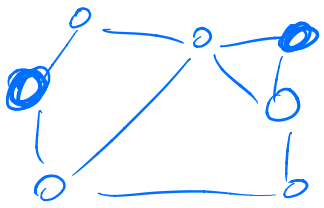
Emery Berger

web-programming.org

Internet — 1969



sftp

email

file transfer protocol

f1 ! f2 ! f3 ! f4 ! — @ —

single point of failure

SMTP

foo.cs.utexas.edu/ /

gopher

Tim Berners-Lee CERN

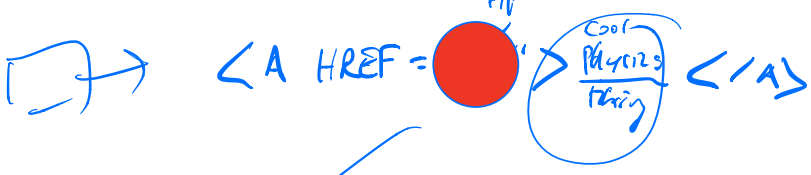
HTML

hypertext markup language

hypertext trans. protocol

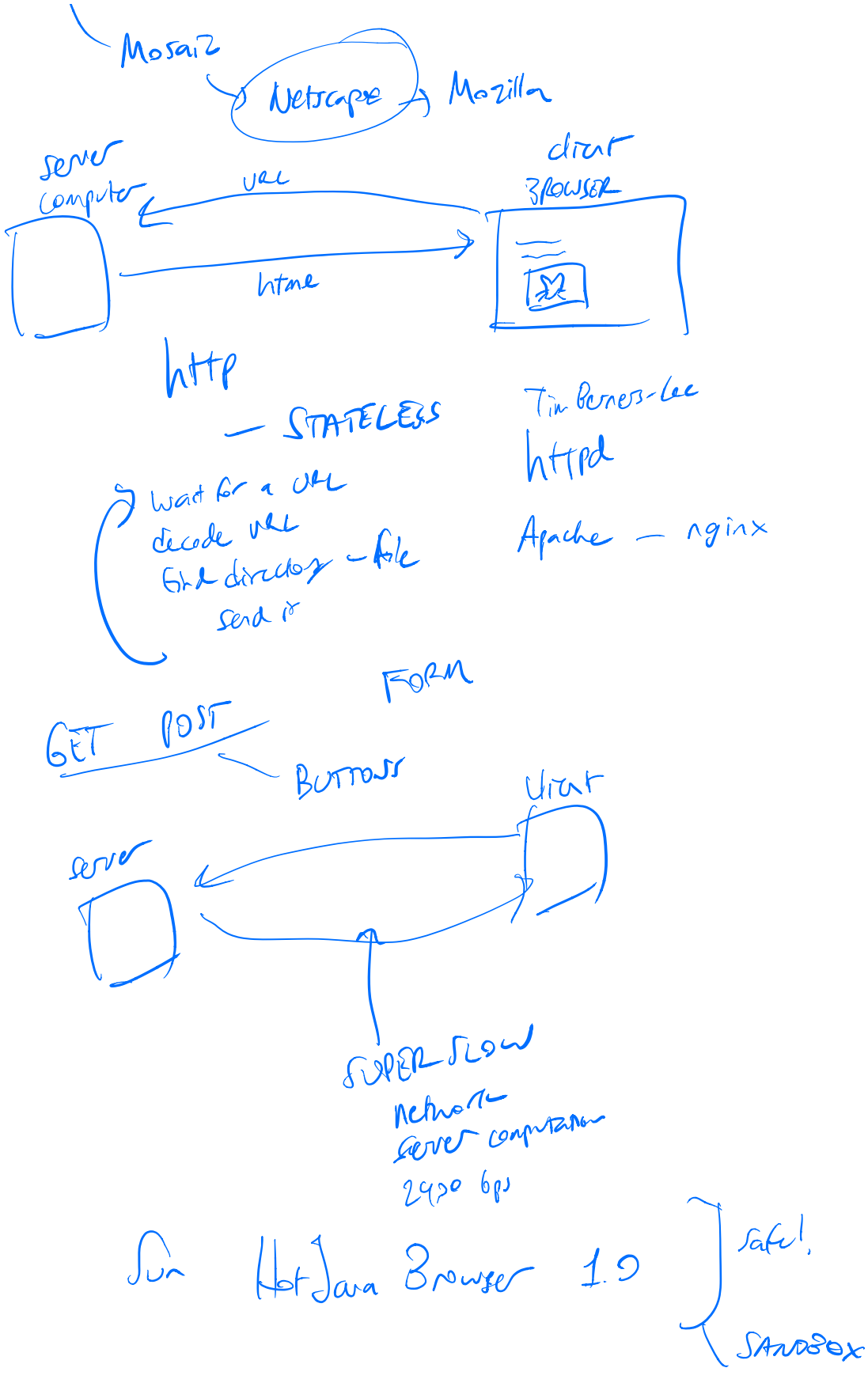
http

ftp



one degree of indirection

NCSA <IMG SRC=" ">


Mosaic
Netscape
Mozilla
server computer
client BROWSER
URL
html
HTTP
– STATELESS
Tim Berners-Lee
httpd
Apache – nginx
wait for a URL
decode URL
find directory – file
send it
FORM
GET POST
BUTTONS
client
server
SUPER SLOW
network
server computation
2400 bps
Sun HotJava Browser 1.0
safe!
SANDBOX

Microsoft applet

↓

ActiveX

Brendan Eich ~ JavaScript

/

dynamically typed

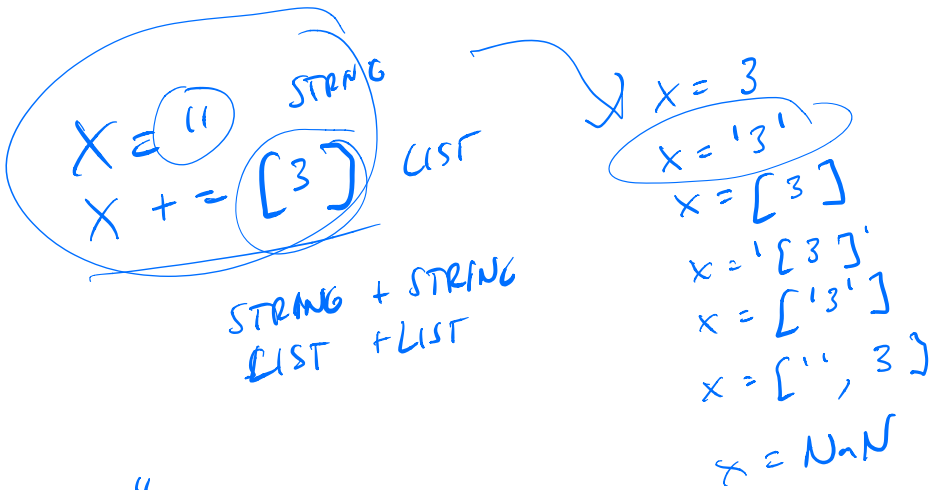
Object.toString ∞

x = " "

x = 1.0

x += [3]

x += ~



"WAT"

TypeScript

x : string

x += [3]

x : any

console.log( )

WASM

Web Assembly

C C++ Rust

JS

"assembly"

x = x + 1

INC AX

x = x + 1

WTF ?

x++

David Unsar

↳ Self GC

1

if

JIT

Generational GC

just in time compiler

Craig Chambers

Urs Holzle

Google

→ lead architect of V8 JIT

HTML
CSS

browser
DOM
document object model

dynamic web pages
AJAX
asynchronous JavaScript & XML

IE
Internet Explorer

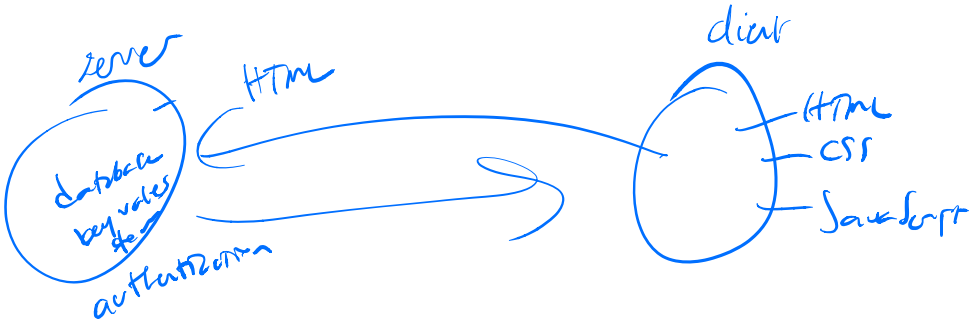
Steve Ballmer
- monkey dance
- developers!..

web version of Outlook

CLIENT SIDE
SERVER SIDE

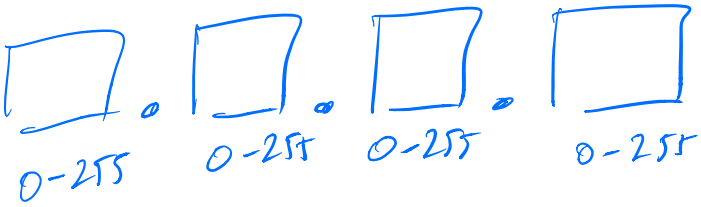
database
key-value stores

x = {}
x[' ']
x.set(k, value)
x.set(k)

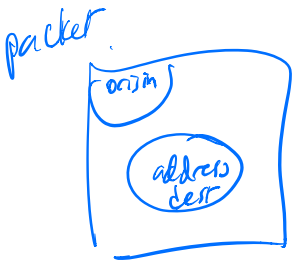


Security through obscurity

IP addresses

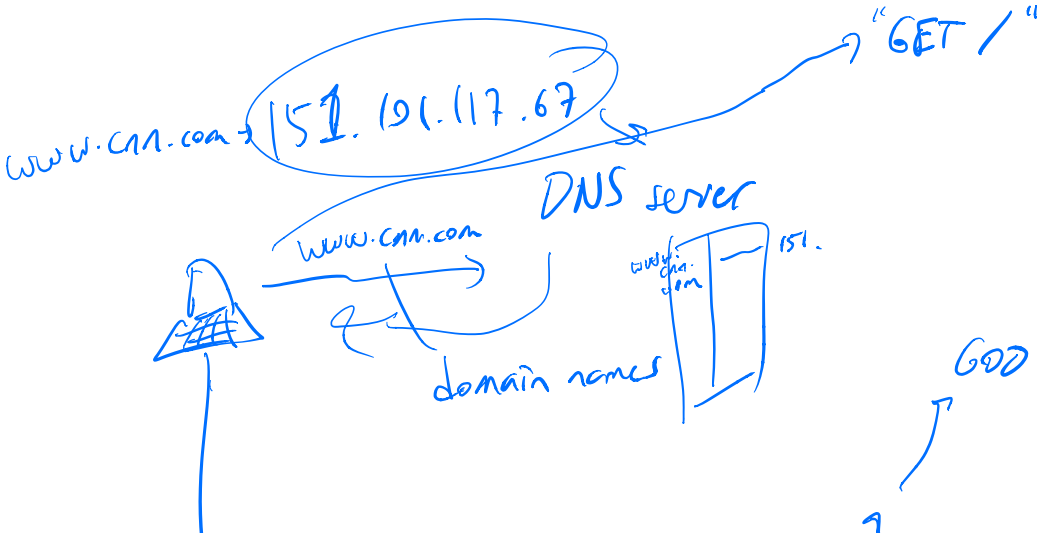


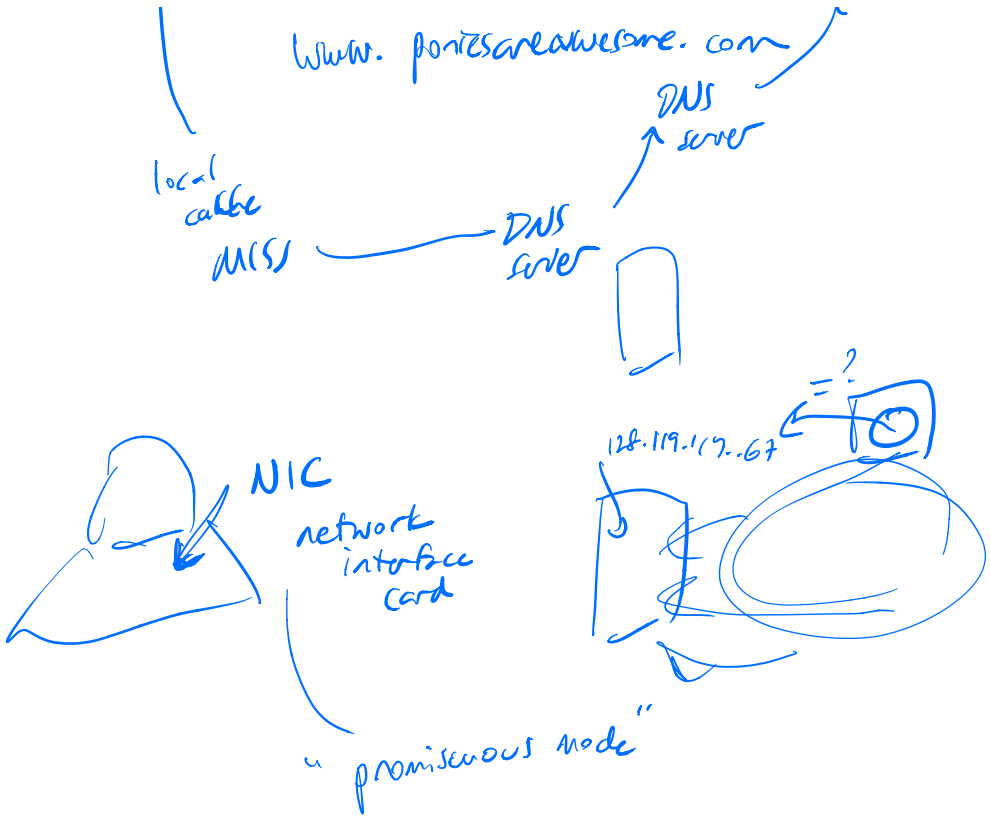
IPv4



"localhost"
→ 127.0.0.1

128.119.101.3

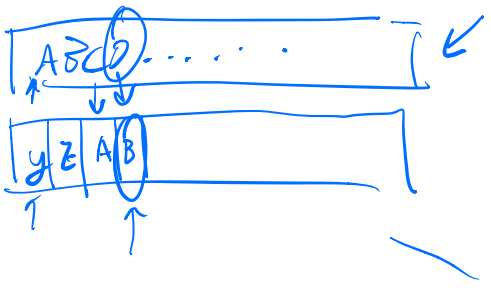




https secure RSA

Symmetric encryption

Caesar cipher

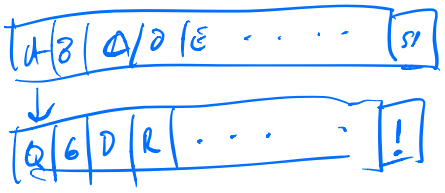


DAD → BYB

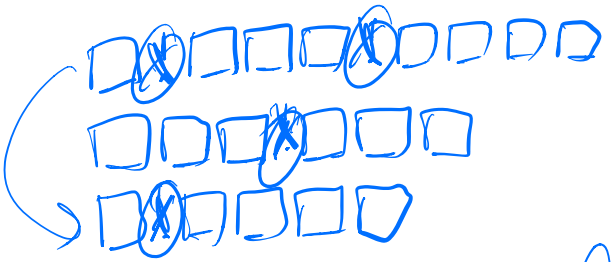
Symmetric both encryption & decryption

ROT13

transcription cipher



$\frac{26!}{2} >> \frac{26}{2}$



AN
IN
ON
TO

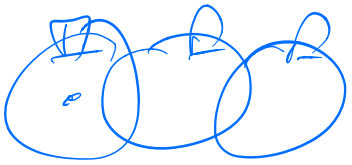
Frequency analysis

E 25%
T 17%
A
O
I
N
...

CXME_TZ_TQE_MAZZ

Edgar Allan Poe

The Golden



Enigma

Alan Turing

"computer"

automatic computers

key - private secret!

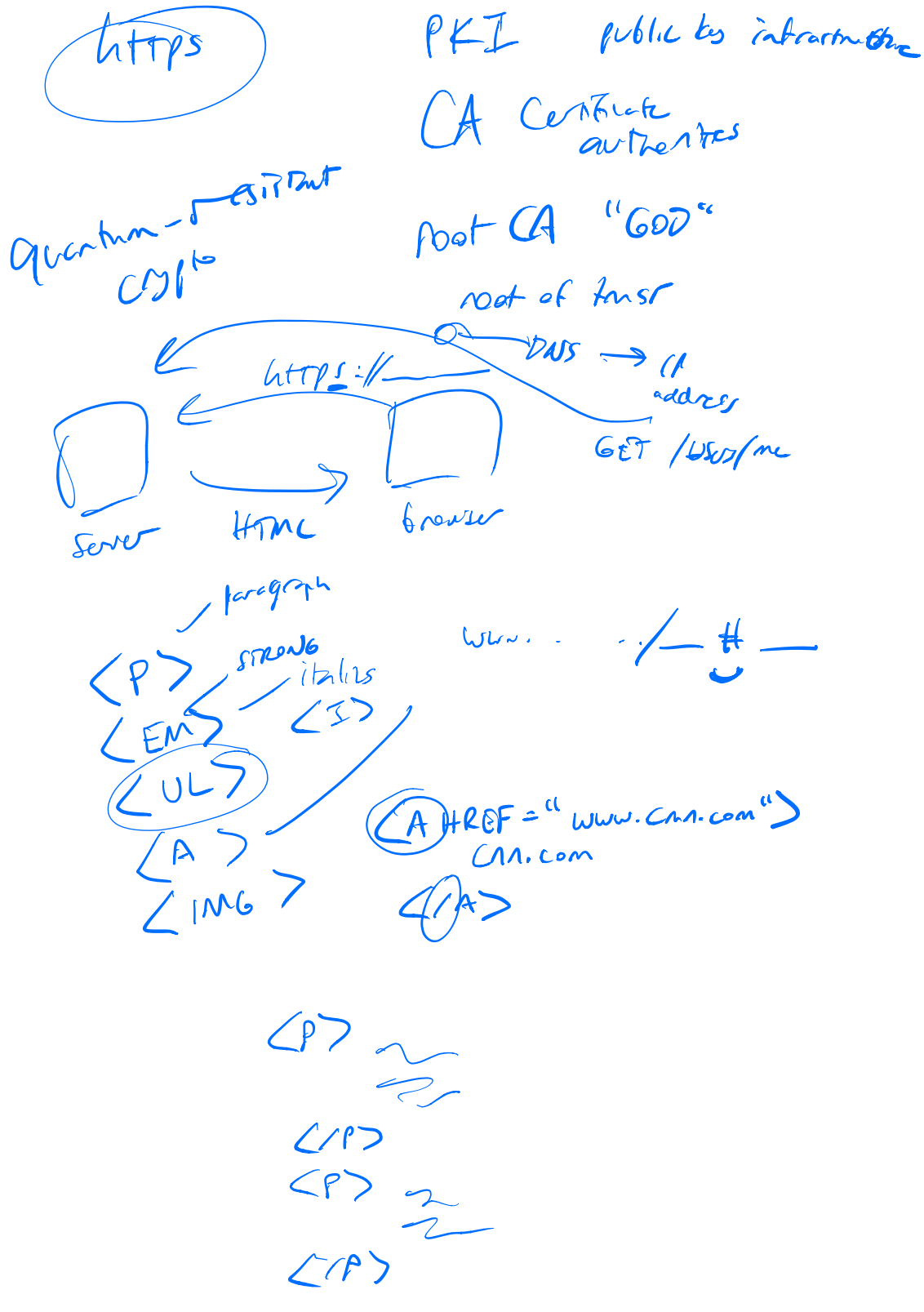
RSA

Rivest
Shamir
Adelman

public key encryption

hardness of
factoring prime numbers

$q = p_1 * p_2$


HTTPS
PKI public key infrastructure
CA Certificate authorities
quantum-resistant crypto
Root CA "GOD"
root of trust
DNS → IP address
HTTPS://
GET /users/me
Server
HTML
Browser
paragraph
<P>
STRONG
italics
<I>
<EM>
<UL>
<A>
<IMG>
www. . . /— # —
<A HREF = "www.cnn.com">
cnn.com
</A>
<P>
</P>
<P>
</P>

<UL>
<LI> foo </LI>
<LI> bar </LI>
</UL>

- foo
- bar

<OL>

1. foo
2. bar

table header

<TABLE> <TH> <TD> Car </TD>
<TD> Price </TD>
</TABLE> </TH>

table row

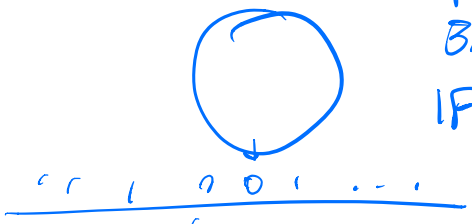
<TR>
</TR>

not Turing-complete

<HR>

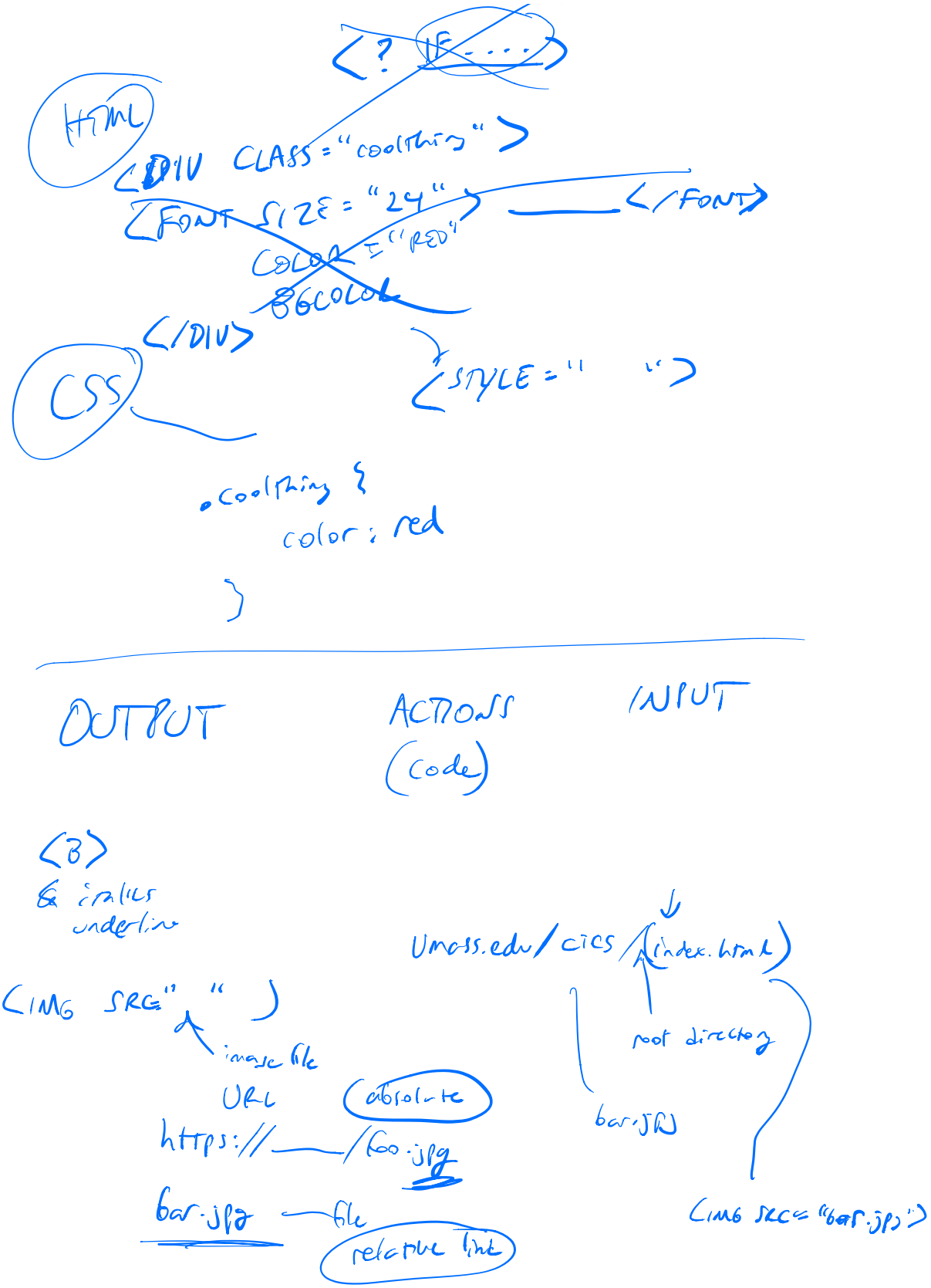
Turing machine

X = VALUE
VALUE = _ 0 / 1
FORWARD
BACKWARD
IF VALUE == ? (0 / 1)
GO _ , OR FORWARDS



PHP

<HTML> <HEAD> / metadata
<TITLE> - </TITLE>
< >
</HEAD>
<BODY>
</HTML>
</BODY>


<? IE ... >
HTML
<DIV CLASS="coolthing">
<FONT SIZE="24"> </FONT>
COLOR="RED"
BGCOLOR
</DIV>
CSS
STYLE=" "
.coolthing {
color: red
}
OUTPUT
ACTIONS (code)
INPUT
<B>
& italics underline
<IMG SRC=" ">
image file
URL
absolute
https://___/foo.jpg
bar.jpg
file
relative link
umass.edu/cics/(index.html)
root directory
bar.JPG
<IMG SRC="bar.jpg">

GIF

Graphics Interchange Format

JPEG

PNG

portable graphics

R,G,B 0-255, 0-255, 0-255

1024 × 1024

Compression

TIFF

(255, 255, 255)

99%

RLE

run-length encoding

255, 255, 255

(1024, —WHITE—) COUNT — same

OUTPUT (#, val)

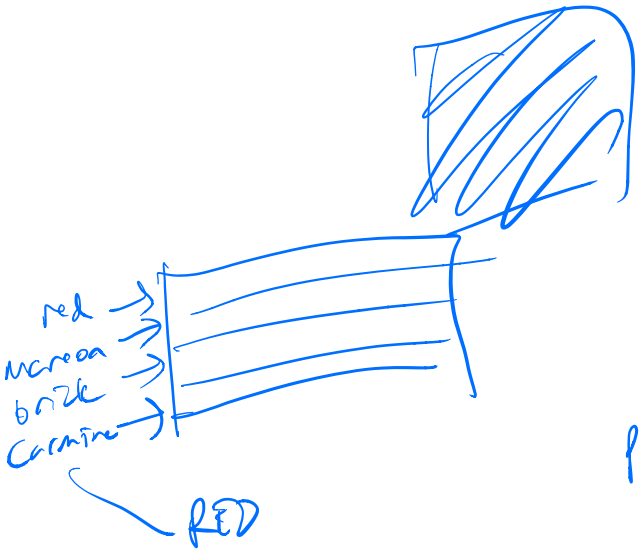
(10, white) (20, black) 900 white

JPEG

LOSSY — graphic — original ≠ uncompressed) loses information

LOSSLESS — =

GIF PNG



JPEG

perceptual-based compression

2MB → 100K

original

MPEG

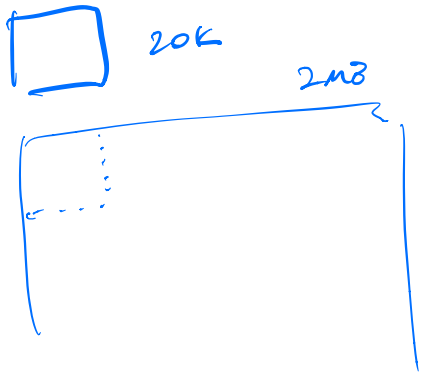
load speed

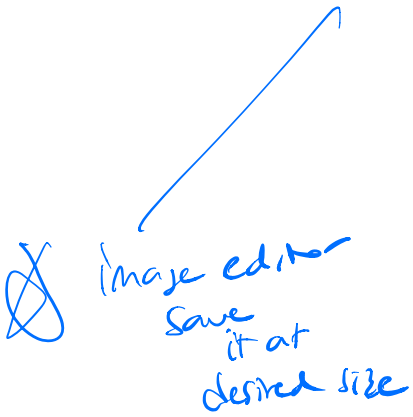
GIF PNG

LZW77

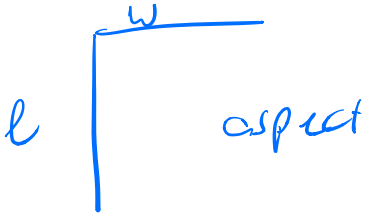
Lempel-Ziv

<IMG SRC = " "
HEIGHT = 2in
WIDTH = 2in >

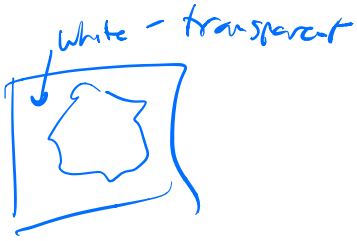




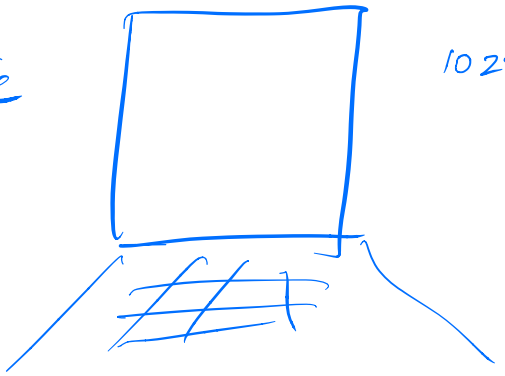
Upsampling →
downsampling



transparent



screens used to be



"responsive" — dynamically adjusts desktop & mobile

Bootstrap style and layout

"favicon" ↖

<link rel="icon" href="—">

audio <audio> } HTML5
video <video>

Adobe Flash — ActionScript — vector for malware

Steve Jobs
— iPhone
X Flash — Adobe
X Java — Sun/Oracle

death of Java on "client"

Apple II, Apple II plus
visicalc

Apple III
IIc

Java —
no buffer overflow
no use after free
— GC

Xerox PARC
Palo Alto Research Center
laser printer

AT&T
— Bell Labs
Unix C
modem

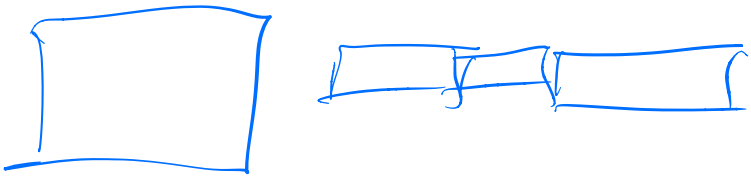
IBM — TJ Watson Research Labs

Microsoft — MS-DOS, PC-DOS — Microsoft Research

~~gets~~ (↓ P)

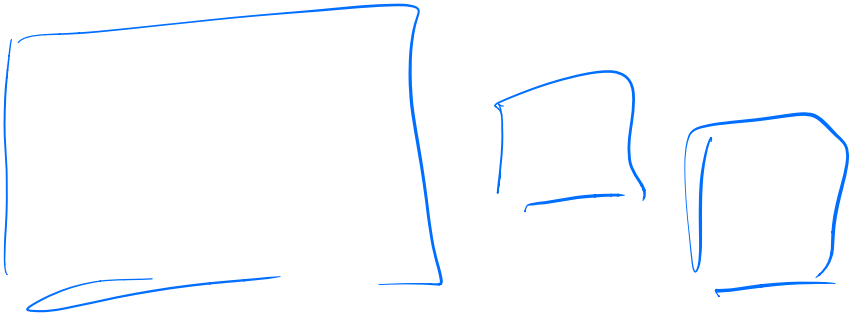
_STYLE="" " CLASS="" "

<DIV>..</DIV> — P — CSS
<SPAN>..</SPAN>



<IMG ALT = " " accessibility
<VIDEO
<AUDIO

put man alt tag

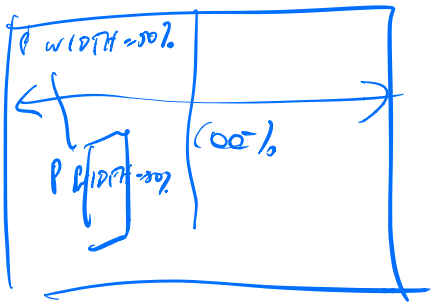


HTML layout

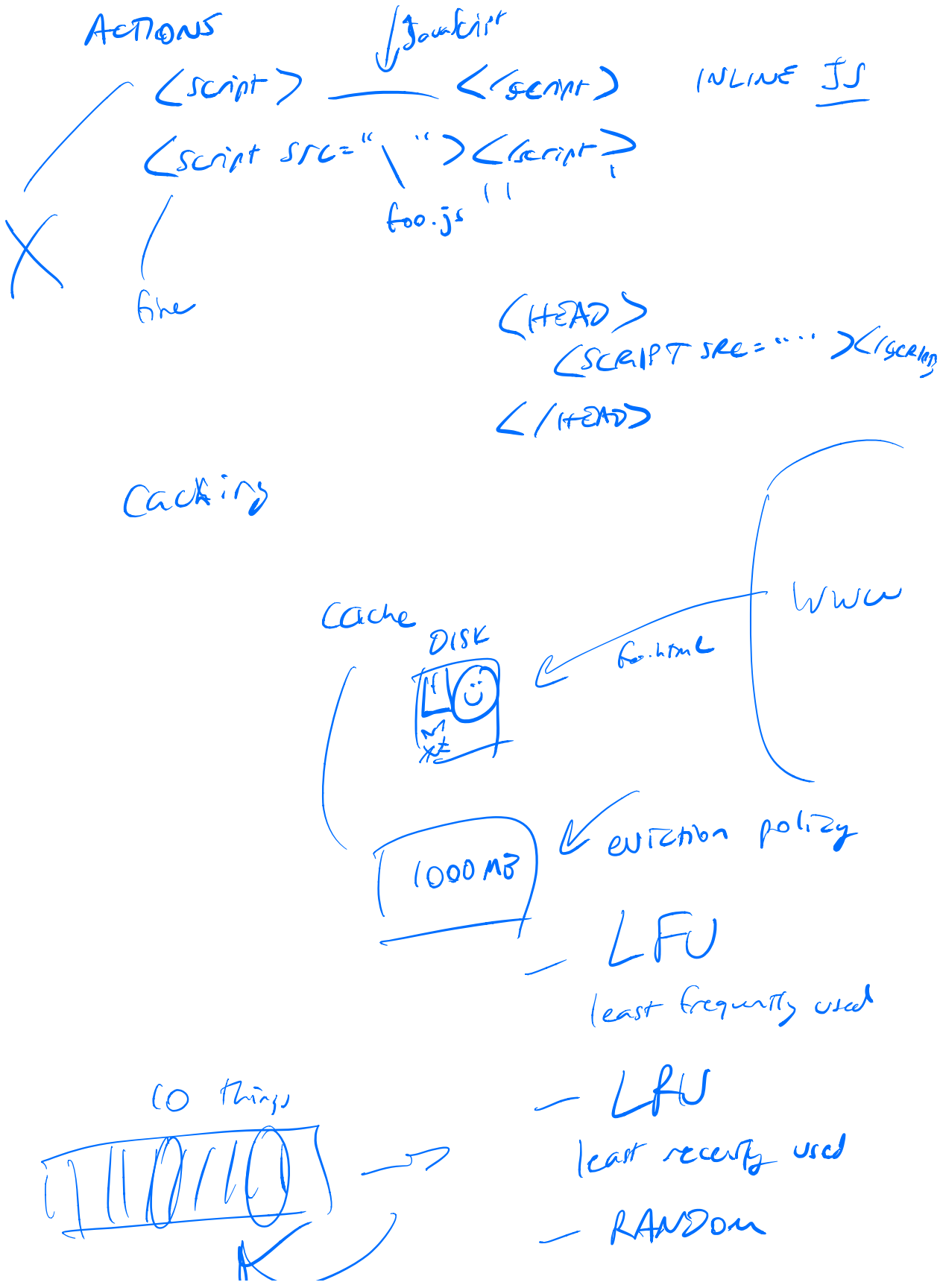
"block" table
<P>
:

<P WIDTH = "50%">

cm
in
px

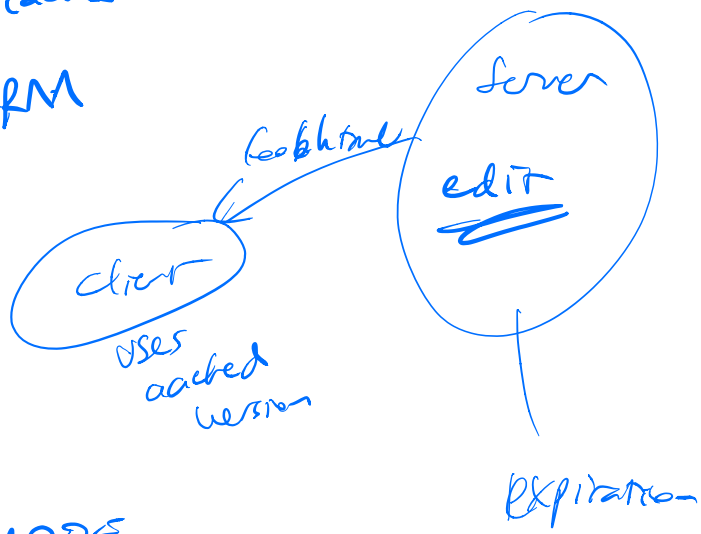


OUTPUT


ACTIONS
JavaScript
<script> </script>
INLINE JS
<script src="foo.js"></script>
foo.js
<HEAD>
<SCRIPT SRC="..."></SCRIPT>
</HEAD>
Caching
Cache
DISK
WWW
foo.html
1000 MB
eviction policy
LFU
least frequently used
10 things
LRU
least recently used
RANDOM

chips have caches

ARM



INCOGNITO MODE

SHIFT refresh

January 30

INPUT

{ OUTPUT } CSS

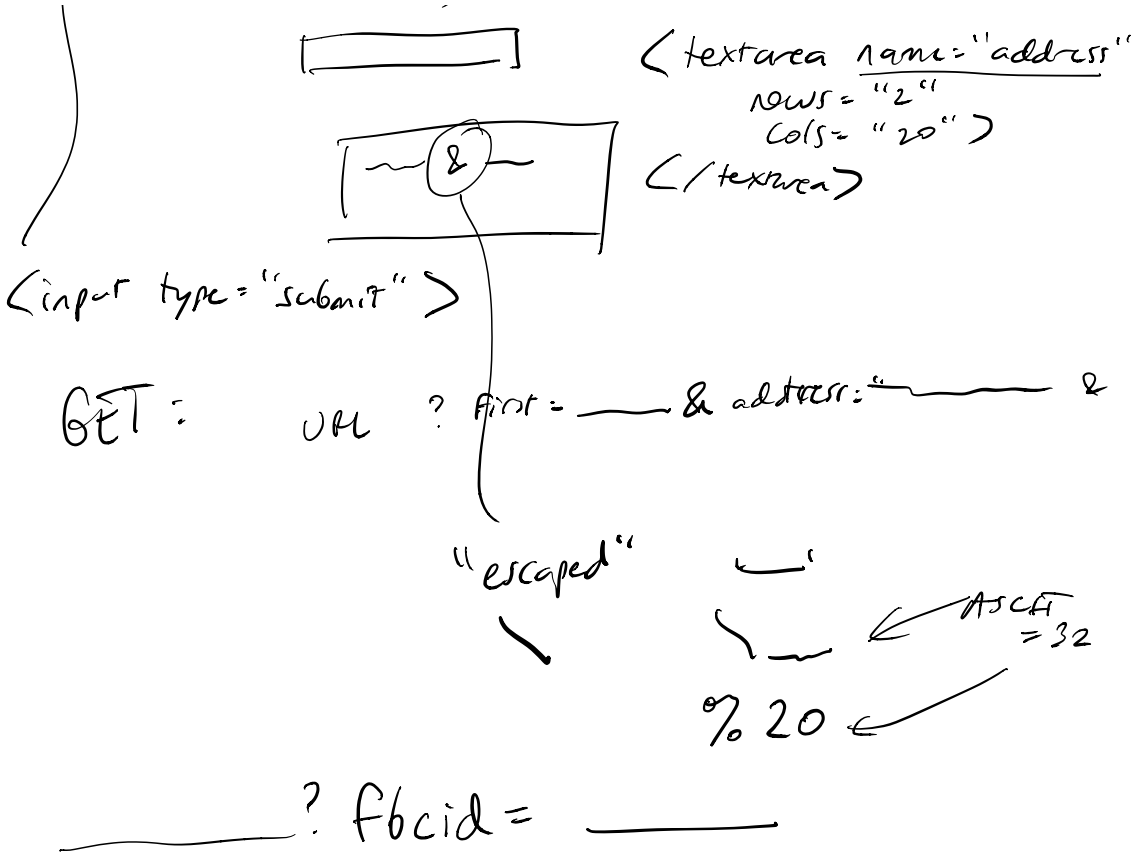
< form action = " ____ " method = "post" "GET"

URL (server)

</form>

<input name="first" type="text">

</input>



total = 10.20 & cups = 200

"URL hacking"
tweaking

& login=

POST "safer"

CANNOT
TRUST
BROWSER
→ validation on server

BUTTONS

<input type = "checkbox"

"radio"

name = "choice1" value = "-" >

checked

MENUS / DROP-DOWNS

<select name = "school">

<option value = "umass"> UMass Amherst </option>

" " "mammoth"> Amherst College </option>

hamster

</select>

<input type = "button" src = "image.png" >

ALT = " "

<input type = "password" >

not allowing paste XKCD

<< 26^{25}

d3 4d l33t-speak

0 O

{ OUTPUT } CSS

Cascading Style Sheets

HTML
structure
"skeleton"

CSS
Style
"skin"

WHAT matches { HOW do I want it styled? }

selector

- inline

GREEN MONSTAH

 —— } unaffected

- Styling

<HEAD>
<STYLE> —— p {
color:green;
font-size:48pt;
}
</STYLE>
</HEAD>

— "style sheet"

- CSS

<link href="biggreen.css" rel="stylesheet" />

color — blue, red

rgb(, ,) — red, green, blue — additive, subtractive

#F0F0EE

CMYK — black, cyan, red, yellow

background-color

background-image

rgba(, , , 0-1) — 0-255 — alpha, opacity

font-family: serif, sans-serif

serif — MS Comic Sans

The

text, "body"

sans-serif — titles

font-family: "Times New Roman"

"web safe fonts"

- ttf
  TrueType font

asynchronous

synchronous

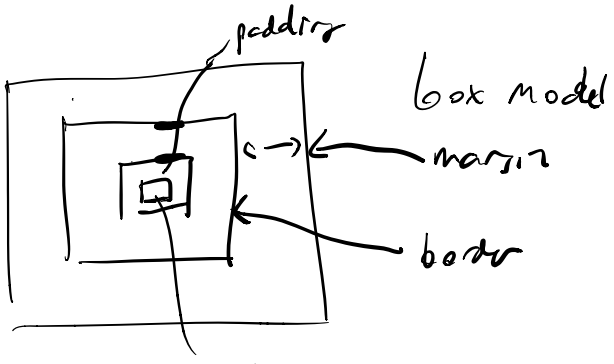
jank
↘ smooth

font-weight : bold
font-style : italic

font-variant : small-caps

Hello
HELLO

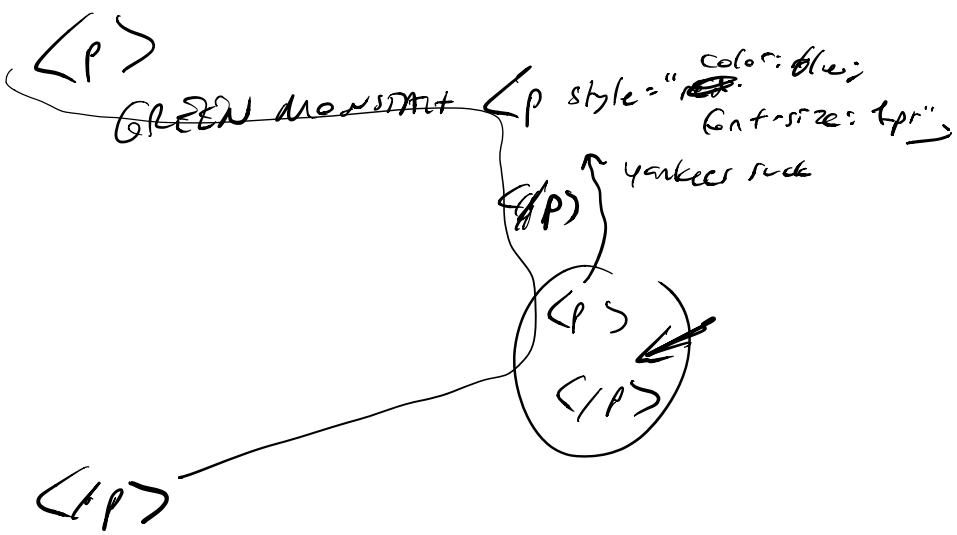
width
height



float
- variables
- animation

width
height
%
5px ... 2in

Cascading



.bigscreen {

}

#sox {

}

CLASS — multiple

ID — unique

---

p span { + > ~

}

---

pseudo-class — events

p:hover {

color: red

}

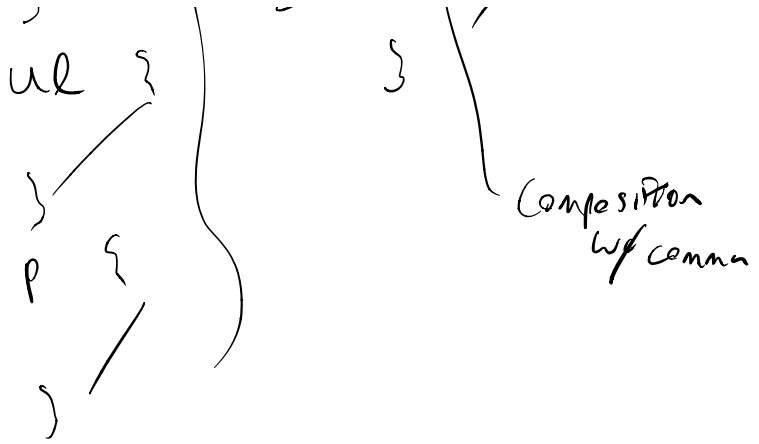
hovertip

---

ol {

}

ol, ul, p {



Feb 4

SPECIFICITY

CSS

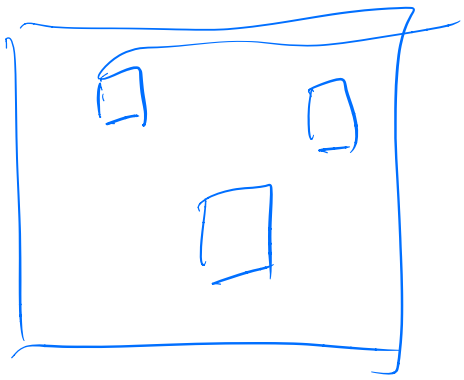
```
p {
  color: red;
}

.classy {
  color: blue;
}

.classy {
  color: green;
}
```

HTML

```
<p class="classy">
  What color am I?
</p>
```

classes "more specific" than **elements**

SOURCE ORDER:
last one wins!



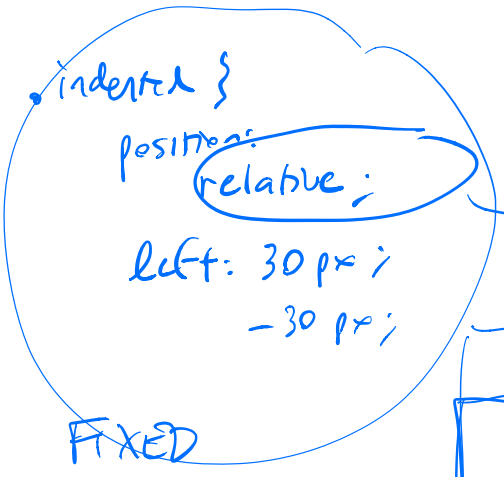
POSITIONING

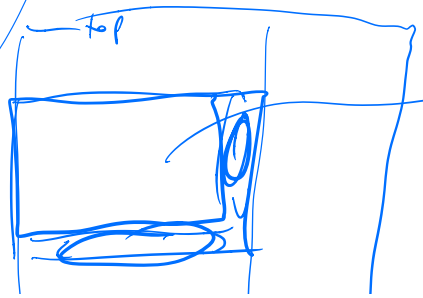
individual

ONE-DIMENSIONAL

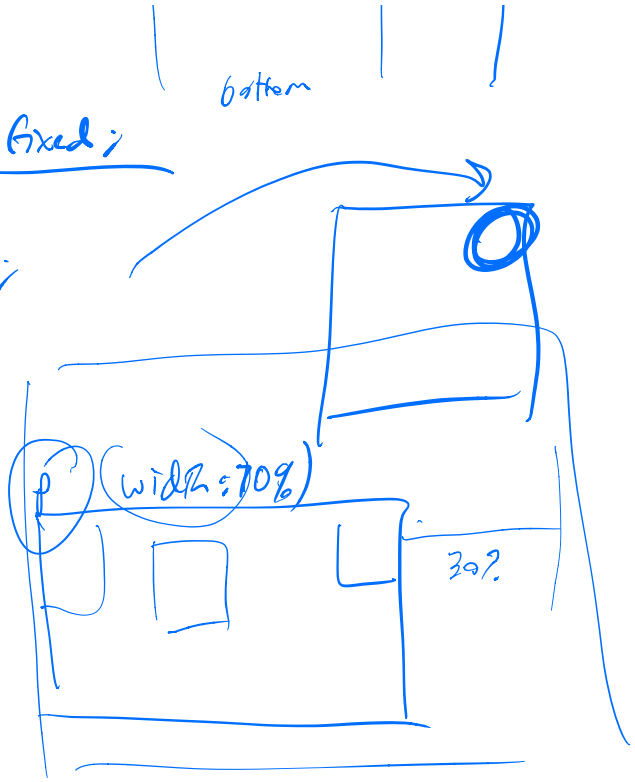
TWO-DIMENSIONAL

animation

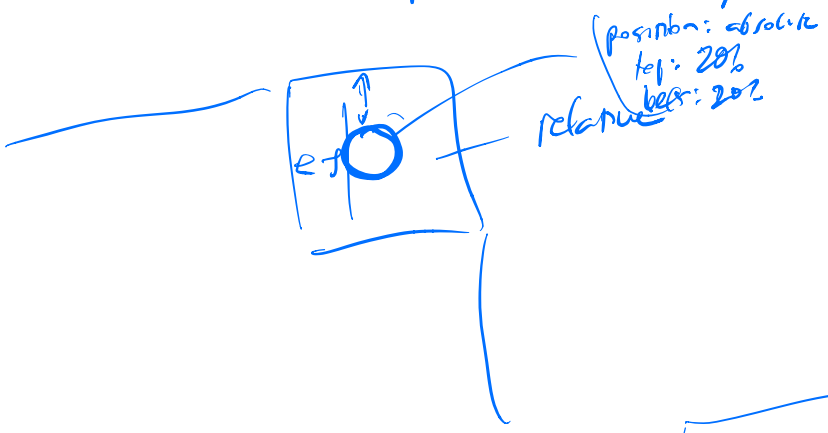




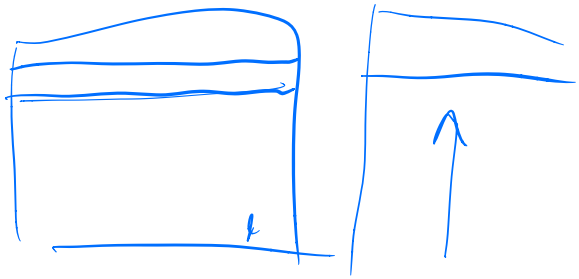
```
.banner {
  position: fixed;
  top: 0;
  right: 0;
}
```



position: absolute;



position: sticky



headlines
navigation links

Chrome: 36
Firefox: 11
Braw: 2
Safari: 1
IE/Edge: 4

Webkit

Opera: 1
Tor: 2

Brendan Eich

VIEW →
DEVELOPER →
- view source
- developer tools
- inspect element
- javascript console

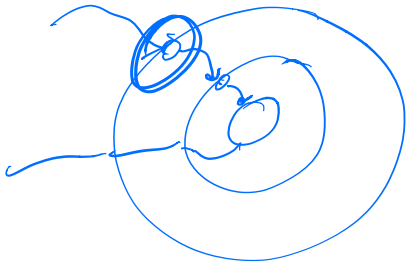
& fbclid
FB

Same origin policy

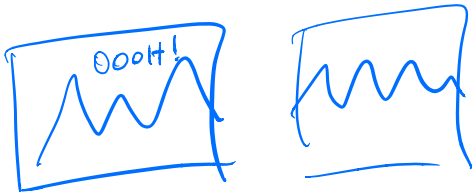
Google

console.log(" ")

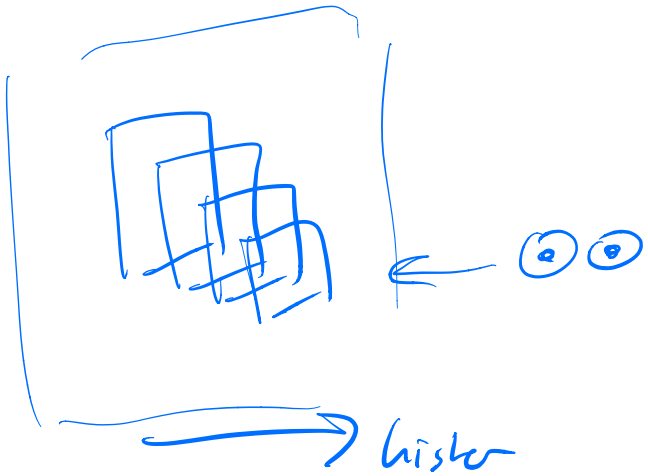
Tor the onion router



OVERLAP



Z-index

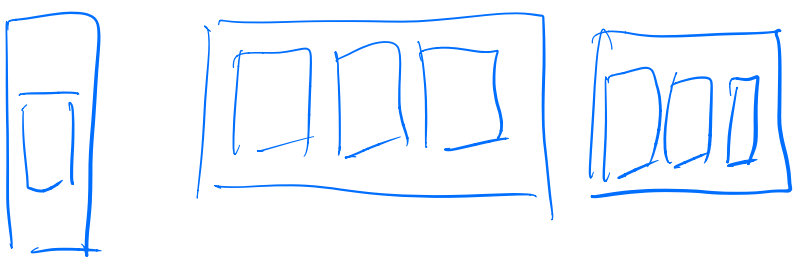


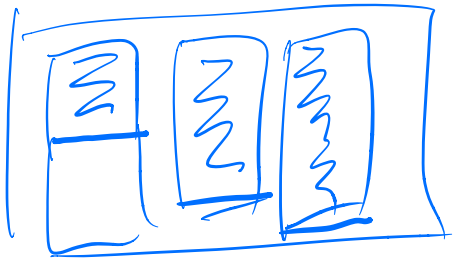
Z-index: __

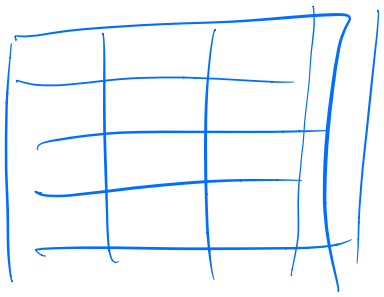
2 > 1 2 on top of 2

```
img {
  position: absolute;
  left: 0px;
  top: 0px;
  z-index: -1;
}
```

Challenges to layout



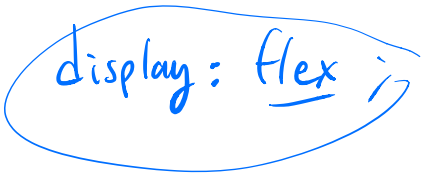




HTML
tables

1-dimensional

flex boxes



flex-direction: column
row
column-reverse

row-reverse

flex-wrap

2-dimensional **grid**

CSS `display: grid`

```
.sneddy {
    display: snd;
    grid-template-columns:
        auto auto auto auto
}
```

200px 2fr 1fr 1fr 1fr

HTML

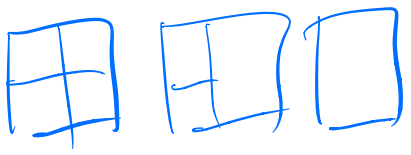
```
<div class="sneddy">
    <div> 1 </div>
    <div> 2 </div>
    ⋮
</div>
```

---

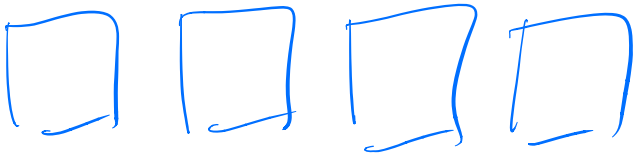
24 frames per second

persistence of vision

129,600 frames



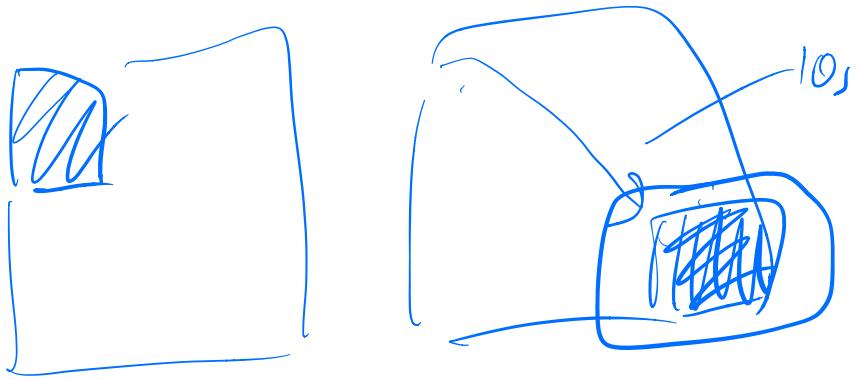
Storyboards



key frames 2

1

in-betweens



animation: NAME DURATION REPEAT

box 10s infinite

```
@keyframes box {
    from { color: red;
           left: 0px;
           top: 0px; }
    to   { color: blue;
           right: 0px;
           bottom: 0px; }
}
```

JavaScript — Bad Parts

JSON — Douglas Crockford

JavaScript Object Notation

```
{ "foo" : 3,
  "bars" : [1, 2, ]
  "forest": { }
}
```

Brendan Eich

Netscape Navigator

Mosaic

Marc Andreessen

Jim Clark — Sun

Stanford Univ Network

```
(defun (f x y)
  (if ( )
      ( )
      )
  )
```

<HTML> <HTML>

{ Scheme Self }

JavaScript ≈ 1000x faster than Python

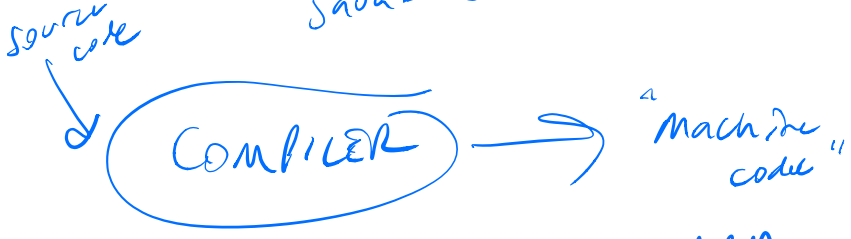
JIT compiled just in time

AOT compiled

clang foo.c -o foo

javac

java — JIT



ARM
x86

OPTIMIZATION

- redundant computation

```
x = 1;
x = 1;
x = 1;
```

```
for(i) {
  expensive
}
```

hoisting

dead code

node.js

nodejs.org

% node
>

JVM bytecodes

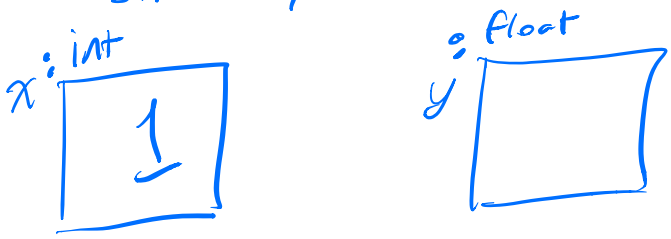
javac f.java → f.class

ADD
SUB
⋮

Safe

every variable ~~& object~~ is typed

STATICALLY TYPED



int x;
x = 1;
~~x = 1.0~~

x = "hello"

int, bools, floats, strings, arrays, objects

ArrayList

immutable

class
inheritance
interface

array [1, 2, "hello", True]

Integer(2)
Integer(3)
StringBuffer() BooleanU

Object

int array

Integer

ArrayList(Object)

Integer
int

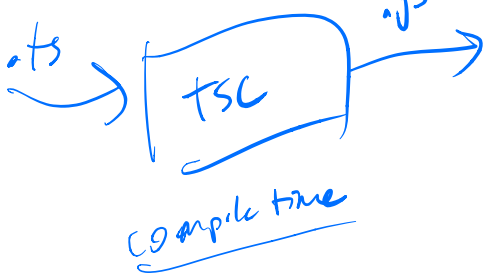
foo(v) {
v = 3;

```
}
let x = 12;
foo(x);
console.log(x);
```

JavaScript "DYNAMICALLY TYPED"

1 ∈ number
2.0 ∈ number
True ∈ boolean

TypeScript



gradual typing

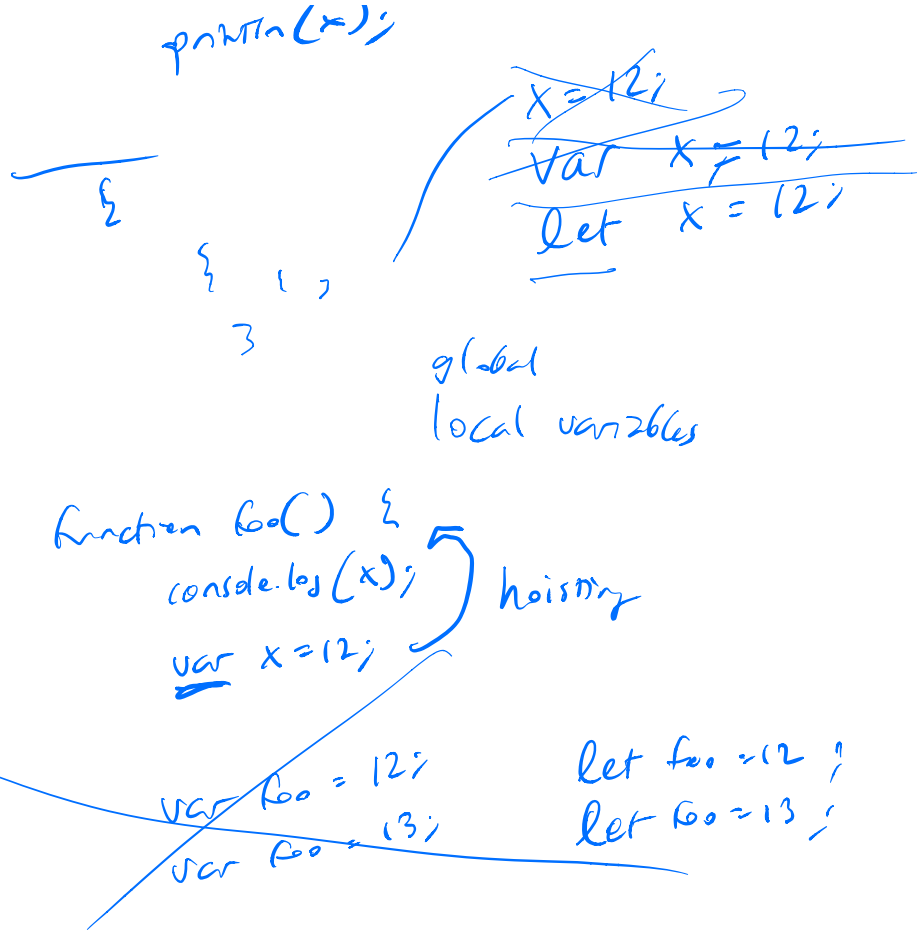
```
let x : number = 12;
        : any
```

Java

```
int x = 0;
{
  int x;
  x = 12;
}
```

LEXICAL SCOPE

nested



Java

== identity (addresses)

.equals() / semantically

```
x = Integer(13);
y = Integer(13);
```

`x == y` false

`x.equals(y)` true

dynamic type coercion

```
> "2" + 2
"22"
> "2" - 3
-1
> -1 + 0
-1
> "-1" + 0
"-10"
> 0 + "1"
"01"
```

`"2" != 2` → do coercion! → `"2"`

true

`===` `!==` — diff-type ⇒ not equal

```
> [] + 1
"1"
```

`[1]`

"1"

> [] + [1]

[[] [1]]

[1]

"" "1"

"1"

> [] + [1,2]

"1,2"

"1,2"

> [] - 1

-1

(1,2,3)

x = { 'a': 5,
'b': 9
}

x['a']

x.a

> "hello" - "hell"

NaN

> [] + {}

"[object Object]"

= "" ∅

> {} + []

0

> {} + {}

3

> x = { 'a' : 12 }

> x + 0

"[object Object]0"

str = JSON.stringify(obj)

obj = JSON.parse("{ }")

x.a 12

x["a"] 12

if (18) {

A;

} else {

B;

}

0

nonzero number truthy

0 falsy

negative truthy

null

~~None~~

NaN

undefined

"" falsy

"0" truthy

[] truthy

Tcl

Tk — 3 + 1 — Canvas

x

[incr x]

~~FAE~~

~~[incr Tcl]~~

Brainfuck

Whitespace

---

fooFunction ← camel case — JavaScript

foo_function ← snake case — Python

```
for (let i = 0; i < N; i++) {
    arr[i]
}

for (let x in arr) {
}

for (let x of obj) {
}
```

```
arr = [1, 2, 3]

arr.forEach(
    function(item, index){
        console.log(item)
    }
)
```

```
1
2
3
```

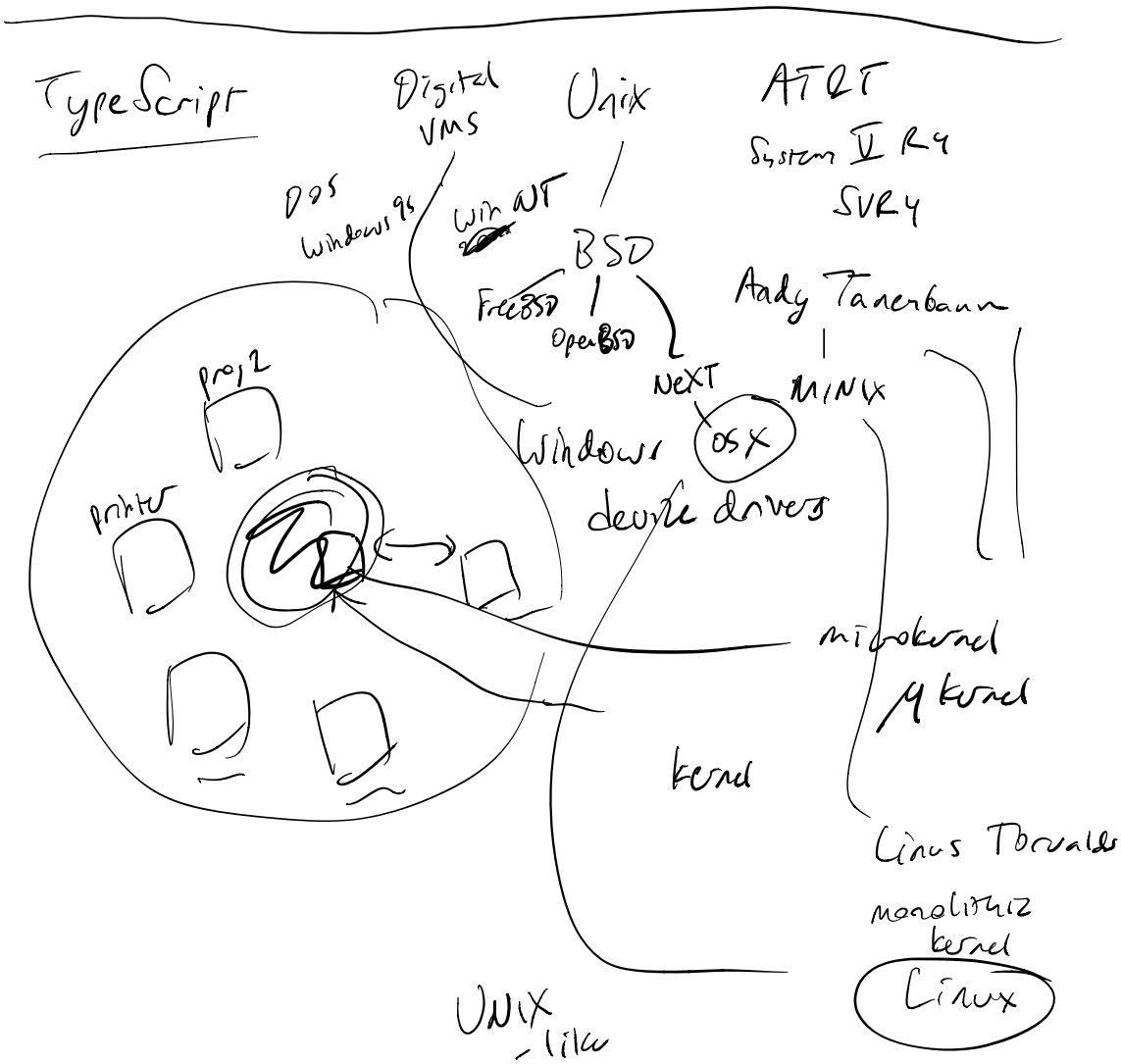
sort()

"10"
"2"

> 2 < 10
true

> "2" < "10"
false

> "a" < "2"



→ open a file for writing
↓ write ~~~
↳ close a file

synchronous
"blocking"

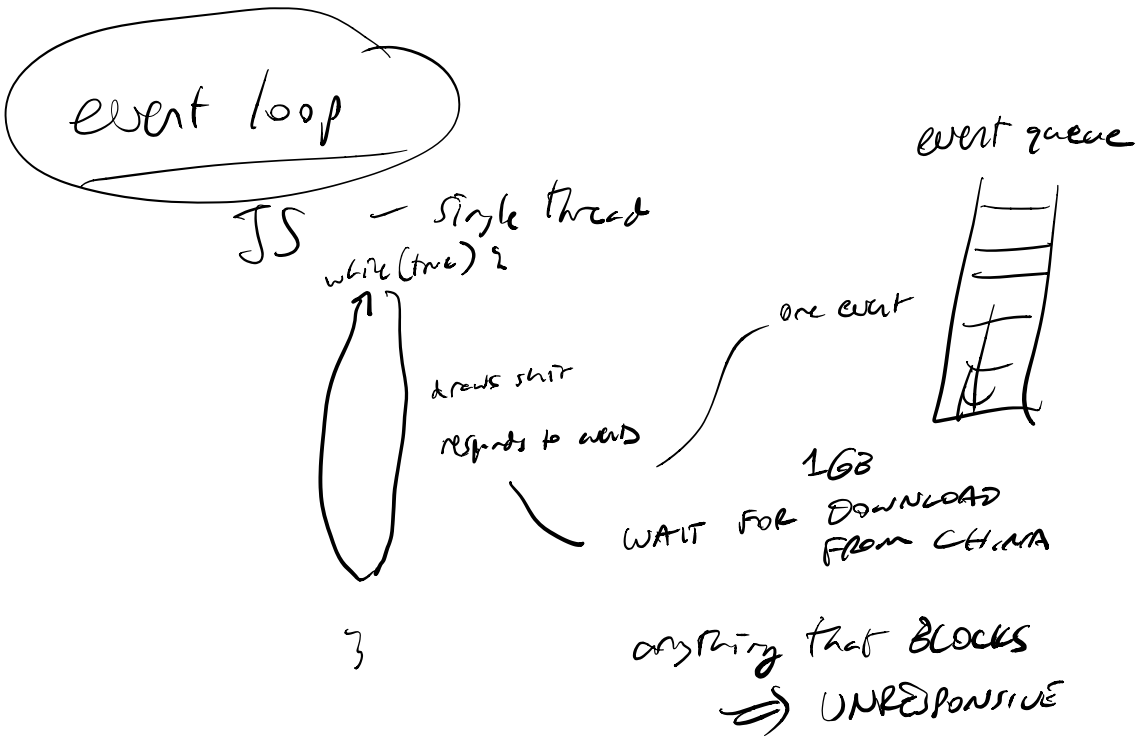
event-driven
asynchronous

WEB NEVER BLOCKS

```
x = 1
x = x + 1
x = x + 2
```
) CPU

I/O

responsiveness



anything that BLOCKS
⇒ UNRESPONSIVE

All of the APIs → I/O

NON BLOCKING / ASYNCHRONOUS / EVENT-DRIVEN

openAGile(() ⇒ { nextStuff })

with (() ⇒ ...)

do something & then do the next thing

Continuation

open(() ⇒ { read(() ⇒ { use data; ~~close()~~ write(() ⇒ ...) }

---

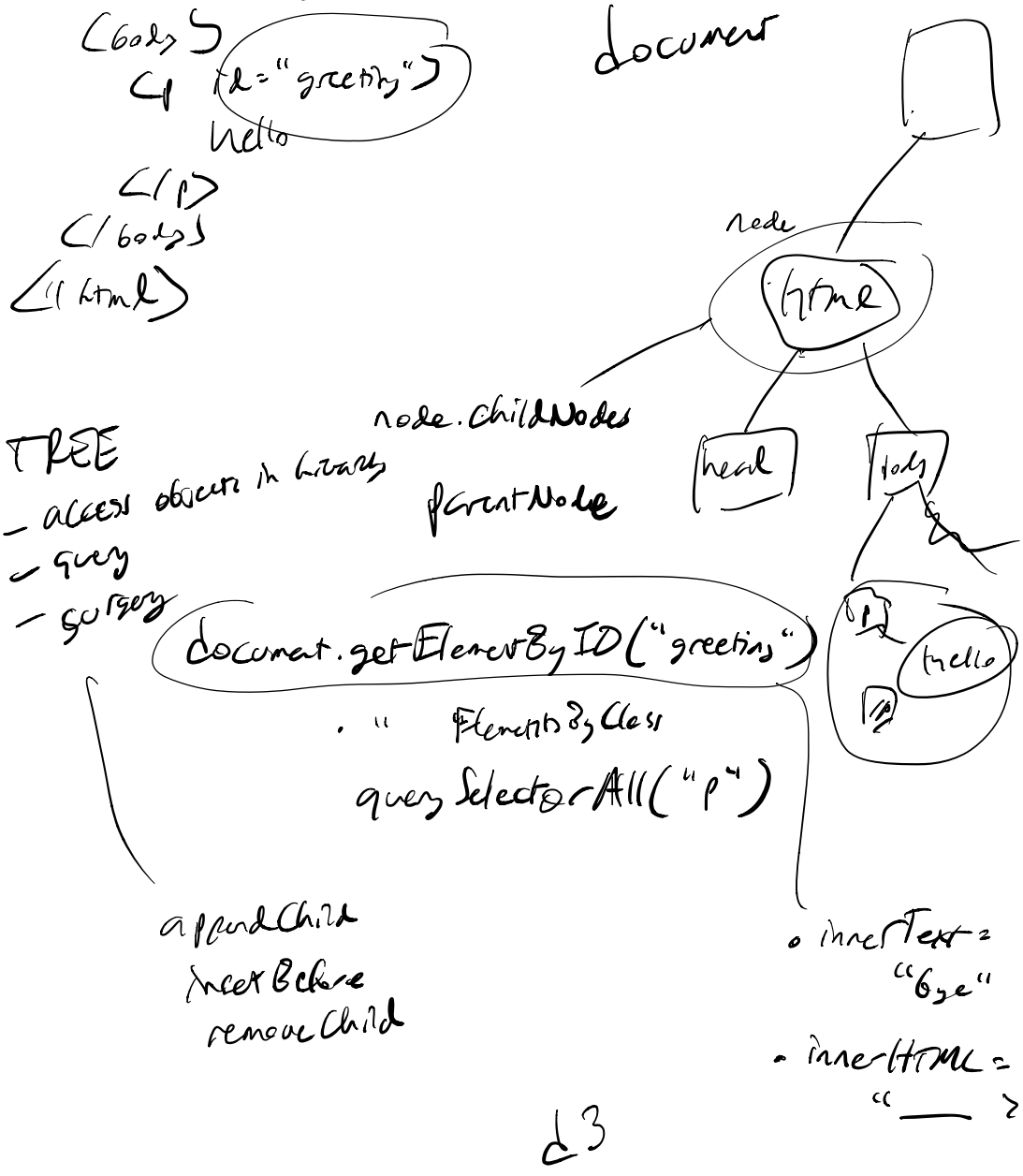
three globals

document
window
console — console.log("—")

browser
window.location — URL URL
window.open
innerHeight / width

DOM

document object model



d3

Events — Usually triggered by user or program

mouse movement
key press
clicks
network responses

attach event listener "mouseenter" → do something

```
doc. setEbyID ("greeting"). addEventListener("mouse enterr",
(e) => {
    console.log("WTF...")
});
```

e = event object

Google Analytics

AJAX

Asynchronous JS and ~~XML~~ JSON

XHR

client JS → sends

event network response () => { }

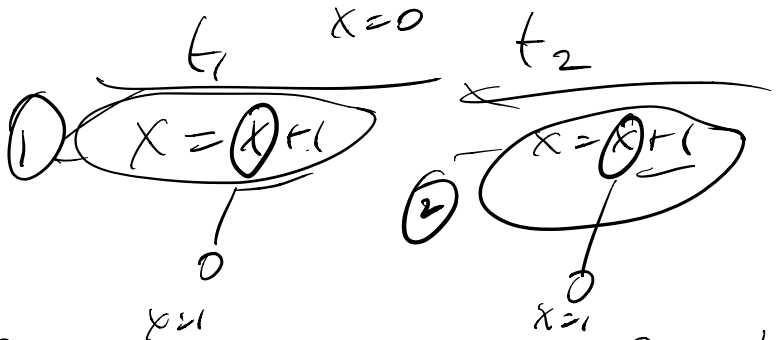
HTML DOM

node.addEventListener(" ", () => { })

- mouseenter mouseleave
- focus
- blur
- change
- input
- submit
- load

- item 1
- item 2
- item 3

timer



race condition

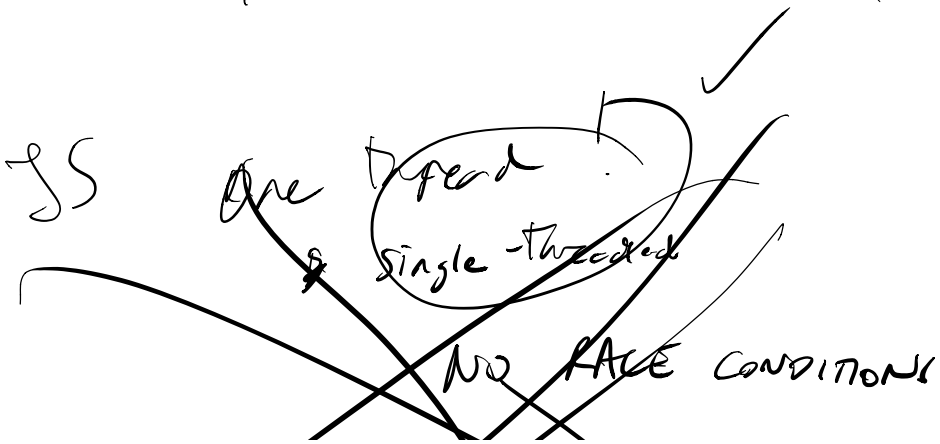
x=2

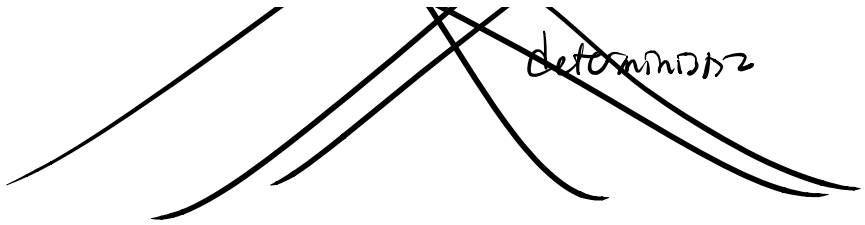
0 1 2

100

999999

C++ Memory Model





ASYNC

form validation

addEventListener("submit", (e) => { e.preventDefault(); })

XHR

xhr = XMLHttpRequest()

xhr.open("GET", "foo.com/___")

API token

xhr.send()

xhr.addEventListener("load", { xhr.response })

REST APIs

*aaS