Codesmith

CS Prep

Day 2

Overview

POD

Recap - arrow functions

Objects

- Overview
- Access data in objects
- Object methods
- JavaScript's built-in objects and methods

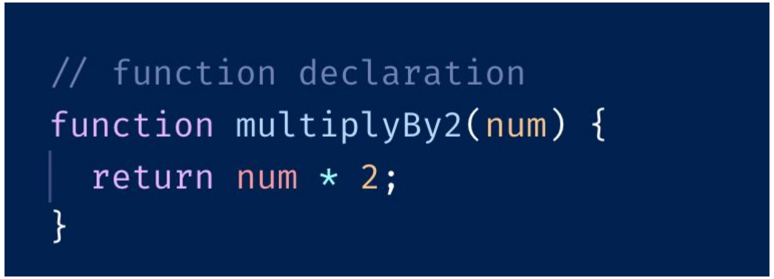
'This'

Inheritance and prototypes

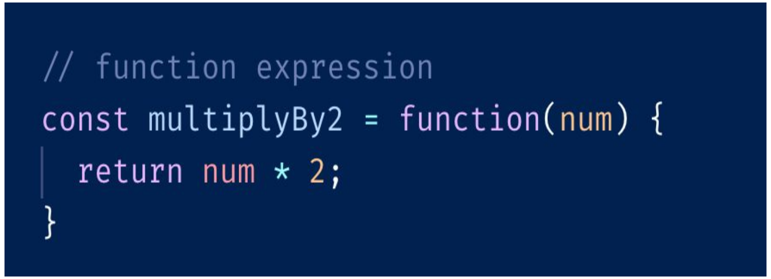
Technical communication recordings

Arrow Functions (ES6+)

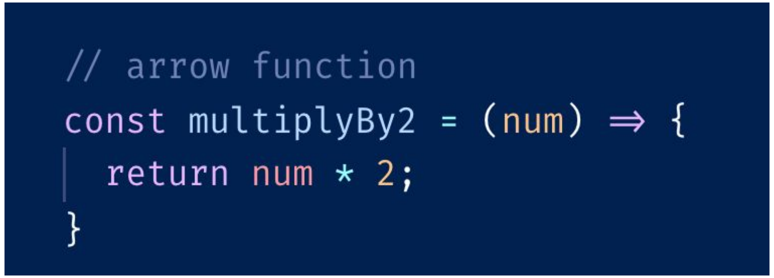
Arrow Functions and the Implicit Return



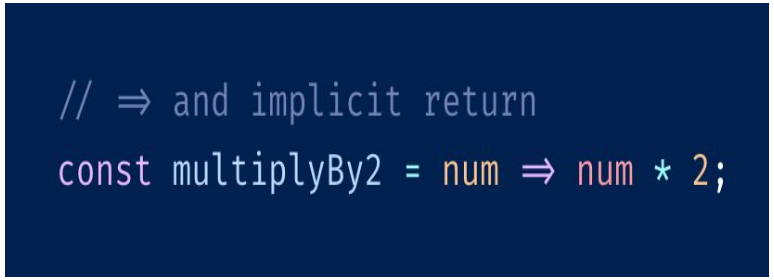
```
// function declaration
function multiplyBy2(num) {
  return num * 2;
}
```



```
// function expression
const multiplyBy2 = function(num) {
  return num * 2;
}
```



```
// arrow function
const multiplyBy2 = (num) ⇒ {
  return num * 2;
}
```



```
// ⇒ and implicit return
const multiplyBy2 = num ⇒ num * 2;
```

Objects

Under the hood...

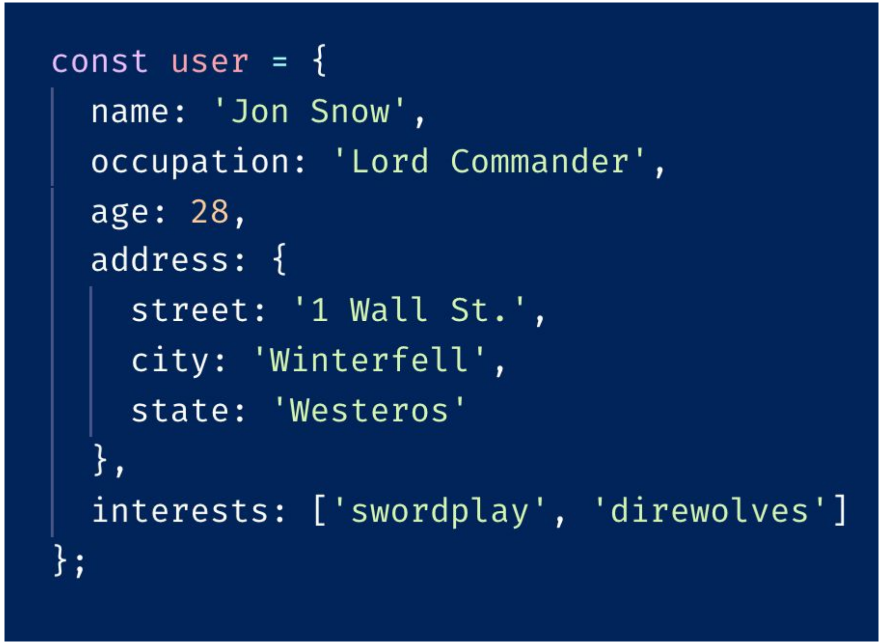
What is an object?

An object is a key-value pair data type

Keys MUST be unique and are stored as strings (even if they don't have quotes around them)

Values can be primitives or other objects

This is an object literal =>



```
const user = {
  name: 'Jon Snow',
  occupation: 'Lord Commander',
  age: 28,
  address: {
    street: '1 Wall St.',
    city: 'Winterfell',
    state: 'Westeros'
  },
  interests: ['swordplay', 'direwolves']
};
```

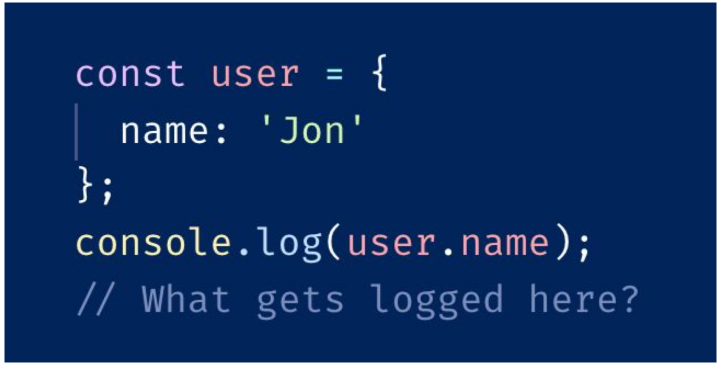
Why do we have objects?

- They allow us to store related data and functionality together

Dot vs. Bracket notation

Dot (.) notation

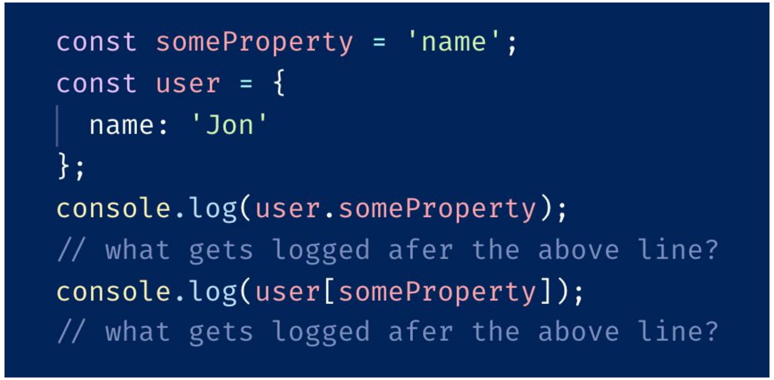
- You're retrieving or adding specific key name



```
const user = {
  name: 'Jon'
};
console.log(user.name);
// What gets logged here?
```

Bracket ([ ]) notation

- You need to use a variable or string type to represent the key name

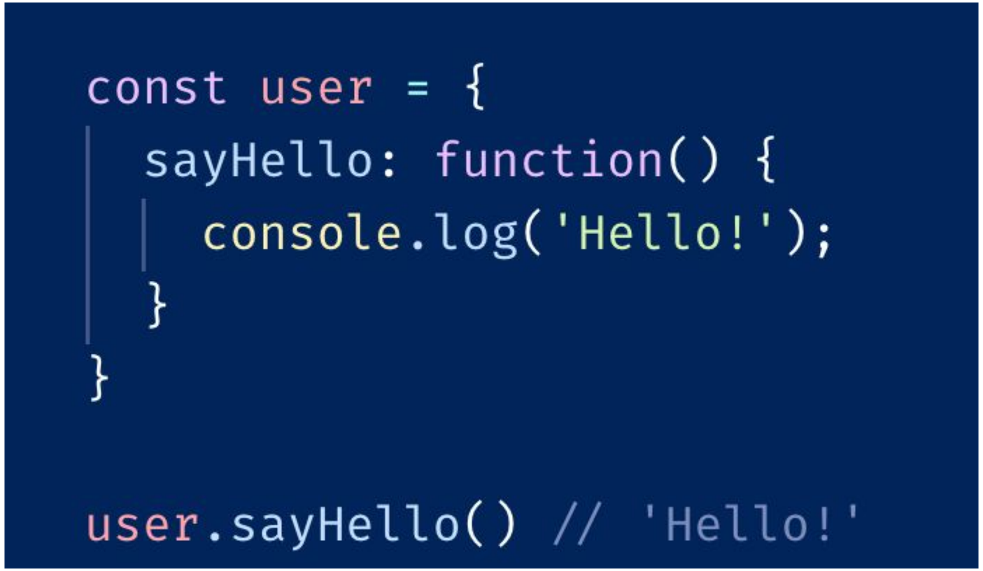


```
const someProperty = 'name';
const user = {
  name: 'Jon'
};
console.log(user.someProperty);
// what gets logged afer the above line?
console.log(user[someProperty]);
// what gets logged afer the above line?
```

Objects have methods

Properties on objects that are functions are called methods

Methods on objects can be invoked



```
const user = {
  sayHello: function() {
    console.log('Hello!');
  }
}

user.sayHello() // 'Hello!'
```

Putting it all together

littleLanister is an ____

What are the data types of the properties in here?

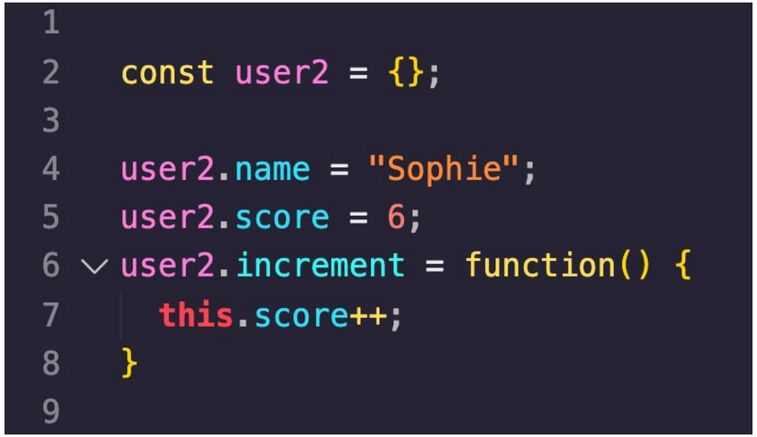
Why aren't the property names wrapped in single quotes?

What will be logged to the console at the end of this code?

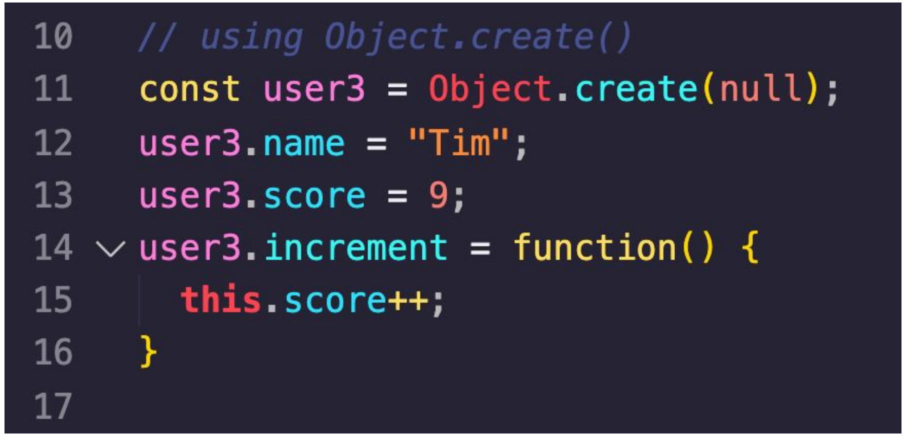
What is 'this' ?

```javascript
const littleLannister = {
  name: 'Tyrion',
  familyName: 'Lannister',
  height: 'low',
  job: 'hand of the queen',
  logCurrentJob: function() {
    console.log(`I am currently ${this.job}`)
  }
}

const somePropertyWeWant = 'familyName'
const lastName = littleLannister[somePropertyWeWant]
littleLannister.logCurrentJob();
```

We could keep creating new objects like this… but say we wanted to create hundreds of objects with these same properties.



```
const user2 = {};

user2.name = "Sophie";
user2.score = 6;
user2.increment = function() {
  this.score++;
}

```

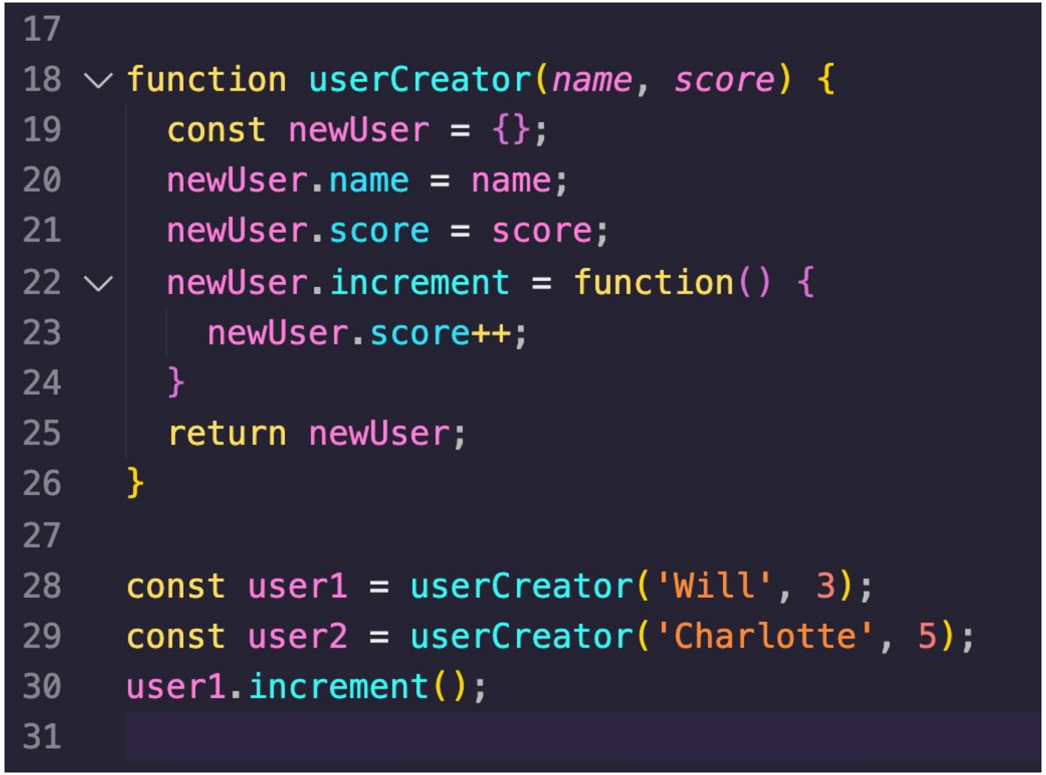


```
// using Object.create()
const user3 = Object.create(null);
user3.name = "Tim";
user3.score = 9;
user3.increment = function() {
  this.score++;
}

```

What principle are we breaking?

Generate objects using a function!

There's still an issue with this approach though…



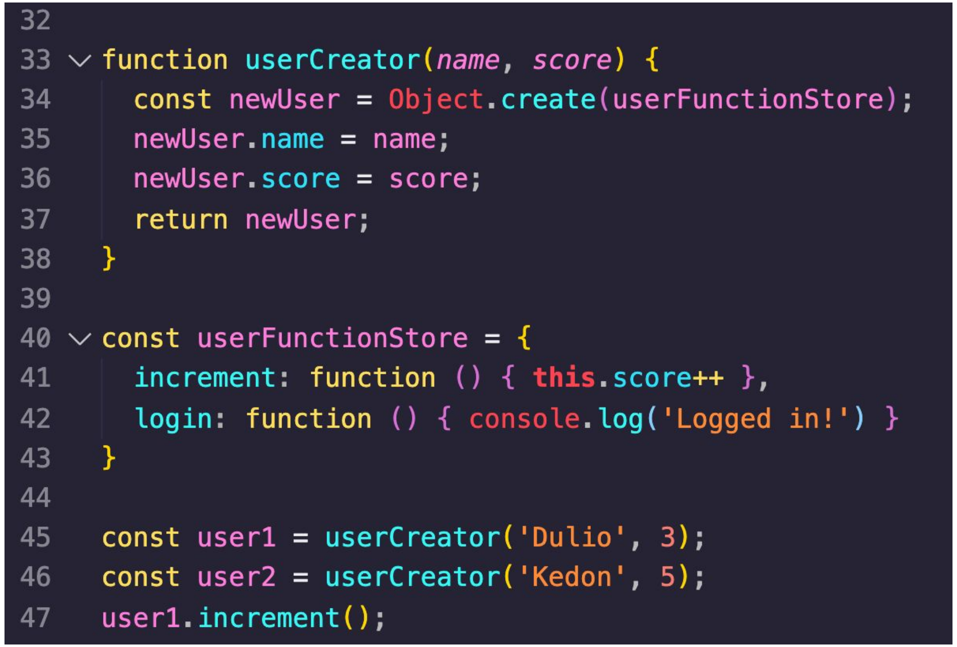
```javascript

function userCreator(name, score) {
  const newUser = {};
  newUser.name = name;
  newUser.score = score;
  newUser.increment = function() {
    newUser.score++;
  }
  return newUser;
}

const user1 = userCreator('Will', 3);
const user2 = userCreator('Charlotte', 5);
user1.increment();

```

Enter inheritance and prototypes

Using the prototypal chain - we can store the increment function in just one object and have the interpreter, if it doesn't find the function on user1, look up to that object to check if it's there.

We can make this link using the Object.create() technique.

Let's diagram it!



```javascript

function userCreator(name, score) {
  const newUser = Object.create(userFunctionStore);
  newUser.name = name;
  newUser.score = score;
  return newUser;
}

const userFunctionStore = {
  increment: function () { this.score++ },
  login: function () { console.log('Logged in!') }
}

const user1 = userCreator('Dulio', 3);
const user2 = userCreator('Kedon', 5);
user1.increment();
```

Wait…what about array.push()?
That looks like it's invoking a method on an object!
(it is)

Enter inheritance and prototypes

- Objects all inherit properties and methods from a parent object; (arrays, objects, and functions are all “children” of objects)
- These properties and methods are held on the constructor’s prototype attribute
- Creating an array with [ ] is simply a shortcut for creating an instance of the Array object, an object with properties and methods on its prototype that you can use when you create arrays.

Let’s look at inheritance and prototypes in action…

Other JavaScript objects

- We've discussed some of JavaScript's "native" (built-in) objects (everything

  JavaScript gives you e.g. Function, Array, Error)
  - Another example: Date

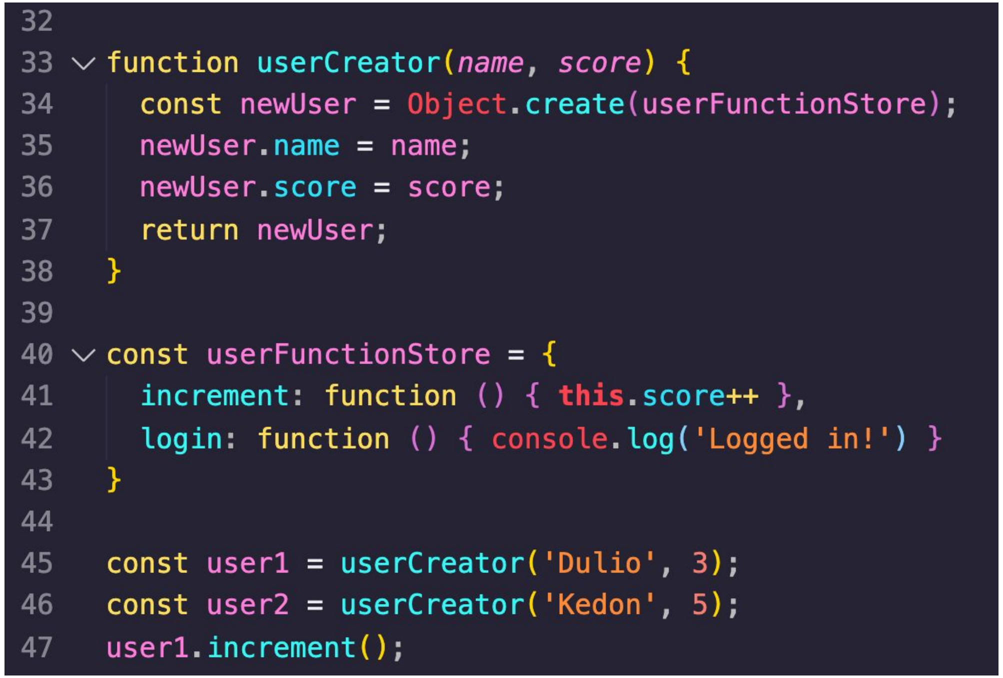
```
const time = new Date();
console.log(time.getDate()); //returns the day of the month
→ 25
```

- Next week, we'll start taking a look at "host" objects (everything a JavaScript environment gives you)
  - document

So what's important here?

- JavaScript objects have properties and methods
- Objects can inherit properties and methods from their parents
- Inheritance is a bigger topic in the world of OOP (object-oriented programming)
- Understanding how objects work is key in JavaScript and the majority of other programming languages

One last thing - arrow function scope and “this”



```
function userCreator(name, score) {
  const newUser = Object.create(userFunctionStore);
  newUser.name = name;
  newUser.score = score;
  return newUser;
}

const userFunctionStore = {
  increment: function () { this.score++ },
  login: function () { console.log('Logged in!') }
}

const user1 = userCreator('Dulio', 3);
const user2 = userCreator('Kedon', 5);
user1.increment();
```

Pair Programming Time!

Recording your technical communication

What to look for:

- Communicate overall strategy/plan + individual line-by-line description
- Verbalizing moments of confusion and blocks
- Using technical terms + intuitive terms

I hit a total block

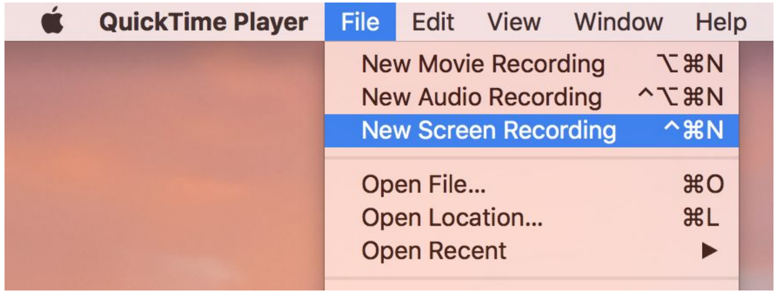
- What was the goal
- What did I try
- What actually happened
- Any suspicions why that happened

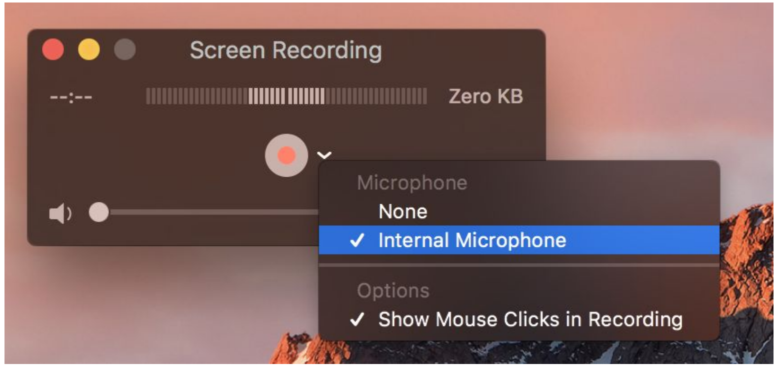
You will be comparing your technical communication at the end of the program to this recording

How to record

Screen record using Quicktime player

- New Screen Recording
- Set microphone on
- Start recording audio + screen recording code editor





Record your technical communication!

:)