TouchDevelop:
A touch-first IDE for the Web created with TypeScript

Michał Moskal
Researcher
3-018
This was my first computer.
It wanted to be programmed.
These are their first computers

Hi-res displays, hi-powered CPUs, fancy sensors, network.

No keyboard to speak of.
No BASIC prompt.
At Microsoft Research we try to see what’s possible.

Can we program on these things?
Programming the phone – no PC required!

TouchDevelop is a Windows Phone 7 app bringing the joy of programming 8-bit computers to the smartphone age.
Demo: TouchDevelop on the phone
It actually works!

200,000+ downloads
Full social experience
10,000 scripts
Great for education

http://touchdevelop.com/gallery/
Bigger is better.
More is more.
Make it a webapp!
What’s a web app?

What do we have?
152,000 lines of C#
12,000 lines of XAML
125,000 lines of C#
(plus 10,000 lines of tools in C#)
HTML5 & CSS3 instead of XAML

- It’s immediate (just refresh the page)
- It’s popular (just bing around for answers)
- It’s fast (CSS animations, rendering speed)
- It’s simple (e.g., list viewer is just a `<div>`)
- It’s resilient (doesn’t crash on errors)

HTML5 & CSS3 instead of XAML
Now to that JavaScript thing...

It’s immediate (just refresh the page)
It’s popular (just bing around for answers)
It’s **reasonably** fast (with modern JITs)
It’s **not** simple (e.g., 20 ways of doing OO)
I really like types

TypeScript makes for an easier C# to JavaScript transition.

TypeScript – a superset of JavaScript, compiles to idiomatic JS and adds:

- Type annotations and inference
- Syntax for classes, interfaces & modules
- Enables tool support (in VS and web)
- In developer preview
TypeScript ⊆ JavaScript

What goes in JavaScript goes in TypeScript.

For better or worse.

You can skip type annotations or just use :any

You can still do crazy JS stuff
   Even eval() is still there and reflection is super-easy. Great for interpreters/compilers ;-)   

You can copy&paste JS snippets from the web

You can use existing JS libraries
   Type annotations for DOM, WinRT and jQuery are included – you get type-checking there!   
   You can write your own...
namespace Microsoft.TouchDevelop.Runtime {
    public sealed class SpriteRepr : ReferenceRepr
    {
        private GameBoardRepr parent;
        private List<SpringRepr> springs = new List<SpringRepr>();

        public Vector2 ComputeForces(
            Vector4 posSpeed)
        {
            ... }
    }
}

module TDev.RT {
    export class Sprite
    extends RTValue
    {
        private parent: Board;
        private springs: Spring[] = [];

        public computeForces(
            posSpeed: Vector4): Vector2
        {
            ... }
    }
}
foreach (var spring in this.springs) {
    force +=
    spring.ForceOn(this);
}

for (int i = 0; i < touchPts.Count; i++)
{
    var unitNormal = touchPts[i];
    var d =
        Vector2.Dot(unitNormal, force);
    if (d > 0) continue;
    force = unitParallel * d;
}

this.springs.forEach((spring) => {
    force =
    force.add(spring.forceOn(this));
});

for (var i = 0; i < this.touchPts.length; i++)
{
    var unitNormal = this.touchPts[i];
    var d =
        Vector2.dot(unitNormal, force);
    if (d > 0) continue;
    force = unitParallel * d;
}
Demo: TypeScript premiere
Define a class

class Greeter {
    greeting: string;
    constructor (message: string) {
        this.greeting = message;
    }
    greet() {
        echo("Hello, " + this.greeting);
    }
}

var greeter = new Greeter("BUILD");
greeter.greet();
Override a method, with JS flavor

class Greeter {
    constructor (public greeting: string) {} 
    greet(s:string) {
        echo(s + " " + this.greeting)
    }
}

Greeter.prototype.greet = function (s) {
    echo(s + " and " + this.greeting);
};
I was always missing unlink() in HTMLElement

interface HTMLElement {
    unlink(): void;
}

HTMLElement.prototype.unlink = function () {
    if (this.parentNode)
        this.parentNode.removeChild(this);
}
Interfaces are for classes, right?

```javascript
module Cloud1 {
  export function fetch(path:string) { return ""; }
  export function somethingElse() { ... }
}

interface CloudIface {
  fetch(path:string):string;
}

var Cloud:CloudIface = Cloud1;
Cloud1.somethingElse = () => { };
```
The ‘this’ pointer – you *will* get confused

class Greeter {
    constructor (public greeting: string) {} 
    greet() {
        // the difference between function and =>
        [2011, 2012].forEach(function (n) {
            echo(this.greeting + " " + n);
        });
    }
}
There is no block scope

```javascript
function echo(s) => console.log(s);
var fruits = ["pears", "oranges", "bananas"];
var callbacks = [];
var i;
for (i = 0; i < fruits.length; ++i) {
    callbacks.push(() => echo(fruits[i]))
}
callbacks.forEach((cb) => cb());

// prints: bananas, bananas, bananas

// there is just one copy of 'fruit' variable; the loop/block doesn't introduce a scope!

callbacks = [];
for (i = 0; i < fruits.length; ++i) {
    var fruit = fruits[i];
    callbacks.push(() => echo(fruit))
}
callbacks.forEach((cb) => cb());

// prints: pears, oranges, bananas

// if you really need a scope inside of a loop - create a function and call it immediately:
callbacks = [];
for (i = 0; i < fruits.length; ++i) {
```
```
callbacks = [];
fruits.forEach((fruit) => {
  callbacks.push(() => echo(fruit))
})
callbacks.forEach((cb) => cb());
// prints: pears, oranges, bananas
// as expected; it's also shorter

// speaking of short - this also works:
// (or should; right now it seems to crash the compiler)
callbacks = fruits.map((fruit) => {
  return () => echo(fruit) })
callbacks.forEach((cb) => cb());
// prints: pears, oranges, bananas
Don’t use `for(each)`

```javascript
// no no no
for (var v in obj) {
    ...
}

// yes yes yes
Object.keys(obj).forEach((v) => {
    ...
});
```
The high-order bit.

Smoother migration from C# to crazy JavaScript.

Avoids lots of runtime errors
(even though there are no 100% guarantees)

IntelliSense is useful

Usually quite functional style
WP7 App:
152,000 lines of C#
12,000 lines of XAML

Web App:
47,000 lines of TypeScript
3,500 lines of CSS
79 lines of HTML

(but we’re not quite done yet)
Demo: TouchDevelop on the Web
Take aways

Big apps? TypeScript is going to help you!

Types are good.

TouchDevelop is cool :-)
Resources

touchdevelop.com/app

typescriptlang.org

Questions?
Color palette

//build/ template colors

Lead and accent colors have been formatted into this template. In general, use these 4 colors for all slides.

All Windows brand colors
### Table

<table>
<thead>
<tr>
<th>Component</th>
<th>C#</th>
<th>XAML</th>
<th>TypeScript</th>
<th>CSS+HTML</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cloud</td>
<td>125,000</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Client</td>
<td>152,000</td>
<td>12,000</td>
<td>47,000</td>
<td>3,500</td>
</tr>
<tr>
<td>Tools</td>
<td>10,000</td>
<td>300</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

This is a content box, always 24pts and sentence case
TouchDevelop is a complex webapp

Not a replacement for professional development tools.

But it does involve a compiler, intellisense engine, lots of UI code for the editor, etc.
Things I didn’t know existed

The web development has changed in the last couple years
Client-side storage

localStorage:
  easy, sync, cross-browser, 5MB max
  localStorage[“key”] = “value”;

IndexedDB & WebSQL
  async, often 50MB max
  IndexedDB – W3C, IE, Firefox
  WebSQL – Webkit (Safari, Chrome)
board

- we have a board here; it doesn't do anything alone, insert post to wall if you want to display it or just delete it
- high box := board -> create text((150, 20, 40), "High" * [high score])

if orientation = 1 then
- [score box := set angle(90)]
- [score box := set pos(board -> width, 100)]
- [high box := set angle(90)]
- [high box := set pos(board -> width, board -> height - 150)]
else do nothing
- [high box := set friction(1)]

if orientation = 3 then
- [score box := set angle(-90)]
- [score box := set pos(0, board -> height - 100)]
- [high box := set angle(-90)]
- [high box := set pos(0, 150)]
```javascript
34897:16 | function recompileScript() {  
34898:18 | var cs = TDev.AST.Compiler.getCompiledScript(TDev.Script, false, this, withTracing, this, currentScript);  
34899:18 | this.currentScript.initFrom(cs);  
34900:19 | }  
34901:19 |  
34902:19 | function runAction(a, args) {  
34903:21 | if (typeof args !== "undefined") { args = null; }  
34904:21 | var this = this;  
34905:21 | this.recompileScript();  
34906:21 | var missing = this.currentScript.compiled.missingAPIs;  
34907:21 | if (this.complainedAboutMissingAPIS && missing.length > 0) {  
34908:23 | TDev.ModalDialog.ask("The following APIs are not implemented on the current device: 
34909:23 | this.complainedAboutMissingAPIS = true;  
34910:23 | this.runAction(a, args);  
34911:23 | };  
34912:23 | return;  
34913:23 | }  
34914:26 | rt = this.currentScript;  
34915:26 | rt.setHost(this.host);  
34916:26 | this.host.showWall();  
34917:26 | this.initPageStack();  
34918:26 | TDev.SIZEManager.applySizes();  
34919:26 | if (a instanceof TDev.AST.Action) {  
34920:28 | a = null;  
34921:28 | }  
34922:28 | if (a == null) {  
34923:30 | rt.postError("can't run this");  
34924:30 | }  
34925:30 | else {  
34926:32 | isMain = a == TDev.Script.mainAction;  
34927:32 | function setHash() {  
34928:34 | if (isNew) { 
34929:36 | _this.historyMgr.setHash("run" + TDev.Script.localGUID, TDev.Script.getName());  
34930:36 | } else { 
34931:38 | _this.historyMgr.setHash("run-action:" + TDev.Script.localGUID + ":" + a.stableName);  
34932:38 | }  
34933:40 | setHash();  
34934:40 | var name = a.stableName;  
34935:40 | function runAction() {  
34936:42 | TDev.Ticks.tick(TDev.Ticks.coreRun, _this.currentPublicId);  
34937:42 | fn = rt.compiledActionsByName[name];  
34938:42 | rt.run(fn, a, args);  
34939:42 | }  
34940:42 | this.runAction = function() {  
34941:44 | TDev.Ticks.tick(TDev.Ticks.coreRun);  
34942:46 | }  
```

CSS3 transitions and animations

```css
@keyframes showDn {
  0% {
    opacity: 0;
    transform: translate(0, 2em);
  }
  100% {
    opacity: 1;
    transform: none;
  }
}

.show {
  animation: showDn 0.3s ease-out;
}
```
And browser prefixes :

```css
@-webkit-keyframes showDn {
  0% {
    opacity: 0;
    -webkit-transform: translate(0, 2em);
  }
  100% {
    opacity: 1;
    -webkit-transform: none;
  }
}

.show {
  -webkit-animation: showDn 0.3s ease-out;
}
```
CSS3
transitions and
data-*

<button data-active="no">
  Click here!
</button>

.button[data-active^="yes"] {
  font-size: 1.5em;
  transition: font-size 0.5s;
}
Promises

```javascript
readLineAsync()
  .then((s:string) =>
    db.getRecord(s))
  .then((r:MyRecord) => {
    ui.showDialog(r.foobar)
  }).done();

Promise.join({
  db: getDbAsync(),
  id: userIdAsync()
}).then((ctx) =>
  ctx.db.getAsync(ctx.id))
```
Pains of single-origin policy

By default www.foo.com can only make HTTP requests to www.foo.com.

Great for cookies, horrible for general programming.

Webserver can allow other domains via CORS.
We didn’t yet get there...

WebSockets: let you keep telnet-like connection open.

WebWorkers: let you run stuff in background and communicate via massage-and resource-passing
TypeScript wish list

Nobody’s perfect.

Direct debugger support
Faster IntelliSense
Generics
Stricter strict mode
   “var” scoping, “this” typing, stricter “any"
Conditional compilation
Algebraic data-types
http://touchdevelop.com/app/
Bigger screens.
More devices.
Take it to the Web!