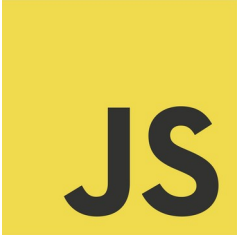


ISCG6420 JavaScript

Semester 1, 2024

Object oriented programming

The basic idea of OOP is that we use objects to model real world things that we want to represent inside our programs, and/or provide a simple way to access functionality that would otherwise be hard or impossible to make use of.

Defining and object template

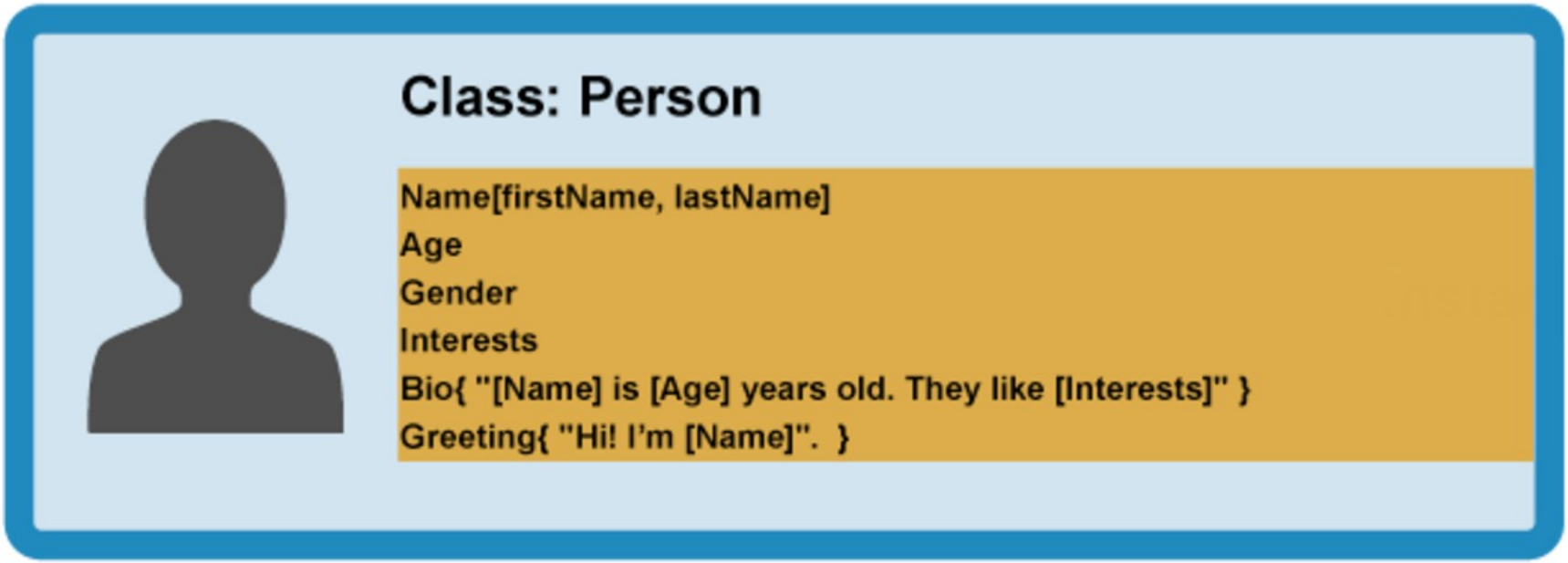
Consider a simple program that displays information about the students and lecturers at Unitec.

We may know a lot of information about a particular person, but should start with providing some very generic information, such as name, age, gender.

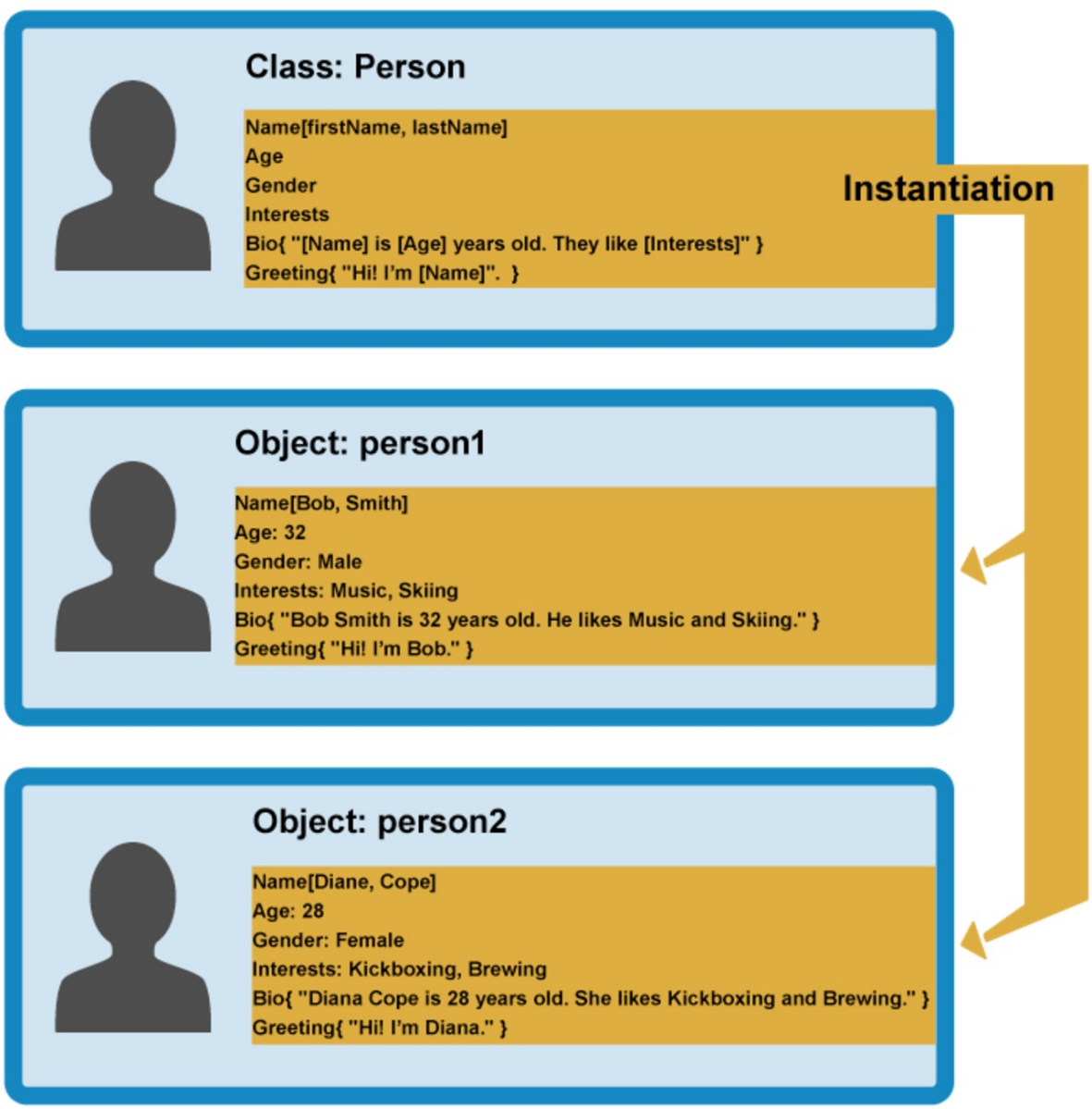
This is known as ***abstraction*** – creating a simple model of a more complex thing that represents its most important aspects in a way that is easy to work.

Unitec
Institute of Technology
TE WHARE WĀNANGA O WAIRAKA

JS



Creating actual objects

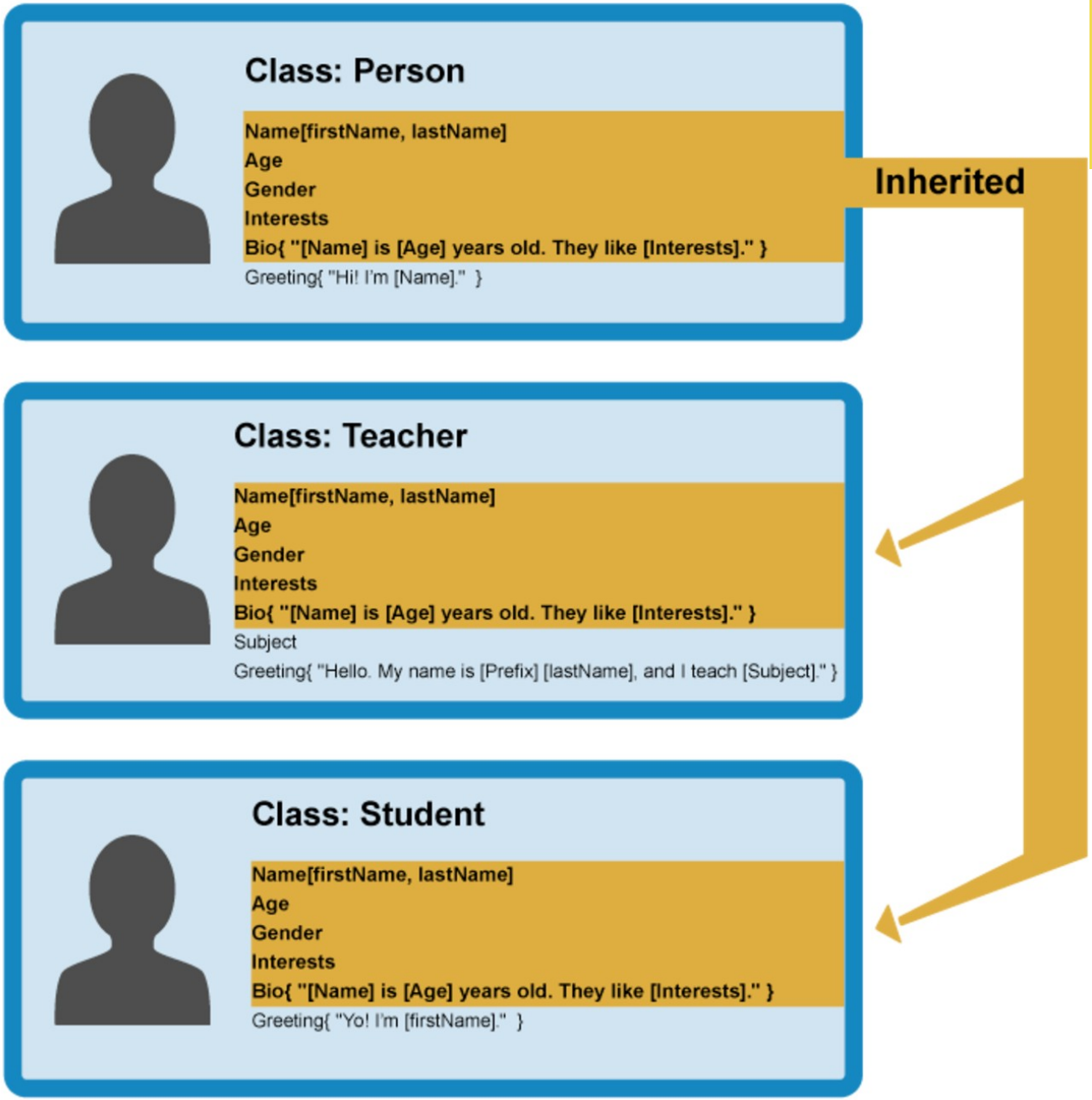


Creating actual objects

When an object instance is created from a class, then class's ***constructor function*** is run to create it. This process of creating an object instance from a class is called ***instantiation*** – the object instance is instantiated from the class.

Specialist classes

In OOP we can create classes based on other classes – these new ***child classes*** can be made to ***inherit*** the data and code features of their ***parent class,*** so you can avoid duplicating some functionality which is common to all the objects.


Class: Person
Name[firstName, lastName]
Age
Gender
Interests
Bio{ "[Name] is [Age] years old. They like [Interests]." }
Greeting{ "Hi! I'm [Name]." }
Inherited
Class: Teacher
Name[firstName, lastName]
Age
Gender
Interests
Bio{ "[Name] is [Age] years old. They like [Interests]." }
Subject
Greeting{ "Hello. My name is [Prefix] [lastName], and I teach [Subject]." }
Class: Student
Name[firstName, lastName]
Age
Gender
Interests
Bio{ "[Name] is [Age] years old. They like [Interests]." }
Greeting{ "Yo! I'm [firstName]." }

Constructor and object instances

JavaScript uses special functions called ***constructor functions*** (instead of classes) to define objects and their features.

```
function createNewPerson(name) {
  var obj = {};
  obj.name = name;
  obj.greeting = function() {
    alert('Hi! I\'m ' + this.name + '.');
  };
  return obj;
}
```

In your console:

```
var dila = createNewPerson('Dila')
undefined
dila.name
"Dila"
dila.greeting()
```

This worked well, however, we first needed to create an empty object first. Instead, we should use constructors

What is ‘this’?

This refers to the current object the code is being written inside – so in the case outlined on the previous slide ***this*** is equivalent to ***person***. Why bother? Useful when we have, say, two different ***person*** objects and want to ensure we refer to the correct ones.

```
var person1 = {
  name: 'David',
  greeting: function() {
    alert('Hi! I\'m ' + this.name + '.');
  }
}

var person2 = {
  name: 'Dila',
  greeting: function() {
    alert('Hi! I\'m ' + this.name + '.');
  }
}
```

Constructors

```
function Person(name) {
  this.name = name;
  this.greeting = function() {
    alert('Hi! I\'m ' + this.name + '.');
  };
}
```

And now in order for a constructor to create some objects, add the following to your code above:

```
var person1 = new Person('David')
var person2 = new Person('Mark')
```

```
person1.name
"David"
person1.greeting()
undefined
```

Unitec
Institute of Technology
TE WHARE WĀNANGA O WAIRAKA

```
function Person(first, last, age, gender, interests) {
  this.name = {
    first,
    last
  };
  this.age = age;
  this.gender = gender;
  this.interests = interests;
  this.bio = function() {
    alert(this.name.first + ' ' + this.name.last + ' is ' + this.age +
    ' years old. He likes ' + this.interests[0] + ' and ' + this.interests[1] + '.');
  };
  this.greeting = function() {
    alert('Hi! I\'m ' + this.name.first + '.');
  };
};

var person1 = new Person('John', 'Smith', 32, 'male', ['music', 'skiing']);
```


```
> Person.prototype
< ▼ Object i
    ▶ constructor: Person(first, last, age, gender, interests)
    ▶ __proto__: Object
> Object.prototype
< ▼ Object i
    ▶ __defineGetter__: __defineGetter__()
    ▶ __defineSetter__: __defineSetter__()
    ▶ __lookupGetter__: __lookupGetter__()
    ▶ __lookupSetter__: __lookupSetter__()
    ▶ constructor: Object()
    ▶ hasOwnProperty: hasOwnProperty()
    ▶ isPrototypeOf: isPrototypeOf()
    ▶ propertyIsEnumerable: propertyIsEnumerable()
    ▶ toLocaleString: toLocaleString()
    ▶ toString: toString()
    ▶ valueOf: valueOf()
    ▶ get __proto__: __proto__()
    ▶ set __proto__: __proto__()
```

Create()

```
> var person2 = Object.create(person1)
<· undefined
```

Create() actually does is to create a new object from a specified prototype object. Here' *person2* is being created using *person1* as a prototype object.

You can check this by typing the following:

```
> person2.__proto__
<· ▼Person i
    age: 32
  ▶ bio: ()
    gender: "male"
  ▶ greeting: ()
  ▶ interests: Array[2]
  ▶ name: Object
  ▶ __proto__: Object
>
```

The constructor property

Every function has a prototype property whose value is an object containing a *constructor* property. This constructor property points to the original constructor function.

```
> person1.constructor
< function Person(first, last, age, gender, interests) {
    this.name = {
      first,
      last
    };
    this.age = age;
    this.gender = gender;
    this.interests = interests;
    this.bio = function() {
      al…
> person2.constructor
< function Person(first, last, age, gender, interests) {
    this.name = {
      first,
      last
    };
    this.age = age;
    this.gender = gender;
    this.interests = interests;
    this.bio = function() {
      al…
```

Unitec
Institute of Technology
TE WHARE WĀNANGA O WAIRAKA

JS

Now you can create another object instance from that constructor by doing the following:

```
var person3 = new person1.constructor('Gerard', 'Lovell', 25, 'male', ['gaming', 'JS']);
undefined
```

Now try accessing your new object's features:

```
person3.name.first
"Gerard"
```

```
person3.age
25
```

```
person3.bio()
undefined
```