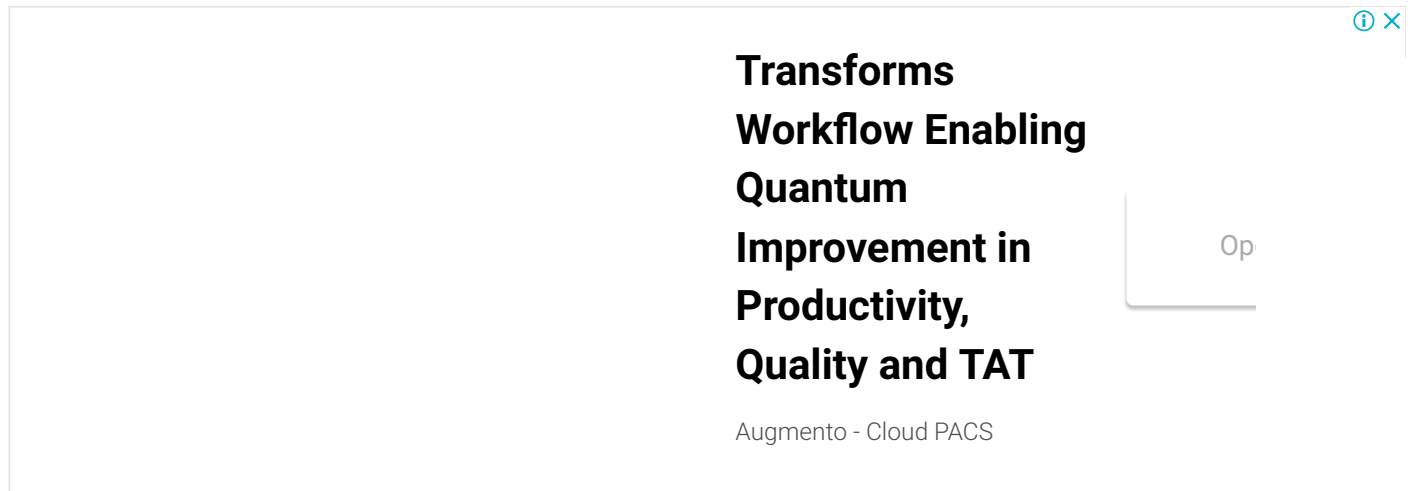




Learn dart event loop and asynchronous call

Time: 2022-2-17



Asynchronous code (<https://developpaper.com/tag/code/>) can be found everywhere in dart. Many library functions return future objects, and you can register handlers to respond to event (<https://developpaper.com/tag/event/>)s such as mouse clicks, file I / O completion, and timing.

This article describes DART's event loop architecture, so you can write better asynchronous code with fewer problems. You will learn how to use future and be able to predict the execution order of the program.

Note: everything in this article applies to both native dart applications (using the dart virtual machine) and dart applications that have been compiled into JavaScript (the output of dart2js). This article uses the term dart to distinguish between dart applications and software written in other languages

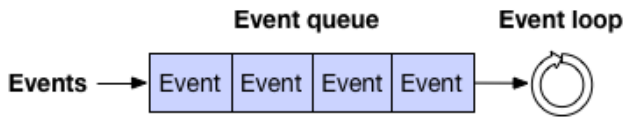
Before reading this article, you should be familiar with the basic knowledge of future and error handling. (<https://developpaper.com/go.php?go=aHR0cHM6Ly9kYXJ0LmNuL3R1dG9yaWFscy9sYW5ndWFnZS9mdXR1cmVz>)

Basic concepts

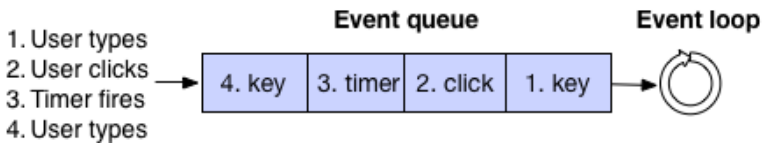
If you have written UI code, you may be familiar with the concepts of event loop and event queue (<https://developpaper.com/tag/queue/>). They ensure that graphical operations and events (such as mouse clicks) are processed one at a time.

Event loops and queues

The job of event loop is to get an event from the event queue and process it. As long as there are events in the queue, repeat these two steps.



Events in the queue may represent user input, file I / O notifications, timers, etc. For example, the following is a picture of an event queue with timers and user input events:

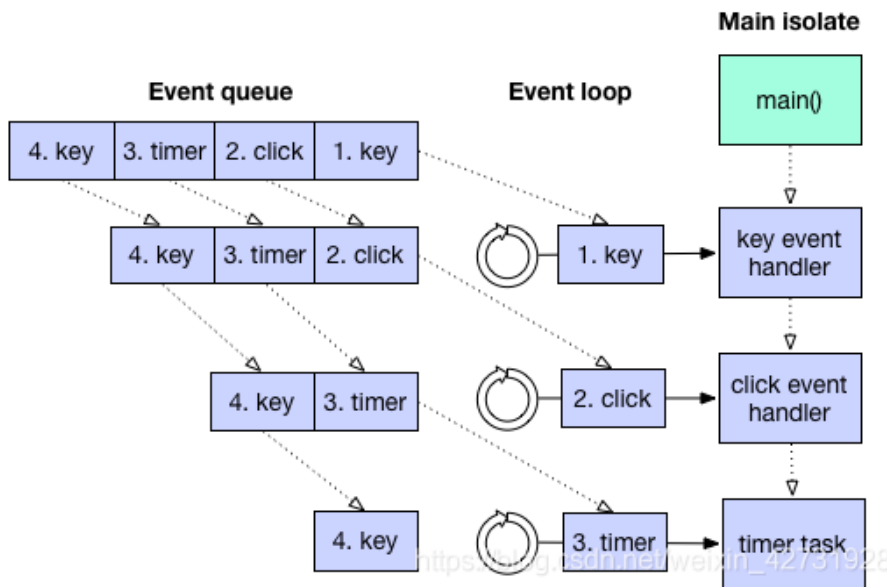


You may be familiar with these in other languages. Now let's talk about how dart language is implemented.

DART's single thread

Once a dart function starts executing, it will continue executing until it exits. In other words, dart functions cannot be interrupted by other dart code.

As shown in the figure below, the first step of a dart program is to execute the main () function by the main isolate thread. When the main () exits, the main isolate thread starts to process all events on the program event queue one by one.

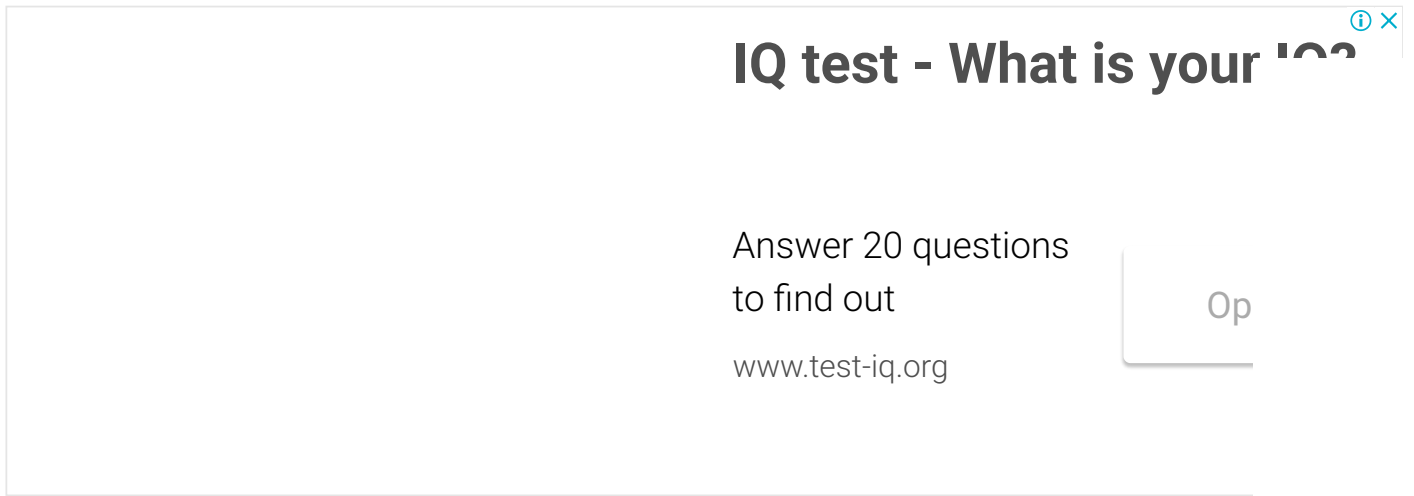


In fact, this is a little too simplistic.

Event loop and queue for dart

The event loop of the dart application has two queues — the event queue and the micro task (<https://developpaper.com/tag/task/>) queue.

The event queue contains all external events: I / O, mouse events, drawing events, timers, communication between dart isolates, and so on.

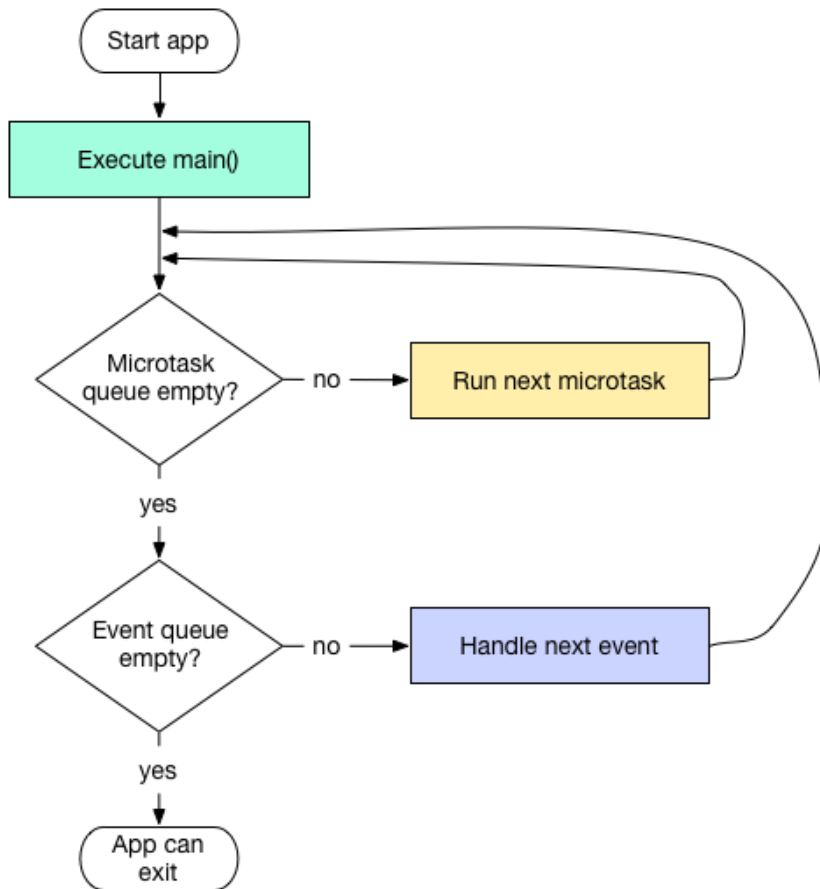


Micro task queues are necessary because event handling code sometimes needs to complete a task later, but before returning control to the event loop. For example, when an observable object changes, it combines several mutation changes and reports them synchronously. Micro task queues allow observable objects to report these abrupt changes before the DOM displays an inconsistent state.

The event queue contains events from dart and other events in the system, and the micro task queue only contains events from dart core code.

As shown in the figure below, when the `main()` function exits, the event loop starts working. First, it executes all micro tasks in FIFO (first in first out) order. Then, it dequeues and processes the first item in the event queue, and then it repeats the cycle: executes all micro tasks, and then processes the next event on the event queue. Once both queues are empty and no more events occur, the application's embedded program (such as browser or test framework) can release the application.

<u> Note: if a user of a web application closes its window, the web application may forcibly exit before its event queue is empty</ u>



Important: when the event loop is executing a task in the micro task queue, the event queue will get stuck: the application cannot draw graphics, process mouse clicks, respond to I / O, etc.

Although you can predict the order in which tasks are executed, you cannot accurately predict when an event loop will remove a task from the queue. Dart event processing system is based on single thread loop; It is not based on any type of time standard. For example, when you create a delayed task, the event will enter the queue at the time you specify. He still has to wait for all events before it in the event queue (including each event in the micro task queue) to be executed. (the delayed task does not jump the queue, but enters the queue at the specified time)

Tip: chain call future to specify the task order

If your code has dependencies, write them explicitly. Explicit dependencies help other developers understand your code and make your program more resistant to code refactoring.

The following is an example of error coding:

```
//Because there is no clear dependency between setting variables and using variables, it is not good.
future. Then (() {... Set an important variable...}).
Timer. Run (() {... Use important variables...}).
```

Instead, write code like this:

```
//Better, because dependencies are explicit.
```

```
future. Then (... Set an important variable...)

then((_) {... Use important variables...});
```

You must set this variable before using it. (you can use `whencomplete ()` instead of `then ()` if you want to execute code even if there are errors.)

If using variables takes time and can be done later, consider putting the code in a new future:

```
//Maybe better: explicit dependencies plus deferred execution.

future. Then (... Set an important variable...)



then((_) {new future (() {... Use important variables...})});
```

Using the new future gives the event loop the opportunity to handle other events in the event queue. The next section details the scheduling code for deferred runs.

How to arrange tasks

When you need to specify some code that needs to be delayed, you can use the following APIs provided by the dart: async Library:

The future class, which adds an item to the end of the event queue.

Register Now for 2022 - 2023

Maths, Physics, Chemistry and Biology made ea:

AhaGuru Education

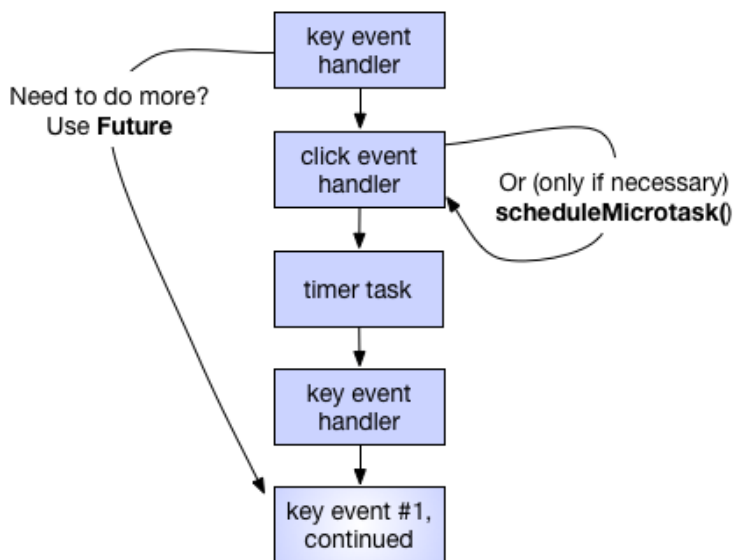
The top-level `schedulemicrotask()` function, which adds an item to the end of the microtask queue.

Examples of using these APIs are in the next section. Event queue: `new future()` and micro task queue: `schedulemicrotask()`

Use the appropriate queue (usually the event queue)

Schedule tasks on the event queue as much as possible and use future. The use of event queue helps to keep the micro task queue short and reduce the possibility that the micro task queue affects the event queue.

If a task needs to be completed before processing any events from the event queue, you should usually execute the function first. If it cannot be executed first, use `schedulemicrotask()` to add the task to the microtask queue.



Event queue: new future()

To schedule tasks on the event queue, you can use `new future()` or `new future delayed()`. These are two future constructors defined in the dart: async library.

Note: you can also use timer to schedule tasks, but if any uncapped exceptions occur in the timer task, your application will exit. Instead, we recommend using future, which is based on timer and adds functions such as detecting task completion and responding to errors.

To immediately put an event in the event queue, use `new future()`:

```
//Add a task to the event queue.

new Future(){

  /... the code is right here

};
```

You can add calls to `then ()` or `whencomplete ()` to execute some code immediately after the new future is completed. For example, when the task of new future leaves the queue, the following code outputs “42”:

```
new Future(() => 21)
  .then((v) => v*2)
  .then((v) => print(v));
```

Use `new future Delayed()` adds an event to the queue after a period of time:

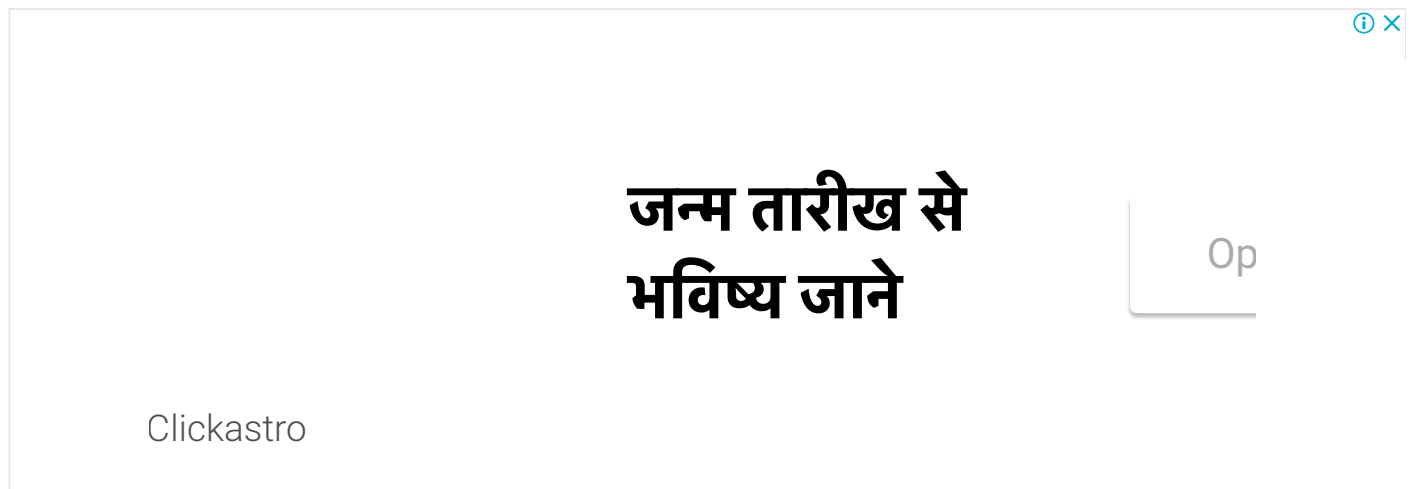
```
//After a period of time, add the event to the queue
new Future.delayed(const Duration(seconds:1), () {
  // ... The code is here
});
```

Although the previous example adds a task to the event queue after one second, the task can only be executed after the main isolate is idle, the micro task queue is empty, and all the tasks previously queued in the event queue have been executed. For example, if the main () function or event handler is running a complex calculation, the task cannot be executed until the calculation is completed. In this case, the delay may be much more than one second.

Important details about future:

1 the function passed to the then () method of future is executed immediately when future completes. (the function did not enter the queue, but was called)

2 if future has completed before calling then (), a task is added to the micro task queue, and then the task executes the function passed to then ().



3 future () and future The delayed () constructor will not complete immediately; They add an item to the event queue.

4. The value () constructor is completed in the micro task, similar to #2

5 Future. The sync () constructor immediately executes its function parameters, and (unless the function returns future, the code will enter the event queue if it returns future) is completed in the micro task, similar to 2. (future. Sync (futureor < T > computation()) this function accepts a function parameter)

Micro task queue: schedulemicrotask()

The async library defines schedulemicrotask () as a top-level function. You can call schedulemicrotask() like this:

```
scheduleMicrotask(() {
  // ... The code is here
});
```

Due to bugs 9001 and 9002, the first call to `schedulemicrotask ()` will put an event that creates a microtask queue in the event queue; This event creates a micro task queue and puts the function assigned to `schedulemicrotask()` into the micro task queue. As long as there is at least one event in the micro task queue, subsequent calls to `schedulemicrotask()` will be correctly added to the micro task queue. Once the micro task queue is empty, it must be recreated the next time `schedulemicrotask()` is called (which means that the first call to `schedulemicrotask()` will not directly enter the micro task queue for immediate execution, and an event creating the micro task queue will be inserted into the event queue first, and this event will still be queued in the event queue).

The result of these errors is that the first task scheduled using `schedulemicrotask()` appears to be on the event queue.

Dart2.9 will insert this code into the first bit of the event queue when scheduling microtask () is called for the first time

One way to add a task to the micro task queue is to call `then ()` on the completed future. For more information, see the previous section (important about future)

Use isolates or workers if necessary

Now that you have read all about scheduling tasks, let's test your understanding.

Keep in mind that you should not rely on DART's event queue implementation to specify the order of tasks. The implementation may change, and `future's then ()` and `whencomplete ()` methods are better choices. However, if you can correctly answer the following questions, you have learned.

practice

Question #1

What does this example print out?

```
import 'dart:async';
void main() {
  print('main #1 of 2');
  scheduleMicrotask(() => print('microtask #1 of 2'));

  new Future.delayed(new Duration(seconds:1),
    () => print('future #1 (delayed)'));
  new Future(() => print('future #2 of 3'));
  new Future(() => print('future #3 of 3'));

  scheduleMicrotask(() => print('microtask #2 of 2'));

  print('main #2 of 2');
}
```

answer


```
main #1 of 2  
main #2 of 2  
microtask #1 of 2  
microtask #2 of 2  
future #2 of 3  
future #3 of 3  
future #1 (delayed)
```


This order should be as you can expect, because the sample code is executed in three batches:

1 code in main() function

2 tasks in micro task queue (schedulemicrotask())

3 tasks in the event queue (New future() or new future delayed())

Remember that all calls in the main () function are executed synchronously from beginning to end. First, main () calls print (), then calls scheduleMicrotask (), then calls new Future.. Delayed (), then call new Future (), and so on. Only callback — as schedulemicrotask(), new future The parameter code of delayed () and new future () will be executed at a later time.



Learn 5th Grade Math Skills

Teachers Create Content, Game Designers Make
with 15,000+ Explanation Videos.

AdaptedMind

Note: at present, if the first call to schedulemicrotask () is commented out, the callbacks to #2 and #3 will be executed before the microtask #2. This is due to bugs 9001 and 9002, as described in microtask queue: schedulemicrotask().

Question #2

Here is a more complex example. If you can correctly predict the output of this code, you will get a shining star.

```
import 'dart:async';

void main() {
  print('main #1 of 2');
  scheduleMicrotask(() => print('microtask #1 of 3'));

  new Future.delayed(new Duration(seconds:1),
    () => print('future #1 (delayed)'));

  new Future(() => print('future #2 of 4'))
    .then((_) => print('future #2a'))
    .then((_) {
      print('future #2b');
      scheduleMicrotask(() => print('microtask #0 (from future #2b)'));
    })
    .then((_) => print('future #2c'));

  scheduleMicrotask(() => print('microtask #2 of 3'));

  new Future(() => print('future #3 of 4'))
    .then((_) => new Future(
      () => print('future #3a (a new future)'))
    .then((_) => print('future #3b'));

  new Future(() => print('future #4 of 4'));
  scheduleMicrotask(() => print('microtask #3 of 3'));
  print('main #2 of 2');
}
```

Assuming that the error 9001 / 9002 is not fixed, the output is as follows:

```
main #1 of 2
main #2 of 2
microtask #1 of 3
microtask #2 of 3
microtask #3 of 3
future #2 of 4
future #2a
future #2b
future #2c
future #3 of 4
future #4 of 4
microtask #0 (from future #2b)
future #3a (a new future)
future #3b
future #1 (delayed)
```

(translator's note)

```

main #1 of 2
main #2 of 2
microtask #1 of 3
microtask #2 of 3
microtask #3 of 3
libraries in 361ms.
future #2 of 4
future #2a
future #2b
future #2c
microtask #0 (from future #
future #3 of 4
future #4 of 4
future #3a (a new future)
future #3b
future #1 (delayed)

```

This is the translator in dart2 Results of running on 9. Dart program will insert the code to create the micro task queue into the first place of the event queue when creating the micro task queue for the first time, which is equivalent to queue jumping.

The bug said by the original author has been fixed

summary

You should now understand DART's event loop and how dart arranges tasks. The following are some of the main concepts of event loops in dart:

The event loop of the dart application uses two queues to execute tasks: the event queue and the micro task queue.

Event queues contain events from darts (futures, timers, isolated messages) and systems (user actions, I / O, etc.).

At present, the micro task queue only has events from dart core code. If you want your code to enter the micro task queue for execution, use `schedulemicrotask()`.

The event loop empties the micro task queue before exiting the queue and processing the next item on the event queue.

Once both queues are empty, the application completes its work and (depending on its embedded program) can exit.

The `main ()` function and all items from the micro task and event queues run on the main isolates of the dart application.

When you schedule an event, follow these rules:

If possible, place it in the event queue (using `new Future()` or `new Future.delayed()`).

Use the `then()` or `whenComplete()` methods of `Future` to specify the task order.

In order to keep the queue of micro tasks as short as possible.

To keep the application responsive, avoid performing compute intensive tasks in any event loop.

To perform compute intensive tasks, create additional isolates or workers.

(the translator originally wanted to summarize an article on dart event loop and asynchronous use, but it is not necessary after translating this article. This article has described all the details clearly)

[English article address]

(<https://dart.cn/articles/arch...> ([https://developpaper.com/go.php?go=aHR0cHM6Ly9kYXJ0LmNuL2FydGljbGVzL2FyY2hpdmUvZXZlbnQtbG9vcCM=\):~:text=A%20Dart%20app%20has%20a,queue%20and%20the%20microtask%20queue.&text=First,%20it%20executes%20any%20microtasks,item%20on%20the%20event%20queue.](https://developpaper.com/go.php?go=aHR0cHM6Ly9kYXJ0LmNuL2FydGljbGVzL2FyY2hpdmUvZXZlbnQtbG9vcCM=):~:text=A%20Dart%20app%20has%20a,queue%20and%20the%20microtask%20queue.&text=First,%20it%20executes%20any%20microtasks,item%20on%20the%20event%20queue.))

Learning English is very necessary for programmers

Tags: application program (<https://developpaper.com/tag/application-program/>), code (<https://developpaper.com/tag/code/>), event (<https://developpaper.com/tag/event/>), queue (<https://developpaper.com/tag/queue/>), task (<https://developpaper.com/tag/task/>)

Recommended Today

JPA hibernate generates DDL by comparing entity and DB (<https://developpaper.com/jpa-hibernate-generates-ddl-by-comparing-entity-and-db/>)

JPA hibernate generates DDL by comparing entity and DB Get entity information from JPA //build configuration //If you do not need to connect to DB, the configuration can be empty, that is, you do not need to set property Configuration configuration = new Configuration(); configuration.setProperty(Environment.DIALECT, DIALECT.getName...)

Generics of typescript (<https://developpaper.com/generics-of-typescript/>)

Write the dumbest Hello world with react + Redux + ES6 (<https://developpaper.com/write-the-dumbest-hello-world-with-react-redux-es6/>)

Is the setstate of react synchronous or asynchronous? (<https://developpaper.com/is-the-setstate-of-react-synchronous-or-asynchronous/>)

Create an interesting bookmark (favorites) Manager — cross ...

Swagger2 reports no operations defined in spec! After closin...

Slog57_ Play with NPM_ Production, release and use of pack...

[12. Leetc — Roman \(https://developpaper.com/12-leetc-rom...](https://developpaper.com/12-leetc-rom...)[CSS cold knowledge scenario \(https://developpaper.com/cs...](https://developpaper.com/cs...)[Front end interview daily 3 + 1 — day 731 \(https://developpa...](https://developpa...)[Spring boot official document Chinese version \(https://devel...](https://devel...)

Pre: [Leetcode-002-adding two numbers \(https://developpaper.com/leetcode-002-adding-two-numbers/\)](https://developpaper.com/leetcode-002-adding-two-numbers/)

Next: [Recruitment order! Yanrong Yunzhou recruits 100 experience officers for new products](#)

[java \(https://developpaper.com/question/tag/java/\)](https://developpaper.com/question/tag/java/)

Search

[php \(https://developpaper.com/question/tag/php/\)](https://developpaper.com/question/tag/php/)[python \(https://developpaper.com/question/tag/python/\)](https://developpaper.com/question/tag/python/)[linux \(https://developpaper.com/question/tag/linux/\)](https://developpaper.com/question/tag/linux/)[windows \(https://developpaper.com/question/tag/windows/\)](https://developpaper.com/question/tag/windows/)[android \(https://developpaper.com/question/tag/android/\)](https://developpaper.com/question/tag/android/)[ios \(https://developpaper.com/question/tag/ios/\)](https://developpaper.com/question/tag/ios/)[mysql \(https://developpaper.com/question/tag/mysql/\)](https://developpaper.com/question/tag/mysql/)[html \(https://developpaper.com/question/tag/html/\)](https://developpaper.com/question/tag/html/)[.net \(https://developpaper.com/question/tag/net/\)](https://developpaper.com/question/tag/net/)[github \(https://developpaper.com/question/tag/github/\)](https://developpaper.com/question/tag/github/)[node.js \(https://developpaper.com/question/tag/node-js/\)](https://developpaper.com/question/tag/node-js/)

Copyright © 2022 Develop Paper (<https://developpaper.com>) All Rights Reserved

(<http://www.miibeian.gov.cn/>) [Sitemap](#)

(<https://developpaper.com/sitemap.xml>) [About DevelopPaper](#)

(<https://developpaper.com/about-developpaper/>) [Privacy Policy](#)

(<https://developpaper.com/privacy-policy/>) [Contact Us](#)

(<https://developpaper.com/contact-us/>)