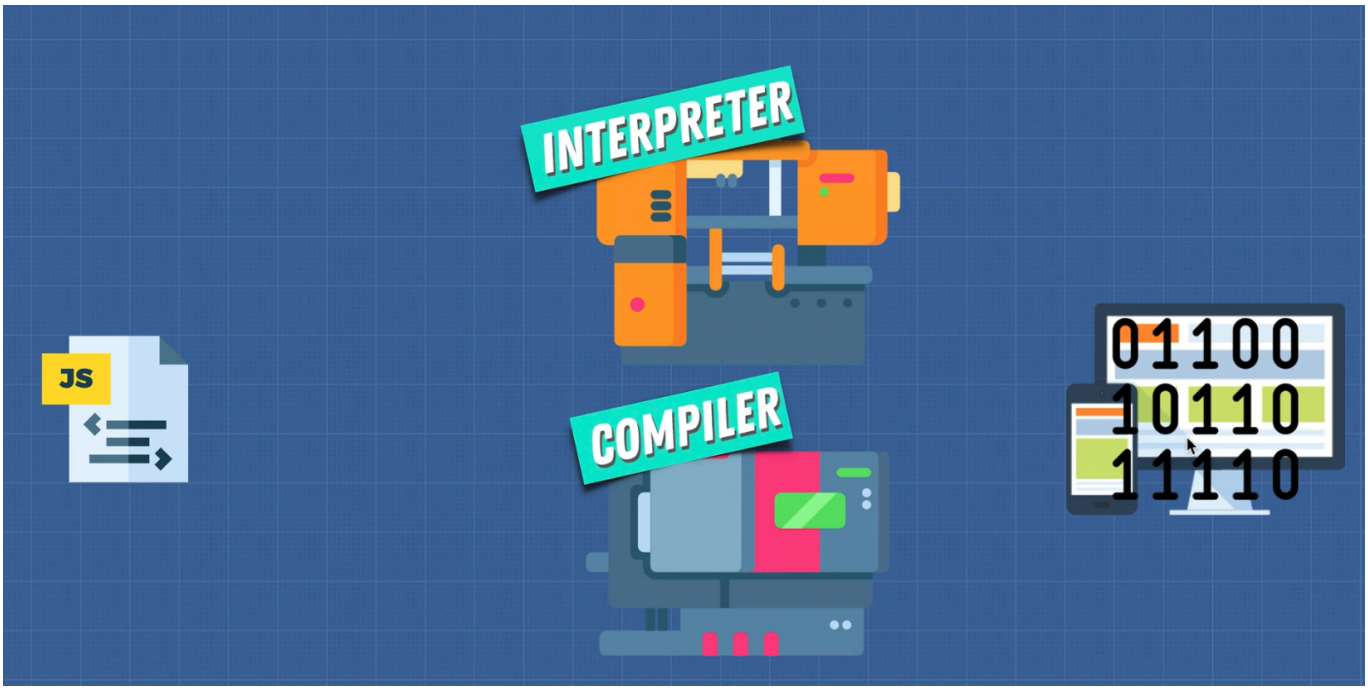
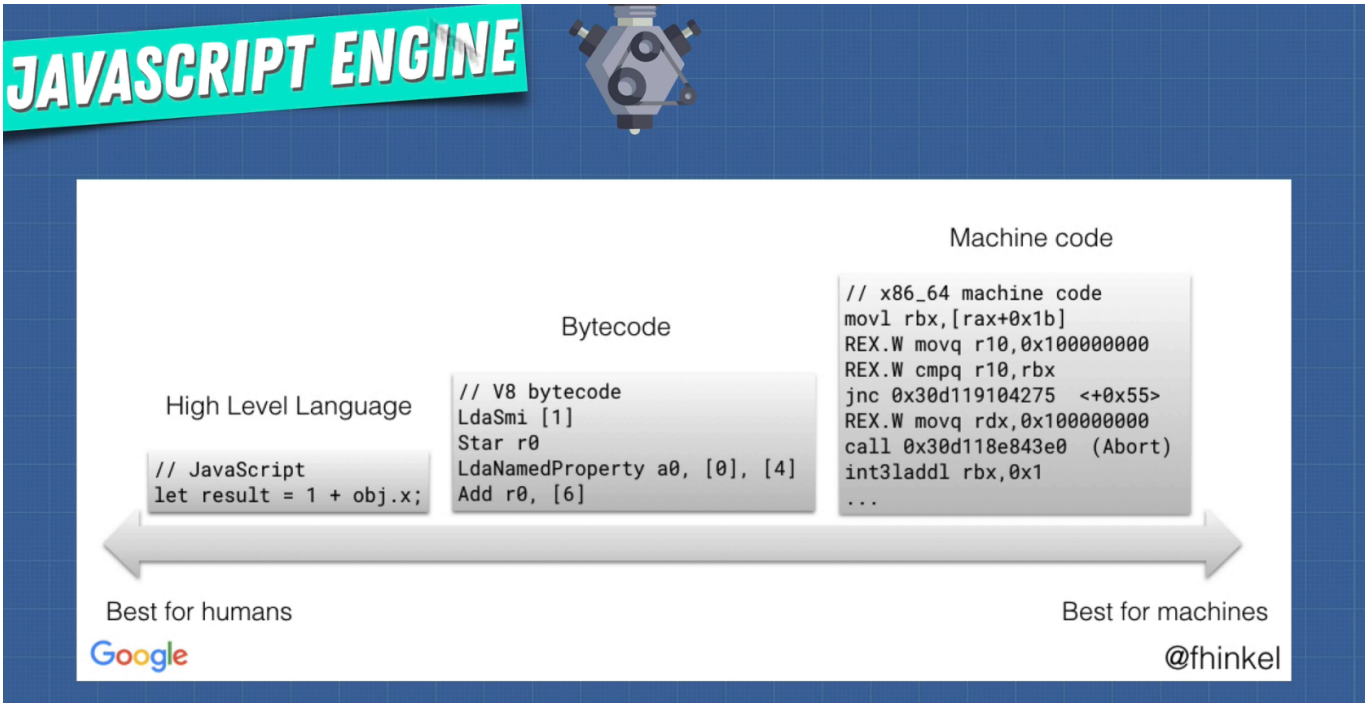
Introduction to JS

JavaScript has been introduced since 1995. JS is not Java different like man and mango, car and carpet. As sound javascript, it is script language. It can be called the client-side language. Because in the early age, js can only be used on the browser. That's why they are one of the languages in the group of interpreter language.





Intrepreter are quick to read. The client doesn't need to wait longer. So being able to run as fast as possible is ideal in their beginning era. As an example- Google had back a lot problems when they had google maps running a lot of JS, it will get slower and slower. The problem with js is that if you are running the same code more than once, like in a loop over and over even though it gives same results. It will get slower.

Variables

A typical modern computer has more than 30 billion bits in its volatile data storage(working memory). We can define the variable in JS with var, let and const keywords.

```
var hello = "hello world"
var number = 1
let number = 2
const PI = 3.142
```

Above let keywords is block scope variable as they can't be used outside of the scope. I will explain it later in details.

In Defining the variable must care about the name.

Rule 1

You can use whatever you like means a b c capital or small But the variables name must be understandable and different from others because we might use it again and again in our code.

Rule 2

Numbers can be included in variable like num1, num2 but don't ever start with number!

Rule 3

Whatever you build file, it is a good habit not to use space. In defining variable it is the same thing. Don't ever use space in defining variables.

Rule 4

Instead of space, you can use dash or underscore like your-name, color_design.

Rule 5

Don't ever use special characters like + - & # @. But one exception we can use $ sign in defining variables and in most language especially in PHP, variable must be defined with $ sign.

Data Types

Most Programming languages have four basic data types(primitive) - character, number, boolean and string. In js, as a loosely typed language, we can normally note it has three basic types - number, boolean, string. Unlike other languages, JS include interger and float under number types. It is 64-bit float type. As an example of loosely type language.

```
console.log("Aardvark" < "Zoroaster")
// → true
console.log("Itchy" != "Scratchy")
// → true
console.log("Apple" == "Orange")
// → false
"ABC" > 123
// → false
```

Special Data types

Null and undefined.They are themselves can be called values, but they carry no information.

Many operations in the language that don't produce a meaningful value (you'll see some later) yield undefined simply because they have to yield some value. The difference in meaning between undefined and null is an accident of JavaScript's design, and it doesn't matter most of the time. In cases where you actually have to concern yourself with these values, I recommend treating them as mostly interchangeable.

Operators

Operators are plus +, minus -, multiply *, divide / , remainder %. Other than that there are some logical operators (as we learnt logic gates at high school).

```
1 + 1  //2
'a' + 'b'  //ab
1 + 'a'  //1a
'a' + 1  //a1
5 == "5"  //true
5 === "5"   //false
```

There are some operators like square

```
2 ** 2  //4
2 ** 3  //8
```

The weird operator-

```
let a = 2
a++ //2
a  //3
```

Above code - we are adding one to 2 but if you use a++ first, it will only show initial value first.

Logical Operators

AND OR NOTAND NOTOR are mostly used logical operators.

In AND operators, both the conditions must be true, if not it will return false

```
let num = 3
(num > 3) && (num < 5)  // false && true → false
(num < 5) && (num > 3)  // false && false → false
(num < 5) && (num == 3) // true && true → true
```

In OR operators, if one of the conditions is ture, it will return the true.

```
let num = 3
(num > 3) || (num < 5)  // false || true → true
(num < 5) || (num > 3)  // true || false → true
(num < 5) || (num == 3) // true || true → true
```

In NAND

```
let num = 3
!(num > 3 && num < 5)  // false && true → true
!(num < 5 && num > 3)  // false && false → true
!(num < 5 && num == 3) // true && true → false
```