A Web console is a tool which is mainly used to log information associated with a web page like: *network requests, javascript, security errors, warnings, CSS etc.* It enables us to interact with a web page by executing javascript expression in the contents of the page.

n javascript, the console is an object which provides access to the browser debugging console. We can open a console in web browser by using: *Ctrl + Shift + K* for windows and *Command + Option + K* for Mac. The console object provides us with several different methods, like :

- log()
- error()
- warn()
- clear()
- time() and timeEnd()
- table()
- count()
- group() and groupEnd()
- custom console logs

console.log()

Mainly used to log(print) the output to the console. We can put any type inside the log(), be it a string, array, object, boolean etc.

```
// console.log() method
console.log('abc');
console.log(1);
console.log(true);
console.log(null);
console.log(undefined);
console.log([1, 2, 3, 4]); // array inside log
console.log({a:1, b:2, c:3}); // object inside log
```

console.error()

Used to log error message to the console. Useful in testing of code. By default the error message will be highlighted with red color.

filter_none

brightness_4

```
// console.error() method
console.error('This is a
simple error');
```

console.warn()

Used to log warning message to the console. By default the warning message will be highlighted with yellow color.

// console.warn() method
console.warn('This is a warning.');

console.clear()

Used to clear the console. The console will be cleared, in case of Chrome a simple overlayed text will be printed like : ‘Console was cleared’ while in firefox no message is returned.

filter_none

brightness_4

brightness_4

```
// console.time() and
console.timeEnd() method
console.time('abc');
 let fun =  function(){
     console.log('fun is
running');
 }
 let fun2 = function(){
     console.log('fun2 is
running..');
 }
 fun(); // calling fun();
 fun2(); // calling
fun2();
console.timeEnd('abc');
```

In the above code sample, we can see that the label is ‘abc’ which is same for both the time() and the timeEnd() method. If we increase the amount of code inside the block defined by these methods, then the time will increase. It is also worth remembering that the time returned to the console will be in milliseconds and might be different each time we refresh the page.

console.table()

This method allows us to generate a table inside a console. The input must be an array or an object which will be shown as a table.

filter_none

brightness_4

```
// console.table() method
console.table({'a':1,
'b':2});
```

Console.count()

This method is used to count the number that the function hit by this counting method.

filter_none

brightness_4

```
// console.count() method
for(let i=0;i<5;i++){
    console.count(i);
}
```

()console.group() and console.groupEnd

group() and groupEnd() methods of the console object allows us to group contents in a separate block, which will be indented. Just like the time() and the timeEnd() they also accepts label, again of same value.

filter_none

brightness_4

```
// console.group() and
console.groupEnd()
method
console.group('simple');

console.warn('warning!')
;
  console.error('error
```

Custom Console Logs

User can add Styling to the console logs in order to make logs Custom . The Syntax for it is to add the css styling as a parameter to the logs which will replace %c in the logs as shown in the example below .

filter_none

brightness_4

```
// Custom Console log
example
  const spacing =
'10px';
  const styles =
        `padding:
${spacing};
background-color:
white; color: green;
font-style:
         italic;
border: 1px solid
black; font-size:
2em;`;
  console.log('%cGeeks
for Geeks', styles);
```

Console Sidebar

Console sidebar is used to organize logs and provides clarity in debugging experience.