Cross Platform Development

on iOS with Objective-C and PhoneGap, HTML5 and Javascript

Cassian LUP

Lead Developer

SC **Netzinkubator** SRL

netzinkubator

Current State

Mobile Development, Current State.

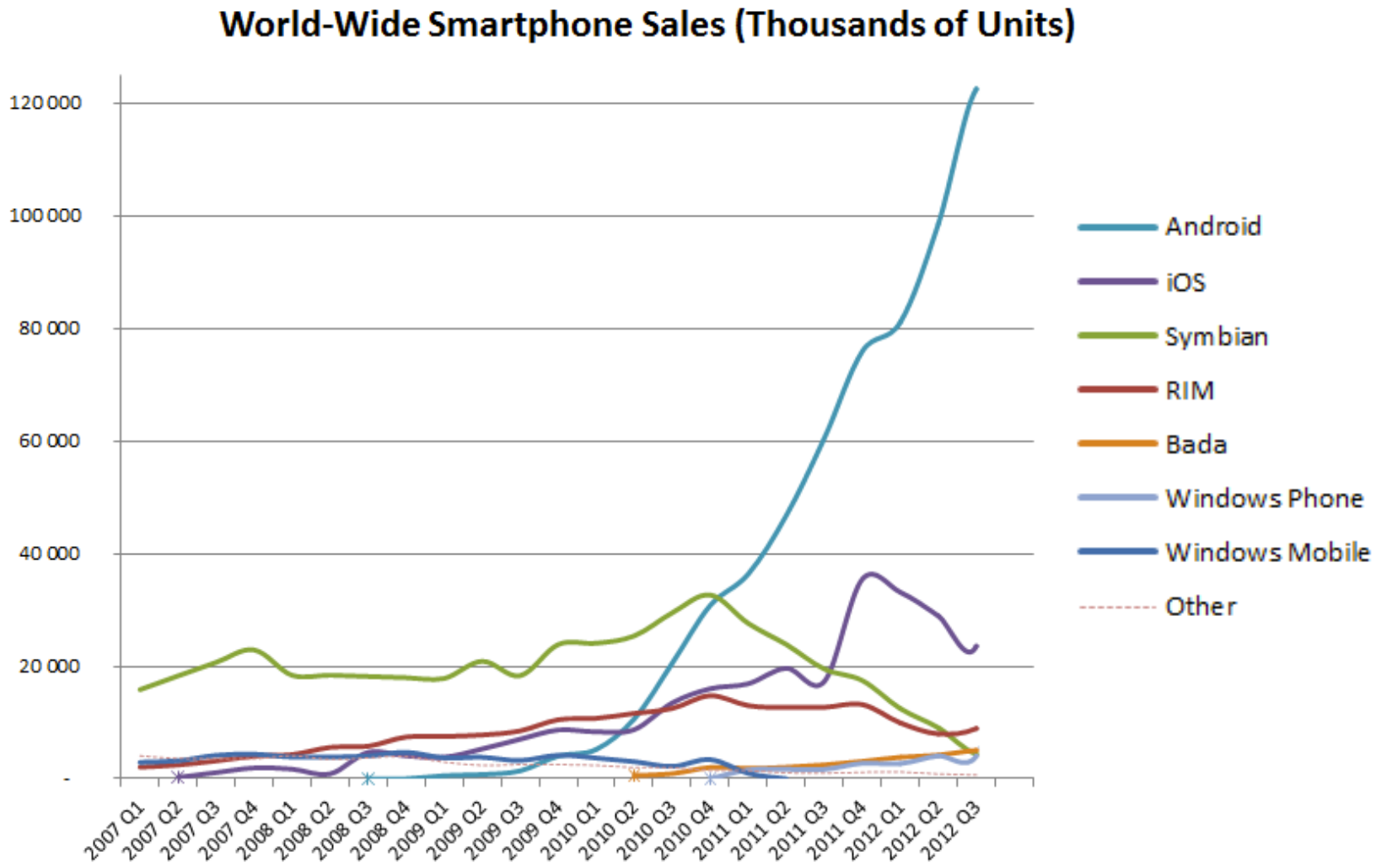
iOS

Windows 8 Phone

BlackBerry

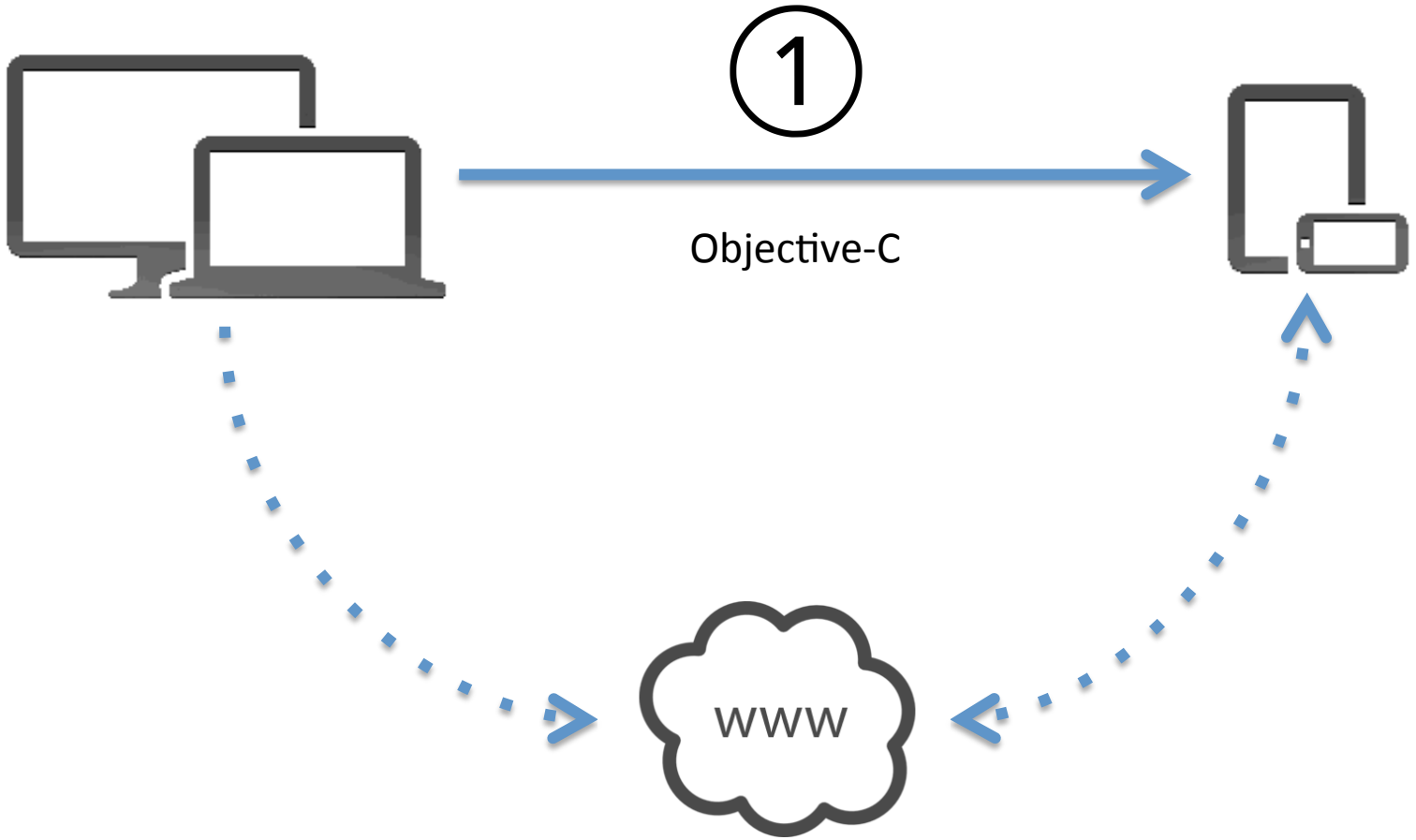
?

Mobile Development, Current State.

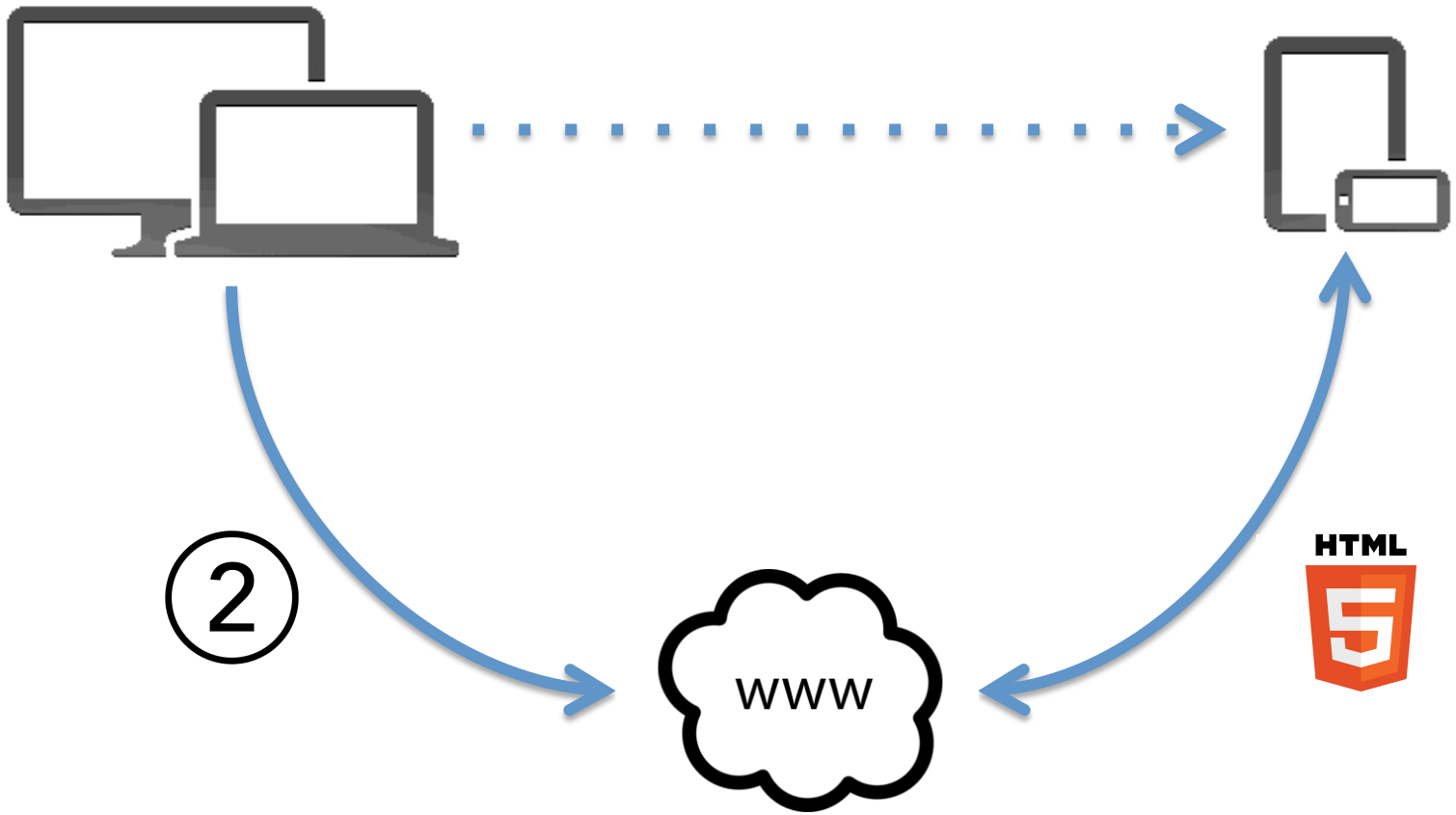


Deployment

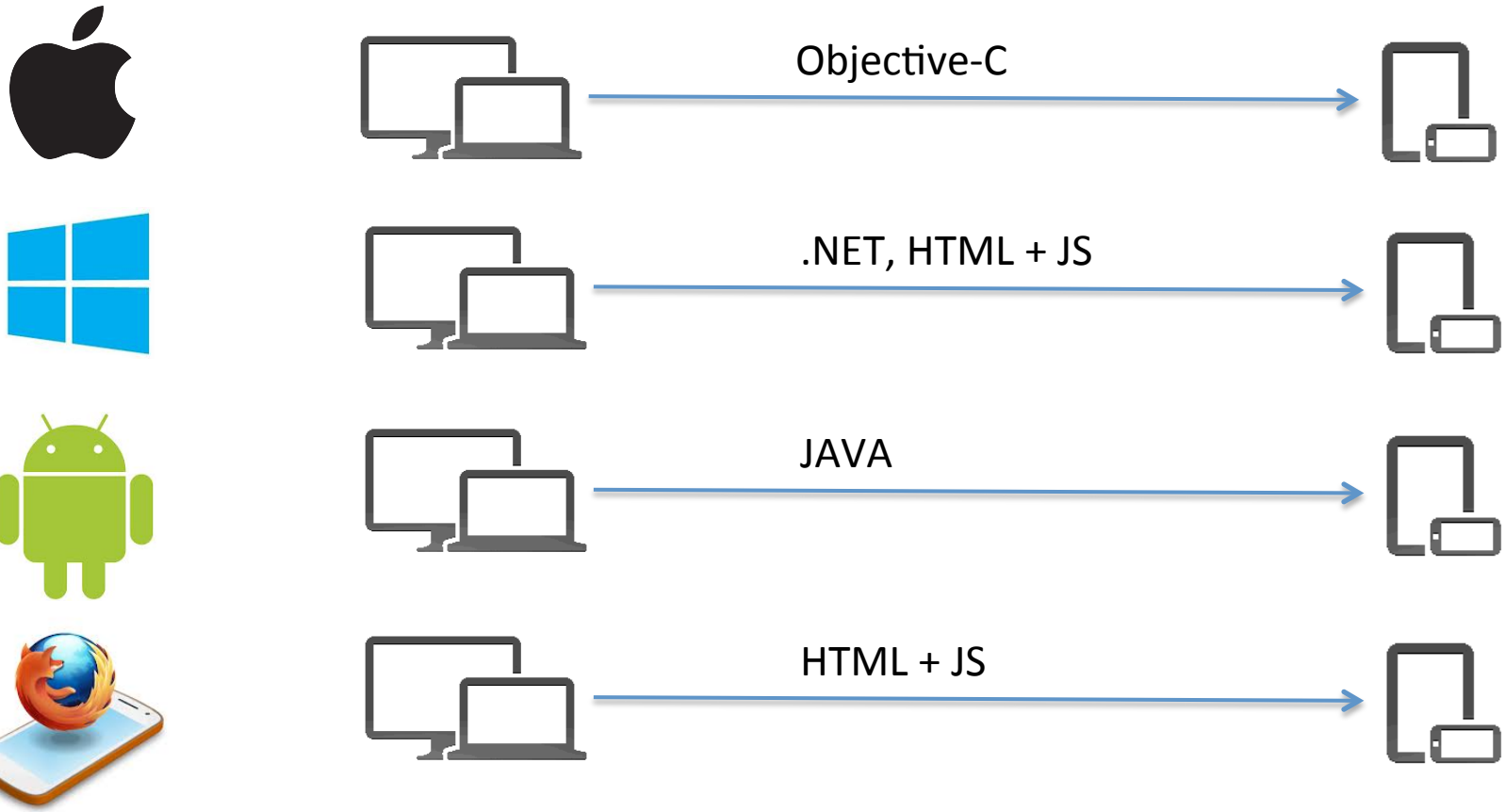
Mobile Development, iOS, Native.



Mobile Development, iOS, WWW.



Mobile Development, Native.

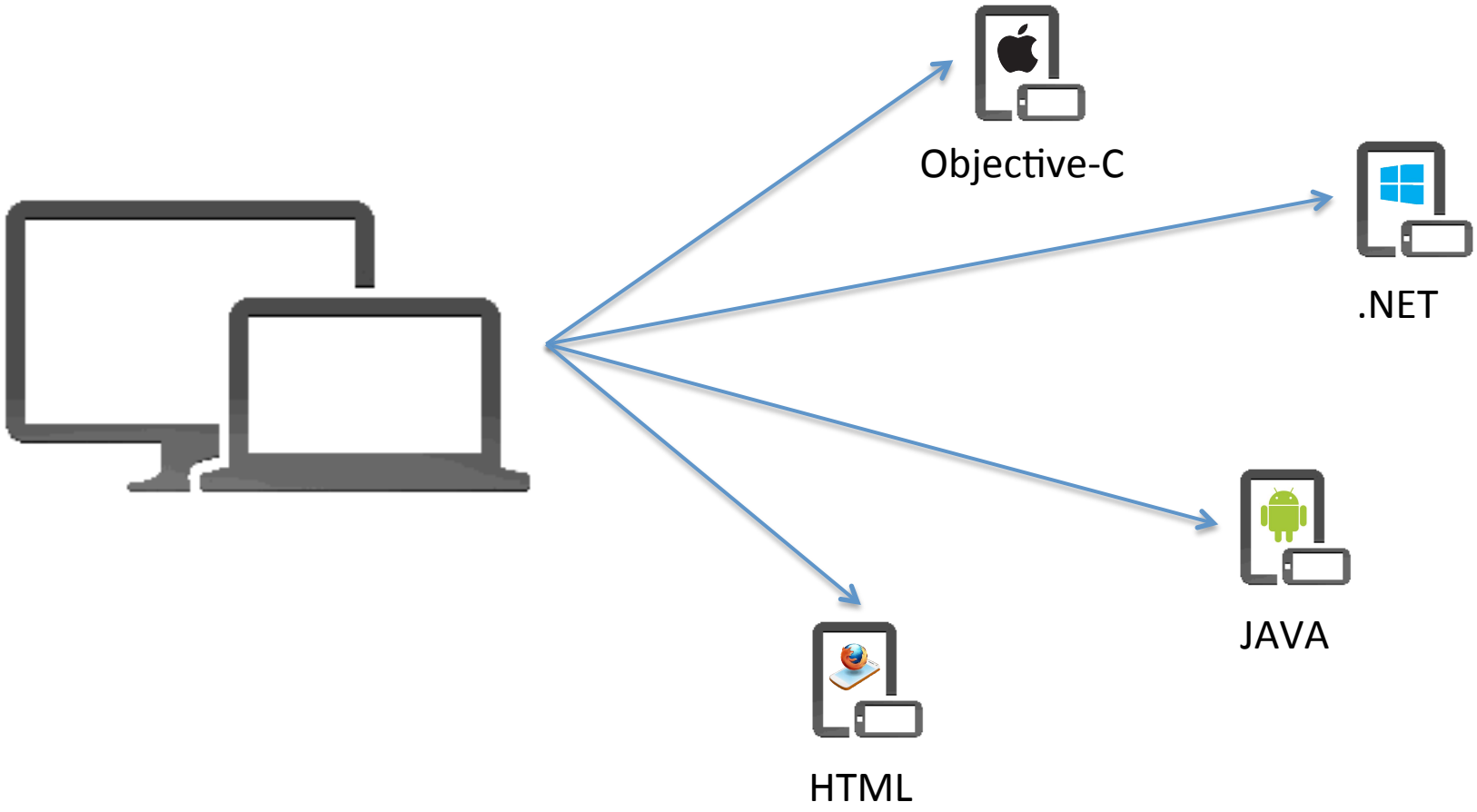


...

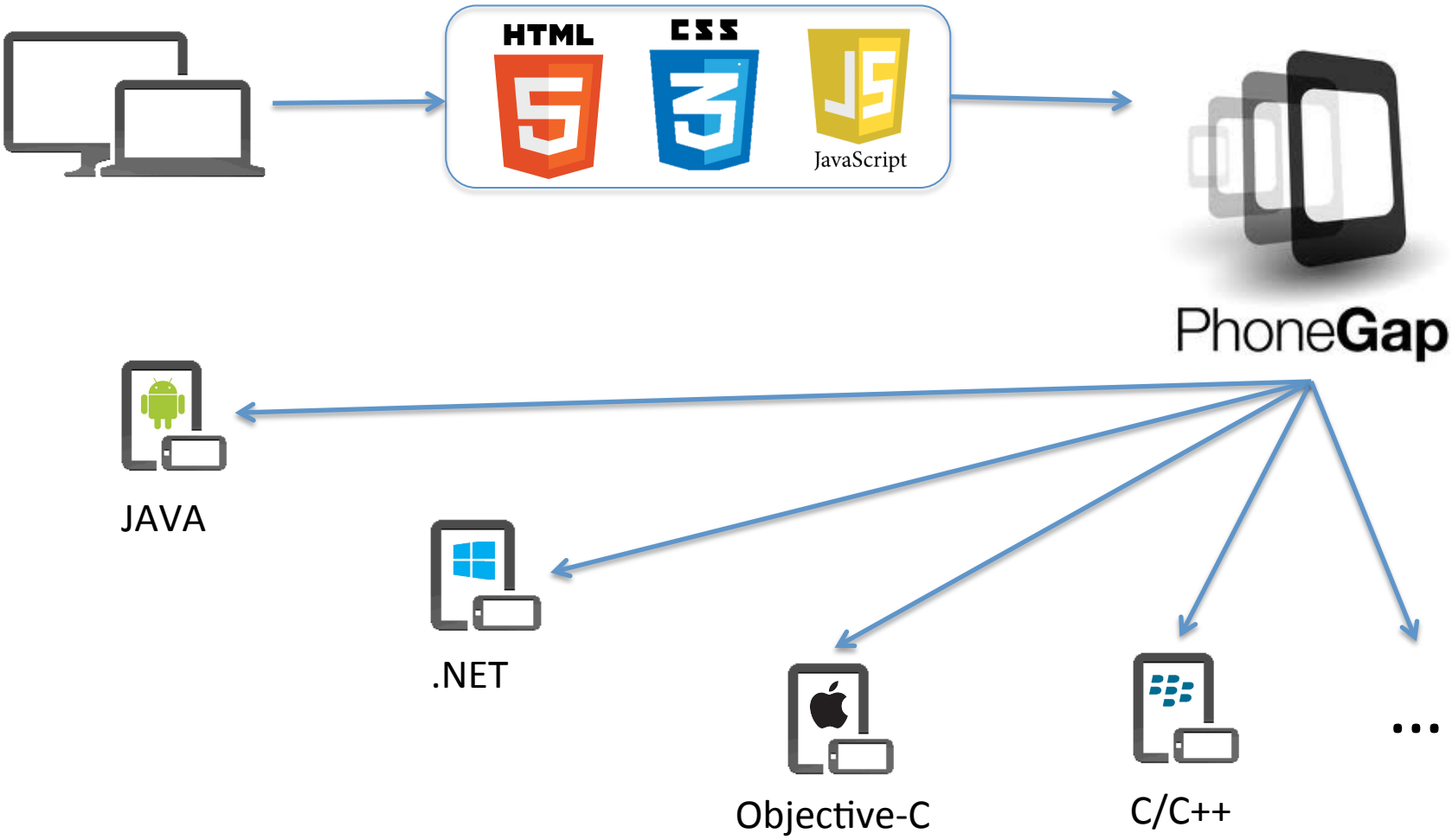
Mobile Development

Cross-Platform

Mobile Development, Cross-Platform.



Mobile Development, PhoneGap.



Mobile Development, Approaches.

Cross-Platform

1. Cost
2. Learning Curve
3. Service vs. App

Native

1. Native look
2. Computing power
3. App vs. Service

Mobile Development

Demo

Mobile Development, Code.

Code Example:

Reading the accelerometer using Objective-C.

Mobile Development, Obj-C.

```
if ([self.motionManager isAccelerometerAvailable])
{
    AccDrawView * av = (AccDrawView *)[self view];
    [av setLastX:100];
    [av setLastY:100];

    NSOperationQueue *queue = [[NSOperationQueue alloc] init];
    [self.motionManager
     startAccelerometerUpdatesToQueue:queue
     withHandler:^(CMAccelerometerData *accelerometerData, NSError *error) {

         AccDrawView * av = (AccDrawView *)[self view];

         float xShift = [av xShift] * 0.8 + accelerometerData.acceleration.x * 0.8;
         float yShift = [av yShift] * 0.8 - accelerometerData.acceleration.y * 0.8;

         [av setXShift:xShift];
         [av setYShift:yShift];

         // dispatch to main thread where our view is
         dispatch_async(dispatch_get_main_queue(), ^{
             [av setNeedsDisplay];
         });
     }];
} else {
    NSLog(@"Accelerometer is not available.");
}
```

Mobile Development, Obj-C.

```
if ([self.motionManager isAccelerometerAvailable])
{
    AccDrawView * av = (AccDrawView *)[self view];
    [av setLastX:100];
    [av setLastY:100];

    NSOperationQueue *queue = [[NSOperationQueue alloc] init];
    [self.motionManager
     startAccelerometerUpdatesToQueue:queue
     withHandler:^(CMAccelerometerData *accelerometerData, NSError *error) {

         AccDrawView * av = (AccDrawView *)[self view];

         float xShift = [av xShift] * 0.8 + accelerometerData.acceleration.x * 0.8;
         float yShift = [av yShift] * 0.8 - accelerometerData.acceleration.y * 0.8;

         [av setXShift:xShift];
         [av setYShift:yShift];

         // dispatch to main thread where our view is
         dispatch_async(dispatch_get_main_queue(), ^{
             [av setNeedsDisplay];
         });
     }];
} else {
    NSLog(@"Accelerometer is not available.");
}
```

Mobile Development, Obj-C.

```
if ([self.motionManager isAccelerometerAvailable])
{
    AccDrawView * av = (AccDrawView *)[self view];
    [av setLastX:100];
    [av setLastY:100];

    NSOperationQueue *queue = [[NSOperationQueue alloc] init];
    [self.motionManager
     startAccelerometerUpdatesToQueue:queue
     withHandler:^(CMAccelerometerData *accelerometerData, NSError *error) {

         AccDrawView * av = (AccDrawView *)[self view];

         float xShift = [av xShift] * 0.8 + accelerometerData.acceleration.x * 0.8;
         float yShift = [av yShift] * 0.8 - accelerometerData.acceleration.y * 0.8;

         [av setXShift:xShift];
         [av setYShift:yShift];

         // dispatch to main thread where our view is
         dispatch_async(dispatch_get_main_queue(), ^{
             [av setNeedsDisplay];
         });
     }];
} else {
    NSLog(@"Accelerometer is not available.");
}
```

Mobile Development, Obj-C.

```
if ([self.motionManager isAccelerometerAvailable])
{
    AccDrawView * av = (AccDrawView *)[self view];
    [av setLastX:100];
    [av setLastY:100];

    NSOperationQueue *queue = [[NSOperationQueue alloc] init];
    [self.motionManager
     startAccelerometerUpdatesToQueue:queue
     withHandler:^(CMAccelerometerData *accelerometerData, NSError *error) {

         AccDrawView * av = (AccDrawView *)[self view];

         float xShift = [av xShift] * 0.8 + accelerometerData.acceleration.x * 0.8;
         float yShift = [av yShift] * 0.8 - accelerometerData.acceleration.y * 0.8;

         [av setXShift:xShift];
         [av setYShift:yShift];

         // dispatch to main thread where our view is
         dispatch_async(dispatch_get_main_queue(), ^{
             [av setNeedsDisplay];
         });
     }];
} else {
    NSLog(@"Accelerometer is not available.");
}
```

Mobile Development, Obj-C.

```
if ([self.motionManager isAccelerometerAvailable])
{
    AccDrawView * av = (AccDrawView *)[self view];
    [av setLastX:100];
    [av setLastY:100];

    NSOperationQueue *queue = [[NSOperationQueue alloc] init];
    [self.motionManager
     startAccelerometerUpdatesToQueue:queue
     withHandler:^(CMAccelerometerData *accelerometerData, NSError *error) {

         AccDrawView * av = (AccDrawView *)[self view];

         float xShift = [av xShift] * 0.8 + accelerometerData.acceleration.x * 0.8;
         float yShift = [av yShift] * 0.8 - accelerometerData.acceleration.y * 0.8;

         [av setXShift:xShift];
         [av setYShift:yShift];

         // dispatch to main thread where our view is
         dispatch_async(dispatch_get_main_queue(), ^{
             [av setNeedsDisplay];
         });
     }];
} else {
    NSLog(@"Accelerometer is not available.");
}
```

Mobile Development, Obj-C.

```
if ([self.motionManager isAccelerometerAvailable])
{
    AccDrawView * av = (AccDrawView *)[self view];
    [av setLastX:100];
    [av setLastY:100];

    NSOperationQueue *queue = [[NSOperationQueue alloc] init];
    [self.motionManager
     startAccelerometerUpdatesToQueue:queue
     withHandler:^(CMAccelerometerData *accelerometerData, NSError *error) {

         AccDrawView * av = (AccDrawView *)[self view];

         float xShift = [av xShift] * 0.8 + accelerometerData.acceleration.x * 0.8;
         float yShift = [av yShift] * 0.8 - accelerometerData.acceleration.y * 0.8;

         [av setXShift:xShift];
         [av setYShift:yShift];

         // dispatch to main thread where our view is
         dispatch_async(dispatch_get_main_queue(), ^{
             [av setNeedsDisplay];
         });
     }];
} else {
    NSLog(@"Accelerometer is not available.");
}
```

Mobile Development, Obj-C.

```
if ([self.motionManager isAccelerometerAvailable])
{
    AccDrawView * av = (AccDrawView *)[self view];
    [av setLastX:100];
    [av setLastY:100];

    NSOperationQueue *queue = [[NSOperationQueue alloc] init];
    [self.motionManager
     startAccelerometerUpdatesToQueue:queue
     withHandler:^(CMAccelerometerData *accelerometerData, NSError *error) {

         AccDrawView * av = (AccDrawView *)[self view];

         float xShift = [av xShift] * 0.8 + accelerometerData.acceleration.x * 0.8;
         float yShift = [av yShift] * 0.8 - accelerometerData.acceleration.y * 0.8;

         [av setXShift:xShift];
         [av setYShift:yShift];

         // dispatch to main thread where our view is
         dispatch_async(dispatch_get_main_queue(), ^{
             [av setNeedsDisplay];
         });
     }];
} else {
    NSLog(@"Accelerometer is not available.");
}
```

Mobile Development, Code.

Code Example:

Reading the accelerometer using JavaScript.

Mobile Development, JS (continued).

```
navigator.accelerometer.watchAcceleration(
    // the success function
    function(acc) {
        if(sketcher != null) {
            sketcher.changePosition({x: -acc.x, y: acc.y});
            sketcher.draw();
        }
    },
    // the error handler
    function() { console.log("error"); },
    // refresh frequency
    { frequency: 40 /* ms */ }
);
```

Mobile Development, JS (continued).

```
navigator.accelerometer.watchAcceleration
    // the success function
    function(acc) {
        if(sketcher != null) {
            sketcher.changePosition({x: -acc.x, y: acc.y});
            sketcher.draw();
        }
    },
    // the error handler
    function() { console.log("error"); },
    // refresh frequency
    { frequency: 40 /* ms */ }
);
```

Mobile Development, JS (continued).

```
navigator.accelerometer.watchAcceleration(
    // the success function
    function(acc) {
        if(sketcher != null) {
            sketcher.changePosition({x: -acc.x, y: acc.y});
            sketcher.draw();
        }
    },
    // the error handler
    function() { console.log("error"); },
    // refresh frequency
    { frequency: 40 /* ms */ }
);
```

Mobile Development, JS (continued).

```
navigator.accelerometer.watchAcceleration(
    // the success function
    function(acc) {
        if(sketcher != null) {
            sketcher.changePosition({x: -acc.x, y: acc.y});
            sketcher.draw();
        }
    },
    // the error handler
    function() { console.log("error"); },
    // refresh frequency
    { frequency: 40 /* ms */ }
);
```

Code Example:

Drawing in Objective-C.

Mobile Development, Obj-C.

```
float newX = lastX + xShift;
float newY = lastY + yShift;

[pointsArray addObject:[NSValue valueWithCGPoint:CGPointMake(newX, newY)]];

CGContextSetLineWidth(c, 10);
CGContextSetLineCap(c, kCGLineCapRound);
CGContextSetLineJoin(c, kCGLineJoinRound);
CGContextSetStrokeColorWithColor(c, [UIColor grayColor].CGColor);

float prevX, prevY;
prevX = prevY = 100;

CGContextBeginPath(c);
CGContextMoveToPoint(c, 100, 100);

for(NSValue *v in pointsArray) {
    CGPoint p = v.CGPointValue;

    CGContextAddLineToPoint(c, p.x, p.y);
}

CGContextStrokePath(c);
```

Mobile Development, Obj-C.

```
float newX = lastX + xShift;
float newY = lastY + yShift;

[pointsArray addObject:[NSValue valueWithCGPoint:CGPointMake(newX, newY)]];

CGContextSetLineWidth(c, 10);
CGContextSetLineCap(c, kCGLineCapRound);
CGContextSetLineJoin(c, kCGLineJoinRound);
CGContextSetStrokeColorWithColor(c, [UIColor grayColor].CGColor);

float prevX, prevY;
prevX = prevY = 100;

CGContextBeginPath(c);
CGContextMoveToPoint(c, 100, 100);

for(NSValue *v in pointsArray) {
    CGPoint p = v.CGPointValue;

    CGContextAddLineToPoint(c, p.x, p.y);
}

CGContextStrokePath(c);
```

Mobile Development, Obj-C.

```
float newX = lastX + xShift;
float newY = lastY + yShift;

[pointsArray addObject:[NSValue valueWithCGPoint:CGPointMake(newX, newY)]];

CGContextSetLineWidth(c, 10);
CGContextSetLineCap(c, kCGLineCapRound);
CGContextSetLineJoin(c, kCGLineJoinRound);
CGContextSetStrokeColorWithColor(c, [UIColor grayColor].CGColor);

float prevX, prevY;
prevX = prevY = 100;

CGContextBeginPath(c);
CGContextMoveToPoint(c, 100, 100);

for(NSValue *v in pointsArray) {
    CGPoint p = v.CGPointValue;

    CGContextAddLineToPoint(c, p.x, p.y);
}

CGContextStrokePath(c);
```

Mobile Development, Obj-C.

```
float newX = lastX + xShift;
float newY = lastY + yShift;

[pointsArray addObject:[NSValue valueWithCGPoint:CGPointMake(newX, newY)]];

CGContextSetLineWidth(c, 10);
CGContextSetLineCap(c, kCGLineCapRound);
CGContextSetLineJoin(c, kCGLineJoinRound);
CGContextSetStrokeColorWithColor(c, [UIColor grayColor].CGColor);

float prevX, prevY;
prevX = prevY = 100;

CGContextBeginPath(c);
CGContextMoveToPoint(c, 100, 100);

for(NSValue *v in pointsArray) {
    CGPoint p = v.CGPointValue;

    CGContextAddLineToPoint(c, p.x, p.y);
}

CGContextStrokePath(c);
```

Mobile Development, Obj-C.

```
float newX = lastX + xShift;
float newY = lastY + yShift;

[pointsArray addObject:[NSValue valueWithCGPoint:CGPointMake(newX, newY)]];

CGContextSetLineWidth(c, 10);
CGContextSetLineCap(c, kCGLineCapRound);
CGContextSetLineJoin(c, kCGLineJoinRound);
CGContextSetStrokeColorWithColor(c, [UIColor grayColor].CGColor);

float prevX, prevY;
prevX = prevY = 100;

CGContextBeginPath(c);
CGContextMoveToPoint(c, 100, 100);

for(NSValue *v in pointsArray) {
    CGPoint p = v.CGPointValue;

    CGContextAddLineToPoint(c, p.x, p.y);
}

CGContextStrokePath(c);
```

Mobile Development, Code.

Code Example:

Drawing in JavaScript.

Mobile Development, JS.

```
Sketcher.prototype.draw = function() {
    //invoke HTML5 canvas beginPath() method
    this.ctx.beginPath();

    //define coordinates for each intermediary line
    for(i=0; i < this.options.intermediary_points - 1; i++) {
        this.ctx.lineTo(this.xi[i], this.yi[i]);
    }

    //define final line coordinates
    this.ctx.lineTo(this.x,this.y);

    //set drawing style
    this.ctx.lineCap = "round";

    //perform drawing of defined path
    this.ctx.stroke();
};
```

Mobile Development, JS.

```
Sketcher.prototype.draw = function() {
    //invoke HTML5 canvas beginPath() method
    this.ctx.beginPath();

    //define coordinates for each intermediary line
    for(i=0; i < this.options.intermediary_points - 1; i++) {
        this.ctx.lineTo(this.xi[i], this.yi[i]);
    }

    //define final line coordinates
    this.ctx.lineTo(this.x,this.y);

    //set drawing style
    this.ctx.lineCap = "round";

    //perform drawing of defined path
    this.ctx.stroke();
};
```

Mobile Development, JS.

```
Sketcher.prototype.draw = function() {
    //invoke HTML5 canvas beginPath() method
    this.ctx.beginPath();

    //define coordinates for each intermediary line
    for(i=0; i < this.options.intermediary_points - 1; i++) {
        this.ctx.lineTo(this.xi[i], this.yi[i]);
    }

    //define final line coordinates
    this.ctx.lineTo(this.x,this.y);

    //set drawing style
    this.ctx.lineCap = "round";

    //perform drawing of defined path
    this.ctx.stroke();
};
```

Mobile Development, JS.

```
Sketcher.prototype.draw = function() {
    //invoke HTML5 canvas beginPath() method
    this.ctx.beginPath();

    //define coordinates for each intermediary line
    for(i=0; i < this.options.intermediary_points - 1; i++) {
        this.ctx.lineTo(this.xi[i], this.yi[i]);
    }

    //define final line coordinates
    this.ctx.lineTo(this.x,this.y);

    //set drawing style
    this.ctx.lineCap = "round";

    //perform drawing of defined path
    this.ctx.stroke();
};
```

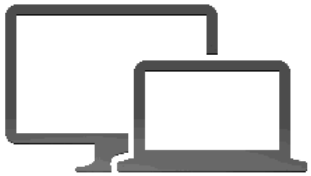
Mobile Development, JS.

```
Sketcher.prototype.draw = function() {
    //invoke HTML5 canvas beginPath() method
    this.ctx.beginPath();

    //define coordinates for each intermediary line
    for(i=0; i < this.options.intermediary_points - 1; i++) {
        this.ctx.lineTo(this.xi[i], this.yi[i]);
    }

    //define final line coordinates
    this.ctx.lineTo(this.x,this.y);

    //set drawing style
    this.ctx.lineCap = "round";

    //perform drawing of defined path
    this.ctx.stroke();
};
```

Mobile Development, JS.

```
Sketcher.prototype.draw = function() {
    //invoke HTML5 canvas beginPath() method
    this.ctx.beginPath();

    //define coordinates for each intermediary line
    for(i=0; i < this.options.intermediary_points - 1; i++) {
        this.ctx.lineTo(this.xi[i], this.yi[i]);
    }

    //define final line coordinates
    this.ctx.lineTo(this.x,this.y);

    //set drawing style
    this.ctx.lineCap = "round";

    //perform drawing of defined path
    this.ctx.stroke();
};
```

Mobile Development, Proof.



Similarities

Mobile Development,
Cross Platform Myths.

Myths

Thank you.