



Performance

# Lets dive into Performance issues.

- Everything in JavaScript defaults to being on the same thread. Too much work on main thread
- Android nested layouts
- Functions and objects defined in loops
- Statements like debugger, eval, with.
- How to access Native Engine information
- Object class modifications.

# Threading

- In JavaScript by default everything runs on the main thread. This design has both pro's and cons
- Pros: Easy access to anything dealing with GUI
- Bad: Jank when doing lengthy work
- Potential Solutions: Worker thread

# Nested Layouts

- Primarily a Android issue; but does impact iOS.
- GridLayout is the King of simplifying layouts. You can eliminate virtually all stack layouts with one GridLayout.

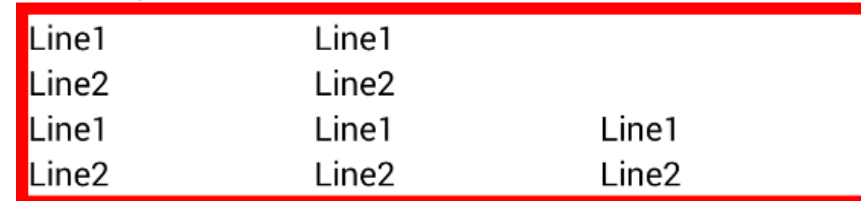
## <StackLayout>

```
<Label width="300" text="StackLayout w/ Borders"/>
<StackLayout class="bred">
  <StackLayout class="bgreen" orientation="horizontal">
    <StackLayout class="bbblue">
      <Label text="Line1"/>
      <Label text="Line2"/>
    </StackLayout>
    <StackLayout class="borange">
      <Label text="Line1"/>
      <Label text="Line2"/>
    </StackLayout>
  </StackLayout>
  <StackLayout class="bpurple" orientation="horizontal">
    <StackLayout class="bbblue">
      <Label text="Line1"/>
      <Label text="Line2"/>
    </StackLayout>
    <StackLayout class="borange">
      <Label text="Line1"/>
      <Label text="Line2"/>
    </StackLayout>
    <StackLayout class="bgreen">
      <Label text="Line1"/>
      <Label text="Line2"/>
    </StackLayout>
  </StackLayout>
</StackLayout>
```

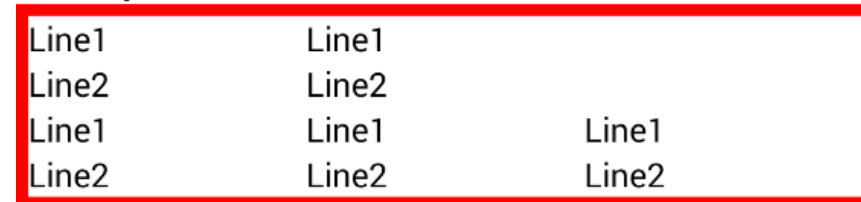
## StackLayout w/ Borders



## StackLayout w/o borders



## Grid Layout



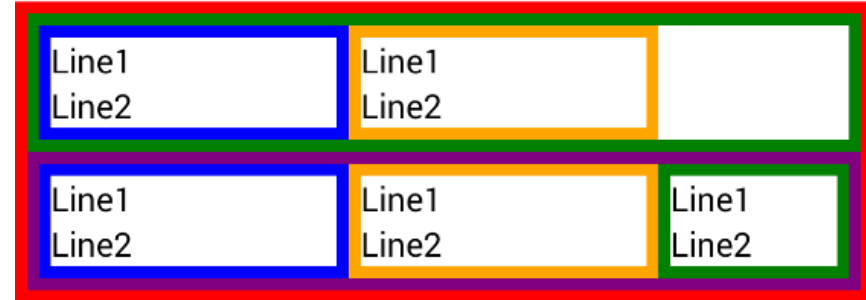
```

<GridLayout class="bred" rows="auto,auto,auto,auto"
            columns="*,*,*,*,*,*">
    <Label text="Line1" row="0" colSpan="2"/>
    <Label text="Line2" row="1" colSpan="2"/>
    <Label text="Line1" row="0" col="2" colSpan="2"/>
    <Label text="Line2" row="1" col="2" colSpan="2"/>

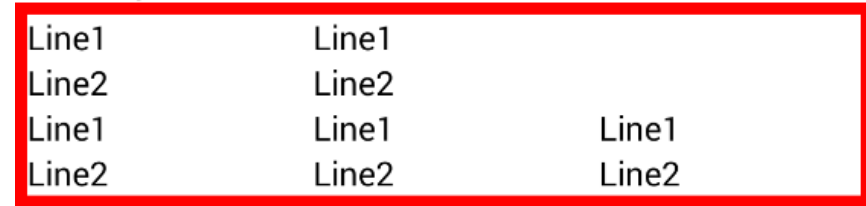
    <Label text="Line1" row="2" colSpan="2"/>
    <Label text="Line2" row="3" colSpan="2"/>
    <Label text="Line1" row="2" col="2" colSpan="2"/>
    <Label text="Line2" row="3" col="2" colSpan="2"/>
    <Label text="Line1" row="2" col="4" colSpan="2"/>
    <Label text="Line2" row="3" col="4" colSpan="2"/>
</GridLayout>

```

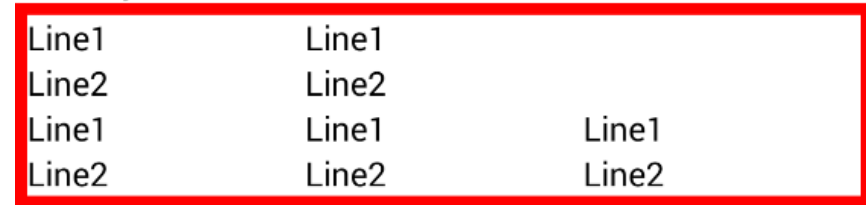
### StackLayout w/ Borders



### StackLayout w/o borders



### Grid Layout



# Loops

- Beware of function and memory allocations inside of loops.

You see this as a standard pattern.

```
function loopFunction () {  
    var val = 0;  
    for (var i=0;i<100000;i++) {  
        doSomething( () => { val++; } );  
    }  
}
```

# Loops

- Proper way

```
function loops() {  
    var val = 0, fun = function() {  
        val++;  
    };  
    for (var i=0;i<100000;i++) {  
        doSomething(fun);  
    }  
}
```



# Data Processing

- Anything that does any amount of work; if possible move to your worker thread.
- Gathering and saving data from/to your database is a prime example.

# Enabling Optimization Testing

- `npm i v8-natives --save`
- Edit your `app/package.json`
- Add `--allow-native-syntax` to `v8Flags`

```
{
  "android": {
    "v8Flags": "--expose_gc --allow-natives-syntax"
  },
  "main": "app.js",
  "name": "tns-template-hello-world",
  "version": "3.2.0"
}
```

# V8 Natives

- Docs: <https://github.com/NathanaelA/v8-Natives>
- Gives you access to low level engine information. Things like: `getHeapUsage()`, `deoptimizeNow`, `optimizeFunctionOnNextCall`, `getOptimizationStatus`
- `v8.helpers.testOptimization(func[, funcNames])` - Allows you to test a set of functions to see if v8 will be able to optimize it.

# Example code that can't be optimized

- debugger; statement.
- Polymorphic functions



```
function Debugger() {  
    var test="hi";  
    if (false) {  
        debugger;  
    }  
}
```

```
function addTest() {  
    add(10, 20);  
    add("testing ", "this");  
}  
  
function add(x, y) {  
    return x + y;  
}
```

# Beware of Object changes!

```
var MyClass = (function () {  
  function myClass() {  
    this._hi = "Hello";  
  }  
  
  myClass.prototype.nativeScript = function() {  
    this._nativeScript = "NativeScript";  
  };  
  
  return myClass;  
} ());
```

# A little about me

- Developer of the NativeScript.rocks sites:
- > 20 years doing programming and DevOps
- > 20 Languages and many OS's
- > 30 NativeScript Plugins
- Wrote books on NativeScript
-  Owner of Master.Technology
-  Senior partner in nStudio, LLC.



- Blog: <http://fluentReports.com/blog>
- Twitter: @CongoCart
- Email: Nathan@Master.Technology
- Email: NAnderson@nstudio.io

