Department of Information Engineering
and Computer Science

UNIVERSITÀ
DI TRENTO

Interactive Code Playgrounds on Android

Using mobile devices to make educational technologies more accessible in contexts of scarcity

Author: Salvatore Gilles Cassarà

Supervisors: Lorenzo Angeli, Maurizio Marchese

PROBLEM STATEMENT

Scarcity:

- 1/3 of the global population lacks internet access, which might not be peristent when present
- Access to only low-end, less performant devices
- Universities can not afford enough computers to accomodate all their students
- Lack of digital skills

Educational technologies used in these contexts must take into account such constraints

PROBLEM STATEMENT

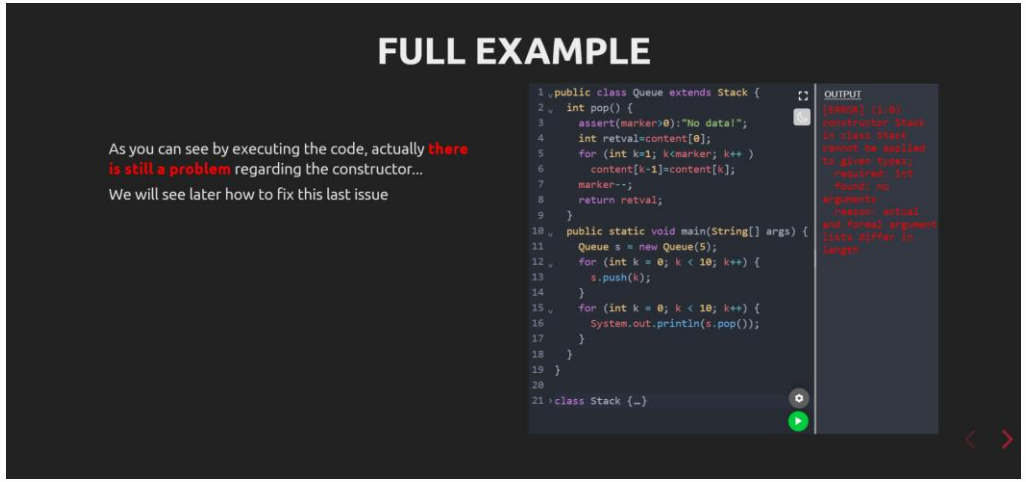
At the same time, an educational technology should:

- Blend theory with hands-on activities
- Present an alternative to IDEs, which can be difficult to set up and use
- Provide a standardized environment

INTERACTIVE CODE PLAYGROUNDS (ICP)

An alternative slideshow system with core ideas of accessibility and inclusivity

- Standardized and platform-independent
- Accessed through browsers
- Familiar environment: editors put into slides
- Works offline
- Different languages available



Example of ICP slides

GOAL

Interactive Code Playgrounds on mobile:

- Need self or cloud hosting
- Hidden contents in the slides

Mobile devices are an important tool for inclusivity:

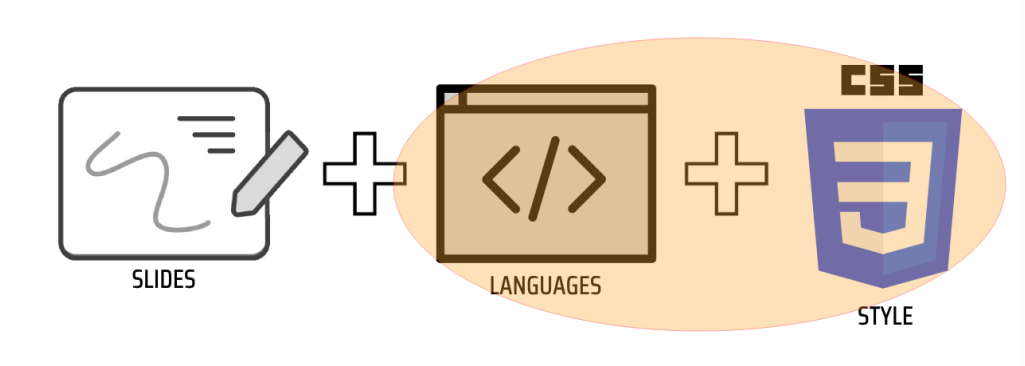
- Important percentage of mobile-only users in the Global South
- Easier to use and more affordable than computers

Create an app to better distribute ICP slides on Android mobile devices

DEVELOPMENT: 1st ARCHITECTURE

- Hosts the slides and shows them in an embedded browser
  - Embedded browser: GeckoView

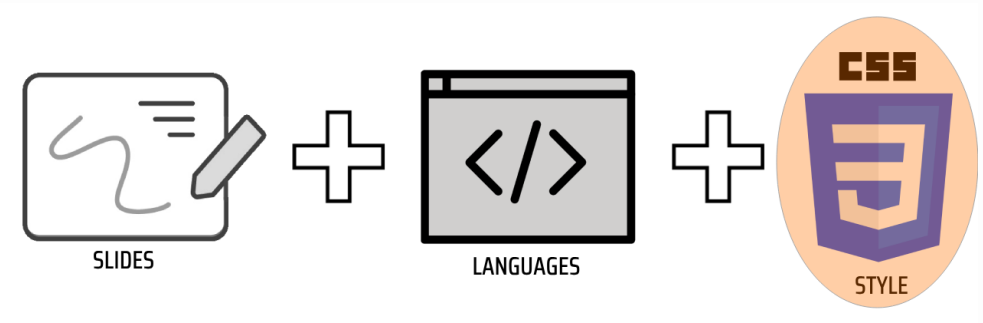
- App contains support for all ICP languages and style files



- Big storage usage (around 400MB) and slow (initialization times can take up to 3 minutes)

DEVELOPMENT: FINAL ARCHITECTURE

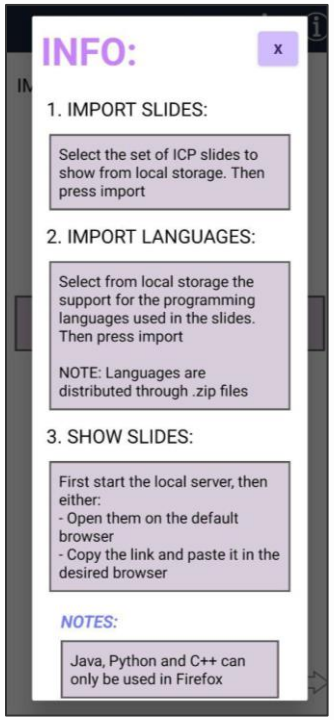
- Modular structure: app only provides style files, user imports slides and support for languages

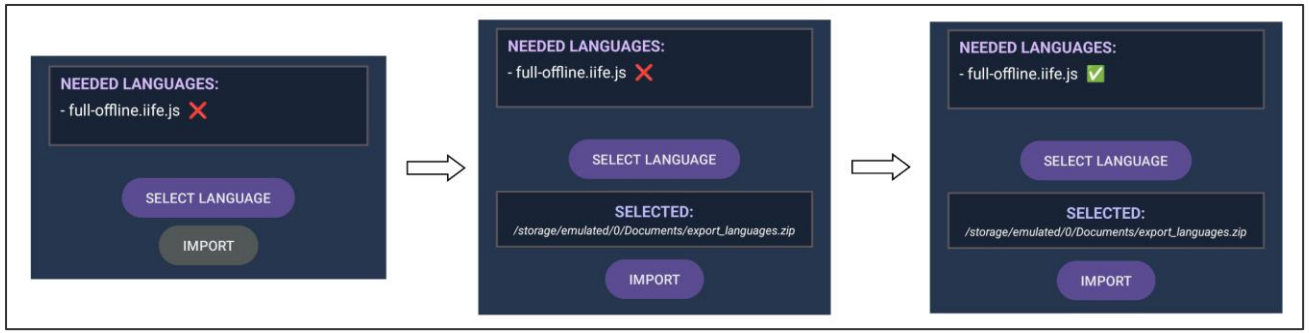


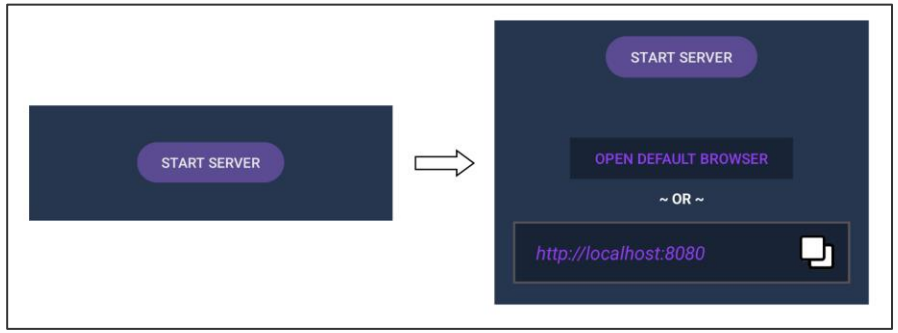
- Uses browsers natively available on the device, only takes care of locally hosting
- Much faster and lighter (around 30MB)
- Targets down to Android 5

DEVELOPMENT: FINAL ARCHITECTURE

- UI focuses on guiding the user through the app in a simple way

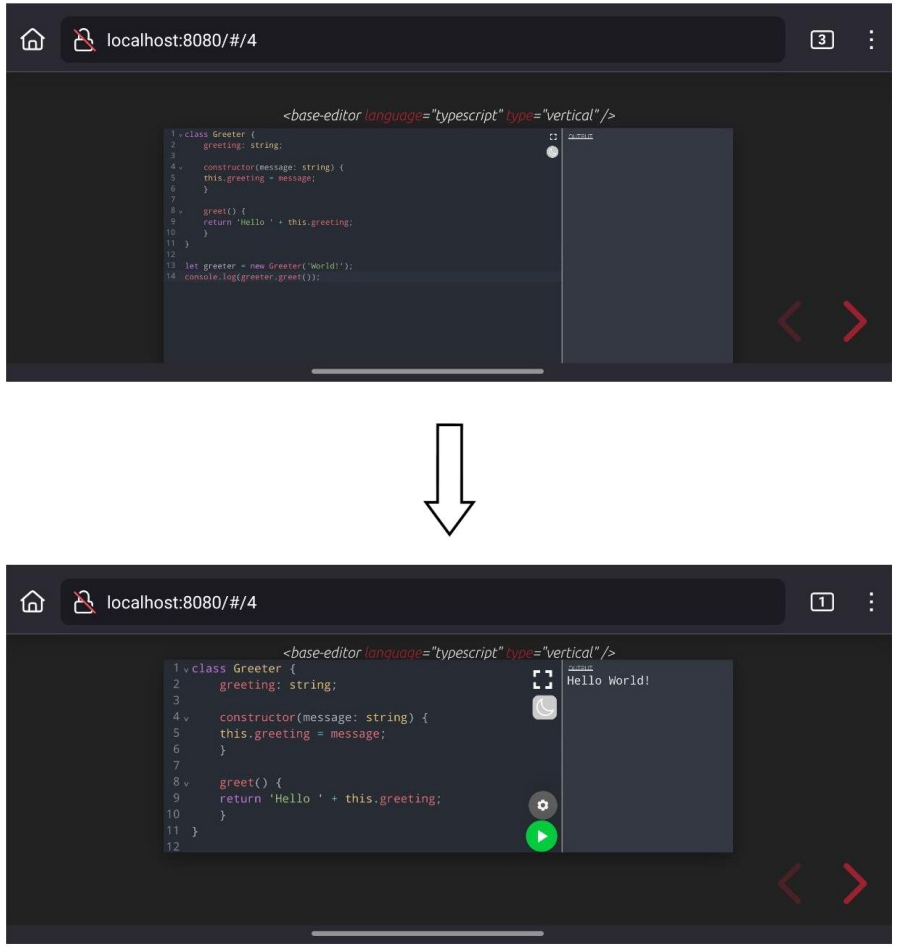






DEVELOPMENT: ORIGINAL ICP PROJECT

- Added possibility to export support for languages for mobile devices
- Fixed style to fit mobile screens



RESULTS: PERFORMANCES

The app was tested on:

- OnePlus Nord 2t, Android 14, 8GB RAM (2022)
- Huawei Nova Young, Android 6, 2GB RAM (2017)
- Huawei Mediapad T3 7, Android 7, 1GB RAM (2017)

Following the example of the original ICP project, the app was mainly tested on the lowest-end device available: the **Huawei Mediapad T3 7**

RESULTS: PERFORMANCES

Programming Language	Initialization (s)	Code Execution (s)
Java & java-offline	100	18.5
Java & full-offline	105	19
Python & python-offline	41	Immediate
Python & full-offline	46.5	Immediate
C++ & cpp-offline	[Hangs]	[Hangs]
C++ & full-offline	[Hangs]	[Hangs]
Standard ML	Immediate	9
Typescript	Immediate	1.7
SQL	Immediate	8
Processing, P5, Javascript	Immediate	Immediate

- On the top, the performances of ICP on a Huawei Mediapad T3 7 using the app

- On the bottom, the performances of ICP on a PC with 4GB DDR3 RAM and a dual-core AMD E-300 1.3GHz CPU

Programming Language	Initialization (s)	Code Execution (s)
Java	59	15.5
Python	59	Immediate
C++	209	0.5 to 8
Processing	Immediate	Immediate
P5	Immediate	Immediate
Standard ML	Immediate	4.6
Javascript	Immediate	Immediate
Typescript	Immediate	0.6
SQL	Immediate	0.5

RESULTS: PERFORMANCES

Programming Language	Initialization (s)	Code Execution (s)
Java & java-offline	100	18.5
Java & full-offline	105	19
Python & python-offline	41	Immediate
Python & full-offline	46.5	Immediate
C++ & cpp-offline	[Hangs]	[Hangs]
C++ & full-offline	[Hangs]	[Hangs]
Standard ML	Immediate	9
Typescript	Immediate	1.7
SQL	Immediate	8
Processing, P5, Javascript	Immediate	Immediate

- On the top, the performances of ICP on a Huawei Mediapad T3 7 using the app

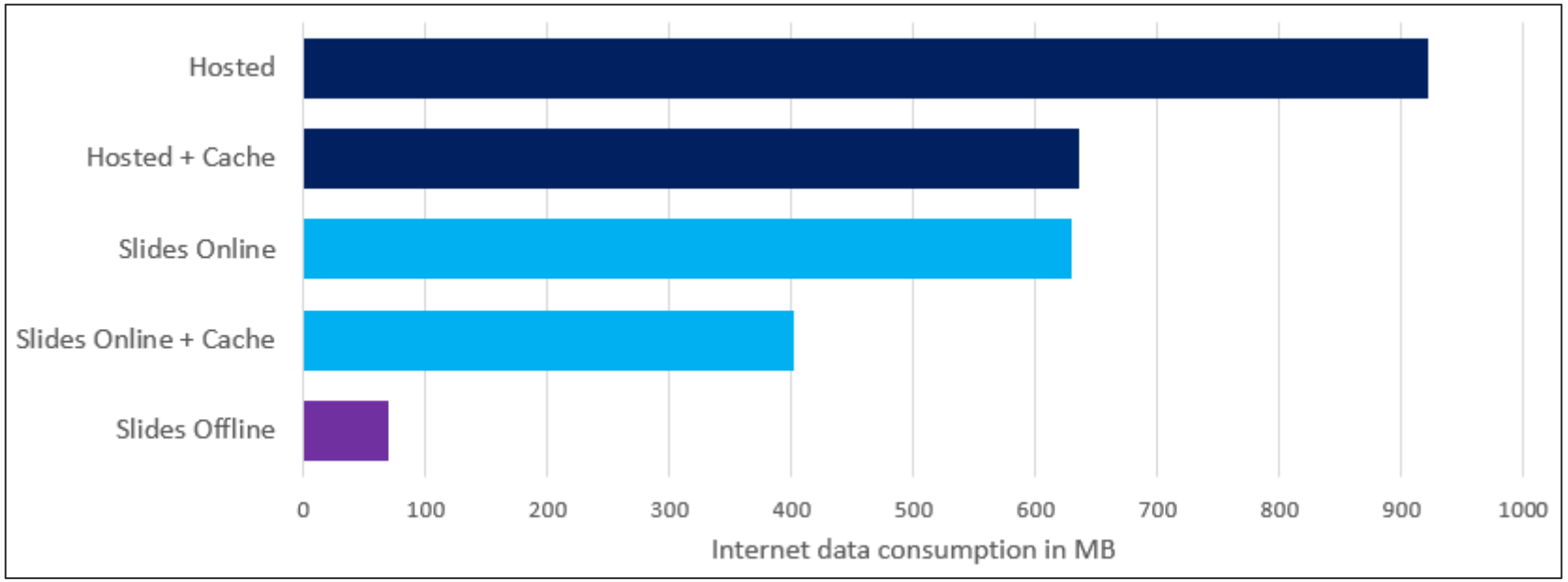
- On the bottom, the performances of ICP on a PC with 4GB DDR3 RAM and a dual-core AMD E-300 1.3GHz CPU

Programming Language	Initialization (s)	Code Execution (s)
Java	59	15.5
Python	59	Immediate
C++	209	0.5 to 8
Processing	Immediate	Immediate
P5	Immediate	Immediate
Standard ML	Immediate	4.6
Javascript	Immediate	Immediate
Typescript	Immediate	0.6
SQL	Immediate	0.5

RESULTS: PERFORMANCES

- Simulated an use case: 12 EC introductory course to programming
  - Average of 25 sets of slides, each containing 40 slides

- Assumption: during a course, each set of slides is reproduced 3 times



RESULTS: LIMITATIONS

Two kind of limitations:

1. Some bugs, possibly caused by incompatibilities with mobile browsers
   - They do not affect the usability of ICP slides, but can compromise their quality

2. Uncertainty about mobile devices characteristics in contexts of scarcity

CONCLUSIONS

Strengths

- Good performances on low-end devices with no internet connection (Internet data usage can be reduced to 0)
- Works on old devices (Android 5 was released at the end of 2014)
- Keeps all the advantages of using an educational technology like ICP
- Possibility to use mobile devices instead of computers is an important inclusivity factor

Limitations

- Some minor bugs due to compatibility issues
- Missing testing in real courses and contexts of scarcity

Department of Information Engineering
and Computer Science

UNIVERSITÀ
DI TRENTO

Interactive Code Playgrounds on Android

Using mobile devices to make educational technologies more accessible in contexts of scarcity

Author: Salvatore Gilles Cassarà

Supervisors: Lorenzo Angeli, Maurizio Marchese