Substrate_ 入门课第8期
Course 6【Polkadot{.js} - 作业点评】- 助教腾达 林涛

作业点评

访问 Polkadot-JS App 的网址是什么？

C: **https://polkadot.js.org/apps**

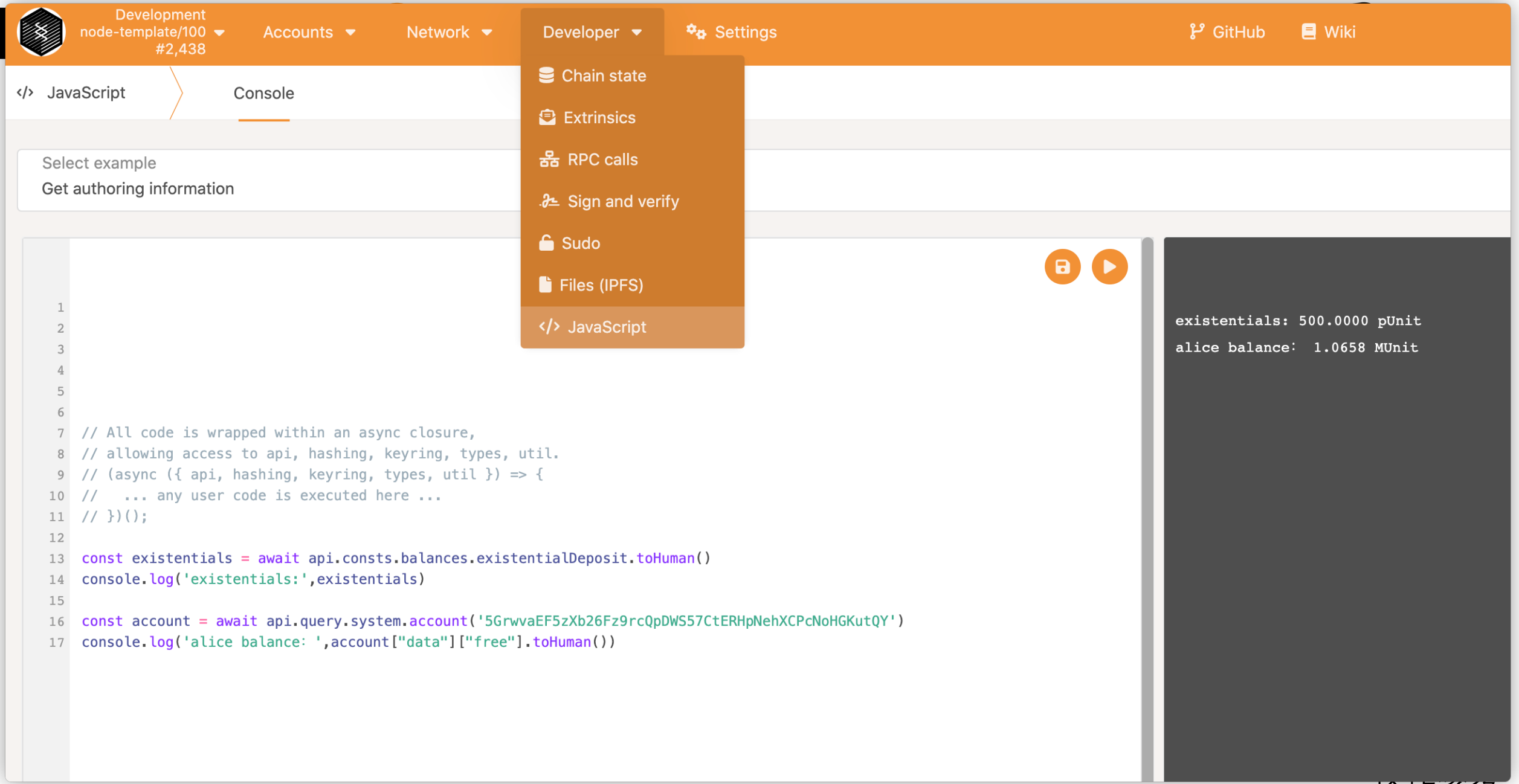
Polkadot–JS App 和 Polkadot–JS API 的区别是什么?

A: 名字上一看就有不同，一个尾缀是 App,一个尾缀是 API

C、Polkadot–JS App 是官方的前端与 Substrate 网络交互，
而 Polkadot–JS API 则提供了官方的 JS 库连去 Substrate 网络。

你可以在 Polkadot–JS App 内做什么操作?

A、查看 Substrate 网络 区块信息
B、对 Substrate 网络作出交易 (Extrinsics)
C、有一个 javascript 编辑器，可对 Substrate 网络写出基础 javascript 与之互动
E、可以对一个信息以某个帐号作签名


Development
node-template/100
#2,438
Accounts
Network
Developer
Settings
GitHub
Wiki
Chain state
Extrinsics
RPC calls
Sign and verify
Sudo
Files (IPFS)
JavaScript
JavaScript
Console
Select example
Get authoring information
// All code is wrapped within an async closure,
// allowing access to api, hashing, keyring, types, util.
// (async ({ api, hashing, keyring, types, util }) => {
//   ... any user code is executed here ...
// })();
const existentials = await api.consts.balances.existentialDeposit.toHuman()
console.log('existentials:',existentials)
const account = await api.query.system.account('5GrwvaEF5zXb26Fz9rcQpDWS57CtERHpNehXCPcNoHGKutQY')
console.log('alice balance: ',account["data"]["free"].toHuman())
existentials: 500.0000 pUnit
alice balance: 1.0658 MUnit

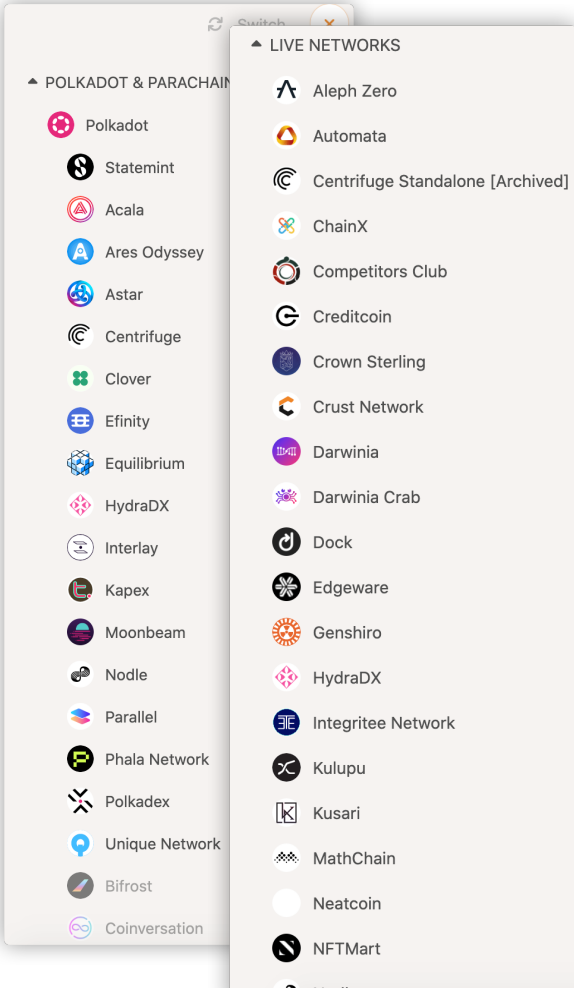
作业点评

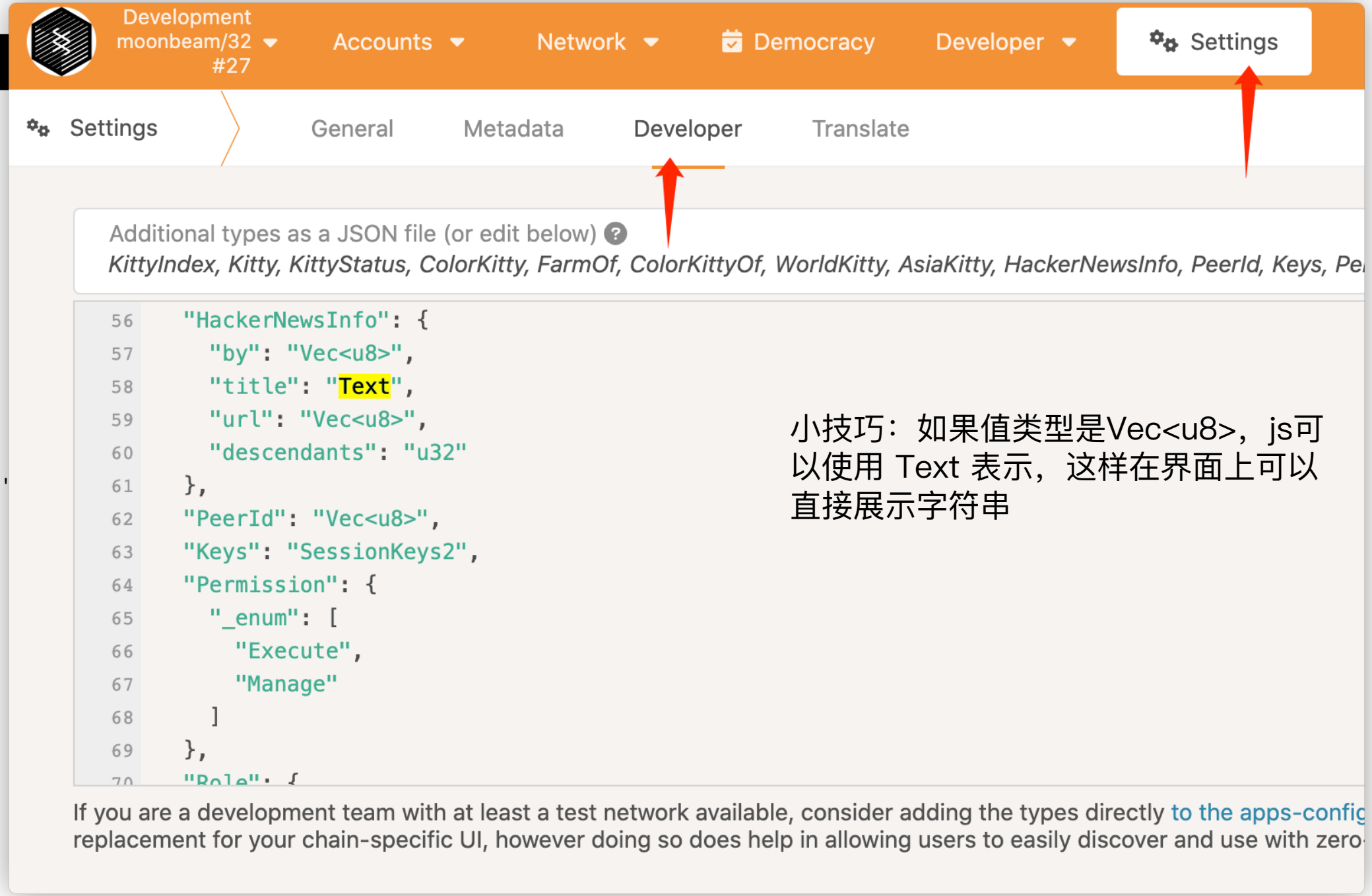
以下哪些生产环境的网络 (LIVE NETWORK) 是 Polkadot-JS App 里默认有支持的?

A、Kusama　B、Acala　C、Kulupu　D、Centrifuge　E、ChainX

如果在 Substrate 端加了自定义类型，我们在 Polkadot-JS App 里需要作什么才能支持连到这个 Substrate 节点?

C、在 Setting 里, Developer tab 里，加自定义的 JSON 对象。见后图





小技巧：如果值类型是Vec<u8>，js可以使用 Text 表示，这样在界面上可以直接展示字符串

作业点评

在 Polkadot-JS API 里，数字默认是用哪个类型代表?

C、[bn.js](https://github.com/indutny/bn.js/)

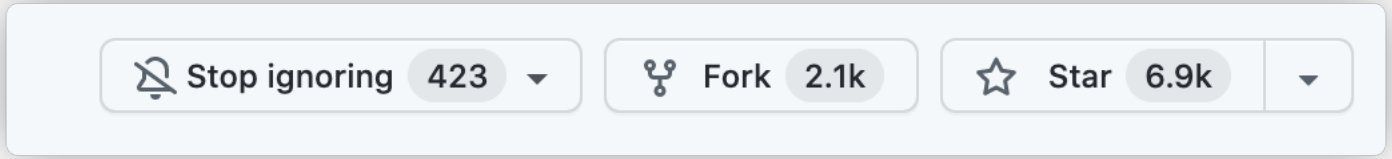
我要查询 Substrate 链上的存储变量,并订阅它的变更，应该用以下哪个方法?

B、`const unsub = await api.query..(value => {...})`

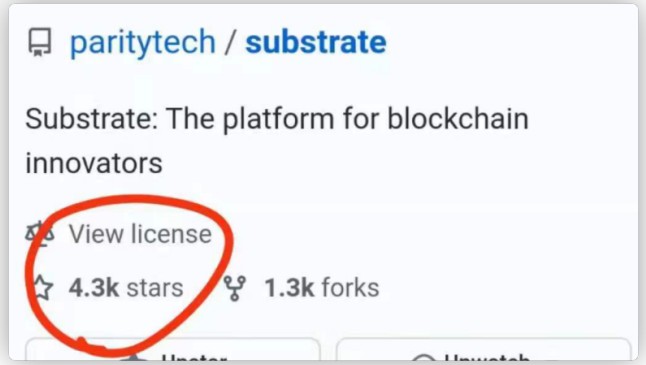
我要对 Substrate 链上发出一次交易，但 **不需要** 订阅交易处理状态，应该用以下哪个方法?

D、`const val = await api.tx..().signAndSend()`

现在在 Github 上的 Substrate repo 约有多少用户给它打了星



一年前的截图：



作业点评

使用polkadot js api来写一个订阅Event的程序

```
const transferFromAliceToBob_Heard_Event = async (api: ApiPromise, amount: number) =>{
  let tx = await api.tx.balances.transfer(bob.address,amount)
          .signAndSend(alice, { }, ({ events = [], status })=>{
                    console.log( 'Transaction status:' , status.type);

                    if (status.isInBlock) {
                              console.log( 'Included at block hash' , status.asInBlock.toHex());
                              console.log( 'Events:' );

                              events.forEach(({ event: { data, method, section }, phase }) => {
                                        console.log( '\t' , phase.toString(), `: ${section}.${method}`, data.toString());
                              });
                    } else if (status.isFinalized) {
                              console.log( 'Finalized block hash' , status.asFinalized.toHex());
                    }

          })
}
```

https://polkadot.js.org/docs/api/examples/promise/system-events https://polkadot.js.org/docs/api/examples/promise/transfer-events

作业点评

使用polkadot js api来写一个订阅Event的程序

```
const subscribeEvent = async (api: ApiPromise) => {
  // Subscribe to system events via storage
  api.query.system.events((events) => {
        console.log(`\nReceived ${events.length} events:`);
        // Loop through the Vec<EventRecord>
        events.forEach((record) => {
              // Extract the phase, event and the event types
              const { event, phase } = record;
              const types = event.typeDef;
             // Show what we are busy with
             if (event.section==   'balances'   ){
                    console.log(`\t${event.section}:${event.method}:: (phase=${phase.toString()})`);
                    // Loop through each of the parameters, displaying the type and data
                    event.data.forEach((data, index) => {
                          console.log(`\t\t\t${types[index].type}: ${data.toString()}`);
                    });
              }
        });
    });
}
```

https://polkadot.js.org/docs/api/examples/promise/system-events https://polkadot.js.org/docs/api/examples/promise/transfer-events

作业点评

使用polkadot js api来写一个订阅Event的程序

Transaction status: Ready
Transaction status: InBlock
Included at block hash 0x631011dc28328526adc3b35d7aefd8d6001b6c05dd9f75ff1bda1be22b04a136
Events:
{"applyExtrinsic":1} : balances.Withdraw ["5GrwvaEF5zXb26Fz9rcQpDWS57CtERHpNehXCPcNoHGKutQY",125000148]
{ “applyExtrinsic” :1} : balances.Transfer [
“5GrwvaEF5zXb26Fz9rcQpDWS57CtERHpNehXCPcNoHGKutQY” ,
“5FHneW46xGXgs5mUiveU4sbTyGBzmstUspZC92UhjJM694ty” ,
1000000000000000]
{"applyExtrinsic":1} : system.ExtrinsicSuccess [{"weight":159200000,"class":"Normal","paysFee":"Yes"}]
Transaction status: Finalized
Finalized block hash 0x631011dc28328526adc3b35d7aefd8d6001b6c05dd9f75ff1bda1be22b04a136

https://polkadot.js.org/docs/api/examples/promise/system-events
https://polkadot.js.org/docs/api/examples/promise/transfer-events

作业点评

优秀作业

https://github.com/whtoo/subA_-polkadot_ts/blob/main/src/tsc/main.ts

仅作参考

Substrate

它是区块链吗？ 它的核心数据结构是merkle 树吗？
最基础的数据结构在: **sp-runtime**

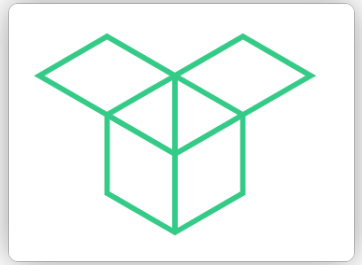
Header:
https://github.com/paritytech/substrate/blob/master/primitives/runtime/src/generic/header.rs

Block:
https://github.com/paritytech/substrate/blob/master/primitives/runtime/src/generic/block.rs

如何 获取时间

Runtime/lib.rs

```
impl pallet_template::Config for Runtime {
    type Event = Event;
    type UnixTime = pallet_timestamp::Pallet<Runtime>; //引入时间实例
}
```



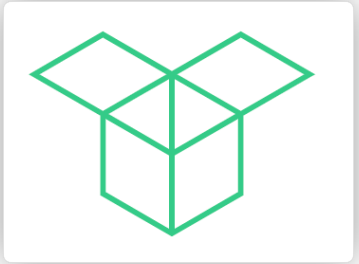
Pallets/template/lib.rs

```
use frame_support::traits::UnixTime; //引入时间

#[pallet::config]
pub trait Config: frame_system::Config {
  type UnixTime: UnixTime; //引入时间类型定义
  ......
}
```

```
pub fn cause_error(origin: OriginFor<T>) -> DispatchResult
{
      let time =T::UnixTime::now(); //实际使用时间
  .........
```

Substrate 扩展 如何 在 BlockNumber、Rust基础类型 之间相互转换



```
use sp_runtime::SaturatedConversion;
use sp_runtime::traits::
     { CheckedAdd, CheckedDiv, CheckedMul, CheckedSub, Saturating };
```

```
let current_block = <frame_system::Pallet<T>>::block_number();    // 从system pallet 获取当前区块的id
let delay_blk: T::BlockNumber = 30u32.into();                    // 从u32转换为 T::BlockNumber
let next_block = current_block + delay_blk;                      //T::BlockNumber类型相加
let next_block = next_block.checked_add(&delay_blk).unwrap();                  //使用安全的加法
let next_block = next_block.saturating_sub(200u32.into());                     //返回饱和值的加减法
let a_u64: u64 = u32::MAX as u64 + 2;                          // 把u64 进过一些计算后 转换为 block nunber
let block_number: Option<T::BlockNumber> = a_u64.checked_add(100).unwrap().try_into().ok();
```

```
//把BlockNumber转换为 u8
let number = TryInto::<u8>::try_into(next_block);
match number {
   Ok(n) =>log::info!("next block:{:?}",n),
   Err(e) => log::info!("current block convert error"),
}
```

https://stackoverflow.com/questions/56081117/how-do-you-convert-between-substrate-specific-types-and-rust-primitive-types

Substrate 扩展 如何 打印日志

```
use sp_std::if_std;
use sp_runtime::print;
```

```
[dependencies.log] # log::info! 需要
[dependencies.sp-std] # if_std! 需要
[dependencies.sp-runtime] # print 需要
```

```
pub fn cause_error(origin: OriginFor<T>) -> DispatchResult {
        let _who = ensure_signed(origin)?;

        log::info!("这里发生了错误，记录日志");
         // 要看到日志，运行需要带上evn参数：RUST_LOG，具体command:
         // RUST_LOG=runtime=debug ./target/release/node-template --dev --lruntime=debug
         print("test `cause_error`!!!!");

    if_std!{
      // This code is only being compiled and executed when the `std` feature is enabled.
      println!("This code is only being compiled. the `std` feature is enabled.");
      println!("这个代码只有在std这个feature启用的时候，才会执行！！");
      println!("the tranx caller is :{:#?}",_who);
    }
  ……
```

```
2022-04-27 21:05:28 INFO 💤Idle (0 peers), best: #8 (0x2eba…ad00), finalized #6 (0x0723…0381)

This code is only being compiled and executed when the `std` feature is enabled.
这个代码只有在std这个feature启用的时候，才会执行！！
the tranx caller is :d43593c715fdd31c61141abd04a99fd6822c8558854ccde39a5684e7a56da27d (5GrwvaEF...)

2022-04-27 21:05:30 INFO 🙌 Starting consensus session on top of parent 0x2ebad……075579ad00
2022-04-27 21:05:30 INFO time now:1651064730.005s
2022-04-27 21:05:30 INFO 这里发生了错误，记录日志
2022-04-27 21:25:48 DEBUG tokio-runtime-worker runtime: test `cause_error`!!!!
```

Rust 参考资料

参考：
关联类型及花式玩法
https://rustcc.cn/article?id=1a2e348e-a4d0-4ee1-9368-353730f2e212

One Block+

Substrate_

Incubate **Developers** | Build **Communities** | Create **Impact** | Shape the **Ecosystem**