

announcements 11-Sep-2017

vote!

poll on whether we do catering or not



help wanted: final demo day organization

do you...

know how to book rooms?

know how to make web visualizations?
(funding page)

have a car to pick stuff up?



tasty sandwiches? I'm in

msg me on slack w/ what you can help
with. I'll add to our channel



CS 4241

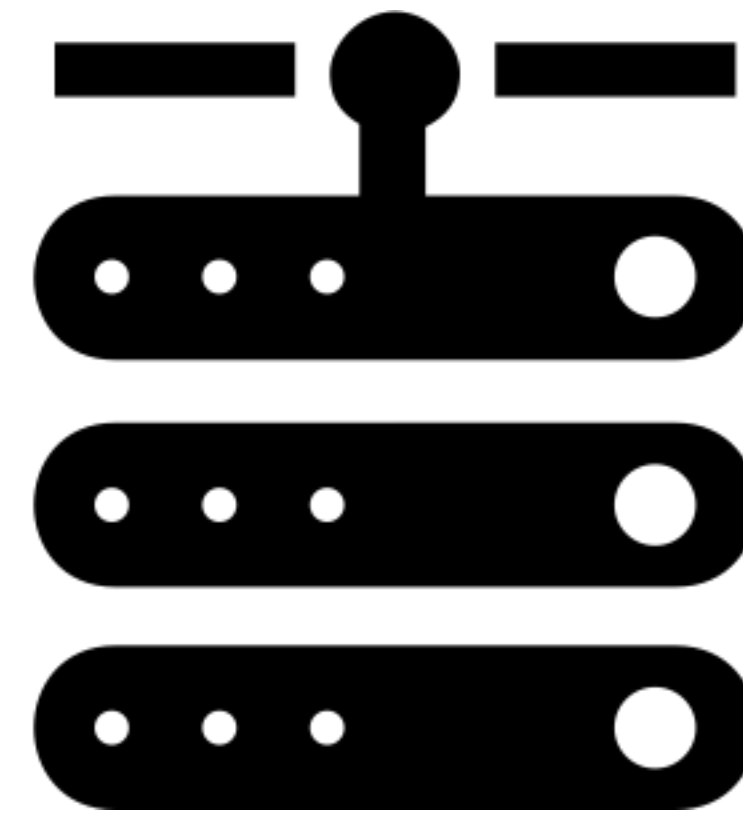
WebWare

database, persistence layer

browsers



servers



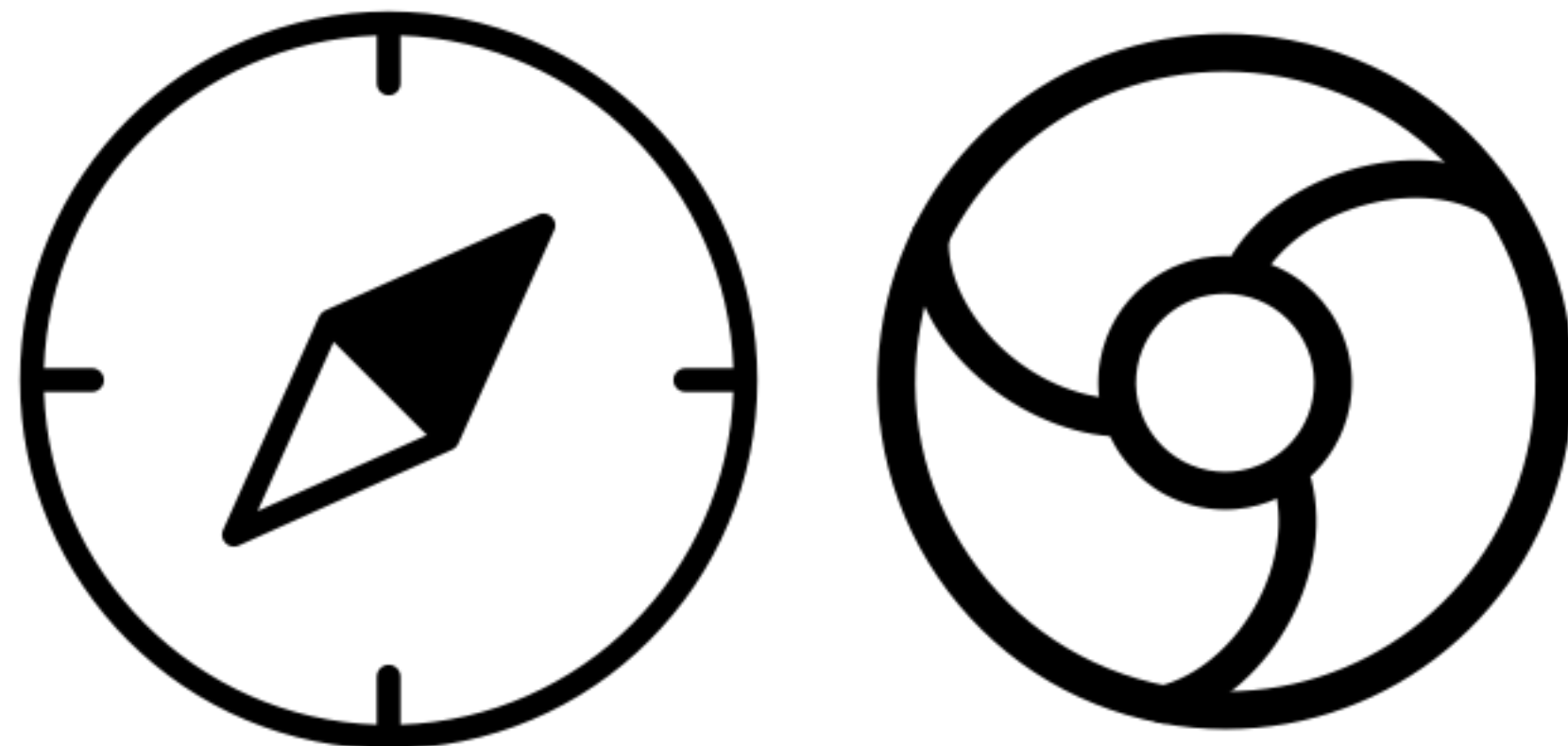
db



http



local
(net)



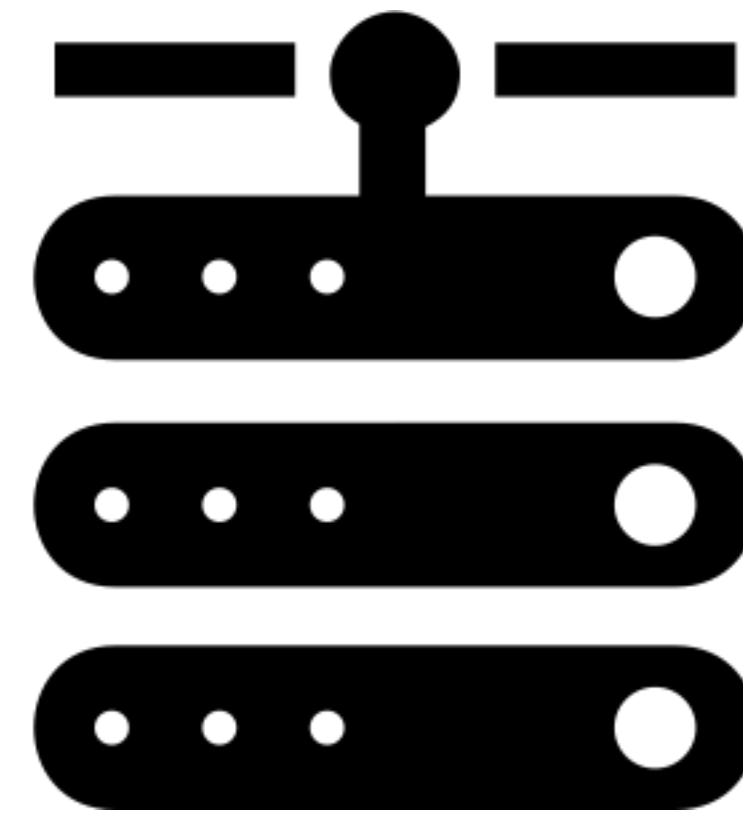
front-end

back-end

browsers



servers



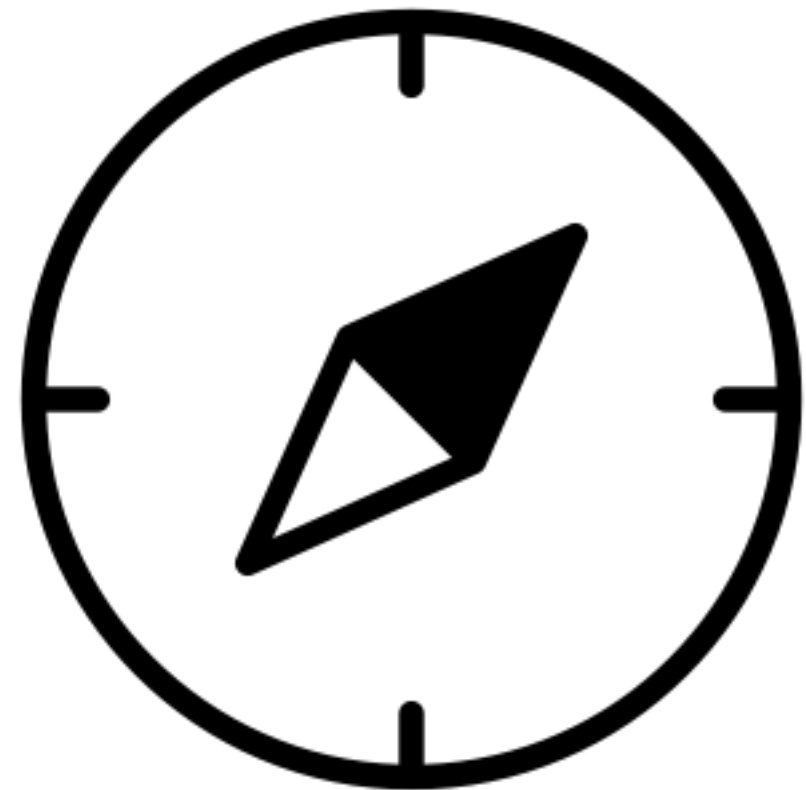
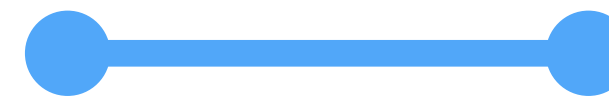
db



http



local
(net)



front-end

back-end

why persist

you can't keep everything in memory

crashes, restarts, etc...

persistence via databases can give you performance benefits and greatly simplify your code

persistence via databases can give you performance nightmares and greatly complicate your code

in-memory approaches

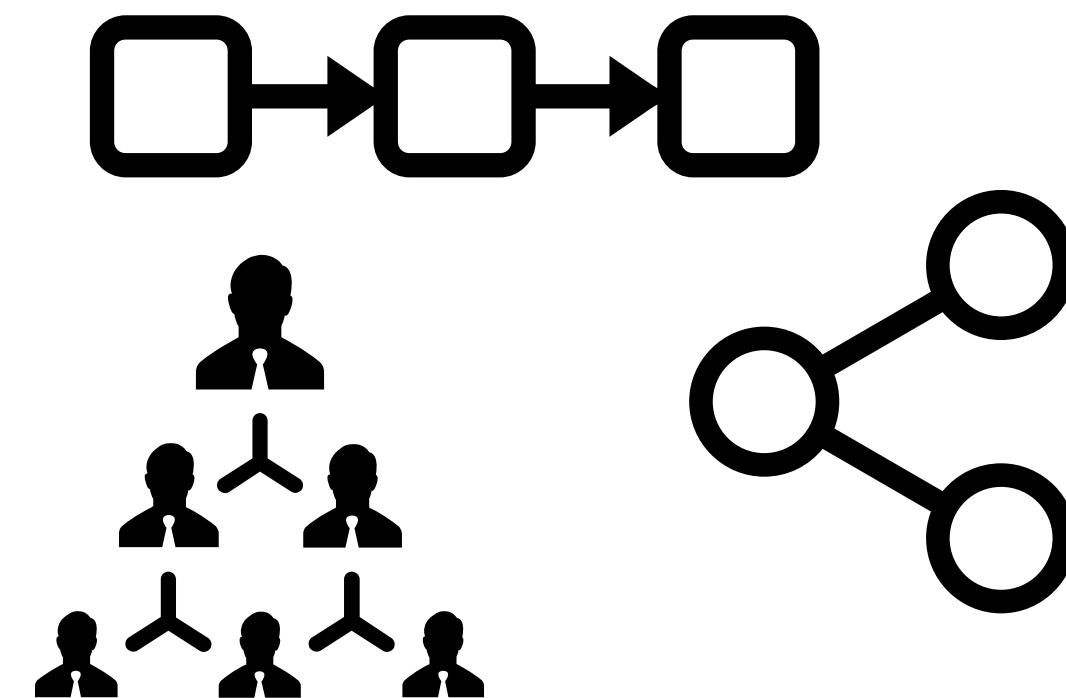
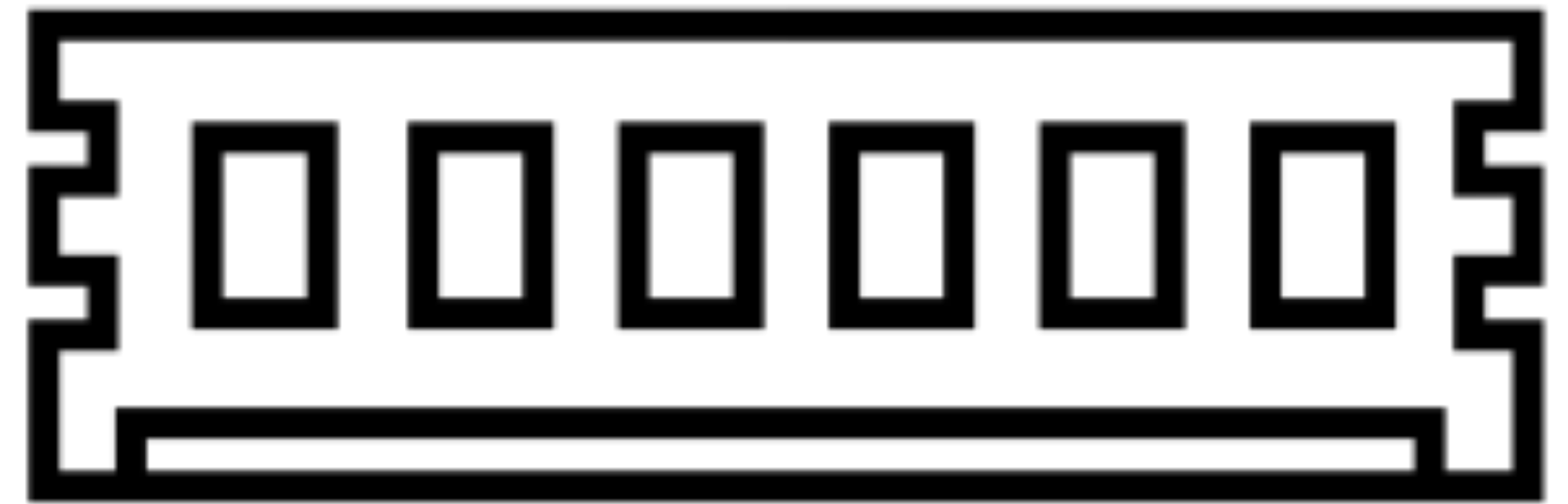
pro: fast

con: limited by memory

con: on crash, you're toast

con: must write a lot of access/updates functions yourself

who remembers data structures?



file-based approaches



pro: reading/writing files is common

con: slow

con: append is easy; update/delete can be annoying

common file types:

- txt
- CSV
- json (javascript object notation, example)

file: streams vs sync

```
1 // sync
2 var d = fs.readFileSync('all_sponsors.json')
3 console.log( d )
4 console.log( '---' )
5 console.log( JSON.parse(d) )
6
```

```
7 // async
8 fs.readFile('all_sponsors.json', function(err, d) {
9   if (err) throw err
10  console.log( d )
11  console.log( '---' )
12  console.log( JSON.parse(d) )
13 });
```

```
1 // stream
2 var rs = fs.createReadStream('all_sponsors.json');
3 var buf = ''
4 rs.on('data', function(d) {
5   buf += d
6 });
7 rs.on('end', function(d) {
8   console.log(buf)
9 });
```

```
<Buffer 5b 0a 20 20 20 20 7b 0a 20 20 20
2 2f 61 74 74 61 63 68 6d 65 6e 74 73 2f
4 ... >
---
[ { src: '/attachments/supporterssites/v
ith_text_white.png',
  href: 'http://www.kaust.edu.sa/',
  class: 'Platinum',
  year: 2011 },
  { src: '/attachments/supporterssites/v
o_bL.png',
  href: 'http://www.microsoft.com/',
```

databases

pro: fast; built in add/update/delete(!)

con: database choice can be hard

database types/considerations:

- relational
- nosql
- facets: size, speed, scale



sqlite

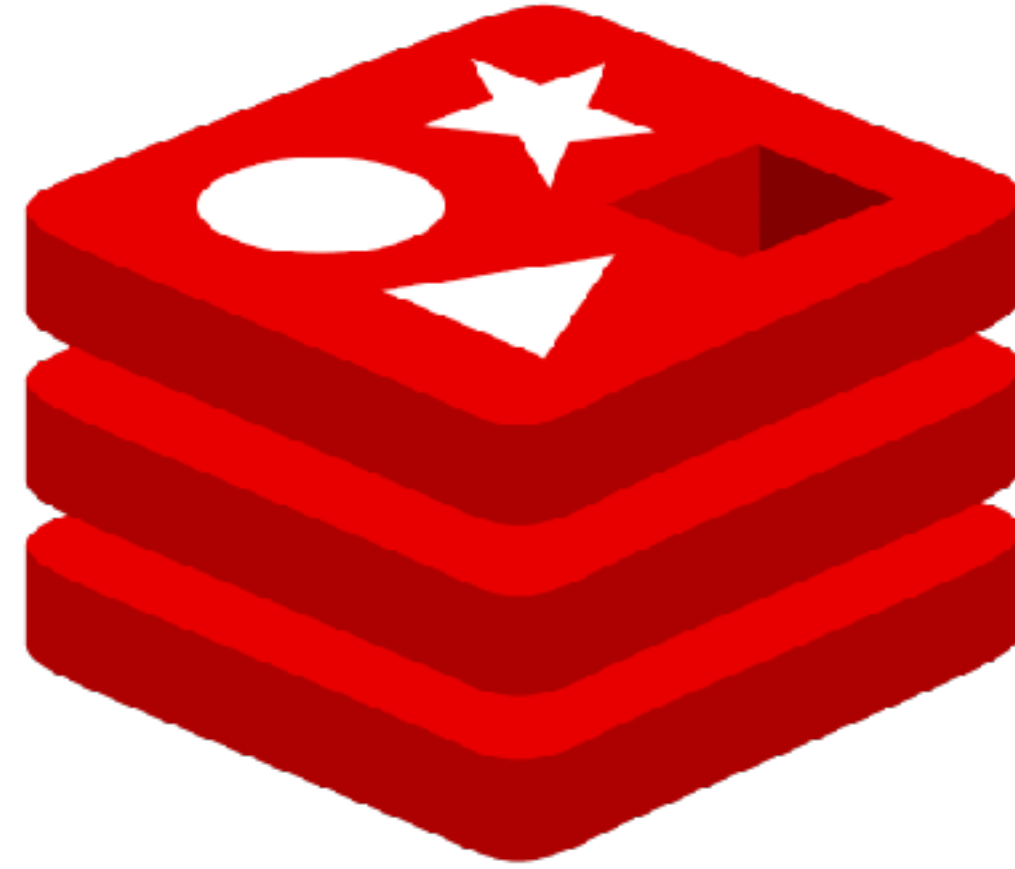


lightweight

widespread: mobile, web, desktop

sql is very common

redis



redis

featherweight

in-memory cache: f.a.s.t.

think of it as a hashmap
(key+value store)

mongo



mongoDB®

heavyweight

takes anything you throw at it

supports queries

the best, and worst,
of both worlds

```
var insertDocument = function(db, callback) {  
  db.collection('restaurants').insertOne( {  
    "address" : {  
      "street" : "2 Avenue",  
      "zipcode" : "10075",  
      "building" : "1480",  
      "coord" : [ -73.9557413, 40.7720266 ]  
    },  
    "borough" : "Manhattan",
```

firebase



Firestore

google

lets you make serverless apps

must buy-in to google

```
function writeUserData(userId, name, email, imageUrl) {  
  firebase.database().ref('users/' + userId).set({  
    username: name,  
    email: email,  
    profile_picture : imageUrl  
  });  
}
```