WEB APPLICATION DEVELOPMENT, PRODUCTION, AND DEPLOYMENT

Code 301

AGENDA

- Review: REST, GitHub API, blog code
- Development vs. production: concepts and real example
- Development
  - Local env. vars. and shell startup
  - Step-by-step
- Production
  - Server env. vars using Heroku's "Dashboard" site
  - Step-by-step Heroku deployment
- Preview of final project: "Teams & topics" selection process
- Announcements

REVIEW

REST, GitHub API, scopes, tokens
and blog code

DEVELOPMENT VS. PRODUCTION

DEVELOPMENT VS. PRODUCTION

- “I swear this all worked just fine yesterday…”
- “Now where is that version I wanted to show?…”
- “I haven’t tested it *that* much, but it’s ready to go!”
- “Hey, these files exist! We’re done!”

DEVELOPMENT VS. PRODUCTION

- Whiteboard sketch:
  - Experimental branches
  - Parallel development and release branches
  - Merging
  - Parallel environments (not same as parallel branches)
    - Combat training vs. real combat
    - Prototypes vs. COTS units
    - Beta test vs. product launch
    - Local server, environment, & client code vs. server code

DEVELOPMENT VS. PRODUCTION: LINUX KERNEL

- What is a kernel?

DEVELOPMENT VS. PRODUCTION: LINUX KERNEL

- What is a kernel?
- This?

DEVELOPMENT VS. PRODUCTION: LINUX KERNEL

- What is a kernel?
- Or maybe… ?

DEVELOPMENT VS. PRODUCTION: LINUX KERNEL

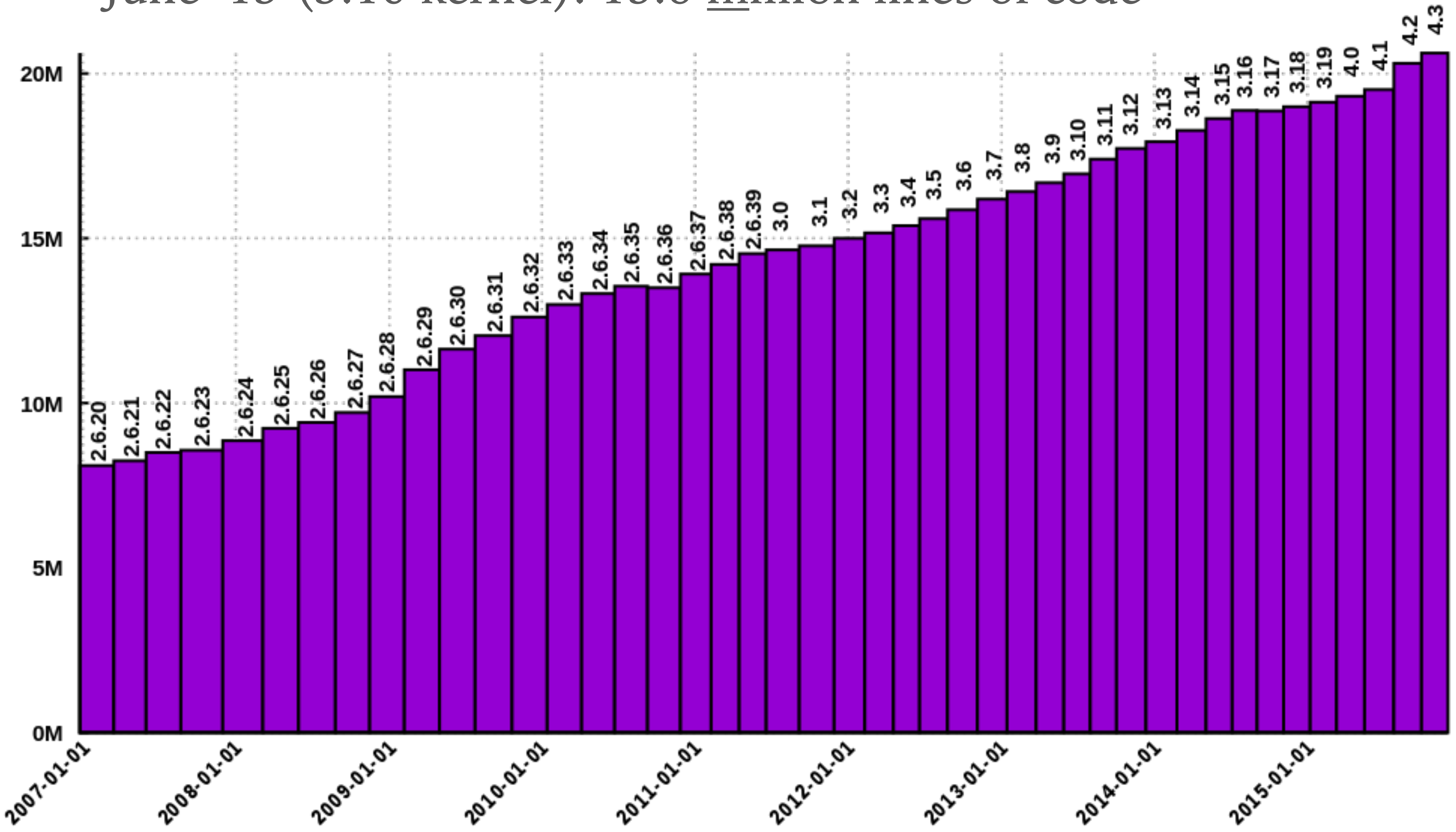
- What is a kernel?
  - Heart of an operating system (OS).
  - Manages and controls access to a system's resources.
  - Linux kernel is all/mostly/partially pre-emptive, depending on config and variation.
- Linux kernel is extremely useful and valuable.

DEVELOPMENT VS. PRODUCTION: LINUX KERNEL SIGNIFICANCE

- Used by 36.2% of all websites (Jan 20 '16, w3techs.com)
- Main OS on IBM's Blue Gene supercomputer.
- **98.8%** of world's 500 fastest supercomputers run a variant of Linux, including the **top 207** (As of Nov. '15).
- Modified versions: Google **Android**, Firefox OS, HP webOS
- ChromeOS
- Router firmware
- One of the most widely ported OS kernels, runs on a range of systems from ARM to IBM Z mainframes.

DEVELOPMENT VS. PRODUCTION: LINUX KERNEL

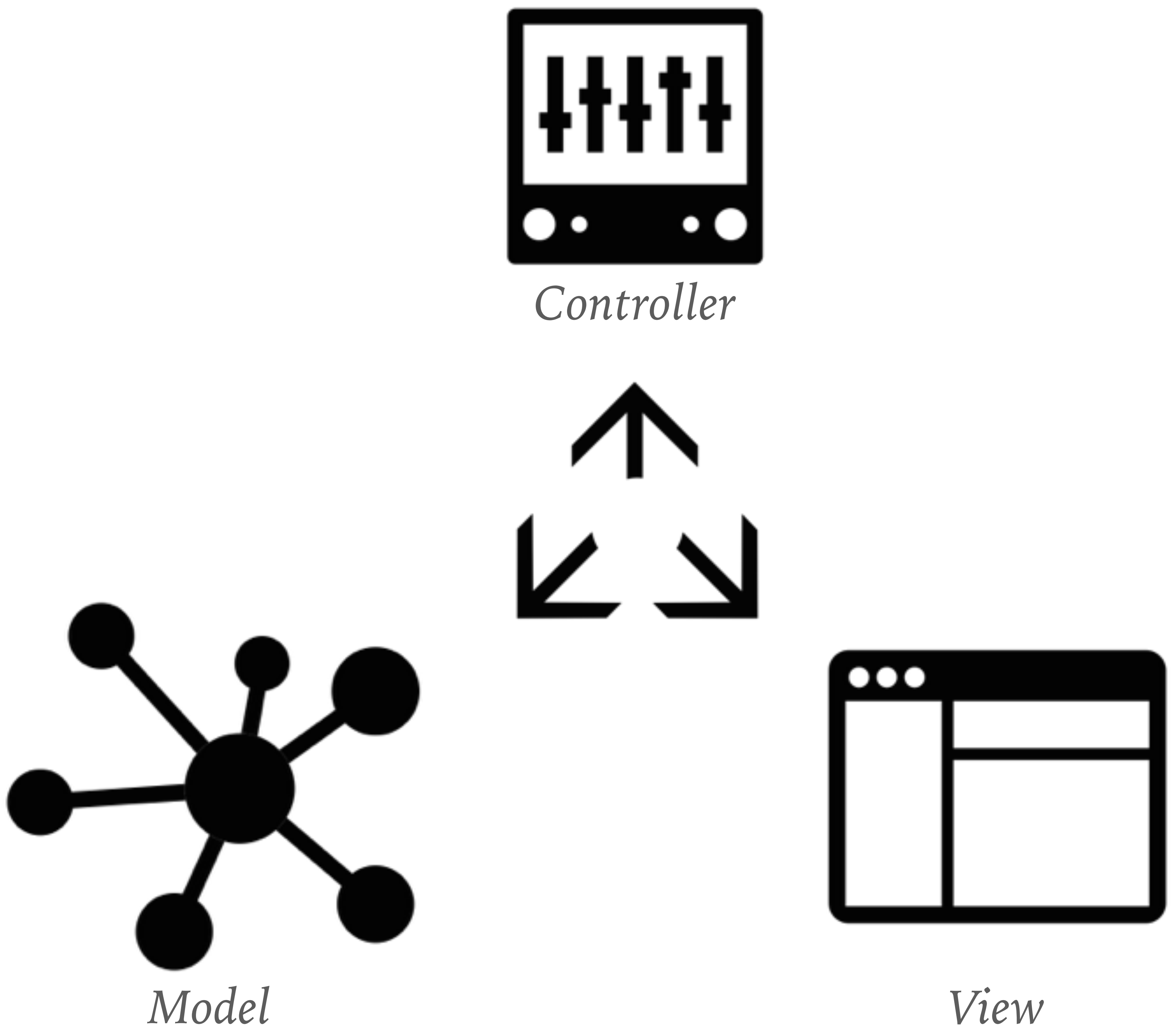
- Mar. ’11 (2.6.x kernel): $3 Billion (est.) to re-develop
- June ’13 (3.10 kernel): 15.8 million lines of code



FUN FACT: LINUX KERNEL & GIT

- In 2002, Linux kernel development switched to BitKeeper, an SCM system which satisfied Torvalds' technical requirements.
- In April 2005…efforts to reverse-engineer…BitKeeper…led BitMover… to stop supporting the Linux…community. In response, Torvalds and others wrote a new source code control system… called **Git**…written within weeks…
- Git soon developed into a separate project in its own right and gained **wide adoption** in the free software community.

LOOSE COUPLING IS IDEAL (RECALL OUR “MVC FLOW” SLIDE)



DEV. VS. PRODUCTION: LINUX KERNEL - MANAGING COMPLEXITY

The…architecture of the Linux kernel has proved its success. …Essential factors…were the provision for the organization of developers, and… system extensibility.

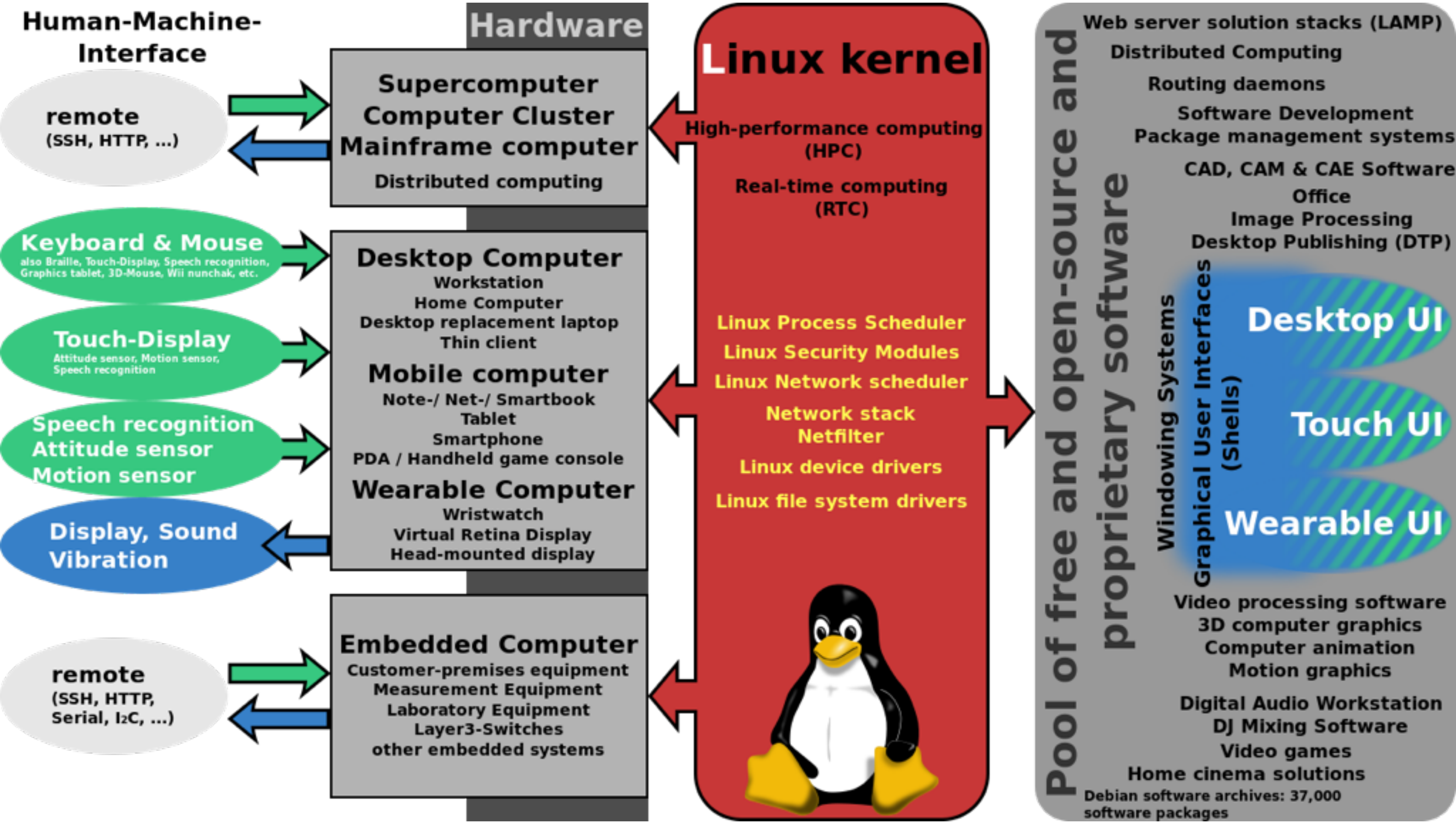
The…architecture was required to support many…developers, which suggested that the…portions that require the most development—hardware device drivers, file systems…—be implemented in an **extensible** fashion.

The …architecture chose to make these systems extensible using a **data abstraction technique** – each hardware device driver is…a separate module that supports a common interface.

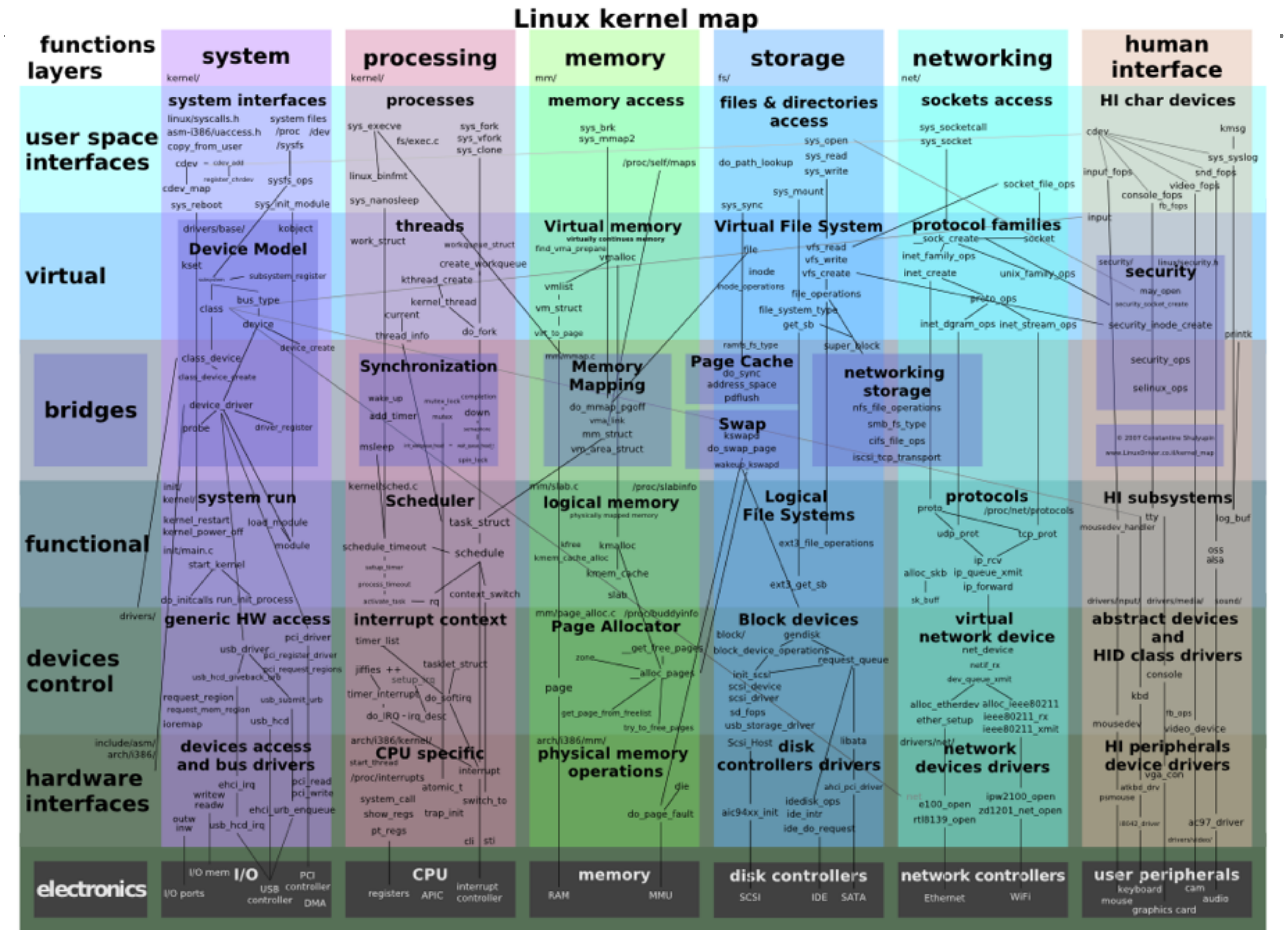
In this way, a single developer can add a new device driver, with **minimal interaction** required with other developers of the Linux kernel.

Sounds like **loose coupling**, and that it works in the real world!

LOOSE COUPLING IS IDEAL: SEPARATION OF CONCERNS



LOOSE COUPLING EXISTS, EVEN INSIDE THE KERNEL ITSELF



DEVELOPMENT VS. PRODUCTION: LINUX KERNEL

- Uses stable and experimental/development branches.
  - Feb '08: Stephen Rothwell created the *linux-next* tree …as a place where patches…to be merged during the next development cycle are gathered.
  - Jan '14: Development version of the kernel is… in an unstable branch… *linux-next*.

DEVELOPMENT VS. PRODUCTION: LINUX KERNEL

- "Linux is evolution, not intelligent design"

  — Linus Torvalds, 2005

- …evolution often does odd (and "sub-optimal") things exactly because it does incremental changes which **do not break at any point**. As a result, any released version of the Linux kernel is fully usable, even if, for example, device drivers do not support all features of the hardware they are written for.

- "There's always a pretty recent version of Linux that works"

  — Some pun-loving Code Fellows instructor

DEVELOPMENT VS. PRODUCTION

- Git
  - Workspace - very temporary
  - Development branches
  - Master or other branch(es) for release and customer demos
  - Demo: Git^2 201 final project - GitHub network graph
- Web and database servers
  - Demo: Visual cue in shell prompts
  - Local workspace, development server, symlinks
  - Development server & DBs/tables, with tests
  - Production server (“always on”)
    No other server roles: Stability and security are critical!

REVISIT: "CLIENT-SIDE ROUTING: HELPFUL LIBRARIES" SLIDE

- page.js routes on the *client* side:
  - Connect routes with handling function:

```
page('/', user.list) // list is a method
page('/', index);  // index is a function
page('/about', about);
page('/contact', contact);
```

- Local web server:

```
$ node server.js
```

ENV VARS: REVISIT "ROUTING USING NODE & EXPRESS: PACKAGES" SLIDE

- Install dependencies

```
$ sudo npm install -g express
$ sudo npm install -g express-request-proxy
```

- Shell start-up file

```
NODE_PATH='/usr/local/lib/node_modules/'
export NODE_PATH
```

- Run server from inside the "starter-code" folder

```
$ node server.js
```

ENVIRONMENT VARIABLES

- Same shell start-up file, i.e.,

  `.bashrc`, `.profile`, `.bash_profile`, `.bashrc`, or `.zcsh`

- Add these lines:

```
GITHUB_TOKEN=‘/usr/local/lib/node_modules/'
export GITHUB_TOKEN
```

LOCAL & REMOTE SERVER ENVIRONMENTS

- “Environment” - for local development and remote execution.
- “Environment variable” - part of a shell’s execution context, variables are dynamic, like global variables.
- Whiteboard sketch:
  - Local dev env. vs remote env. for web app deployment
  - Local env. variable & server env. variable
  - “Local” git vs. remote git

LOCAL DEVELOPMENT ENVIRONMENT SETUP

PRODUCTION / DEPLOYMENT DEMO

RECAP

RECAP

- Review: REST, GitHub API, blog code
- Development vs. production: concepts and real example
- Development
  - Local env. vars. and shell startup
  - Step-by-step
- Production
  - Server env. vars using Heroku's "Dashboard" site
  - Step-by-step Heroku deployment
- Preview of final project: "Teams & topics" selection process
- Announcements

ANNOUNCEMENTS

- Student status
  - 301 pass/fail
  - 401 readiness
  - Assignment grading standard
  - Quiz as indicator
- Today: 3 TAs during lab time
- Thr & Fri: 1:1 time slots
- Thr 1-2 pm: closed-book quiz
- Fri 11:30-12:30 am Optional talk: functional programming
  - Swift code
  - Optimizing functions; OS and mobile/embedded devices

ATTRIBUTIONS

- http://healthyeating.sfgate.com/DM-Resize/photos.demandstudios.com/179/203/fotolia_8565835_XS.jpg?w=442&h=442&keep_ratio=1
- http://img.wonderhowto.com/img/79/22/63416564233383/0/colonel-sanders-kfc-recipe-revealed.w654.jpg
- https://upload.wikimedia.org/wikipedia/commons/thumb/1/11/Lines_of_Code_Linux_Kernel.svg/1000px-Lines_of_Code_Linux_Kernel.svg.png
- http://www.makelinux.net/kernel_map
- https://upload.wikimedia.org/wikipedia/commons/thumb/3/3a/Linux_kernel_ubiquity.svg/1280px-Linux_kernel_ubiquity.svg.png