WEEK 14

CS 105
CS @ Illinois

WEEK 14

ANNOUNCEMENTS

14 04.17.2017	Monday • Due: MP 6 (3:00 pm) • Due: Activity 11 (at the start of your LAB) • Out: MP 7	Javascript III • JavaScript cheatsheet • Lecture handout	Labs: Mon-Wed. • Out: LAB 11	Friday • TBD:
15 04.24.2017	Monday • Due: MP 7/Lab 11 (3:00 pm)	Javascript IV	Labs: Mon-Wed.	Friday • TBD:
16 05.01.2017	Monday	Javascript V	Labs: Mon-Wed.	Friday • TBD:
Finals Week 05.08.2017	Monday		**Final Wednesday: 05.10.2017** • Foellinger Auditorium • 1:30 - 4:30 pm	

FINAL

SCHEDULE

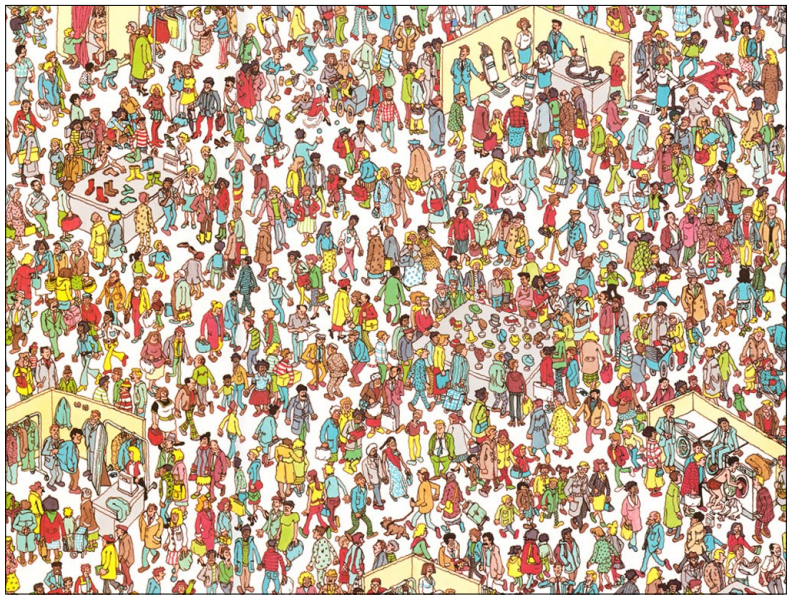
Course	Section	CRN	Date	Day	Start Time	End Time	Room
CS 105	AL2	31128	05/10/2017	W	1:30 PM	4:30 PM	AUD Foellinger Auditorium

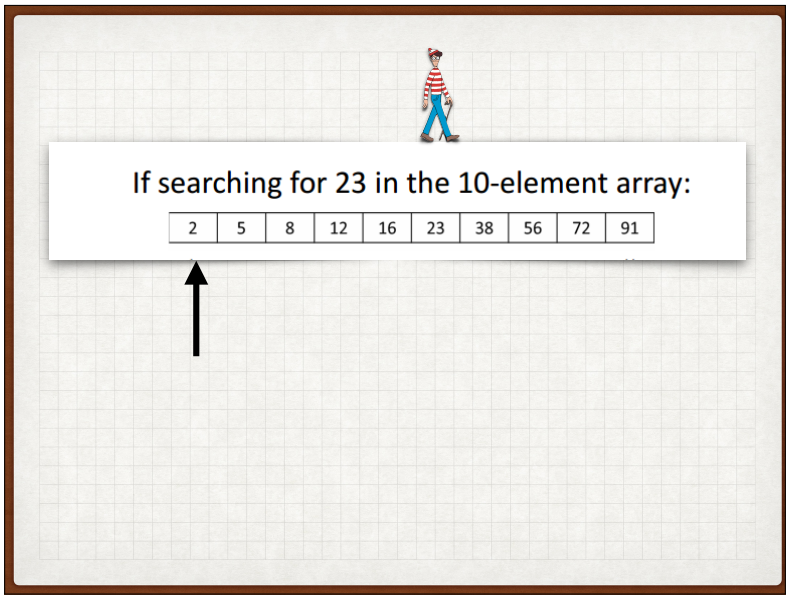
CONFLICT:
TUESDAY 8:00 AM
PLACE: TBD
[NO MORE THAN 2 FINALS]
SEND EMAIL WITH OTHER CLASSES

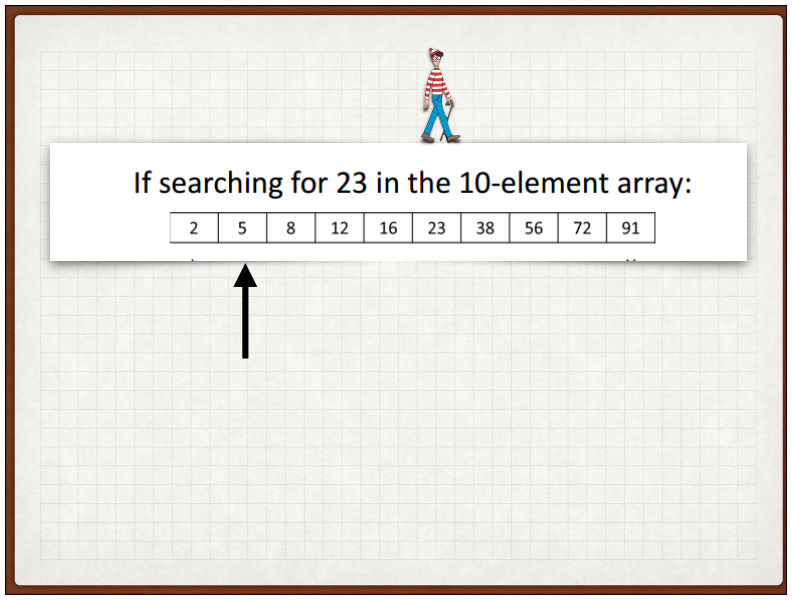
SEARCHING AND SORTING

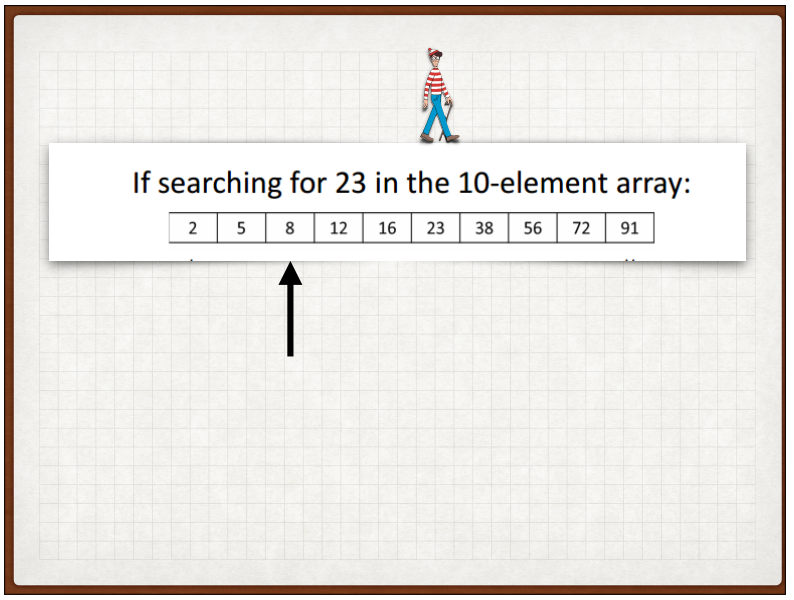
PART II

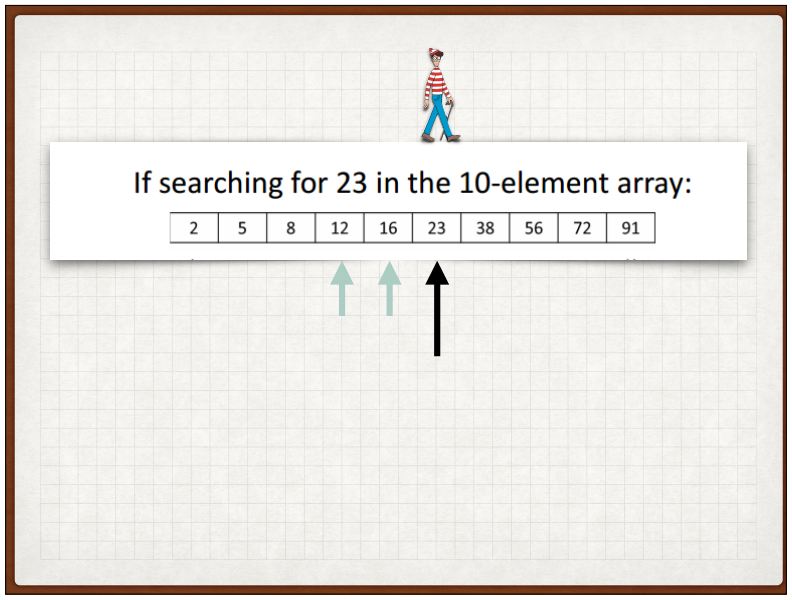
WHERE"S WALDO?











UNSORTED DATA

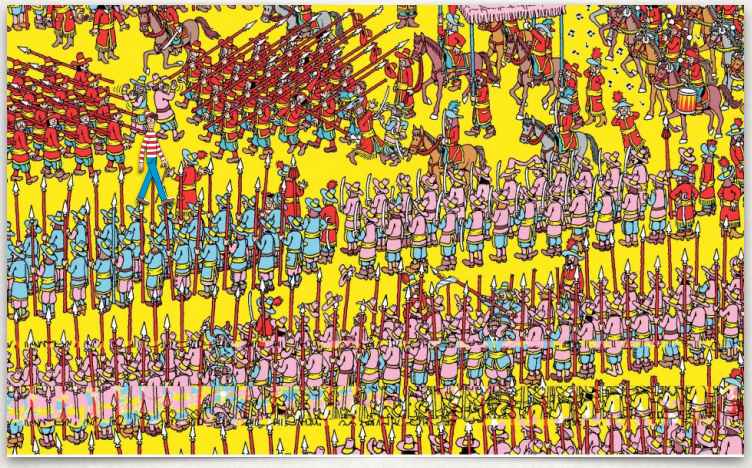
```
function linearSearch(element, items) {
  for (var i = 0; i < items.length; i++) {
    console.log("Look at", items[i]);
    if (items[i] === element) {
      console.log("FOUND at idx", i);
      return i;
    }
  }
  return -1;
}
```

UNSORTED DATA

F1 =MATCH(10572, A2:A6, 1)

	A	B	C	D	F	G
1	Order ID	Product	Unit Price	Quantity	3	
2	10567	Apples	$14.00	12		
3	10569	Oranges	$9.80	10		
4	10571	Bananas	$34.80	5		
5	10573	Pears	$18.60	9		
6	10574	Grapes	$42.30	40		
7						

```
function linearSearch(element, items) {
}
```



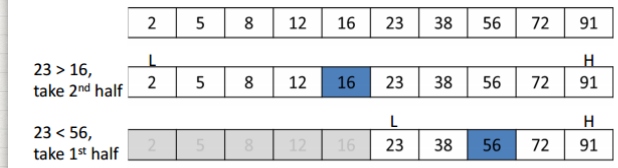
If searching for 23 in the 10-element array:

2	5	8	12	16	23	38	56	72	91

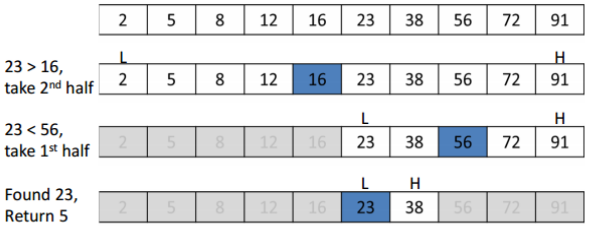
If searching for 23 in the 10-element array:

	2	5	8	12	16	23	38	56	72	91
										H
23 > 16, take 2nd half	L									
	2	5	8	12	16	23	38	56	72	91

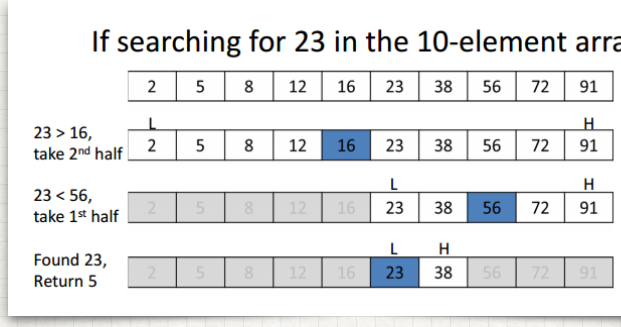
If searching for 23 in the 10-element array:



If searching for 23 in the 10-element array:



If searching for 23 in the 10-element array:



VLOOKUP (lookup_value, table_array, col_index_num, **TRUE**)

MATCH(lookup_value, lookup_array, **1**)

WHY SORT DATA:

SORTED DATA LEADS TO FASTER SEARCHING

SORTING DATA

HANDOUT

[42, 9, 8, 32, 0]

ANOTHER SORT

VISUALIZING SORTING

see https://visualgo.net/sorting

JAVASCRIPT SORT

.sort()

```
var fruit = ["pear",   "apple", "grape",
             "banana", "orange"];

console.log(fruit.sort());
```

.sort()

```
var numbers = [80,23,2,4,45,10,1];
console.log(numbers.sort());
```

.sort()

```
[ 1, 10, 2, 23, 4, 45, 80 ]
```

why?
10 < 23

["A", "10", "#", "!"].sort()

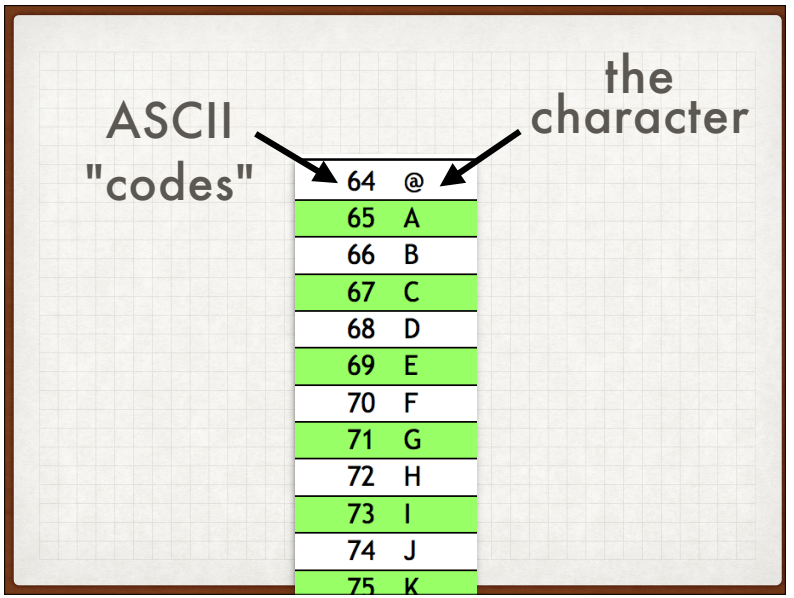
['!', '#', '10', 'A']

why ?
"#" < "A"
"!" < "#"

we need 'rules' how
to uniformly compare
values

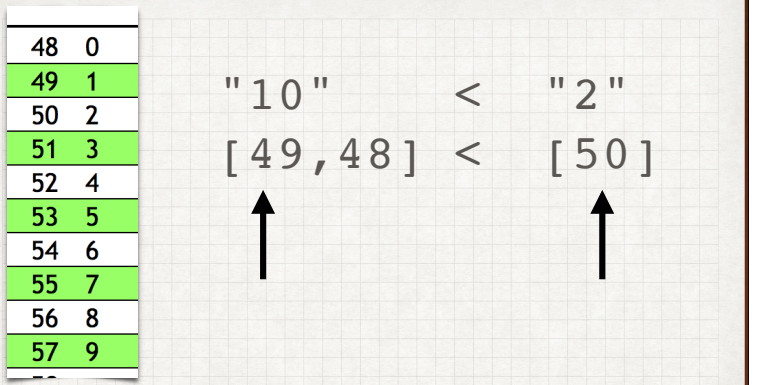
ASCII "codes"

0	NUL	16	DLE	32	SP	48	0	64	@	80	P	96	`	112	p
1	SOH	17	DC1	33	!	49	1	65	A	81	Q	97	a	113	q
2	STX	18	DC2	34	"	50	2	66	B	82	R	98	b	114	r
3	ETX	19	DC3	35	#	51	3	67	C	83	S	99	c	115	s
4	EOT	20	DC4	36	$	52	4	68	D	84	T	100	d	116	t
5	ENQ	21	NAK	37	%	53	5	69	E	85	U	101	e	117	u
6	ACK	22	SYN	38	&	54	6	70	F	86	V	102	f	118	v
7	BEL	23	ETB	39	'	55	7	71	G	87	W	103	g	119	w
8	BS	24	CAN	40	(	56	8	72	H	88	X	104	h	120	x
9	HT	25	EM	41	)	57	9	73	I	89	Y	105	i	121	y
10	LF	26	SUB	42	*	58	:	74	J	90	Z	106	j	122	z
11	VT	27	ESC	43	+	59	;	75	K	91	[	107	k	123	{
12	FF	28	FS	44	,	60	<	76	L	92	\	108	l	124	|
13	CR	29	GS	45	-	61	=	77	M	93	]	109	m	125	}
14	SO	30	RS	46	.	62	>	78	N	94	^	110	n	126	~
15	SI	31	US	47	/	63	?	79	O	95	_	111	o	127	DEL



48	0
49	1
50	2
51	3
52	4
53	5
54	6
55	7
56	8
57	9

```
"10"      <   "2"
[49,48]   <   [50]
```



ASCII "codes"

64	@
65	A
66	B
67	C
68	D
69	E
70	F
71	G
72	H
73	I
74	J

Letter A: value 65

we need rules to convert EVERYTHING into something a computer can use

(read, write, send, receive)

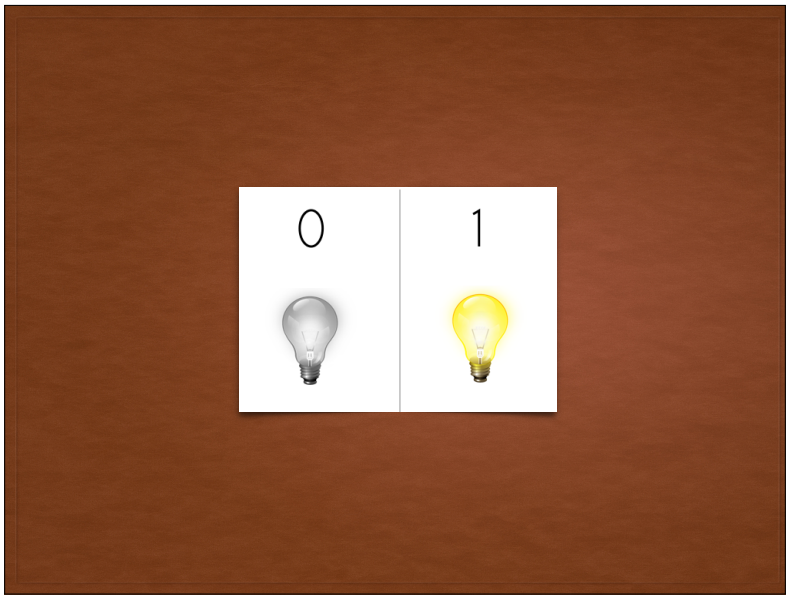
computers represent **all** data in a common format

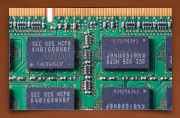
(memory, hard drives, storage, communications)

So every type of data needs to be converted into the same format

What format?

Why Binary?



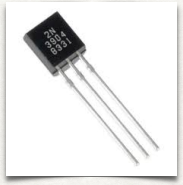


MEMORY

DEEP DIVE: PART II

MEMORY DEEP DIVE: PART II

TRANSISTORS (ELECTRICAL SWITCH)



ONLY KEEP TRACK OF 1 THING

is it ON or OFF

A BIT: the smallest unit to remember

1 bit computer:
remember 2 items:
Off
On

off: 0

on: 1

FINITE CONTAINERS

EACH MEMORY "CELL" HAS ONLY 2 STATES: ON, OFF

- 1 bit computer:
  - store 0 or 1
  - 2 possible values == 2^1
  - 1 is the biggest number
  - bit patterns:

```
0: [0]
1: [1]
```

2 bit computer:
remember 4 items

FINITE CONTAINERS

EACH MEMORY "CELL" HAS ONLY 2 STATES: ON, OFF

- 2 bit computer:
  - 2 bits each can only store a 0 or 1
  - $2^2 = 4$ possible values
  - 3 biggest number
  - bit patterns:

0: [0][0]
1: [0][1]
2: [1][0]
3: [1][1]

3 bit computer: remember 8 items

FINITE CONTAINERS

EACH MEMORY "CELL" HAS ONLY 2 STATES: ON, OFF

- 3 bit computer:
  - 3 bits each can only store a 0 or 1
  - $2^3 = 8$ possible values
  - largest is 7
  - bit patters:

0: [0][0][0]
1: [0][0][1]
2: [0][1][0]
3: [0][1][1]
4: [1][0][0]
5: [1][0][1]
6: [1][1][0]
7: [1][1][1]

FINITE CONTAINERS

- 1 bit computer: $2^1 = 2$ possible values
- 2 bit computer: $2^2 = 4$ possible values
- 3 bit computer: $2^3 = 8$ possible values
- 4 bit computer: $2^4 = 16$ possible values
- 8 bit computer: $2^8 = 256$ possible values

ACSII: 7 bit code/UTF-8
8 bits ==> byte

FINITE CONTAINERS

- 8 bit computer: 2^8 = 256 possible values
- 9 bit computer: 2^9 = 512 possible values
- 10 bit computer:2^8 = 1024 possible values

- 32 bit computer: 2^{32} possible values
- 64 bit computer: 2^{64} possible values

binary encode

A QUICK REVIEW

2148 == 2000 + 100 + 40 + 8

3051 == 3000 + 50 + 1

A QUICK REVIEW

2148 == 2000 + 100 + 40 + 8
$(10^3 * 2)$ +
$(10^2 * 1)$ +
$(10^1 * 4)$ +
$(10^0 * 8)$

3051 == 3000 + 50 + 1
$(10^3 * 3)$ +
$(10^2 * 0)$ +
$(10^1 * 5)$ +
$(10^0 * 1)$

4 bit computer: remember 16 items

ENCODE 10 IN BINARY USING 4 BITS

What is 10 in binary ?

$10 == 8 + 2$
$(2^3 * 1) +$
$(2^2 * 0) +$
$(2^1 * 1) +$
$(2^0 * 0)$

ENCODE 15 IN BINARY USING 4 BITS

What is 15 in binary ?

$15 == 8 + 4 + 2 + 1$
$(2^3 * 1) +$
$(2^2 * 1) +$
$(2^1 * 1) +$
$(2^0 * 1)$

8 BIT CONTAINER

Decode Process

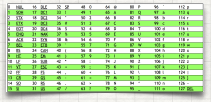
binary

↓

65

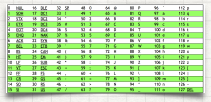
↓

A



Encode Process

A



↓

65

↓

binary

Encode/Decode process also tells us HOW much (ascii 1 byte at a time)

ASCII "codes"

Number 65:

64	@
65	A
66	B
67	C
68	D
69	E
70	F
71	G
72	H
73	I
74	J
75	K
76	L
77	M

65

letter 'A'

48	0
49	1
50	2
51	3
52	4
53	5
54	6
55	7
56	8
57	9

54,53

number 65

The file 'type' hints
which Encode/Decode
process to use:
ascii (.html, .txt, .js)
jpeg
mp3

Answer: JavaScript
converts all items into
strings and then does
a character for
character comparison
using the ascii value

```
var numbers = [34,42,6,7,2,10];

function numberCompare(a,b)
{
}

numbers.sort( numberCompare ) ;
```

JAVASCRIPT OBJECT

?

	A	B	C	D	E
1	ID	Product	Month	Region	Revenue
2	d01n	Darjeeling Tea	January	North	$43,234
3	o01e	Oolong Tea	January	East	$24,140
4	o01w	Oolong Tea	January	West	$24,164
5	o01s	Oolong Tea	January	South	$28,371
6	d01s	Darjeeling Tea	January	South	$43,018
7	e01s	Earl Grey Tea	January	South	$49,713
8	e01e	Earl Grey Tea	January	East	$50,074
9	e01n	Earl Grey Tea	January	North	$54,345
10	o02e	Oolong Tea	February	East	$26,287
11	d02w	Darjeeling Tea	February	West	$42,347
12	d02e	Darjeeling Tea	February	East	$45,805
13	e02e	Earl Grey Tea	February	East	$53,007
14	e02s	Earl Grey Tea	February	South	$54,629
15	o03s	Oolong Tea	March	South	$24,376
16	o03w	Oolong Tea	March	West	$24,813
17	d03n	Darjeeling Tea	March	North	$37,103
18	d03w	Darjeeling Tea	March	West	$40,691
19	d03s	Darjeeling Tea	March	South	$45,044
20	d03e	Darjeeling Tea	March	East	$45,359
21	e03e	Earl Grey Tea	March	East	$51,096
22	o04n	Oolong Tea	April	North	$36,770
23	e05w	Earl Grey Tea	May	West	$50,092
24	o06n	Oolong Tea	June	North	$39,246
25	e06s	Earl Grey Tea	June	South	$54,716

	A	B	C	D	E
1	ID	Product	Month	Region	Revenue
2	d01n	Darjeeling Tea	January	North	$43,234
3	o01e	Oolong Tea	January	East	$24,140
4	o01w	Oolong Tea	January	West	$24,164
5	o01s	Oolong Tea	January	South	$28,371
6	d01s	Darjeeling Tea	January	South	$43,018
7	e01s	Earl Grey Tea	January	South	$49,713
8	e01e	Earl Grey Tea	January	East	$50,074
9	e01n	Earl Grey Tea	January	North	$54,345
10	o02e	Oolong Tea	February	East	$26,287
11	d02w	Darjeeling Tea	February	West	$42,347
12	d02e	Darjeeling Tea	February	East	$45,805
13	e02e	Earl Grey Tea	February	East	$53,007
14	e02s	Earl Grey Tea	February	South	$54,629
15	o03s	Oolong Tea	March	South	$24,376
16	o03w	Oolong Tea	March	West	$24,813
17	d03n	Darjeeling Tea	March	North	$37,103
18	d03w	Darjeeling Tea	March	West	$40,691
19	d03s	Darjeeling Tea	March	South	$45,044
20	d03e	Darjeeling Tea	March	East	$45,359
21	e03e	Earl Grey Tea	March	East	$51,096
22	o04n	Oolong Tea	April	North	$36,770
23	e05w	Earl Grey Tea	May	West	$50,092
24	o06n	Oolong Tea	June	North	$39,246
25	e06s	Earl Grey Tea	June	South	$54,716

```
var ids      = ["d01n", "o01e", . .]
var products = ["Darjeeling Tea","Oolong Tea
var months   = ["Jan", "Jan", . . ];
var regions  = ["North", "East", .. ]
var revenue  = [43234, 24140, .. ]

item1 = ids[0], products[0], months[0] regi
item2 = ids[1], products[1], months[1], regi
```

```
var item1ID      = "d01n";
var item1Product = "Darjeeling Tea";
var item1Mmonth  = "Jan";
var item1Region  = "North";
var item1Revenue = 43234;

var item2ID      = "o01e";
var item2Product = "Oolong Tea";
var item2Mmonth  = "Jan";
var item2Region  = "East";
var item2Revenue = 24140;
```

JAVASCRIPT OBJECT

```
var item1 = {
  id:      "d01n",
  product: "Darjeeling Tea",
  month:   "Jan",
  region:  "North",
  revenue: 43234
 };

 var item2 = {
  id:      "o01e",
  product: "Oolong Tea",
  month:   "Jan",
  region:  "East",
  revenue: 24140
 };
```

```
var items = [item1, item2];

var items = [
{id:"d01n", product:"Darjeeling Tea", month:"Jan", region:"North", revenue:43234},
{id:"o01e", product:"Oolong Tea", month:"Jan", region:"East",revenue:24140}

];

items[0].product === "Darjeeling Tea"
items[1].revenue === 24140
```

```
var teaSales = [
{id:"d01n", product:"Darjeeling Tea", month:"Jan", region:"North", revenue:43234},
{id:"o01e", product:"Oolong Tea", month:"Jan", region:"East",revenue:24140}

];

function sumRevenue(items) {
  var total = 0;
  for (var i = 0; i < items.length; i++) {
    var tea = items[0];
    total = total + tea.revenue;
  }
  return total;
}

var totalRevenue = sum(teaSales);
```