Microsoft MakeCode
from C++ to TypeScript and Blockly (and back)

C++ Users Group Meeting
April 2018

Thomas Ball, Michal Moskal
and MakeCode team
Microsoft MakeCode
Hands-on Computing for every student

➤ Just works always, everywhere

➤ Physical computing a more inclusive approach to CS education

➤ Path to real-world skills

➤ Extensible platform for partners
Microsoft MakeCode Objectives

1. **Usage** – Increased diversity and number of students engaged/interested in computing and technology

2. **Brand** – Microsoft recognized as an innovator in computing education

3. **Ecosystem** – Democratizing access to the world of intelligent edge devices and enabling a thriving partner ecosystem
The Web (browser)

- Plentiful RAM
- Web App
- JavaScript
- Single-threaded

TypeScript

our contribution bringing the worlds of Web and MCU together

C++

- CODAL
- Reactive/concurrent
- Full bare-metal binary

The microcontroller (MCU)

- Little RAM

World of great frameworks for beginning programming (Blockly)

MakeCode = integration/entry point

Languages, Compilers, Runtime

World of the pro IDE (Eclipse, VS, VS Code)
Innovations

- **Web app** for end-to-end experience
  - no install or need for C/C++ compiler for end-user
  - in-browser compilation to binary

- **TypeScript** as core language
  - API mapping: up to Blockly and down to C++
  - coverage of OO concepts

- **Runtime abstractions**
  - Events, message bus and co-routines
  - support concurrent, reactive programming
TypeScript

Gradually typed superset of JavaScript
• Compiles to JavaScript
• Supports ECMAScript 2015 and latest language features

Types enable productivity tools
• intellisense, navigation, refactoring

http://www.typescriptlang.org/
Blocks and TypeScript

**Blockly**
- on start
  - set counter to 0
- on button A pressed
  - set counter to counter + 1
  - show number counter

**TypeScript**
```typescript
let counter = 0;
input.onButtonPressed(Button.A, () => {
  counter = counter + 1;
  basic.showNumber(counter);
});
```

**Categories**
- Basic
- Input
- Music
- Led
- Radio
- Loops
- Logic
- Variables
- Math

**Namespaces**
- Basic
- Input
- Music
- Led
- Radio
- Game
- Images
- Pins
- Serial
/*
 * Provides access to basic micro:bit functionality.
 */

namespace basic {

  /**
   * Scroll a number on the screen. If the number fits on the screen
   * @param interval speed of scroll; eg: 150, 100, 200, -100
   */
  //% help=basic/show-number
  //% weight=96
  //% blockId=device_show_number block="show\nnumber %number" blockGap=8
  //% async
  //% parts="ledmatrix"

  void showNumber(int value, int interval = 150) {
  
}
API Binding (2)

```cpp
namespace led {

/**
 * Turn on the specified LED using x, y coordinates
 * @param x TODO
 * @param y TODO
 */
//% help=led/plot weight=78
//% blockId=device_plot block="plot|x %x|y %y" block
//% parts="ledmatrix"

void plot(int x, int y) {
    uBit.display.image.setPixelValue(x, y, 1);
}
}
```

Wrapping micro:bit runtime

```typescript
namespace pxsim.led {
    export function plot(x: number, y: number) {
        board().ledMatrixState.image.set(x, y, 255); runtime.queueDisplayUpdate()
    }
}
```
1. From C++ to TypeScript and Blockly

• CODAL: C++ Component-oriented Device Abstraction Layer
  • [https://github.com/lancaster-university/codal-core](https://github.com/lancaster-university/codal-core)
  • Joe Finney and James Devine

• [http://github.com/microsoft/pxt-common-packages](http://github.com/microsoft/pxt-common-packages)
  • glue between CODAL and MakeCode
  • annotated C++ provides standard TypeScript/Blockly APIs for common features

• [http://github.com/microsoft/pxt-adafruit](http://github.com/microsoft/pxt-adafruit)
  • Defines full web app
  • Using common packages and base PXT framework
codal-core

extends, implements

codal-atmega328p

uses

codal-arduino-uno

uses

codal-samd21

uses

codal-circuit-playground

selects from

codal (build shell)
CODAL repos

Build: https://github.com/lancaster-university/codal

Base: https://github.com/lancaster-university/codal-core

• https://github.com/lancaster-university/codal-mbed
  • https://github.com/lancaster-university/codal-samd21
    • https://github.com/lancaster-university/codal-circuit-playground

• https://github.com/lancaster-university/codal-atmega328p
  • https://github.com/lancaster-university/codal-arduino-uno

• ...

codal-core

codal-atmega328p

codal-arduino-uno

codal-samd21

codal-circuit-playground

Blockly, Typescript, Monaco

pxt

pxt-common-packages

codal (build shell)

extends, implements

uses

references

uses

references

selects from
enum class Gesture {
    /**
    * Raised when shaken
    */
    //% block-shake
    Shake = ACCELEROMETER_EVT_SHAKE,
    /**
    * Raised when the device tilts up
    */
    //% block="tilt up"
    TiltUp = ACCELEROMETER_EVT_TILT_UP,
    /**
    * Raised when the device tilts down
    */
    //% block="tilt down"
    TiltDown = ACCELEROMETER_EVT_TILT_DOWN,
}

namespace input {
    /**
    * Do something when a gesture is done (like shaking the board).
    * @param gesture the type of gesture to track, eg: Gesture.Shake
    * @param body code to run when gesture is raised
    */
    //% help-input/on-gesture
    //% blockId=device_gesture_event block="on |%NAME"
    //% parts="accelerometer"
    //% gesture.fieldEditor="gridpicker"
    //% gesture.fieldOptions.width=220
    //% gesture.fieldOptions.columns=3
    //% weight=92 blockGap=12
    void onGesture(Gesture gesture, Action body) {

    }
}
Compiling C++ for MakeCode

• Preprocessing of C++
  • [https://makecode.com/simshim](https://makecode.com/simshim)
  • Determines which C++ entities will be visible/exported via TypeScript Declaration File

• Mosly 1-1 mappings between C++ and TypeScript entities
  • bool, number(s), string, enums
  • lambdas, functions
  • namespaces
  • runtime collections
  • plus a hack for exposing methods of a C++ class

• Invoke C++ compiler to generate runtime binary
  • With exported points
  • Binary incorporated into web app
Target specific override and shim file

https://github.com/Microsoft/pxt-adafruit/blob/master/libs/accelerometer/axis.h

```c
// Override in target to change inversion of axis

#define ACC_SYSTEM NORTH_EAST_UP
#define ACC_UPSIDEDOWN true
#define ACC_ROTATION COORDINATE_SPACE_ROTATED_0

/*
 Laying flat:    0 0 -1
 Standing normally:  0 1 0
 Standing on left side: -1 0 0
*/
```

https://github.com/Microsoft/pxt-adafruit/blob/master/libs/accelerometer/shims.d.ts

```typescript
// Auto-generated. Do not edit.
declare namespace input {

    /**
     * Do something when when a gesture is done (like shaking the board).
     * @param gesture the type of gesture to track, eg: Gesture.Shake
     * @param body code to run when gesture is raised
     */
    //% help=input/on-gesture
    //% blockId=device_gesture_event block="on |%NAME"
    //% parts="accelerometer"
    //% gesture.fieldEditor="gridpicker"
    //% gesture.fieldOptions.width=220
    //% gesture.fieldOptions.columns=3
    //% weight=92 blockGap=12 shim=input::onGesture
    function onGesture(gesture: Gesture, body: () => void): void;
```
MakeCode GitHub repos

• Framework and support
  • https://github.com/Microsoft/pxt
  • https://github.com/Microsoft/pxt-blockly
  • https://github.com/Microsoft/pxt-monaco-typescript
  • https://github.com/Microsoft/pxt-common-packages (CODAL-specific)

• Targets
  • https://github.com/Microsoft/pxt-adafruit
  • https://github.com/Microsoft/pxt-microbit
  • https://github.com/Microsoft/pxt-maker
  • https://github.com/Microsoft/pxt-chibitronics
2. From Blocky to TypeScript to Binary (C++)

- Static TypeScript
- Blocky to Static TypeScript
- Compiling Static TypeScript to Machine Code
Static TypeScript

- Considerations
  - Compile for low-memory footprint
  - Link against pre-compiled C++ runtime
  - All types known at compile time, no runtime checks

- TypeScript without the Any type and “bad parts”
  - a subset of TypeScript, with some type substitutions (number -> int32)
  - excludes
    - the eval function, the with statement, the typeof expression
    - type assertions, var statement
    - access to prototype property and computed properties
    - access to the “this” pointer outside of a class
What’s Left in Static TypeScript?

- standard control-flow statements
- let and const: lexically-scoped variable declarations
- functions (nested) and lambdas
- classes with instance fields, methods and constructors
- interfaces
- generic classes, methods, and functions for code reuse
- namespaces
Blocky to Static TypeScript

- Blockly has limited notion of type
- Perform Hindley-Milner type inference over Blockly AST
- Type errors possible in Blockly, but very rare
Static TypeScript to Machine Code

• TypeScript language service -> typed AST

• Extra checks for Static TypeScript subset

• AST -> IR -> Assembly -> Machine Code

• Tree shaking of AST to remove all unneeded STS code
Compiler and Runtime

- Tagged integers, boxing to move to doubles (JavaScript)
- Automatic conversion from STS numbers to various C++ types in glue code
- Reference counting, closures
- Vtable-based layout of (nominally typed) classes
- Interfaces (i.e., multiple inheritance)
  - per-method, so classes do not have to declare they implement a particular interface
- Generics, through code duplication for now
- Many ES6 features (for-in, lambdas, get/set accessors, etc.)
- Custom debugger support for both native and JS compilers

[https://github.com/Microsoft/pxt-common-packages/blob/master/libs/base/pxtbase.h](https://github.com/Microsoft/pxt-common-packages/blob/master/libs/base/pxtbase.h)
The Web (browser)

- Plentiful RAM
- Web App
- JavaScript
- Single-threaded

TypeScript

our contribution

bringing the worlds of

Web and MCU together

The microcontroller (MCU)

- CODAL
- Reactive/concurrent
- Full bare-metal binary
- Little RAM

World of great frameworks for beginning programming (Blockly)

MakeCode = integration/entry point

Languages, Compilers, Runtime

World of the pro IDE (Eclipse, VS, VS Code)