Playwright Training

Control Statements

Testleaf
Always Ahead

The Golden Circle

What

What is Control statement?

The statements that define the flow of the program

Why

Why is Control Statement required?

Provide a logic based flow like chrome based vs firefox based execution

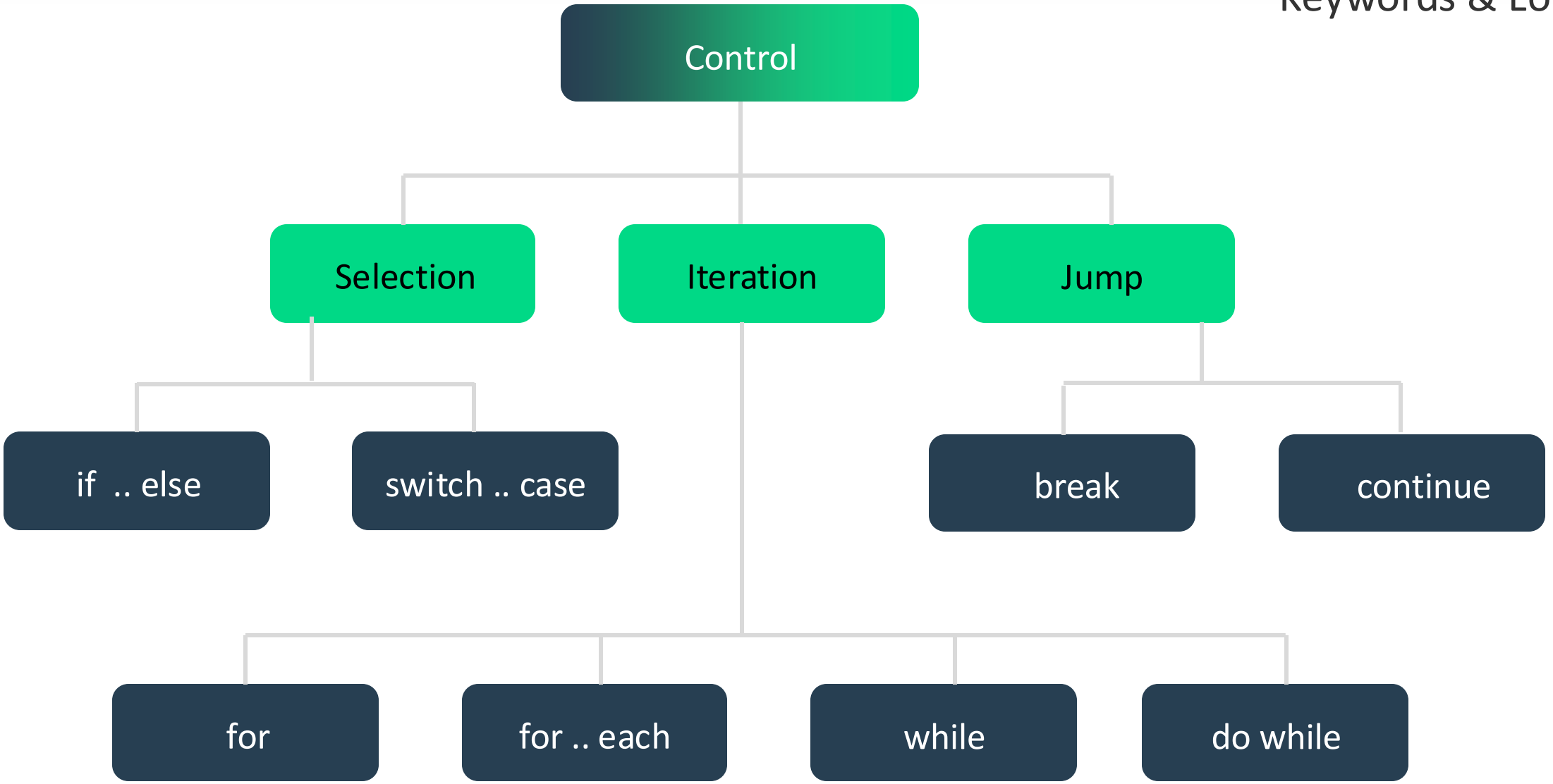
How

How to implement Control Statements?

Using Selection, Iteration and Jump Statements

Control Statements

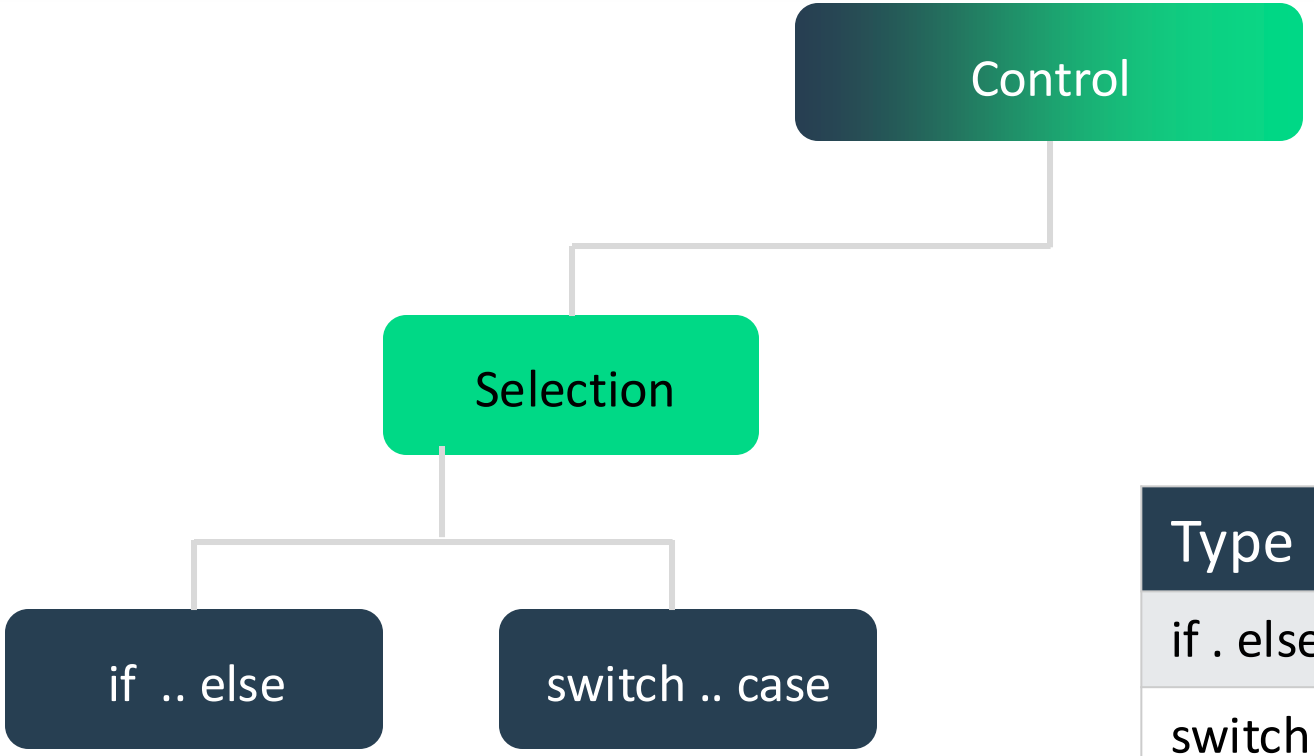
Keywords & Lower case



Testleaf
Always Ahead

Selection Statements : When to use what?

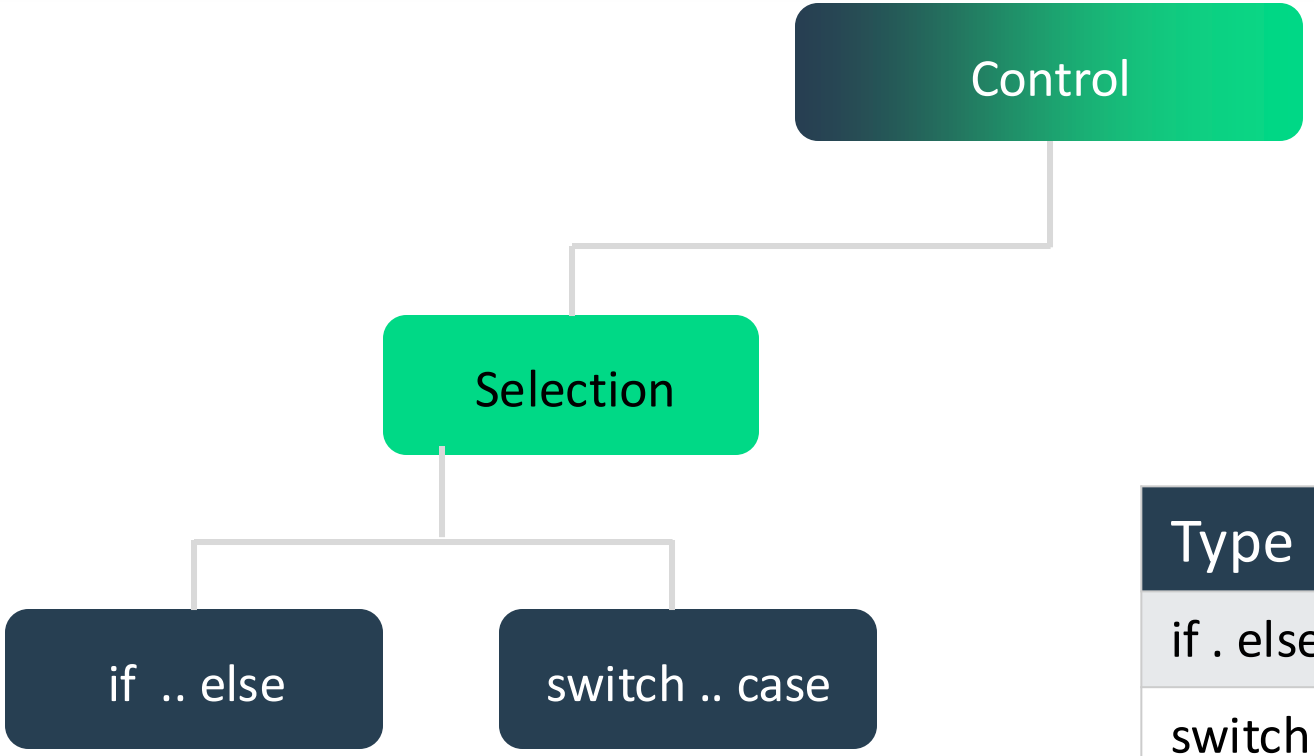
Keywords & Lower case



Type	When
if . else	When you have few conditions
switch . case	When you have multiple conditions

Selection Statements : Examples

Keywords & Lower case



Type	Example
if . else	Report pass or fail
switch . case	Launch based on browser types

if syntax

if Statement (Single Condition)

- The if statement **executes code only if the condition is true**.

Example: Check if a number is positive

```
let num = 10;

if (num > 0) {
  console.log("The number is positive.");
}


// Output: The number is positive.
```

✅ **Use if when you need to check a single condition.**

if..else syntax

if...else Statement (Two Conditions)

- The else block runs **if the if condition is false**.

Example: Check if a person can vote

```
let age = 17;

if (age >= 18) {
  console.log("You can vote.");
} else {
  console.log("You cannot vote.");
}

// Output: You cannot vote.
```

✅ **Use if...else when there are two possible outcomes.**

if..else syntax

if...else if...else (Multiple Conditions)

- Use else if to **check multiple conditions**.
- The first true condition executes, and the rest are skipped.

Example: Grade System

```
let score = 85;

if (score >= 90) {
  console.log("Grade: A");
} else if (score >= 80) {
  console.log("Grade: B");
} else if (score >= 70) {
  console.log("Grade: C");
} else {
  console.log("Grade: F");
}
// Output: Grade: B
```

switch Statement syntax

switch Statement (Multiple Fixed Choices)

- switch is useful when you have **multiple cases to compare** with a single variable.
- The break statement **prevents execution from continuing to the next case**.

Example: Check day of the week

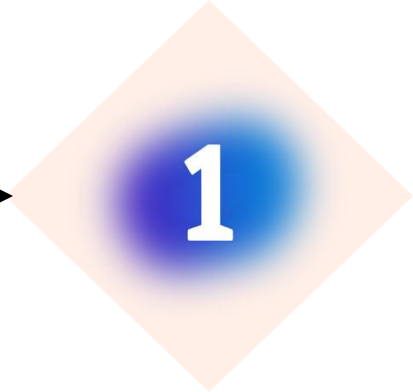
```
let day = “Sunday";

switch (day) {
  case "Monday":
    console.log("Start of the workweek!");
    break;
  case "Friday":
    console.log("Weekend is near!");
    break;
  case "Sunday":
    console.log(“Playwright Training, it's Sunday!");
    break;
  default:
    console.log("It's just another day.");
}
// Output: Playwright Training, it's Sunday!
```

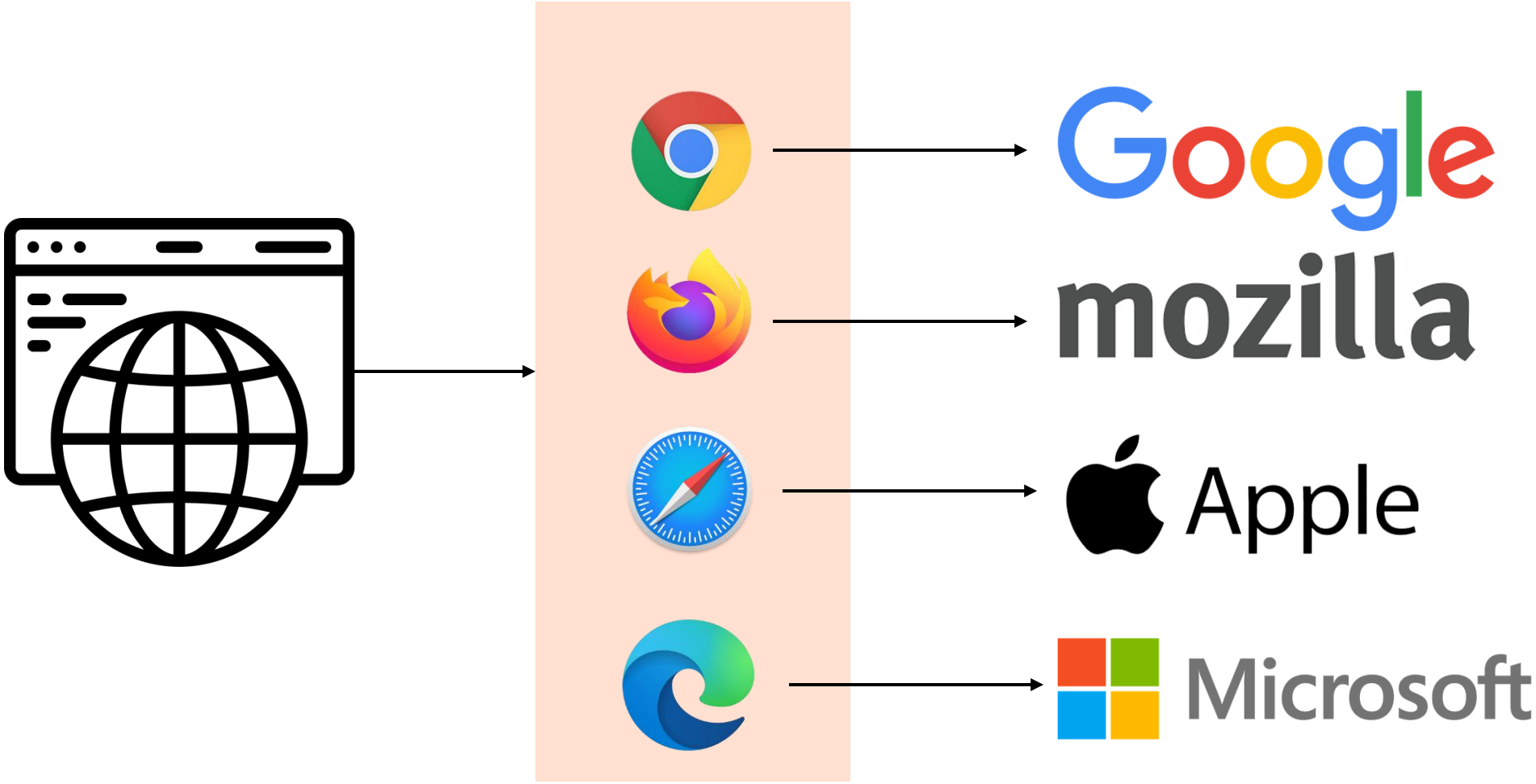
✅ **Use switch when a variable has multiple possible values (like different browser options).**

Testleaf
Always Ahead

Lets write a sample code (positive number)

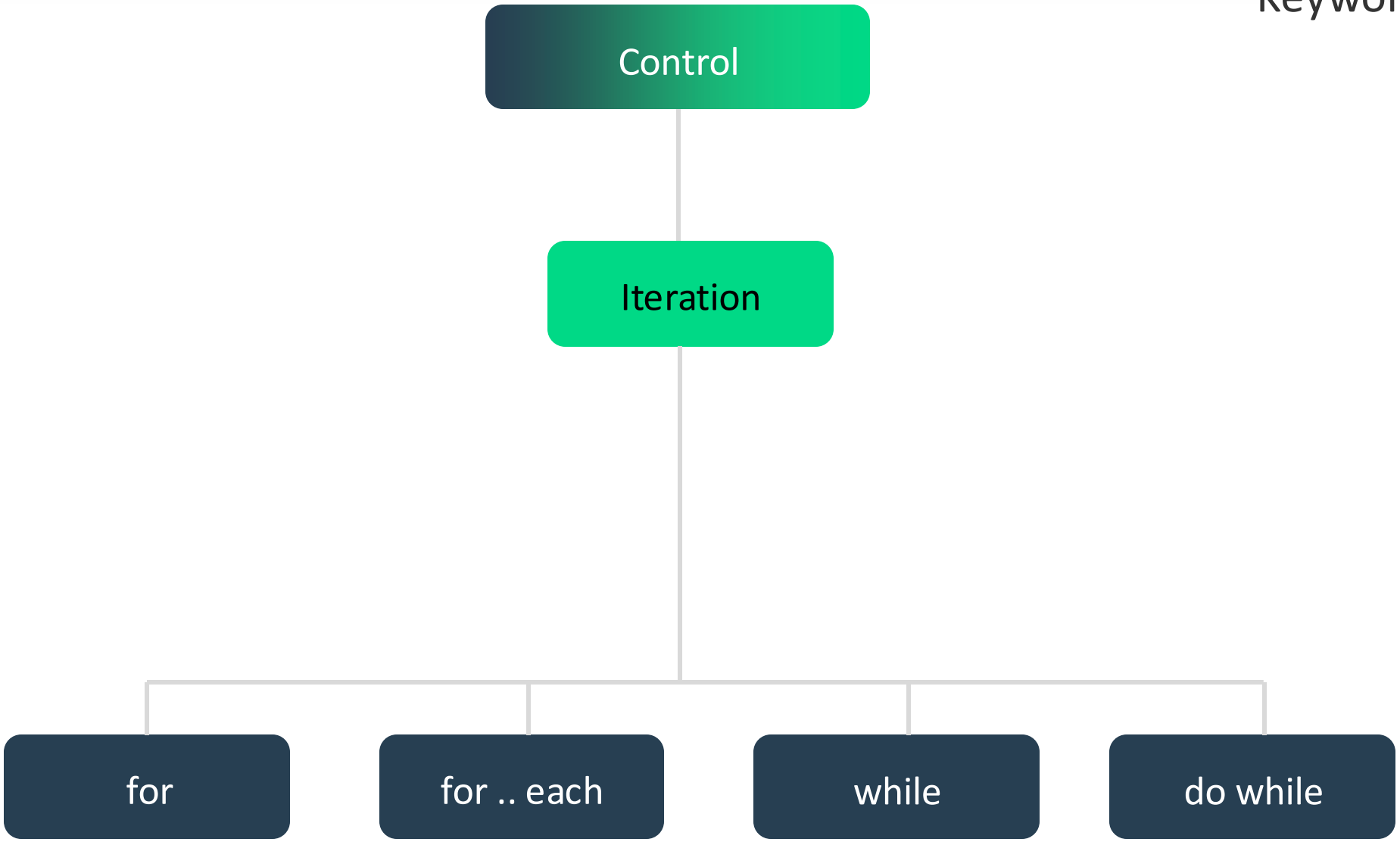


Lets write a sample code (browser vendor)



Control Statements

Keywords & Lower case



Testleaf
Always Ahead

for Loop

for Loop (Fixed Iterations)

- The for loop runs **a specific number of times**.
- It has **three parts**:

1. **Initialization** → let i = 0 (Starting point)
2. **Condition** → i < 5 (Loop runs while i is less than 5)
3. **Increment/Update** → i++ (Increase i by 1 each time)

Example: Print numbers from 1 to 5

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

// Output: 1 2 3 4 5
```

✅ **Use for when you know exactly how many times you want to loop.**

forEach Loop

forEach Loop (Array Looping)

- forEach() is used for **iterating over arrays**.
- It executes a function for each **element** in the array.

Example: Print each fruit in an array

```
let automationTools = [“Playwright”, “Selenium", “Puppeteer"];

automationTools.forEach(function (automation) {
  console.log(automation);
});

// Output:
// Playwright
// Selenium
// Puppeteer
```

✅ **Use forEach for looping through arrays.**

while Loop

while Loop (Condition-Based)

- Runs **as long as** the condition is true.
- **Useful when we don't know how many times the loop should run.**

Example: Count from 1 to 5

```
let count = 1;
while (count <= 5) {
  console.log(count);
  count++;  // Increment count
}

// Output: 1 2 3 4 5
```

✅ **Use while when the number of iterations is unknown.**

do..while Loop

do...while Loop (Runs at least once)

- Runs **at least once**, even if the condition is false.
- The **condition is checked after** the loop executes.

Example: Run once even if condition is false

```
let num = 10;
do {
  console.log(num);
  num++;

} while (num <= 5); // Condition is false (10 is not ≤ 5)
```

Output: 10

✅ **Use do...while when you want the loop to run at least once.**

Lets write a sample code (print numbers 1 … 10)

Operators

1. Arithmetic Operators.
2. Assignment Operators.
3. Comparison Operators.
4. Equality Operators.
5. Logical Operators.
6. Ternary (Conditional) Operator (?:).
7. Optional Chaining Operator (?.) .

Arithmetic Operators

Operator	Name	Example	Result
+	Addition	10 + 5	15
-	Subtraction	10 - 5	5
*	Multiplication	10 * 5	50
/	Division	10 / 5	2
%	Modulus	10 % 5	0
**	Exponentiation	2 ** 3	8
++	Increment	let x = 5; x++	6
--	Decrement	let y = 5; y--	4

Assignment Operators

Operator	Name	Example	Equivalent To
=	Assign	x = 10	x = 10
+=	Add and assign	x += 5	x = x + 5
-=	Subtract and assign	x -= 3	x = x - 3
*=	Multiply and assign	x *= 2	x = x * 2
/=	Divide and assign	x /= 2	x = x / 2
%=	Modulus and assign	x %= 2	x = x % 2
**=	Exponentiation and assign	x **= 3	x = x ** 3

Comparison Operators

Operator	Name	Example		Result
==	Equal to (loose equality)	5 == '5'		true
===	Strict equal (type + value)	5 === '5'		false
!=	Not equal	5 != '5'		false
!==	Strict not equal	5 !== '5'		true
>	Greater than	10 > 5		true
<	Less than	10 < 5		false
>=	Greater than or equal to	10 >= 10		true
<=	Less than or equal to	10 <= 5		false

Equality Operator

== (Equality Operator - Loose Comparison)

- **Checks only values, not data types**.
- Converts (coerces) values to the **same type** before comparing.

Example:

console.log(5 == "5"); // Output: true

Why?

- "5" (string) is automatically converted to 5 (number).v
- Since 5 == 5, the result is true.

✅ **Use == when you want JavaScript to convert types automatically.**

Operands	What JavaScript Does
Boolean == Any	Convert Boolean to Number (true → 1)
String == Number	Convert String to Number ("5" → 5)

Strict Equality Operators

=== (Strict Equality - Strong Comparison)

- **Checks both values and data types** (No type conversion).
- If the types **don't match**, it returns false immediately.

Example:

```
console.log(5 === "5"); // Output: false
```

Why?

- 5 is a **number**, "5" is a **string**.
- Since the types are **different**, the result is false.

✅ **Use === when you want to compare values without type conversion.**

Inequality Operator

!= (Inequality - Loose Comparison)

- **Checks if values are different** (ignores data type).
- Converts (coerces) values to the **same type** before comparing.

Example:

console.log(5 != "5"); // Output: false

Why?

- "5" (string) is converted to 5 (number).
- Since 5 == 5 is true, 5 != "5" is false.

✅ **Use != when you want to check for inequality but allow type conversion.**

Strict Inequality Operators

!== (Strict Inequality - Strong Comparison)

- **Checks both value and data type** (No type conversion).
- If types are different, returns true immediately.

Example:

```
console.log(5 !== "5"); // Output: true
```

Why?

- 5 (number) is **not** the same type as "5" (string).
- Since types are different, it returns true.

✅ **Use !== when you want to check inequality without type conversion.**

Logical Operators

Operator	Name	Example	Result
&&	Logical AND	true && false	false
||	Logical OR	true || false	true
!	Logical NOT	!true	false

&& - Logical AND

```
let age = 25;
if (age > 18 && age < 60) {
console.log("You are eligible to work.");
}
```

|| - Logical OR

```
let isWeekend = true;
let isHoliday = false;
if (isWeekend || isHoliday) {
console.log("You can relax today!");
}
```

! – Logical NOT

```
let isLoggedIn = false;
if (!isLoggedIn) {
console.log("Please log in to continue.");
}
```

Ternary(Conditional) Operators

Operator	Name	Example	Result
? :	Ternary Operator	10 > 5 ? "Yes" : "No"	"Yes"

What is it?

The **ternary operator** is a shorthand for writing an if-else condition in a single line.
[Refer config file for real time understanding]

Syntax:

condition ? value_if_true : value_if_false;

- If the condition is **true**, it returns value_if_true.
- If the condition is **false**, it returns value_if_false.

Example:

```
let age = 18;
let message = age >= 18 ? "You can vote!" : "You cannot vote.";
console.log(message); // Output: "You can vote!"
```

Equivalent using if-else:

```
let age = 18;
let message;
if (age >= 18) {
    message = "You can vote!";
} else {
    message = "You cannot vote.";
}
console.log(message);
```

✅ **Ternary operator is useful when you want to write simple conditions in a short way.**

Optional Chaining Operator (?.)

Operator	Name	Example	Result
?.	Optional Chaining	obj?.property	undefined (if obj is null)

Optional Chaining Operator (?.)

What is it?

The ?. operator helps **safely access properties of an object** without causing an error if the property doesn't exist. [Refer frames concept for real time understanding.]

Syntax: object?.property;

- If object exists, return object.property.
- If object is null or undefined, return undefined instead of throwing an error.

Example 1: Without Optional Chaining (Error)

```
let user = null;
console.log(user.name); // ❌ TypeError: Cannot read properties of null
```

```
let user = { name: “Ravi"};
console.log(user.name); // Output: “Ravi"
```

Example 2: With Optional Chaining (Safe)

```
let user = null;
console.log(user?.name); // ✅ Output: undefined (No error)
```

Testleaf
Always Ahead

Summary

- 3 types of control statements: Selection, Iteration and Jump
- **Selection**: if – else vs switch – case (performance oriented)
- **Iteration** : for (count based) vs while (condition based)
- **Jump** : break (out of iteration), continue (skip iteration)
- **Operators**: Different operators used in JavaScript.

Classroom Exercise (Breakout)

- Write a program to print only odd numbers between 1 to 20
- Before writing the code – follow the 3 step process:
  - Understand the problem
  - Solve the problem (Using Pseudocode)
  - Write the code