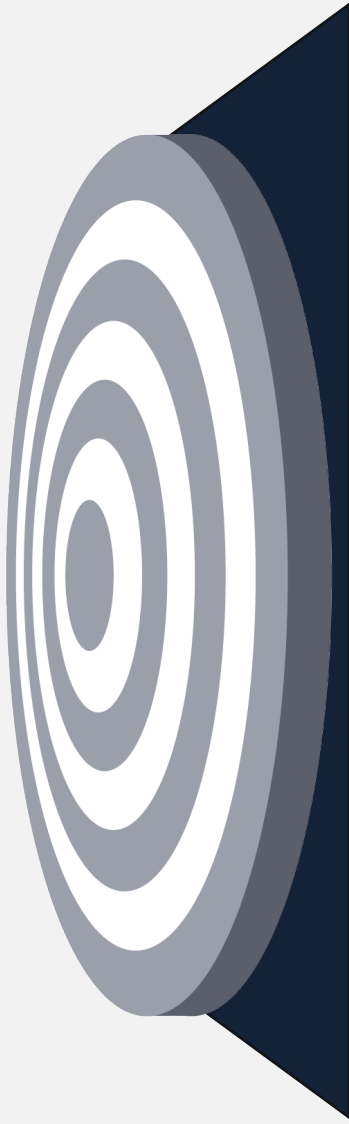
Introduction to JavaScript

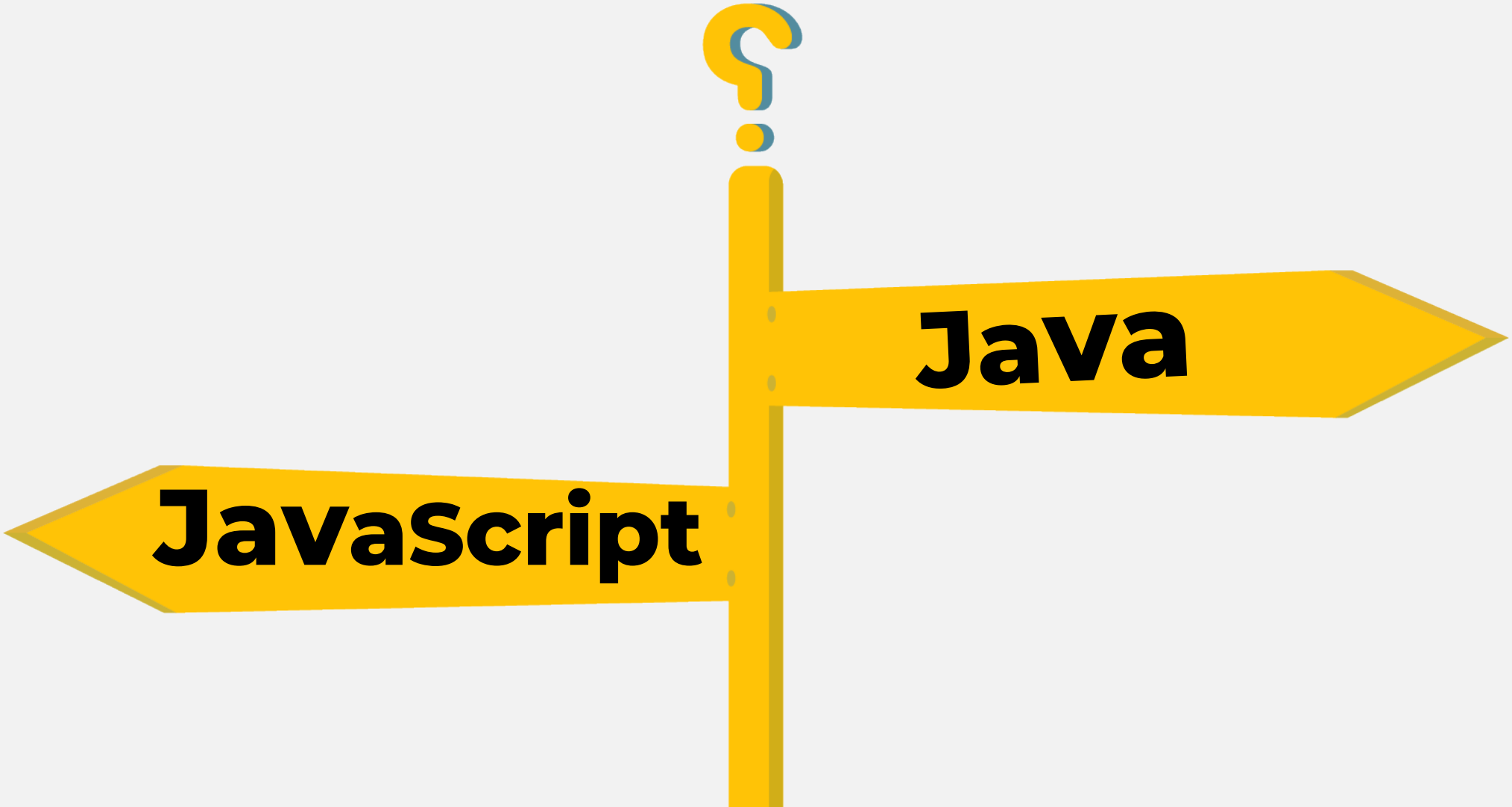
Lecture No. W01D02

GOALS

1. The definition of JavaScript.
2. Knowing the basic JavaScript value types.
3. Using the console to create JS expressions.



Difference Between



Difference Between

- Server and Client Side Language.
- Used for both UI and Core business logic.

JavaScript

Java

- Server Side Language.
- Primarily used for Core business logic.

What is JavaScript?

JavaScript is mainly a client-side scripting or programming language that is **used to implement dynamic features for the web**, it can be run on most browsers and especially the mainstream browsers such as Google Chrome, Firefox, Internet Explorer and Safari.

How to run JavaScript Code?

On Google Chrome you can open the console by right-clicking the page then choosing inspect or by opening the Console tab.

You can also use the following shortcuts if you are on:

- **Windows** Ctrl + Shift + I or F12 .
- **MacOS** you can use Option + Command + i or F12.

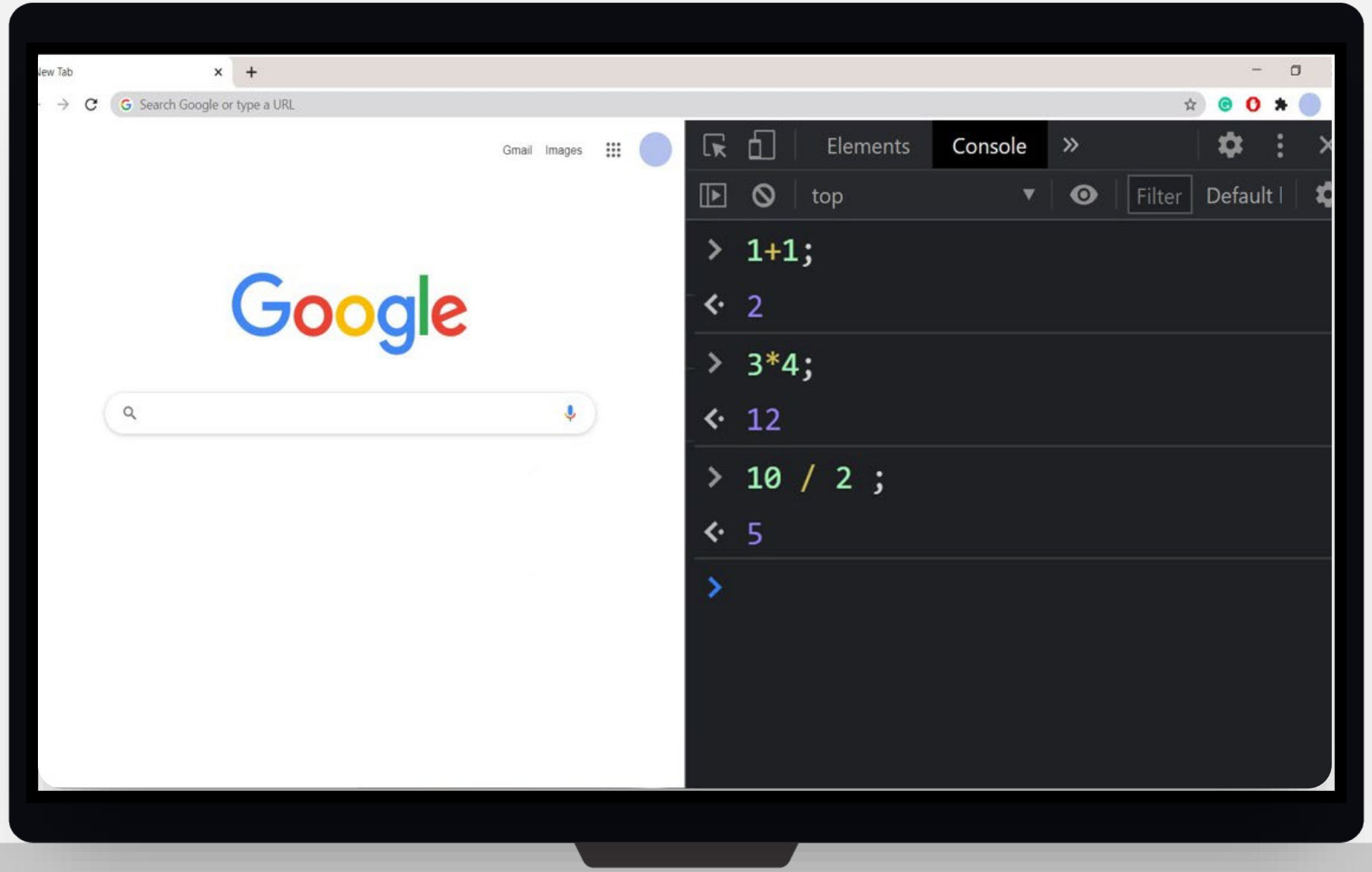
What is **Syntax?**

In computer science, the syntax of a computer language is the **set of rules** that defines the combinations of symbols that are considered to be correctly structured statements or expressions in that language.

Try the following using the Console:

Using mathematical symbols it is possible to form a JavaScript expression

```
1 + 1;
3 * 4;
10 / 2;
```


New Tab
Search Google or type a URL
Gmail
Images
Google
Elements
Console
top
Filter
Default
1+1;
2
3*4;
12
10 / 2 ;
5

Basic Value Types

Let's take a look on some of the value types in JavaScript

1. Strings

- A JavaScript string stores a series of characters
- To create a string wrap the characters with:

Double quotes “ ” or Single quotes ‘ ’

“This is an example of what a string looks like”

'Another example'

To create an expression to **combine** strings together use the `+` operator, known as string concatenation.

```
> "Hello " + "World" ;
```

```
<• "Hello World"
```

2. Numbers

- 120, 300, -56, 0, 1.3, 0.8 …
- To perform mathematical operations in JS combine numbers and operators **(+, -, *, /, %)** in an expression.
- Also in JS these operations will follow the Mathematical order of operations (PEMDAS).

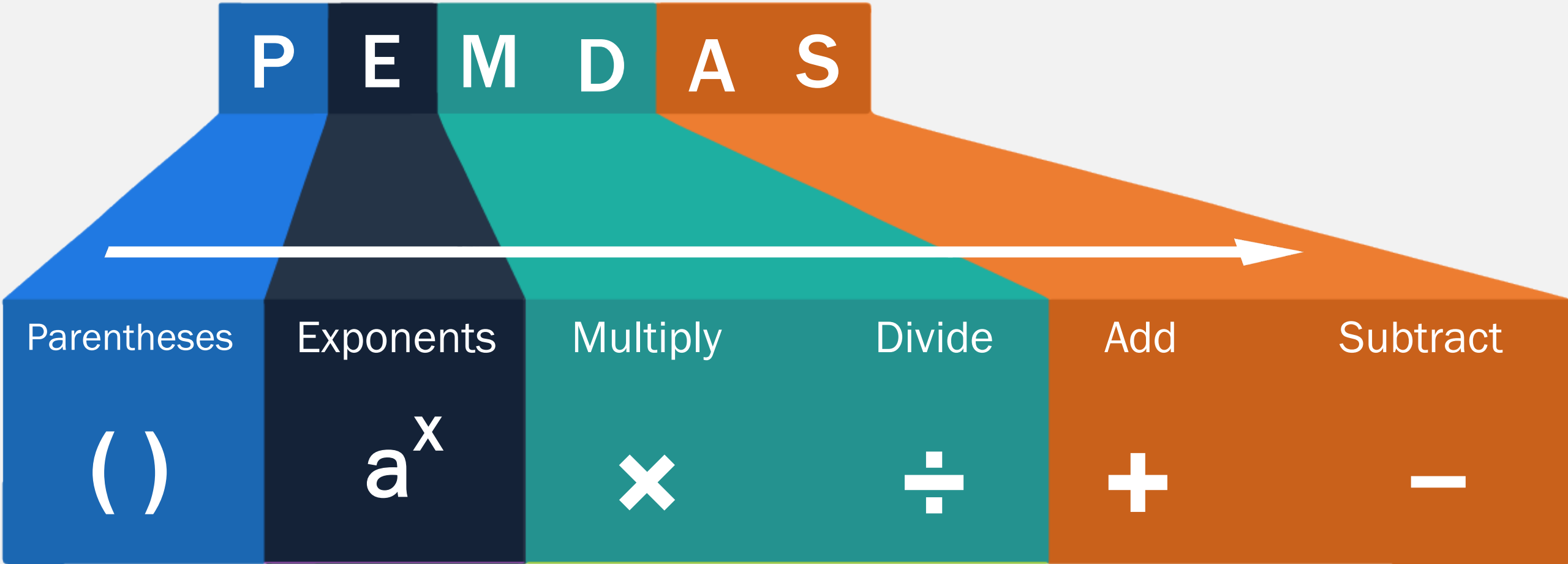
```
> 100 + ((10 - 5) * 2) / 5;
```

```
<• 102
```

```
> 10 % 2;
```

```
<• 0
```

Order Of Operations



Mixing Types

Since JavaScript isn't a strictly-typed language it is possible to mix different values of different types together.

Try the following

An example of concatenating a string with a number:

```
> "I am " + 10 + " years old";
```

```
<• "I am 10 years old"
```

In other strictly-typed languages it is not possible to add integers and float numbers together:

```
> 10 + 5.5;
```

```
<• 15.5
```

An example of using arithmetic operators between numeric string and numeric values:

```
> "10" + 1;          <• "101"
> "10" - 5;          <• 5
> "10" / 5;          <• 2
> "10" * 5;          <• 50
> "10" % 2;          <• 0
```

Mixing types makes the language a little bit **beginner friendly** since errors will not be thrown when trying to mix types together which is a mistake usually done by beginners,

but that doesn't mean that it doesn't have it's down sides since when mixing types you might encounter unexpected behavior.

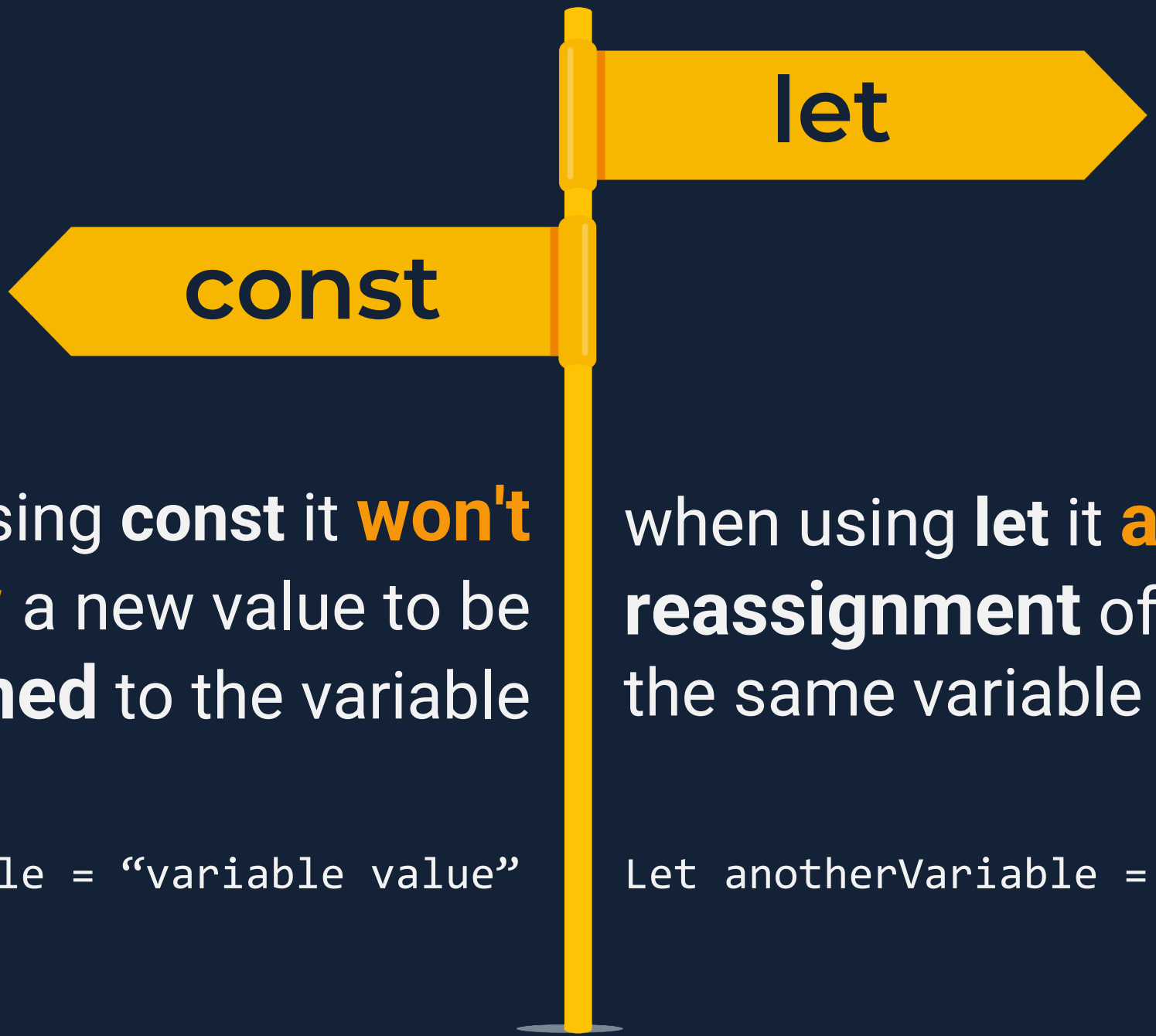
Variables

- Variables are **containers** used for storing values, they are **similar** to variables in mathematics.
- There are **three main ways** to declare a variable in JavaScript using three different keywords:
  - var
  - let
  - const

Defining A Variable And Assignment

The basic syntax for defining a variable:

The Difference Between const and let



When using **const** it **won't allow** a new value to be **reassigned** to the variable

```
const ourVariable = “variable value”
```

when using **let** it **allows** the **reassignment** of values for the same variable

```
Let anotherVariable = 10
```

The Difference Between

const

When using **const** it **won't allow** a new value to be **reassigned** to the variable

```
Const ourVariable= “variable value”;
```

let

when using **let** it **allows** the **reassignment** of values for the same variable

```
let anotherVariable=10;
```

Reassign a Value

In order to reassign A value, you can just use the following syntax:

- Variable declare by const then reassign a value:

```
> const ourVariable = “variable value”
  ourVariable = 10;
```

```
Uncaught TypeError: Assignment to constant variable.
```

- Variable declare by let then reassign a value:

```
> let anotherVariable = 10;
  anotherVariable = 20;
```

```
<• 20
```

You can notice that the value of the variable stays the same even if the reference has changed

```
> let a = 10;
> let b = a;
> a = 20;
> a , b
```

```
<• 20 , 10
```

While reassigning you can use the current value of the variable to modify it and assign it back to the same variable

```
> let c = 10;
  let c = c + 5;
  c;
```

```
15
```

```
> let string = “John”;
  string = string + “ Doe”;
  string;
```

```
John Doe
```

if you do not assign a value to the variable it will be given a value of undefined

```
> let anotherVariable = 10;
  anotherVariable;
```

```
<• Undefined
```

```
> const ourVariable;
```

```
✕ Uncaught SyntaxError: Missing
  initializer in const declaration
```

Using the keyword var was the old way of defining variables. You can still see it being used but it is better to // work with let and const instead.

```
var variableName = "cat";
```