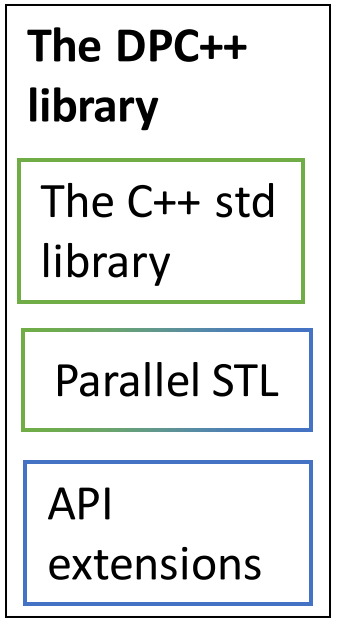
oneAPI DPC++ Library

Concept and overview

The DPC++ library concept

- Goal: Extend DPC++ language making it applicable to a broader set of problems
- The data parallel API is fully C++ compliant
  - No new syntax & keywords, looks like a C++ library
- Best if the standard C++ library "just works"
  - Unlike a library duplicating parts of it
  - May need to subset based on capabilities of device/accelerator
- Parallel STL to run standard algorithms on devices
  - Based on C++17, but needs a special backend and extensions (e.g. non-standard execution policies)



Parallel STL brief overview

oneAPI

- C++17/20 parallel extensions
  - Execution policies define how to run standard algorithms

Execution policy	Meaning
seq	Sequential execution
unseq	Unsequenced SIMD execution. This policy requires that all functions provided are SIMD-safe.
par	Parallel execution by multiple threads
par_unseq	Combined effect of unseq and par

DPC++ execution policies

- A DPC++ execution policy encapsulates a queue
- The queue defines (at run time) the device to execute

```
std::sort(par, v.begin(), v.end());
```

➔

```
std::sort(sycl_policy{q}, dpstd::begin(buffer),
dpstd::end(buffer));
```

Moving toward:

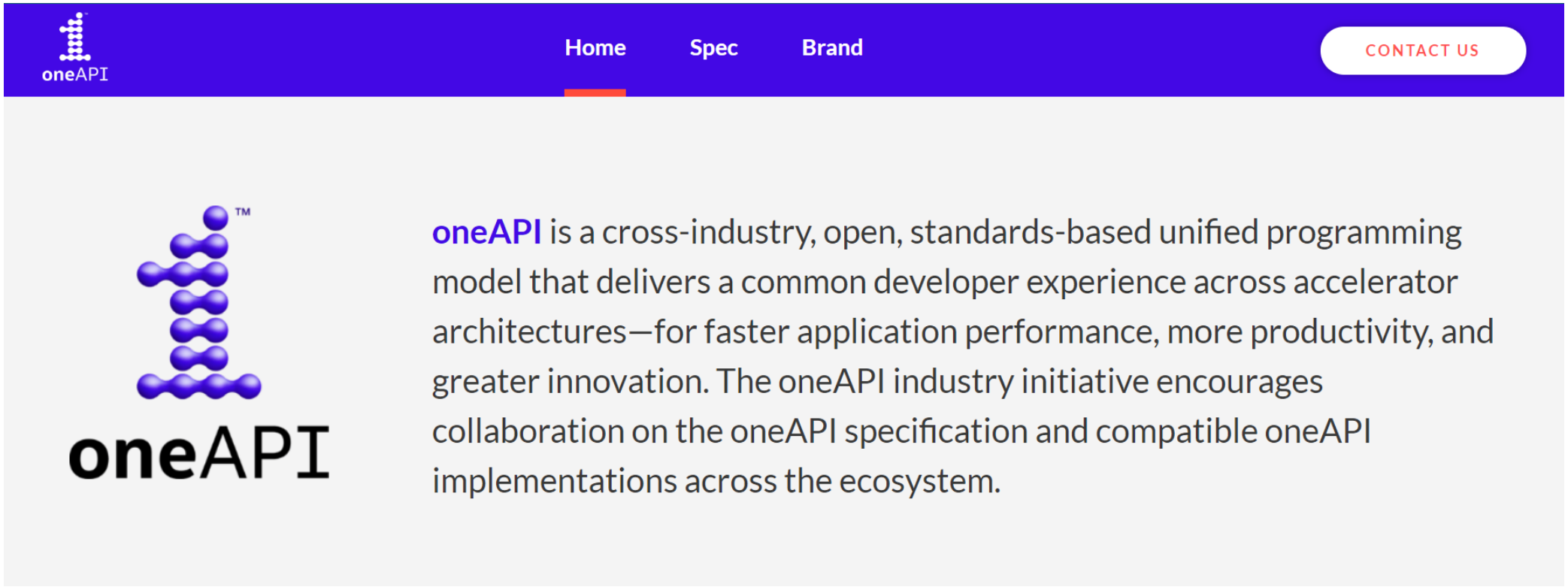
```
std::sort(sycl_policy{q}, v.begin(), v.end());
```

API extensions overview

- Fancy iterators: zip iterator, transform iterator, counting iterator (supported)
- Utility functions: e.g. identity (partially supported)
- Additional algorithms: e.g. segmented prefix scans and reduction (planned)
- Container: vector (planned)
- Enable selected C++14/17/20 features for C++11 (planned)
  - E.g. make_index_sequence (C++14)

Thank you for attending the inaugural oneAPI Technical Advisory Board Meeting!

- Spec available today on oneAPI.com



What's next?

- Next event: ISC'20, Frankfurt, Germany
- Follow-up phone conferences (starting Jan '20)
- Dinner tonight @ Venice Ristorante & Wine Bar
  - Reservation under Sanjiv Shah

backup

Current status for std

- **host:**
- As the code is running on CPU platform, developers can use all everything in std.

- **device:**
- As SYCL spec has restrictions on C++ features used in SYCL device code, some std class/function can’t work in device code. For example, std items using exception, dynamic memory allocation, virtual function… can’t work in SYCL device code.

What basic functionality is available in DPC++ kernels?

- Supported:
- For example: std::swap, std::less, std::tuple, std::negate, std::plus, std::array, std::initializer_list, … [full list in beta documentation]
- Targeted for support:
- <atomic>, <cassert>, <cmath>, <cstdint>, <cfloat>, <climits>, <chrono> (partially), <compare>, <complex>, <limits>, <new> (compile-time APIs), <numeric> (partially), <optional>, <ratio>, <variant>