MIT App Inventor 2

Andrea Sterbini – sterbini@di.uniroma1.it

App Inventor 2: building simple Android apps

Built with Blockly **http://ai2.appinventor.mit.edu**

Build, compile, and deploy Android App on the phone

on IPhone also (but you cannot package apps yet)

Automatic visualization of changes while editing, on Phone or on Emulator

Install **AI2 Companion App**

Run the Companion and connect by QR or code or USB

Apps can be Packaged and installed stand-alone on the phone (Android)

Special tricks

Use an emulator instead than a phone

Genymotion for Windows, MAC or Linux

Note: in Genymotion install the Arm Translation Toolkit

BlueStacks for Windows or MAC (faster)

BEST: share your phone screen on PC with scrcpy (via ADB debug)

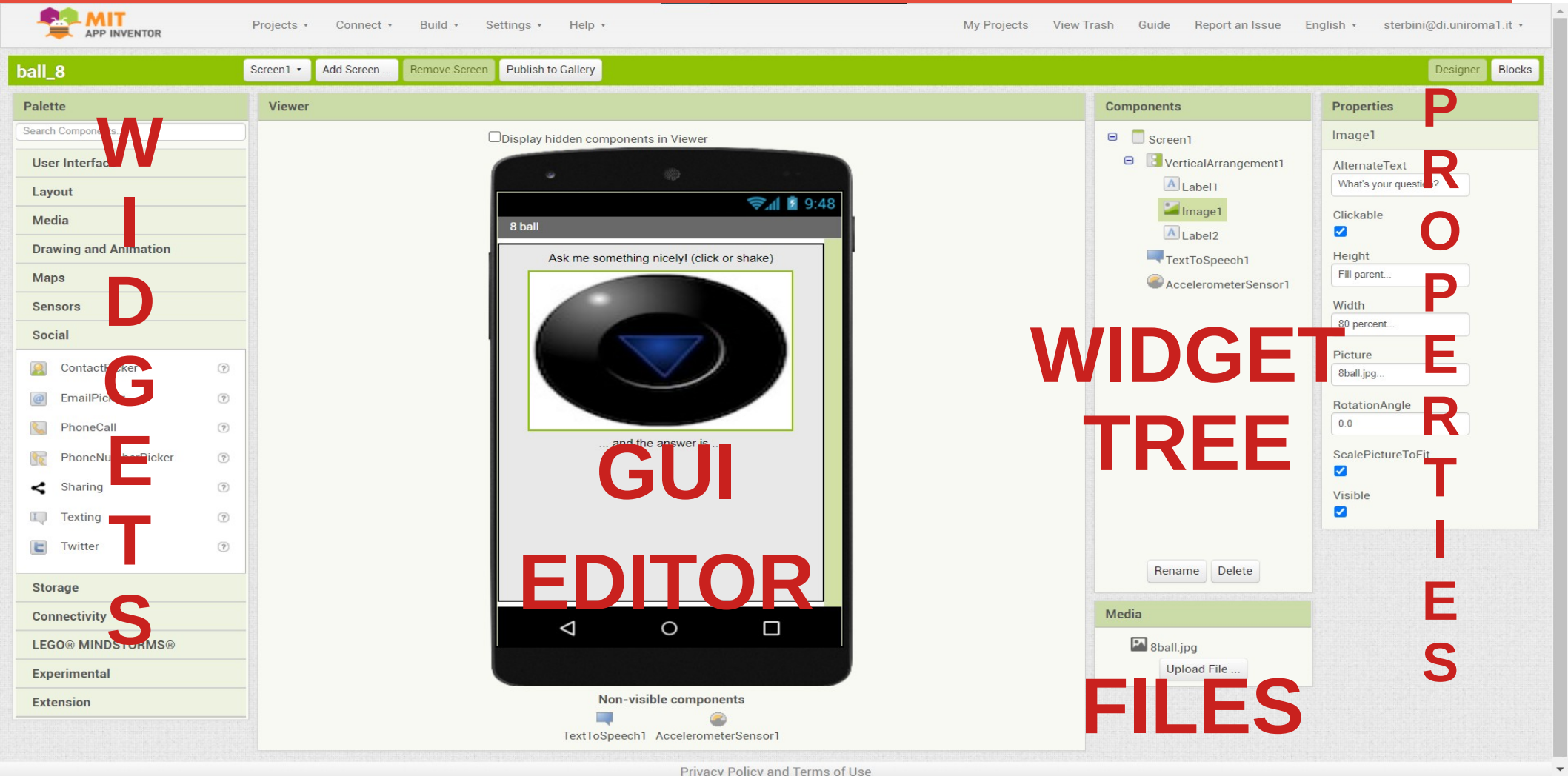
via USB or Wifi (if your phone allows it)

Or you could use a LOCAL server to avoid network problems!!!

AI2Offline [2024] (with minor issues in MacOs)

(or you can compile and run it from http://appinventor.mit.edu/appinventor-sources)

Web-based GUI editor



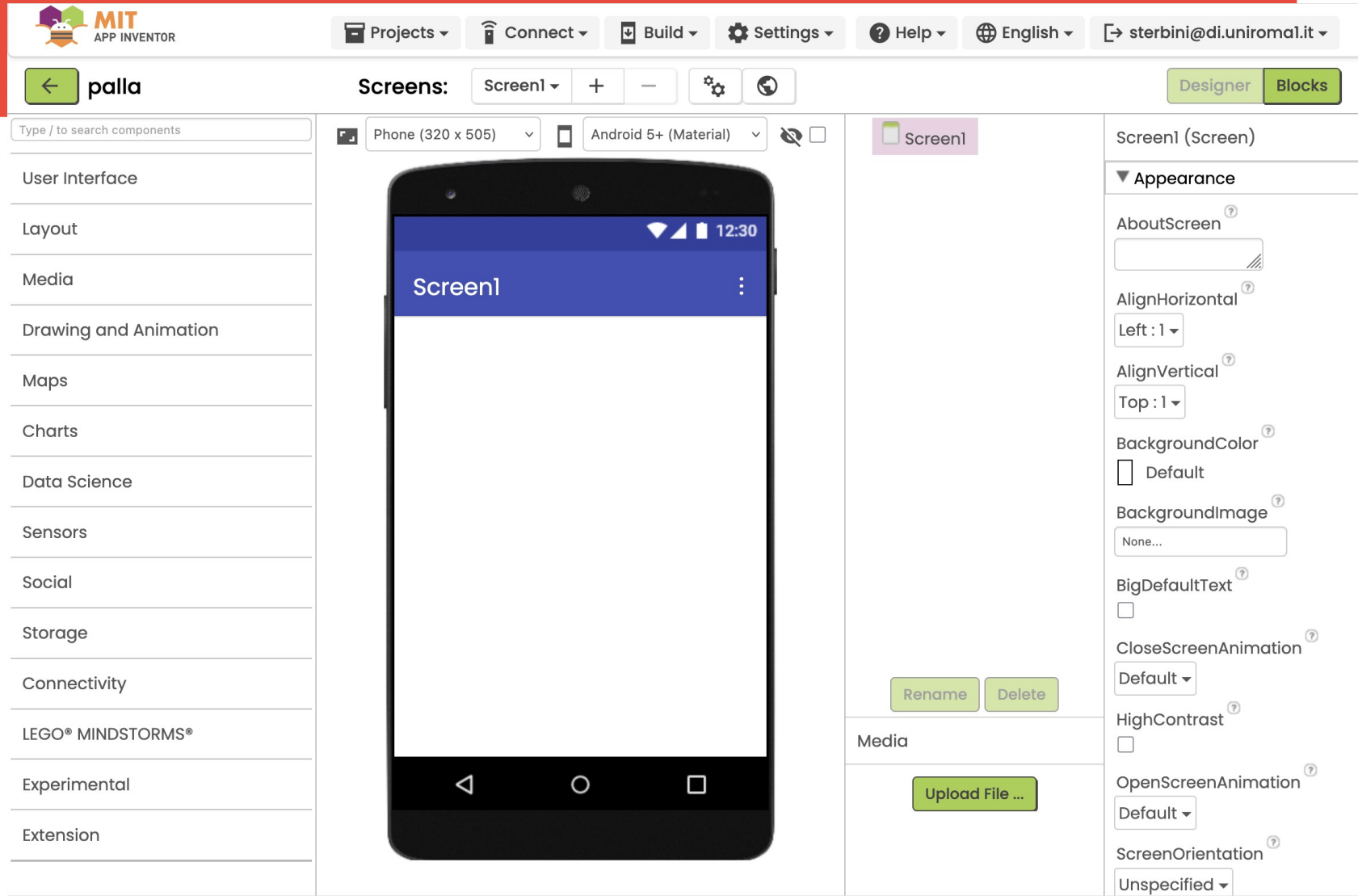
New interface

Cleaner

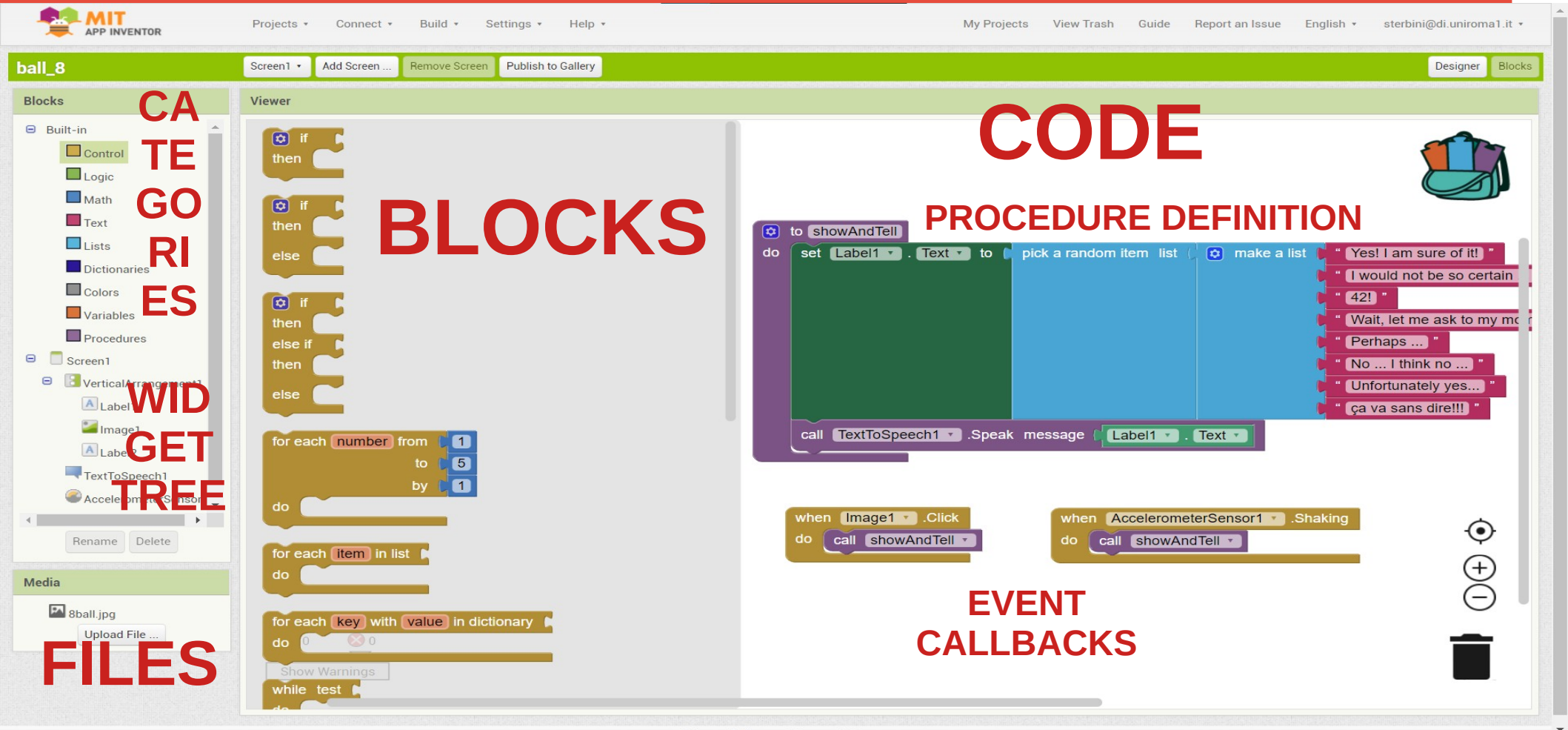
More devices

More properties

AI chatbot



Code editor


MIT APP INVENTOR
Projects
Connect
Build
Settings
Help
My Projects
View Trash
Guide
Report an Issue
English
sterbini@di.uniroma1.it
ball_8
Screen1
Add Screen ...
Remove Screen
Publish to Gallery
Designer
Blocks
Blocks
Built-in
Control
Logic
Math
Text
Lists
Dictionaries
Colors
Variables
Procedures
Screen1
Label
Image1
Label
TextToSpeech1
Rename
Delete
Media
8ball.jpg
Upload File ...
CATEGORIES
WIDGET TREE
FILES
Viewer
if then
if then else
if then else if then else
for each number from 1 to 5 by 1 do
for each item in list do
for each key with value in dictionary do
Show Warnings
while test
BLOCKS
CODE
PROCEDURE DEFINITION
to showAndTell
do
set Label1 . Text to
pick a random item list
make a list
" Yes! I am sure of it! "
" I would not be so certain
" 42! "
" Wait, let me ask to my mo
" Perhaps ... "
" No ... I think no ... "
" Unfortunately yes... "
" ça va sans dire!!! "
call TextToSpeech1 .Speak message Label1 . Text
when Image1 .Click
do call showAndTell
when AccelerometerSensor1 .Shaking
do call showAndTell
EVENT CALLBACKS
Privacy Policy and Terms of Use

App structure

One separate “screen” for each phase (config, login, play levels, results …)

Screens are independent and DO NOT share data or code between them

(but you can use a local **TinyDB** key/value DB component that allows exchanging data)

Or you can **pass/retrieve** some text when **switching to another screen**

Different Apps are independent and DO NOT share data or code (Android)

(but you can exchange data by using an external **WebService + WebDB/CloudDB** or with a **Spreadsheet**)

Resources (video, audio, files, images ...) are bundled in the app apk

Practical Limit: 10 screens max

To mimic many screens **and share code between them** you can hide/show widgets in the same screen by leveraging the widget tree (you just hide/show the parent widget)

Many widgets/objects available

Widgets:	**Buttons and other input fields**
Layout:	**Automatic layout constraints (horizontal, vertical, grid ...)**
Media:	**Sound, Movie, Camera, SoundRecorder, SpeechRecognizer, TextToSpeech, Translator, VideoPlayer, …**
Drawing:	**Canvas, Sprite, Ball**
Maps:	**Maps, Polygonals, Markers, FeaturesCollection (from GeoJson), …**
Charts:	**Chart, ChartData2D, TrendLine, … (NEW!!!)**
Data Science:	**Regression, AnomalyDetection, … (NEW!!!)**
Sensors:	**Accel, Temp, Baro, Gyro, Barcode, Pedometer, NFC, …**
Social:	**Contacts, PhoneCall, PhoneNumber, Email, ~~Twitter,~~ Sharing, Texting**
Storage:	**TinyDB, TinyWebDB, CloudDB (Redis), File, DataFile (CSV/JSON), Spreadsheet**
Connectivity:	**BT Client, BT Server, Web, Serial, ActivityStarter (start other apps)**
Lego:	**NXT, EV3**

New Widgets/Components !!!

Charts:	**Chart**	**(Line, Area, Scatter, Bar and Pie)**
	ChartData2D	**(XY plots)**
	TrendLine	**(Linear, Quadratic, Logarithmic, Exp)**
Data Science:	**Regression**	**(produces a TrendLine)**
	AnomalyDetection	**(abnormal = Z-score above a threshold)**
Storage:	**DataFile**	**(CSV or JSON → table)**
	Spreadsheet	**(on Google Drive)**
Experimental:	**ChatBot**	**(OpenAI ChatGPT or Google PaLM),**
	ImageBot	**(OpenAI DALL-E),**
	FirebaseDB	**(Google Cloud DB)**

Data types

Numbers, Strings, Lists, Lists of Lists, Dictionaries, (Booleans)

All interface widgets are objects with:

- **Predefined Properties** (pre-set in the IDE, or read/changed by program)
- **Events** that they can generate on interaction
- **Methods** that can be called

Some objects are not visual (i.e. BluetoothClient, File, DBFile, Sound, …)

Computed results are shown with a “puzzle” connector (ovals in Scratch)

Some static data type enforcement is present (is checked but not shown)

NEW data types and methods

Text: obfuscated text

Lists: foreach iterator

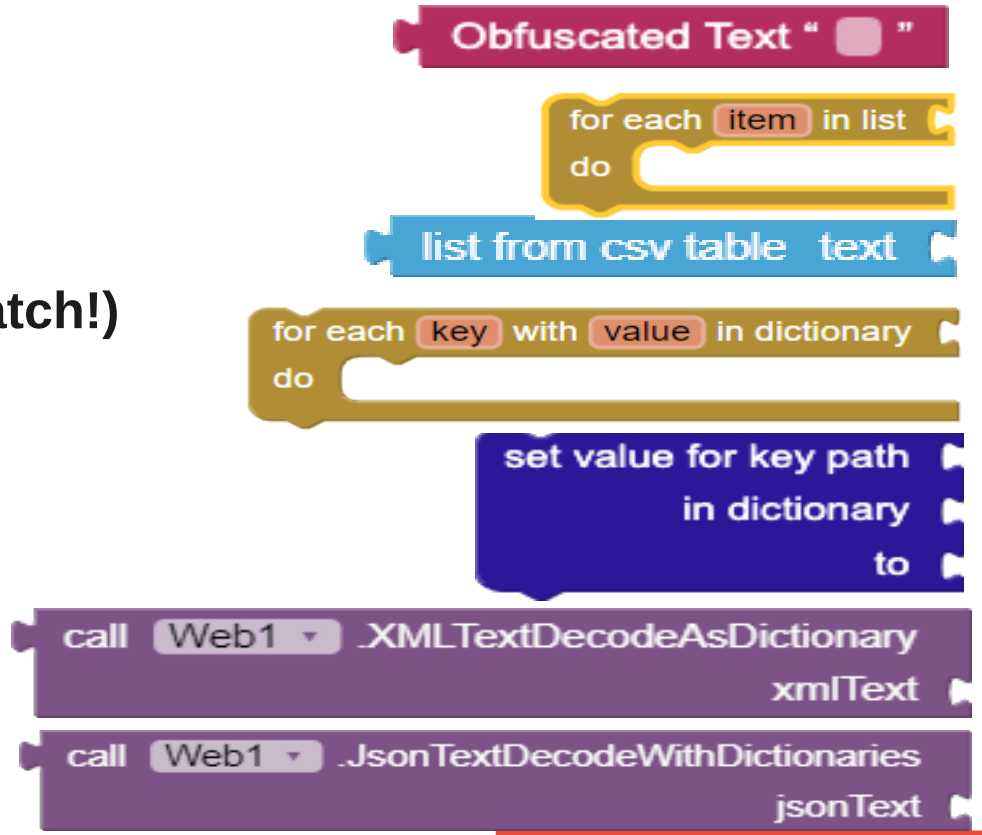
- **CSV <=> list of lists**
- **list of pairs as a dictionary (FIRST match!)**

Dictionaries!

- **with key/value enumerator**
- **with path access to inner values**

XML => convert to dictionary

JSON => convert to dictionary

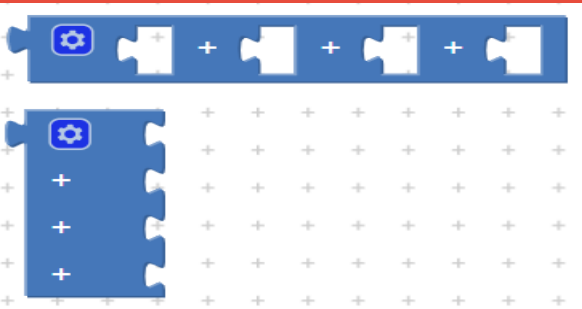


(Visual) Language style / Blocks symbology

Inline or external inputs

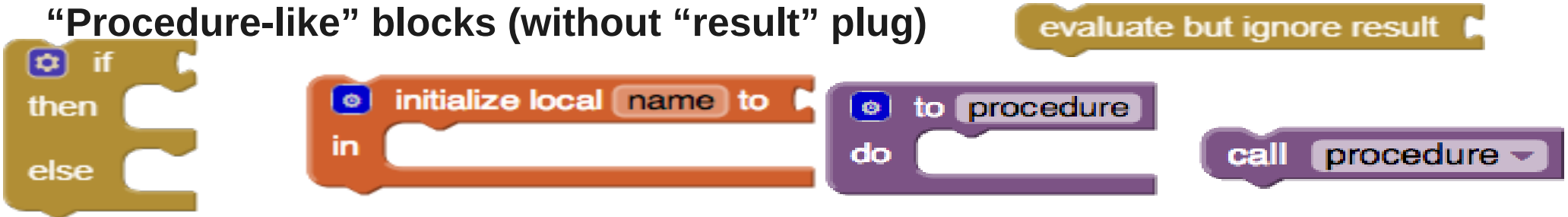
Extensible blocks to allow for many inputs

Text-based blocks (no pre-scholar)



“Function-like” blocks (with “result” plug)

“Procedure-like” blocks (without “result” plug)



Code style: event-based – function-based modularization

You implement mainly Events, Procedures and Functions

GLOBAL variables are defined outside any Event/Function/Procedure

You can define variables LOCAL to the procedure/function

- Can be changed/used **only within their "scope bracket"** (or as a return value)

This allows a "functional decomposition" style (but no lambdas/function passing)

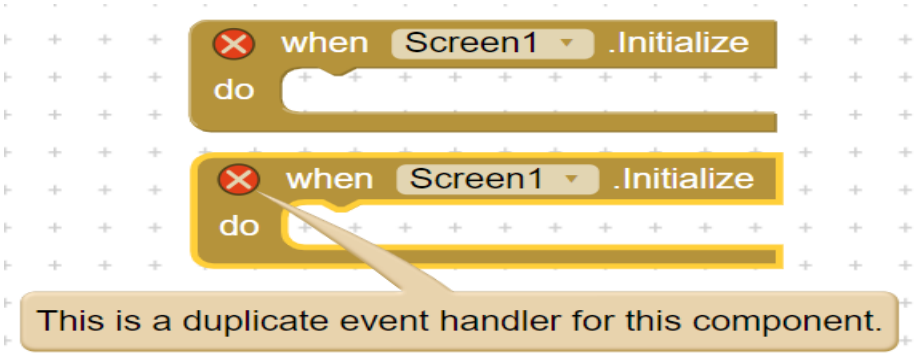
Limited support to debugging

- You can "collapse" the functions/events/procedures result
- You can "Do it" a block and show the result
- You can enable/disable some blocks
- You can "comment" your blocks
- Warning and Errors appear as yellow or red triangles
- All changes are automatically reflected in the **Appinventor Companion** app

Execution model:
event-based programming WITHOUT concurrency

NO multiple concurrent code for same event

NO message passing



Almost all objects generate events when interacted with

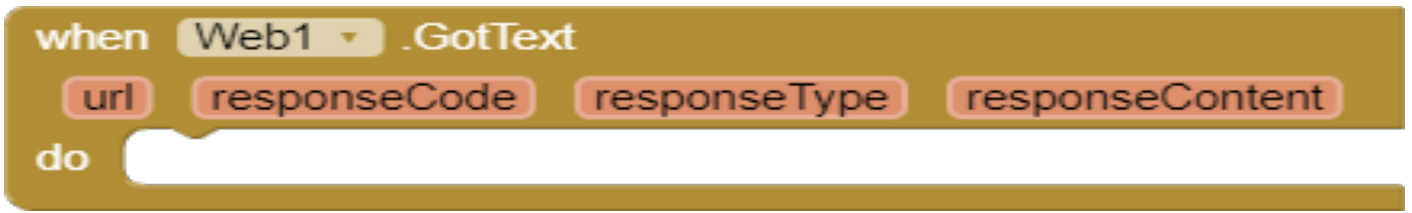
E.g. “When the screen changes”, “When the button is clicked”,
“When got/lost focus”, “Before/After choosing an item”,
“When the screen orientation is changed”, “When the file has been read”
”When the web page has been retrieved”, “When the ball hits a border”,
”When the icon is dragged” …

Asynchronous protocols!

Asynchronous protocols are split in 2 or more phases

E.g. “Ajax query to get a web URL”

“When the response arrives” events



This to remove busy wait and to get an async interaction

To behave differently for different cases, you can use globals as semaphores

PARTIAL object orientation (no way to add properties or to clone)

How to enable students' cooperation

[Kate Feeney's MA thesis at the Mills College]

Ask each student to implement just one screen of a complex App

Start with a template App (just 10 empty screens and media files)

Students should agree on data interactions, data formats and names

- Common resources/files can be shared among screens
- Communication between screens is handled by TinyDB objects

At the end you merge all the screens made by the students into a single App (with the AI2 Project Merger Tool)

Homework: build an app/game cooperatively

Other ways to organize collaboration projects

Multiple interacting applications can communicate through

- **Bluetooth** (direct communication + protocol implementation) (no async communication)
- **Wifi + CloudDB** (central coordination by data sharing)

Examples:

- **Collect and map features on the field in real time** (geolocalized data collection)
- **Collect data from sensors and visualize them in real time** (physics experiments)
- **Collect data and do data-analysis and visualization** (statistics)

Extensions (written in Java/native)

ImageProcessor:	**weighted combination of images pixel-by-pixel**
VectorArithmetic:	**vector sum**
SoundAnalysis:	**pitch decoder (note recognition)**
Posenet:	**body pose estimation in a video (key joints and eyes/nose of a person)**
BluetoothLE:	**Bluetooth Low Energy**
ScaleDetector:	**pinch zoom/reduce interaction**
Look:	**classify images/videos**
ImageClassifier:	**classify images/videos with your model**
	And MANY MANY MANY MORE!

Computational Thinking topics

Algorithm, structured coding, functions, local variables, data structures, types (enforced but not visually highlighted)

GUI programming, Event programming

NO simple concurrency (all events are single flow of computations + async)

More limited and easier than Snap! More powerful than Scratch

Mobile games

Multiplayer apps (connected by WebDB or Bluetooth)

Cooperative development!

Interdisciplinary topics ideas

So many sensors on a phone!!!	→ →	Physics experiments! Data collection!
Serial communication with Arduino	→ →	Home automation, robotics?
Protocol simulations with Bluetooth	→ →	Networks
NFC or QR codes	→ →	tangible interaction? Tagged info?
Maps, GPS, Maps Annotations	→ →	Geography, History, Geotagged data collection?
Media	→ →	Art, Literature
Text to Speech/Speech recognition	→ →	2nd Language?
Lego EV3	→ →	Robotics? Physics? ….
… please suggest!		

PLENTY of alternative systems inspired/extending AppInventor

KODULAR.io

NIOTRON.com

ANDOIDBUILDER.in

THUNKABLE.com

APPRAT.io