What is Nim?

» Compiled
» Statically typed
» Garbage collected
» Speed of C, ease of Python, flexibility of Perl

```nim
# Compute average line length
# From nim-lang.org
var
    sum = 0
    count = 0

for line in stdin.lines:
    sum += line.len
    count += 1

echo("Average line length: ",
    if count > 0: sum / count else: 0)
```
Nims killer feature - Macros

template withLock(lock: Lock, body: untyped) =
    acquire lock
    try:
        body
    finally:
        release lock

var ourLock: Lock
initLock ourLock

withLock ourLock:
    echo "Do something that requires locking"
    echo "This might throw an exception"
import macros, strutils

macro toLookupTable(data: static[string]): untyped =
    result = newTree(nnkBracket)
    for w in data.split(';;'):
        result.add newLit(w)

const
data = "mov;btc;cli;xor"
opcodes = toLookupTable(data)

for o in opcodes:
    echo o
Compilation targets

» Compiles to other languages C/C++/JS
» Can target any platform
» Can use native libraries
» Standing on the shoulders of giants
» Creates fast code, not human code
» Why not LLVM/WebAssembly?
Javascript vs. C/C++

» No lowest common denominator
» Builds on a common syntax and capabilities
» Not all code can run on all targets

```javascript
let sockPointer = case sock_addr.ss_family:
   of AF_INET.TSa_Family:
      cast[pointer](cast[int](sock_addr.addr)
         + offsetOf(Sockaddr_in, sin_addr))
   of AF_INET6.TSa_Family:
      cast[pointer](cast[int](sock_addr.addr)
         + offsetOf(Sockaddr_in6, sin6_addr))
   else:
      cast[pointer](cast[int](sock_addr.addr)
         + sizeof(TSa_Family))
if inet_ntop(sock_addr.ss_family.cint,
    sockPointer, result[0].addr, size) == nil:
    result = ""
result.setLen(result.find('\0'))
```
Nim on the smallest
Nim on the smallest

loadSprite(logo, "arduboy_logo_border.bmp", addSize = false)
loadSprite(logoMask, "arduboy_logo_border_mask.bmp", addSize = false)

proc setup*() {.exportc.} =
  NimMain()
  drawBitmap(20, 10, logo, logoMask, 90, 18, 55, 18 div 2, SpriteMasked)
  boot()
  display()

proc loop*() {.exportc.} =
  display()
  let buttons = buttonsState()
  RedLed.off()
  GreenLed.off()
  if buttons.pressed(AButton):
    RedLed.on()
  if buttons.pressed(BButton):
    GreenLed.on()
Nim on the smallest

```nim
import arduino

setup:
    pinMode LED_BUILTIN, OUTPUT
    Serial.begin 9600
    Serial.print "Hello, Nim!
"

loop:
    digitalWrite LED_BUILTIN, HIGH
    delay 500
    digitalWrite LED_BUILTIN, LOW
    delay 500
```

Done compiling.
Nim on the server/desktop

- Can again use all libraries
- Runs super fast
- Forum, playground, and games servers
- Many terminal tools, some GUI apps
- Games
Nim on the server/desktop
Nim on the server/desktop
Nim on the web

» Compilation with JS
» Has JS specific modules
» Can use JS libraries
» Uses the JS garbage collector
Nim on the web

It's about time to earn some chips. Restaurants produce chips. Press here to restore this old place and start earning.
## Nim on the web

<table>
<thead>
<tr>
<th>Topic</th>
<th>Users</th>
<th>Replies</th>
<th>Views</th>
<th>Activity</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hot code reloading</td>
<td></td>
<td>0</td>
<td>48</td>
<td>10h</td>
</tr>
<tr>
<td>Help: Zip (Creating / Opening)</td>
<td></td>
<td>1</td>
<td>108</td>
<td>11h</td>
</tr>
<tr>
<td>Introducing Norm: a Nim ORM</td>
<td></td>
<td></td>
<td>35</td>
<td>15h</td>
</tr>
<tr>
<td>Equivalent of VBA With <code>&lt;x&gt;</code> structure</td>
<td></td>
<td>5</td>
<td>131</td>
<td>15h</td>
</tr>
<tr>
<td>Jester: How to serve static files in production (Heroku)?</td>
<td></td>
<td>4</td>
<td>239</td>
<td>17h</td>
</tr>
<tr>
<td>Unicode support for Windows 10 console</td>
<td></td>
<td>0</td>
<td>86</td>
<td>20h</td>
</tr>
<tr>
<td>Nim calling Lemon parser and SIGSEGV</td>
<td></td>
<td>5</td>
<td>425</td>
<td>22h</td>
</tr>
<tr>
<td>how to use Nimpretty ??</td>
<td></td>
<td>4</td>
<td>308</td>
<td>1d</td>
</tr>
<tr>
<td>Dash docsets now available</td>
<td></td>
<td>9</td>
<td>926</td>
<td>1d</td>
</tr>
</tbody>
</table>