Object

Objects

Object motivation

Until now, we had limited options in representing data.
We were confined to:

- Boolean
- NULL
- Undefined
- Number
- String

Object motivation 2

How can we represent complicated data?

Say we wanted to build a data type that had the info for **a person**, for example.

Ideas?

Motivation

Let's think about these 2 little guys:

Name: "*SpongeBob*" Age: 7	Name: "*Dora*" Age: 10

How can we represent them?
For example, a person can have:

- Name
- age

Objects to the rescue

- We could of just write it like this:

```
doraName = "Dora";
doraAge = 10;

bobName = "Sponge Bob";
bobAge = 7;
```

But what are the disadvantages?

1. Variable names becomes very long.
2. We could have chaos in the file if not all variables are one after the other.
3. When we have one variable we know nothing about the other ones.

Objects Examples

Object Definition:

Object = a collection of properties

In our case, a person can have:

- Name
- age

```
var smallPerson = {
    name: "Dora",
    age: 10
}
```

Property

Property

Properties

Think of an object as a bag of variables.

When a variable belongs to an object,
we call it a **property**,
as in, it is a property of the object

```
var smallPerson = {
    name: "Dora",
    age: 10
}
```

The bag

Property

Property

The bag

Objects

So, an object has properties, and a property has a key and value.
property = key + value.

```
var smallPerson = {
  name: "Dora",
  age: 10
}
```

key → name: "Dora", ← Value type string

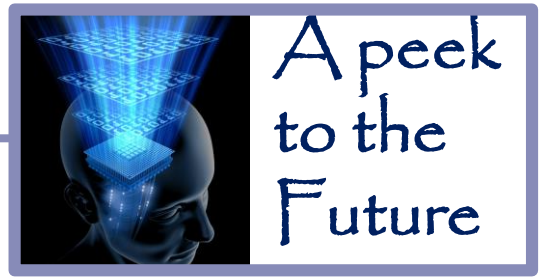
key → age: 10 ← Value type number

A property key is a string.
A property value can be any value (string, boolean, undefined, another object).

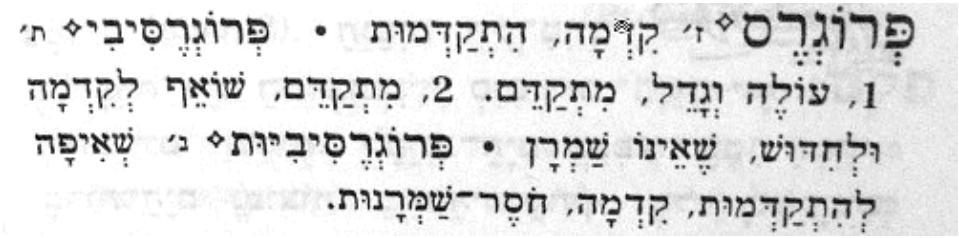
Properties

You can think of an object like a dictionary

(this is how it is called in python)



where you can find many keys(words)
each connected to a value.



Let's Look at an Example:

An object can represent a concept:

A person

A Country

An NBA player

```
var awesome = {
  name: "Manal Al Sharif",
  occupation: "IT security specialist"
}

var israel = {
  capital: "Jerusalem",
  population: 8000000
}

var lebron = {
  team: "Los Angeles Lakers",
  height: 2.03
}
```

Creating an Object

- Declaring an object example:

```
var emptyObject = {};
//this is an empty object


var user = {
   name: "Ninja Mary",
   level: 3,
   score: 1001
}
```

Objects – literal notation

When we declare an object with braces {}

```
var obj = {

}
```

() = parenthesis
[] = brackets
{} = braces

it is called literal object notation.

- The nice thing about literal notation is that the declaration of the object "looks" just like the result.

```
> var obj = {
      name: "Dora"
  }
< undefined
> obj
< ▶ {name: "Dora"}
```

Creating an object

Printing the object

Getting the value

```
var smallPerson = {
    name: "Dora",
    age: 10
}
```

We have 2 ways of getting values from objects:

1. Dot notation

```
var doraAge = smallPerson.age;
```

2. Bracket notation

```
var doraAge = smallPerson["age"];
```

Assigning a new property

```
var smallPerson = {
    name: "Dora",
    age: 10
}
```

We want to add a nick name for our object.

We have 2 ways of assigning a property to an object:

1. Dot notation

```
smallPerson.nickName = "the Explorer";
```

2. Bracket notation

```
smallPerson["nickName"] = "the Explorer";
```

Updating an object

```
var smallPerson = {
    name: "Dora",
    age: 10
}
```

It has been a while now and Dora has grown!

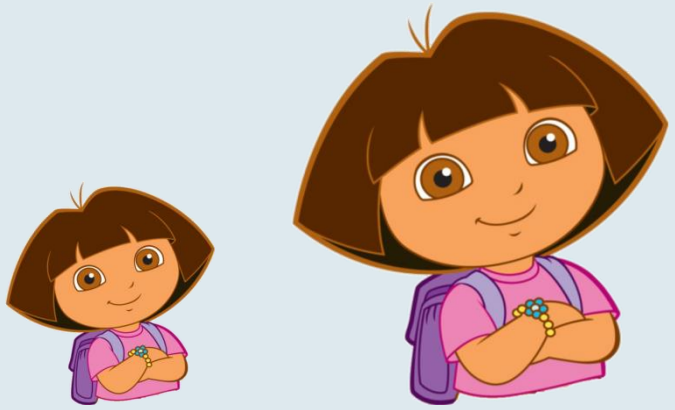
Let's update her age:

1. Dot notation

```
smallPerson.age = 12;
```

2. Bracket notation

```
smallPerson["age"] = 12;
```



Object inside an Object

How do we access the name?

```
var awesome = {
  name: "Manal Al Sharif",
  occupation: "IT security specialist"
}
```

Practice!

```
var name = awesome.name;
```

Object inside an Object

How do we access the poetry section?

```
var library = {
  poetry: {
    p12: {
      title: "Howl and Other Poems",
      author: "Allen Ginsberg",
      year: 1956
    }
  },
  fiction: {
    f34: {
      title: "Hamlet",
      author: "William Shakespeare",
      year: 1599
    },
    f52: {
      title: "Anna Karenina",
      author: "Leo Tolstoy",
      year: 1878
    }
  }
}
```

Practice!

```
var book = library.poetry;
```

Object inside an Object

How do we access book f52?

```
var library = {
  poetry: {
    p12: {
      title: "Howl and Other Poems",
      author: "Allen Ginsberg",
      year: 1956
    }
  },
  fiction: {
    f34: {
      title: "Hamlet",
      author: "William Shakespeare",
      year: 1599
    },
    f52: {
      title: "Anna Karenina",
      author: "Leo Tolstoy",
      year: 1878
    }
  }
}
```

Practice!

```
var book = library.fiction.f52;
```

Object inside an Object

How do we access the year of book p12?

```
var library = {
  poetry: {
    p12: {
      title: "Howl and Other Poems",
      author: "Allen Ginsberg",
      year: 1956
    }
  },
  fiction: {
    f34: {
      title: "Hamlet",
      author: "William Shakespeare",
      year: 1599
    },
    f52: {
      title: "Anna Karenina",
      author: "Leo Tolstoy",
      year: 1878
    }
  }
}
```

Practice!

```
var publishYear = library.poetry.p12.year
```

Questions?

```
console.log("Questions?");
```

Getting the value

We said we have 2 ways of getting values from objects:

1. Dot notation

```
var doraAge = smallPerson.age;
```

2. Bracket notation

```
var doraAge = smallPerson["age"];
```

When will we use the second way?

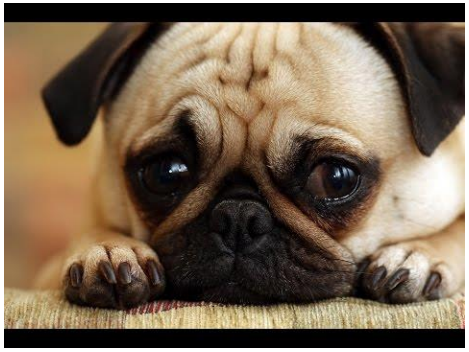
Objects to the rescue - Help

```
var collections = {
  cars: "as01t34",
  elephants: "3dff455"
}

function getCategory(number){
  if (number === 1){
    return "cars";
  } else {
    return "elephants";
  }
}


var category = getCategory(1);
collections.category; // ?
```

How can we get it right?

```
collections[category]; // yay!
```

What happened?
We tried to find a property category inside the collection object.
But we want the category value not key!



Not what we wanted

Variable as dynamics

```
var userWantsToSeeName = true;

if(userWantsToSeeName){
    console.log(obj1.name)
} else {
    console.log(obj1.powerLevel)
}

// VS

var key = userWantsToSeeName ? 'name' :'powerLevel';
console.log(obj1[key]);
```

Another Example

```
var colors = ["blue", "green", "gold"];

var colorCodes = {
  red: "d23fa4",
  gold: "w34dd3ov",
  scarlet: "2jd3wwd"
}
```

How do we get the gold code?

```
colorCodes[colors[2]]
```

Creating objects on the fly

```
function greet(person) {
    console.log('Hi ' + person.firstname);
}

greet({
    firstname: 'Jane',
    lastname: 'Doe'
}); // 'Hi Jane'
```

Questions?

```
console.log("Questions?");
```

by Value by Reference

Primitives are passed by value

```
// by value (primitives)
var a = 3;
var b;

b = a;
a = 2;

console.log(a);
console.log(b);
```

by Value by Reference

Objects are passed by reference

```
var first = { greeting: "hi"};
var second; //undefined

second = first ;// second now points to the same object first is
pointing to

first.greeting = "hello"; //changing first

console.log(first);//hello
console.log(second);//hello

second.greeting = "hola";
first = { greeting: "howdy"};//first points now on a new object

console.log(first);//howdy
console.log(second);//hola
```

Let's draw it!

Keeping track of data

- What if we have more than one object of the same type (with similar structure)?
- Before, we might have had a single user. That user was modeled as an object.

- Now we have several users:

- we want to somehow model the users as a Collection.

JS Arrays

We can use arrays of objects:

```
var student1 = {
   name: "Momo",
   averageGrade: 78
};

var student2 = {
   name: "Mimi",
   averageGrade: 98
};

var student3 = {
   name: "Mami",
   averageGrade: 88
};
```

```
var students = [
    {
        name: "Momo",
        averageGrade: 78
    },
    {
        name: "Mimi",
        averageGrade: 98
    },
    {
        name: "Mami",
        averageGrade: 88
    }
];
```

JS Arrays

- Arrays are special type of objects!
  You can access an object properties by key you can do the same with an array.

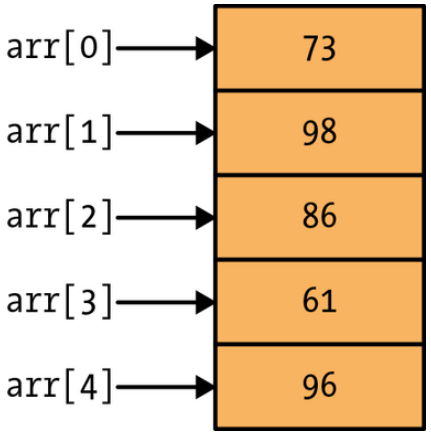
Writing:

```
var arr = [4,5,6];
```

Is just like writing:

```
var arr = {
  0: 4,
  1: 5,
  2: 6
};
```

[0]	[1]	[2]	[3]	[4]
73	98	86	61	96




```
arr["0"]; //4
```

  - Arrays have an additional option, they can store items by index.

Questions?

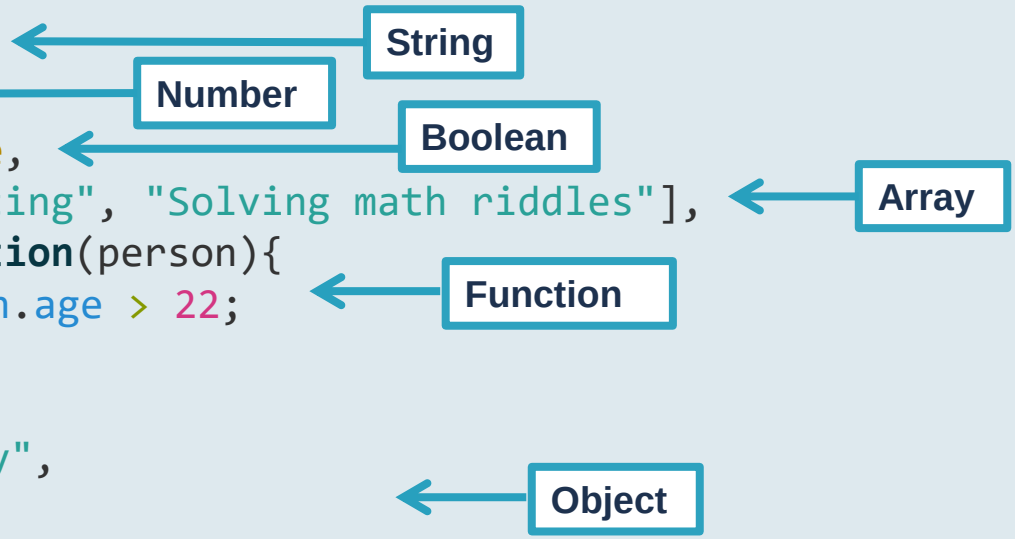
```
console.log("Questions?");
```

All Together

Now Let's Mix Everything Up!

So We can have all types (primitives and objects) as properties:

```
var person = {
  name: "Gony",
  age: 22,
  isWorking: true,
  hobbies: ["Dancing", "Solving math riddles"],
  isBigger: function(person){
    return person.age > 22;
  },
  boyfriend: {
    name: "Jeremy",
    age: 20,
    isWorking: true,
    isBigger: function(person){
      return person.age > 22;
    }
  }
};
```



All Together

How can we get Gony's hobbies?

```
var person = {
   name: "Gony",
   age: 22,
   isWorking: true,
   hobbies: ["Dancing", "Solving math riddles"],
   isBigger: function(person){
      return person.age > 22;
   },
   boyfriend: {
      name: "Jeremy",
      age: 20,
      isWorking: true,
      isBigger: function(person){
         return person.age > 20;
      }
   }
};
```

Practice!

Answer: `person.hobbies`

All Together

How can we get Gony's First hobby?

```
var person = {
   name: "Gony",
   age: 22,
   isWorking: true,
   hobbies: ["Dancing", "Solving math riddles"],
   isBigger: function(person){
      return person.age > 22;
   },
   boyfriend: {
      name: "Jeremy",
      age: 20,
      isWorking: true,
      isBigger: function(person){
         return person.age > 20;
      }
   }
};
```

Practice!

Answer: `person.hobbies[0]`

All Together

How can we get Gony's boyfriend's name?

```
var person = {
   name: "Gony",
   age: 22,
   isWorking: true,
   hobbies: ["Dancing", "Solving math riddles"],
   isBigger: function(person){
      return person.age > 22;
   },
   boyfriend: {
      name: "Jeremy",
      age: 20,
      isWorking: true,
      isBigger: function(person){
         return person.age > 20;
      }
   }
};
```

Practice!

Answer: `person.boyfriend.name`

All Together

How can we check if Jeremy is bigger than Gony?

```
var person = {
   name: "Gony",
   age: 22,
   isWorking: true,
   hobbies: ["Dancing", "Solving math riddles"],
   isBigger: function(person){
      return person.age > 22;
   },
   boyfriend: {
      name: "Jeremy",
      age: 20,
      isWorking: true,
      isBigger: function(person){
         return person.age > 20;
      }
   }
};
```

Practice!

We want to use the isBigger function!

Answer: `person.isBigger(person.boyfriend)`

Or: `person.boyfriend.isBigger(person)`

All Together

Now Let's Mix Everything Up!

We saw that objects can have functions as properties.
Function can also get objects as parameters and return objects!

For example, we have 2 food items:

```
var p1 = {
   name: "banana",
   protein_g: 1.09,
   calcium_mg: 5
}
```

```
var p2 = {
   name: "orange",
   protein_g: 0.94,
   calcium_mg: 40
}
```

And a function to create new food product:

```
function createProduct(name, item1, item2){
   var newFood = {
      name: name,
      protein_g: item1.protein_g + item2.protein_g
   }
   newFood.calcium_mg = item1.calcium_mg + item2.calcium_mg;
   return newFood;
}
```

Adding properties on object creation

Or to an existing object

All Together

Now Let's Mix Everything Up!

```
var p1 = {
   name: "banana",
   protein_g: 1.09,
   calcium_mg: 5
}

var p2 = {
   name: "orange",
   protein_g: 0.94,
   calcium_mg: 40
}

function createProduct(name, item1, item2){
   var newFood = {
      name: name,
      protein_g: item1.protein_g + item2.protein_g
   }
   newFood.calcium_mg = item1.calcium_mg + item2.calcium_mg;
   return newFood;
}
```

How can we call this function to create a fruit salad?

```
var newItem = createProduct("fruit salad"
```

What will the function call return?

```
{
   name: "fruit salad",
   protein_g: 2.03,
   calcium_mg: 45
}
```

All Together

```
var p1 = {
   name: "banana",
   protein_g: 1.09,
   calcium_mg: 5
}

var p2 = {
   name: "orange",
   protein_g: 0.94,
   calcium_mg: 40
}

function createProduct(name, item1, item2){
   var newFood = {
      name: name,
      protein_g: item1.protein_g + item2.protein_g,
      getIngredients: function(){
         return [item1.name, item2.name];
      }
   }
   newFood.calcium_mg = item1.calcium_mg + item2.calcium_mg;
   return newFood;
}
```

Let's add to the new product a function that will return the new product ingredients.

How can we get the first item in the ingredients of the new product?

```
createProduct(("fruit salad",p1, p2).getIngredients()[0]; // “banana”
```

Questions?

```
console.log("Questions?");
```

Cheat Sheet

Object

create:

```
var empty = {}; //literal notation

var obj = {
   key: "value" //property (key,value)
}
```

Get value:

```
var doraAge = smallPerson.age;
var doraAge = smallPerson["age"];
```

Assigning a new property / Update the object

```
smallPerson.nickName = "the Explorer";
smallPerson["nickName"] = "the Explorer";
```