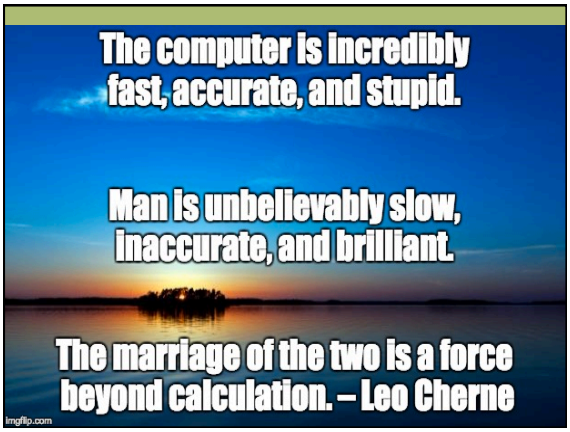
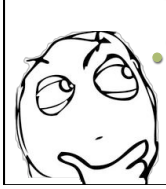
JavaScript Logic



Logic is the stuff that makes software software

Conditional statements
&
Looping



Conditional statements

if statements

```
if ( perryIsHere ) {
     say("Oh, there you are, Perry.");
}
else {
     say("Where's Perry?");
}
```

JavaScript has no elseif or elsif or elif

```
if ( boolean ) {
     doStuff();
}
else if (anotherBoolean) {
     doOtherStuff();
}
else {
     doSomeOtherStuff();
}
```

Switch statements are more abstract ways of doing multiple else/ifs

```
switch (catchPhrase) {
  case "Whatcha doin?":
    buildTodaysProject();
    break;
  case "Aren't you too young?":
    alert("Yes. Yes I am.");
    break;
  case "Curse you Perry!":
    evilPlotFail();
    break;
  default:
    tellMom();
}
```

Switch fallthrough

```
switch (x) {
  case 'a':
  case 'c':
    // Do C stuff
  case 'd':
    // Do D stuff
    break;
  case 'b':
    // Do B stuff
    break;
  default:
    // Do other stuff
}
```

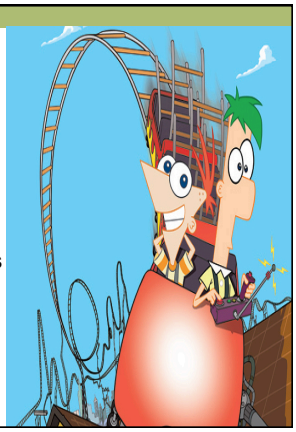
Use the ternary operator

```
(condition) ? ifTrue : ifFalse ;
var status = (a == 'correct') ? 'good' : 'bad';
```

Loops

There are two kinds of loops ...

1. Continuous loops
   - Done with *while* loops
2. Discrete loops
   - Done with *for* loops



while loops

```
while (somethingTruthy) {
  doStuff();
}
```

for loops are just like in Java, C, C++, C#, perl, etc. etc.

```
for (initializer; continue criteria; incrementer) {
    doStuff();
}
```

- For example

```
for (var i = 1 ; i < 10 ; i++) {
  document.write("<p>" + i + "</p>");
}
```

Loop Control

break

- Exits the loop

```
while (x < 100) {
  x = x + 1;
  if (x === 67){
     break; // breaks out of loop completely
  }
  document.write( x + "<br />");
}
```

continue

- Go to the top of the loop

```
while (x < 100) {
  x = x + 1;
  if (x === 67){
     continue; // go to the top
  }
  document.write( x + "<br />");
}
```

Looping through arrays

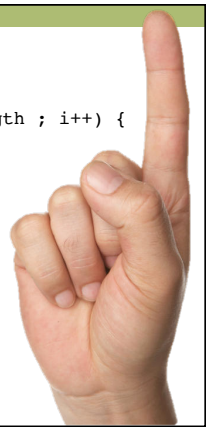
Top four ways to loop through arrays

1. Old-school for
2. for-in
3. forEach()
4. for-of

The Old-school for loop

```
for (var i=0 ; i < array.length ; i++) {
  console.log(array[i]);
}
```

IE 1+



for-in

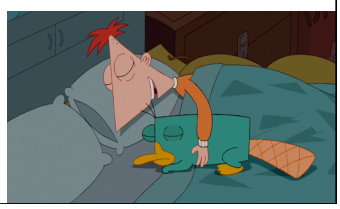
```
for (var i in array) {
  console.log(array[i]);
}
```

IE 6+

for-in was written to loop through objects

- Arrays just happen to work also!
- We can loop through the object keys with *for-in*

```
for (var prop in destructinator) {
    console.log(destructinator[prop]);
}
```



But be careful! *for-in* can be wrong.

```
var theGang= ["Buford", "Baljeet",
              "Isabella", "Ferb", "Phineas" ];
Array.prototype.todaysProject = "Backyard Beach";
for (var kid in theGang) {
  console.log(theGang[kid]);
}
//includes "Backyard Beach"
```

forEach

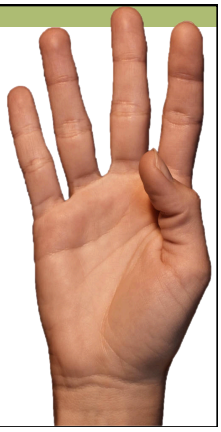
```
array.forEach(function (a) {
  console.log(a);
}
```

IE 9+

for-of

```
for (var a of array) {
  console.log(a);
}
```

Edge 13+




```
for (let kid of theGang) {
  console.log(kid);
});
// Does not include "Backyard Beach"
```

Best practice: Use the new *for of* to loop through arrays

Array functions

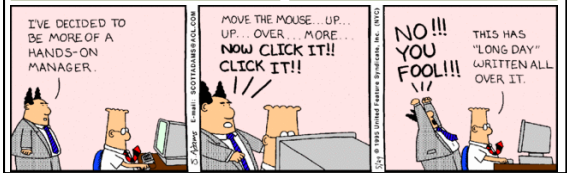
Looping arrays is usually too imperative!

```
let allAdults = true;
for (let p of people) {
  if (p.age<18)
    allAdults=false;
}
```

meh.

```
let allAdults = people
  .every(p => p.age>18);
```

Better!



Array.find()

- Returns the first thing in the array that matches some criteria

```
const searchString = "Isabella";
const p = people.find(p =>
        p.firstName === searchString);
console.log(p);

// { id:2483,
//   firstName:"Isabella",
//   lastName:"Garcia-Shapiro"
// }
```

Array.findIndex()

- Returns the location of the first thing in the array that matches some criteria

```
const searchString = "Isabella";
const p = people.findIndex(p =>
            p.firstName === searchString);
console.log(p);

// 4303
```

Array.filter()

- When you want only the members that match some criteria.
- Returns a new, smaller array based on an old one.

```
var a = [1, 2, 3, 4, 5, 6, 9];
var newArray =
  a.filter(x => x % 2 === 0);
console.log(newArray); // [2, 4, 6]
```

Array.map()

- When you want a new array the same size as an existing one.
- Processes each member of the old array.

```
var a = [1, 3, 5, 7, 9];
newArray = a.map( x => Math.pow(x, 2) );
console.log(newArray); // [1, 9, 25, 49, 81]
```

Array.reduce()

- Runs a function over each item returns a new, smaller array based on an old one.
- Great for currying!

```
var a = [21, 52, 83, 94, 15, 76, 39];
var x = a.reduce((a,b) => Math.Min(a,b) );
console.log(x); // 15
```

tl;dr

- Conditional tests
  - if
  - else if
  - else
- Loops
  - while
  - do/while
  - for
  - for-in
  - for-of
  - Array.forEach
- break and continue