1.Invoke Function

The code inside a function is executed when the function is invoke. The code inside a function in not executed when the function is defined

```
function calculation(a , b) {

}
undefined
function calculation(a , b) {
    return a / b;
}
undefined
calculation(10,5)
2
```

2. Break Statement

The break statement can also be used to jump out of a loop.

```
<h2>JavaScript Loops</h2>

<p>A loop with a <b>break</b> statement.</p>

<p id="demo"></p>

<script>
let text = "";
for (let i = 0; i < 10; i++) {
  if (i === 3) { break; }
  text += "The number is " + i + "<br>";
}

document.getElementById("demo").innerHTML = text;
</script>
```

The number is 0
The number is 1
The number is 2

Continue Statement

The continue statement breaks one iteration (in the loop), if a specified condition occurs, and continues with the next iteration in the loop.

```
h2>JavaScript Loops</h2>

p>A loop with a <b>continue</b> statement.</p>

p>A loop which will skip the step where i = 3.</p>

p id="demo"></p>

script>
et text = "";
or (let i = 0; i < 10; i++) {
 if (i === 3) { continue; }
 text += "The number is " + i + "<br>";

ocument.getElementById("demo").innerHTML = text;
/script>
```

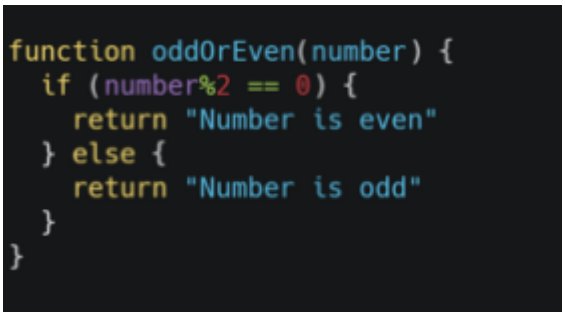
A loop which will skip the step where i = 3.

The number is 0
The number is 1
The number is 2
The number is 4
The number is 5
The number is 6
The number is 7
The number is 8
The number is 9

3. **Types of Function**

Named Function:

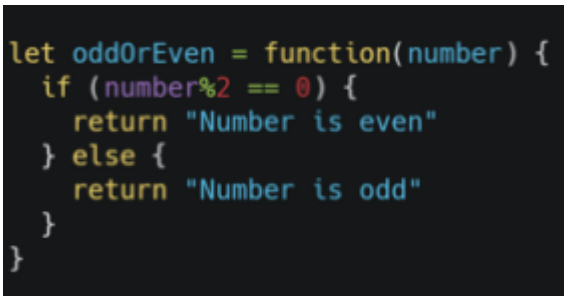
Named function is the function that we ***define*** it in the code and then call it whenever we need it by referencing its *name and passing some arguments* to it. Named functions are useful if we need to call a function **many times** to pass **different values** to it or run it several times.



```
function oddOrEven(number) {
  if (number%2 == 0) {
    return "Number is even"
  } else {
    return "Number is odd"
  }
}
```

Anonymous Function

The anonymous functions don't have **names**. They need to be tied to something: *variable or an event to run*.



```
let oddOrEven = function(number) {
  if (number%2 == 0) {
    return "Number is even"
  } else {
    return "Number is odd"
  }
}
```

Immediately Invoked Function

Invoked function expression runs as soon as the ***browser encounters*** it. The benefit of this function is that it **runs immediately** where it's located in the code and produces a *direct output*. That means it is **unaffected** by code which appears further down in the script which can be useful.

```
let message = (function () {
    let name = "Jon Snow";
    return name;
})();
// Immediately creates the output:
result; // "Jon Snow"
```

4.**String Methods**

1. **at()**

It takes Integer value and returns a new String. This methods allows for positive and negative Integers. Negative integers count back from the last string characters.

It is same As **CharAt()** methods.

```
var sentence = 'Hai, How are you?'
undefined
var storeData = sentence.at(6);
undefined
storeData
'o'
```

2. charCodeAt()

It returns an integer value between 0 to 65535 representing UTF-16 Value. (ASCII Value)

```
> var sentence = 'Hai, How are you?'
<· undefined
> var storeData = sentence.charCodeAt('o');
<· undefined
> storeData
<· 72
```

3. concat()

It returns concatenates the string arguments to the calling string and returns a new string.

```
> var str_1 = 'I learn';
< undefined
> var str_2 = 'Javascript';
< undefined
> var addString = str_1 . concat('',str_2);
< undefined
> addString
< 'I learnJavascript'
> var addString = str_1 . concat(',',str_2);
< undefined
> addString
< 'I learn,Javascript'
```

4. endsWith()
The Methods used to string ends with the character of a specified string, it return True of False statement. (string, length)

```
var addString = str_1 . concat(',',str_2);
undefined
addString.endsWith('Javascript');
true
addString.endsWith('Javascript',15);
false
```

5. fromCharCode()

The result is string created from the specified sequence of UTF-16 code units.

```
var values = (99);
undefined
String.fromCharCode(values);
'c'
var values = (114);
undefined
String.fromCharCode(values);
'r'
```

6.includes()

It performs case-sensitive search to determine whether one string may be found within another string, returning true or false condition.

```
var word = 'Js is Behaviour Layer';
undefined
word.includes('Js');
true
word.includes('and');
false
```

7.indexOf()
Returns Index position.

```
var check = 'The sunrise is Beautiful!!';
undefined
check.indexOf('sunrise');
4
```

8. padEnd()

String reaches a given length. The padding is applied from the end of the current string.

```
var applyStar = 'Hello, World';
undefined
applyStar.padEnd(20,'*');
'Hello, World********'
```

9. padStart()

String reaches a given length. The padding is applied from the start of the current string.

```
var applyStar = 'Hello, World';
undefined
applyStar.padEnd(20,'*');
'Hello, World********'
applyStar.padStart(20,'*');
'********Hello, World'
```

10.repeat()

returns a new string which contains the specified number of copies of the string on which it was called, concatenated together.

```
var happy = 'we are!!';
undefined
happy.repeat(5);
'we are!!we are!!we are!!we are!!we are!!'
```

11. replace()
It return replace the given string. The changes included in (old string) given string.

```
var change = 'The tiger is king of Forest!!';
undefined
change.replace('tiger','lion');
'The lion is king of Forest!!'
```

12. replaceall()

It same as replace() method.

```
var string = 'The quick brown fox jumps over the lazy dog. If the dog reacted, was it really lazy?';
undefined
string.replaceAll('dog','cat');
'The quick brown fox jumps over the lazy cat. If the cat reacted, was it really lazy?'
```

13. search()

method executes a search for a match between a regular expression and this String object.

```
var find = 'Fabevy Technology, Welcome all';
undefined
find.search('Welcome');
19
```

14. slice()

This method extracts a section of a string and returns it as a new string, without modifying the original string.

```
var word = 'American Standard Code for Information and Interchange';
undefined
word.slice(20);
'de for Information and Interchange'
word.slice(4,9);
'ican '
word.slice(-4,-2);
'an'
word.slice(-5);
'hange'
```

16.startsWith()

It returns string begins with the characters of a specified string, returning true or false statements. (word, position)

```
var course = 'I am Developer..!!';
undefined
course.startsWith('I')
true
course.startsWith('am')
false
```

17. substring()

It part of the string between start and end index value, it is saame as slice.

```
> var browsers = 'Google Chrome';
< undefined
> browsers.substring(5);
< 'e Chrome'
> browsers.substring(3,9);
< 'gle Ch'
```

18. toString()

It representing a string of the specified object.

```
var phones = 'samsung';
undefined
phones.toString();
'samsung'
```

19. toUpperCase()

The Result is an Uppercase model.

```
var device = 'I have Mac Book Pro';
undefined
device.toUpperCase();
'I HAVE MAC BOOK PRO'
```

20. trim()

It return removes whitespaces from both ends, without modifying a original string.It is Include all characters (space, tab,and etc).

```
var sentence = '   ReactJs is Very Important   ';
undefined
sentence.trim();
'ReactJs is Very Important'
```

21. trimEnd()

This method removes whitespaces from the end of a string.

```
var sentence = '   ReactJs is Very Important   ';
undefined
sentence.trimEnd();
'   ReactJs is Very Important'
```

22. trimStart()

This method removes whitespaces from the beginning of a string.

```
var sentence = '   ReactJs is Very Important   ';
undefined
sentence.trimStart();
'ReactJs is Very Important   '
sentence.trimRight();
'   ReactJs is Very Important'
sentence.trimLeft();
'ReactJs is Very Important   '
```

23.valueOf()

It return primitive value of a string.

```
var sentence = '   ReactJs is Very Important   ';
undefined
sentence.valueOf()
'   ReactJs is Very Important   '
'
```

6. Ternary operator

The conditional (ternary) operator is the only JavaScript operator that takes three operands: a condition followed by a question mark ( ), then an expression to execute if the condition is truthy followed by a colon ( ), and finally the expression to execute if the condition is falsy. This operator is frequently used as an alternative to an if...else statement.

```
function ternary(value){
    return value ? 10 : 20;}
undefined
ternary(true)
10
ternary(false)
20
ternary(null)
20
```

7. Link in css files

There are three ways of inserting a style sheet

1. External Css

you can change the look of an entire website by changing just one file!
Each HTML page must include a reference to the external style sheet file inside the <link> element, inside the head section.
<link rel = 'stylesheet', href = 'reset.css'>

2. Internal Css

An internal style sheet may be used if one single HTML page has a unique style.The internal style is defined inside the <style> element, inside the head section.

```
<style > {
        Styles…..
            }
```

3. Inline Css

An inline style may be used to apply a unique style for a single element.To use inline styles, add the style attribute to the relevant element. The style attribute can contain any CSS property.

Hello

8. **Display Property**

Inline - Displays an element as an inline element (like). Any height and width properties will have no effect

Block - Displays an element as a block element (like). It starts on a new line, and takes up the whole width**.**

Inline-block - Displays an element as an inline-level block container. The element itself is formatted as an inline element, but you can apply height and width values.

None - The element is completely removed

Visibility

Visible - Default value. The element is visible

Hidden - The element is hidden (but still takes up space).

9. Splice()

The splice() method changes the contents of an array by removing or replacing existing elements and/or adding new elements in place.

JavaScript Demo: Array.splice()

```
const months = ['Jan', 'March', 'April', 'June'];
months.splice(1, 0, 'Feb');
// inserts at index 1
console.log(months);
// expected output: Array ["Jan", "Feb", "March", "April", "June"]

months.splice(4, 1, 'May');
// replaces 1 element at index 4
console.log(months);
// expected output: Array ["Jan", "Feb", "March", "April", "May"]

```

slice()

The slice() method returns a shallow copy of a portion of an array into a new array object selected from Start to end (end not included) where start and end represent the index of items in that array. The original array will not be modified.

```
const animals = ['ant', 'bison', 'camel', 'duck', 'elephant'];

console.log(animals.slice(2));
// expected output: Array ["camel", "duck", "elephant"]

console.log(animals.slice(2, 4));
// expected output: Array ["camel", "duck"]

console.log(animals.slice(1, 5));
// expected output: Array ["bison", "camel", "duck", "elephant"]

console.log(animals.slice(-2));
// expected output: Array ["duck", "elephant"]

console.log(animals.slice(2, -1));
// expected output: Array ["camel", "duck"]

console.log(animals.slice());
// expected output: Array ["ant", "bison", "camel", "duck", "elephant"]
```

10. Spread Operator

The JavaScript spread operator (`...`) allows us to quickly copy all or part of an existing array or object into another array or object.

```
var string_1 = [1,2,3,4,5];
undefined
var string_2 = [6,7,8,9,0];
undefined
var addString = [...string_1,...string_2]
undefined
addString
▶ (10) [1, 2, 3, 4, 5, 6, 7, 8, 9, 0]
```

11. Local Storage

The localStorage object allows you to save key/value pairs in the browser.
The localStorage object stores data with no expiration date.The data is not deleted when the browser is closed, and are available for future sessions.

Syntax:
localStorage.setItem(*key*, *value*);

Session Storage

The sessionStorage object lets you store key/value pairs in the browser.The sessionStorage object stores data for only one session.(The data is deleted when the browser is closed)**.**