**1 KISS**

**Hello. My name is Igor, and today I will talk to you about K.I.S.S.**

**No, you got it all wrong. We will study the principle of software development, its advantages, why design principles are useful to us, and how to implement them in our daily programming. We will explore KISS software development principle.**

**2 First mention**

**KISS, an acronym for keep it simple, stupid, is a design principle noted by the U.S. Navy in 1960.**

**The acronym was reportedly coined by Kelly Johnson, lead engineer at the Lockheed Skunk Works.**

**The principle is best exemplified by the story of фJohnson handing a team of design engineers a handful of tools, with the challenge that the jet aircraft they were designing must be repairable by an average mechanic in the field under combat conditions with only these tools.**

**3 Descriptive**

**The KISS principle is descriptive to keep the code simple and clear, making it easy to understand.**

**After all, programming languages are for humans to understand "computers can only understand 0 and 1" so keep coding simple and straightforward.**

**4 Time to get smarter**

**As our everyday coding includes javascript a lot, things tend to get a bit messy quite quickly.**

**Adding more code to the codebase means more mess added. And the end story is a hell no one want to witness, if the coding pattern was not clean and well planned from the beginning.**

**But, maintaining a few simple patterns can make the whole experience a lot better. Let`s see some examples**

**5 Short-circuit Evaluation**

**First it`s Short-circuit Evaluation. Look at this. The if statement executes a statement if a specified condition is truthy.**

**If the condition is falsy, another statement can be executed.**

**The code above is pretty common, let`s do it smartly**

**We can use short-circuit evaluation in many cases like**

**This will return the persons firstName if the person is not invalid (undefined, null, etc).**

**6 Arrow Functions**

**Ok, go to the next slide. Arrow functions are undoubtedly one of the more popular features of ES6.**

**This was a trivial function, which can be replaced with arrow function.**

**How clean it is!**

**WARNING: function and arrow functions have different scoping methodology. So, use them carefully.**

**7 Avoid boring old for loop**

**I think I don`t have to describe, code speaks for itself**

**what will happen next**

**8 Implicit Return**

**It is Implicit Return**

**In this example, we will consider Implicit Return**

**When using arrow functions, we can return implicitly if the function has a single line/statement**

**without using the return keyword.**

**We could return more complex values implicitly like**

**The code above returns an Object. Notice the braces () before curly braces {}.**

**9 Default Parameters**

**Next slide about default functions parameters**

**Default function parameters allow named parameters to be initialized with default values if no value or undefined is passed.**

**10 Destructuring**

**What can we say about destruction?**

**.The repetitive assignment work is boring, so let`s make it less-boring**

**Now, we can also set default values here if the property doesn`t exist**

**We can also rename some property if need to,**

**Now, the last name will be assigned to surname**

**11 Spread Operator**

**What we know about spread operator   
this operator also came to us from ES6**

**Opetator spread allows you to write beautifully and simply code**

**This also works with Object types**

**We could use spread operator anywhere (beginning, end, middle)**

**12 Importing Modules**

**Next slide about import in js. Most of the time we don`t need the whole library to work with. We need only a few things most of the time.**

**We can easily do that using partial imports which uses destructuring.**

**13 Async/Await over Promises**

**now we will talk about promises**

**This is quite an important topic to discuss and can be discussed for some time. In javascript we use callbacks here and there and also promises. Things can get pretty nasty if we use multiple nested promises. Even a single promise can lose readability pretty soon.**

**If we transform this promise based code to async/await code block, it`ll look like below**

**Notice the indentation thing. This makes the code more readable, easily debuggable.**

**14 Map, Filter, Reduce**

**And so. It's time for a big** example **Let`s imagine a scenario. We need to calculate the total distance in miles of all items in our distances array where distance is less than 10000 km If we use our traditional for-loop**

**The code works! Now, let`s try it with map-filter-reduce**

**See, how readable and clean the code is!**

**Some of these practices may be debatable, but that depends on the person.**

**Now, let's summarize.How we can apply the KISS principle to our work**

**1)Be Humble, don't think of yourself as a super genius, this is your first mistake**

**2)Break down your tasks into sub tasks that you think should take no longer than 4-12 hours to code**

**3)Break down your problems into many small problems. Each problem should be able to be solved within one or a very few classes**

**4)Keep your methods small, each method should never be more than 30-40 lines. Each method should only solve one little problem, not many uses cases**

**If you have a lot of conditions in your method, break these out into smaller methods.**

**Not only will this be easier to read and maintain, but you will find bugs a lot faster.**

**You will learn to love Right Click+Refactor in your editor.**

**5)Don't be afraid to throw away code. Refactoring and recoding are two very important areas. As you come across requirements that didn't exist, or you weren't aware of when you wrote the code to begin with you might be able to solve the old and the new problems with an even better solution.**