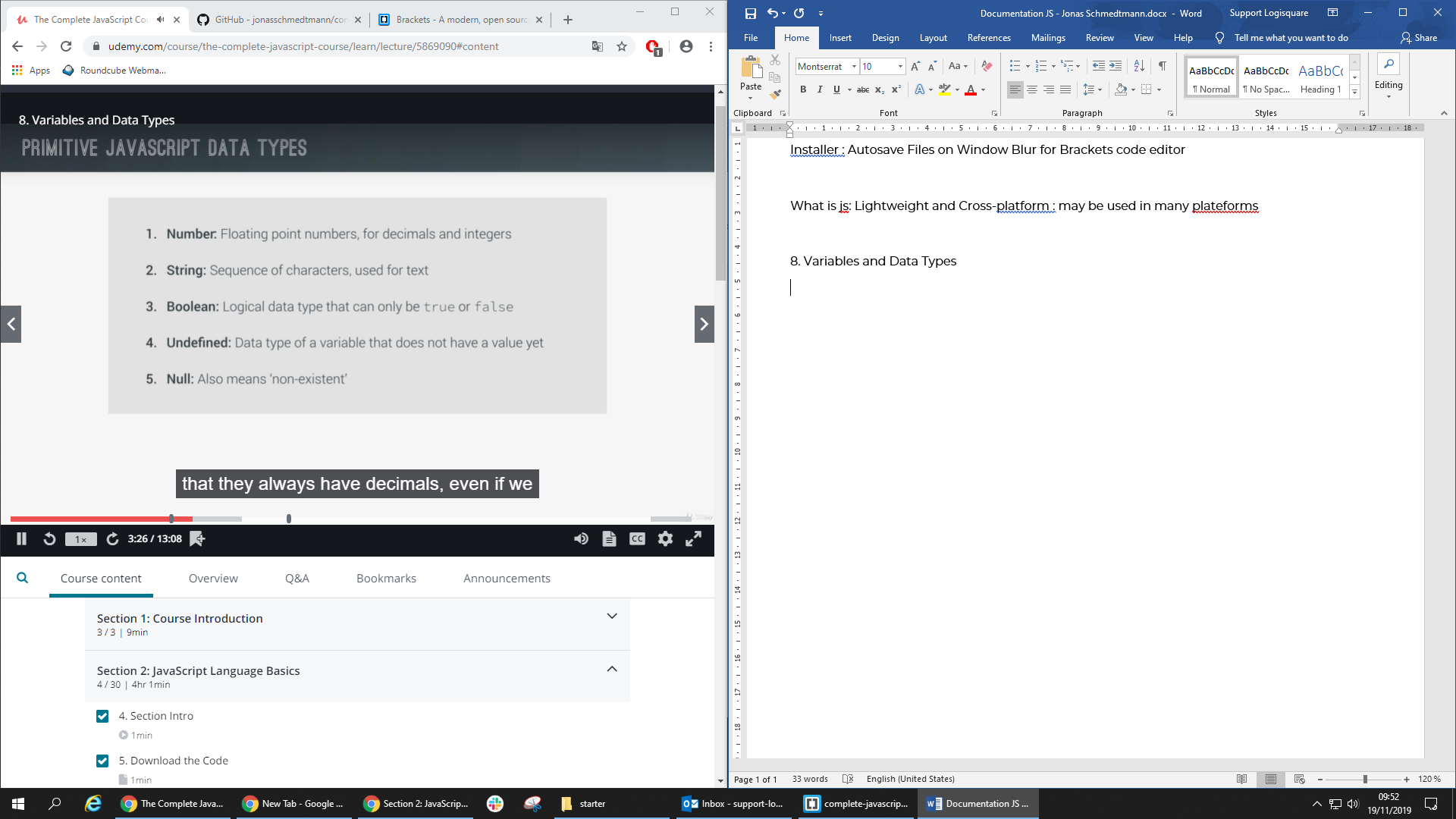
Section 1: Course Introduction

Installer : Autosave Files on Window Blur for Brackets code editor

What is js: Lightweight and Cross-platform : may be used in many plateforms

8. Variables and Data Types



All numbers are floating number even if they are integer number: 5 = 5.0

Javascript has dynamic typing : data types are automatically assigned to variables

Boolean are converted to string

* Console.log(false) 🡺 false

10. Basic Operators

var johnOlder = ageJohn < ageMark;

// typeof operator

console.log(typeof johnOlder); 🡺 boolean

16. The Ternary Operator

var firstName = 'John';

var age = 14;

// Ternary operator

age >= 18 ? console.log(firstName + ' drinks beer.') : console.log(firstName + ' drinks juice.');

17. Truthy and Falsy values and equality operators

// falsy values: undefined, null, 0, '', NaN

// truthy values: NOT falsy values

21. Function Statements and Expressions

var whatDoYouDo = function(job, firstName) {

switch(job) {

case 'teacher':

return firstName + ' teaches kids how to code';

case 'driver':

return firstName + ' drives a cab in Lisbon.'

case 'designer':

return firstName + ' designs beautiful websites';

default:

return firstName + ' does something else';

}

}

console.log(whatDoYouDo('teacher', 'John'));

22. Arrays (=== -1 : check if not exists)

var john = ['John', 'Smith', 1990, 'designer', false];

var isDesigner = john.indexOf('designer') **=== -1** ? 'John is NOT a designer' : 'John IS a designer';

console.log(isDesigner);

24. CODING CHALLENGE 3

number \* 20% === number \* . 2

25. Objects and Properties

Two way to initialize an object:

var mohamed = {

firstName: 'Mohamed',

lastName: 'CHIBANI',

birthYear: 1996,

family: ['Am', 'Za', 'Ma', 'Ba'],

job: 'developer',

isMarried : false

}

mohamed.job = 'enginner';

mohamed['isMarried'] = true;

console.log(mohamed);

var amine = new Object();

amine.firstName = 'Amine';

amine.birthYear = 2003;

amine['lastName'] = 'CHIBANI';

console.log(amine);

31-32 : Challenge

var john = {

fullName : 'John Smith',

bills : [124,48,268,180,42],

calcTip: function(){

this.tips = [];

this.finalValues = [];

var percentage;

for(var i = 0; i<this.bills.length; i++){

bill = this.bills[i];

if(bill < 50){

percentage = .2;

}else if(bill >= 50 && bill <= 200){

percentage = .15;

}else if(bill > 200){

percentage = .1;

}

this.tips[i] = bill \* percentage;

this.finalValues[i] = bill + this.tips[i];

}

}

}

var mark = {

fullName : 'Mark Miller',

bills : [77,375,110,45],

calcTip: function(){

this.tips = [];

this.finalValues = [];

var percentage;

for(var i = 0; i<this.bills.length; i++){

bill = this.bills[i];

if(bill < 100){

percentage = .2;

}else if(bill >= 100 && bill <= 300){

percentage = .1;

}else if(bill > 300){

percentage = .25;

}

this.tips[i] = bill \* percentage;

this.finalValues[i] = bill + this.tips[i];

}

}

}

john.calcTip();

mark.calcTip();

function calcAverage(tips){

var sum = 0;

for(var i = 0; i<tips.length; i++){

sum = sum + tips[i];

}

return sum / tips.length;

}

john.average = calcAverage(john.tips);

mark.average = calcAverage(mark.tips);

console.log(john,mark);

if(john.average > mark.average){

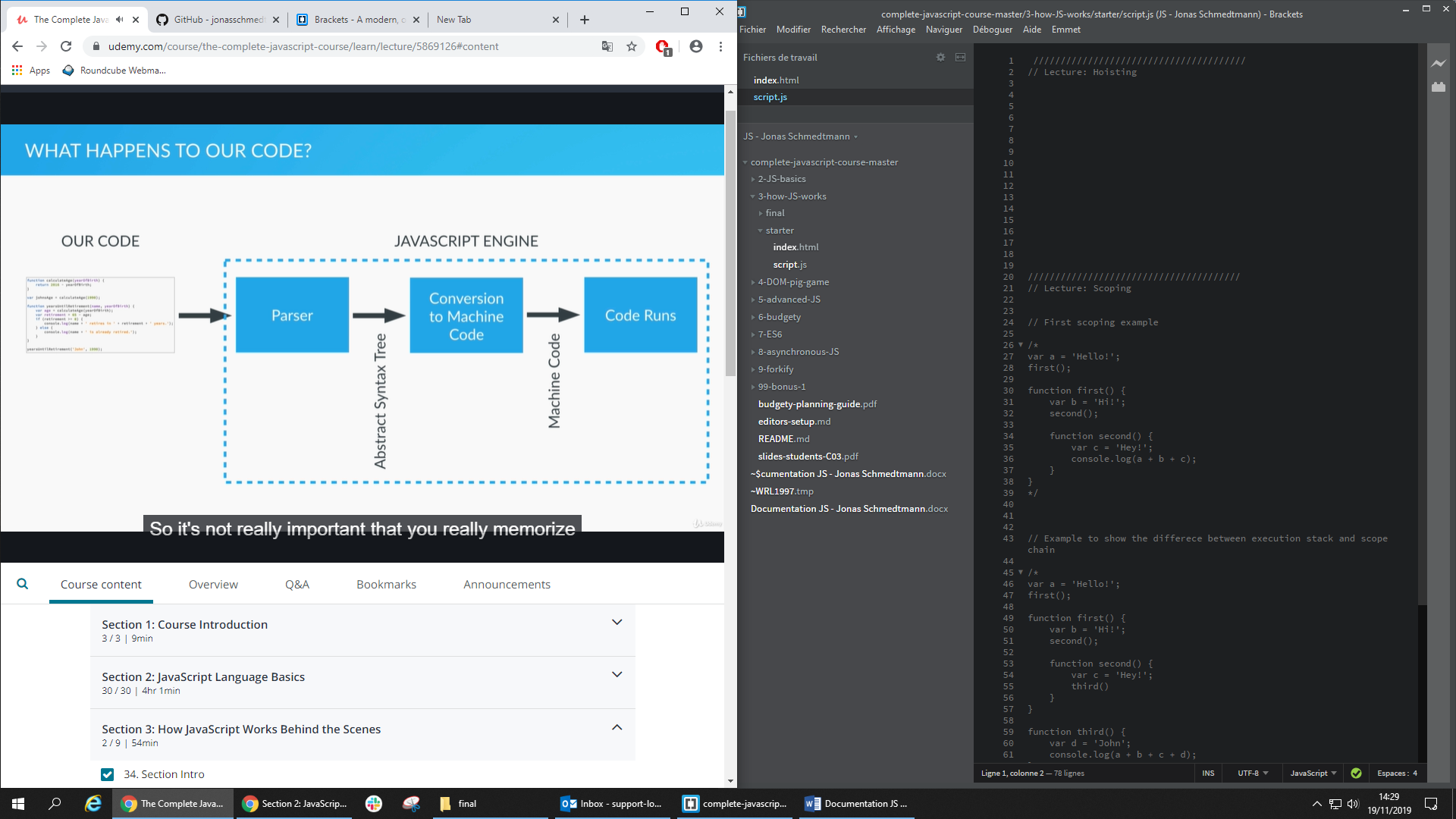
console.log(john.fullName+'s\' family paid the highest tips on average with '+ john.average);

}else{

console.log(mark.fullName+'s\' family paid the highest tips on average with '+ mark.average);

}

Section 3: How JavaScript Works Behind the Scenes



Hoasting only works with function declaration

Hoisting works only with function declarations:

calculateAge(1996);

function calculateAge(year){

console.log(2019 - year);

}

Hoisting doesn’t work with function expression:

retirement(1996);

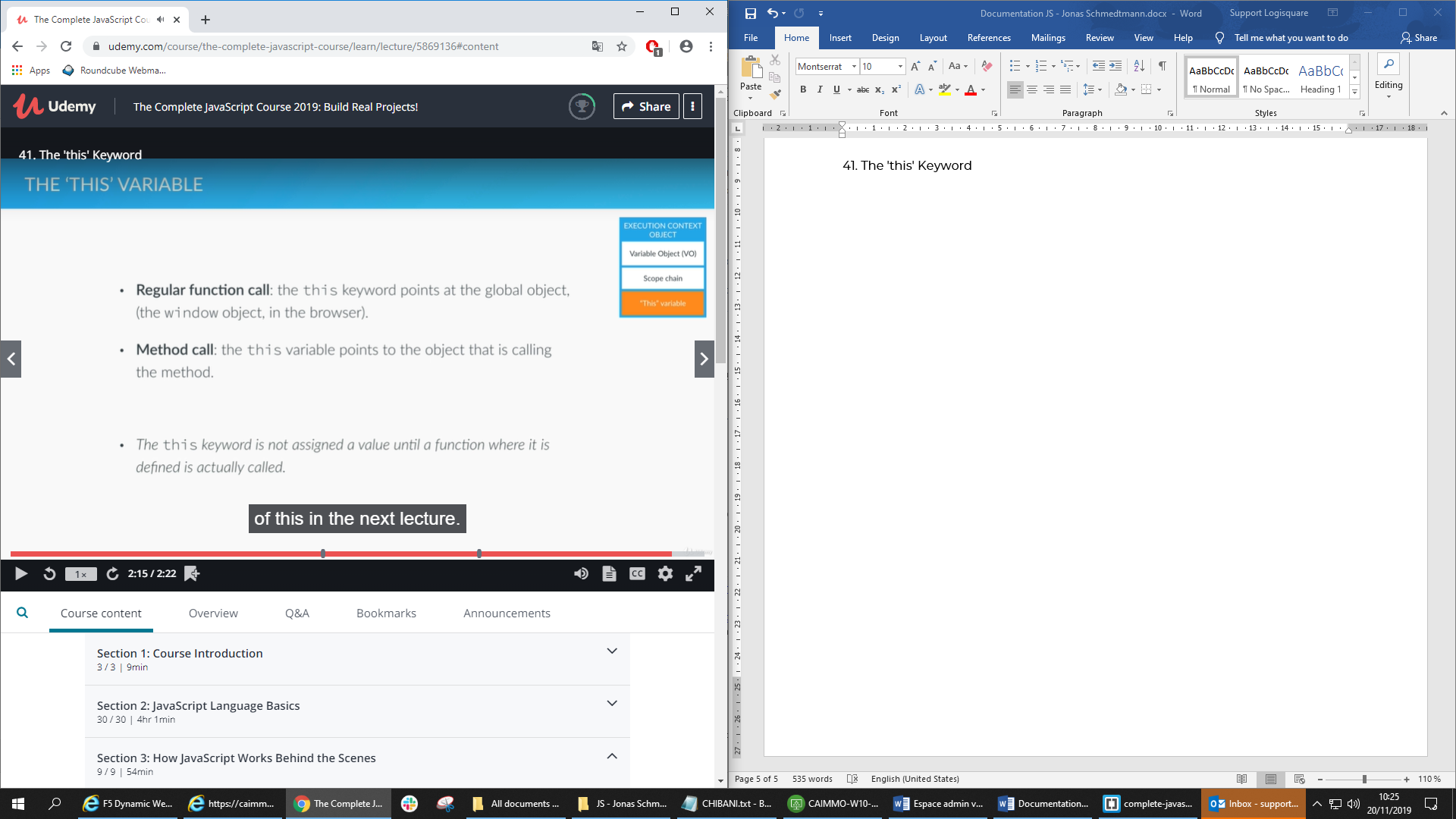
retirement(1996);

var retirement = function(year){

console.log(65-(2019-year));

}

41. The 'this' Keyword



Boths points on the global object window

1) points on the global object window

console.log(this);

2) points on the global object window because a regular function is attached to the window object :

calculateAge(1996);

function calculateAge(year){

console.log(2016-year);

console.log(this);

}

3) this points on the current object john

var john = {

name : 'John',

yearOfBirth : 1995,

calcAge: function(){

console.log(this);

console.log(2019 – this.yearOfBirth);

}

}

john.calcAge();

4) this points on the window object and not the john object, because innerFunction stills a regular function

var john = {

name : 'John',

yearOfBirth : 1995,

calcAge: function(){

console.log(this);

console.log(2019 - this.yearOfBirth);

function innerFunction(){

console.log(this);

}

innerFunction();

}

}

john.calcAge();

5) even if we create a new object, and use the method in other object it , the this keyword still point on the mike object and not the window object

var john = {

name : 'John',

yearOfBirth : 1995,

calcAge: function(){

console.log(this);

console.log(2019 - this.yearOfBirth);

function innerFunction(){

console.log(this);

}

innerFunction();

}

}

john.calcAge();

var mike = {

name : 'John',

yearOfBirth : 1984

}

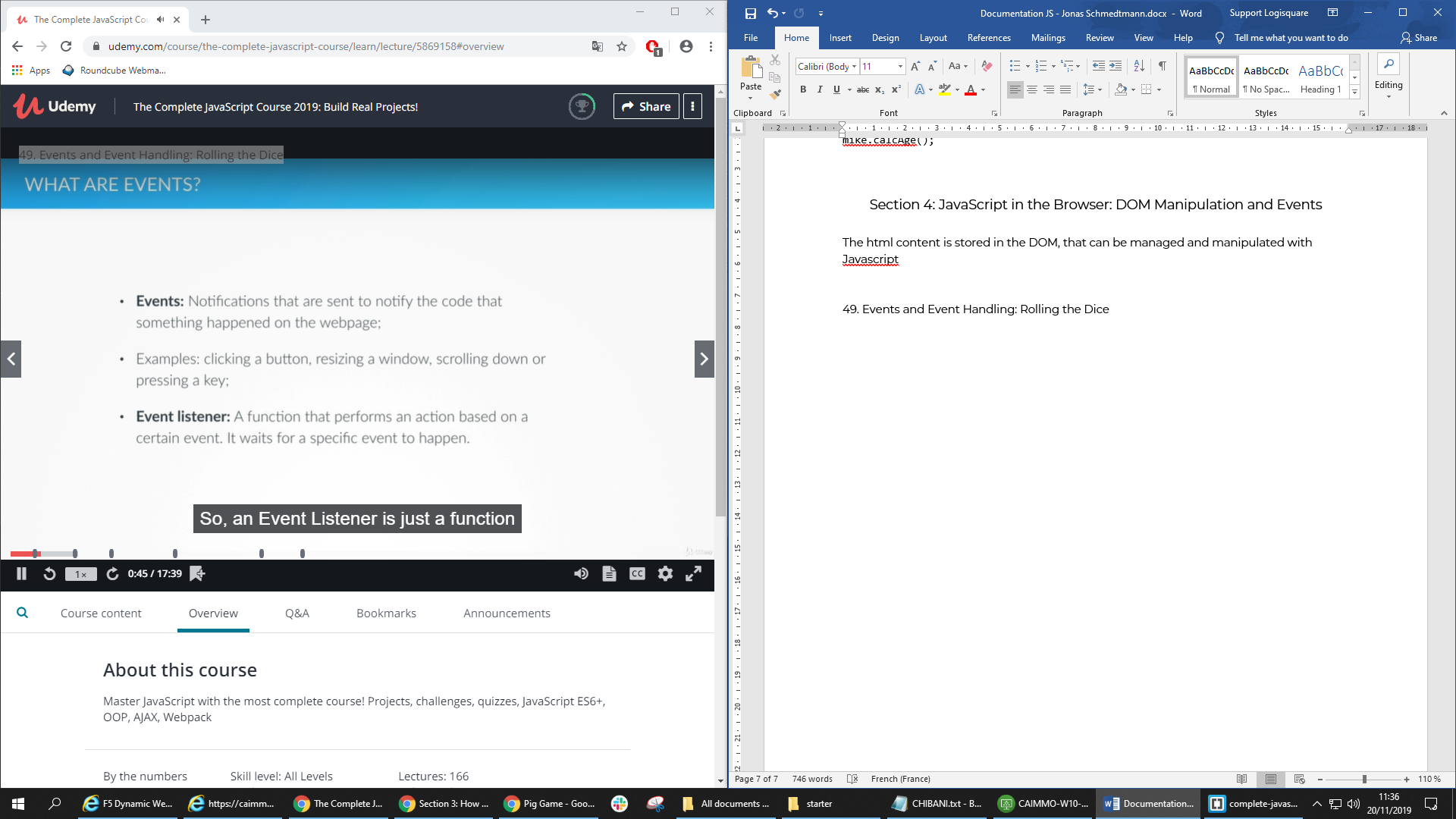
**mike.calcAge = john.calcAge;**

mike.calcAge();

Section 4: JavaScript in the Browser: DOM Manipulation and Events

The html content is stored in the DOM, that can be managed and manipulated with Javascript

49. Events and Event Handling: Rolling the Dice



Anonymous function: is a function that does not have a name, and can’t be reused

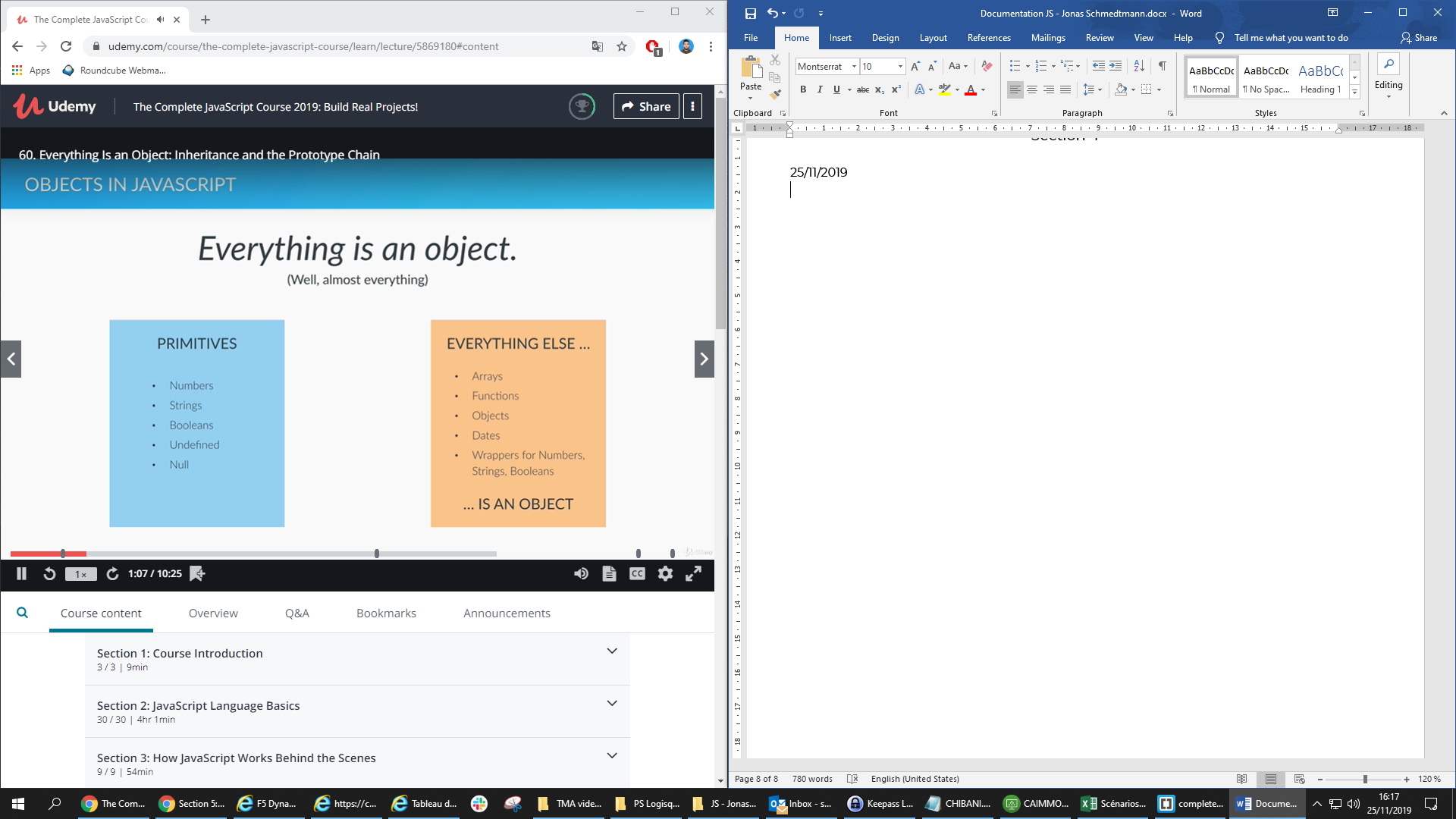
53. Finishing Touches: State Variables

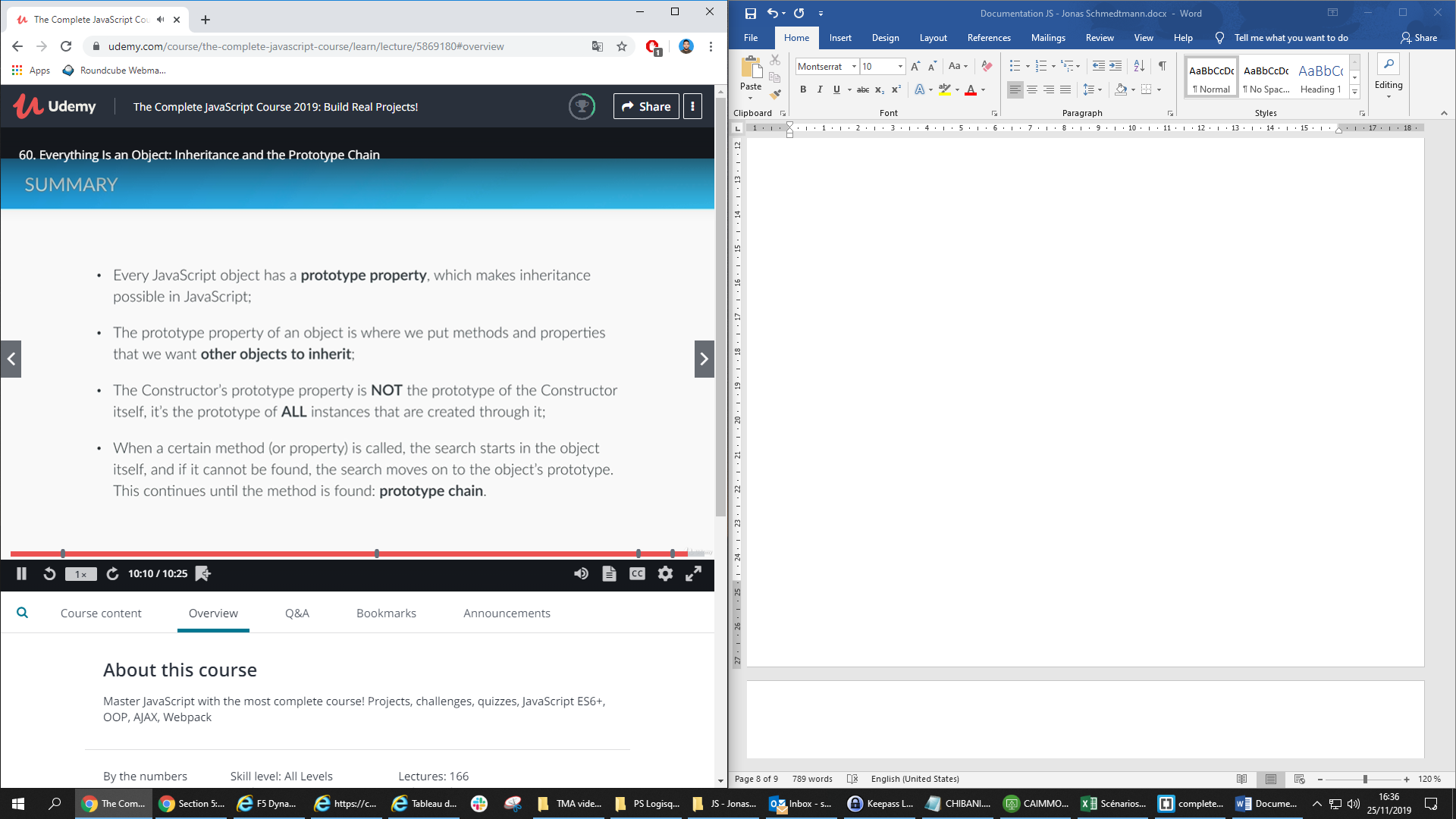
State Variable: store the state of an variable on our system

Section 4

25/11/2019

Everything Is an Object: Inheritance and the Prototype Chain





First method to create object:

var john = {

name : 'John',

yearOfBirth : 1996,

job : 'teacher'

};

var Person = function(name, yearOfBirth, job){

this.name = name;

this.yearOfBirth = yearOfBirth;

this.job = job;

}

Person.prototype.calcAge = function(){

console.log(2019 - this.yearOfBirth);

}

/\*

new :

1 - empty object created

2 - constructor Person is called

3 - create a new execution context

4 - point the john variable to the new function Person

\*/

var john = new Person('John', 1996, 'teacher');

var jane = new Person('Jane', 1969, 'Designer');

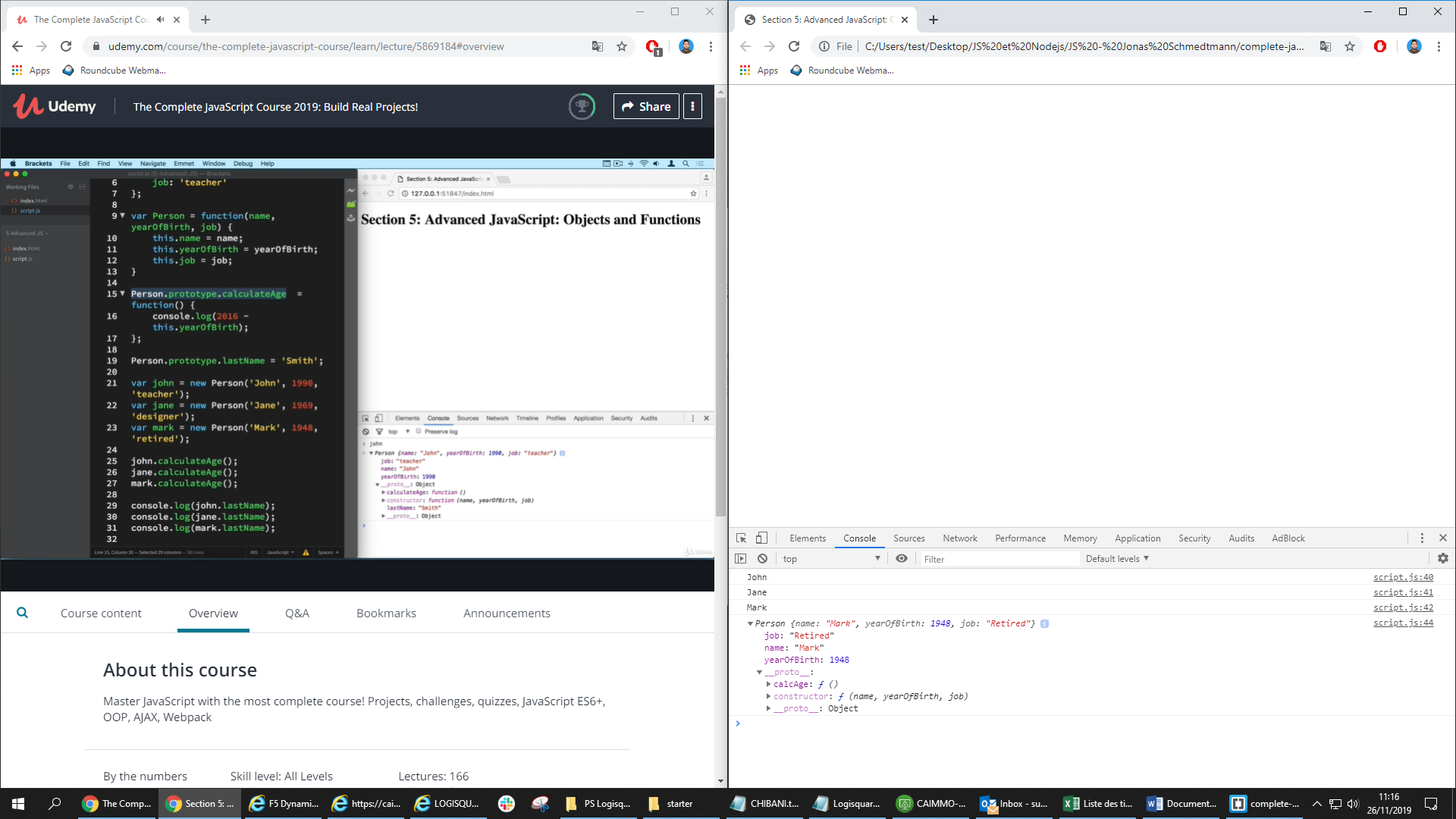
var mark = new Person('Mark', 1948, 'Retired');

john.calcAge();

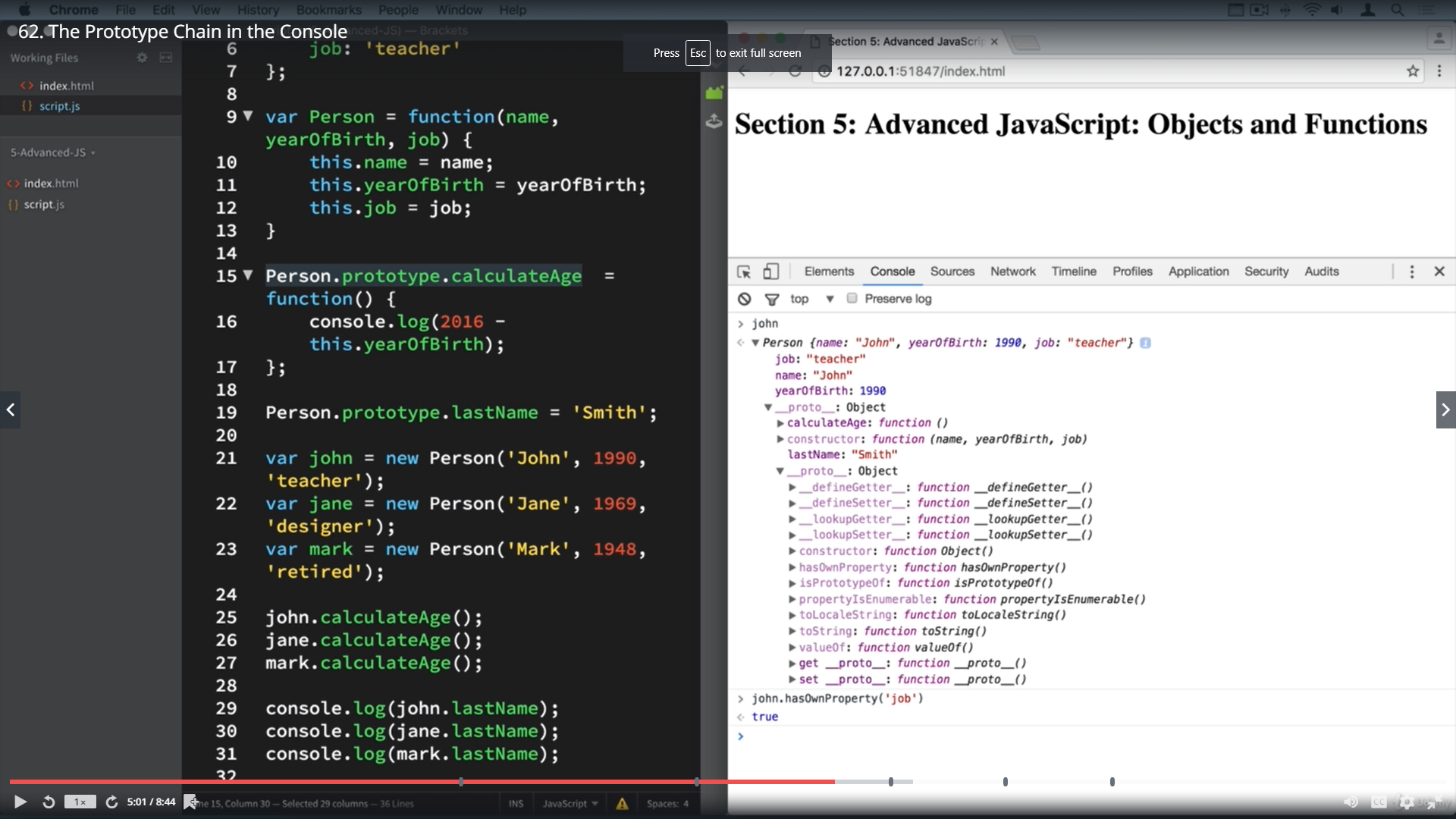
jane.calcAge();

mark.calcAge();

The prototype property of the person constructer



Those methods can be helpful:



Second method to create object:

var personProto = {

calcAge : function(){

console.log(2019 - this.yearOfBirth);

}

}

var john = Object.create(personProto);

john.name = 'John';

john.yearOfBirth = 1990;

john.job = 'teacher';

**65. First Class Functions: Passing Functions as Arguments**

var years = [1990, 1965, 1937, 2005, 1998];

function arrayCalc(arr, fn){

var arrRes = [];

for (var i =0; i < arr.length; i++){

arrRes.push(fn(arr[i]));

}

return arrRes;

}

function calculateAge(el){

return 2019 - el;

}

function isFullAge(el){

return el >= 18;

}

function maxHeartRate(el){

if(el >= 18 && el <= 81){

return Math.round(206.9 - (0.67 \* el));

}

}

var ages = arrayCalc(years, calculateAge);

var fullAges = arrayCalc(ages, isFullAge);

var rates = arrayCalc(ages, maxHeartRate);

console.log(ages);

console.log(rates);

**67. Immediately Invoked Function Expressions (IIFE)**

Use anonymous function to avoid to declare the hold function and call it

(function(){

var score = Math.random() \* 10;

console.log(score >= 5);

})();

//console.log(score);

(function(goodLuck){

var score = Math.random() \* 10;

console.log(score >= 5 - goodLuck);

})(5);

**68. Closures**

function retirement(retirementAge){

    var a = ' years left until retirement';

    return function(yearOfBirth){

        var age = 2019 - yearOfBirth;

        console.log((retirementAge-age) + a);

    }

}

var retirementUS = retirement(66);

var retirementGermany = retirement(65);

var retirementIceland = retirement(67);

var retirementMaroc = retirement(60);

retirementUS(1996);

retirementGermany(1996);

retirementIceland(1996);

retirementMaroc(1996);

Use Closures in this example:

function interviewQuestion(job) {

    if (job === 'designer') {

        return function(name) {

            console.log(name + ', can you please explain what UX design is?');

        }

    } else if (job === 'teacher') {

        return function(name) {

            console.log('What subject do you teach, ' + name + '?');

        }

    } else {

        return function(name) {

            console.log('Hello ' + name + ', what do you do?');

        }

    }

}

With closures

function interviewQuestion(job) {

    return function(name) {

        if (job === 'designer') {

            console.log(name + ', can you please explain what UX design is?');

        } else if (job === 'teacher') {

            console.log('What subject do you teach, ' + name + '?');

        } else {

            console.log('Hello ' + name + ', what do you do?');

        }

    }

}

var interviewQuestionDesigner = interviewQuestion("designer");

var interviewQuestionTeacher = interviewQuestion("teacher");

interviewQuestionDesigner('Ilyass');

interviewQuestionTeacher('Amine');

**69. Bind, Call and Apply**

Call method borrowing: Call (this, parameter1, parameter2, …)

var john = {

    name: 'John',

    age: 26,

    job: 'teacher',

    presentation: function(style, timeOfDay) {

        if (style === 'formal') {

            console.log('Good ' + timeOfDay + ', Ladies and gentlemen! I\'m ' +  this.name + ', I\'m a ' + this.job + ' and I\'m ' + this.age + ' years old.');

        } else if (style === 'friendly') {

            console.log('Hey! What\'s up? I\'m ' +  this.name + ', I\'m a ' + this.job + ' and I\'m ' + this.age + ' years old. Have a nice ' + timeOfDay + '.');

        }

    }

};

var emily = {

    name: 'Emily',

    age : 35,

    job: 'designer'

}

john.presentation('formal', 'morning');

john.presentation.call(emily, 'friendly', 'afternoon');

**Apply** method

// apply method accepts the arguments as an array

john.presentation.apply(emily, ['friendly', 'afternoon']);

The **bind** method doesn’t immediately call the function but it generates a copy of the function, so that we can store it somewhere, it helps to create function with preset argument.

What we does did here is the : **Carrying** is a technique when we create a function based on another function but with some preset parameters

var johnFriendly = john.presentation.bind(john, 'friendly');

johnFriendly('morning');

johnFriendly('night');

var emilyFormal = john.presentation.bind(emily, 'formal');

emilyFormal('afternoon');

Practice the bind method on this example:

// Another cool example

var years = [1990, 1965, 1937, 2005, 1998];

function arrayCalc(arr, fn) {

    var arrRes = [];

    for (var i = 0; i < arr.length; i++) {

        arrRes.push(fn(arr[i]));

    }

    return arrRes;

}

function calculateAge(el) {

    return 2016 - el;

}

function isFullAge(limit, el) {

    return el >= limit;

}

// bind method

var ages = arrayCalc(years, calculateAge);

var fullJapan = arrayCalc(ages, isFullAge.bind(this, 20));

console.log(ages);

console.log(fullJapan);

Challenge code : Quiz App

(function(){ // invoke method

    function Question(question, answers, correct){

        this.question = question;

        this.answers = answers;

        this.correct = correct;

    }

    Question.prototype.displayQuestion = function() { // prototype, inheretance

        console.log(this.question);

        for (var i = 0; i <this.answers.length; i++){

            console.log(i + ': ' +this.answers[i])

        }

    }

    Question.prototype.checkAnswer = function(ans){

        if(ans === this.correct){

            console.log('Correct answer!');

        }else{

            console.log('Wrong answer');

        }

    }

    var q1 = new Question('Is Javascript the coolest programming language in the world?',

                         ['Yes', 'No'],

                         0);

    var q2 = new Question('What the name of this course\'s teacher?', ['John', 'Micheal', 'Jonas'], 2);

    var q3 = new Question('What does best describe coding', ['Boring','Hard','Fun','Tediuos'], 2);

    var questions = [q1,q2,q3];

    var n = Math.floor(Math.random() \* questions.length);

    questions[n].displayQuestion();

    var answer = parseInt(prompt('Please select the correct answer.'));

    questions[n].checkAnswer(answer);

})();

Expert Challenge

(function(){ // invoke method

    function Question(question, answers, correct){

        this.question = question;

        this.answers = answers;

        this.correct = correct;

    }

    Question.prototype.displayQuestion = function() { // prototype, inheretance

        console.log(this.question);

        for (var i = 0; i <this.answers.length; i++){

            console.log(i + ': ' +this.answers[i])

        }

    }

    Question.prototype.checkAnswer = function(ans, callback){

        var sc;

        if(ans === this.correct){

            console.log('Correct answer!');

            sc = callback(true);

        }else{

            console.log('Wrong answer');

            sc = callback(false);

        }

        this.displayScore(sc);

    }

    Question.prototype.displayScore = function(score){

        console.log('Your current score is: ' + score);

        console.log('-----------------------------------------');

    }

    var q1 = new Question('Is Javascript the coolest programming language in the world?',

                         ['Yes', 'No'],

                         0);

    var q2 = new Question('What the name of this course\'s teacher?', ['John', 'Micheal', 'Jonas'], 2);

    var q3 = new Question('What does best describe coding', ['Boring','Hard','Fun','Tediuos'], 2);

    var questions = [q1,q2,q3];

    function score(){

        var sc = 0;

        return function(correct){

            if(correct){

                sc++;

            }

            return sc;

        }

    }

    var keepScore = score();

    function nextQuestion(){

        var n = Math.floor(Math.random() \* questions.length);

        questions[n].displayQuestion();

        var answer = prompt('Please select the correct answer.');

        if(answer !== 'exit'){

            questions[n].checkAnswer(parseInt(answer), keepScore);

            nextQuestion();

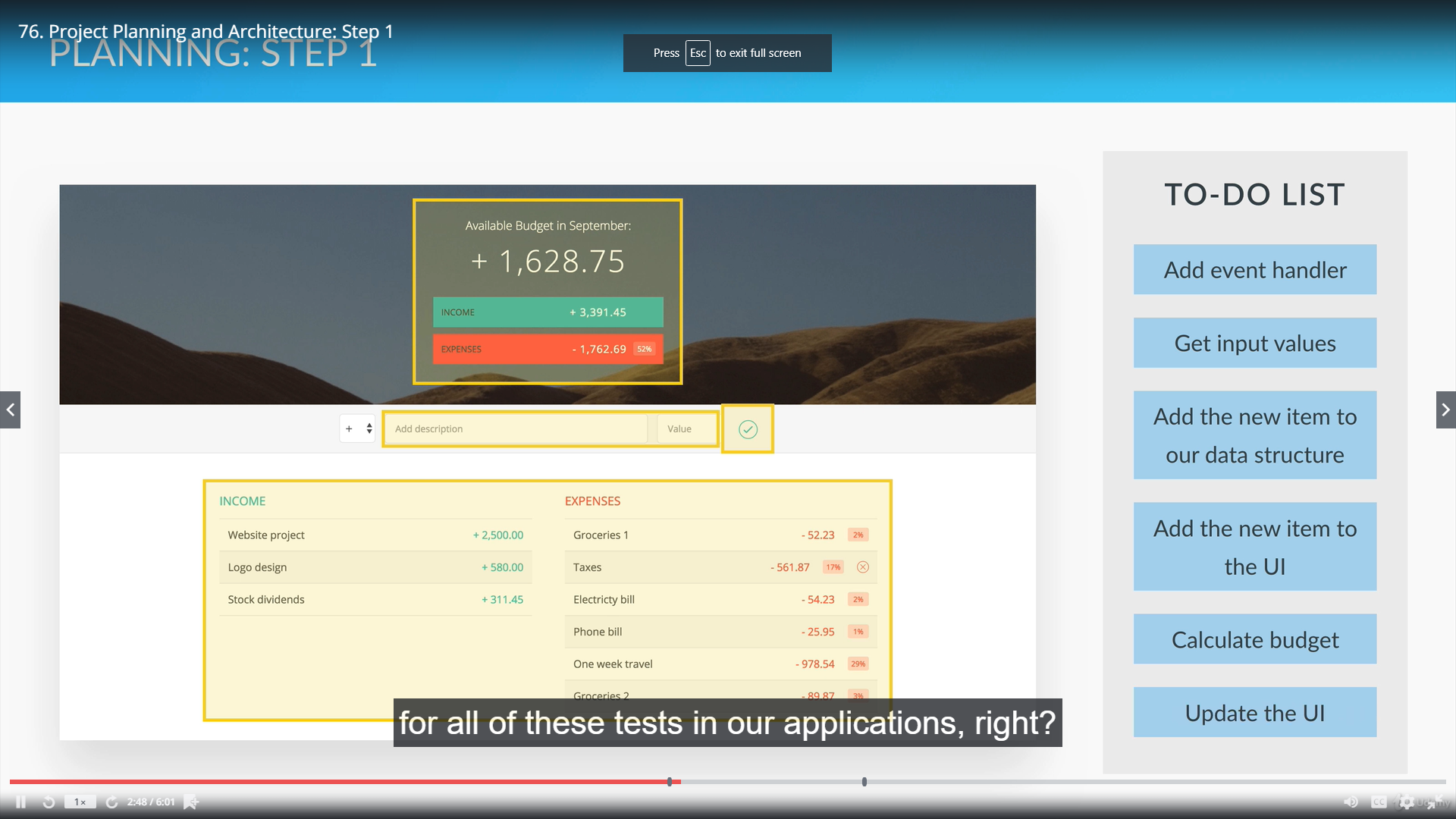
        }

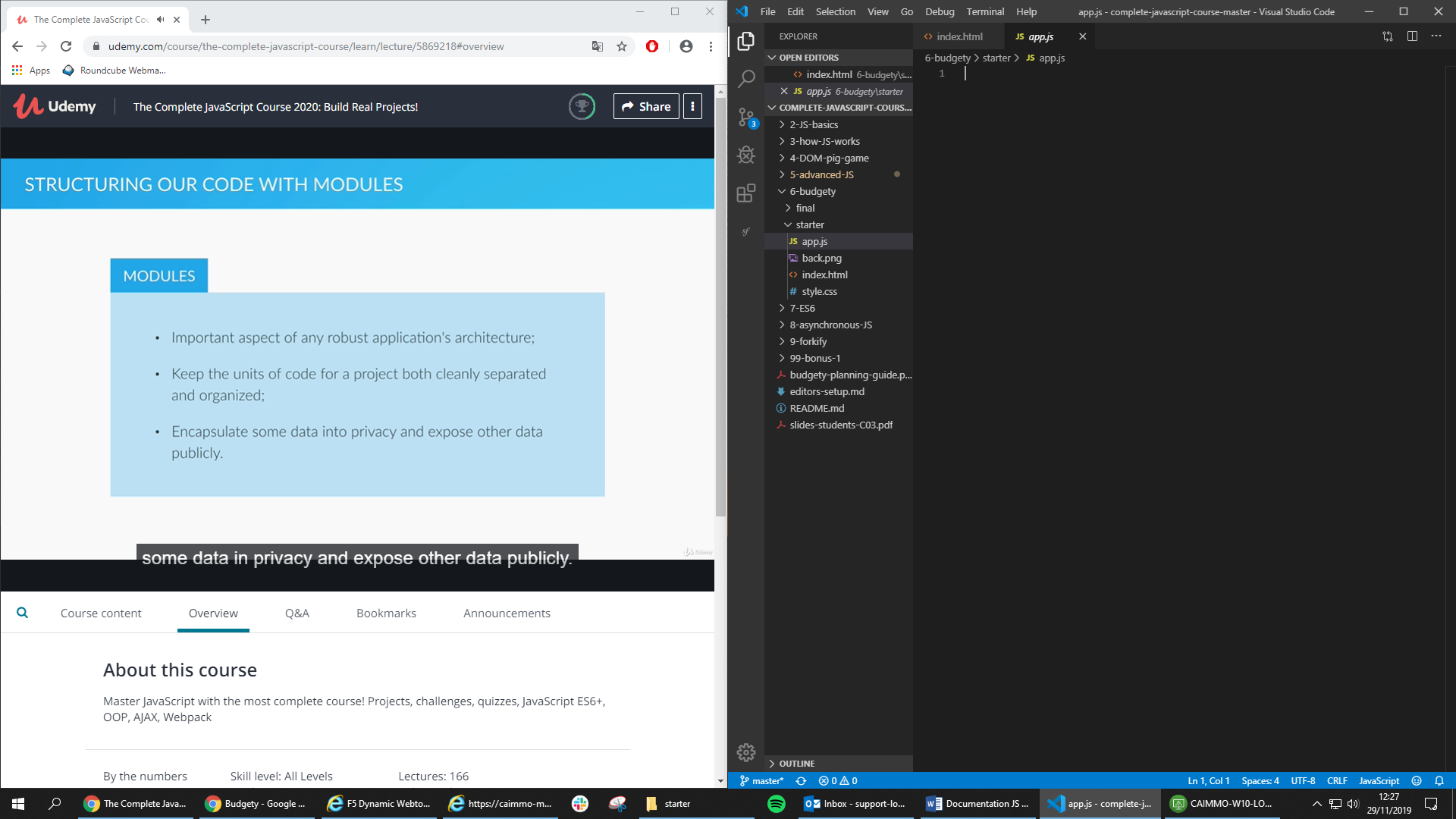
    }

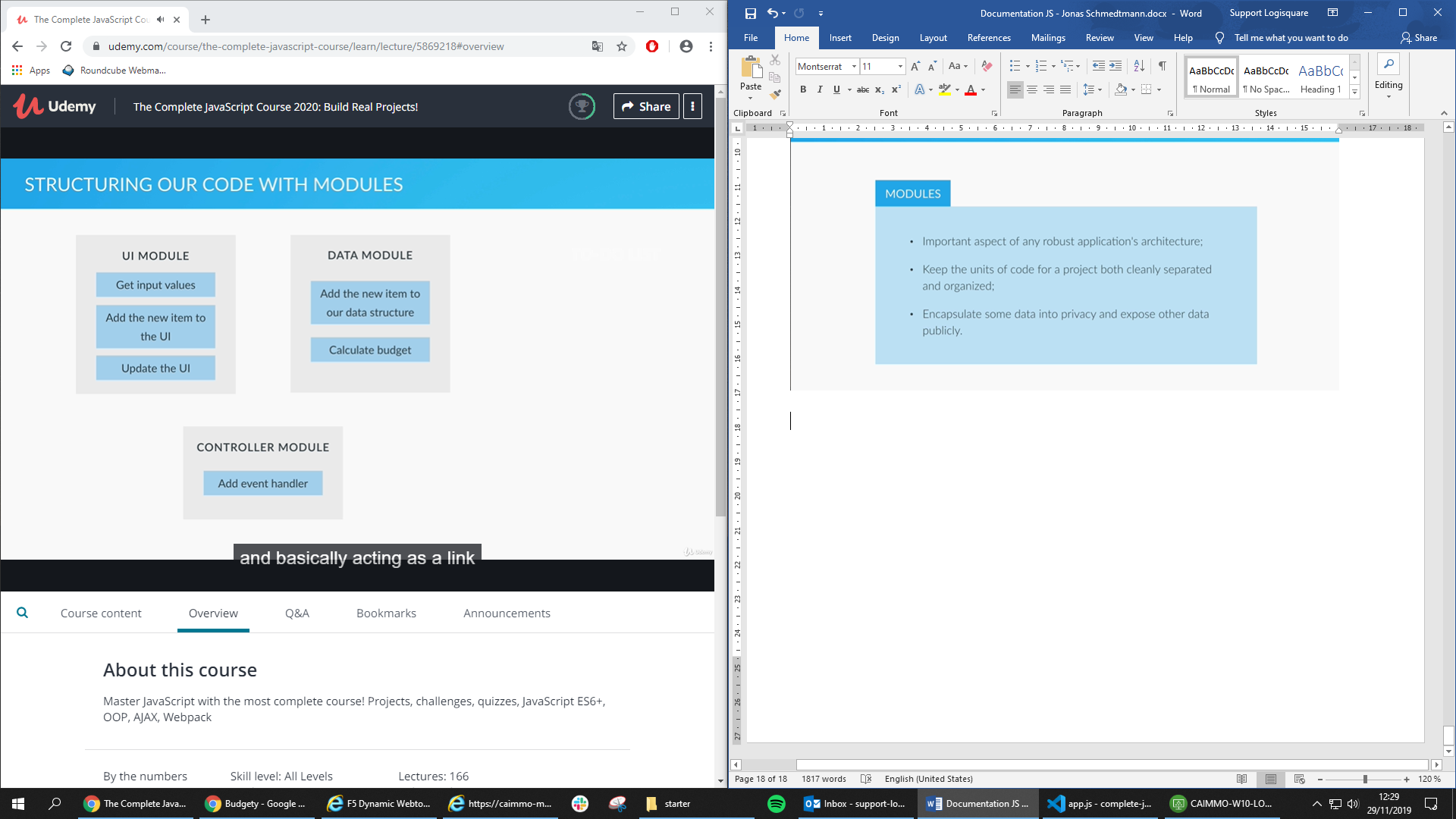
    nextQuestion();

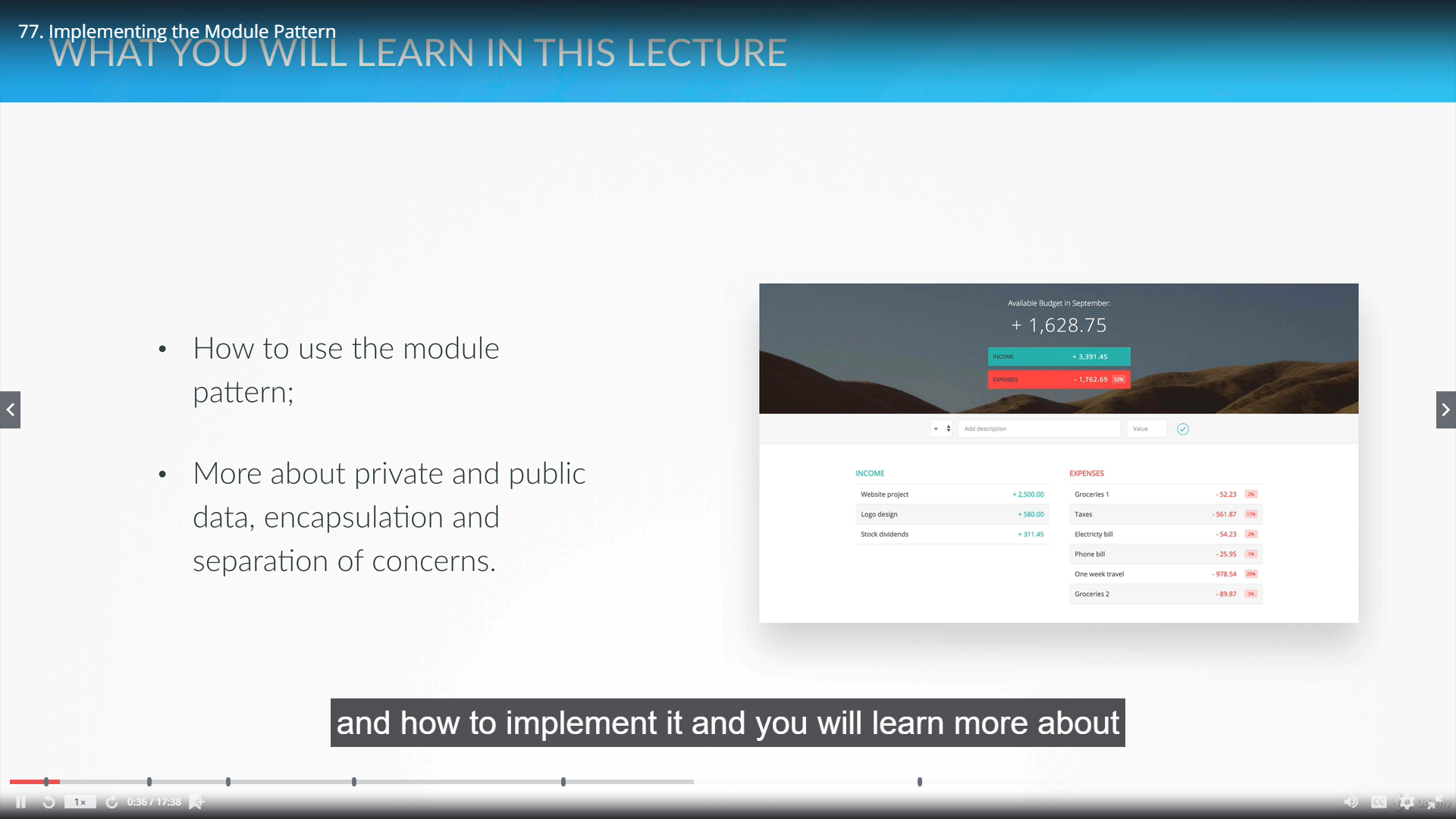
})();

PLANNING STEP 1 :









77. Implementing the Module Pattern

var budgetController = (function(){

    var x = 23;

    var add = function(a){

        return x + a;

    }

    return {

        publicTest: function(b){

            return add(b);

        }

    }

})();

var UIController = (function(){

    // some code

})();

var controller = (function(budgetCtrl, UICtrl){

    var z = budgetCtrl.publicTest(5);

    return {

        anotherPublic: function(){

            console.log(z);

        }

    }

})(budgetController, UIController);

At the console, we print controller.anotherPublic()

