০ build() মেথড কেনো State-এ, কেনো StatefulWidgets-এ নয়?

build() মেথডটি যে কারণগুলোর জন্য StatefulWidgets এর বদলে State এ রয়েছে, তা হলো-

১। এটি সাবক্লাসিংয়ে আরও ফ্লেক্সিবিলিটি এলাও করে। বিল্ড () মেথড স্টেটফুল উইজেটে থাকলে, স্টেটফুল উইজেটের যেকোনো সাবক্লাস তাদের প্রয়োজন না থাকলেও বিল্ড() মেথড ওভার রাইড করতে বাধ্য হবে। এটি কোডটিকে আরও verbose এবং কম ফ্লেক্সিবল করে তুলবে।

২। এটি বেটার encapsulation এলাও করে। স্টেট ক্লাস উইজেটের স্টেট ম্যানেজ করার জন্য রেসপন্সিবল, এবং বিল্ড () মেথড হলো একমাত্র উপায় যা স্টেট ক্লাস উইজেট ট্রির সাথে interact করতে পারে। এটি উইজেটের স্টেটকে বাকি কোড থেকে আলাদা রাখা সহজ করে তোলে।

৩। এটি টেস্ট করা সহজ করে তোলে। বিল্ড () মেথডটি একটি পিউর ফাংশন, যার মানে এর কোনো সাইড ইফেক্টস নেই। এটি বাকি কোড সম্পর্কে চিন্তা না করেই বিল্ড () মেথডটিকে আলাদাভাবে টেস্ট করা সহজ করে তোলে।

৬১ কেনো আমাদের ফ্লাটারে mixins প্রয়োজন?

মিক্সিন হলো ফ্লাটারে কোড রিইউজ করার একটি উপায়। আমাদের ফ্লাটারে মিক্সিন প্রয়োজন হওয়ার কিছু কারণ হলো-

১। mixins শেয়ার করা কোডের প্রতিটি অংশের জন্য একটি নতুন ক্লাস তৈরি না করেই একাধিক ক্লাসে কোড শেয়ার করতে দেয়। এটি কোডকে DRY (Don't Repeat Yourself) রাখতে এবং এটি মেইন্টেইন করতে সাহায্য করে।

২। একাধিক ক্লাসে ম্যাসেজ লগ করতে একটি মিক্সিন ব্যবহার করা যেতে পারে। যেমন- একটি মিক্সিন তৈরি করা যেতে পারে যা ম্যাসেজ লগ করার জন্য একটি মেথড নির্ধারণ করে। এবং পরে সেই মিক্সিনটিকে যে কোনও ক্লাসের সাথে মিক্স করা যেতে পারে যেখানে ম্যাসেজগুলি লগ করা প্রয়োজন।

৩। একটি অ্যাপ্লিকেশন internationalize করতে একটি mixin ব্যবহার করা যেতে পারে। যেমন- একটি মিক্সিন তৈরি করা যেতে পারে যা স্ট্রিং ট্রান্সলেশন করার জন্য একটি মেথড নির্ধারণ করে। এরপর স্ট্রিং ট্রান্সলেট করতে হবে এমন কোনো ক্লাসের সাথে মিক্সিন মিক্স করা হলে এ অ্যাপ্লিকেশনটি internationalized হয়ে যাবে।

৬২ আমরা কেনো Flutter এ একটি Ticker ব্যবহার করি?

ফ্লাটারে Ticker হলো অ্যানিমেশনের একটি রিফ্রেশ রেট। এটি এমন একটি ক্লাস যা রেগুলার interval এ একটি সিগন্যাল পাঠায়, অর্থাৎ প্রতি সেকেন্ডে প্রায় 60 বার। আমাদের ঘড়ি দিয়ে বুঝতে পারি, যখন এটি প্রত্যেক বিরতির পর এটি টিক করে। টিকার একটি কলব্যাক মেথড প্রোভাইড করে যা প্রতিবার সিগন্যাল পাঠানো হলে কল করা হয়। এই কলব্যাক মেথড UI আপডেট করতে বা অন্যান্য কাজ সম্পাদন করতে ব্যবহার করা যেতে পারে। টিকারটি অটোমেটিক্যালি সিঙ্ক্রোনাইজ হয়, যার অর্থ হলো দুটি টিকার ভিন্ন সময়ে শুরু হলেও, তারা সবসময় সিঙ্কে থাকবে। এটি অ্যানিমেশনগুলির জন্য গুরুত্বপূর্ণ কারণ এটি নিশ্চিত করে যে অ্যানিমেশনগুলির প্রোগ্রেস smooth হবে।

৬৩ ডার্টে কখন mixins এবং কখন interfaces ব্যবহার করতে হয়?

মিক্সিন এবং ইন্টারফেস উভয়ই ডার্টে কোড রিইউজ করার উপায়। কিন্তু এদের উদ্দেশ্য ভিন্ন হওয়ায় এদের আলাদা আলাদা সিচুয়েশনে ব্যবহার করা হয়।

Mixins ক্লাসগুলোর মধ্যে বিহেভিয়ার শেয়ার করতে ব্যবহৃত হয়। মিক্সিন ক্লাস এক্সটেন্ড না করেই এতে ফানশনালিটি এড করার একটি উপায়। ইভেন্ট হ্যান্ডলিং, লগিং বা ইন্টারন্যাশনালাইজেশনের মতো বিষয়গুলি শেয়ার করার জন্য এটি ইউজফুল হতে পারে।

ইন্টারফেসগুলি একটি কন্ট্রাক্ট ডিফাইন করতে ব্যবহৃত হয় যা অন্যান্য ক্লাসগুলো ইমপ্লিমেন্ট করতে পারে। একটি ক্লাসে যে বিহেভিয়ারগুলো অবশ্যই থাকতে হবে, সেই বিহেভিয়ারগুলো স্পেসিফাই করার একটি উপায় হচ্ছে ইন্টারফেস। আলাদা আলাদা ক্লাসগুলো একটি আরেকটির সাথে ঠিকভাবে ইন্টারেক্ট করতে পারার বিষয়টি নিশ্চিত করতে ইন্টারফেস ব্যবহার করা যেতে পারে।

তাই, একটির অপরটির সাথে সরাসরি সম্পর্ক নেই এমন ক্লাসের মধ্যে বিহেভিয়ার শেয়ার করতে চাইলে মিক্সিন ব্যবহার করা উচিত। কিন্তু, যখন উদ্দেশ্য হলো একটি কন্ট্রাক্ট ডিফাইন করা যাতে অন্যান্য ক্লাসগুলো ইমপ্লিমেন্ট করতে পারে, এসকল ক্ষেত্রে ইন্টারফেস ব্যবহার করা উচিত।

৬৪

ফ্লাটারে Container class কী?

কন্টেইনার ক্লাস padding, borders, heighContainert, width ইত্যাদির মতো প্রোপার্টিজ সহ একটি উইজেট তৈরি করার এবিলিটি প্রোভাইড করে।

Ostad

৬৫ একটি Container এর কী একটির চেয়ে বেশি child থাকতে পারে?

একটি কন্টেইনার হলো Single-child লেআউট উইজেট। কিন্তু Multi-child লেআউট উইজেট এটির চাইল্ড হিসেবে ব্যবহার করে একটি কন্টেইনারের একাধিক চিলড্রেন পাওয়া সম্ভব।

৬৬

ডার্টে extension পদ্ধতিগুলো কী কী? কেনো এটা ব্যবহার করা হয়?

ডার্ট 2.7-এ ইন্ট্রোডিউস করা এক্সটেনশন মেথডগুলি এক্সিস্টিং লাইব্রেরিতে ফাংশনালিটি এড করার একটি উপায়। এক্সটেনশন মেথড সম্পর্কে না জেনেও এটি ব্যবহার করা যেতে পারে। যেমন- IDE তে একটি কোড completion করার পর এটি রেগুলার মেথডের পাশাপাশি এক্সটেনশন মেথডও সাজেস্ট করে।

এটি ব্যবহার করার কিছু সুবিধা হলো-

১। মূল কোড পরিবর্তন না করেই এক্সিস্টিং ক্লাসে ফাংশনালিটি এড করতে এক্সটেনশন মেথড ব্যবহার করা যায়। এটি কোডটিকে আরও রিইউজেবল এবং মেইন্টেইন করা সহজ করে।

২। নতুন সাবক্লাস তৈরি না করেই এক্সিস্টিং ক্লাসের ফাংশনালিটি বাড়ানোর জন্য এক্সটেনশন মেথড ব্যবহার করা যেতে পারে। এটি কোডটিকে আরও এক্সটেনসিবল এবং নতুন রিকোয়্যারমেন্টের সাথে মানিয়ে নেওয়া সহজ করে তোলে।

৩। এক্সটেনশন মেথডগুলি এমনভাবে এক্সিস্টিং ক্লাসগুলিতে ফাংশনালিটি এড করতে ব্যবহার করা যেতে পারে যা পড়া এবং বোঝা সহজ। এটি কোডটিকে আরও রিডেবল এবং হ্যান্ডেল করা সহজ করে তোলে।

Ostad

৬৭

আপনি কতোগুলো উপায়ে ডার্টের parameter-গুলো pass করতে পারবেন?

মেইনলি তিনটি উপায়ে, প্যারামিটার পাস করা যায়: ডিফল্ট, অপশনাল এবং নেইম।

এছাড়াও, ডার্ট ফাংশন পজিশনাল প্যারামিটার, নেইমড প্যারামিটার, অপশনাল প্যারামিটার বা সবগুলির কম্বিনেশন এলাও করে।

Ostad

৬৮ কীভাবে ডার্টে conditionally property বা method access করতে পারবেন?

Null কোনো অব্জেক্ট বা প্রোপার্টি তে এক্সেস গার্ড করার জন্য ?. কন্ডিশনাল প্রোপার্টি এক্সেস অপারেটর হিসেবে ইউজ করা হয়।

৬৯ Spread operator ব্যাখ্যা করুন।

ডার্টে, Spread Operator (...) এবং Null-aware Spread Operator (...?) একটি কালেকশনে একাধিক এলিমেন্ট ইন্সার্ট

করার জন্য ব্যবহার করা হয়। যেমন- লিস্ট, সেট ম্যাপ ইত্যাদি।

Spread operator:

...Data_structureExample:-

```
var a = [0,1,2,3,4];
var b = [6,7,8,9];
var c = [...a,5,...b];

print(c);  // prints: [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
```

Null-aware Spread operator:

...?Data_structure

Example :-

```
List<int> l1 = [1, 2, 3];

List<int> nullList = null;

List<int> result = [...l1, ...?nullList];

print(result); // prints:   [1, 2, 3]
```

Ostad

৭০ ফ্লাটারের Provider Pub/ Library ব্যাখ্যা করুন।

প্রোভাইডার হলো ফ্লাটারের একটি লাইব্রেরি যা আপনাকে আপনার ফ্লাটার অ্যাপের স্টেট ম্যানেজ করতে সাহায্য করে। এটি InheritedWidget এর চারপাশে একটি wrapper যা উইজেটগুলির মধ্যে স্টেট শেয়ার করা সহজ করে তোলে। প্রোভাইডার আরও কিছু ফিচারও প্রোভাইড করে যা স্টেট ম্যানেজ করা আরও সহজ করে তোলে। যেমন- lazy loading এবং dependency injection।

প্রোভাইডার প্যাকেজটি ব্যবহার করে, ফ্লাটার অ্যাপ্লিকেশনগুলিতে efficient এবং granular state ম্যানেজমেন্ট এচিভ করা যেতে পারে। এটি UI লেয়ার থেকে স্টেট ম্যানেজমেন্টের বিষয়গুলো আলাদা করে এবং একটি decoupled এবং scalable architecture প্রমোট করে।

৭১ Solid Principle ব্যাখ্যা করুন।

SOLID principle গুলো হলো design principles এর একটি সেট যেটির লক্ষ্য হলো সফ্টওয়্যার সিস্টেমগুলিকে আরও maintainable, scalable, এবং robust করা। SOLID যে প্রিন্সিপালগুলোর acronym, এগুলো হলো-

Single Responsibility Principle (SRP): একটি ক্লাস চেঞ্জ করার শুধুমাত্র একটি কারণ থাকতে হবে। এই প্রিন্সিপাল মেনে চলার মাধ্যমে কোডটি আরও মেইন্টেইনেবল এবং বাগগুলির জন্য রিস্ক কমানো যেতে পারে।

Open/Closed Principle (OCP): এই প্রিন্সিপাল অনুযায়ী, সফ্টওয়্যার এনটিটি (classes, modules, functions, ইত্যাদি) এক্সটেনশনের জন্য ওপেন থাওলেও মোডিফিকেশনের জন্য ক্লোজড থাকা উচিত। অন্য কথায়,কোডটি এমনভাবে ডিজাইন করা উচিত যাতে এটি চেঞ্জ না করেই এক্সিস্টিং কোড এক্সটেন্ড করে নতুন ফাংশনালিটি এড করতে এলাও করে।

Liskov Substitution Principle (LSP): লিসকভ সাবস্টিটিউশন প্রিন্সিপাল অনুযায়ী, একটি সুপারক্লাসের অবজেক্টগুলো প্রোগ্রামের কারেক্টনেসকে ইফেক্ট না করেই এর সাবক্লাসের অবজেক্টের সাথে রিপ্লেসেবল হওয়া উচিত। সহজভাবে, কোনো আনেক্সপেক্টেড বিহেভিয়ার ছাড়াই সাব-টাইপগুলি এদের বেস টাইপের জন্য substitutable হওয়া উচিত।

Interface Segregation Principle (ISP): ইন্টারফেস সেগ্রিগেশন প্রিন্সিপাল অনুযায়ী, ব্যবহার না করা ইন্টারফেসের উপর নির্ভর করতে ক্লায়েন্টদের বাধ্য করা উচিত নয়। অর্থাৎ, ইন্টারফেসগুলি এমনভাবে ডিজাইন করা উচিত যাতে ক্লায়েন্টদের (ক্লাস বা মডিউল) তাদের ফানশনালিটির সাথে অপ্রাসঙ্গিক মেথডগুলো ইমপ্লিমেন্টের প্রয়োজন না হয়।

Dependency Inversion Principle (DIP): ডিপেনডেন্সি ইনভার্সন প্রিন্সিপাল অনুযায়ী, হাই-লেভেল মডিউল/ক্লাস সরাসরি লো-লেভেল মডিউল/ক্লাসের উপর ডিপেন্ড করা উচিত নয়। এর বদলে এদের abstractions এর উপর ডিপেন্ড করতে হয়।

এই SOLID principle গুলো অ্যাপ্লাই করার মাধ্যমে আপনি এমন কোড ক্রিয়েট করতে পারবেন যা বোঝা, মেইন্টেইন করা এবং এক্সটেন্ড করা সহজ।

৭২ ফ্লাটারের singleton class ব্যাখ্যা করুন।

ফ্লাটারে, একটি singleton class হলো এমন একটি ক্লাস যা শুধুমাত্র নিজের একটি single instance তৈরি করতে দেয় এবং সেই instance-এ একটি গ্লোবাল পয়েন্ট অব এক্সেস প্রদান করে। singleton class গুলো একটি অ্যাপ্লিকেশনের মাল্টিপল পার্টস দ্বারা শেয়ার করা গ্লোবাল স্টেট বা রিসোর্সগুলোকে রিপ্রেজেন্ট করতে ব্যবহৃত হয়।

ফ্লাটারে একটি সিঙ্গেলটন ক্লাস তৈরি করতে একটি সিঙ্গেলটন প্যাটার্ন ব্যবহার করা যেতে পারে যার মধ্যে একটি প্রাইভেট কনস্ট্রাক্টর এবং আরেকটি স্ট্যাটিক পদ্ধতি তৈরি করে, যা ক্লাসের single instance প্রদান করে।

ফ্লাটারে একটি সিঙ্গলটন ক্লাসের একটি উদাহরণ-

```
class MySingletonClass {

 static final MySingletonClass _instance = MySingletonClass._internal();

 factory MySingletonClass() {

  return _instance;

 }  MySingletonClass._internal();  // Other methods and properties go here

}
```

সিঙ্গেলটন ক্লাস ব্যবহার করতে, আপনি যে স্ট্যাটিক ফ্যাক্টরি মেথড কল করবেন:

```
MySingletonClass().doSomething();
```

এটি মনে রাখা গুরুত্বপূর্ণ যে Flutter-এ, singletons এর উপর নির্ভর না করে, শেয়ার্ড স্টেট এবং রিসোর্স ম্যানেজ করার জন্য ডিপেন্ডেন্সি

ইনজেকশন ব্যবহার করা সাধারণত ভালো। কারণ এটি কোডের testability এবং maintainability ইমপ্রুভ করতে সাহায্য করে।

৭৩ ফ্লাটার প্রোভাইডারে একটি একক উইজেটের জন্য multiple Consumers কীভাবে ব্যবহার করতে হয়?

Flutter's Provider Package এ, আপনি স্টেটের বিভিন্ন পার্ট থেকে একাধিক consumers উইজেট ব্যবহার করতে এবং সেই অনুযায়ী আপনার UI আপডেট করতে পারেন। আপনি কীভাবে একটি সিঙ্গেল উইজেটের জন্য একাধিক Consumer widgets ব্যবহার করবেন এর কিছু উদাহরণ-

```
Consumer2<AuthProvider, StorageProvider>(

builder: (context, authProvider, storageProvider, child) { }

)
```

```
Consumer4(

 builder: (context, changeNotifier1, changeNotifier2, changeNotifier3, changeNotifier4, child) {

   // your widget

 }

)
```

Ostad

৭৪ Provider vsInheritedWidget এর মধ্যে পার্থক্য কী?

Provider এবং InheritedWidget উভয়ই ফ্লাটার অ্যাপ্লিকেশনে স্টেট ম্যানেজের জন্য ব্যবহার করা হলেও এদের মধ্যে কিছু পার্থক্য রয়েছে। যেমন-

১। API and Simplicity: Provider প্যাকেজ একটি higher-level API প্রোভাইড করে যা সরাসরি InheritedWidget ব্যবহার করার তুলনায় স্টেট ম্যানেজমেন্টকে সহজ করে।

২। Scoped vs. Global State: Provider ডিজাইন করা হয়েছে scoped state ম্যানেজ করার জন্য। এক্ষেত্রে অ্যাপের বিভিন্ন অংশের নিজস্ব ইন্ডিপেন্ডেন্ট স্টেট থাকতে পারে। অন্যদিকে পুরো উইজেট ট্রির জুড়ে গ্লোবাল স্টেট শেয়ার করার জন্য InheritedWidget সুইটেবল।

৩। Change Notification: Provider এ চেঞ্জ নোটিফিকেশন হ্যান্ডেল করার ক্ষেত্রে এবং স্টেটের পরিবর্তনের সাথে সাথে উইজেট রিবিল্ড করা সহজ করে তোলে। অন্যদিকে, InheritedWidget এ আপনাকে চেঞ্জ নোটিফিকেশন ম্যানুয়ালি হ্যান্ডেল করতে হবে এবং উইজেটগুলো আপডেট করার জন্য didChangeDependencies() কল করতে হবে।

৪। Performance Optimization: Provider "proxy providers" নামে একটি কন্সেপ্ট ব্যবহার করে যা ডিপেন্ডেন্ট উইজেটগুলোর অপ্রয়োজনীয় রিবিল্ড এড়িয়ে পারফরম্যান্স আপ্টিমাইজ করতে এলাও করে। অন্য দিকে, InheritedWidget এর ক্ষেত্রে যখনই ইনহেরিটেড ডেটা চেঞ্জ হয়, তখন ডিপেন্ডেন্ট ডেটা আবার রিবিল্ড করতে হয়।

৫। Third-Party Integration: ফ্লাটার কমিউনিটিতে Provider ব্যাপকভাবে ব্যবহৃত হয় এবং অন্যান্য পপুলার প্যাকেজের সাথেও এর ভালো ইন্টিগ্রেশন আছে। যেমন- flutter_bloc, riverpod। এদিকে InheritedWidget হলো একটি lower-level কন্সেপ্ট। স্টেট ম্যানেজমেন্টের উপর fine-grained কন্ট্রোল প্রয়োজন হলে এটি ডিরেক্টলি ব্যবহার করা যেতে পারে।

৭৫ Clean architecture ব্যাখ্যা করুন।

Clean Architecture হলো একটি সফ্টওয়্যার ডিজাইন প্যাটার্ন। কোডকে অর্গানাইজ এবং আরও সহজ স্ট্রাকচার দিতে এটি ব্যবহার করা হয় যাতে কোডটি বোঝার পাশাপাশি টেস্ট এবং মেইন্টেইন করতেও সুবিধা হয়। রেসপন্সিবিলিটির উপর ভিত্তি করে কোডের বিভিন্ন অংশকে এটি আলাদা করে।

Clean Architecture এর সেন্টারে core business logic-টি থাকে যেটি সব external tool বা framework থেকে ইন্ডিপেন্ডেন্ট। এই core logic- টি লেয়ার দ্বারা আবৃত থাকে এবং databases বা user interfaces এর মতো external dependencies হ্যান্ডেল করে।

এটির মূল উদ্দেশ্য হলো core logic-টি external factors এর পরিবর্তনগুলো থেকে clean এবং unaffected রাখা।

৭৬ কিছু বিখ্যাত কোম্পানির নাম বলো যারা ফ্লাটার ব্যবহার করছে।

বর্তমানে বেশ কিছু বিখ্যাত কোম্পানি তাদের অ্যাপ ডেভেলপমেন্টের প্রয়োজনের জন্য ফ্লাটার ব্যবহার করছে। যেমন- Google, Alibaba, Tencent, Square, BMW, Abbey Road Studios, Reflectly, Groupon, eBay Motors, Philips Hue ইত্যাদি।

৭৭ শুধুমাত্র debug mode এ আপনি কীভাবে কোড execute করবেন?

প্রথমে এটি ইম্পোর্ট করতে হবে,

```
import 'package:flutter/foundation.dart' as Foundation;
```

এরপর এখানে kReleaseMode ব্যবহার করা যেতে পারে,

```
if(Foundation.kReleaseMode){ // is Release Mode ??
        print('release mode');
} else {
        print('debug mode');
}
```

এই কন্ডিশন ফলো করে শুধুমাত্র ডিবাগ মোডে কোড এক্সেকিউট করা সম্ভব।

Ostad

৭৮

ডার্টে Iterable collections কী?

ডার্টে, Iterable collection গুলো ভ্যালুর সিকোয়েন্সগুলোকে রিপ্রেজেন্ট করে যা iterated বা looped করা যায়। এরা List, Set, এবং Queue এর মতো ক্লাসগুলো ইনক্লুড করে।

Ostad

৭৯

ডার্টে Concurrency কী?

ডার্টে Concurrency বলতে বোঝায়, একসাথে multiple tasks এক্সেকিউট করার এবিলিটি। ডার্ট কনকারেন্সি হ্যান্ডেল করার জন্য async/await, Futures, Isolates, এবং Streams এর মতো মেকানিজমগুলো প্রোভাইড করে। এই concurrency mechanism গুলো মূল এক্সিকিউশন থ্রেডকে ব্লক না করেই network requests, computations, এবং user interactions এর মতো অ্যাসিঙ্ক্রোনাস টাস্কগুলির এফিশিয়েন্টলি হ্যান্ডেল করার এবিলিটি প্রোভাইড করে।

৮০ ডার্টে Typedef কী?

Typedef হলো একটি কি-ওয়ার্ড। এটি আপনাকে একটি নির্দিষ্ট ফাংশন সিগনেচারে একটি কাস্টম নাম দিতে এলাও করে। এটি একটি স্পেসিফিক টাইপের ফাংশনের জন্য একটি শর্টকাট বা নিকনেইম দেওয়ার মতো।

ধরা যাক, আপনার একটি কমপ্লেক্স সিগনেচারসহ একটি ফাংশন আছে যাতে একাধিক প্যারামিটার এবং একটি রিটার্ন টাইপ involved। এক্ষেত্রে, আপনার যখনই এই ধরনের ফাংশন ব্যবহার করার প্রয়োজন হয় তখন বারবার পুরো সিগনেচারটি লেখার পরিবর্তে, আপনি একটি ছোট, এবং আরও descriptive নাম দিয়ে এটি রিপ্রেজেন্ট করার জন্য একটি টাইপডেফ তৈরি করতে পারেন। যেমন-

```
typedef String CustomFunction(int, double);
```

৮১

ডার্টে Generics কী?

ডার্টে, generics আপনাকে রিইউজেবল কোড তৈরি করতে এলাও করে যা বিভিন্ন টাইপের সাথে কাজ করতে পারে। এটি classes, functions, বা interfaces গুলো ডিফাইন করার একটি ওয়ে প্রোভাইড করে যা টাইপ সেইফটি সেক্রিফাইস করা ছাড়াই বিভিন্ন ডেটা টাইপের সাথে মানিয়ে নিতে পারে।

Generics ব্যবহার করে আপনি আরো ফ্লেক্সিবল এবং রিইউজেবল কোড লিখতে পারেন, কারণ এটি কম্পাইল-টাইম টাইপ চেক মেইনটেইন করার সময় বিভিন্ন ডেটা টেইপও হ্যান্ডেল করতে পারে। এটি রানটাইম এরর এভোয়েড করে এবং বেটার কোড ডকুমেন্টেশন প্রোভাইড করে।

ডার্টে জেনেরিকের কন্সেপ্টটি আরও ভালোভাবে বোঝার জন্য একটি উদাহরণ:

```
class Box<T> {
  T value;

  Box(this.value);

  T getValue() {
    return value;
  }
```

```
  void setValue(T newValue) {
    value = newValue;
  }
}

void main() {
  Box<int> intBox = Box<int>(42);
  print(intBox.getValue()); // Output: 42

  Box<String> stringBox = Box<String>('Hello');
  print(stringBox.getValue()); // Output: Hello
}
```

৮২

ডার্টে Runes কী?

ডার্টে, Runes class, Unicode characters এর সিকোয়েন্স রিপ্রেজেন্ট করে। এটি various languages, emojis, symbols ইত্যাদির পাশাপাশি বেইসিক ASCII range এর বাইরের ক্যারেক্টারগুলোকে ইনক্লুড করে এমন স্ট্রিংগুলির সাথে কাজ করার একটি ওয়ে প্রোভাইড করে।

ডার্টে ইউনিকোড ক্যারেক্টারগুলো UTF-16 এনকোডিং স্কিম ব্যবহার করে রিপ্রেজেন্ট করা হয়, যেখানে কিছু ক্যারেক্টার একের অধিক 16-বিট মান ব্যবহার করে এনকোড করা যেতে পারে। Runes ক্লাস আপনাকে এই এক্সটেন্ডেড ক্যারেক্টারগুলোকে সিঙ্গেল ইউনিট হিসেবে ম্যানেজ করতে দেয়।

Runes এর ব্যবহার আরও ভালোভাবে প্রদর্শন করার জন্য একটি উদাহরণ:

```
void main() {

  Runes runes = Runes('\u{1F600}'); // Unicode code point for the smiley face emoji

  String emoji = String.fromCharCodes(runes);



  print(emoji); // Output: 😀
```

Runes এর ব্যবহার আরও ভালোভাবে প্রদর্শন করার জন্য একটি উদাহরণ:

```
void main() {

  Runes runes = Runes('\u{1F600}'); // Unicode code point for the smiley face emoji

  String emoji = String.fromCharCodes(runes);



  print(emoji); // Output: 😀

}
```

Ostad

৮৩ ডার্টে HTML DOM কী?

ডার্টে HTML DOM হলো একটি HTML ডকুমেন্টের সাথে কাজ করার এবং এর এলিমেন্টগুলো ম্যানিপুলেট করার একটি ওয়ে। এটি একটি tree-like স্ট্রাকচার যা একটি ওয়েবপেজের বিভিন্ন অংশকে যেমন- headings, paragraphs, buttons, এবং images-কে রিপ্রেজেন্ট করে।

Ostad

৮৪ ডার্টে কতোগুলো ডেটা টাইপস সাপোর্ট করে?

ডার্ট বেশ কয়েক ধরনের ডেটা টাইপ সাপোর্ট করে। এর মধ্যে রয়েছে- int, double, String, bool, List, Map, Set, Rune, Symbol ইত্যাদি।

৮৫ ডার্টে Symbol কী?

ডার্টে, Symbol হলো একটি স্পেশাল ডেটা টাইপ যেটি identifiers বা names রিপ্রেজেন্ট করার জন্য ব্যবহার করা হয়। মূলত metaprogramming এবং reflection-এ Symbols ব্যবহার করা হয়। এর পাশাপাশি methods অথবা properties রেফার করতেও Symbol ব্যবহার করা হয়।

Symbol গুলো প্রাইমারিলি এডভান্সড অপারেশনগুলোর জন্য ব্যবহার করা হয়। যেমন- রানটাইম টাইপ চেকিং, মেটাডেটা অ্যাক্সেস করা এবং ডাইনামিক কোড তৈরি করা।

৮৬ Dart Constants কী?

Dart Constant কে একটি immutable object হিসেবে ডিফাইন করা হয়। এর মানে হলো প্রোগ্রামটি এক্সেকিউশনের সময় Dart Constants চেঞ্জ বা মডিফাই করা যাবেনা। এদের মধ্যে রয়েছে- numeric constants (integers এবং floating-point numbers), string constants, boolean constants (true and false), symbol constants (representing identifiers), এবং null constant। এগুলি ভ্যারিয়েবল অ্যাসাইনমেন্ট, এক্সপ্রেশন এবং ফাংশন আর্গুমেন্টে ব্যবহার করা যেতে পারে।

৮৭

Dart Callable Classes কী?

ডার্টে, callable class গুলো হলো এমন Class, যেগুলিকে ফাংশন হিসেবে ট্রিট করা যেতে পারে। কল () মেথড ইমপ্লিমেন্ট করে, একটি ডার্ট

ক্লাস এমনভাবে ব্যবহার করা যেতে পারে যেন এটিও একটি ফাংশন। এর একটি উদাহরণ-

```
class MyCallableClass {
  int call(int a, int b) {
    return a + b;
  }
}

void main() {
  var myClass = MyCallableClass();
  var result = myClass(3, 4); // Invoking the instance as if it were a function
  print(result); // Output: 7
}
```

৮৮ ডার্টে Super Constructor কী?

ডার্টে, সাবক্লাস এক্সটেন্ডস কি-ওয়ার্ড ব্যবহার করে প্যারেন্ট ক্লাসের সব ভেরিয়েবল এবং মেথডগুলো ইনহেরিট করতে পারে। কিন্তু এটি প্যারেন্ট ক্লাসের কনস্ট্রাক্ট ইনহেরিট করতে পারেনা। তাই এটি করতে আমরা ডার্টে Super Constructor ব্যবহার করি। একটি Super Constructor কল করার দুটি উপায় আছে: Implicitly এবং Explicitly

Implicit super: এই ক্ষেত্রে, যখন চাইল্ড ক্লাসের অবজেক্ট ক্রিয়েশন হয়, তখন প্যারেন্ট ক্লাসকে Implicitly বলা হয়। এখানে আমরা সুপার কনস্ট্রাক্টর ব্যবহার করি না কিন্তু যখন চাইল্ড ক্লাস কনস্ট্রাক্টরকে invoke করা হয় তখন এটি ডিফল্ট প্যারেন্ট ক্লাস কনস্ট্রাক্টরকে কল করে।

Explicit super: Parent constructor যদি ডিফল্ট হয় তাহলে আমরা এটাকে Implicit super বলে থাকি, কিন্তু যদি এটি parameters নেয় তবে উপরে উল্লিখিত সিনট্যাক্স অনুযায়ী সুপারক্লাসটি invoke করা হয়।

৮৯ ডার্টে Anonymous Function কী?

ডার্টে, Anonymous Function হলো এমন একটি ফাংশন যার কোনো নাম নেই। এটি স্পষ্টভাবে একটি নাম অ্যাসাইন না করেই একটি ফাংশন ইনলাইন ডেফাইন করার একটি ওয়ে। Anonymous function গুলো lambda functions বা function literals হিসেবেও পরিচিত।

৯০

ডার্টে Recursion কী?

ডার্টে Recursion বলতে এমন একটি প্রোগ্রামিং টেকনিককে বোঝায় যেখানে একটি নির্দিষ্ট কন্ডিশন পূরণ না হওয়া পর্যন্ত একটি ফাংশন বারবার নিজেকে কল করে। এটি প্রব্লেমগুলিকে smaller এবং similar subproblems এ ভাগ করে সলভ করার একটি উপায়।

একটি recursive function এ, ফাংশনের এক্সেকিউশন ২টি মেইন পার্ট ইনক্লুড করে। base case এবং recursive case। base case এমন একটি কন্ডিশন ডিফাইন করে যখন ফাংশনটি নিজেই কল করা বন্ধ করে, এটি নিশ্চিত করতে যে recursion-টি terminate হয়েছে। এদিকে, recursive case এমন লজিককে ডিফাইন করে যেখানে ফাংশনটি মডিফাইড সেট অব প্যারামিটারের সাথে নিজেকে কল করে, base case-এর দিকে প্রোগ্রেস করে।

৯১ ডার্টে Threading ব্যাখ্যা করুন।

ডার্টে, Threading বলতে একটি ডার্ট প্রোগ্রামের মধ্যে একের অধিক থ্রেডের concurrent execution-কে বোঝায়। একটি থ্রেড হলো instructions এর একটি সিকোয়েন্স। এটি প্যারালাল ও অ্যাসিঙ্ক্রোনাস এক্সেকিউশন এলাও করে এবং অন্যান্য থ্রেডের পাশাপাশি ইন্ডিপেন্ডেন্টলি রান করতে পারে।

৯২ Equatable vs Freezed এর মধ্যে পার্থক্য কী?

Equatable এবং Freezed হলো ডার্টের দুটি পপুলার প্যাকেজ যা object equality এবং immutability এর ক্ষেত্রে সাহায্য করে।

এদের কিছু পার্থক্য হলো-

Equatable	Freezed
১। Equatable প্রাইমারিলি ডার্ট ক্লাসে ভ্যালু-বেইসড ইকুয়ালিটি টেস্ট ইমপ্লিমেন্টের জন্য ব্যবহৃত হয়।	১। Freezed হলো একটি কোড জেনারেশন প্যাকেজ যা immutability এর উপর ফোকাস করে।
২। এটি immutability এনফোর্স করেনা। এখানে, অবজেক্টগুলো কীভাবে implemented হচ্ছে এর উপর ভিত্তি করে mutable বা immutable হতে পারে।	২। এটি immutability এনফোর্স করে। এখানে জেনারেট করা ক্লাসগুলো immutable by default। যার মানে হলো, এদের প্রোপার্টিগুলো instantiation এর পর আর মডিফাই করা যায় না।
৩। অবজেক্ট প্রোপার্টিগুলোর equality কম্পেয়ারের জন্য == operator এবং hashCode মেথড এর explicit implementation প্রয়োজন।	৩। এটি অটোমেটিক্যালি ক্লাস প্রোপার্টিগুলোর উপর ভিত্তি করে == অপারেটর এবং হ্যাশকোড মেথড অপারেট করে, যা ডেভেলপারদের boilerplate code লেখা থেকে রিলিভ দেয়।

Ostad

৯৩ FittedBox Widget ব্যাখ্যা করুন।

Flutter-এ FittedBox উইজেটটি child এর aspect ratio বজায় রেখে অ্যাভেইলেবল স্পেসের মধ্যে এর চাইল্ডটি adjust এবং fit করতে ব্যবহৃত হয়। এটি ফিটেডবক্সের সীমাবদ্ধতার মধ্যে ফিট করার জন্য চাইল্ড উইজেটটির স্কেল, সেন্টার এবং পজিশন ফিক্স করে।

১৪ Lazy Loading কী?

Lazy Loading হলো একটি ডিজাইন প্যাটার্ন যা সাধারণত কম্পিউটার প্রোগ্রামিং এবং বেশিরভাগ ওয়েব ডিজাইন এবং ডেভেলপমেন্টে ব্যবহৃত হয় ডেটার লোডিং defer করার জন্য, যতক্ষণ পর্যন্ত না তাদের প্রকৃতপক্ষে প্রয়োজন হয়। সবকিছু আগে থেকে লোড করার পরিবর্তে লেজি লোডিং কেবল on-demand রিসোর্সগুলোকে লোড করতে এলাও করে। যেমন- যখন ইউজার দ্বারা রিকুয়েস্ট করা হচ্ছে বা রিসোর্সগুলো স্ক্রিনে ভিজিবল হচ্ছে। এই এপ্রোচটি পার্টিকুলারলি ইউজ করা হয় বড় অ্যাপ্লিকেশনগুলির সাথে কাজ করার সময়, যেখানে সব রিসোর্স একসাথে লোড করলে পারফরমেন্স স্লো হয়ে যাওয়ার পাশাপাশি এক্সেসিভ মেমোরি লস হয়।

৯৫ BLOC Pattern কী?

BLoC (Business Logic Component) হলো একটি আর্কিটেকচারাল প্যাটার্ন যা সাধারণত বিভিন্ন লেয়ারের মধ্যে ডেটা ফ্লো এবং কমিউনিকেশন ম্যানেজ করার জন্য ব্যবহার করা হয়। এটি বিজনেস লজিককে UI থেকে আলাদা করে কোডটিকে আরও মডুলার এবং মেইন্টেইনেবল করে তোলে। BLoC প্যাটার্নে, একটি BLOC, UI লেয়ার এবং ডেটা লেয়ারের মধ্যে mediator হিসেবে কাজ করে।

৯৬ কোডের কিছু Performance best practices বর্ণনা করুন।

কিছু Performance best practices:

১। প্রয়োজন নেই এমন ক্যালকুলেশন বা অপারেশন পারফর্ম করা থেকে বিরত থাকা।

২। কোডের প্রয়োজনীয়তার উপর ভিত্তি করে এপ্রোপিয়েট ডেটা স্ট্রাকচার সিলেক্ট করা। অর্থাৎ, এমন ডেটা স্ট্রাকচার সিলেক্ট করা যেটি অপ্টিমাল পারফরমেন্সের জন্য efficient lookup, insertion, এবং deletion অপারেশন অফার করে।

৩। ইনপুট/আউটপুট (I/O)অপারেশন সাধারণত ইন-মেমরি অপারেশনের তুলনায় স্লো হয়। তাই I/O অপারেশনগুলির নাম্বার রিডিউস করা বা বাফারিং এবং অ্যাসিঙ্ক্রোনাস টেকনিক ব্যবহার করে এগুলো অপ্টিমাইজ করা।

৪। কোডিং এর ক্ষেত্রে Loops এবং iterations কমন এবং এদের অপ্টিমাইজেশন পারফরমেন্সের উপর প্রভাব ফেলতে পারে। তাই অপ্রয়োজনীয় iterations এভোয়েড করতে হবে।

৯৭ ফ্লাটারে কীভাবে Android StatusBar হাইড করতে হয়?

SystemChrome.setEnabledSystemUIOverlays([])

কোডটি ব্যবহার করার মাধ্যমে এটি হাইড, এবং

SystemChrome.setEnabledSystemUIOverlays(SystemUiOverlay.values)

এই কোডটি ব্যবহারের মাধ্যমে এটি আবার ফিরিয়ে আনা সম্ভব।

৯৮

কীভাবে আপনি ডার্ট কোড থেকে host platform detect করেন?

ডার্ট কোড থেকে হোস্ট প্ল্যাটফর্ম ডিটেক্ট করতে, dart:io লাইব্রেরি ব্যবহার করা যেতে পারে কারণ এটি platform-specific information এবং utilities প্রোভাইড করে। এর একটি এক্সাম্পল-

```
import 'dart:io';

void main() {
  if (Platform.isAndroid) {
    print('Running on Android');
```

```
  } else if (Platform.isIOS) {
    print('Running on iOS');
  } else if (Platform.isWindows) {
    print('Running on Windows');
  } else if (Platform.isLinux) {
    print('Running on Linux');
  } else if (Platform.isMacOS) {
    print('Running on macOS');
  } else {
    print('Unknown platform');
  }
}
```

Ostad

৯৯ কীভাবে ফ্লাটার থেকে .apk এবং .ipa ফাইলস পাওয়া যায়?

apk (Android) এর জন্য যে কোডটি রান করতে হবে-

flutter build apk --release

ipa (iOS) এর জন্য যে কোডটি রান করতে হবে-

flutter build ios --release

১০০

ফ্লাটারে 'yield' keyword-টি কী করে? বুঝিয়ে বলুন।

Flutter- এ 'yield' keyword-টি ব্যবহৃত হয় জেনারেটরস নামের বিশেষ ফাংশনগুলোতে যা সময়ের সাথে ভ্যালুগুলির একটি সিকোয়েন্স তৈরি করে। ফ্লাটারে, আপনি যখন একটি জেনারেটর ফাংশনে 'yield' ব্যবহার করেন, এটি আপনাকে একটি সময়ে কেবল একটি মান প্রডিউস করতে এবং রিটার্ন দিতে দেয়। ফাংশনটি প্রতিটি yield statement-এ পজ করে, কলার কে একটি ভ্যালু প্রোভাইড করে এবং যেখানে থেমেছিল সেখান থেকে আবার কন্টিনিউ করে।