HOW TO FALL IN LOVE WITH PYTHON AGAIN

(A COMPARISON WITH JAVASCRIPT)

WHAT'S THIS NOW?

- Mostly about traditional web apps (User Interface + Business Logic + Database)
- This presentation is my opinion
- Some of this presentation is hyperbolic, for effect
- Opinions and biases based on personal experience:
  - Personally written 8991 LoC of production Javascript
  - Co-contributor to project with 14016 LoC of production Javascript
  - Written things in Python, Javascript, C++, Omnis Studio, Rust, C#, and Java
  - Professional Developer for ~14 years
- This is a huge topic, and I can't cover it all

WE ALREADY LOVE PYTHON

Down the Rabbit Hole

JAVASCRIPT AND ECMASCRIPT

- Javascript is the implementation
- ECMAScript is the language standard
- Technical Committee 39 (TC39) is the standards committee
- ECMA International is the organization

http://ecma-international.org/

https://github.com/tc39

BRIEF HISTORY OF JAVASCRIPT

- May 1995: Created in a week at Netscape by Brendan Eich.
- Originally called “Mocha”.
- Intended to be a small scripting language for Netscape (a.k.a browsers).
- Functional design
- 1997: ECMAScript created
- 1998: ECMAScript 2 (ES2)
- 1999: ECMAScript 3 (ES3)
- 2000: Scuffle between ECMAScript 4 (original) and ECMAScript 3.1
- 2008: ECMAScript 3.1 slated to become ECMAScript 5 and the “Harmony” agenda was started
- 2009: ECMAScript 5 (ES5)
- 2012: Harmony finalizes as ECMAScript 6 (ES6)

BRIEF HISTORY OF PYTHON

- 1991: Created by Guido Van Rossum
- Intended as a successor to ABC (A language to teach new programmers)
- 1994: Version 1.0
- 2000: Version 2.0
- 2008: Version 3.0
- 2010: Version 2.7 (last of the Python 2 branch)
- 2016: Version 3.6

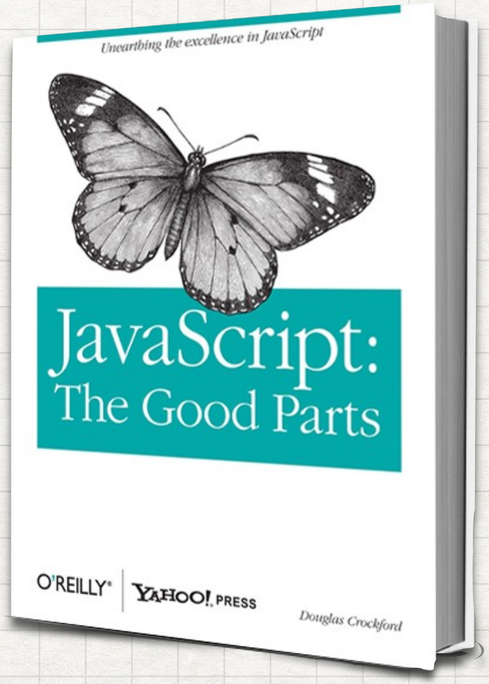
Part 1

“ANY APPLICATION THAT CAN BE WRITTEN IN JAVASCRIPT, WILL EVENTUALLY BE WRITTEN IN JAVASCRIPT.”

- ATWOOD’S LAW (2007)

https://blog.codinghorror.com/the-principle-of-least-power/

JAVASCRIPT IN THE BROWSER



JAVASCRIPT ON THE SERVER

ES6

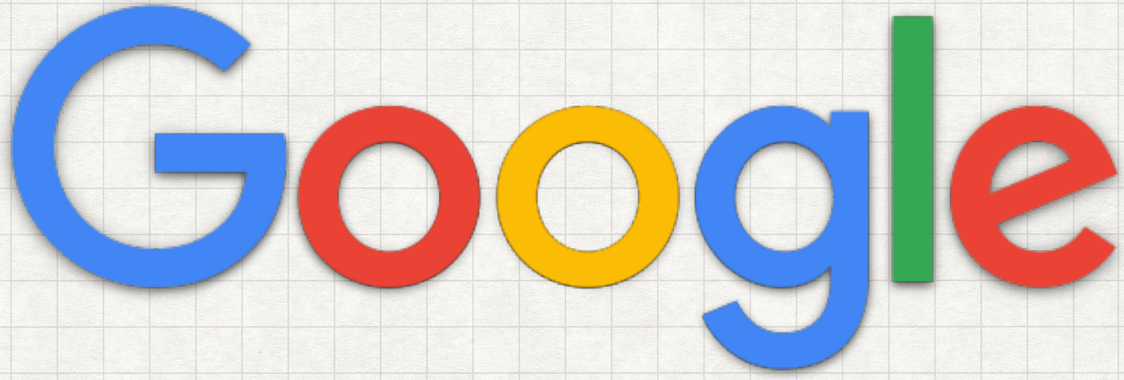
The bits you'll *actually* use

JS
JavaScript

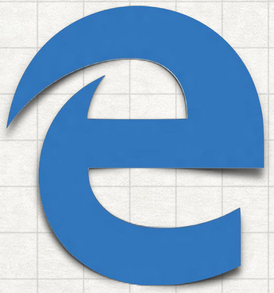
ecma
INTERNATIONAL

facebook

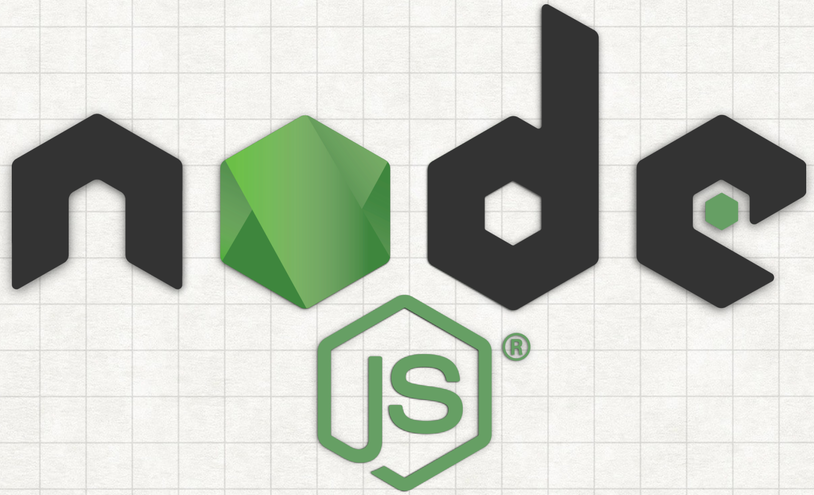
Microsoft

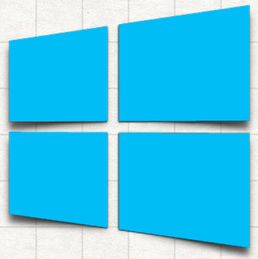

Google

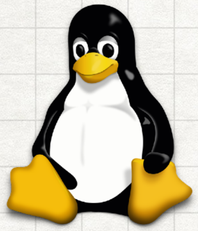
Client (Browser) Side



Server Side

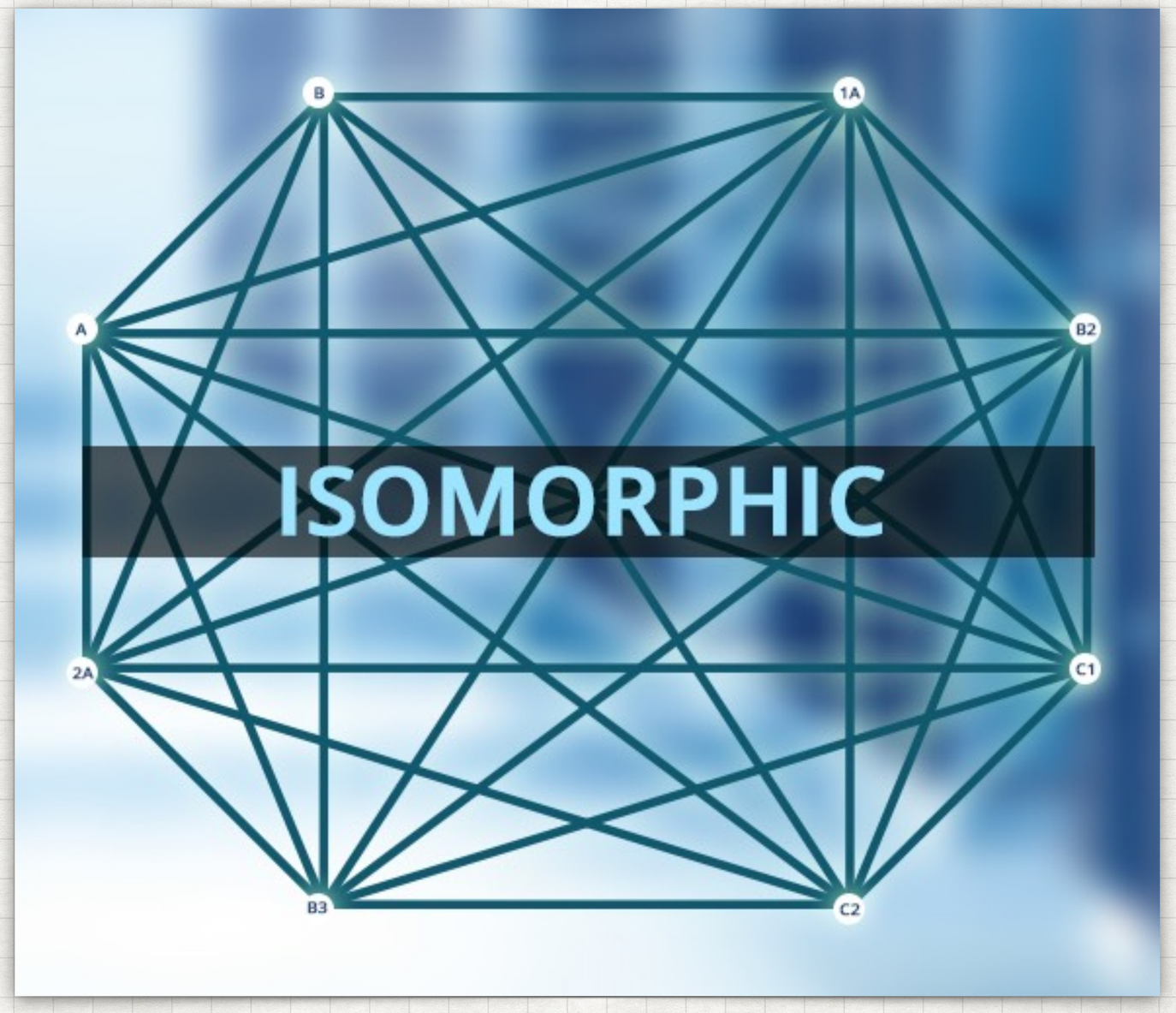






JAVASCRIPT:

CLIENT AND SERVER FULL STACK



Woah.

AND IT'S ALL TRUE!

BUT THERE IS ONE PROBLEM…

YOU HAVE TO USE JAVASCRIPT

AND JAVASCRIPT IS A LITTLE ODD SOMETIMES

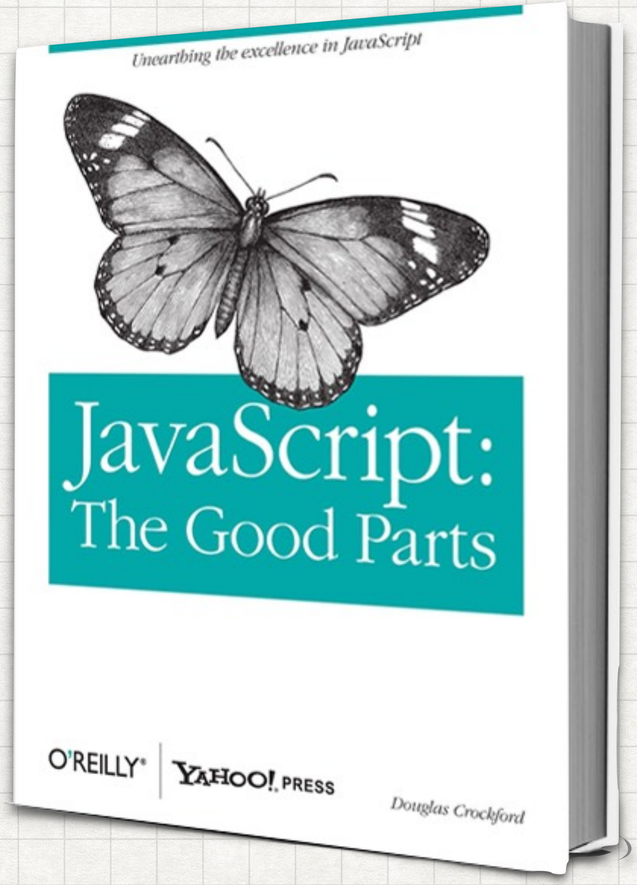
The === Limousine

KEANU IS MORE LIKE

THE “SMELLS” ARE EVERYWHERE

ES6

The bits you’ll *actually* use



COGNITIVE DISSONANCE IS COMMON

"... I OFTEN ENVY THE QUALITY OF THE DOCUMENTATION YOU FIND IN PYTHON PROJECTS.

IN SOME AREAS, PYTHON HAS SOME OF THE BEST LIBRARIES OUT THERE... PYTHON DEVELOPERS KNOW HOW TO COMMUNICATE ABOUT THEIR CODE. THEY DOCUMENT WHAT THEY DO AND ARE PROCESS ORIENTED WHILE BEING PRAGMATIC ABOUT THEIR APPROACHES.

...

PYTHON MAINLY ATTRACTS MORE PRAGMATIC, EXPERIENCED, FULL-STACK DEVELOPERS WANTING A MODERN BUT WELL-PROVEN LANGUAGE."

BUT PYTHON WAS CREATED WAY BEFORE THE INTERNET BECAME POPULAR AND IF CONCURRENCY AND HIGH THROUGHPUT IS A CONCERN FOR YOU, A DYNAMIC, INTERPRETED LANGUAGE WITH POOR CONCURRENCY MIGHT NOT BE THE RIGHT CHOICE.

https://matt.aimonetti.net/posts/2013/08/27/what-technology-should-my-startup-use/

COGNITIVE DISSONANCE IS COMMON (CONT.)

"NODE OFFERS GREAT THROUGHPUT (AS LONG AS YOU STICK TO IO OPERATIONS), IS EASY TO GET STARTED, AND IS FUN TO WRITE.

"DUE TO THE NATURE OF EVENT-BASED PROGRAMMING, DEBUGGING AND TESTING IS CHALLENGING. DEALING WITH CALLBACKS CAN BE MAINTENANCE HELL. ... AND DOCUMENTATION IS TYPICALLY SPOTTY MAKING JUMPING ON AN EXISTING PROJECT DIFFICULT.

...

NODE DEVELOPERS ARE DEFINITELY EARLY ADOPTERS AND COMFORTABLE CREATING A CUSTOM STRUCTURE/PATTERN RATHER THAN FOLLOWING CONVENTION"

https://matt.aimonetti.net/posts/2013/08/27/what-technology-should-my-startup-use/

SUMMARY

- Python
  - Good documentation
  - Great libraries (maybe even best in class)
  - Experienced, pragmatic colleagues
  - Not so fast
  - Actual fun
- Javascript (Node.js)
  - Fast, because Async I/O
  - Masochistic fun

BUT IS IT FASTER THO?

“

“WE SHOULD FORGET ABOUT SMALL EFFICIENCIES, SAY ABOUT 97% OF THE TIME: PREMATURE OPTIMIZATION IS THE ROOT OF ALL EVIL." AND I AGREE WITH THIS. ITS USUALLY NOT WORTH SPENDING A LOT OF TIME MICRO-OPTIMIZING CODE BEFORE ITS OBVIOUS WHERE THE PERFORMANCE BOTTLENECKS ARE.

— Donald Knuth

”

Plain Text

Framework	Best performance (higher is better)	Cls	Lng	Plt	FE	Aos	IA	Errors
wheezy.web	975,225 100.0% (26.1%)	Mcr	Py	Non	Mei	Lin	Rea	0
wheezy.web-py3	901,750 92.5% (24.2%)	Mcr	Py	Non	Mei	Lin	Rea	0
falcon	862,200 88.4% (23.1%)	Mcr	Py	Non	Mei	Lin	Rea	0
falcon-py3	724,863 74.3% (19.4%)	Mcr	Py	Non	Mei	Lin	Rea	0
bottle	492,064 50.5% (13.2%)	Mcr	Py	Mei	Non	Lin	Rea	0
nodejs	467,246 47.9% (12.5%)	Plt	JS	njs	Non	Lin	Rea	774
bottle-py3	408,379 41.9% (10.9%)	Mcr	Py	Mei	Non	Lin	Rea	0
pyramid-py3	343,757 35.2% (9.2%)	Ful	Py	Non	Mei	Lin	Rea	0
pyramid-py2	308,203 31.6% (8.3%)	Ful	Py	Non	Mei	Lin	Rea	0
falcon-pypy	228,810 23.5% (6.1%)	Mcr	Py	Non	Tor	Lin	Rea	0
express	213,982 21.9% (5.7%)	Mcr	JS	Non	Non	Lin	Rea	0
flask-py3	124,800 12.8% (3.3%)	Mcr	Py	Mei	Non	Lin	Rea	0
api-hour+aiohttp.web	115,954 11.9% (3.1%)	Mcr	Py	API	Gun	Lin	Rea	0
hapi	94,123 9.7% (2.5%)	Mcr	JS	Non	Non	Lin	Rea	479

Data Updates

Framework	Performance (higher is better)	Cls	Lng	Plt	FE	Aos	DB	Dos	Orm	IA	Errors
express-mongodb	2,841 100.0% (93.1%)	Mcr	JS	Non	Non	Lin	Mo	Lin	Ful	Rea	0
nodejs	1,982 69.8% (65.0%)	Plt	JS	njs	Non	Lin	My	Lin	Raw	Rea	0
nodejs	1,843 64.9% (60.4%)	Plt	JS	njs	Non	Lin	Pg	Lin	Ful	Rea	0
hapi	1,542 54.3% (50.6%)	Mcr	JS	Non	Non	Lin	Pg	Lin	Ful	Rea	0
pyramid-py2	1,354 47.7% (44.4%)	Ful	Py	Non	Mei	Lin	Pg	Lin	Ful	Rea	0
pyramid-py3	1,238 43.6% (40.6%)	Ful	Py	Non	Mei	Lin	Pg	Lin	Ful	Rea	0
bottle-mysql-raw	1,171 41.2% (38.4%)	Mcr	Py	Mei	Non	Lin	My	Lin	Raw	Rea	0
flask	1,109 39.0% (36.4%)	Mcr	Py	Mei	Non	Lin	My	Lin	Raw	Rea	0
flask-pypy	1,103 38.8% (36.2%)	Mcr	Py	Tor	Non	Lin	My	Lin	Raw	Rea	0
nodejs-mysql	1,047 36.9% (34.3%)	Plt	JS	njs	Non	Lin	My	Lin	Ful	Rea	0
ringojs	992 34.9% (32.5%)	Plt	JS	Jty	Non	Lin	My	Lin	Raw	Rea	0
ringojs-conv	959 33.8% (31.4%)	Mcr	JS	Rin	Non	Lin	My	Lin	Mcr	Rea	0
flask-py3	920 32.4% (30.2%)	Mcr	Py	Mei	Non	Lin	My	Lin	Ful	Rea	0
bottle-py3	901 31.7% (29.5%)	Mcr	Py	Mei	Non	Lin	My	Lin	Ful	Rea	0
django-py3	880 31.0% (28.9%)	Ful	Py	Non	Mei	Lin	My	Lin	Ful	Rea	0
hapi-mysql	880 31.0% (28.9%)	Mcr	JS	Non	Non	Lin	My	Lin	Ful	Rea	0

Fortunes

Framework	Best performance (higher is better)	Cls	Lng	Plt	FE	Aos	DB	Dos	Orm	IA	Errors
nodejs	93,795 100.0% (52.6%)	Plt	JS	njs	Non	Lin	My	Lin	Raw	Rea	0
nodejs-mongodb-raw	87,226 93.0% (48.9%)	Plt	JS	njs	Non	Lin	Mo	Lin	Raw	Rea	0
bottle-mysql-raw	45,230 48.2% (25.4%)	Mcr	Py	Mei	Non	Lin	My	Lin	Raw	Rea	0
nodejs-mysql	38,885 41.5% (21.8%)	Plt	JS	njs	Non	Lin	My	Lin	Ful	Rea	0
nodejs	38,239 40.8% (21.4%)	Plt	JS	njs	Non	Lin	Pg	Lin	Ful	Rea	0
express-mysql	34,691 37.0% (19.5%)	Mcr	JS	Non	Non	Lin	My	Lin	Ful	Rea	0
wheezy.web	28,599 30.5% (16.0%)	Mcr	Py	Non	Mei	Lin	My	Lin	Ful	Rea	0
wheezy.web-py3	26,976 28.8% (15.1%)	Mcr	Py	Non	Mei	Lin	My	Lin	Ful	Rea	0
flask	24,872 26.5% (13.9%)	Mcr	Py	Mei	Non	Lin	My	Lin	Raw	Rea	0
koa	21,328 22.7% (12.0%)	Mcr	JS	Non	Non	Lin	Mo	Lin	Raw	Rea	0
pyramid-py3	21,164 22.6% (11.9%)	Ful	Py	Non	Mei	Lin	Pg	Lin	Ful	Rea	0
pyramid-py2	20,291 21.6% (11.4%)	Ful	Py	Non	Mei	Lin	Pg	Lin	Ful	Rea	0
bottle-py3	17,258 18.4% (9.7%)	Mcr	Py	Mei	Non	Lin	My	Lin	Ful	Rea	0
bottle	16,311 17.4% (9.1%)	Mcr	Py	Mei	Non	Lin	My	Lin	Ful	Rea	0
turbogears	15,279 16.3% (8.6%)	Mcr	Py	Non	Mei	Lin	My	Lin	Ful	Rea	0
flask-py3	12,854 13.7% (7.2%)	Mcr	Py	Mei	Non	Lin	My	Lin	Ful	Rea	0
django-py3	12,601 13.4% (7.1%)	Ful	Py	Non	Mei	Lin	My	Lin	Ful	Rea	0
django-postgresql	12,266 13.1% (6.9%)	Ful	Py	Non	Mei	Lin	Pg	Lin	Ful	Rea	0

Multiple Queries

Framework	Performance (higher is better)	Cls	Lng	Plt	FE	Aos	DB	Dos	Orm	IA	Errors
nodejs	6,923 100.0% (55.6%)	Plt	JS	njs	Non	Lin	My	Lin	Raw	Rea	0
bottle-mysql-raw	6,688 96.6% (53.7%)	Mcr	Py	Mei	Non	Lin	My	Lin	Raw	Rea	0
flask	6,516 94.1% (52.3%)	Mcr	Py	Mei	Non	Lin	My	Lin	Raw	Rea	0
hapi	6,123 88.4% (49.1%)	Mcr	JS	Non	Non	Lin	Mo	Lin	Ful	Rea	0
ringojs	5,947 85.9% (47.7%)	Plt	JS	Jty	Non	Lin	My	Lin	Raw	Rea	0
nodejs-mongodb-raw	5,818 84.0% (46.7%)	Plt	JS	njs	Non	Lin	Mo	Lin	Raw	Rea	0
koa	5,321 76.9% (42.7%)	Mcr	JS	Non	Non	Lin	Mo	Lin	Raw	Rea	0
web2py-optimized	4,784 69.1% (38.4%)	Ful	Py	Mei	Non	Lin	My	Lin	Ful	Rea	0
nodejs-mysql	3,958 57.2% (31.8%)	Plt	JS	njs	Non	Lin	My	Lin	Ful	Rea	0
asyncio	3,934 56.8% (31.6%)	Mcr	Py	API	Gun	Lin	My	Lin	Raw	Rea	0
express-mysql	3,840 55.5% (30.8%)	Mcr	JS	Non	Non	Lin	My	Lin	Ful	Rea	0
express-mongodb	3,772 54.5% (30.3%)	Mcr	JS	Non	Non	Lin	Mo	Lin	Ful	Rea	0
nodejs-mongodb	3,755 54.2% (30.1%)	Plt	JS	njs	Non	Lin	Mo	Lin	Ful	Rea	0
ringojs-conv	3,658 52.8% (29.4%)	Mcr	JS	Rin	Non	Lin	My	Lin	Mcr	Rea	0
nodejs	3,152 45.5% (25.3%)	Plt	JS	njs	Non	Lin	Pg	Lin	Ful	Rea	0
hapi-mysql	3,017 43.6% (24.2%)	Mcr	JS	Non	Non	Lin	My	Lin	Ful	Rea	0
web2py	2,532 36.6% (20.3%)	Ful	Py	Mei	Non	Lin	My	Lin	Ful	Rea	0
hapi	2,511 36.3% (20.2%)	Mcr	JS	Non	Non	Lin	Pg	Lin	Ful	Rea	0

https://www.techempower.com/benchmarks/#section=data-r13&hw=ph&test=query&l=4ftgcf

Single Query

Framework	Best performance (higher is better)		Cls	Lng	Plt	FE	Aos	DB	Dos	Orm	IA	Errors
nodejs	131,904	100.0% (68.1%)	Plt	JS	njs	Non	Lin	My	Lin	Raw	Rea	0
nodejs-mongodb-raw	110,002	83.4% (56.8%)	Plt	JS	njs	Non	Lin	Mo	Lin	Raw	Rea	0
bottle-mysql-raw	80,347	60.9% (41.5%)	Mcr	Py	Mei	Non	Lin	My	Lin	Raw	Rea	0
nodejs-mongodb	58,194	44.1% (30.0%)	Plt	JS	njs	Non	Lin	Mo	Lin	Ful	Rea	0
nodejs-mysql	56,574	42.9% (29.2%)	Plt	JS	njs	Non	Lin	My	Lin	Ful	Rea	0
koa	55,869	42.4% (28.8%)	Mcr	JS	Non	Non	Lin	Mo	Lin	Raw	Rea	0
express-mongodb	48,392	36.7% (25.0%)	Mcr	JS	Non	Non	Lin	Mo	Lin	Ful	Rea	0
nodejs	47,952	36.4% (24.8%)	Plt	JS	njs	Non	Lin	Pg	Lin	Ful	Rea	0
flask	47,261	35.8% (24.4%)	Mcr	Py	Mei	Non	Lin	My	Lin	Raw	Rea	0
ringojs	45,864	34.8% (23.7%)	Plt	JS	Jty	Non	Lin	My	Lin	Raw	Rea	0
express-mysql	45,550	34.5% (23.5%)	Mcr	JS	Non	Non	Lin	My	Lin	Ful	Rea	0
asyncio	33,108	25.1% (17.1%)	Mcr	Py	API	Gun	Lin	My	Lin	Raw	Rea	19
wheezy.web	29,477	22.3% (15.2%)	Mcr	Py	Non	Mei	Lin	My	Lin	Ful	Rea	0
wheezy.web-py3	28,974	22.0% (15.0%)	Mcr	Py	Non	Mei	Lin	My	Lin	Ful	Rea	0
pyramid-py2	27,815	21.1% (14.4%)	Ful	Py	Non	Mei	Lin	Pg	Lin	Ful	Rea	0
pyramid-py3	27,377	20.8% (14.1%)	Ful	Py	Non	Mei	Lin	Pg	Lin	Ful	Rea	0
django-postgresql	26,549	20.1% (13.7%)	Ful	Py	Non	Mei	Lin	Pg	Lin	Ful	Rea	0
tornado	26,235	19.9% (13.5%)	Plt	Py	Non	Tor	Lin	Pg	Lin	Raw	Rea	0

JSON Serialization

Framework	Best performance (higher is better)		Cls	Lng	Plt	FE	Aos	IA	Errors
falcon	420,820	100.0% (75.7%)	Mcr	Py	Non	Mei	Lin	Rea	0
falcon-py3	413,205	98.2% (74.3%)	Mcr	Py	Non	Mei	Lin	Rea	0
wheezy.web	387,402	92.1% (69.7%)	Mcr	Py	Non	Mei	Lin	Rea	0
wheezy.web-py3	378,857	90.0% (68.1%)	Mcr	Py	Non	Mei	Lin	Rea	0
bottle	339,180	80.6% (61.0%)	Mcr	Py	Mei	Non	Lin	Rea	0
nodejs	304,372	72.3% (54.7%)	Plt	JS	njs	Non	Lin	Rea	0
bottle-py3	292,545	69.5% (52.6%)	Mcr	Py	Mei	Non	Lin	Rea	0
falcon-pypy	188,730	44.8% (33.9%)	Mcr	Py	Non	Tor	Lin	Rea	0
express	186,548	44.3% (33.5%)	Mcr	JS	Non	Non	Lin	Rea	0
pyramid-py3	167,915	39.9% (30.2%)	Ful	Py	Non	Mei	Lin	Rea	0
pyramid-py2	139,978	33.3% (25.2%)	Ful	Py	Non	Mei	Lin	Rea	0
koa	136,573	32.5% (24.6%)	Mcr	JS	Non	Non	Lin	Rea	0
flask-py3	107,098	25.4% (19.3%)	Mcr	Py	Mei	Non	Lin	Rea	0
api-hour+aiohttp.web	90,464	21.5% (16.3%)	Mcr	Py	API	Gun	Lin	Rea	0
turbogears	88,898	21.1% (16.0%)	Mcr	Py	Non	Mei	Lin	Rea	0
django	87,706	20.8% (15.8%)	Ful	Py	Non	Mei	Lin	Rea	0
django-py3	83,865	19.9% (15.1%)	Ful	Py	Non	Mei	Lin	Rea	0
bottle-pypy	68,370	16.2% (12.3%)	Mcr	Py	Tor	Non	Lin	Rea	0

https://www.techempower.com/benchmarks/#section=data-r13&hw=ph&test=query&l=4ftgcf

RAW VS. FRAMEWORKS

Multiple Queries

Framework	Performance (higher is better)		Cls	Lng	Plt	FE	Aos	DB	Dos	Orm	IA	Errors
nodejs	6,923	100.0% (55.6%)	Plt	JS	njs	Non	Lin	My	Lin	Raw	Rea	0
bottle-mysql-raw	6,688	96.6% (53.7%)	Mcr	Py	Mei	Non	Lin	My	Lin	Raw	Rea	0
flask	6,516	94.1% (52.3%)	Mcr	Py	Mei	Non	Lin	My	Lin	Raw	Rea	0
hapi	6,123	88.4% (49.1%)	Mcr	JS	Non	Non	Lin	Mo	Lin	Ful	Rea	0
ringojs	5,947	85.9% (47.7%)	Plt	JS	Jty	Non	Lin	My	Lin	Raw	Rea	0
nodejs-mongodb-raw	5,818	84.0% (46.7%)	Plt	JS	njs	Non	Lin	Mo	Lin	Raw	Rea	0
koa	5,321	76.9% (42.7%)	Mcr	JS	Non	Non	Lin	Mo	Lin	Raw	Rea	0
web2py-optimized	4,784	69.1% (38.4%)	Ful	Py	Mei	Non	Lin	My	Lin	Ful	Rea	0
nodejs-mysql	3,958	57.2% (31.8%)	Plt	JS	njs	Non	Lin	My	Lin	Ful	Rea	0
asyncio	3,934	56.8% (31.6%)	Mcr	Py	API	Gun	Lin	My	Lin	Raw	Rea	0
express-mysql	3,840	55.5% (30.8%)	Mcr	JS	Non	Non	Lin	My	Lin	Ful	Rea	0
express-mongodb	3,772	54.5% (30.3%)	Mcr	JS	Non	Non	Lin	Mo	Lin	Ful	Rea	0
nodejs-mongodb	3,755	54.2% (30.1%)	Plt	JS	njs	Non	Lin	Mo	Lin	Ful	Rea	0
ringojs-conv	3,658	52.8% (29.4%)	Mcr	JS	Rin	Non	Lin	My	Lin	Mcr	Rea	0
nodejs	3,152	45.5% (25.3%)	Plt	JS	njs	Non	Lin	Pg	Lin	Ful	Rea	0
hapi-mysql	3,017	43.6% (24.2%)	Mcr	JS	Non	Non	Lin	My	Lin	Ful	Rea	0
web2py	2,532	36.6% (20.3%)	Ful	Py	Mei	Non	Lin	My	Lin	Ful	Rea	0
hapi	2,511	36.3% (20.2%)	Mcr	JS	Non	Non	Lin	Pg	Lin	Ful	Rea	0

https://www.techempower.com/benchmarks/#section=data-r13&hw=ph&test=query&l=4ftgcf

Compared To Everything

Framework	Performance (higher is better)		Cls	Lng	Plt	FE	Aos	DB	Dos	Orm	IA	Errors
stream	12,459	100.0%	Mcr	Dar	Non	ngx	Lin	Mo	Lin	Raw	Rea	0
start	12,287	98.6%	Mcr	Dar	Non	ngx	Lin	Mo	Lin	Raw	Rea	0
redstone-mongodb	10,974	88.1%	Mcr	Dar	Non	Non	Lin	Mo	Lin	Mcr	Rea	0
treefrog-mongo	10,014	80.4%	Ful	C++	Non	Non	Lin	Mo	Lin	Mcr	Rea	0
ulib-mongodb	9,996	80.2%	Plt	C++	Non	ULi	Lin	Mo	Lin	Mcr	Rea	0
ulib-postgres	8,930	71.7%	Plt	C++	Non	ULi	Lin	Pg	Lin	Mcr	Rea	0
fasthttp-postgresql-	8,731	70.1%	Plt	Go	Non	Non	Lin	Pg	Lin	Raw	Rea	0
wt-postgres	8,029	64.4%	Ful	C++	Non	Non	Lin	Pg	Lin	Ful	Rea	0
fasthttp	7,865	63.1%	Plt	Go	Non	Non	Lin	My	Lin	Raw	Rea	0
dropwizard	7,864	63.1%	Ful	Jav	JAX	Non	Lin	Pg	Lin	Ful	Rea	0
ulib-mysql	7,781	62.5%	Plt	C++	Non	ULi	Lin	My	Lin	Mcr	Rea	0
phalcon	7,768	62.3%	Ful	PHP	Non	ngx	Lin	Mo	Lin	Raw	Rea	0
urweb-postgres	7,368	59.1%	Ful	Ur	Ur/	Non	Lin	Pg	Lin	Mcr	Rea	0
wt	7,280	58.4%	Ful	C++	Non	Non	Lin	My	Lin	Ful	Rea	0
servlet-postgres-raw	7,041	56.5%	Plt	Jav	Svt	Res	Lin	Pg	Lin	Raw	Rea	0
cpoll_cppsp-postgres	7,007	56.2%	Plt	C++	Non	Non	Lin	Pg	Lin	Raw	Rea	0
silicon-epoll-mysql	6,976	56.0%	Mcr	C++	Non	mic	Lin	My	Lin	Ful	Rea	0
nodejs	6,923	55.6%	Plt	JS	njs	Non	Lin	My	Lin	Raw	Rea	0
cpoll_cppsp-raw	6,884	55.3%	Plt	C++	Non	Non	Lin	My	Lin	Raw	Rea	0
wildfly-ee7	6,855	55.0%	Ful	Jav	Svt	Wil	Lin	My	Lin	Ful	Rea	0
silicon-tpc-mysql	6,802	54.6%	Mcr	C++	Non	mic	Lin	My	Lin	Ful	Rea	0
http-kit	6,798	54.6%	Plt	Clj	Rin	Non	Lin	My	Lin	Raw	Rea	0
kami	6,734	54.0%	Mcr	Go	Non	Non	Lin	My	Lin	Raw	Rea	0
jawn	6,727	54.0%	Ful	Jav	Svt	Utw	Lin	My	Lin	Raw	Rea	0
bottle-mysql-raw	6,688	53.7%	Mcr	Py	Mei	Non	Lin	My	Lin	Raw	Rea	0
ninja-standalone	6,677	53.6%	Ful	Jav	Jty	Non	Lin	My	Lin	Ful	Rea	0
php-phalcon-micro	6,643	53.3%	Mcr	PHP	Non	ngx	Lin	My	Lin	Raw	Rea	0
cutelyst-pf-pg-raw	6,622	53.2%	Plt	C++	Qt	Non	Lin	Pg	Lin	Raw	Rea	0
cutelyst-uwsgi-nginx	6,619	53.1%	Plt	C++	Qt	ngx	Lin	Pg	Lin	Raw	Rea	0
yaf	6,600	53.0%	Ful	PHP	Non	ngx	Lin	My	Lin	Raw	Rea	0
php5-raw	6,595	52.9%	Plt	PHP	Non	ngx	Lin	My	Lin	Raw	Rea	0
cpoll_cppsp-postgres	6,571	52.7%	Plt	C++	Non	Non	Lin	Pg	Lin	Raw	Rea	0
cutelyst-thread-pg-r	6,569	52.7%	Plt	C++	Qt	Non	Lin	Pg	Lin	Raw	Rea	0
php-raw	6,540	52.5%	Plt	PHP	Non	ngx	Lin	My	Lin	Raw	Rea	0
flask	6,516	52.3%	Mcr	Py	Mei	Non	Lin	My	Lin	Raw	Rea	0
revenj.jvm	6,503	52.2%	Ful	Jav	Svt	Res	Lin	Pg	Lin	Ful	Rea	0
vertx-web-jdbc	6,482	52.0%	Mcr	Jav	vtx	Non	Lin	Pg	Lin	Raw	Rea	0
fat-free	6,302	50.6%	Ful	PHP	Non	ngx	Lin	My	Lin	Raw	Rea	0
h2o	6,268	50.3%	Plt	C	Non	Non	Lin	Pg	Lin	Raw	Rea	0
servlet-raw	6,250	50.2%	Plt	Jav	Svt	Res	Lin	My	Lin	Raw	Rea	0
gemini-mysql	6,186	49.7%	Ful	Jav	Svt	Res	Lin	My	Lin	Mcr	Rea	0
hapi	6,123	49.1%	Mcr	JS	Non	Non	Lin	Mo	Lin	Ful	Rea	0
compojure-raw	6,078	48.8%	Mcr	Clj	Svt	Res	Lin	My	Lin	Raw	Rea	0
ringojs	5,947	47.7%	Plt	JS	Jty	Non	Lin	My	Lin	Raw	Rea	0
openresty	5,941	47.7%	Plt	Lua	OpR	ngx	Lin	My	Lin	Raw	Rea	0
silex	5,862	47.1%	Mcr	PHP	Non	ngx	Lin	My	Lin	Raw	Rea	0
nodejs-mongodb-raw	5,818	46.7%	Plt	JS	njs	Non	Lin	Mo	Lin	Raw	Rea	0
fat-free	5,784	46.4%	Ful	PHP	Non	ngx	Lin	My	Lin	Ful	Rea	0
echo-prefork	5,767	46.3%	Mcr	Go	Non	Non	Lin	Pg	Lin	Raw	Rea	0

https://www.techempower.com/benchmarks/#section=data-r13&hw=ph&test=query

PERFORMANCE CONCLUSION

- Sometimes Javascript is faster, sometimes Python is faster
- Frameworks used, algorithms, and design are likely to have a larger performance impact than the language
- Python and Javascript are both slow compared to Go, Java, and C++

- ASIDE: Most of what makes node fast/interesting are C libs **libuv** and **http-parser**, which can be used from Python too
  - https://github.com/libuv/libuv
  - https://github.com/saghul/pyuv — Python 3.4+ asyncio loop: https://github.com/saghul/aiouv
  - https://github.com/nodejs/http-parser
  - https://github.com/benoitc/http-parser/

PYTHON IS MORE WELL ROUNDED

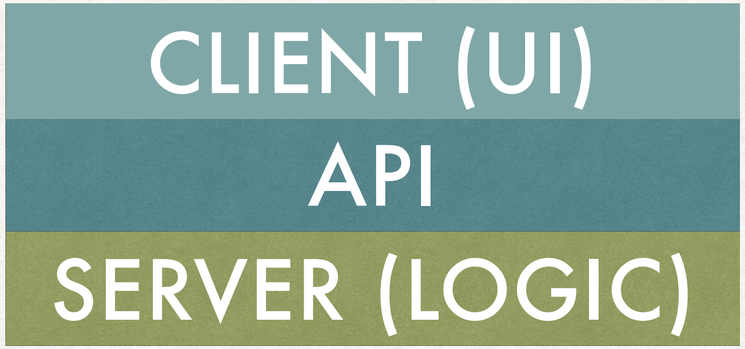
(Take my word for it, for now)

- Just as fast
- Better docs
- Easier to read
- More consistent
- Better libraries, from many domains
- Experienced, respectful community
- Well organized

OK,
SO
WE
STILL
LOVE
PYTHON

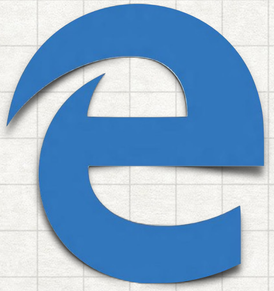
WHAT ABOUT ISOMORPHIC?

- You still have to write User Interface (UI) code
- You still have to write business logic code
- You still have to write code to connect the two
- … And there is the internet in between
- You still have to secure your API
- Only basic validation* can cross the API boundary, and it still needs to be double checked on the server.

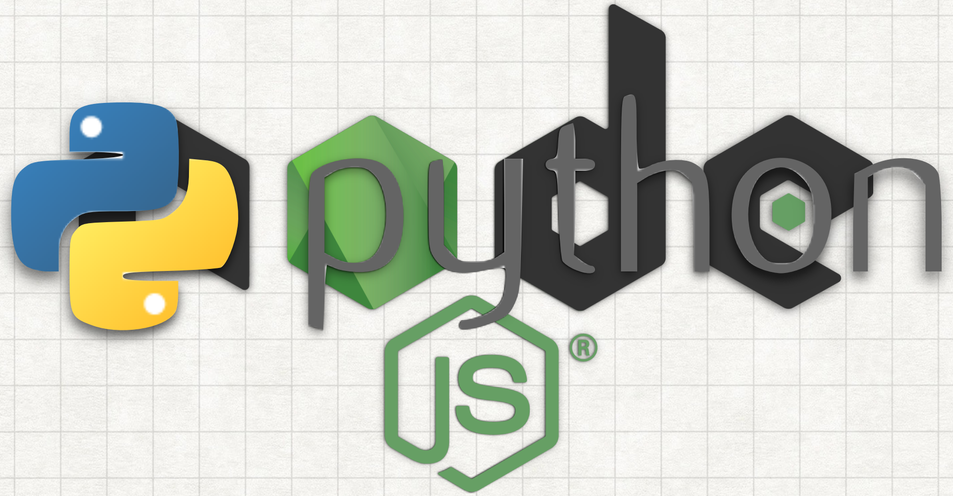


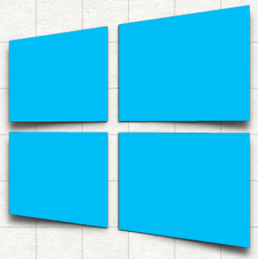
*Complex validation usually requires database, which is usually cheaper to process on the server.

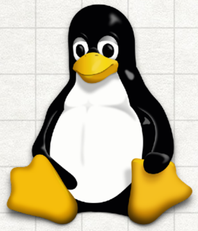
Client (Browser) Side



Server Side

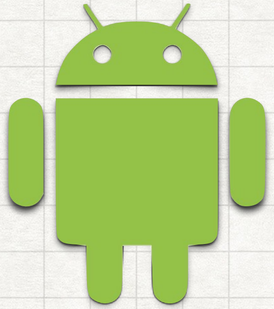




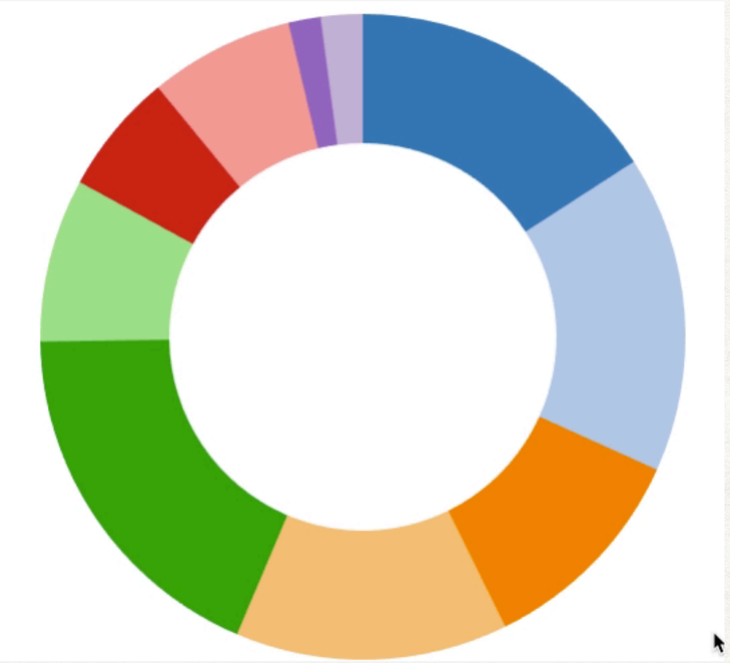


Client (Browser) Side

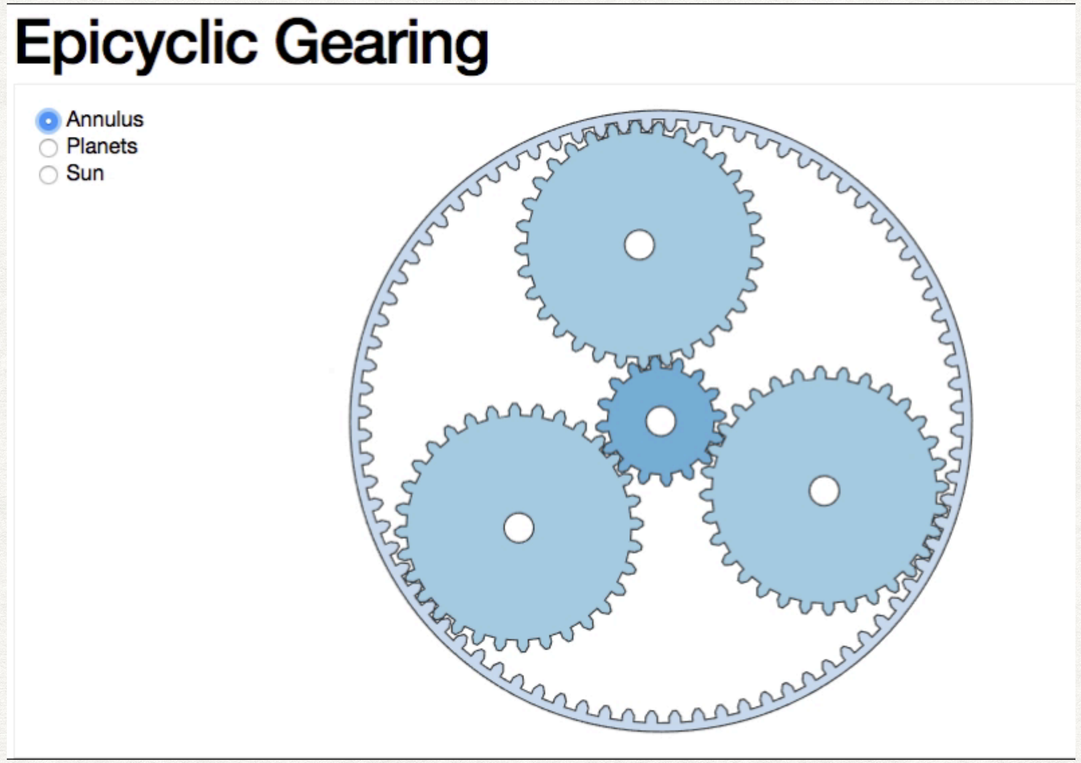
Client (Mobile) Side



THE CLIENT SIDE HAS COOL STUFF



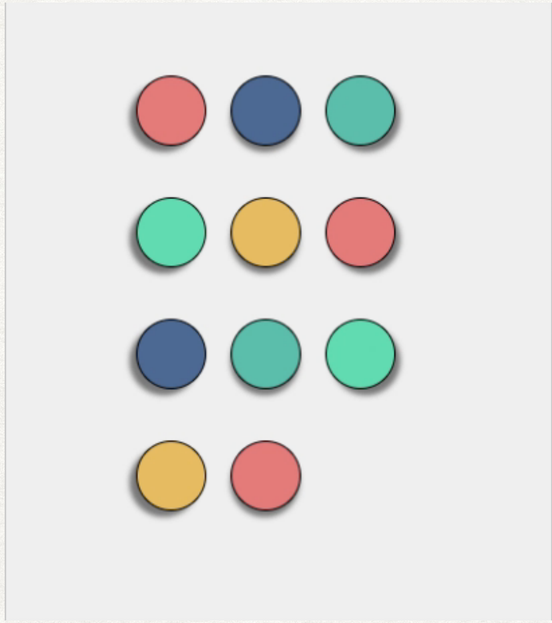
http://bl.ocks.org/mbostock/4341417

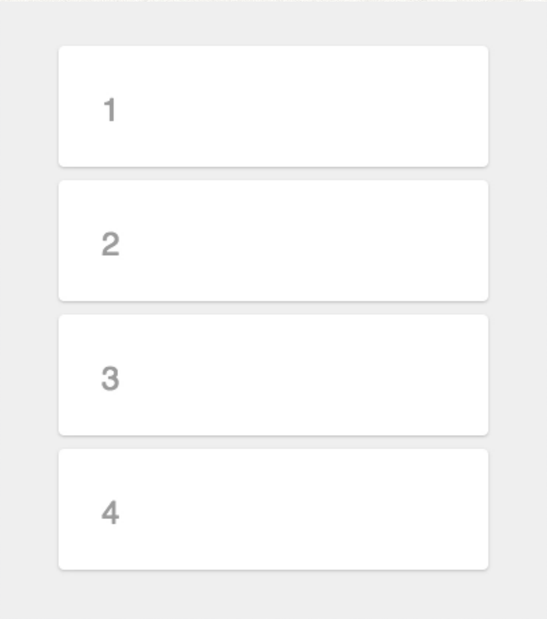


http://bl.ocks.org/mbostock/1353700

https://d3js.org/

THE CLIENT SIDE HAS COOL STUFF (CONT.)





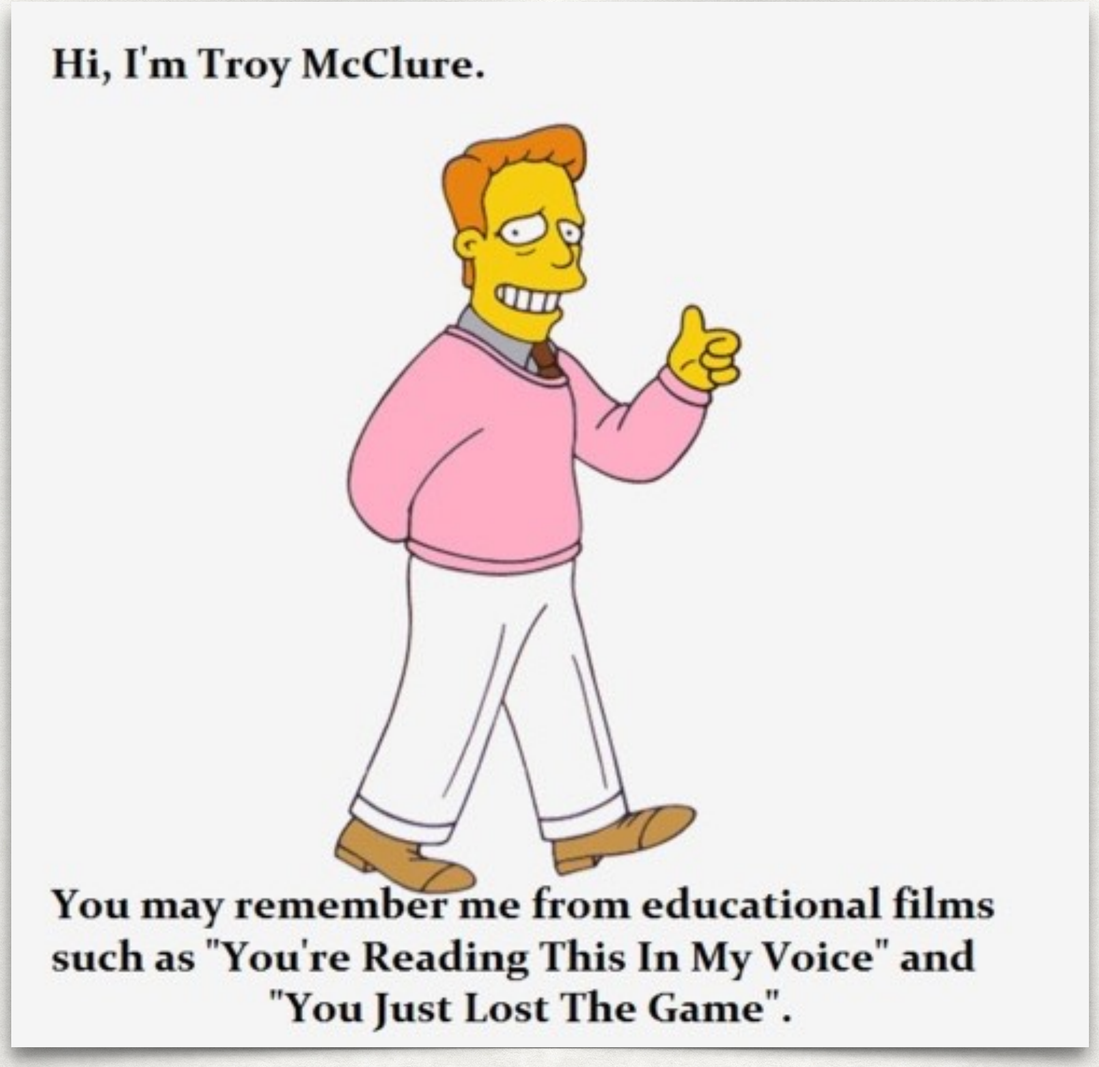
https://github.com/chenglou/react-motion

RESPONSIVE WEB AND MOBILE USER INTERFACES

https://react-bootstrap.github.io/

https://facebook.github.io/react-native/

Part 2



So you've Decided to Learn JavaScript

MDN IS YOUR FRIEND

https://developer.mozilla.org/en-US/docs/Web/JavaScript

YOU'LL PROBABLY WANT TO USE BABEL

- Babel compiles Javascript into Javascript (e.g. ES6 > ES5)
- ES6 is much closer to Python
- You can get ES5 support with babel-polyfill (older browsers)
- You can pre-compile React JSX (if you use React)

https://babeljs.io/

BOTH LANGUAGES ARE MULTI-PARADIGM

Python

- Functions are first class
- Objects are first class
- Multiple Inheritance
- Single-line lambdas
- Reference count garbage collection

Javascript

- Functions are first class
- Objects (only in ES6+) “de-sugar” to a function call
- Single Inheritance (ES6+)
- Prototype Inheritance
- Multi-line lambdas (just inline functions)
- Generational garbage collection

TYPING IS DIFFERENT

Python

- Duck-typed variables ("if it looks like a duck and quacks like a duck, it's a duck")
- Type sensitive dictionary keys
- Integer, fixed point, and floating point numbers
- Basic types are:
  - tuple and list
  - dict
  - bool
  - int, float, long, complex
  - str + unicode in 2, bytes + str in 3
  - function
  - generator
  - object
  - None

Javascript

- Loose-typed variables, often break down to strings
- Object's (dict equivalent) "stringify" their keys
- Only floating point numbers
- Basic types are:
  - array
  - object
  - string
  - boolean
  - number
  - null
  - undefined

COMMUNITY STANDARDS AND RESOURCES

Python

- Python Enhancement Proposal (PEP) Process
- PEP 8 - Style Guide
- Zen of Python (Abstract Style Guide)
- Batteries Included
- Python Package Index (PyPI)

Javascript

- TC39 proposes standards
- No general style guide
  - Semi-colons vs. not (jsmin vs uglify)
  - C-style spacing and indenting
- No batteries
  - Lots of little single function libraries
- Node Package Manager (NPM)

ASIDE: NPM: THE LEFT-PAD DEBACLE

- The library left-pad was removed from NPM
- It padded strings with spaces on the left hand side (a.k.a right justify)
  - Python Equivalent: ‘hi'.rjust(10) # ‘ hi’
- The author unpublished it and 250 other modules from NPM over a dispute with a company called Kik
- Left-pad was downloaded 2,486,696 times in May 2016 (the time of the event)
  - MATH: 1.41 KB * 2,486,696 == ~3,424 MB of bandwidth
- It’s removal broke thousands of development and deployment installations worldwide

http://www.theregister.co.uk/2016/03/23/npm_left_pad_chaos/
http://arstechnica.com/information-technology/2016/03/rage-quit-coder-unpublished-17-lines-of-javascript-and-broke-the-internet/

DECLARING VARIABLES

Python

```python
num = 0
obj = str("0")
str_ = "0"
```

Javascript

```javascript
// ES5 - Global scope
global = 0;

// ES5 - Block scope
var num = 0;
var str = "0";
var obj = new String("0");

// ES6
let numOne = 0;
const numTwo = 0;
```

BLOCK SCOPING

Python

```python
def scope():
    print 'in function'

print 'not in function'
```

Whitespace and indentation matter

Javascript

```javascript
function scope() {
  console.log('in function');
}

console.log('not in function');
```

Whitespace and indentation does not matter

FUNCTIONS

Python

```python
def traditional():
    return 5

lambda_ = lambda: 5
```

Javascript ES6

```javascript
function traditional() {
  return 5;
}

var lambda = function() {
  return 5;
};

// ES6+
let arrow = () => 5;
```

LOOPING ARRAY

Python

```python
numbers = [1, 2, 3]

for number in numbers:
    print number


for i in range(len(numbers)):
    print numbers[i]
```

Javascript ES6

```javascript
let numbers = [1, 2, 3];

// ES6 object specific iterator loop
// (most similar to Python)
for (let number of numbers) {
  console.log(number);
}

// Iterate enumerable properties
for (var i in numbers) {
  if (letters.hasOwnProperty(key)) {
    console.log(numbers[i]);
  }
}

// C++ style index loop
for (var i = 0; i < numbers.length; i++) {
  console.log(numbers[i]);
}

// forEach with lambda
numbers.forEach(function (number) {
  console.log(number);
})
```

LOOPING OBJECT

Python

```python
letters = {"a": 1, "b": 2}

for key in letters.iterkeys():
    print key

for value in letters.itervalues():
    print value

for key, value in letters.iteritems():
    print key, value
```

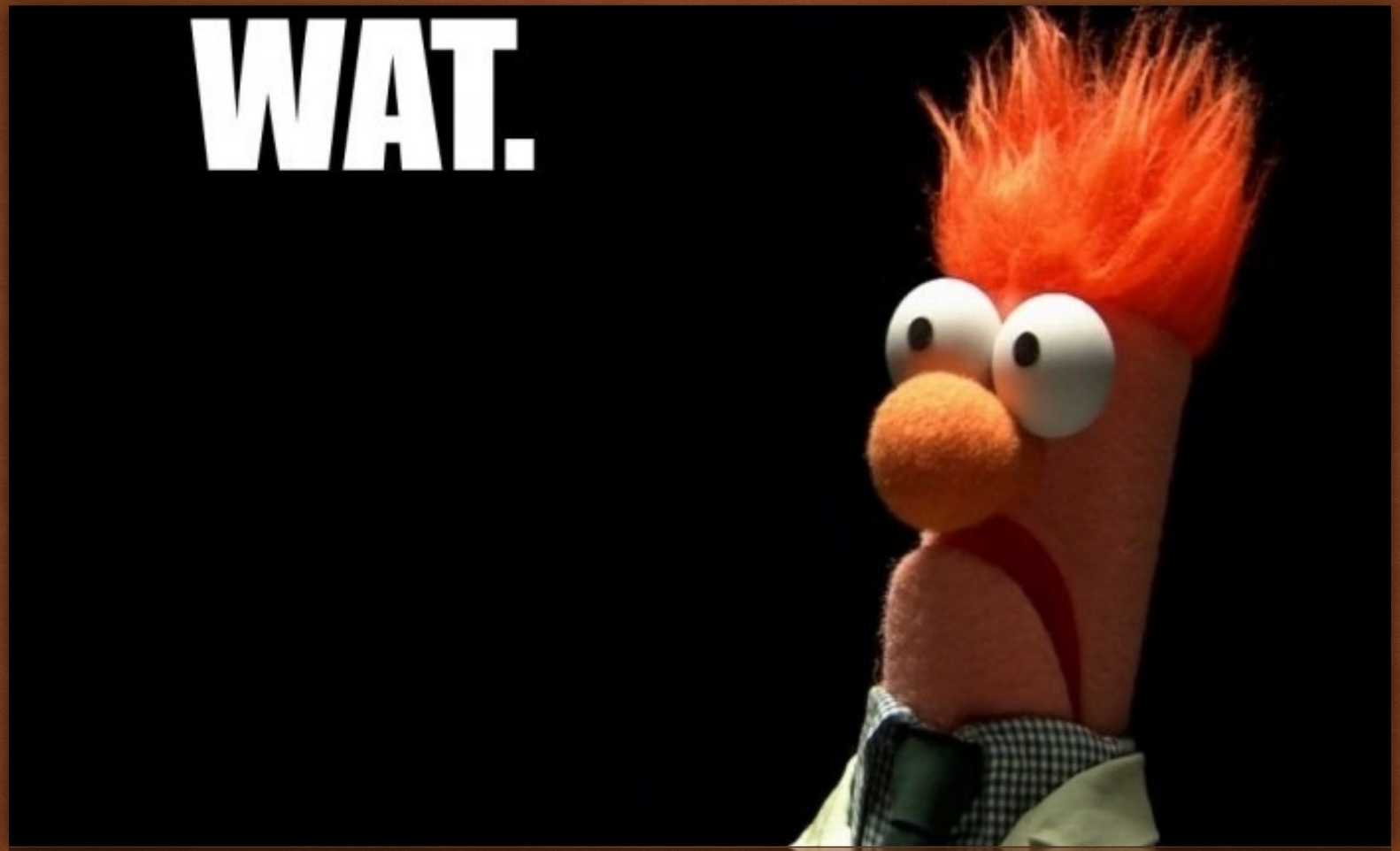
Javascript

```javascript
let letters = { a: 1, b: 2 };

// Old object iteration
var keys = Object.keys(letters);
for (let i = 0; i < keys.length; i++) {
  let key = keys[i];
  let value = letters[key];
  console.log(key, value);
}

// for .. in iteration
for (let key in letters) {
  if (letters.hasOwnProperty(key)) {
    let value = letters[key];
    console.log(key, value);
  }
}

// for .. of iteration with ES5 Map
let letters = new Map([["a", 1], ["b", 2 ]]);
for (let [key, value] of letters) {
  console.log(key, value);
}
```

LOOPING OBJECT (CONT...)

Javascript

```
let letters = { a: 1, b: 2 };

Object.keys(letters).forEach(function (key) {
  var value = letters[key];
  console.log(key, value);
});
```


WAT.

LOOPING OBJECT (CONT...)

Javascript to the Future

```
let letters = { a: 1, b: 2 };

// ES2017 for .. of iteration with Object.keys
for (let key of Object.keys(letters)) {
  console.log(key);
}

// ES2017 for .. of iteration with Object.values
for (let value of Object.values(letters)) {
  console.log(value);
}

// ES2017 for .. of iteration with Object.entries
for (let [key, value] of Object.entries(letters)) {
  console.log(key, value);
}
```

ARRAY SLICING

Python

```python
numbers = [1, 2, 3, 4]

print numbers[0:1]      # [1]
print numbers[0:10]     # [1, 2, 3, 4]
print numbers[2:3]      # [3]
print numbers[3:2:-1]   # [4]
```

Javascript

```javascript
let numbers = [1, 2, 3, 4];

console.log(numbers.slice(0, 1));   // [ 1 ]
console.log(numbers.slice(0, 10));  // [ 1, 2, 3, 4]
console.log(numbers.slice(2, 3));   // [ 3 ]
console.log(numbers.slice(2, 3));   // [ 3 ]
console.log(numbers.slice(3, 2, -1));   // No stepping (but no argument error)
```

ARRAY SORTING

Python

```python
numbers = [1, 2, 3, 4]

print sorted(numbers)   # [1, 2, 3, 4]
print sorted(numbers, reverse=True)  # [4, 3, 2, 1]
```

Javascript

```javascript
let numbers = [1, 2, 3, 4];

console.log(numbers.sort());  // [ 1, 2, 3, 4 ]
console.log(numbers.sort().reverse());  // [ 4, 3, 2, 1 ]
```

ARRAY SORTING (CONT...)

Python

```python
words = ["a", "be", "the"]
print sorted(words, key=lambda word: len(word))  # ["a", "be" "the"]
print sorted(words, key=lambda word: len(word), reverse=True)  # ["the", "be", "a"]

letters = ["c", "a", "b"]
print sorted(letters, key=lambda letter: len(letter))  # Stable
```

Javascript

```javascript
let words = ["a", "be", "the"];
function wordCompare(a, b) {
  return a.length - b.length;
}
console.log(words.sort(wordCompare)); // [ "a", "be", "the" ]
console.log(words.sort(wordCompare).reverse()); // [ "the", "be", "a" ]

// Not guaranteed stable
let letters = ["c", "a", "b"];
console.log(letters.sort(wordCompare));
```

SHALLOW DICT/OBJECT COPY

Python

```python
letters = {"a": 1, "b": 2}

other_letters = dict(letters)

print letters == other_letters  # True
print letters is not other_letters  # True
```

Javascript

```javascript
let letters = { a: 1, b: 2 };
let otherLetters = Object.assign({}, letters);

console.log(JSON.stringify(letters) == JSON.stringify(otherLetters));   // true
console.log(letters !== otherLetters);   // true
```

SHALLOW DICT/OBJECT COPY

Javascript

```
Object.equals = function( x, y ) {
  if ( x === y ) return true;
    // if both x and y are null or undefined and exactly the same

  if ( ! ( x instanceof Object ) || ! ( y instanceof Object ) ) return false;
    // if they are not strictly equal, they both need to be Objects

  if ( x.constructor !== y.constructor ) return false;
    // they must have the exact same prototype chain, the closest we can do is
    // test there constructor.

  for ( var p in x ) {
    if ( ! x.hasOwnProperty( p ) ) continue;
      // other properties were tested using x.constructor === y.constructor

    if ( ! y.hasOwnProperty( p ) ) return false;
      // allows to compare x[ p ] and y[ p ] when set to undefined

    if ( x[ p ] === y[ p ] ) continue;
      // if they have the same strict value or identity then they are equal

    if ( typeof( x[ p ] ) !== "object" ) return false;
      // Numbers, Strings, Functions, Booleans must be strictly equal

    if ( ! Object.equals( x[ p ],  y[ p ] ) ) return false;
      // Objects and Arrays must be tested recursively
  }

  for ( p in y ) {
    if ( y.hasOwnProperty( p ) && ! x.hasOwnProperty( p ) ) return false;
      // allows x[ p ] to be set to undefined
  }
  return true;
}
```

http://stackoverflow.com/a/6713782

BEWARE VARIABLE SCOPING WITH JAVASCRIPT

Python

```python
if True:
  x = 5

print x  # x is 5
```

Javascript

```javascript
x = 5;
console.log(x);  // 5
var x;


if (true) {
  var y = 5;
}
console.log(y);  // y is 5


// ES6 only
if (true) {
  let z = 5;
}
// ReferenceError: z is not defined
console.log(z);
```

https://developer.mozilla.org/en/docs/Web/JavaScript/Reference/Statements/var#var_hoisting

BEWARE VARIABLE SCOPING WITH JAVASCRIPT

Python

```python
test = "I'm global"

def testScope():
  test = "I'm local"
  print test

print test    # output: I'm global
testScope()   # output: I'm local
print test    # output: I'm global
```

Javascript

```javascript
var test = "I'm global";

function testScope() {
  var test = "I'm local";
  console.log(test);
}

console.log(test); // I'm global
testScope();       // I'm local
console.log(test); // I'm global
```

BEWARE VARIABLE SCOPING WITH JAVASCRIPT (CONT...)

Python

```python
test = "I'm global"

def testScope():
  global test
  test = "I'm local"
  print test

print test    # output: I'm global
testScope()   # output: I'm local
print test    # output: I'm local
```

Javascript

```javascript
var test = "I'm global";

function testScope() {
  test = "I'm local";
  console.log(test);
}

console.log(test); // I'm global
testScope();       // I'm local
console.log(test); // I'm local
```

https://developer.mozilla.org/en/docs/Web/JavaScript/Reference/Statements/var#var_hoisting

PYTHON EQUALITY

Value Equality

```
num = 0
obj = str("0")
str_ = "0"

print num == num     # True
print obj == obj     # True
print str_ == str_   # True

print num == obj    # False
print num == str_   # False
print obj == str_   # True
print obj == None   # False
```

Reference Equality

```
num = 0
obj = str("0")
str_ = "0"

print num is num     # True
print obj is obj     # True
print str_ is str_   # True

print num is obj     # False
print num is str_    # False
print obj is str_    # True
print obj is None    # False
```

JAVASCRIPT EQUALITY

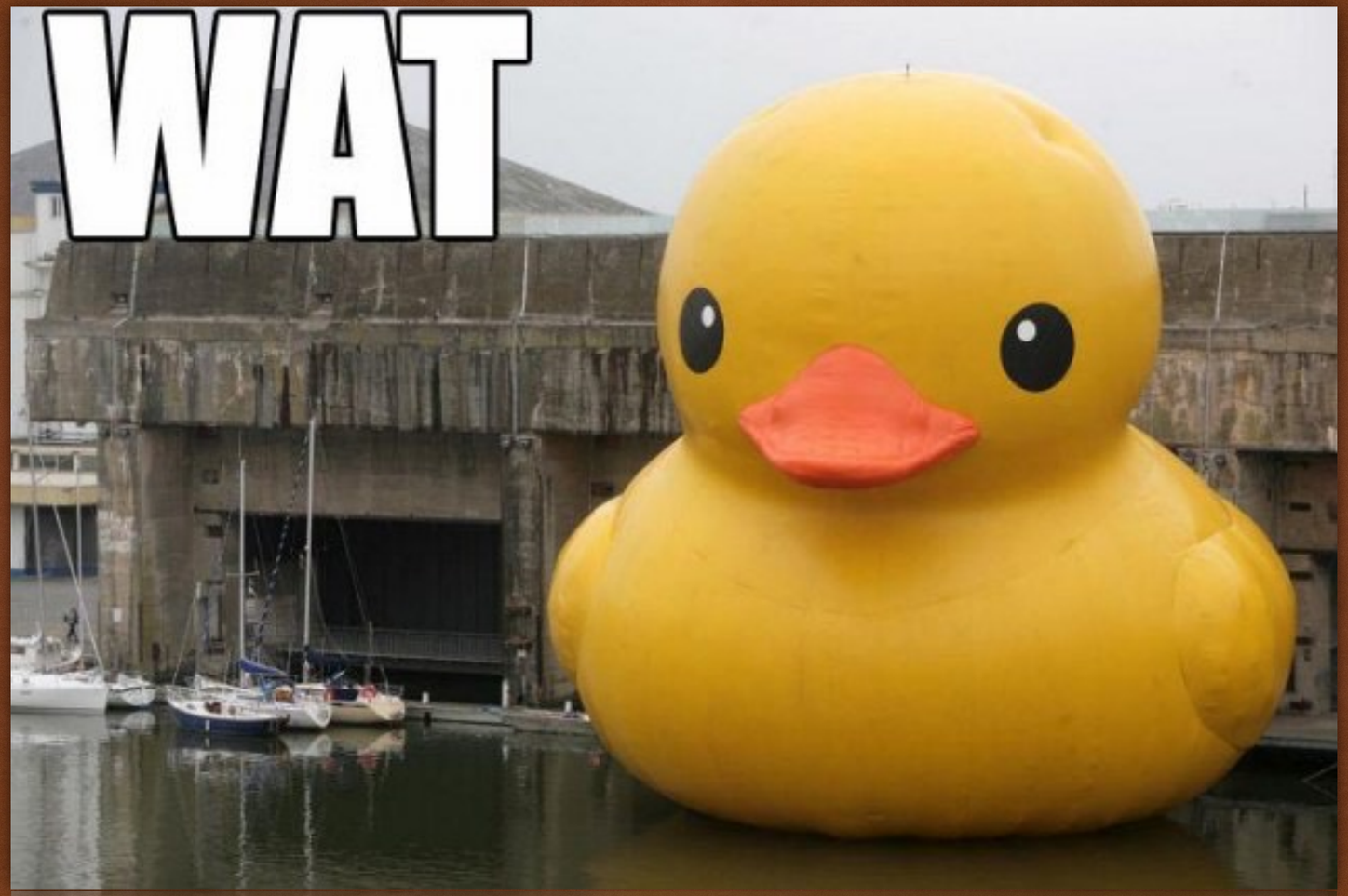
Strict Equality

```
var num = 0;
var obj = new String("0");
var str = "0";

console.log(num === num); // true
console.log(obj === obj); // true
console.log(str === str); // true

console.log(num === obj); // false
console.log(num === str); // false
console.log(obj === str); // false
console.log(null === undefined); // false
console.log(obj === null); // false
console.log(obj === undefined); // false
```

Loose Equality

```
var num = 0;
var obj = new String("0");
var str = "0";

console.log(num == num); // true
console.log(obj == obj); // true
console.log(str == str); // true

console.log(num == obj); // true
console.log(num == str); // true
console.log(obj == str); // true
console.log(null == undefined); // true

// both false, except in rare cases
console.log(obj == null);
console.log(obj == undefined);
```

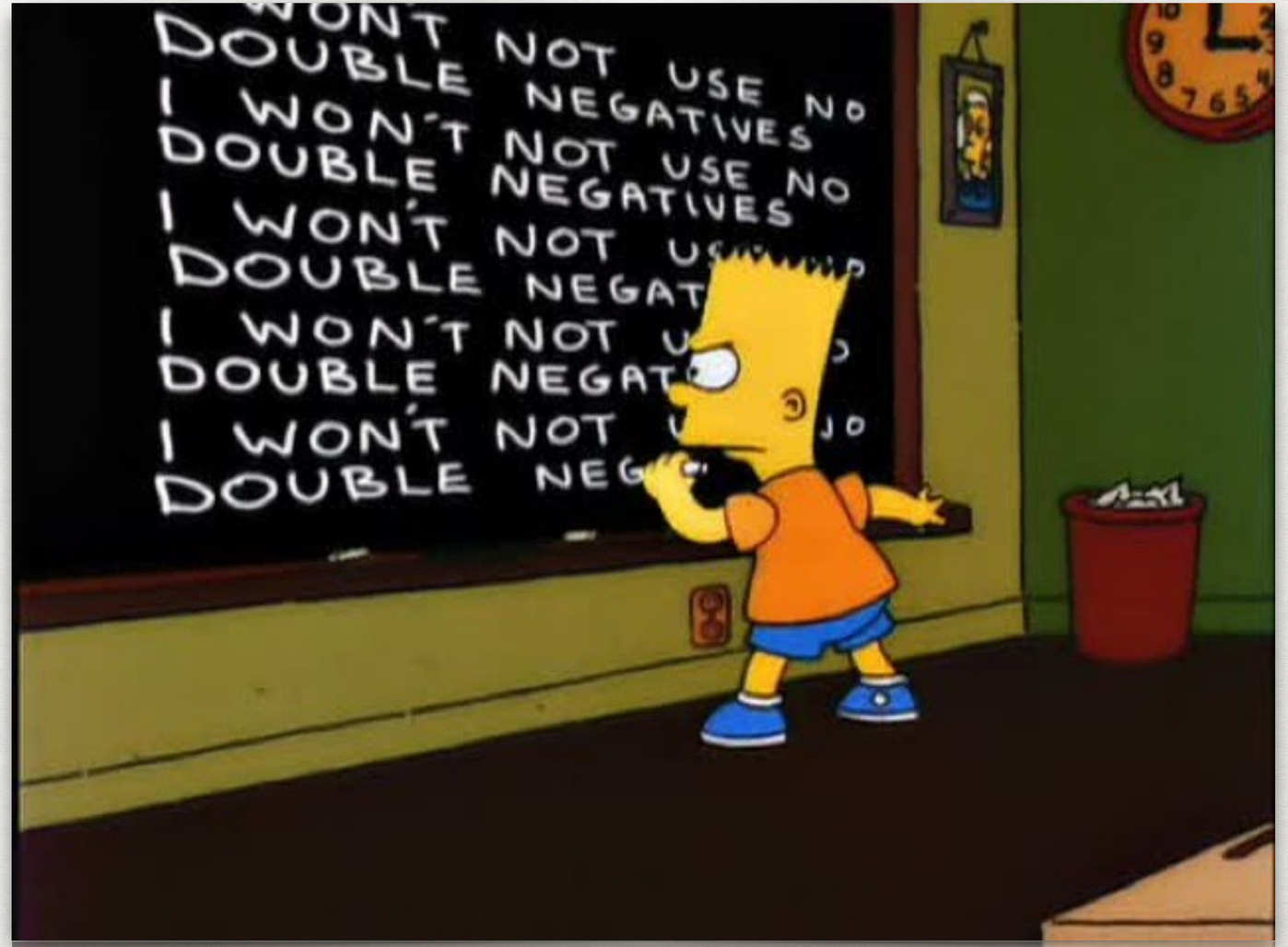
Loose equality compares two values for equality, after converting both values to a common type.


WAT

JAVASCRIPT EQUALITY

Double Not, A.K.A !!

```
let isThing = true;

if (!isThing) {
  console.log('Not is thing')
}

if (!!isThing) {
  console.log("Is thing");
}
```



ATTRIBUTES VS KEYS

Python

```python
class MyDict(dict):
    pass

# a = dict() # Would fail.
# Can not attach attributes to
# a basic type.

a = MyDict()
a['test'] = 1
a.test = 2

print a['test']  # 1
print a.test     # 2
```

Javascript

```javascript
var a = {};
a['test'] = 1;
a.test = 2;

console.log(a['test']);  // 2
console.log(a.test);     // 2
```

KEYS AND ERRORS

Python

```python
a = {}
print a['test']  # raises KeyError
```

Javascript

```javascript
var a = {};
console.log(a['test']);  // undefined
```

Generally speaking, Python raises more errors and Javascript silences and continues

IMPORTS

Python

```python
import math
print math.log(42)

from math import log
print log(42)

# not a good practice
from math import *
print log(42)
```

Javascript ES6

```javascript
import math from 'math';
console.log(math.log(42));

import { log } from 'math';
console.log(log(42));

import * from 'math';
console.log(log(42));
```

IMPORTS (COMMON.JS - NODE)

```
var math = require('math');
console.log(math.log(42));
```

IMPORTS (AMD)

```
define(
  'math',
  ['log'],
  function (log) {
    console.log(log(42));
  }
);
```

SPREAD SYTNAX

Python

```python
def forward(func, *args, **kwargs):
    return func(*args, **kwargs)
```

Javascript ES6

```javascript
function forward(func, ...args) {
  return func(...args);
}
```

DESTRUCTURING

Python

```
status, data = getResult()
```

Javascript ES6

```
var [status, data] = getResult();
```

```
var {status, data} = { data: 1, status: 2}
```

OBJECTS

Python

```python
class MyObject(object):

    def __init__(self, value, optional=None):
        self.value = value
        self.optional = optional


instance = MyObject(5, optional=True)
```

Javascript ES6

```javascript
class MyObject {

  constructor(value, options = {}) {
    let { optional } = options;

    this.value = value;
    this.optional = optional;
  }
}


let instance = new MyObject(5, { optional: true });
```

OBJECT PROPERTIES

Python

```python
class MyObject(object):

    def __init__(self, value):
        self._value = value

    @property
    def value(self):
        return self._value

    @value.setter
    def value(self, new_value):
        self._value = new_value


instance = MyObject(5)
print instance.value  # 5
instance.value = 10
print instance.value  # 10
```

Javascript ES6

```javascript
class MyObject {

  constructor(value) {
    this._value = value;
  }

  get value() {
    return this._value;
  }

  set value(new_value) {
    this._value = new_value;
  }
}


let instance = new MyObject(5);
console.log(instance.value);  // 5
instance.value = 10;
console.log(instance.value);  // 10
```

OBJECT CLASS VARIABLES

Python

```python
class MyObject(object):

    value = 5

instance = MyObject()
print instance.__class__.value   # 5
instance.__class__.value = 10
print instance.__class__.value   # 10

print instance.value   # 10
```

Javascript ES6

```javascript
let myObjectValue = 5;

class MyObject {

  static get value() {
    return myObjectValue;
  }

  static set value(newValue) {
    myObjectValue = newValue;
  }
}


let instance = new MyObject();
console.log(instance.constructor.value);  // 5
instance.constructor.value = 10;
console.log(instance.constructor.value);  // 10

console.log(instance.value);  // undefined
```

OBJECT METHOD CALLBACKS

Python

```python
class MyObject():

    def __init__(self, value):
        self.value = value

    def get_value(self):
        return self.value


def call(func):
    print func()


instance = MyObject(4)
callback = instance.get_value
call(callback)
```

Javascript ES6

```javascript
class MyObject {

  constructor(value) {
    this.value = value;
  }

  getValue() {
    return this.value;
  }
}

function call(func) {
  console.log(func());
}


let instance = new MyObject(4);
let callback = instance.getValue;

// TypeError: Cannot read property 'value' of undefined;
call(callback);
```

wat

OBJECT METHOD CALLBACKS (CONT...)

Javascript (Manual Bind)

```
class MyObject {

  constructor(value) {
    this.value = value;
  }

  getValue() {
    return this.value;
  }
}

function call(func) {
  console.log(func());
}


let instance = new MyObject(4);
let callback = instance.getValue.bind(instance);
call(callback); // 4
```

Javascript (Auto Bind decorator)

```
class MyObject {

  constructor(value) {
    this.value = value;
  }

  @autobind
  getValue() {
    return this.value;
  }
}

function call(func) {
  console.log(func());
}


let instance = new MyObject(4);
let callback = instance.getValue;
call(callback); // 4
```

INSTANCE (SELF/THIS) REFERENCING IN CLOSURES

Python

```python
class AdderFactory(object):

    def __init__(self, a):
        self.a = a

    def get_add(self):
        def _add(b):
            return self.a + b
        return _add


factory = AdderFactory(5)
add = factory.get_add()
print add(3)  # 8
```

Javascript ES6

```javascript
class AdderFactory {

  constructor(a) {
    this.a = a;
  }

  getAdd() {
    function _add(b) {
      return this.a + b;
    }
    return _add;
  }
}

let factory = new AdderFactory(5);
let add = factory.getAdd();

// TypeError: Cannot read property a of undefined
add(3);
```

WAT.

INSTANCE (SELF/THIS) REFERENCING IN CLOSURES

Javascript (Store this in that)

```
class AdderFactory {

  constructor(a) {
    this.a = a;
  }

  getAdd() {
    var that = this;
    function _add(b) {
      return that.a + b;
    }
    return _add;
  }
}

let factory = new AdderFactory(5);
let add = factory.getAdd();
console.log(add(3)); // 8
```

Javascript (ES6 Arrow Functions)

```
class AdderFactory {

  constructor(a) {
    this.a = a;
  }

  getAdd() {
    let _add = (b) => {
      return this.a + b;
    };
    return _add;
  }
}

let factory = new AdderFactory(5);
let add = factory.getAdd();
console.log(add(3)); // 8
```

GENERATORS

Python

```python
def generator():
    yield 1
    yield 2
    yield 3

for i in generator():
    print i
```

Javascript ES6

```javascript
function *generator() {
    yield 1;
    yield 2;
    yield 3;
}

for (let i of generator()) {
  console.log(i);
}
```

ITERATORS

Python

```python
class MyObject(object):

    def __iter__(self):
        yield 1
        yield 2
        yield 3


instance = MyObject()
for i in instance:
    print i
```

Javascript ES6

```javascript
var myIterable = {};
myIterable[Symbol.iterator] = function* () {
    yield 1;
    yield 2;
    yield 3;
};

for (let i of myIterable) {
  console.log(i);
}
```

ITERATORS (CONT...)

Python

```python
class MyObject(object):

    def __init__(self, initial=None):
        self.storage = initial or []

    def __iter__(self):
        return iter(self.storage)


instance = MyObject([1, 2, 3])
for i in instance:
    print i
```

Javascript ES6

```javascript
class MyObject {
  constructor(initial=[]) {
    this.storage = initial;
  }

  [Symbol.iterator]() {
    return this.storage.values()
  }
}

let it = new MyObject([1, 2, 3]);
for (let i of it) {
  console.log(i);
}
```

EXCEPTIONS

Python

```python
class MyException(Exception):
    """
    This is my exception
    """

# Usage
try:
    raise MyException()
except MyException as e:
    print e
```

Javascript (Function only)

```javascript
class BaseException {
  constructor(message) {
    this.message = message;

    this.name = this.constructor.name;
    this.stack = (new Error()).stack;
  }
}
BaseException.prototype = Object.create(Error.prototype);
BaseException.prototype.constructor = BaseException;


class MyException extends BaseException {}

// Usage
try {
  throw new MyException();
} catch (e) {
  if (e instanceof MyException) {
    console.log(e);
  } else {
    throw e;
  }
}
```

FUNCTION DECORATORS

Python

```python
from functools import wraps

def memoize(func):
    @wraps(func)
    def _memoize(*args):
        try:
            cache = getattr(func, '_memoize')
        except AttributeError:
            cache = {}
            setattr(func, '_memoize', cache)

        try:
            return cache[args]
        except KeyError:
            result = func(*args)
            cache[args] = result
            return result
    return _memoize


@memoize
def my_function(arg):
    return arg
```

Javascript (Function only)

```javascript
function memoize(func) {
  var memo = {};
  var slice = Array.prototype.slice;

  return function() {
    var args = slice.call(arguments);

    if (args in memo)
      return memo[args];
    else
      return (memo[args] = func.apply(this, args));

  }
}

var myFunction = memoize(function(arg) {
  return arg;
});
```

METHOD DECORATORS

Python

```python
class MyClass(object):

    @memoize
    def my_method(self, arg):
        return arg

    @property
    @memoize
    def my_property(self):
        return self.value
```

Javascript (Non-Standard Class Only)

```javascript
class MyClass {
  @memoize
  myMethod(arg) {
    return arg;
  }

  @memoize
  get myProperty() {
    return this.value;
  }
}
```

METHOD DECORATORS (CONT...)

```
function decorator(target, key, descriptor) {
  return descriptor;
}


function autobind(target, key, descriptor) {
  let fn = descriptor.value;
  return {
    configurable: true,
    get() {
      if (this === target.prototype || this.hasOwnProperty(key)) {
        return fn;
      }

      let boundCb = fn.bind(this);
      Object.defineProperty(this, key, {
        value: boundCb,
        configurable: true,
        writable: true
      });
      return boundCb;
    }
  };
}
```

METHOD DECORATORS (CONT...)

Javascript (Non-Standard Class Only)

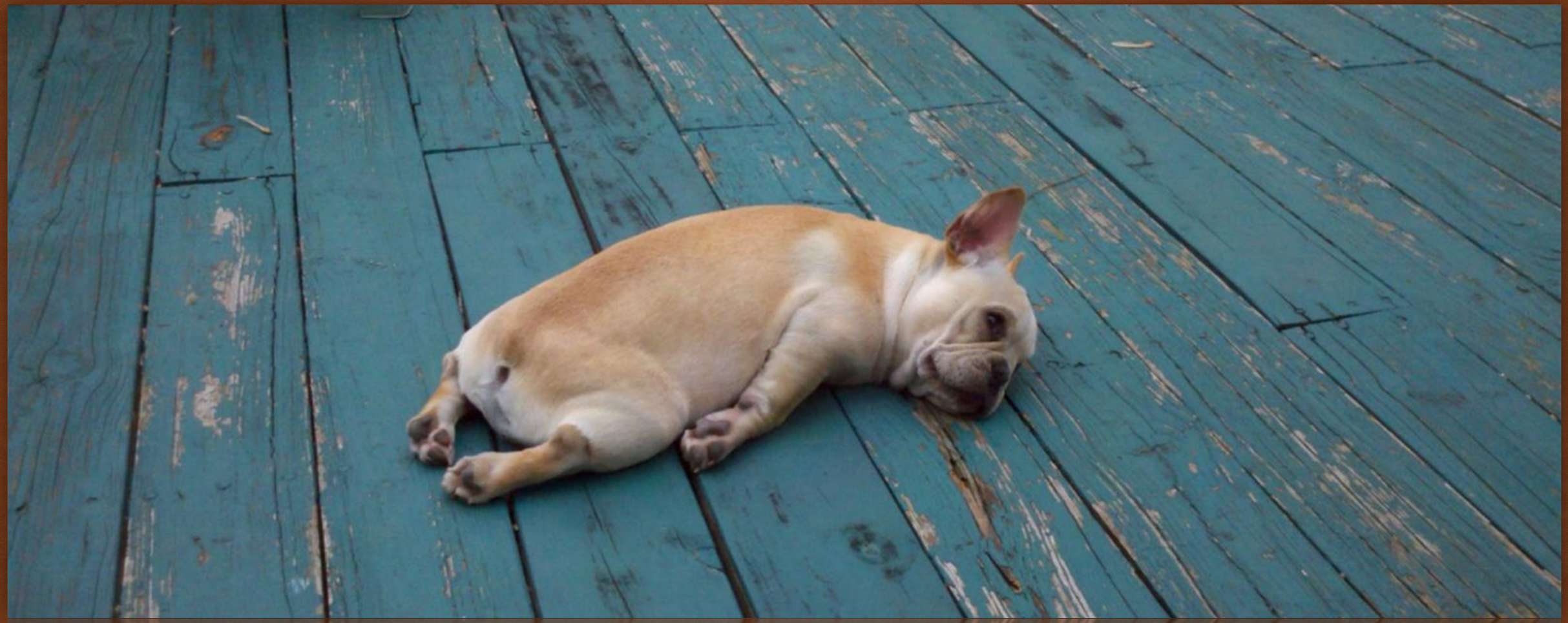
```javascript
/*
 * Bind a method and then also memoize its results
 */
export function memoize(target, key, descriptor) {
  let fn = descriptor.value;
  if ( fn !== undefined) {
    return _memoizeMethod(target, key, descriptor)
  } else {
    return _memoizeProperty(target, key, descriptor)
  }
}

/*
 * Function for producing a compound key of function arguments to
 * be used for memoizing functions
 */
function _compoundKey(args) {
  let result = [];
  args.forEach( arg => {
    if (typeof arg  === 'number' || typeof arg === 'string' ||
typeof arg === 'boolean') {
      result.push(arg);
    } else if ( Array.isArray(arg)) {
      result.push(JSON.stringify(arg))
    } else {
      // Assume object if not number, string, boolean, or array.
      result.push(JSON.stringify(arg))
    }
  });
  return result;
}
```

```javascript
/*
 * Memoize the result of a property using a WeakMap on the object instance as the
 * basis for the eviction of the cache.
 */
function _memoizeProperty(target, key, descriptor) {

  // Create memoize function that is customized to the behaviour of a property
  function _memoize(func, cacheKey) {
    return function _memoizeInner() {
      let instance = this;
      let cacheMap = instance._memoized;
      if (cacheMap === undefined) {
        cacheMap = Map();
      }

      let result;
      if (cacheMap.has(cacheKey)) {
        result = cacheMap.get(cacheKey);
      } else {
        result = func.apply(instance);
        cacheMap = cacheMap.set(cacheKey, result);
        instance._memoized = cacheMap;
      }
      return result;
    }
  }

  // Replace the property with the new memoized one
  descriptor.get = _memoize(descriptor.get, key);
  return descriptor;
}
```

METHOD DECORATORS (CONT...)

```
/*
 * Memoize the result of a method call using a WeakMap on the object instance as the basis for the eviction of
 * the cache.
 */
function _memoizeMethod(target, key, descriptor) {
  let fn = descriptor.value;
  return {
    configurable: true,
    get() {
      if (this === target.prototype || this.hasOwnProperty(key)) {
        return fn;
      }

      // Create memoize function that is customized to the behaviour of a method call with arguments as keys
      function _memoize(func, instance, cacheKey) {
        let slice = Array.prototype.slice;
        return function _memoizeInner() {
          let cacheMap = instance._memoized;
          if (cacheMap === undefined) {
            cacheMap = Map();
          }

          let args = slice.call(arguments);
          let argsCacheKey = JSON.stringify(args);

          let result;

          // Return cached result or build the new result.  Intentionally uses hasIn( ... ) to avoid having to set a
          // sentinel value.
          if (cacheMap.hasIn([cacheKey, argsCacheKey])) {
            result = cacheMap.getIn([cacheKey, argsCacheKey]);

          } else {
            // Run function
            result = func.apply(instance, args);

            // Save result
            if (!cacheMap.has(cacheKey)) {
              cacheMap = cacheMap.set(cacheKey, Map({[argsCacheKey]: result}));
            } else {
              cacheMap = cacheMap.setIn([cacheKey, argsCacheKey], result)
            }
            instance._memoized = cacheMap;
          }
          return result;
        }
      }

      // Replace the method with the new memoized one
      let boundMemoizeCb = _memoize(fn, this, key);
      Object.defineProperty(this, key, {
        value: boundMemoizeCb,
        configurable: true,
        writable: true
      });
      return boundMemoizeCb;
    }
  };
}
```

WHO'S TIRED?



SYNTAX SUMMARY

- Death by a thousand Wats
- None of the individual gotcha are too bad
  - … but there are quite a few individual gotchas
- The legacy of JS can weigh it down
- Not always clear how to integrate the new ways with existing libraries
- Often the answer is to use plugins for a tool like Babel or WebPack to give you the syntax extensions you want. (But that’s not always easy to add)

PACKAGING CODE

PACAKGING

Python (PyPI + pip)

$ pip install flask

$ pip uninstall flask

$ pip install -r requirements.txt

Javascript (npm)

$ npm install webpack

$ npm uninstall webpack

$ npm install (Looks at package.json)

PACKAGE.JSON

```
{
  "name": "mylibrary",
  "version": "1.0.0",
  "description": “Really Cool Library",
  "repository": {
    "type": "git",
    "url": “https://github.com/dmckeone/mylibrary“
  },
  "main": "lib/index.js",
  "peerDependencies": {
    "immutable": "^3.8.1",
    "typed-immutable": ">=0.1.0"
  },
  "babel": {
    "presets": [
      "es2015",
      "stage-0",
      "stage-1",
      "stage-2"
    ],
    "plugins": [
      [
        "transform-decorators-legacy"
      ],
      [
        "transform-object-rest-spread"
      ]
    ]
  },
  "dependencies": {},
  "devDependencies": {
    "babel-cli": "^6.18.0",
    "babel-core": "^6.21.0",
    "babel-plugin-transform-decorators-legacy": "^1.3.4",
    "babel-plugin-transform-object-rest-spread": "^6.20.2",
    "babel-polyfill": "^6.20.0",
    "babel-preset-es2015": "^6.18.0",
    "babel-preset-stage-0": "^6.16.0",
    "babel-preset-stage-1": "^6.16.0",
    "babel-preset-stage-2": "^6.18.0",
    "expect": "^1.20.2",
    "immutable": "^3.8.1",
    "mocha": "^3.2.0",
    "rimraf": "^2.5.4",
    "typed-immutable": ">=0.1.0"
  },
  "scripts": {
    "clean": "rimraf lib",
    "build": "npm run clean && NODE_ENV=production babel src --out-dir lib",
    "test": "mocha --compilers js:babel-core/register 'test'"
  },
  "keywords": [],
  "author": “David McKeone",
  "license": "BSD",
  "private": true
}
```

PACKAGE.JSON - RUN

```
"scripts": {
    "clean": "rimraf lib",
    "build": "npm run clean && NODE_ENV=production babel src --out-dir lib",
    "test": "mocha --compilers js:babel-core/register 'test'"
  },
```

$ npm run clean

$ npm run build

$ npm run test

PACKAGE.JSON - DEPENDENCIES

```
"peerDependencies": {
  "immutable": "^3.8.1",
  "typed-immutable": ">=0.1.0"
},
"dependencies": {},
"devDependencies": {
  "babel-cli": "^6.18.0",
  "babel-core": "^6.21.0",
  "babel-plugin-transform-decorators-legacy": "^1.3.4",
  "babel-plugin-transform-object-rest-spread": "^6.20.2",
  "babel-polyfill": "^6.20.0",
  "babel-preset-es2015": "^6.18.0",
  "babel-preset-stage-0": "^6.16.0",
  "babel-preset-stage-1": "^6.16.0",
  "babel-preset-stage-2": "^6.18.0",
  "expect": "^1.20.2",
  "immutable": "^3.8.1",
  "mocha": "^3.2.0",
  "rimraf": "^2.5.4",
  "typed-immutable": ">=0.1.0"
}
```

- Peer Dependencies:
  - Dependencies between plugins and your package.
- Dependencies
  - Production requirements
- Dev Dependencies:
  - Requirements for development tooling

TESTING CODE

MOCHA + EXPECT TESTS

```
import 'babel-polyfill';
import expect from 'expect';
import { add } from '../src/adder';


describe('adder', function() {
  describe('add', function() {
      it('adds two numbers', function () {
        expect(add(1, 1)).toEqual(2);
      });
  });
});
```

https://mochajs.org/

https://github.com/mjackson/expect

LIBRARY DEMO

BUT WE WANT AN APP!

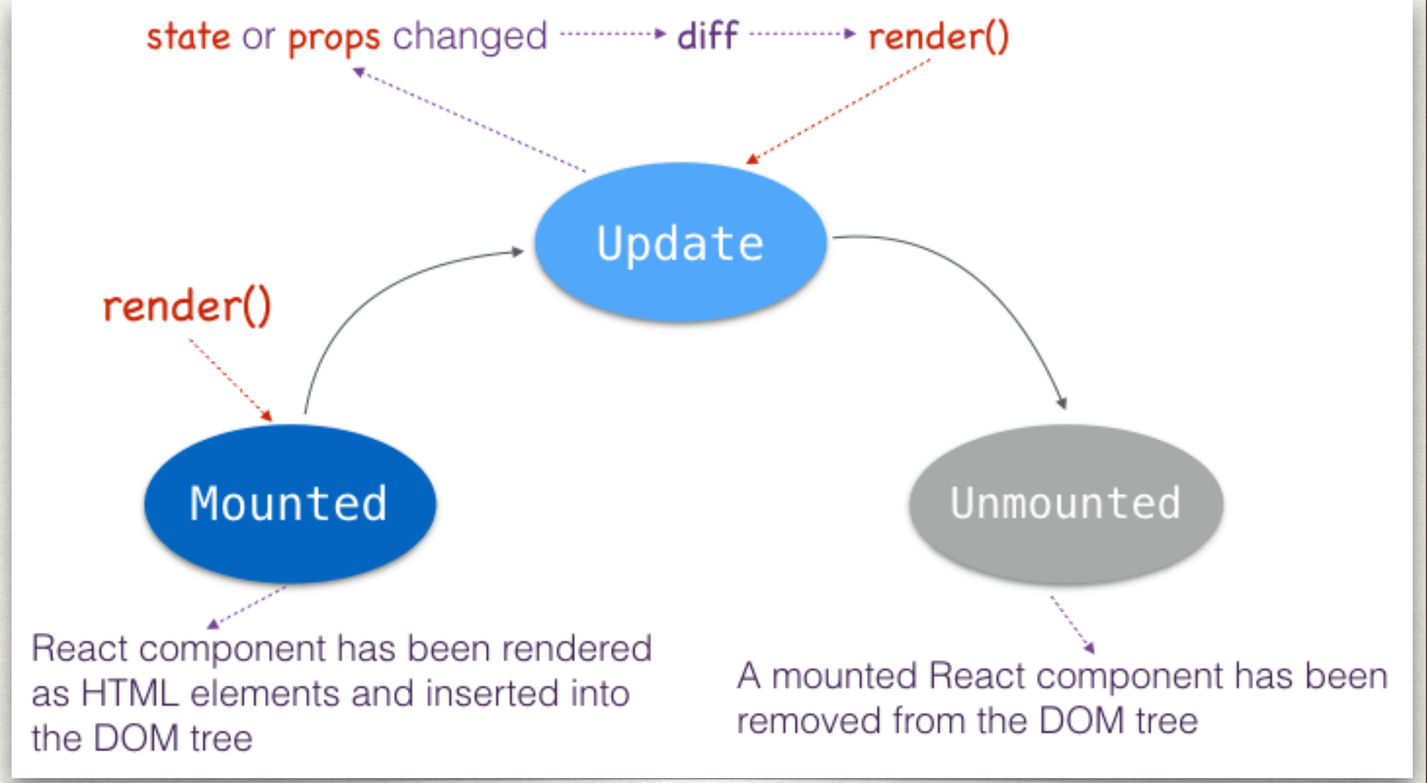
CRASH COURSE IN REACT

React

- UI Framework
- Uses Shadow DOM
- Typically uses JSX to give the UI an HTML-like feel
- Based on **state** and **props**

PROPS

```
<ListGroupItem onClick={ onClick }>
  <span style={ textStyle }>{text}</span>
</ListGroupItem>
```



https://facebook.github.io/react/

REACT COMPONENT (ES6 CLASS STYLE)

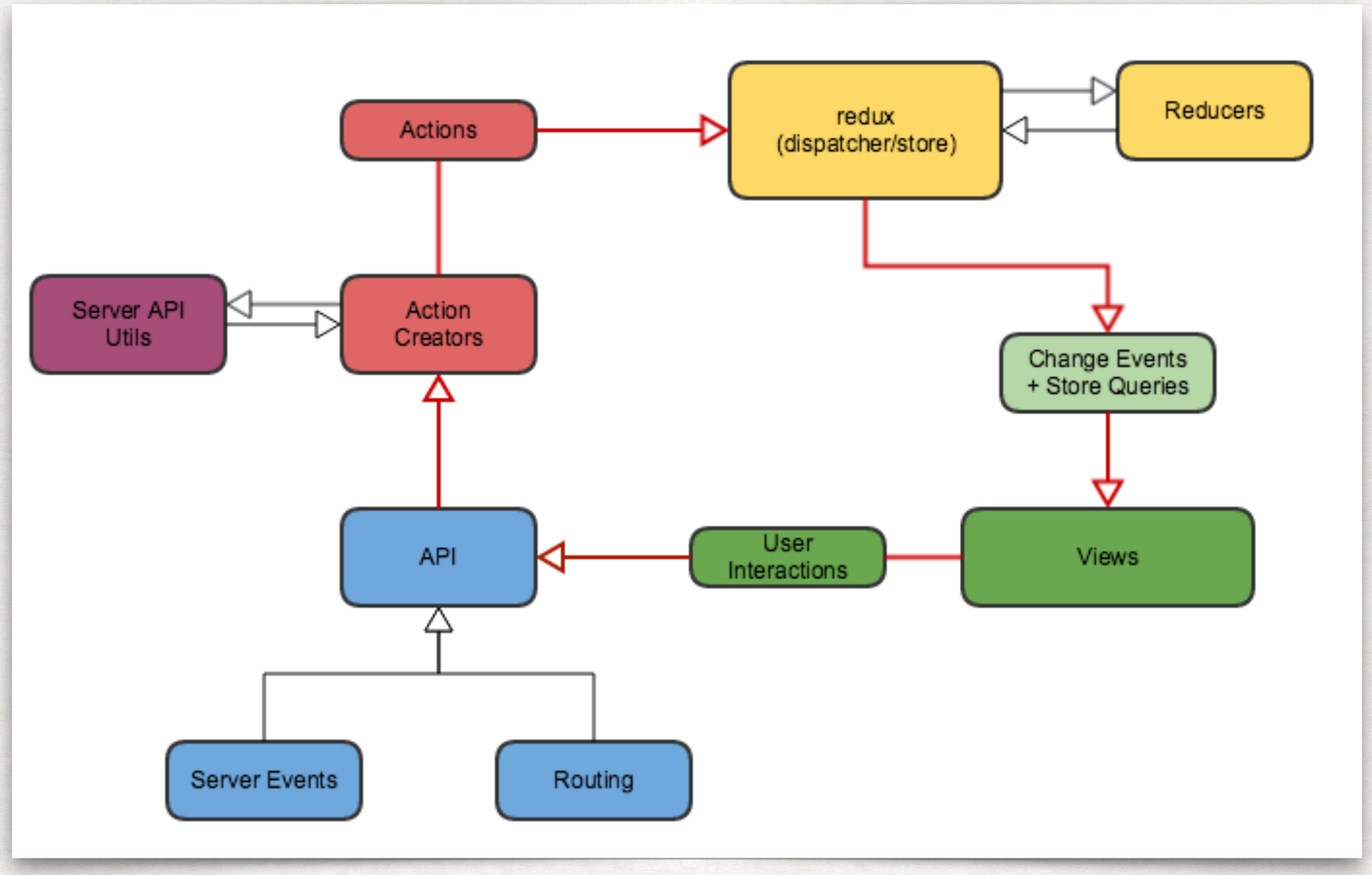
Now with fewer standards!

Main rendering function:
state + props = HTML

```
import React, { Component, PropTypes } from 'react'
import { ListGroupItem } from 'react-bootstrap';

export default class ToDoItem extends Component {

  static propTypes = {
    onClick: PropTypes.func.isRequired,
    completed: PropTypes.bool.isRequired,
    text: PropTypes.string.isRequired
  };

  render() {
    let { onClick, completed, text } = this.props;
    let textStyle = { textDecoration: completed ? 'line-through' : 'none' };
    let bsStyle = completed ? 'success' : undefined;
    return (
      <ListGroupItem onClick={ onClick } bsStyle={ bsStyle }>
        <span style={ textStyle }>{text}</span>
      </ListGroupItem>
    )
  }
}
```

STATE IS SCARY

CRASH COURSE IN REDUX



https://github.com/reactjs/redux

YOU MIGHT WANT TO USE WEBPACK

- Create bundles of your javascript
- Includes all library dependencies
- Supports module chunking (lazy load your app)
- Plugins for common use cases
- Loaders for CoffeeScript, TypeScript, CSS, SASS, etc…
- Basically like pyinstaller, but for JS (it's your "freeze" tool)

webpack
MODULE BUNDLER

https://webpack.github.io/

APP DEMO