KINGSLAND
UNIVERSITY

Introduction to JavaScript

Basic Syntax, Conditions and Loops

Table of Contents

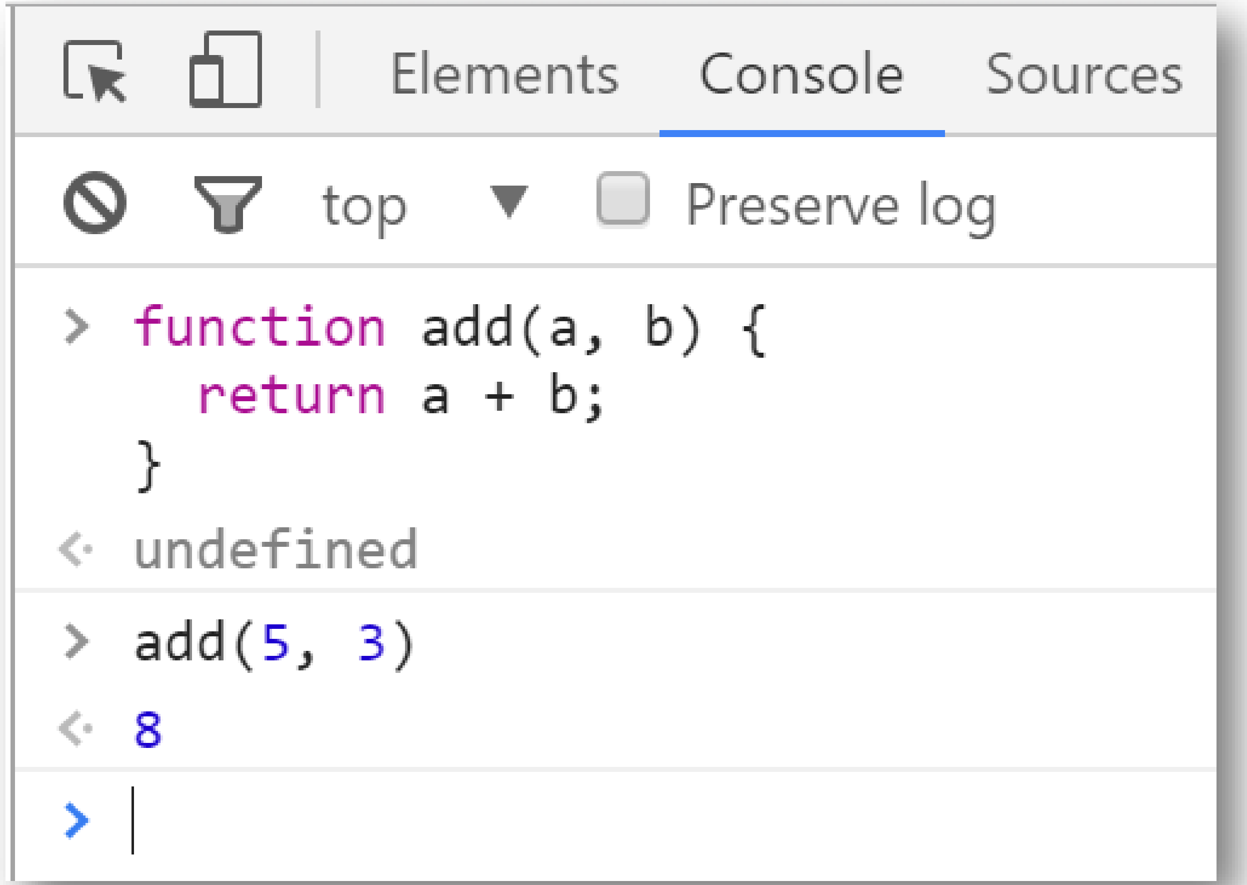
- ✓ Introduction and IDE
- ✓ Conditional Statements
- ✓ Loops
    - ✓ While-Loop
    - ✓ For-Loop
- ✓ Debugging and Troubleshooting

Introduction and IDE

Development Environments for JS

Chrome Web Browser

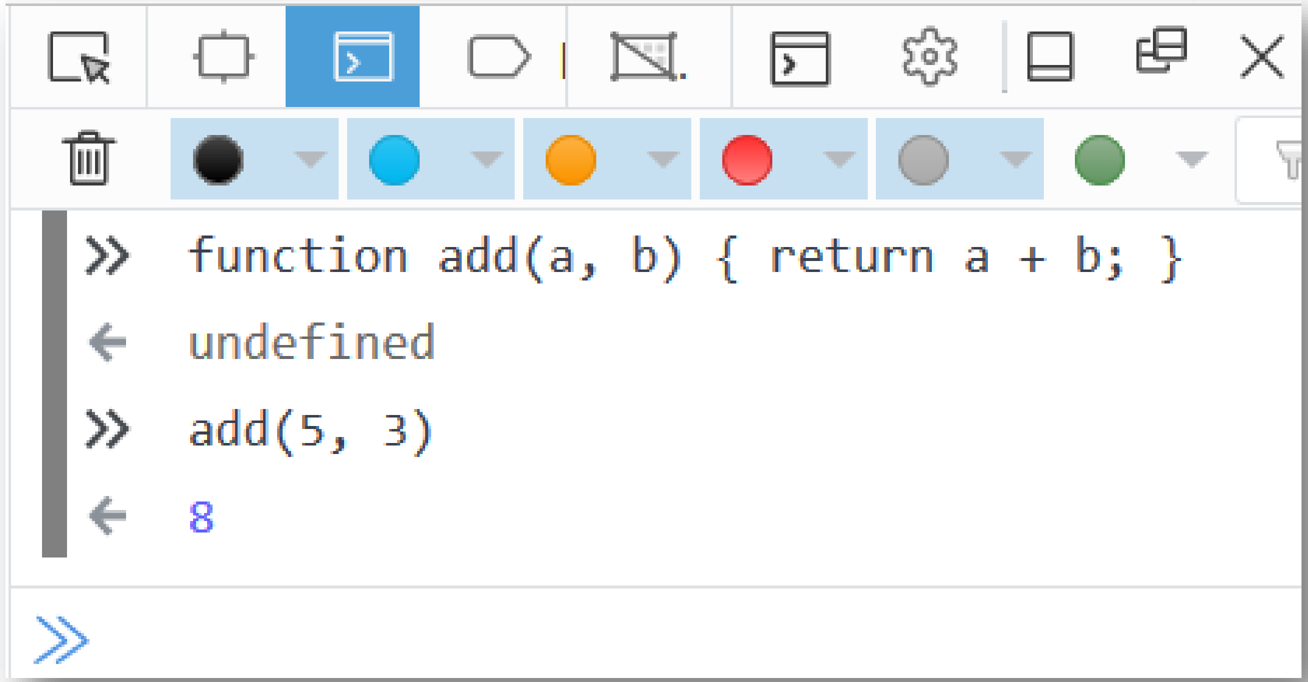
Developer Console: [F12]



```
> function add(a, b) {
    return a + b;
  }
< undefined
> add(5, 3)
< 8
>
```

Firefox Web Browser

Developer Console: [Ctrl] + [Shift] + [i]



```
>> function add(a, b) { return a + b; }
← undefined
>> add(5, 3)
← 8
>>
```

JavaScript Syntax

- The JavaScript syntax is similar to C#, Java and PHP
  - Operators, Variables, Conditional statements, loops, functions, arrays, objects and classes

Declare a variable with let

Conditional statement

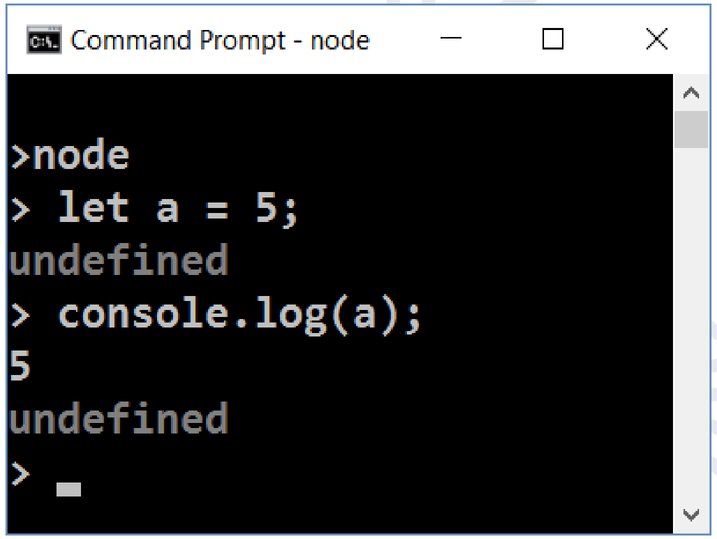
```
let a = 5;
let b = 10;
if (b > a) {
  console.log(b);
}
```

Body of the conditional statement

Node.js

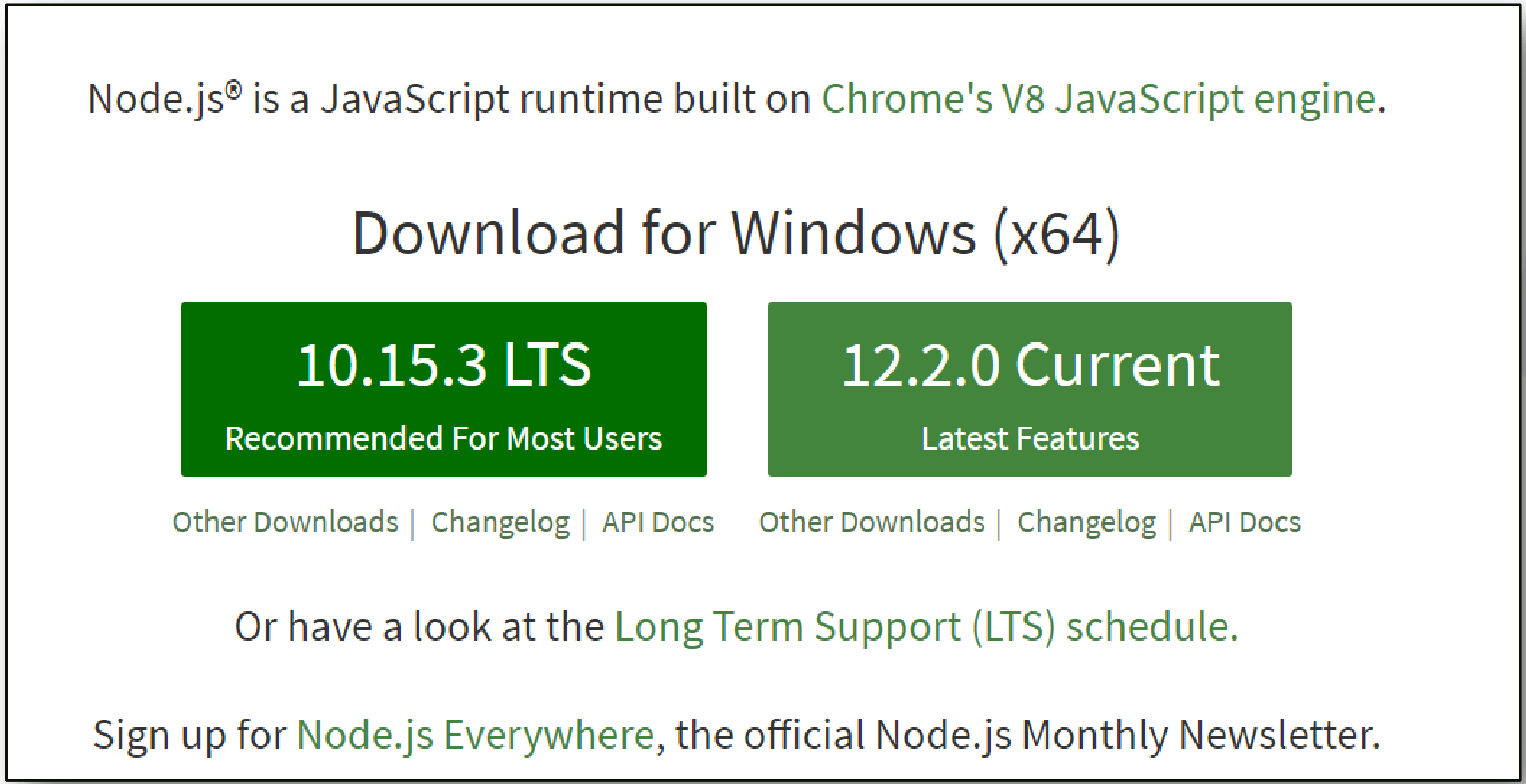
What is **Node.js**?

- ✓ Server-side JavaScript runtime
- ✓ Chrome V8 JavaScript engine
- ✓ NPM package manager
- ✓ Install node packages


Command Prompt - node

```
>node
> let a = 5;
undefined
> console.log(a);
5
undefined
> _
```

Install the Latest Node.js


Node.js® is a JavaScript runtime built on Chrome's V8 JavaScript engine.

Download for Windows (x64)

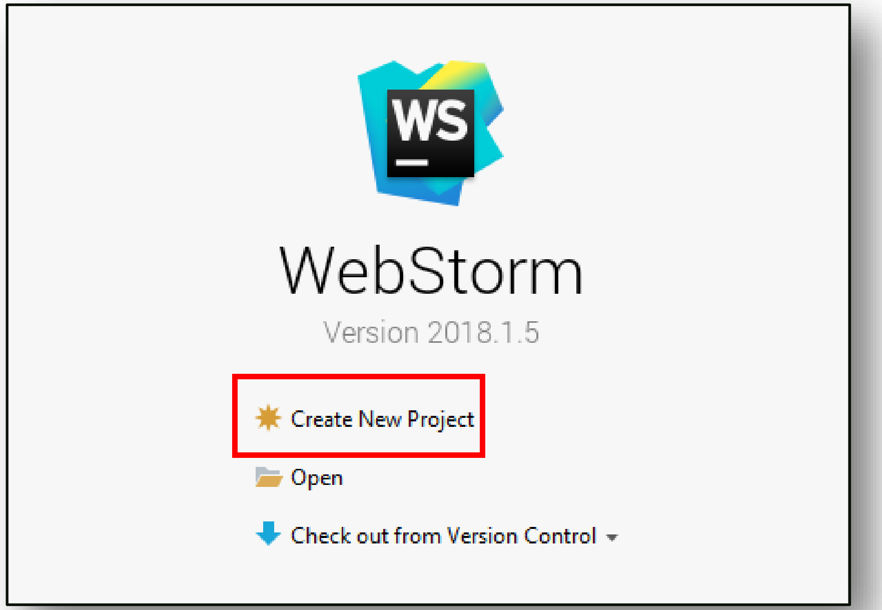
10.15.3 LTS
Recommended For Most Users
12.2.0 Current
Latest Features
Other Downloads | Changelog | API Docs
Other Downloads | Changelog | API Docs

Or have a look at the Long Term Support (LTS) schedule.

Sign up for Node.js Everywhere, the official Node.js Monthly Newsletter.

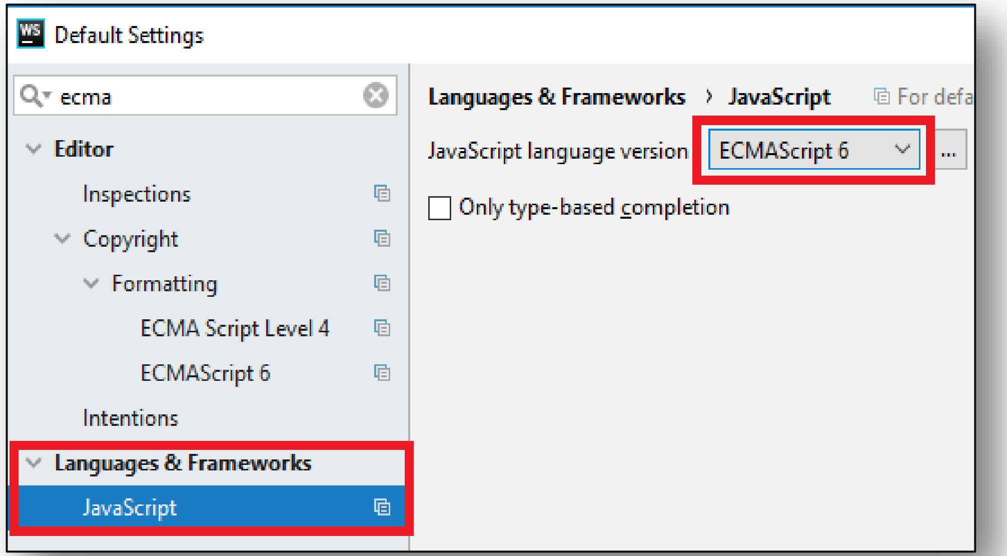
Using WebStorm

- **WebStorm** is powerful IDE for JavaScript and other languages
- Create a **new project**



Configurations

- Set up **ECMAScript 6** and **Node.js**
  - ECMAScript6 is a **standard** for JavaScript
  - Node is **environment** for JavaScript



Default Settings

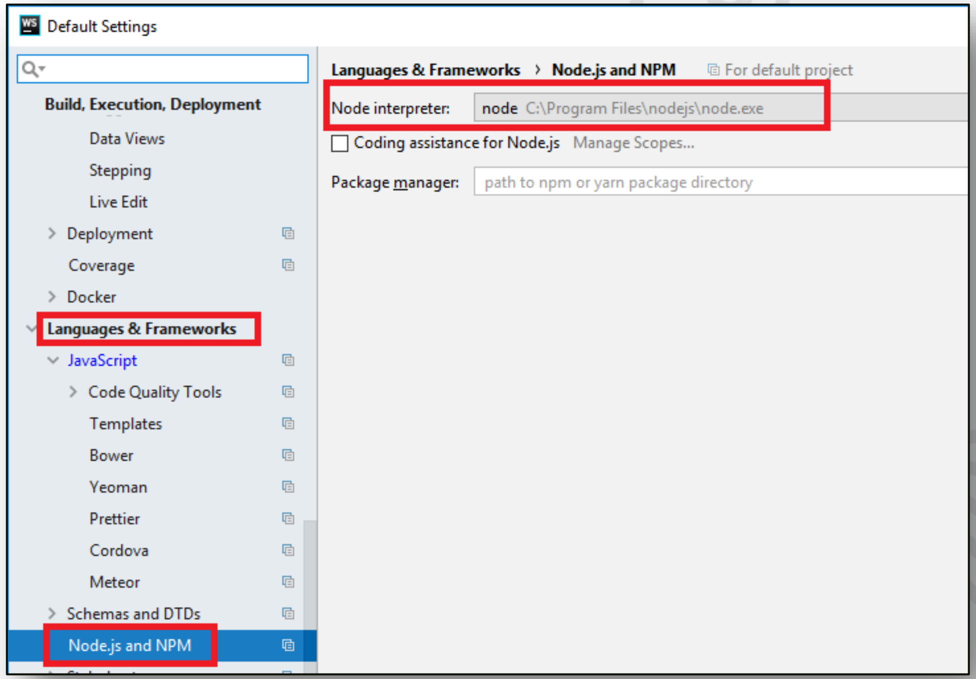
ecma

- Editor
  - Inspections
  - Copyright
    - Formatting
      - ECMA Script Level 4
      - ECMAScript 6
  - Intentions
- Languages & Frameworks
  - JavaScript

Languages & Frameworks › JavaScript — For defa

JavaScript language version: ECMAScript 6

☐ Only type-based completion



Default Settings

- Build, Execution, Deployment
  - Data Views
  - Stepping
  - Live Edit
  - Deployment
  - Coverage
  - Docker
- Languages & Frameworks
  - JavaScript
    - Code Quality Tools
    - Templates
    - Bower
    - Yeoman
    - Prettier
    - Cordova
    - Meteor
  - Schemas and DTDs
  - Node.js and NPM

Languages & Frameworks › Node.js and NPM — For default project

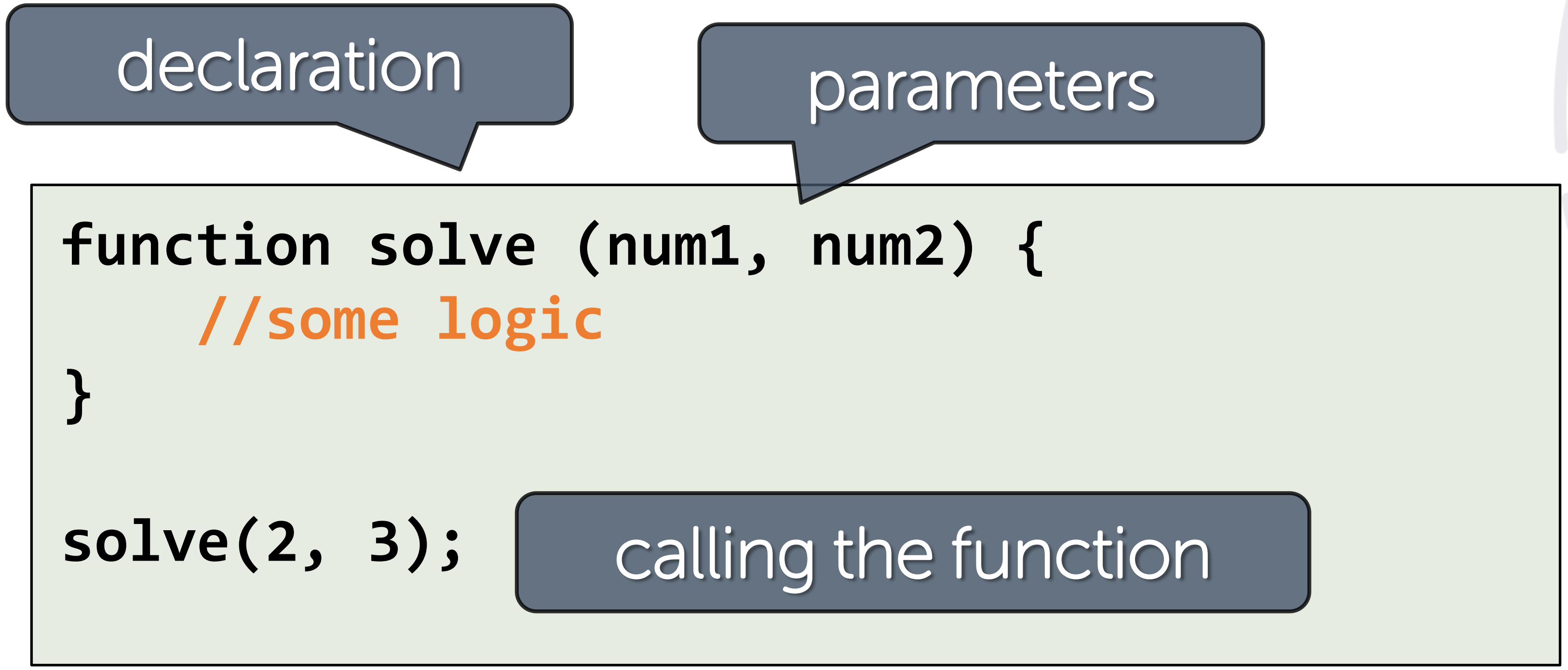
Node interpreter: node C:\Program Files\nodejs\node.exe

☐ Coding assistance for Node.js — Manage Scopes...

Package manager: path to npm or yarn package directory

Functions

- In order to solve different problems, we are going to use **functions** and the input will come as parameters
- A function is block of code, that executes when called



```
function solve (num1, num2) {
    //some logic
}

solve(2, 3);
```

Problem: Multiply Number by Two

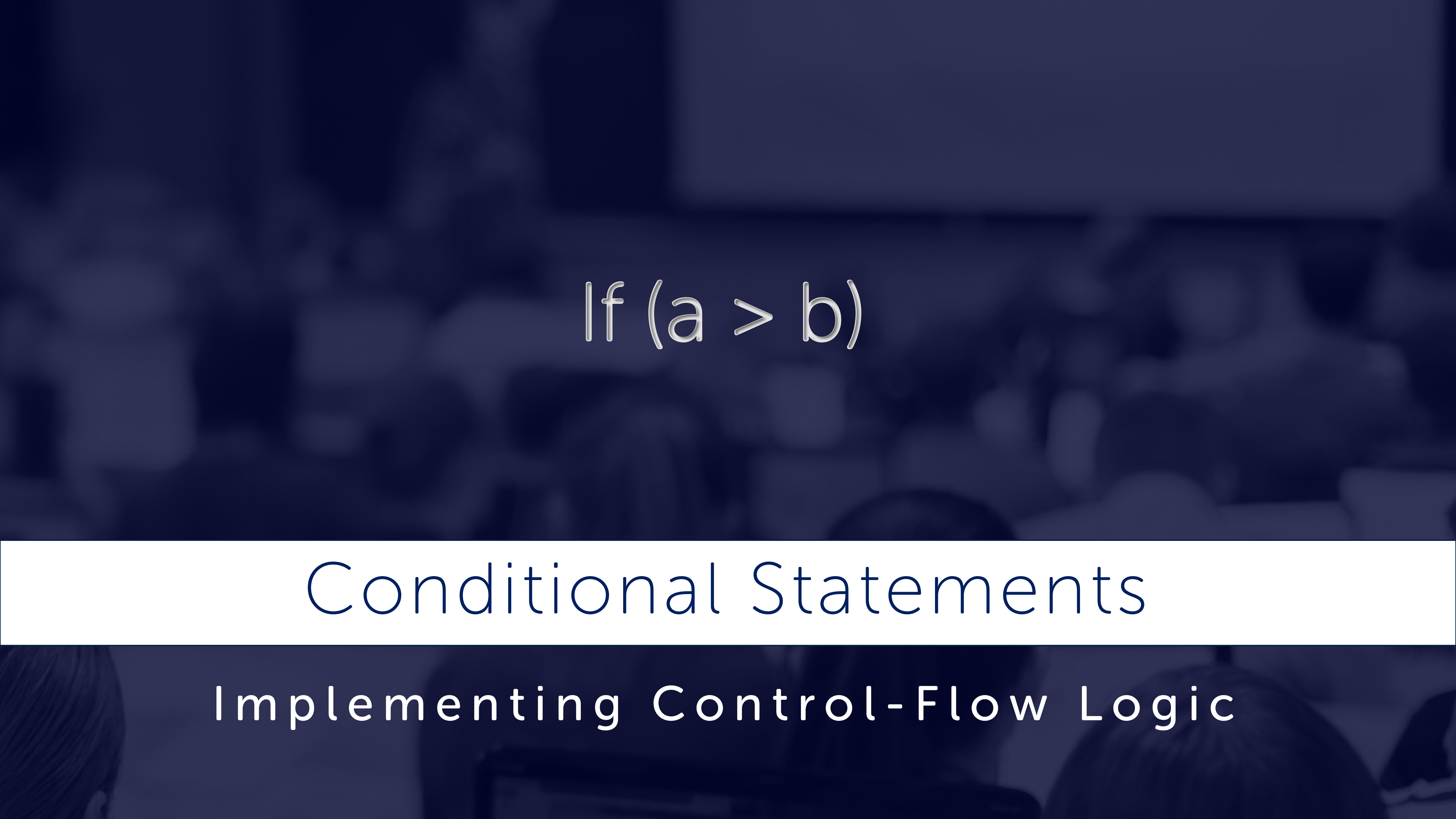
- Write a function that receives a **number** and prints as result that number **multiplied by two**

Input	Output
2	4

```
function solve (num) {
  console.log(num * 2);
}
solve(2);
```

Comparison Operators

Operator	Notation in JS
Equal value Equal value and type	== ===
Not equal value Not equal value/type	!= !==
Greater than	>
Greater than or Equal	>=
Less than	<
Less than or Equal	<=



If (a > b)

Conditional Statements

Implementing Control-Flow Logic

What is Conditional Statement

The **if-else** statement:

- Do action depending on condition

```
let a = 5;
if (a >= 5) {
  console.log(a);
}
```

If the condition is met, the code will execute

- You can chain conditions

```
else {
  console.log('no');
}
```

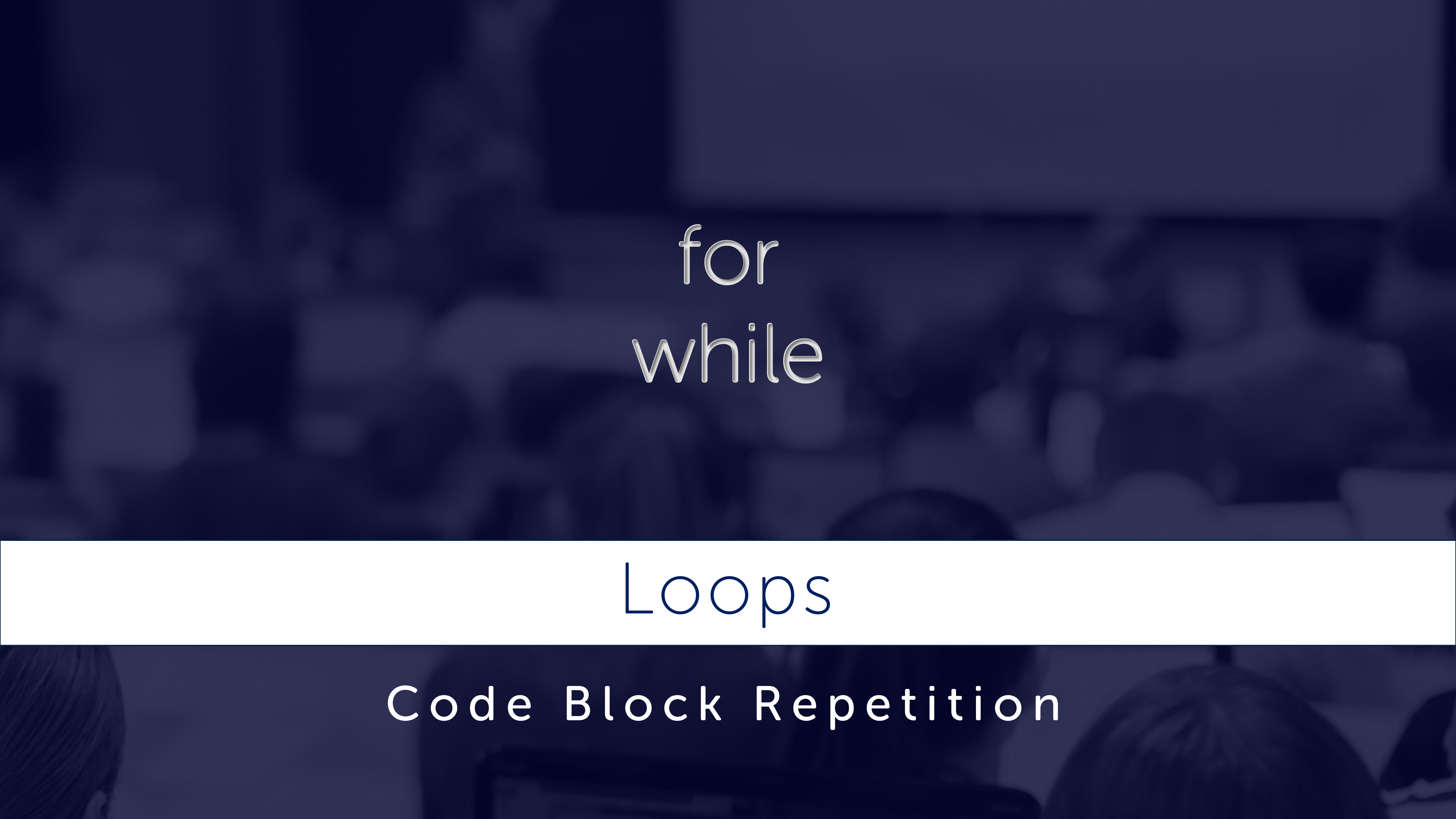
Continue on the next condition, if the first is not met

Problem: Excellent Grade

- Write a function that receives a **single number** and checks if the grade is excellent or not
- If it is, print "**Excellent**", otherwise print **"Not excellent"**

Input	Output
5.50	Excellent
4.35	Not excellent

```
function solve(grade){
  if (grade >= 5.50) {
    //TODO
  } else {
    //TODO
  }
}
```



for

while

Loops

Code Block Repetition

What Are Loops

The **for** loop:

- Repeats until the condition is evaluated

```
for (let i = 1; i <= 5; i++){
  console.log(i)
}
```

Incrementation in the condition

The **while** loop:

- Does the same, but has different structure

```
let i = 1
while (i <= 5) {
  console.log(i)
  i++
}
```

Incrementation outside the condition

Problem: Numbers from 1 to 5

- Create a function that prints all the numbers from 1 to 5 **(inclusive)** each on a separate line

Output
1
2
3
4
5

```
function solve () {
  for (let i = 1; i <= 5; i++) {
    //TODO: print
  }
}
```

Problem: Numbers from N to 1

- Write a function that receives a **number** and prints the numbers from **N to 1**. Try using a **while loop**

Input	Output
5	5 4 3 2 1

```
function solve(n) {
  while(/*TODO*/) {
    console.log(n);
    n--;
  }
}
solve(5);
```

Debugging the Code

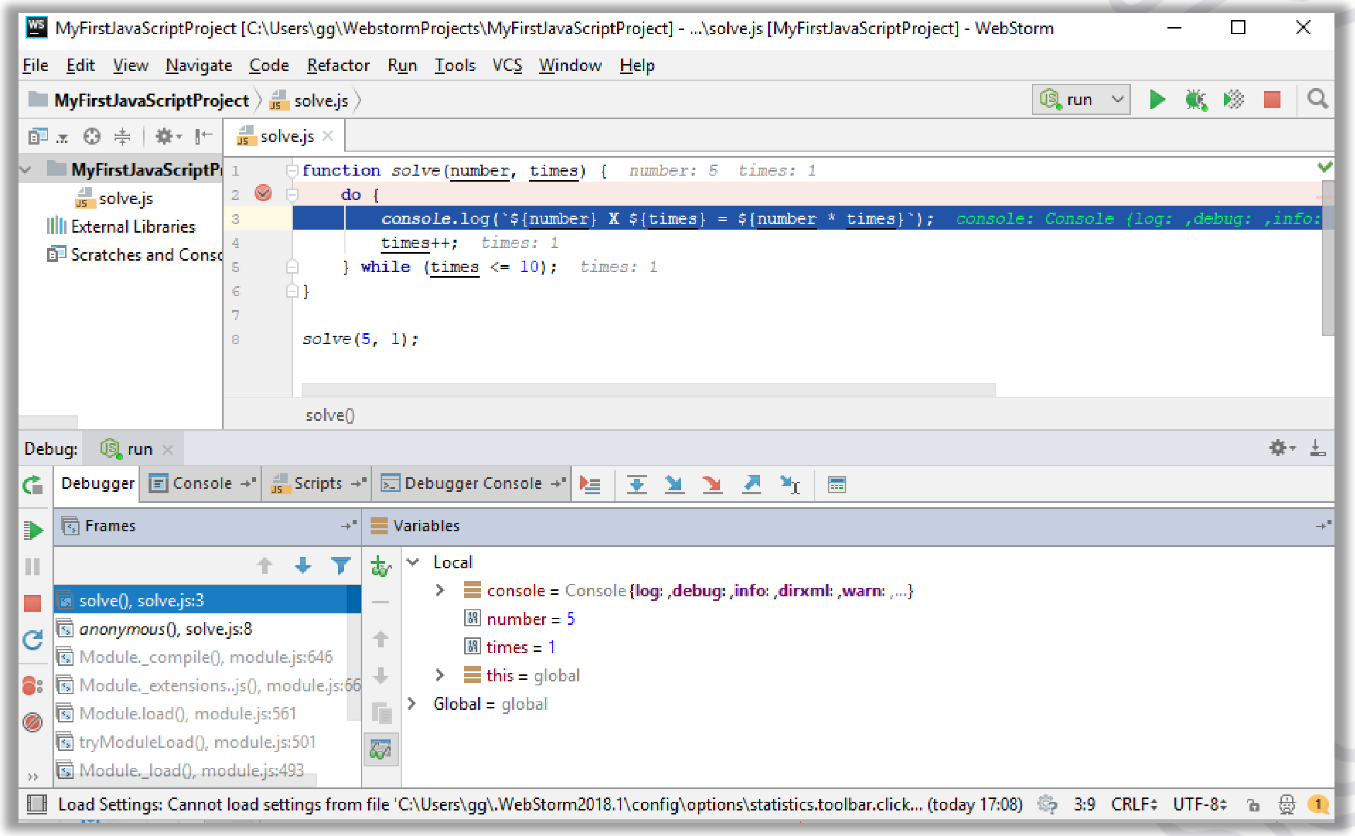
Using the WebStorm Debugger

Debugging the Code

- The process of **debugging application** include
  - Spotting an error
  - Finding the lines of code that cause the error
  - Fixing the error in the code
  - Testing to check if the error is gone and no new errors are introduced
- Iterative and continuous process

Debugging in WebStorm

- WebStorm has a built-in **debugger**
- It provides:
  - **Breakpoints**
  - Ability to **trace** the code execution
  - Ability to **inspect** variables at runtime


MyFirstJavaScriptProject [C:\Users\gg\WebstormProjects\MyFirstJavaScriptProject] - ...\solve.js [MyFirstJavaScriptProject] - WebStorm
File Edit View Navigate Code Refactor Run Tools VCS Window Help
MyFirstJavaScriptProject › solve.js
run
solve.js
MyFirstJavaScriptProject
solve.js
External Libraries
Scratches and Consoles
Debug: run
Debugger
Console
Scripts
Debugger Console
Frames
Variables

```
function solve(number, times) {   number: 5  times: 1
    do {
        console.log(`${number} X ${times} = ${number * times}`);   console: Console {log: ,debug: ,info:
        times++;   times: 1
    } while (times <= 10);   times: 1
}

solve(5, 1);
```

```
solve()
```

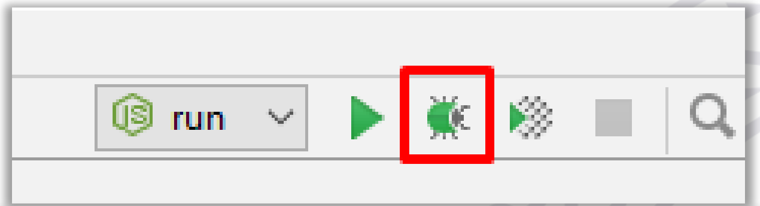
```
solve(), solve.js:3
anonymous(), solve.js:8
Module._compile(), module.js:646
Module._extensions..js(), module.js:66
Module.load(), module.js:561
tryModuleLoad(), module.js:501
Module._load(), module.js:493
```

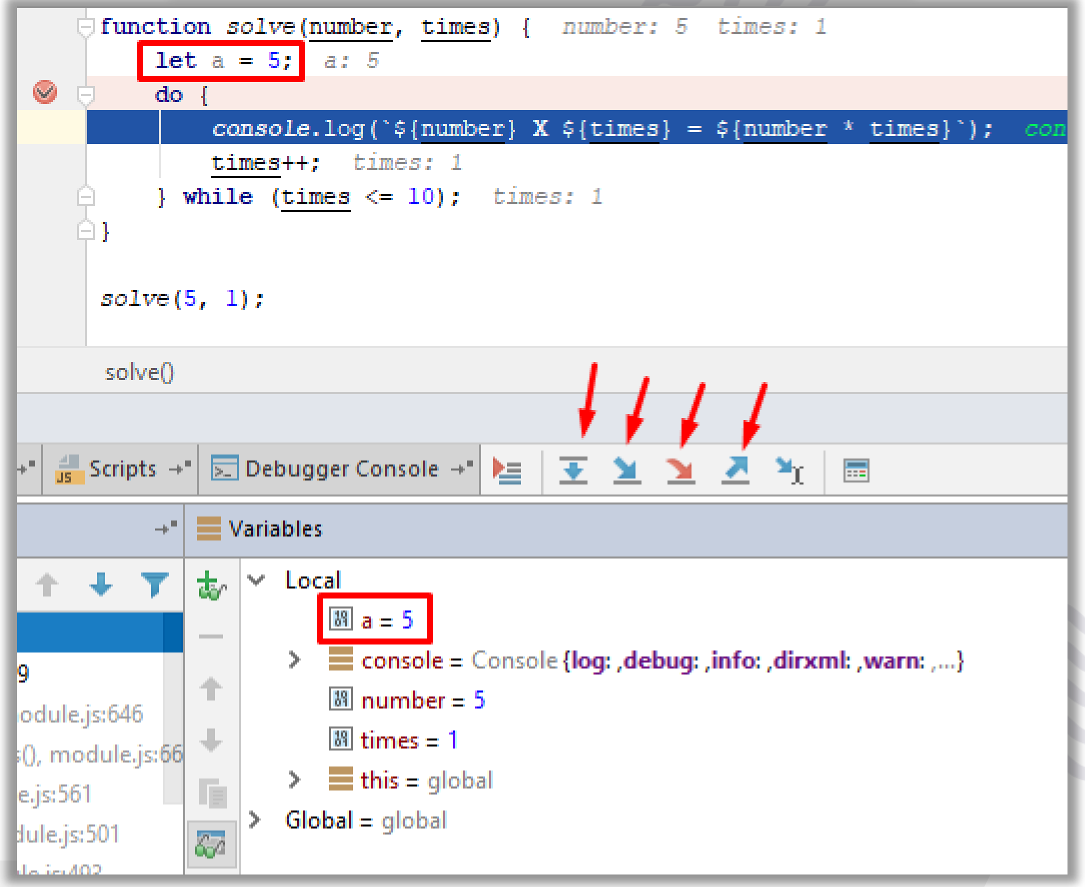
```
Local
  console = Console {log: ,debug: ,info: ,dirxml: ,warn: ,...}
  number = 5
  times = 1
  this = global
Global = global
```

```
Load Settings: Cannot load settings from file 'C:\Users\gg\.WebStorm2018.1\config\options\statistics.toolbar.click... (today 17:08)   3:9  CRLF÷  UTF-8÷
```

Using the Debugger in WebStorm

- Start without Debugger: **[Shift+F10]**
- Toggle a breakpoint: **[Shift+F9]**
- Trace step by step: **[F7]**
- Force step into: **[Alt+Shift+f7]**
- Using the **Local**
- Conditional breakpoints
- Enter debug mode after exception





```
function solve(number, times) {   number: 5  times: 1
    let a = 5;   a: 5
    do {
        console.log(`${number} X ${times} = ${number * times}`);   con
        times++;   times: 1
    } while (times <= 10);   times: 1
}

solve(5, 1);
```

Summary

- Declare variables with **'let'**
- Use **if-else** statements to check for conditions
- Use **loops** to avoid repeating code
- Use the **debugger** to check for mistakes in the code

Questions?

License

KINGSLAND UNIVERSITY

THANK YOU