

What We'll Learn

- What JavaScript is & when we use it
- Basics JavaScript syntax
- Working with data types (integers, strings etc)
- Control flow (loops and conditional statements)
- The Document Object Model (DOM)
- Examples of JavaScript on web pages

2. What is JavaScript?

One of Three Core Languages

HTML - Controls the structure of your web page

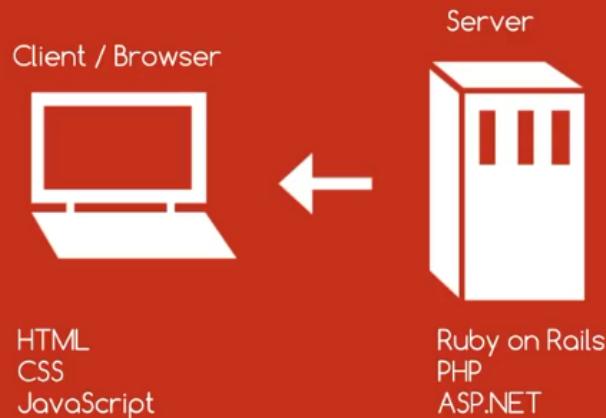
CSS - Controls the presentation/design

JavaScript - Adds behaviour and interactivity

JavaScript is a Scripting Language

- Intentionally limited
- Doesn't have the same features as other programming languages (C++, Java)
- Cannot communicate directly with a database, or file system on a computer
- However it is great at manipulating web pages

JavaScript is a Client Side Language



Do NOT Rely on JavaScript

JavaScript should ENHANCE your audience's experience only.

Do not rely on JavaScript to add core functionality to your website.

And Finally...

- JavaScript is NOT, and has NOTHING to do with JAVA
- JavaScript is just a name that was given to it while JAVA was really popular
- The official name for JavaScript is ECMAScript

3. Hello World! In JavaScript

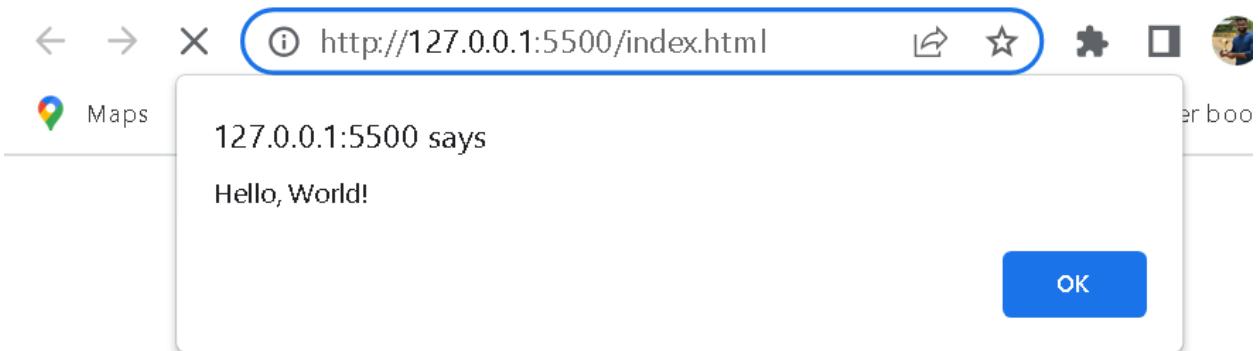
In JavaScript, the `alert()` function is used to display a dialog box with a message to the user. The message is typically a string or variable that you want to present as a notification or prompt.

```
C:\Users\gudiw>cd documents  
C:\Users\gudiw\Documents>cd tuts  
C:\Users\gudiw\Documents\tuts>mkdir JavaScript  
C:\Users\gudiw\Documents\tuts>cd JavaScript  
C:\Users\gudiw\Documents\tuts\JavaScript>echo. > index.html  
C:\Users\gudiw\Documents\tuts\JavaScript>brackets index.html  
C:\Users\gudiw\Documents\tuts\JavaScript>
```

index.html

```
<!DOCTYPE html>  
<html>  
  
<head>  
    <meta charset="utf-8" />  
    <title>Hello, World</title>  
</head>  
<body>  
    <script>  
        alert("Hello, World!");  
    </script>  
</body>  
</html>
```

O/P:



4. Where to put your JS

index.html

```
<!DOCTYPE html>
<html>

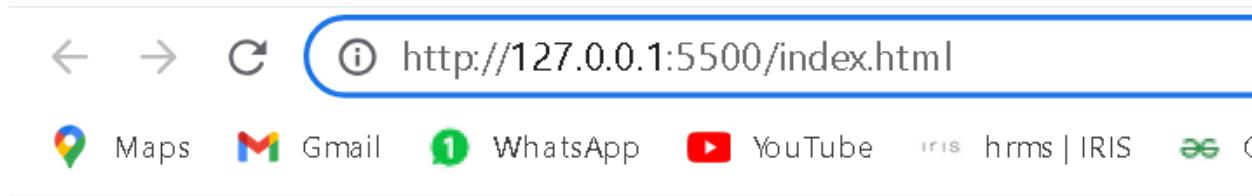
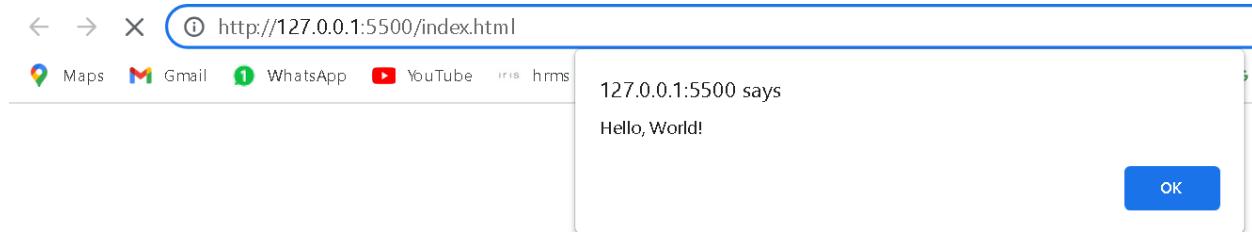
<head>
    <meta charset="utf-8">
    <title>Hello, World</title>
</head>
<body>
    <h1>JS for Beginners</h1>

    <div id="content">
        <p>learn new technologies</p>
    </div>

    <script>
        alert("Hello, World!");
    </script>

</body>
</html>
```

O/P:



index.html

```
<!DOCTYPE html>
<html>

<head>
    <meta charset="utf-8">
    <title>Hello, World</title>
</head>
<body>
    <h1>JS for Beginners</h1>

    <div id="content">
```

```
<p>learn new technologies</p>
</div>

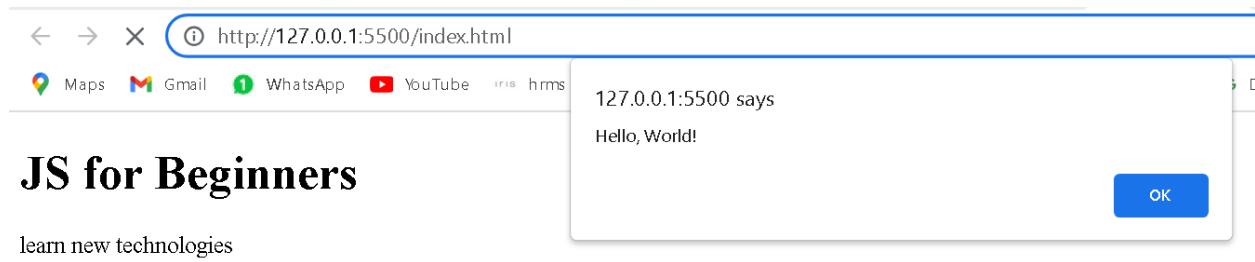
<script src="test.js"></script>

</body>
</html>
```

test.js

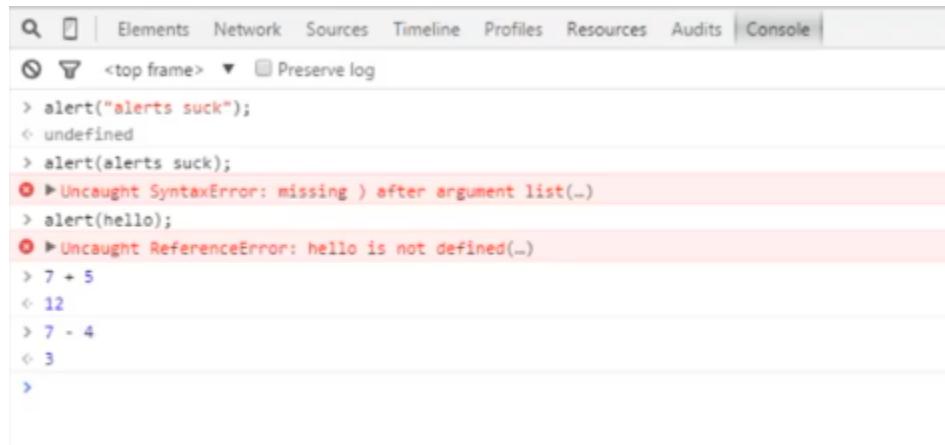
```
alert("Hello, World!");
```

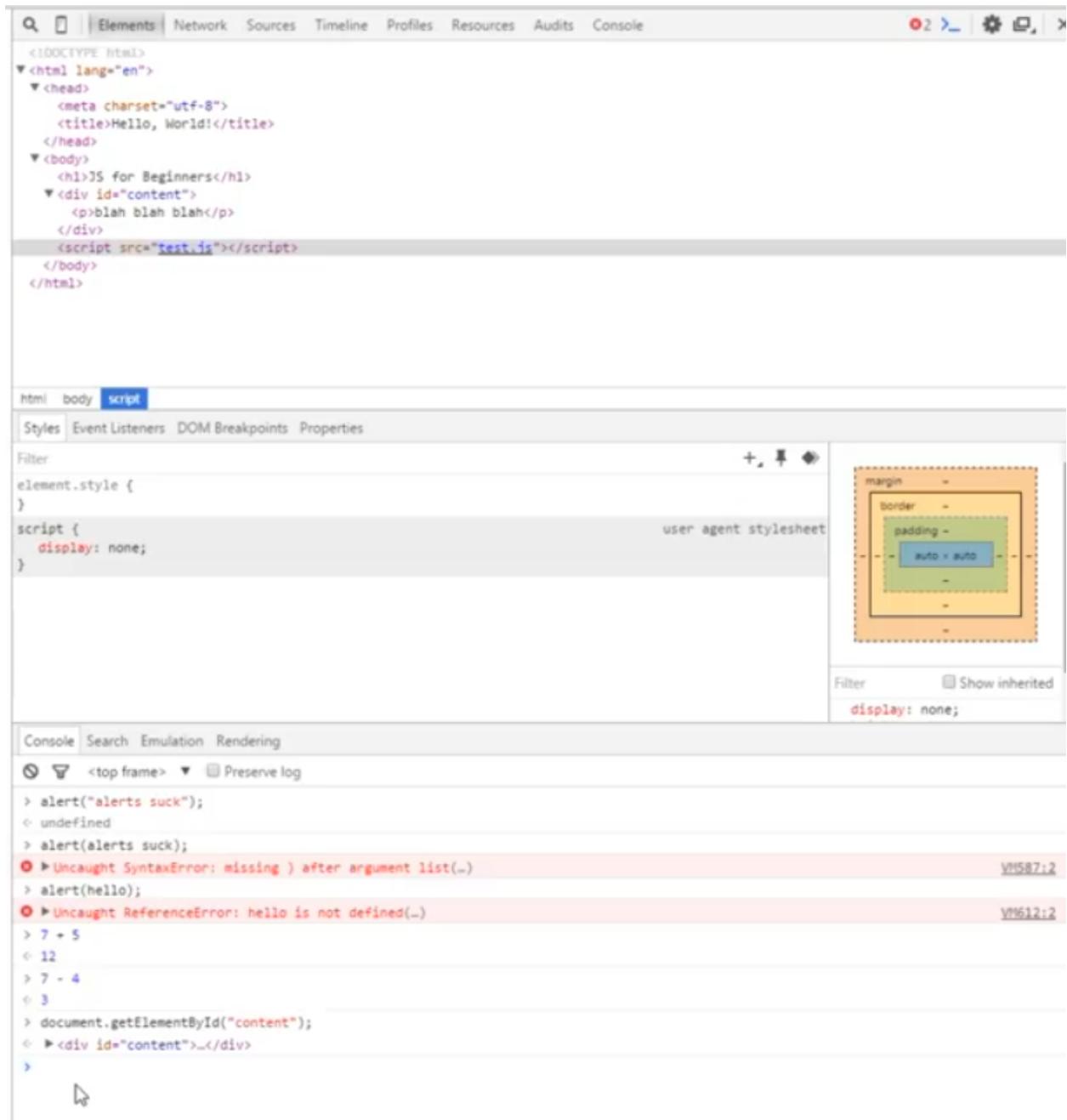
O/P:



5. Google Chrome Developer Tool

→Inspect





6. Basic JavaScript Syntax & Rules

JavaScript is a case-sensitive programming language, which means that it distinguishes between uppercase and lowercase letters. This applies to variable names, function names, object properties, and any other identifiers used in JavaScript code.

```
let message = "Hello, World!";
let Message = "Hi, there!";

console.log(message); // Output: Hello, World!
console.log(Message); // Output: Hi, there!
```

O/P:

```
PS C:\Users\gudiw\Documents\tuts\JavaScript> node test.js
Hello, World!
Hi, there!
```

In this example, `message` and `Message` are two different variables. JavaScript treats them as separate entities because of the difference in casing.

```
//If two alerts then two times press ok in browser
alert("Hello, World!");

alert("Hello, World!");
```

Summary

- JavaScript is case sensitive
- Contains many statements, all ending with a semicolon ;
- Not sensitive to whitespace or line breaks
- Write one-line comments using //
- Write multi line comments using /* */
- JavaScripts runs from top to bottom

7. JavaScript variables

The screenshot shows the 'Console' tab of a browser's developer tools. It displays a series of JavaScript commands and their results:

```
<top frame> Preset log

> var myVariableShaun;
< undefined
> myVariableShaun = 10;
< 10
> var myVar2 = "hello";
< undefined
> myVar2
< "hello"
> myVar2 = "hello again";
< "hello again"
> myVar2
< "hello again"
> myVar2 = 5;
< 5
> var customerAge = 32;
< undefined
> customerAge;
< 32
> var myVar3 = 40;
< undefined
> |
```

A cursor is visible at the bottom of the console input field.

8. Basic Mathematical Operators

The 5 Most Common Operators



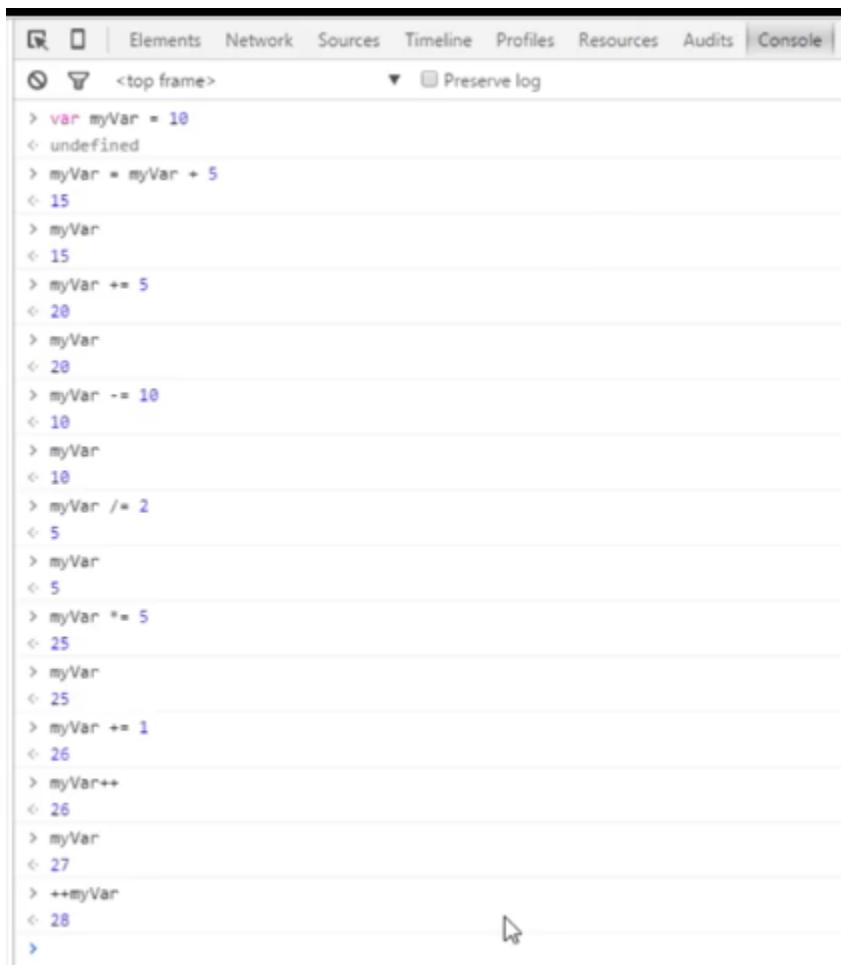
assignment



```
> var myVar = 5
< undefined
> myVar
< 5
> 5 + 5
< 10
> myVar + 10
< 15
> myVar
< 5
> myVar = myVar + 10
< 15
> myVar
< 15
> 10 - 5
< 5
> myVar - 3
< 12
> myVar = myVar - 5
< 10
> myVar
< 10
> 10 * 10
< 100
> 20 / 10
< 2
> myVar = myVar / 2
< 5
> myVar
< 5
>
```

```
> myVar
< 5
> 5 + 5
< 10
> 5 + "hello"
< "5hello"
> 100 + "hello"
< "100hello"
> "hello" + "david"
< "hellobdavid"
> "hello" + " david"
< "hello david"
> 5 * "hello"
< NaN
>
```

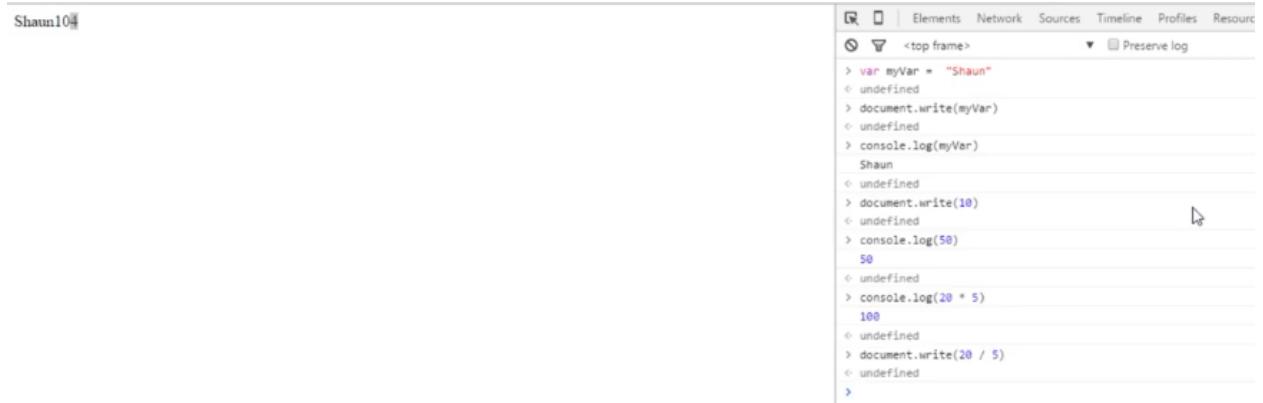
9. Math Operator Short-hand



The screenshot shows a browser's developer tools console tab labeled 'Console'. The interface includes standard navigation tabs like Elements, Network, Sources, Timeline, Profiles, Resources, and Audits. The console log displays a series of operations on a variable named 'myVar'.

```
<top frame> ▾ Preserve log
> var myVar = 10
< undefined
> myVar *= myVar + 5
< 15
> myVar
< 15
> myVar += 5
< 20
> myVar
< 20
> myVar -= 10
< 10
> myVar
< 10
> myVar /= 2
< 5
> myVar
< 5
> myVar *= 5
< 25
> myVar
< 25
> myVar += 1
< 26
> myVar++
< 26
> myVar
< 27
> ++myVar
< 28
>
```

10. Logging to the Console



The screenshot shows a browser's developer tools console window. The title bar includes tabs for Elements, Network, Sources, Timeline, Profiles, and Resources. A dropdown menu shows 'Preserve log' is selected. The console area displays the following code and its execution:

```
Shaun10
> var myVar = "Shaun"
< undefined
> document.write(myVar)
< undefined
> console.log(myVar)
Shaun
< undefined
> document.write(10)
< undefined
> console.log(50)
50
< undefined
> console.log(20 * 5)
100
< undefined
> document.write(20 / 5)
< undefined
>
```

Built in object → console, document

Method → log, write

11. Booleans in JavaScript

```
Elements Network Sources Timeline Profiles Resources Audits | Console |  
✖ <top frame> ▾  Preserve log  
> var iLikeMeat = true  
< undefined  
> iLikeMeat  
< true  
> iLikeMeat = "true"  
< "true"  
> iLikeMeat = false  
< false  
> 7 > 5  
< true  
> 7 < 5  
< false  
> 7 = 7  
✖ ► Uncaught ReferenceError: Invalid left-hand side in assignment(..)  
> 7 == 7  
< true  
> 7 == 5  
< false  
> 0  
< 0  
> Boolean(7 > 5)  
< true  
> Boolean(6)  
< true  
> Boolean(-5)  
< true  
> Boolean(0)  
< false  
> Boolean("hello")  
✖ ► Uncaught ReferenceError: Boolean is not defined(..)  
> Boolean("hello")  
< true  
> |
```

```
> Boolean("hello")  
< true  
> Boolean("")   
< false  
> |
```

12. If Statements

Control Flow & Logic

IF you pay for your train ticket...

- You won't get a fine

IF you take a crap in public...

- You'll wind up in prison

index.html

```
<!DOCTYPE html>
<html>

<head>
    <meta charset="utf-8">
    <title>Hello, World</title>
</head>
<body>

    <script src="test.js"></script>

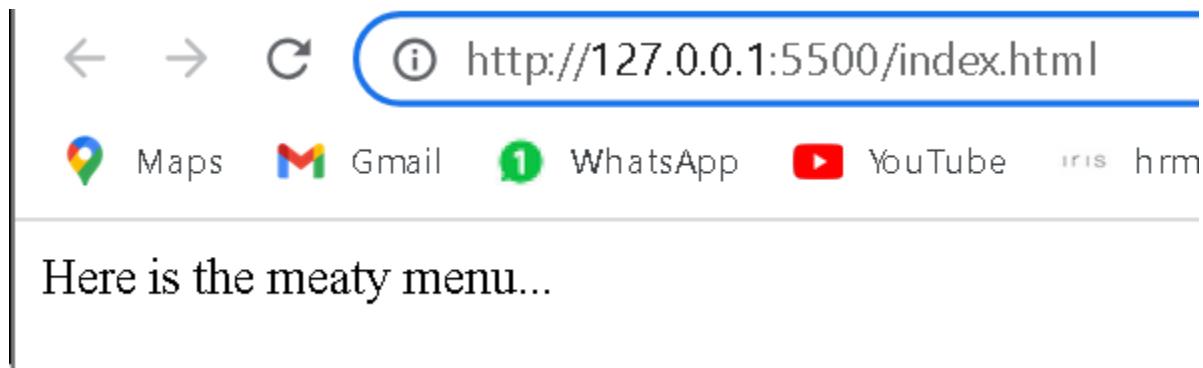
</body>
</html>
```

test.js

```
var youLikeMeat = true;

if (youLikeMeat) {
    document.write("Here is the meaty menu...") ;
}
```

O/P:

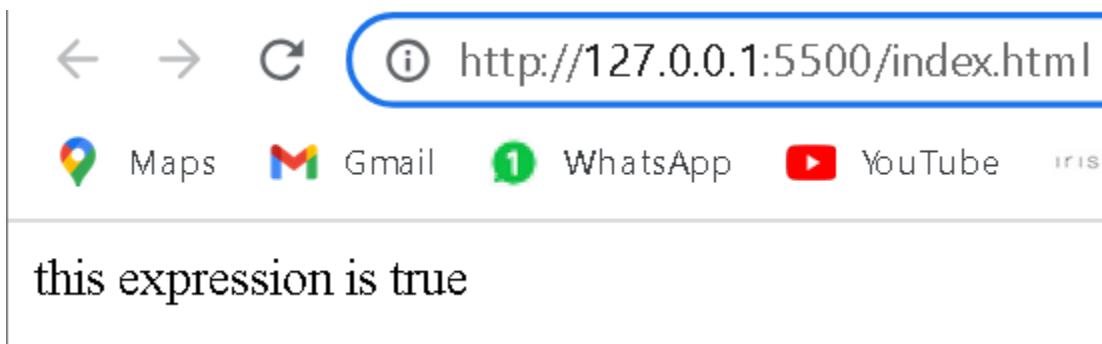


test.js

```
var youLikeMeat = true;

if (7 > 5) {
    document.write("this expression is true");
}
```

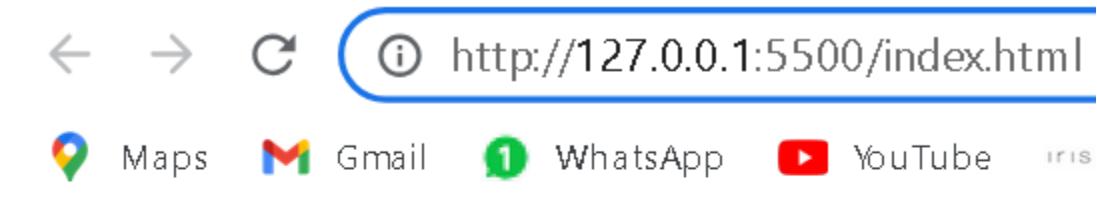
O/P:



test.js

```
var youLikeMeat = true;  
var myNum = 10;  
  
if (myNum > 10){  
    document.write("myNum is greater than 10");  
}
```

O/P:



test.js

← → ⌂ ⓘ http://127.0.0.1:5500/index.html

Maps Gmail WhatsApp YouTube iris

myNum is equal to 10

O/P:

← → ⌂ ⓘ http://127.0.0.1:5500/index.html

Maps Gmail WhatsApp YouTube iris

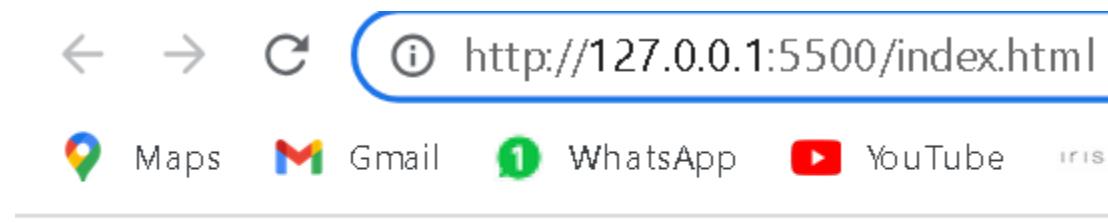
myNum is equal to 10

test.js

```
var youLikeMeat = true;
var myNum = 8;

if (myNum == 10) {
    document.write("myNum is equal to 10");
} else{
    document.write("myNum is not equal to 10");
}
```

O/P:



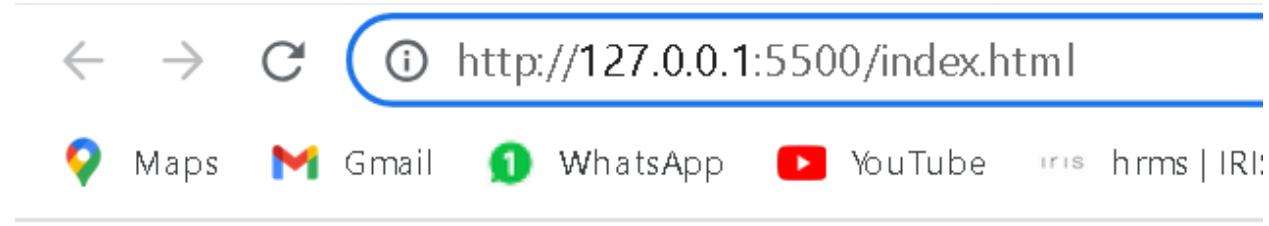
13. Else If Statements

test.js

```
var myAge = 19;

if (myAge > 30){
    document.write("you are over 30!");
} else if(myAge > 20){
    document.write("you are over 20!");
} else if(myAge > 10){
    document.write("you are over 10!");
} else{
    document.write("you are not over 10!");
}
```

O/P:



14. Comparison Operators

A screenshot of a browser's developer tools, specifically the Console tab. The console window shows a series of JavaScript comparisons involving a variable "x" set to the value 5. The tests include: x > 4 (true), x > 5 (false), x >= 5 (true), x >= 4 (true), x < 5 (false), x <= 5 (true), x == 4 (false), x == 5 (true), x === 5 (true), x == "5" (true), x == 5 (true), x === 5 (false), and x === "5" (true).

```
< top frame> ▾ □ Preserve log
> var x = 5
< undefined
> x > 4
< true
> x > 5
< false
> x >= 5
< true
> x >= 4
< true
> x < 5
< false
> x <= 5
< true
> x == 4
< false
> x == 5
< true
> x === 5
< true
> x == "5"
< "5"
> x == 5
< true
> x === 5
< false
> x === "5"
< true
>
```

```
> x === "5"
< true
> x = 5
< 5
> x != 4
< true
> x != 5
< false
> x !== 5
< false
> x = "5"
< "5"
> x !== 5
< true
> |
```

In JavaScript, the `==` operator is used for equality comparison between two values. It compares the values on both sides and performs type coercion if necessary before making the comparison.

1. Type Coercion: When comparing values using `==`, JavaScript will attempt to convert the values to a common type before making the comparison. This can lead to unexpected results in certain cases.

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

In this case, the string "5" is converted to a number before the comparison, resulting in `true`.

2. Loose Equality: The `==` operator performs loose equality comparison. It allows for type conversion and considers values equal if they are deemed equivalent after type coercion.

```
console.log(false == 0); // Output: true
```

In this case, the boolean value `false` is converted to the number `0` before the comparison, resulting in `true`.

3. **Abstract Equality Comparison Algorithm:** The `==` operator follows the Abstract Equality Comparison Algorithm defined in the JavaScript specification. It considers certain rules for comparing different types of values and determining their equality. It's important to be aware of these rules and understand the potential pitfalls of using `==`

In JavaScript, the `===` operator is the strict equality operator. It is used to compare two values for both equality of value and equality of type.

1. **Value and Type Comparison:** The `===` operator compares the values of the operands while also checking their types. It returns `true` if the values and types are identical, and `false` otherwise.

```
console.log(5 === 5);    // Output: true
console.log(5 === "5");  // Output: false
console.log(true === 1); // Output: false
```

In the first comparison, both operands are numbers with the same value, so the result is `true`. In the second comparison, the operands have different types (number and string), so the result is `false`. In the third comparison, the operands have different types (boolean and number), so the result is `false`.

2. **Strict Comparison:** The `===` operator does not perform type coercion. It requires the values being compared to have the same type to be considered equal. This can help prevent unexpected behavior that may occur due to type coercion with the loose equality operator `==`.
3. **Recommended Usage:** It's generally recommended to use the `===` operator for equality comparisons in JavaScript, as it provides more precise control over the comparison and avoids unexpected type coercion. By using strict equality, you can ensure that the values being compared are both equal in value and type.

In JavaScript, the **`!=`** operator is the inequality operator. It is used to check if two values are not equal to each other, regardless of their types.

1. **Value Comparison:** The **`!=`** operator compares the values of the operands and returns **true** if the values are not equal, and **false** if the values are equal. Type coercion is performed if necessary before the comparison.

```
console.log(5 != 5);    // Output: false
console.log(5 != "5");  // Output: false
console.log(true != 1); // Output: false
console.log(true != false); // Output: true
```

In the first three comparisons, the operands have different types, but their values are considered equal when using the **`!=`** operator. Only in the last comparison, where **true** and **false** are compared, the result is **true** because they are not equal.

2. **Type Coercion:** Similar to the loose equality operator **`==`**, the **`!=`** operator performs type coercion if necessary before the comparison. This can lead to unexpected results, especially when comparing values of different types.
3. **Inverse of Equality:** The **`!=`** operator is the inverse of the equality operator **`==`**. It provides the opposite result of the equality comparison. For example, **`x != y`** is equivalent to **`!(x == y)`**.

In JavaScript, the **`!==`** operator is the strict inequality operator. It is used to compare two values for both inequality of value and inequality of type.

1. **Value and Type Comparison:** The **`!==`** operator compares the values of the operands while also checking their types. It returns

true if the values or types are different, and **false** if the values and types are identical.

```
console.log(5 !== 5);    // Output: false
console.log(5 !== "5");  // Output: true
console.log(true !== 1); // Output: true
console.log(true !== false); // Output: true
```

In the first comparison, both operands are numbers with the same value, so the result is **false**. In the second comparison, the operands have different types (number and string), so the result is **true**. In the third comparison, the operands have different types (boolean and number), so the result is **true**. In the last comparison, both operands have the same type (boolean), but their values are different, so the result is **true**.

2. **Strict Comparison:** The `!==` operator does not perform type coercion. It requires the values being compared to have both different values and different types to be considered unequal.
3. **Recommended Usage:** It's generally recommended to use the `!==` operator for inequality comparisons in JavaScript, as it provides more precise control over the comparison and avoids unexpected type coercion. By using strict inequality, you can ensure that the values being compared are both unequal in value and type.

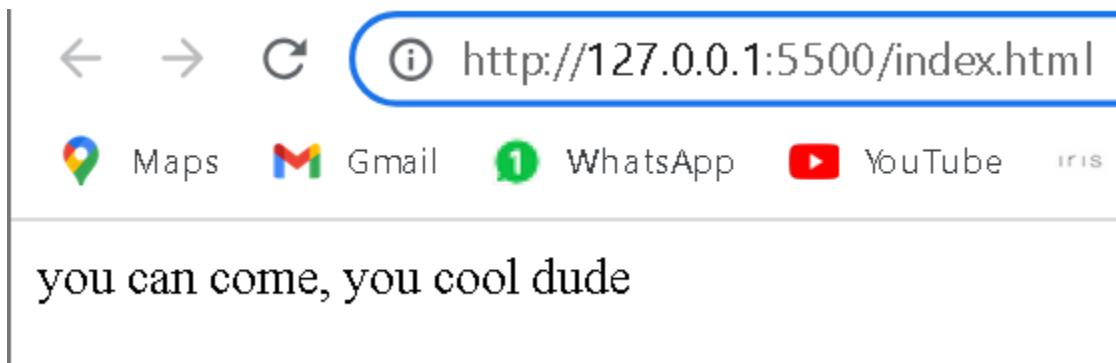
15. Logical Operators

test.js

```
var myAge = 25;

if (myAge >= 18 && myAge <= 30) {
    document.write("you can come, you cool dude");
} else{
    document.write("you aint coming");
}
```

O/P:

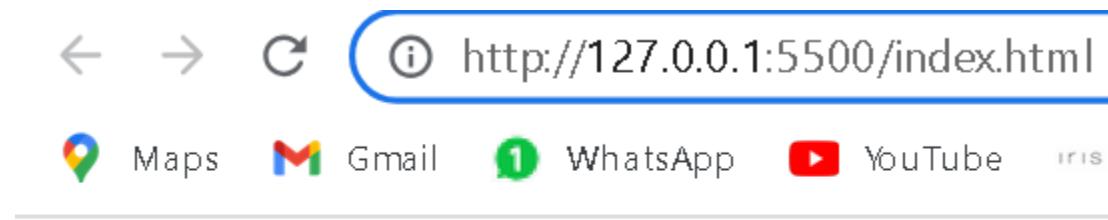


test.js

```
var myAge = 35;

if (myAge < 18 || myAge > 30){
    document.write("you aint coming");
} else{
    document.write("you can come, you cool dude");
}
```

O/P:

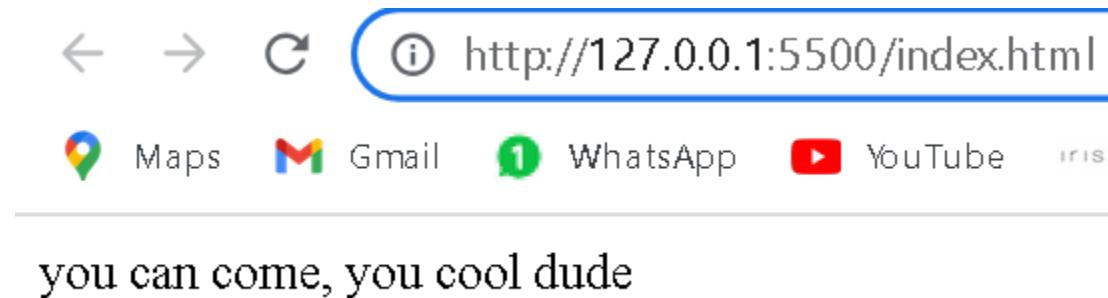


test.js

```
var myAge = 26;

if (myAge < 18 || myAge > 30 || myAge === 25) {
    document.write("you aint coming");
} else{
    document.write("you can come, you cool dude");
}
```

O/P:



16. While Loops

test.js

```
var age = 5;

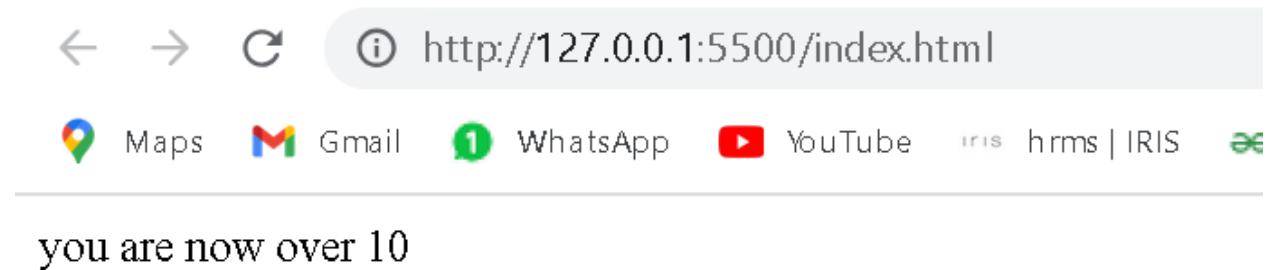
while (age < 10) {

    console.log("Your age is less than 10");
    age++;

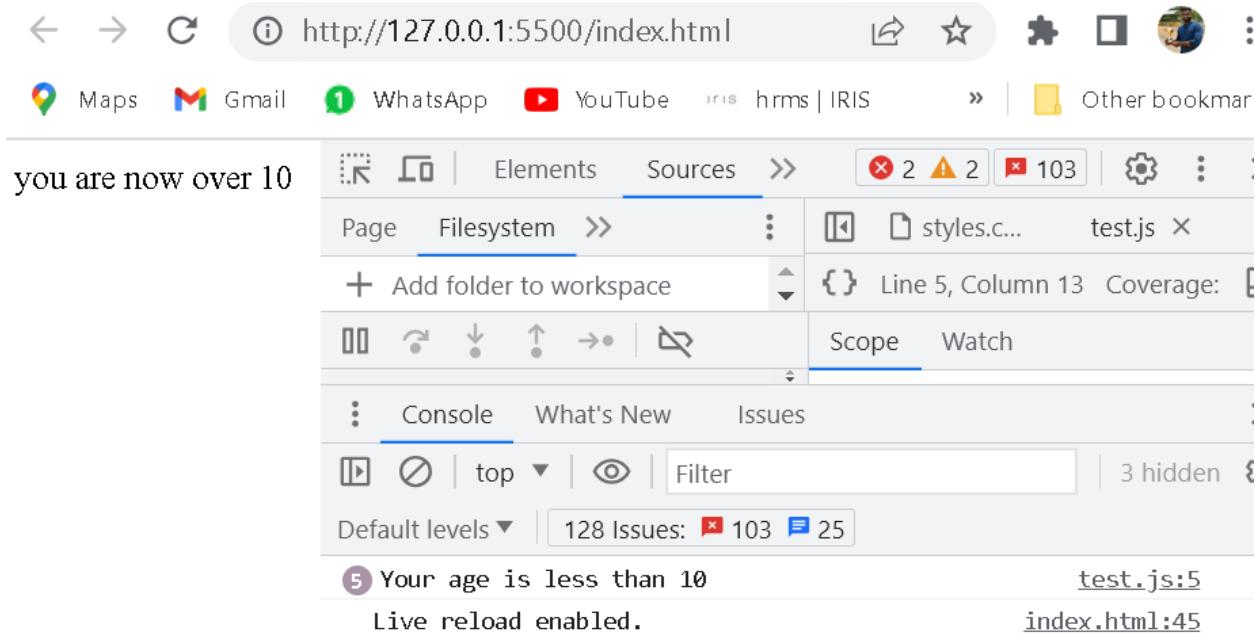
}

document.write("you are now over 10");
```

O/P:



you are now over 10



17. For Loops

index.html

```
<!DOCTYPE html>
<html>

<head>
    <meta charset="utf-8">
    <title>JavaScript for Beginners</title>
</head>
<body>
    <a href="#">Link 1</a>
    <a href="#">Link 2</a>
    <a href="#">Link 3</a>
    <a href="#">Link 4</a>
    <a href="#">Link 5</a>
    <a href="#">Link 6</a>

    <script src="test.js"></script>
```

```
</body>
</html>
```

test.js

```
for (age = 5; age < 10; age++) {

    console.log("Your age is less than 10");

}

document.write("you are now over 10");
```

O/P:

The screenshot shows a browser window with the URL `http://127.0.0.1:5500/index.html`. Below the address bar is a toolbar with various icons. The main area displays a web page with several blue underlined links labeled "Link 1", "Link 2", "Link 3", "Link 4", "Link 5", and "Link 6". Below these links, the text "you are now over 10" is visible.

Below the web content is a developer tools interface. At the top of the interface is a navigation bar with tabs: Elements, Console, Sources, and others. The Sources tab is currently active, showing a list of files: "Page", "Filesystem", "styles.css", and "test.js". A tooltip indicates "Line 5, Column 2 Coverage: r".

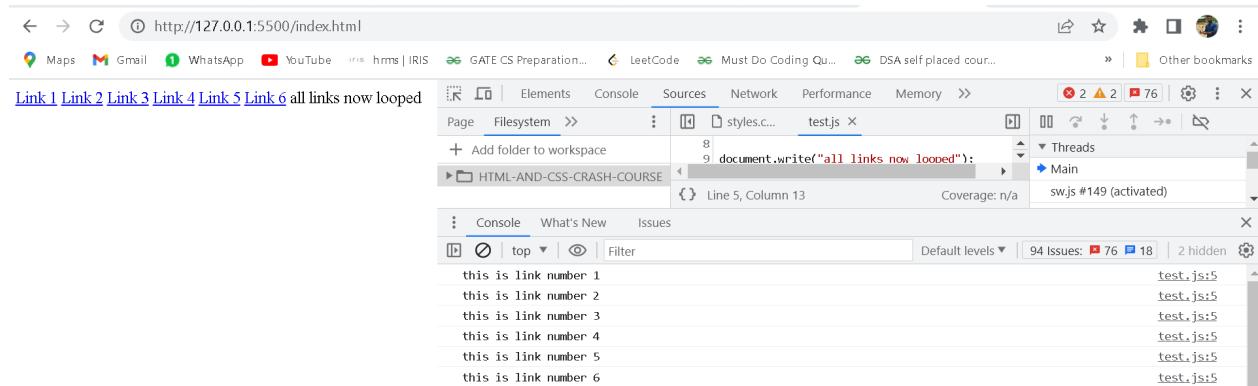
The interface includes a toolbar with icons for file operations like "New", "Open", "Save", and "Delete". Below the toolbar is a search bar with placeholder text "Scope" and "Watch".

The bottom section of the interface is the "Console" tab, which contains a list of messages. One message is highlighted in yellow: "5 Your age is less than 10" followed by the file name "test.js:3".

test.js

```
var links = document.getElementsByTagName("a") ;  
  
for (i = 1; i <= links.length; i++) {  
  
    console.log("this is link number " + i);  
  
}  
  
document.write("all links now looped");
```

O/P:



18. Break & Continue

Break

test.js

```
for (i = 0; i < 10; i++) {  
  
    console.log(i);  
  
    if(i === 7){  
        break;  
    }  
}
```

```
}

}

console.log("I have broken out of the loop");
```

O/P:

The screenshot shows the Chrome DevTools interface with the 'Sources' tab selected. In the main pane, there is a code editor window containing the following JavaScript code:

```
for (let i = 0; i < 10; i++) {
  console.log(i);
}
console.log("I have broken out of the loop");
```

Below the code editor, the 'Console' tab is active, showing the output of the code execution:

- Line 0: 0 → test.js:3
- Line 1: 1 → test.js:3
- Line 2: 2 → test.js:3
- Line 3: 3 → test.js:3
- Line 4: 4 → test.js:3
- Line 5: 5 → test.js:3
- Line 6: 6 → test.js:3
- Line 7: 7 → test.js:3
- Line 8: I have broken out of the loop → test.js:11

The 'Issues' tab shows 79 issues, with 63 errors and 16 warnings.

Continue

test.js

```
for (i = 0; i < 10; i++) {  
  
    if(i === 5 || i === 3){  
        continue;  
    }  
  
    console.log(i);  
  
    if(i === 7){  
        break;  
    }  
  
}  
  
console.log("I have broken out of the loop");
```

O/P:

The screenshot shows a browser window with the URL `http://127.0.0.1:5500/index.html`. The developer tools are open, specifically the Sources tab which is currently selected. Below it is the Filesystem tab. The code editor displays a loop that iterates from 0 to 7, printing each value to the console. At the bottom of the loop, there is a `break` statement followed by the text "I have broken out of the loop". The console panel shows the output of the loop iterations and the message from the `break` statement.

```
Link 1 Link 2 Link 3  
Link 4 Link 5 Link 6
```

```
0 test.js:7  
1 test.js:7  
2 test.js:7  
4 test.js:7  
6 test.js:7  
7 test.js:7  
I have broken out of the loop test.js:15
```

19. Practical Example Using Loops

index.html

```
<!DOCTYPE html>  
<html>  
  
<head>  
    <meta charset="utf-8">  
    <title>JavaScript for Beginners</title>  
</head>  
<body>  
  
    <a href="#">Link 0</a>  
    <a href="#">Link 1</a>  
    <a href="#">Link 2</a>
```

```
<a href="#">Link 3</a>
<a href="#">Link 4</a>
<a href="#">Link 5</a>

<script src="test.js"></script>

</body>
</html>
```

test.js

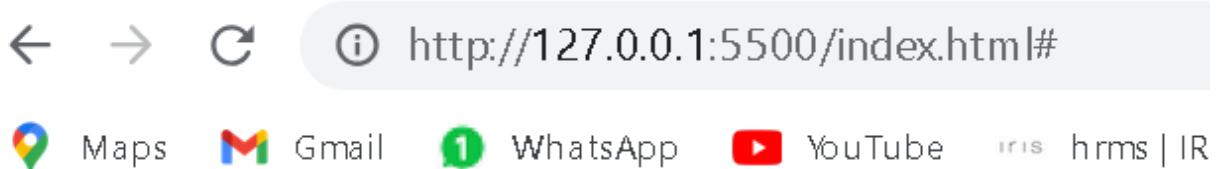
```
var links = document.getElementsByTagName("a");

for (i = 0; i < links.length; i++) {

    links[i].className = "link-" + i;

}
```

O/P:



Link 0 Link 1 Link 2 Link 3 Link 4 Link 5

20. Functions

index.html

```
<!DOCTYPE html>
<html>

<head>
    <meta charset="utf-8">
    <title>JavaScript for Beginners</title>
</head>
<body>

    <a href="#">Link 0</a>
    <a href="#">Link 1</a>
    <a href="#">Link 2</a>
    <a href="#">Link 3</a>
    <a href="#">Link 4</a>
    <a href="#">Link 5</a>

    <script src="test.js"></script>

</body>
</html>
```

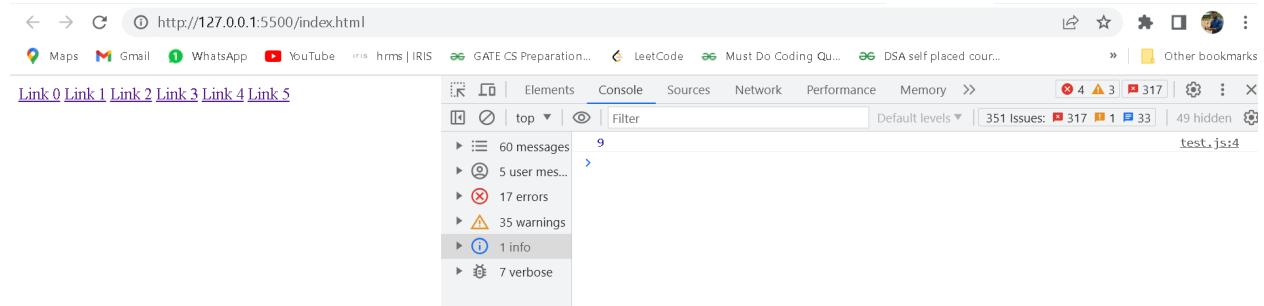
test.js

```
function getAverage(a, b) {
    var average = (a + b) / 2;
    console.log(average);

}

getAverage(7, 11);
```

O/P:

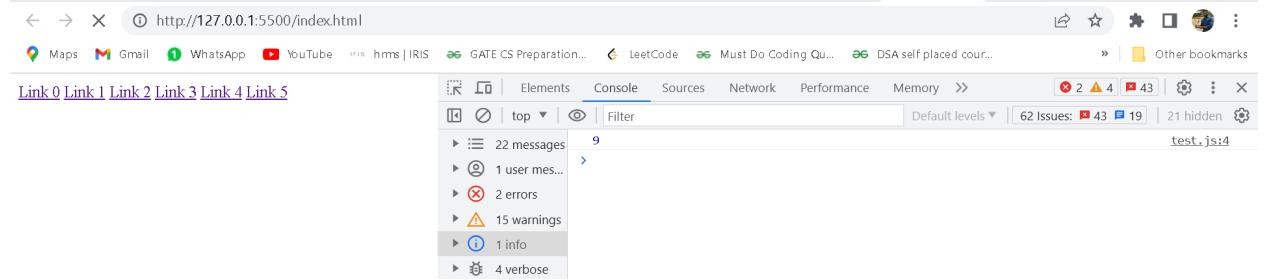


text.js

```
function getAverage(a, b) {  
  
    var average = (a + b) / 2;  
    console.log(average);  
  
}  
  
getAverage(7, 11, 13);
```

O/P:

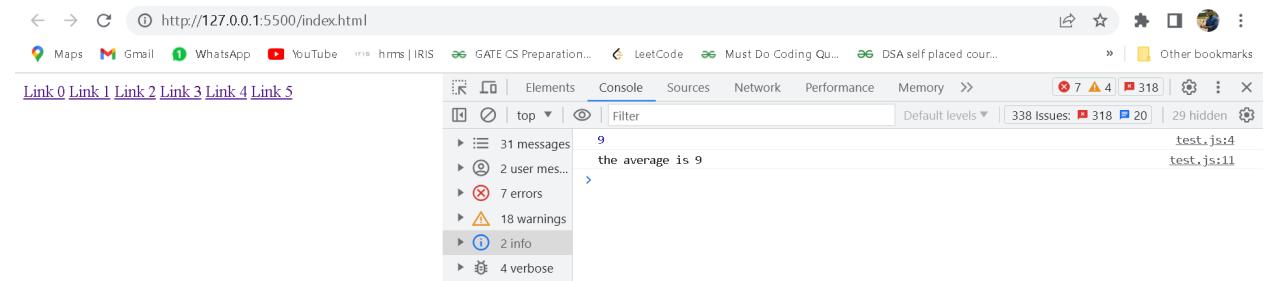
→Here the output of the first two numbers & last one is ignored.



test.js

```
function getAverage(a, b) {  
  
    var average = (a + b) / 2;  
    console.log(average);  
    return average;  
  
}  
  
  
var myResult = getAverage(7, 11);  
console.log("the average is " + myResult);
```

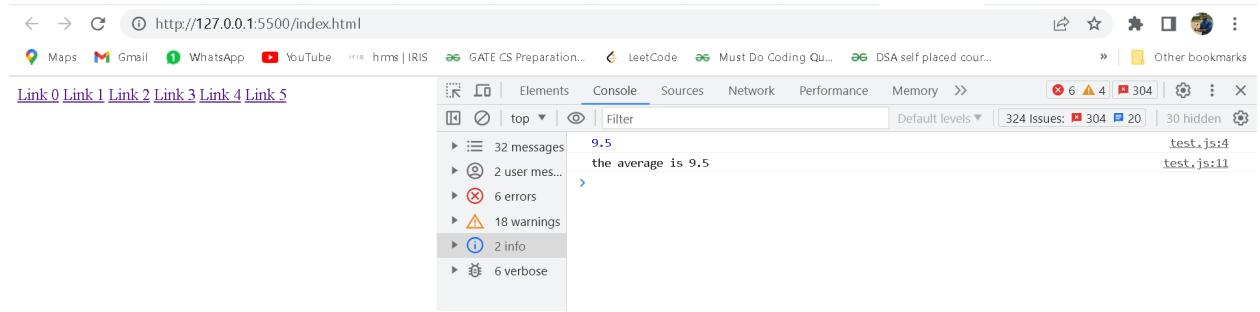
O/P:



test.js

```
function getAverage(a, b, c, d, e, f) {  
  
    var average = (a + b + c + d + e + f) / 6;  
    console.log(average);  
    return average;  
  
}  
  
  
var myResult = getAverage(7, 8, 9, 10, 11, 12);  
console.log("the average is " + myResult);
```

O/P:



A screenshot of a browser's developer tools console. The URL bar shows `http://127.0.0.1:5500/index.html`. The console tab is active, displaying the following log output:

```
9.5
the average is 9.5
>
```

The sidebar on the left lists various log levels: messages, user messages, errors, warnings, info, and verbose. The 'messages' category is currently selected, showing 32 entries. The 'info' category is also visible.

At the bottom right of the console area, there are status indicators: 6 errors, 4 warnings, 304 user messages, 20 info messages, and 30 hidden messages. Below these, two specific lines of code are highlighted: `test.js:4` and `test.js:11`.

21. Variable Scope

LOCAL vs GLOBAL

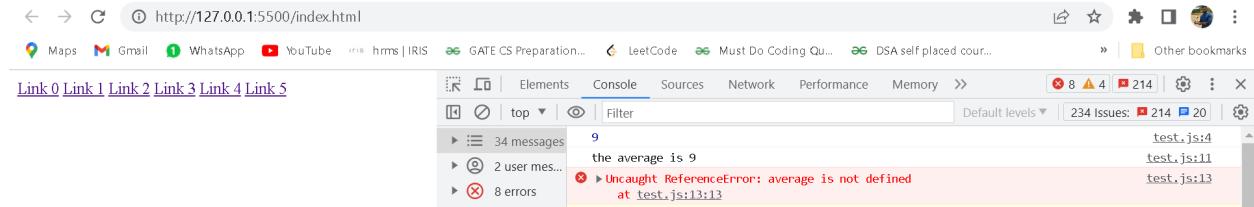
```
var foo = 20;    - Global variable
```

```
function myFunction() {  
    var bar = 10;    - Local variable  
}
```

test.js

```
function getAverage(a, b) {  
  
    var average = (a + b) / 2;          //local variable  
    console.log(average);  
    return average;  
  
}  
  
  
var myResult = getAverage(7, 11);      //global variable  
console.log("the average is " + myResult);  
  
console.log(average);
```

O/P:



test.js

```
function getAverage(a, b) {

    var average = (a + b) / 2;           //local variable
    console.log(average);
    return average;

}

var myResult = getAverage(7, 11);        //global variable

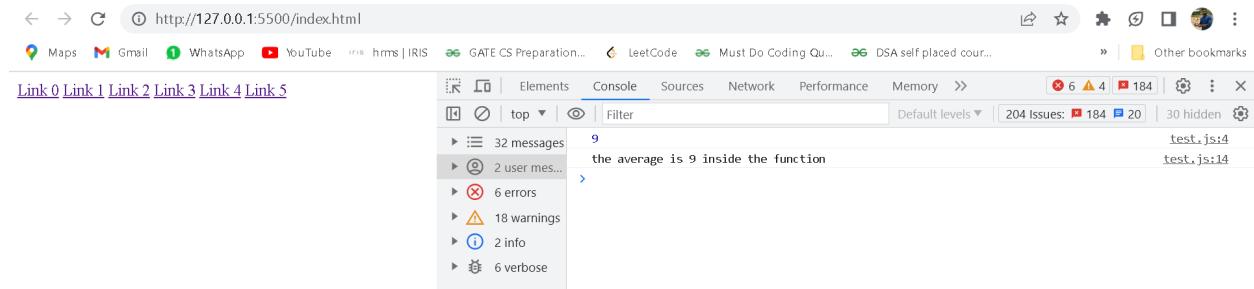
function logResult(){

    console.log("the average is " + myResult + " inside the function");

}

logResult();
```

O/P:



→ Local variable is converted into global variable

```
var average = 0;

function getAverage(a, b) {
    average = (a + b) / 2;           //global variable
    console.log(average);
    return average;
}

var myResult = getAverage(7, 11);      //global variable

function logResult(){
    console.log("the average is " + myResult + " inside the function");
}

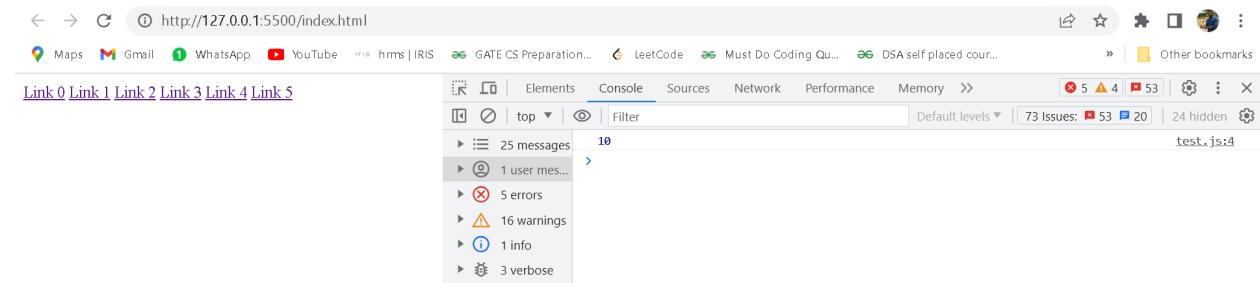
logResult();
```

22. Numbers

test.js

```
var a = 5;  
var b = 5;  
  
console.log(a + b);
```

O/P:

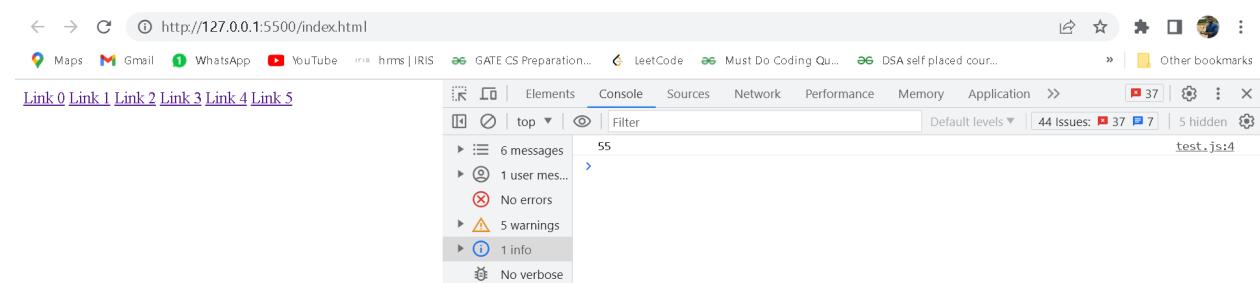


test.js

→ Concatenation

```
var a = "5";  
var b = 5;  
  
console.log(a + b);
```

O/P:



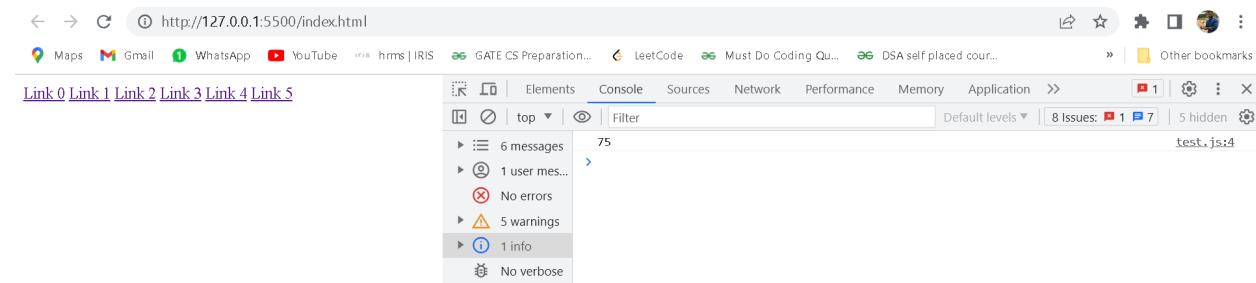
test.js

→ Concatenation

```
var a = "7";
var b = 5;

console.log(a + b);
```

O/P:



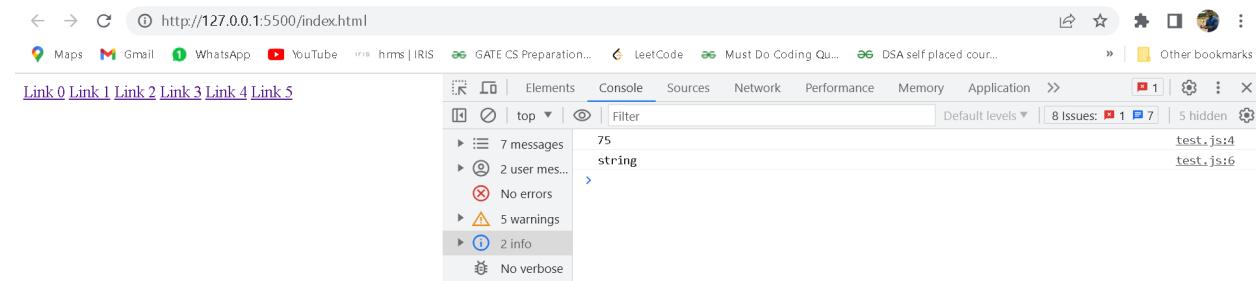
test.js

```
var a = "7";
var b = 5;

console.log(a + b);

console.log(typeof (a + b));
```

O/P:



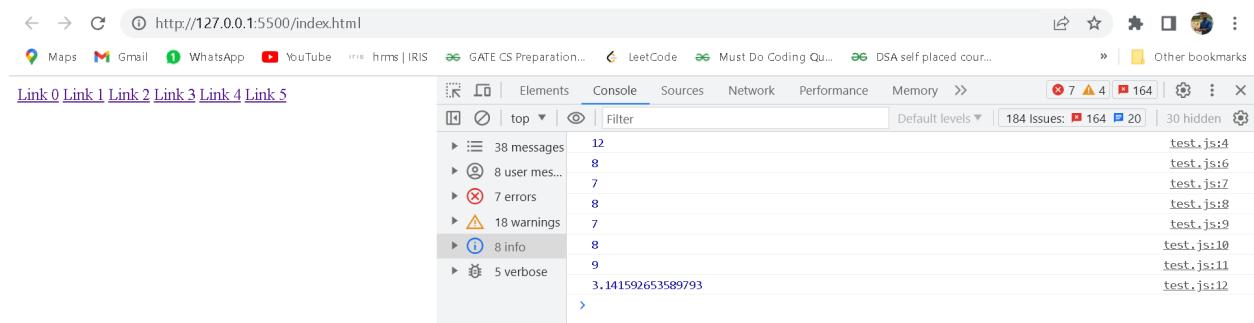
test.js

```
var a = 7;
var b = 5;

console.log(a + b);

console.log(Math.round(7.5));
console.log(Math.round(7.3));
console.log(Math.round(7.8));
console.log(Math.floor(7.9));
console.log(Math.ceil(7.2));
console.log(Math.max(7, 4, 9, 8));
console.log(Math.PI)
```

O/P:



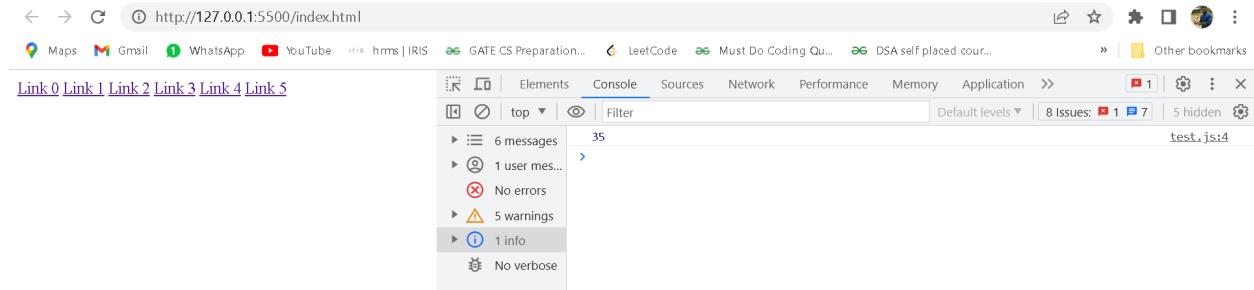
23. NaN(Not a Number)

test.js

```
var a = "7";
var b = 5;

console.log(a * b);
```

O/P:

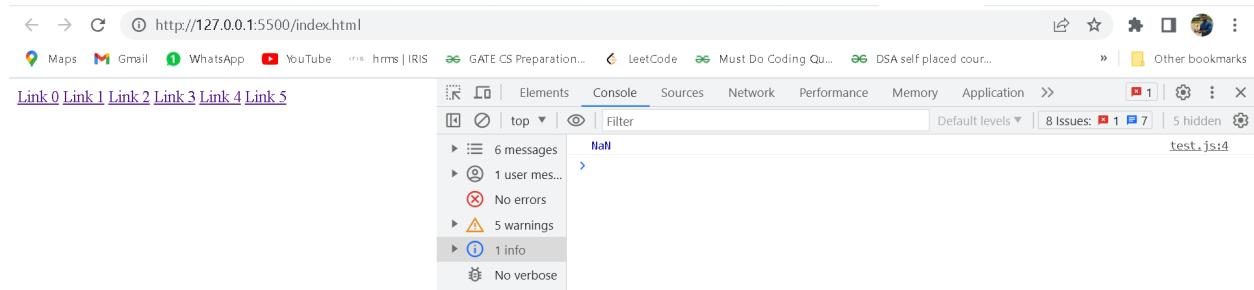


test.js

```
var a = "apple";
var b = 5;

console.log(a * b);
```

O/P:

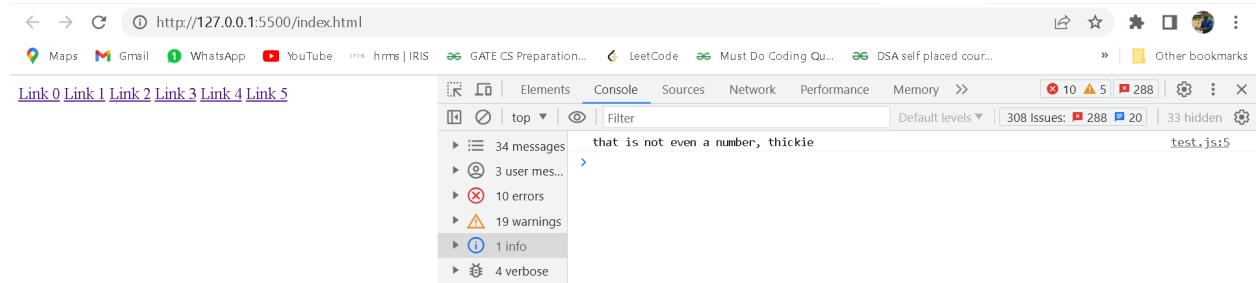


test.js

```
var a = "apple";
var b = 5;

if (isNaN(a)){
    console.log("that is not even a number, thickie");
}
else{
    console.log("meaning of life is " + (a * b));
}
```

O/P:



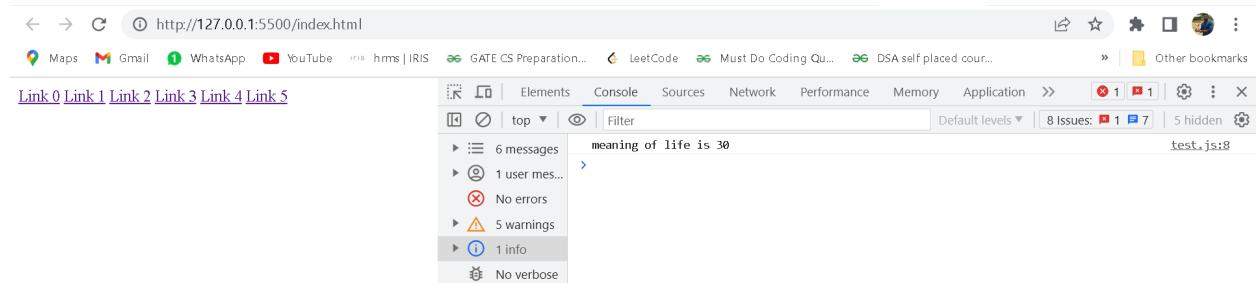
A screenshot of a browser's developer tools console tab. The URL in the address bar is `http://127.0.0.1:5500/index.html`. The console output shows the following message:
that is not even a number, thickie

test.js

```
var a = 6;
var b = 5;

if (isNaN(a)){
    console.log("that is not even a number, thickie");
}
else{
    console.log("meaning of life is " + (a * b));
}
```

O/P:



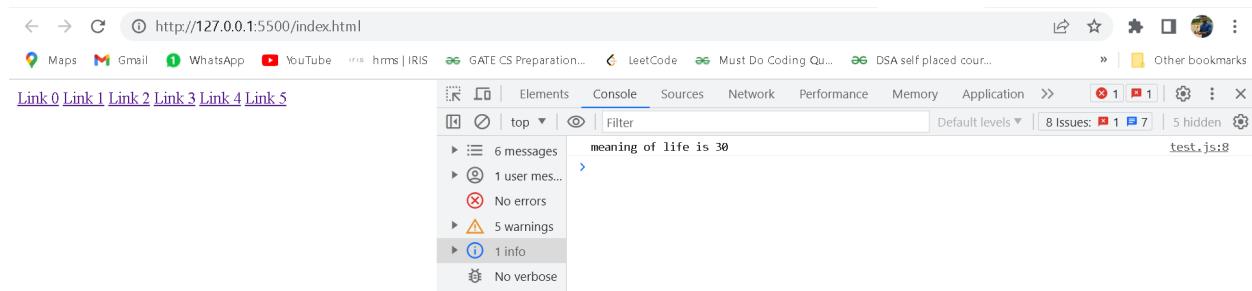
A screenshot of a browser's developer tools console tab. The URL in the address bar is `http://127.0.0.1:5500/index.html`. The console output shows the following message:
meaning of life is 30

test.js

```
var a = 6;
var b = 5;

//double negative
if (!isNaN(a)){
    console.log("meaning of life is " + (a * b));
}
```

O/P:



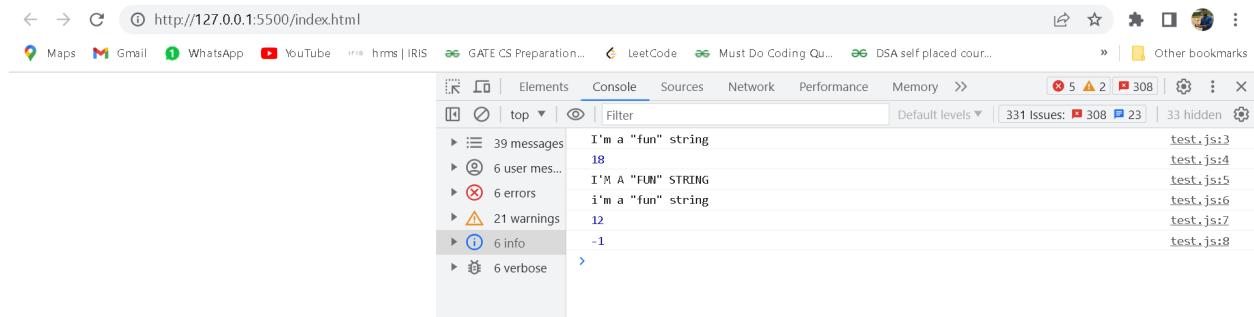
24. Strings

test.js

```
var myString = 'I\'m a "fun" string';

console.log(myString);
console.log(myString.length);
console.log(myString.toUpperCase());
console.log(myString.toLowerCase());
console.log(myString.indexOf("string"));
console.log(myString.indexOf("ninja")); //not found
```

O/P:

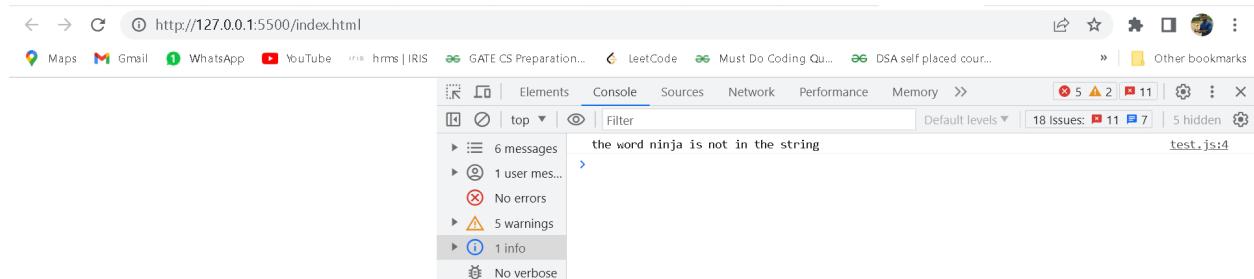


test.js

```
var myString = 'I\'m a "fun" string';

if (myString.indexOf("ninja") === -1) {
    console.log("the word ninja is not in the string");
} else {
    console.log("the word ninja starts at position " +
myString.indexOf("ninja"));
}
```

O/P:

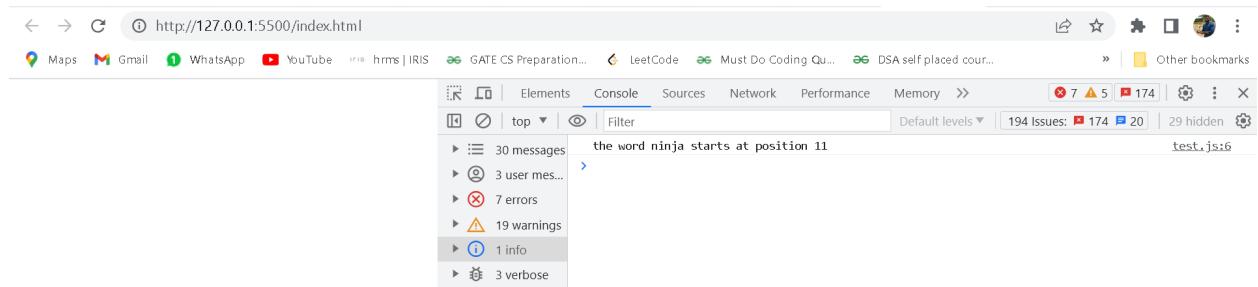


test.js

```
var myString = 'I\'m a "fun ninja" string';

if (myString.indexOf("ninja") === -1){
    console.log("the word ninja is not in the string");
} else {
    console.log("the word ninja starts at position " +
myString.indexOf("ninja"));
}
```

O/P:

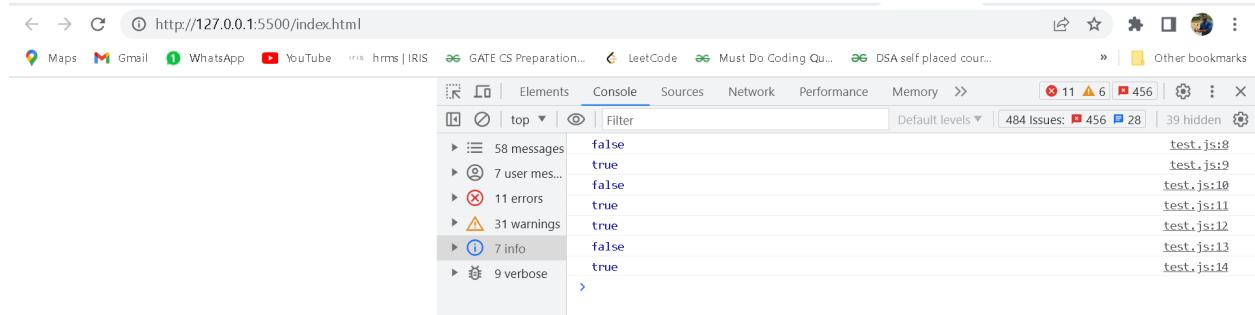


test.js

```
var string1 = "abc";
var string2 = "bcd";
var string3 = "abc";
var string4 = "ABC";
var string5 = "Bcd";
var string6 = "Abc";

console.log(string1 == string2);
console.log(string1 == string3);
console.log(string1 == string4);
console.log(string1.toLowerCase() == string4.toLowerCase());
console.log(string1 < string2);
console.log(string1 < string5);
console.log(string6 < string5);
```

O/P:

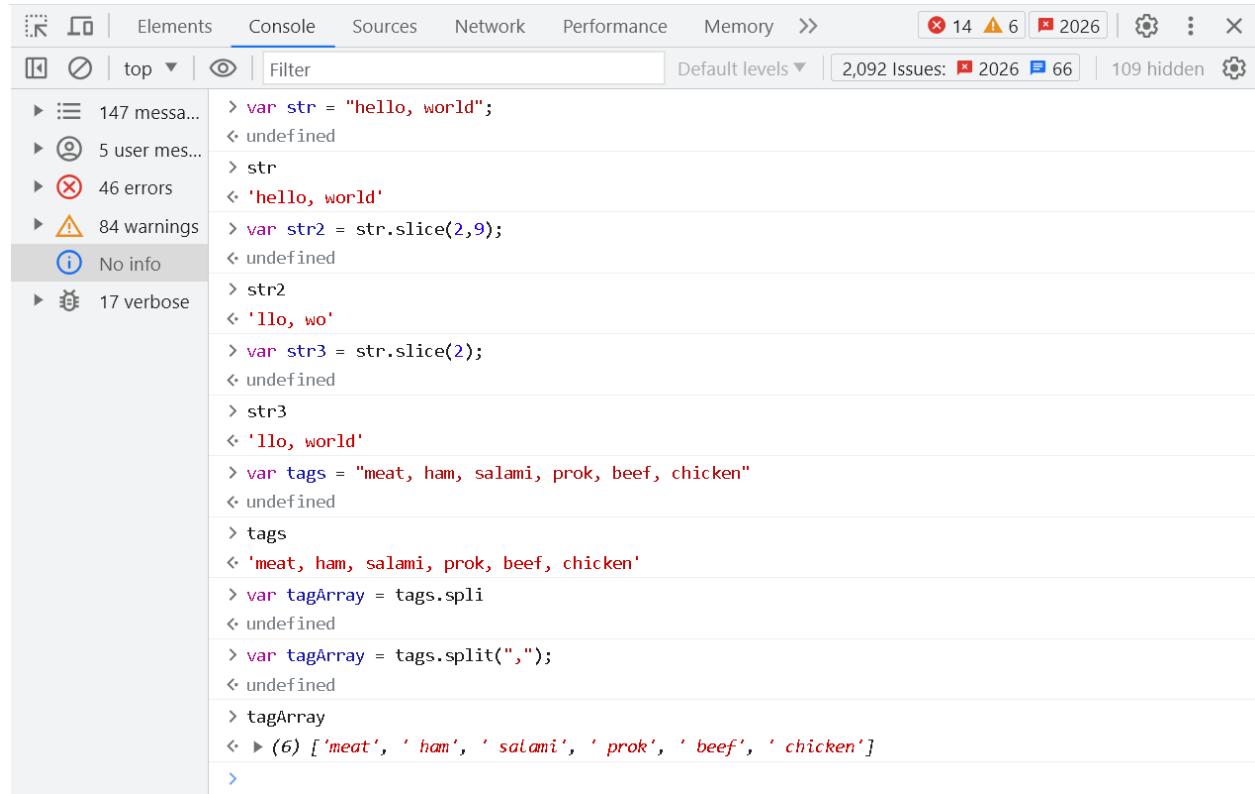


A screenshot of a browser's developer tools console tab. The URL bar shows `http://127.0.0.1:5500/index.html`. The console panel displays the following log output:

```
false
true
false
true
true
false
true
true
> 58 messages
```

The sidebar on the left shows a summary of log levels: 58 messages, 7 user messages, 11 errors, 31 warnings, 7 info, and 9 verbose.

25. Slice & Split Strings

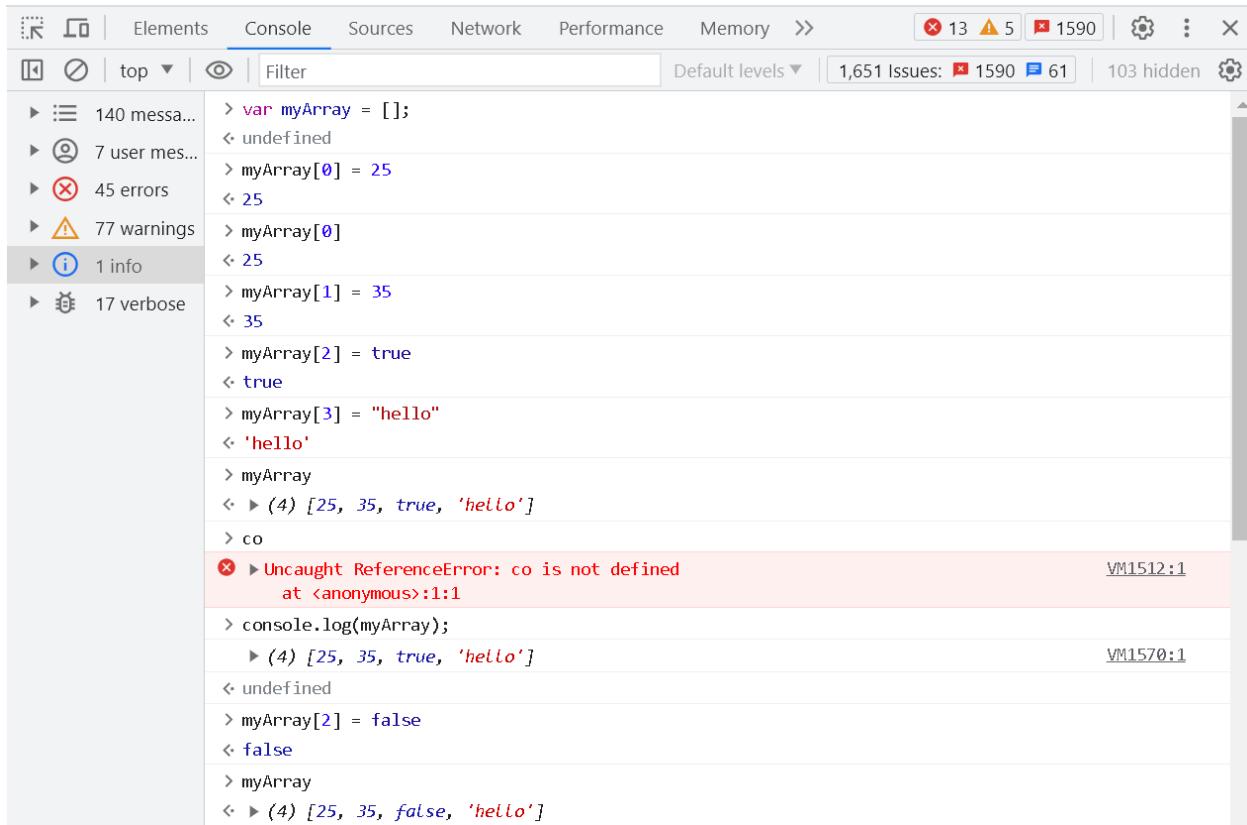


A screenshot of a browser's developer tools console tab. The URL bar shows `http://127.0.0.1:5500/index.html`. The console panel displays the following code execution and output:

```
> var str = "hello, world";
< undefined
> str
< 'hello, world'
> var str2 = str.slice(2,9);
< undefined
> str2
< 'llo, wo'
> var str3 = str.slice(2);
< undefined
> str3
< 'llo, world'
> var tags = "meat, ham, salami, prok, beef, chicken"
< undefined
> tags
< 'meat, ham, salami, prok, beef, chicken'
> var tagArray = tags.split(",");
< undefined
> var tagArray = tags.split(",");
< undefined
> tagArray
< ▶ (6) [ 'meat', 'ham', 'salami', 'prok', 'beef', 'chicken' ]
```

The sidebar on the left shows a summary of log levels: 147 messages, 5 user messages, 46 errors, 84 warnings, No info, and 17 verbose.

26. Arrays



The screenshot shows the Chrome DevTools interface with the 'Console' tab selected. The left sidebar lists various logs: 140 messages, 7 user messages, 45 errors, 77 warnings, 1 info, and 17 verbose logs. The main console area displays the following JavaScript code and its execution results:

```
> var myArray = [];
< undefined
> myArray[0] = 25
< 25
> myArray[0]
< 25
> myArray[1] = 35
< 35
> myArray[2] = true
< true
> myArray[3] = "hello"
< 'hello'
> myArray
< ▶ (4) [25, 35, true, 'hello']
> co
✖ ▶ Uncaught ReferenceError: co is not defined
at <anonymous>:1:1
VM1512:1
> console.log(myArray);
▶ (4) [25, 35, true, 'hello']
VM1570:1
< undefined
> myArray[2] = false
< false
> myArray
< ▶ (4) [25, 35, false, 'hello']
```

The error message 'Uncaught ReferenceError: co is not defined' is highlighted in red, indicating a runtime error. The stack trace points to 'VM1512:1'. The final output of the array is shown as '(4) [25, 35, false, 'hello']'.

```
> myArray
< ▶ (4) [25, 35, false, 'hello']
> var myArray2 = [10, 20, "hi", false];
< undefined
> myArray2
< ▶ (4) [10, 20, 'hi', false]
> var myArray3 = new Array()
< undefined
> var myArray4 = new Array(5)
< undefined
> myArray2.length
< 4
> myArray3.length
< 0
> myArray2.sort()
< ▶ (4) [10, 20, false, 'hi']
> myArray2.reverse()
< ▶ (4) ['hi', false, 20, 10]
>
```

27. Introduction to Objects



The screenshot shows a browser's developer tools console tab. The URL is `http://127.0.0.1:5500/index.html`. The console output is as follows:

```
> var myCar = new car()
✖ > Uncaught ReferenceError: car is not defined
      at <anonymous>:1:13
VM2103:1

> var myString = new string()
✖ > Uncaught ReferenceError: string is not defined
      at <anonymous>:1:16
VM2170:1

> var myString = new String()
< undefined
> myString = "hello";
< 'hello'
> myString.length
< 5
> myString.toLowerCase()
< 'hello'
> myString.toUpperCase()
< 'HELLO'
> var myString2 = "hi there";
< undefined
>
```

28. Creating a new JavaScript Object

test.js

```
var myArray = new Array();
myArray[0] = 8;
myArray[1] = "hello";

var myCar = new Object();
//properties of myCar object
myCar.maxSpeed = 50;
myCar.driver = "Shaun";

//methods
myCar.drive = function(){
    console.log("now driving");
};

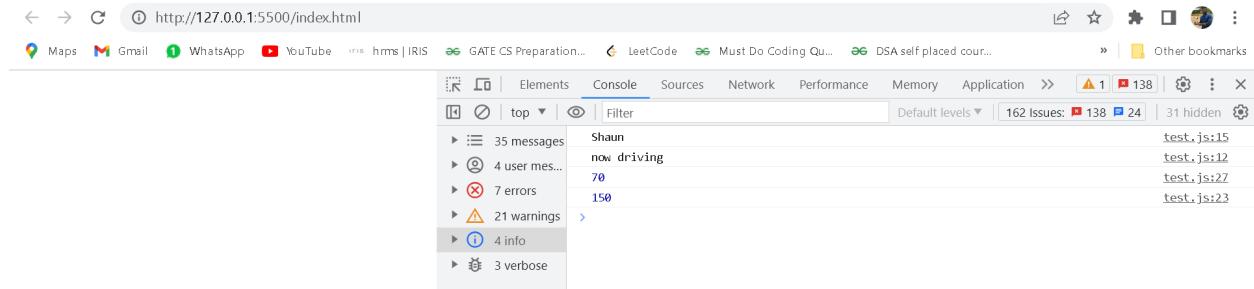
console.log(myCar.driver);

myCar.drive();

var myCar2 = {
    maxSpeed: 70,
    driver: "Net Ninja",
    drive: function(speed, time) {
        console.log(speed * time);
    }
};

console.log(myCar2.maxSpeed);
myCar2.drive(50, 3);
```

O/P:



29. This KeyWord

test.js

```
var myCar2 = {
    maxSpeed: 70,
    driver: "Net Ninja",
    drive: function(speed, time) {
        console.log(speed * time);
    },
    test: function(){
        console.log(this);
    },
    logDriver: function(){
        console.log("driver name is " + this.driver);
    }
};

myCar2.test();
myCar2.logDriver();
console.log(myCar2.maxSpeed);
myCar2.drive(50, 3);
```

O/P:

The screenshot shows a browser's developer tools console tab selected. The left sidebar lists various log levels: 54 messages, 6 user messages, 15 errors, 30 warnings, 4 info, and 5 verbose. The main pane displays the following log output:

```
▶ {maxSpeed: 70, driver: 'Net Ninja', drive: f, test: f, logDriver: f} ⓘ
  ▶ drive: f (speed, time)
  ▶ driver: "Net Ninja"
  ▶ logDriver: f ()
  ▶ maxSpeed: 70
  ▶ test: f ()
  ▶ [[Prototype]]: Object
driver name is Net Ninja
70
150
```

On the right side of the console, there are three lines of code corresponding to the log entries:

- test.js:8
- test.js:11
- test.js:19

30. Constructor Functions

test.js

```
//Example of constructors
// var myArray = new Array();
// var myString = new String();
// var myCar = new Car();

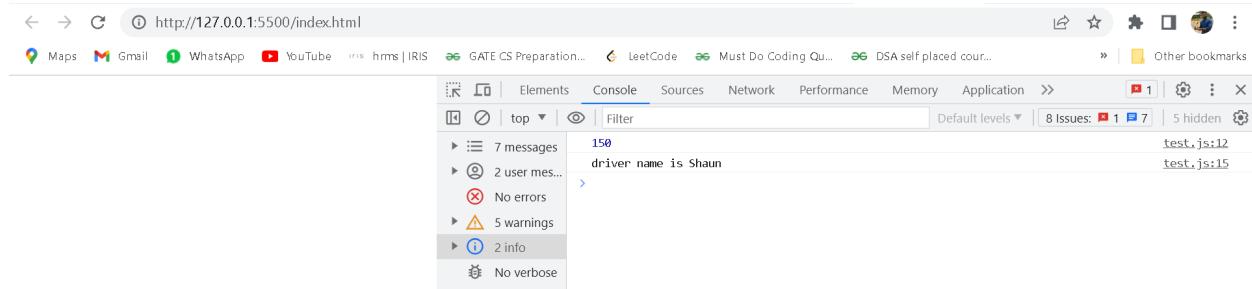
//constructor function
var Car = function(maxSpeed, driver){

    this.maxSpeed = maxSpeed;
    this.driver = driver;
    this.drive = function(speed, time){
        console.log(speed * time);
    };
    this.logDriver = function(){
        console.log("driver name is " + this.driver);
    };
}

//calling constructor function
var myCar = new Car(70, "Net Ninja");
var myCar2 = new Car(40, "Humpy dumpty");
var myCar3 = new Car(10, "Shaun");
var myCar4 = new Car(90, "James Bond");

myCar.drive(30, 5);
myCar3.logDriver();
```

O/P:



The screenshot shows the Chrome DevTools interface with the 'Console' tab selected. The console output window displays the following message:

```
driver name is Shaun
```

The message is preceded by the number '150' and a timestamp 'test.js:12'. The bottom right corner of the output window shows the file name 'test.js' and line number '15'.

31. Date Object

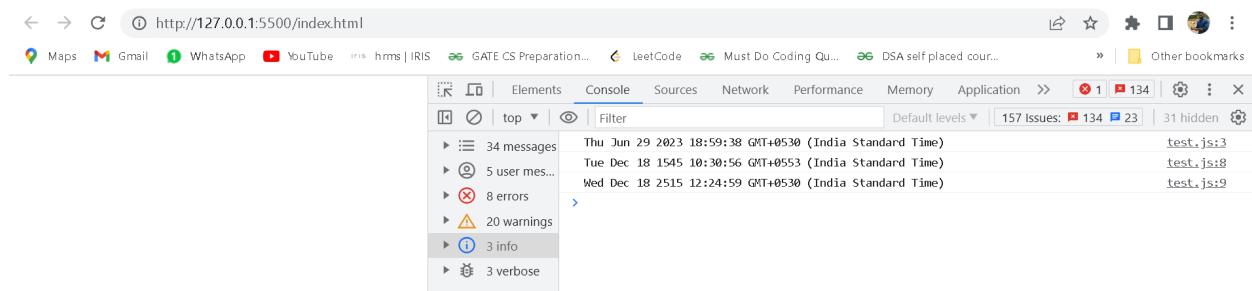
test.js

```
//Inbuilt object date
var myDate = new Date();
console.log(myDate);

var myPastDate = new Date(1545, 11, 18, 10, 30, 56);
var myFutureDate = new Date(2515, 11, 18, 12, 24, 59);

console.log(myPastDate);
console.log(myFutureDate);
```

O/P:



The screenshot shows the Chrome DevTools interface with the 'Console' tab selected. The console output window displays three log entries:

```
Thu Jun 29 2023 18:59:38 GMT+0530 (India Standard Time)
Tue Dec 18 1545 10:30:56 GMT+0533 (India Standard Time)
Wed Dec 18 2515 12:24:59 GMT+0530 (India Standard Time)
```

The first entry is preceded by the number '34' and a timestamp 'test.js:3'. The second entry is preceded by '157' and a timestamp 'test.js:8'. The third entry is preceded by '157' and a timestamp 'test.js:9'. The bottom right corner of the output window shows the file name 'test.js' and line numbers '3', '8', and '9' respectively.

test.js

```
var birthday = new Date(1996, 8, 15, 11, 15, 25);
var birthday2 = new Date(1996, 8, 15, 11, 15, 25);

//get the month of the date (0 - 11)
console.log(birthday.getMonth());

//get the full year (YYYY)
console.log(birthday.getFullYear());

//get the date of the month (1 - 31)
console.log(birthday.getDate());

//get the day of the week (0 - 6) 0 ->sun 1 -> mon
console.log(birthday.getDay());

//get the hour of the date (0 - 23)
console.log(birthday.getHours());

//get the number of milliseconds since 1st Jan 1970
console.log(birthday.getTime());

if(birthday == birthday2){ //both objects are not same

    console.log("birthdays are equal");
} else {

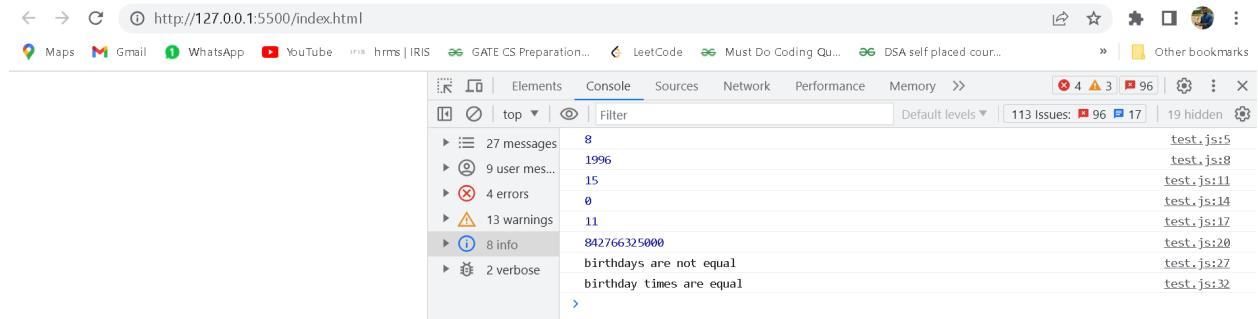
    console.log("birthdays are not equal");
}

if(birthday.getTime() == birthday2.getTime()){

    console.log("birthday times are equal");
} else {

    console.log("birthdays times are not equal");
}
```

O/P:



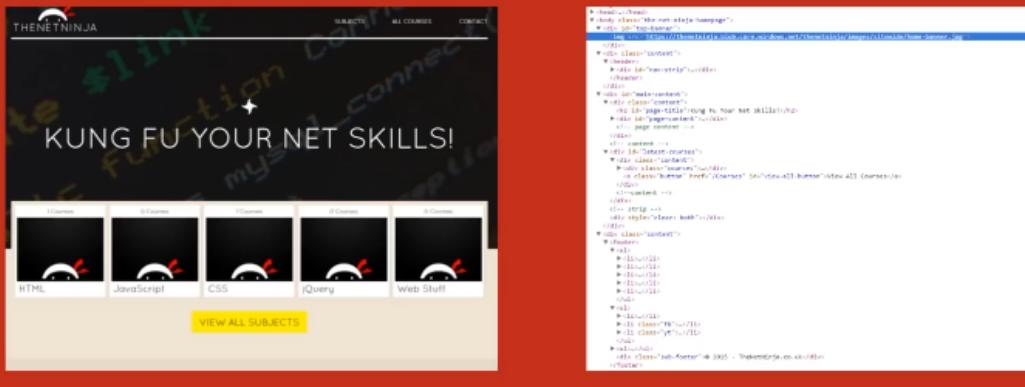
32. What is the DOM in JavaScript?

DOCUMENT OBJECT MODEL

- The DOM is an 'application programming interface'
- Use the DOM when we interact with web pages
 - Add content to a HTML document
 - Delete content from a HTML document
 - Change Content on a HTML document

DOCUMENT

- The document is just the web page

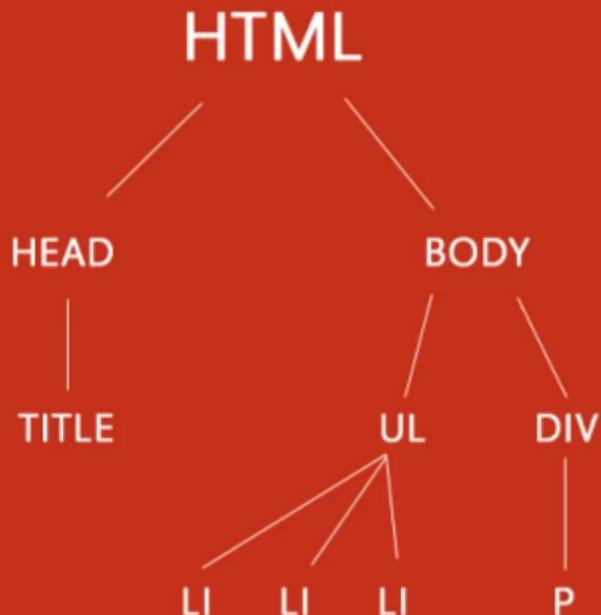


Objects Are Elements

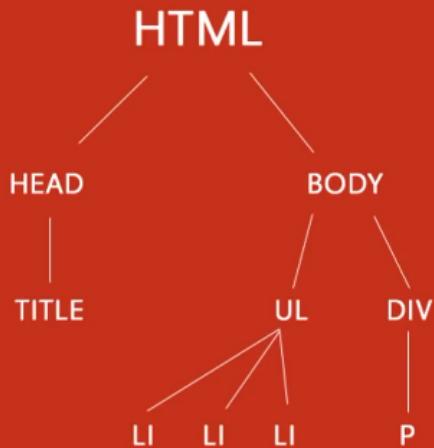
Every HTML element in the document is an object:

- `<head></head>` is an object
- `<body></body>` is an object
- `` is an object
- `<p></p>` is an object

The Model



NODES

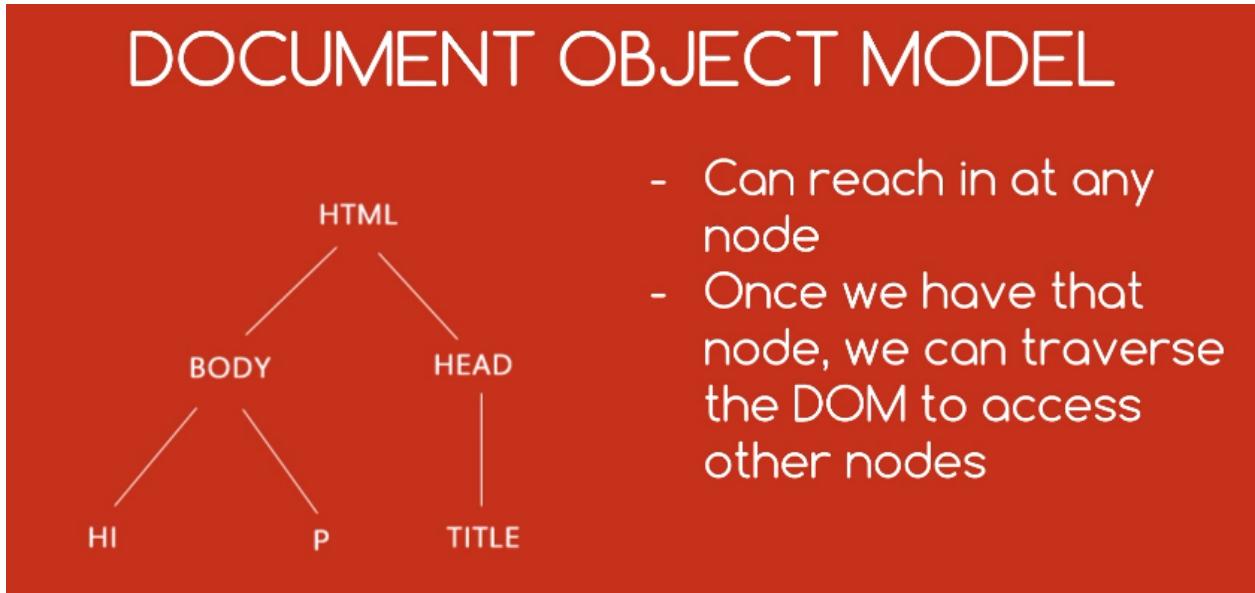


- Everything we can change in the document is a node:
 - Elements
 - Text within elements
 - HTML attributes

What Can We Do With The DOM?

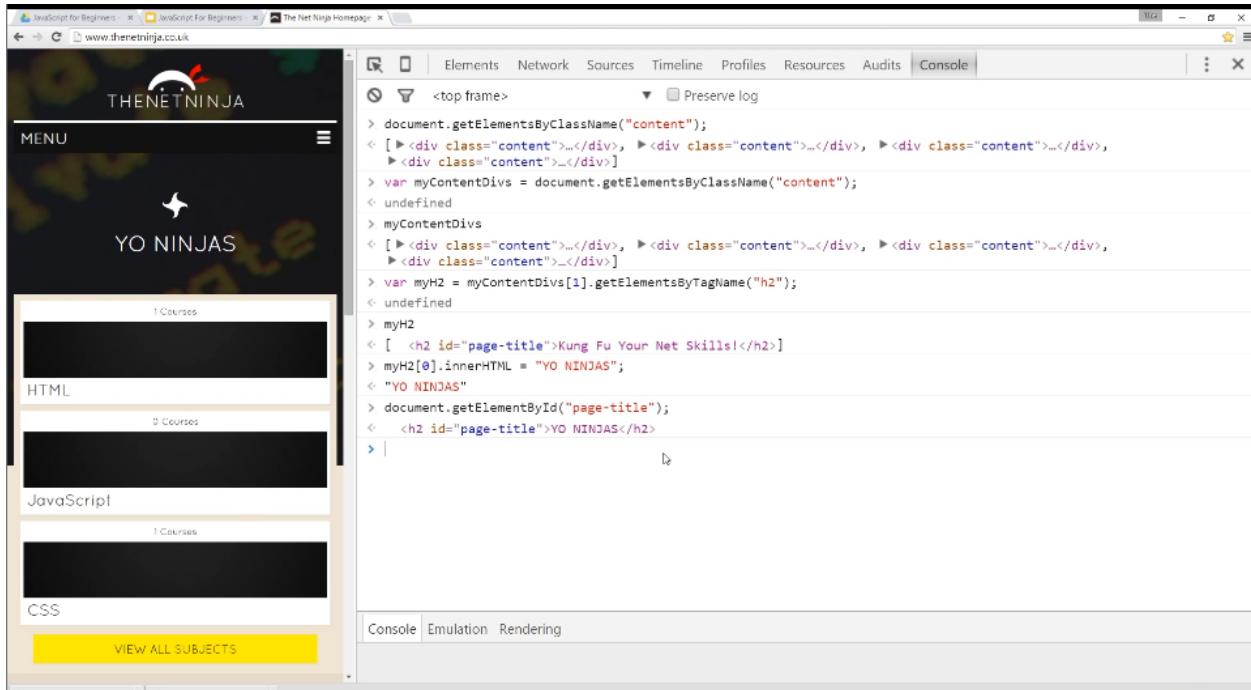
- Change the <h1> text node at the top of the page
- Change the background colour of an element node
- Animate the logo node from left to right
- Expand the height of an element node when you click on it
- PLUS MUCH MORE...

33. Traversing the DOM



```
<top frame>
document.getElementsByClassName("content");
[><div class="content">...</div>, ><div class="content">...</div>, ><div class="content">...</div>]
var myContentDivs = document.getElementsByClassName("content");
<undefined
myContentDivs
[><div class="content">...</div>, ><div class="content">...</div>, ><div class="content">...</div>]
var myH2 = myContentDivs[1].getElementsByName("h2");
<undefined
myH2
[<h2 id="page-title">Kung Fu Your Net Skills!</h2>]
myH2[0].innerHTML = "YO NINJAS"
```

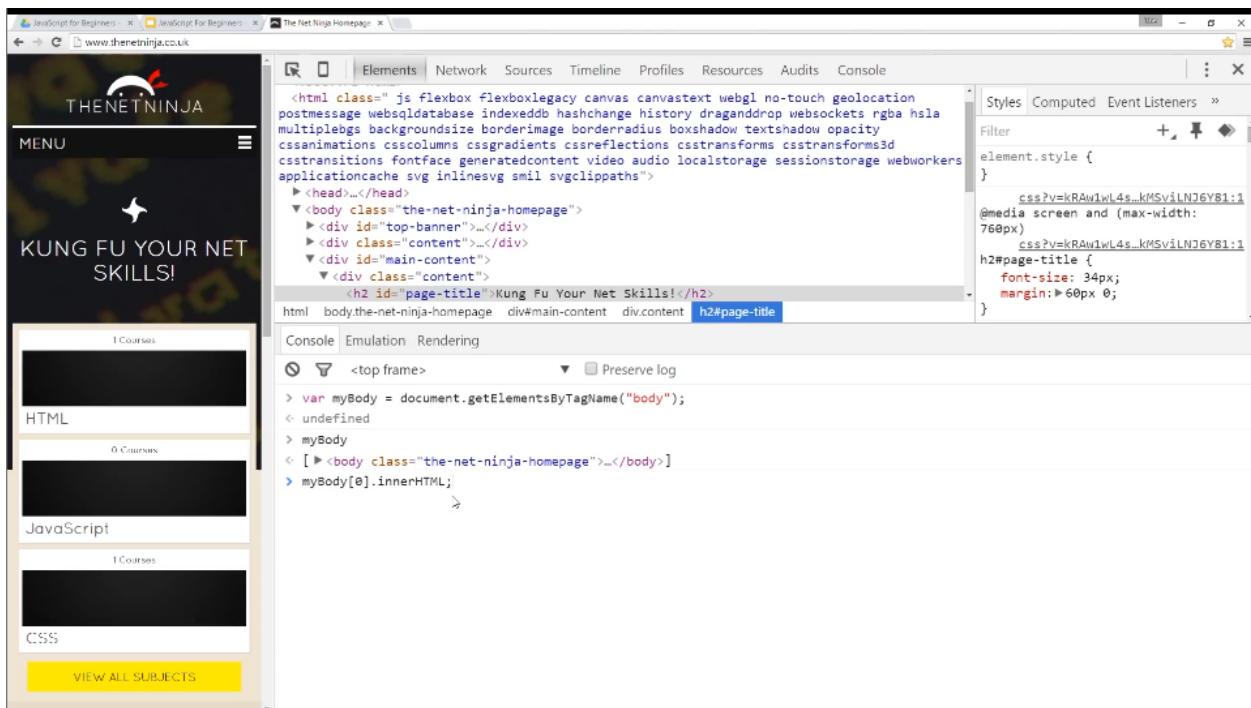
→Change the H2 title name



The screenshot shows the Net Ninja homepage with the developer tools' JavaScript Console tab selected. The console displays the following code:

```
> document.getElementsByClassName("content");
< [ ><div class="content">...</div>, ><div class="content">...</div>, ><div class="content">...</div>,
  ><div class="content">...</div> ]
> var myContentDivs = document.getElementsByClassName("content");
< undefined
> myContentDivs
< [ ><div class="content">...</div>, ><div class="content">...</div>, ><div class="content">...</div>,
  ><div class="content">...</div> ]
> var myH2 = myContentDivs[1].getElementsByTagName("h2");
< undefined
> myH2
< [ <h2 id="page-title">Kung Fu Your Net Skills!</h2>]
> myH2[0].innerHTML = "YO NINJAS";
< "YO NINJAS"
> document.getElementById("page-title");
< <h2 id="page-title">YO NINJAS</h2>
> |
```

34. Changing Page Content

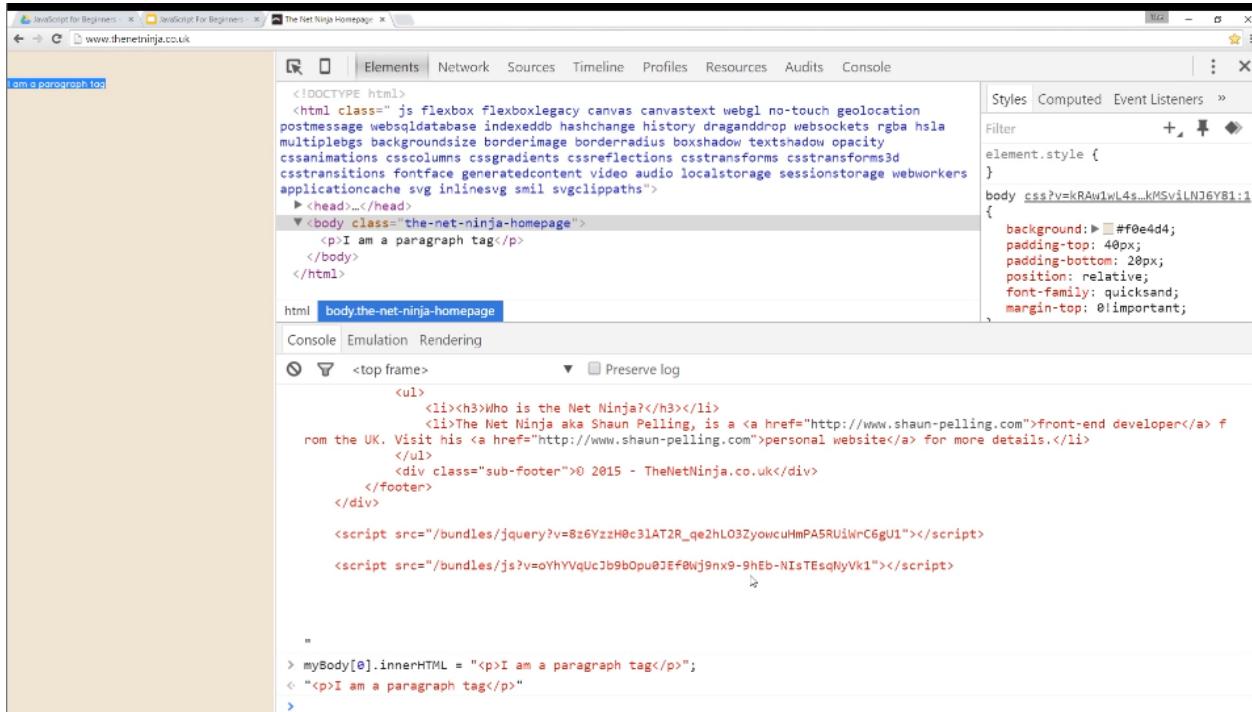


The screenshot shows the Net Ninja homepage with the developer tools' Styles panel open. The Styles panel shows the CSS rule for the h2#page-title element:

```
element.style {
}
.css?v=kRAu1wL4s_kMSviLNj6YB1:1
@media screen and (max-width: 760px)
.css?v=kRAu1wL4s_kMSviLNj6YB1:1
h2#page-title {
  font-size: 34px;
  margin: 60px 0;
}
```

The page content has been updated to "KUNG FU YOUR NET SKILLS!".

→How an element from HTML is changed/ Text on HTML Page is Changed



I am a paragraph tag

```
<!DOCTYPE html>
<html class=" js flexbox flexboxlegacy canvas canvastext webgl no-touch geolocation postmessage websqldatabase indexeddb hashchange history draganddrop websockets rgba hsla multiplebgs backgroundsize borderimage borderradius boxshadow textshadow opacity csanimations cscolumns csgradients csreflections csstransforms csstransforms3d csstransitions fontface generatedcontent video audio localstorage sessionstorage webworkers applicationcache svg inlinesvg smil svgclippaths">
  <head>...</head>
  <body class="the-net-ninja-homepage">
    <p>I am a paragraph tag</p>
  </body>
</html>
```

html body.the-net-ninja-homepage

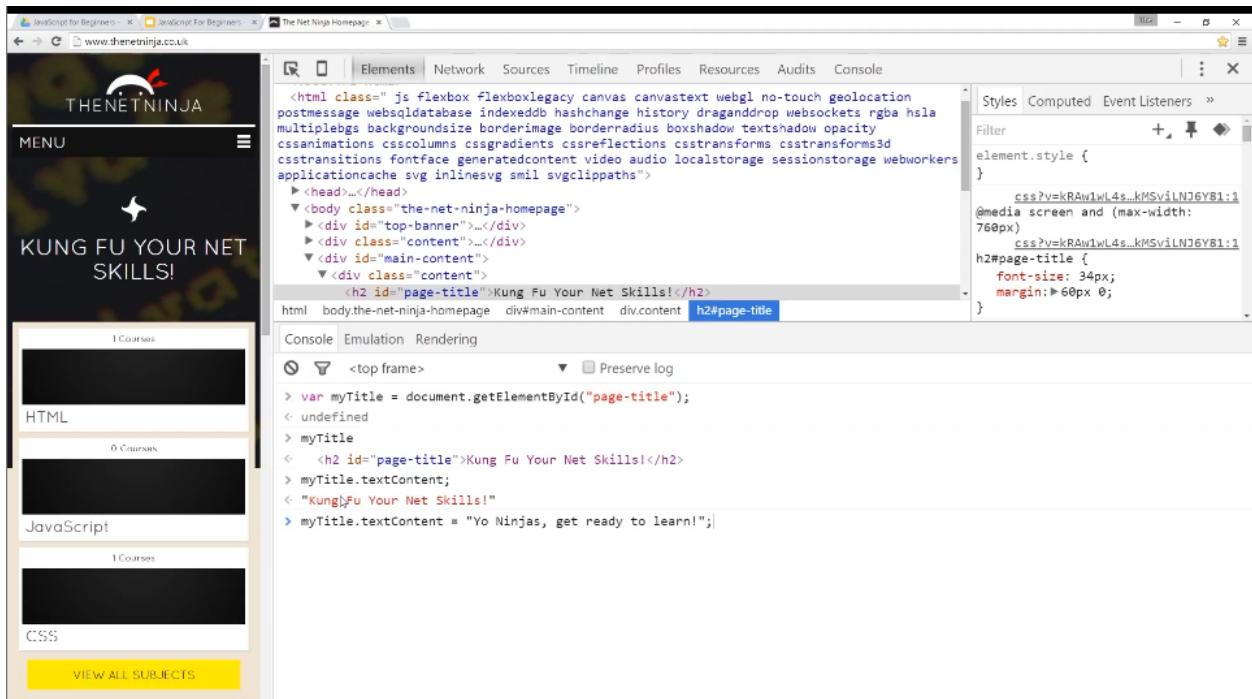
Console Emulation Rendering

```
<ul>
  <li><h3>Who is the Net Ninja?</h3></li>
  <li>The Net Ninja aka Shaun Pelling, is a <a href="http://www.shaun-pelling.com">front-end developer</a> from the UK. Visit his <a href="http://www.shaun-pelling.com">personal website</a> for more details.</li>
</ul>
<div class="sub-footer">© 2015 - TheNetNinja.co.uk</div>
</div>

<script src="/bundles/jquery?v=8z6YzzH0c3lAT2R_qe2hL03ZyowcuHmPA5RUUiWrC6gU1"></script>
<script src="/bundles/js?v=oYhVVqUcJb9bOpU0JEF0Wj9nx9-9hEb-NIsTEsqNyVk1"></script>
<script src="/bundles/js?v=kRAw1wl4s_kMSvilNJD6YB1:1"></script>
```

"

```
> myBody[0].innerHTML = "<p>I am a paragraph tag</p>";
< <p>I am a paragraph tag</p>
>
```



MENU

KUNG FU YOUR NET SKILLS!

1 Courses

HTML

0 Courses

JavaScript

1 Courses

CSS

VIEW ALL SUBJECTS

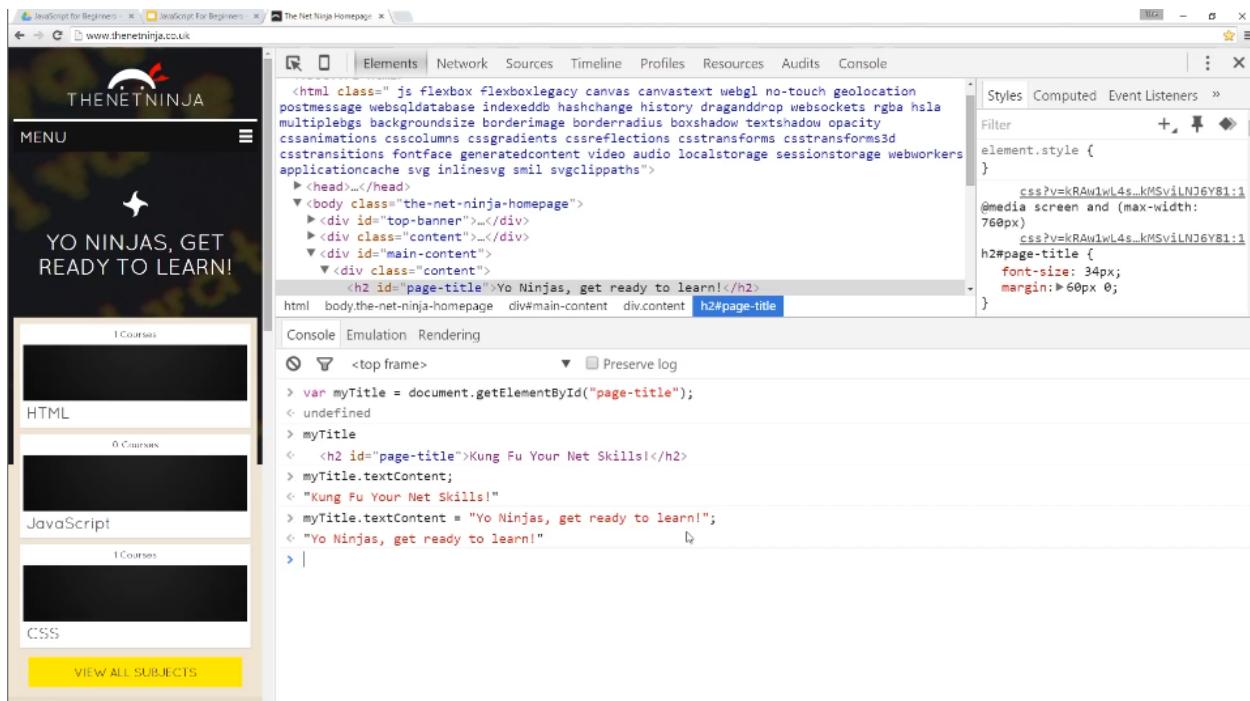
```
<html class=" js flexbox flexboxlegacy canvas canvastext webgl no-touch geolocation postmessage websqldatabase indexeddb hashchange history draganddrop websockets rgba hsla multiplebgs backgroundsize borderimage borderradius boxshadow textshadow opacity csanimations cscolumns csgradients csreflections csstransforms csstransforms3d csstransitions fontface generatedcontent video audio localstorage sessionstorage webworkers applicationcache svg inlinesvg smil svgclippaths">
  <head>...</head>
  <body class="the-net-ninja-homepage">
    <div id="top-banner">...</div>
    <div class="content">...
      <div id="main-content">
        <div class="content">
          <h2 id="page-title">Kung Fu Your Net Skills!</h2>
        </div>
      </div>
    </div>
  </body>
</html>
```

html body.the-net-ninja-homepage div#main-content div.content h2#page-title

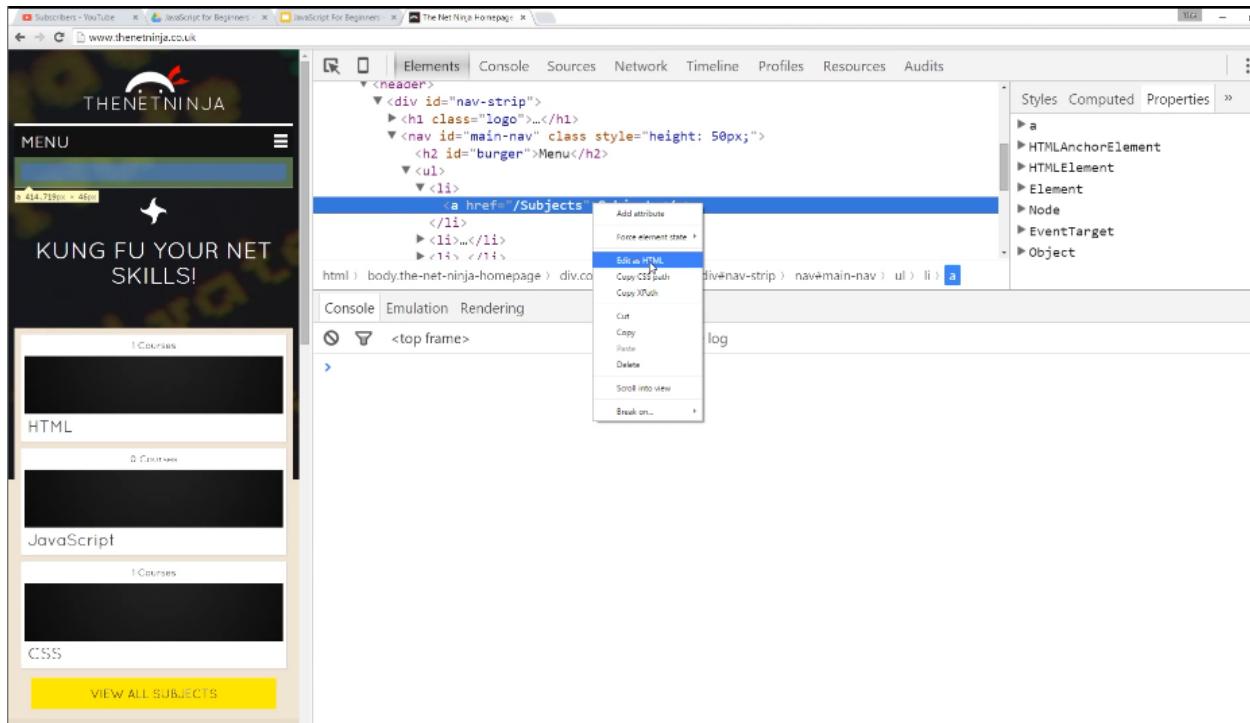
Console Emulation Rendering

```
<var myTitle = document.getElementById("page-title");>
<undefined>
> myTitle
< <h2 id="page-title">Kung Fu Your Net Skills!</h2>
> myTitle.textContent;
< "Kung Fu Your Net Skills!">
> myTitle.textContent = "Yo Ninjas, get ready to learn!"|
```

→Changed the “Title Text”



35. Changing Element Attributes



The screenshot shows a browser window with the developer tools open. The page is 'The Net Ninja Homepage'. The developer tools sidebar shows the following DOM structure:

```

<header>
  <div id="nav-strip">
    <h1 class="logo">...</h1>
    <nav id="main-nav" class="height: 50px;">
      <h2 id="burger">Menu</h2>
      <ul>
        <li>
          <a href="/Subjects" id="test" class="ninja">Subjects</a>
        </li>
        <li>...</li>
        <li>...</li>
      </ul>
    </nav>
  </div>

```

In the console tab, the following JavaScript code is running:

```

var link = document.getElementById("test");
link.setAttribute("class", "pie");

```

→Changing the “class” & adding the “alt” attribute

The screenshot shows the same browser setup as the previous one, but the developer tools sidebar now shows the updated state of the element:

```

<li>
  <a href="/Subjects" id="test" class="pie" alt="hello">Subjects</a>

```

The console tab shows the completed JavaScript code:

```

var link = document.getElementById("test");
link.setAttribute("class", "pie");
link.setAttribute("alt", "hello");

```

The screenshot shows the Chrome DevTools Elements tab open over a browser window displaying the homepage of 'THE NET NINJA'. The page has a dark theme with a header containing a logo and navigation links. Below the header is a main content area with three sections: 'HTML', 'JavaScript', and 'CSS'. On the left, there's a sidebar with a 'VIEW ALL SUBJECTS' button. The DevTools sidebar on the right shows the current element selected is a link with the ID 'test' and the class 'ninja'. The DevTools console below shows the following code:

```

< top frame>
<a href="/Subjects" id="test" class="ninja">Subjects</a>
link.getAttribute("href");
"/Subjects"
link.getAttribute("class");
"ninja"
link.setAttribute("class", "pie");
undefined
link.setAttribute("alt", "hello");
undefined
link.className;
"pie"
link.className = "ninja";
"ninja"
link.href;
"http://www.thenetninja.co.uk/Subjects"
link.style
CSSStyleDeclaration {alignContent: "", alignItems: "", alignSelf: "", alignmentBaseline: "", all: ""}
|

```

36. Changing CSS Styles

The screenshot shows the Chrome DevTools Elements tab open over a browser window displaying the homepage of 'THE NET NINJA'. The page has a dark theme with a header containing a logo and navigation links. Below the header is a main content area with three sections: 'HTML', 'JavaScript', and 'CSS'. On the left, there's a sidebar with a 'VIEW ALL SUBJECTS' button. The DevTools sidebar on the right shows the current element selected is a h2 element with the ID 'page-title'. The DevTools sidebar on the right shows the following computed styles for the h2 element:

```

element.style {
}
css?v=kRAw1wl4s.kMSviLNJ6YB1
@media screen and (max-width: 760px)
css?v=kRAw1wl4s.kMSviLNJ6YB1
h2#page-title {
    font-size: 34px;
}

```

The DevTools console below shows the following code:

```

< top frame>
<a href="/Subjects" id="test" class="ninja">Subjects</a>
link.getAttribute("href");
"/Subjects"
link.getAttribute("class");
"ninja"
link.setAttribute("class", "pie");
undefined
link.setAttribute("alt", "hello");
undefined
link.className;
"pie"
link.className = "ninja";
"ninja"
link.href;
"http://www.thenetninja.co.uk/Subjects"
link.style
CSSStyleDeclaration {alignContent: "", alignItems: "", alignSelf: "", alignmentBaseline: "", all: ""}
|

```

→Updating Style

The screenshot shows the 'The Net Ninja' homepage. The page features a dark header with the 'THE NET NINJA' logo and a 'MENU' button. Below the header is a banner with the text 'KUNG FU YOUR NET SKILLS!'. A sidebar on the left lists '1 Courses' for 'HTML', 'JavaScript', and 'CSS'. At the bottom is a yellow 'VIEW ALL SUBJECTS' button. The developer tools' Elements tab is active, showing the DOM structure. In the Styles panel, a CSS rule for the 'h2#page-title' element is being edited. The original rule is 'position: relative;'. A new rule is being added: '@media screen and (max-width: 760px) { h2#page-title { left: 10px; } }'. The browser's console tab shows the JavaScript code used to apply these styles.

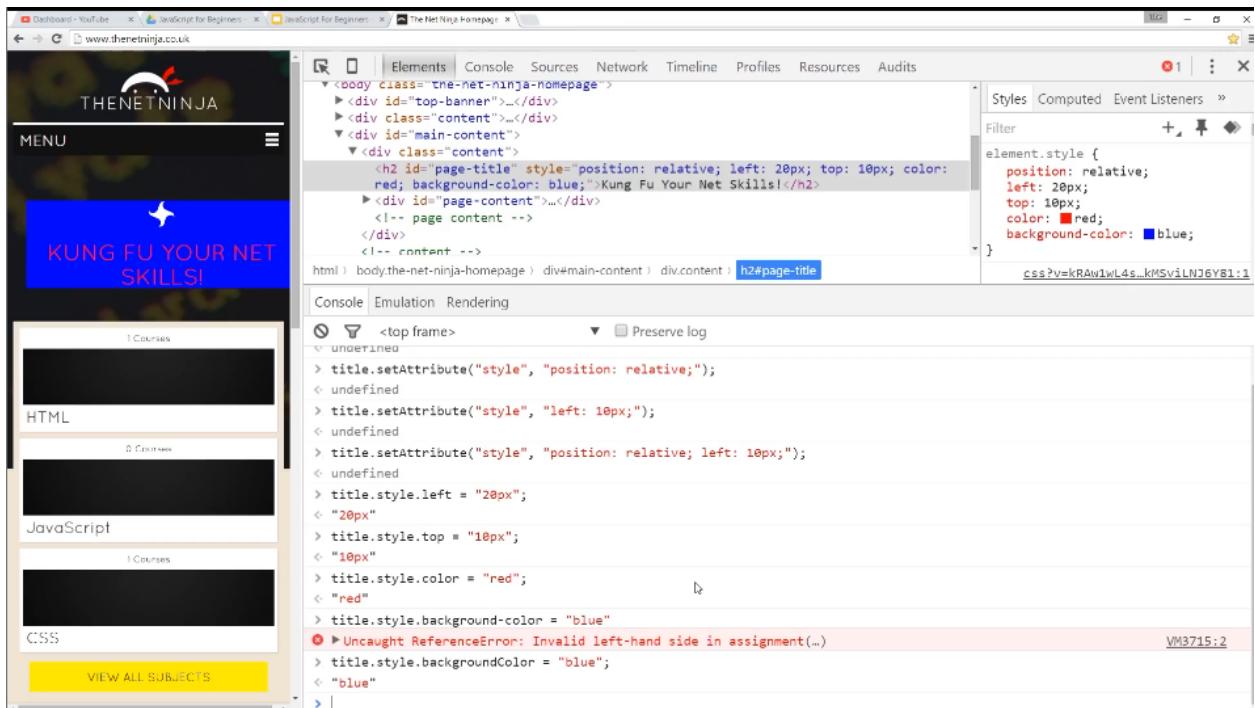
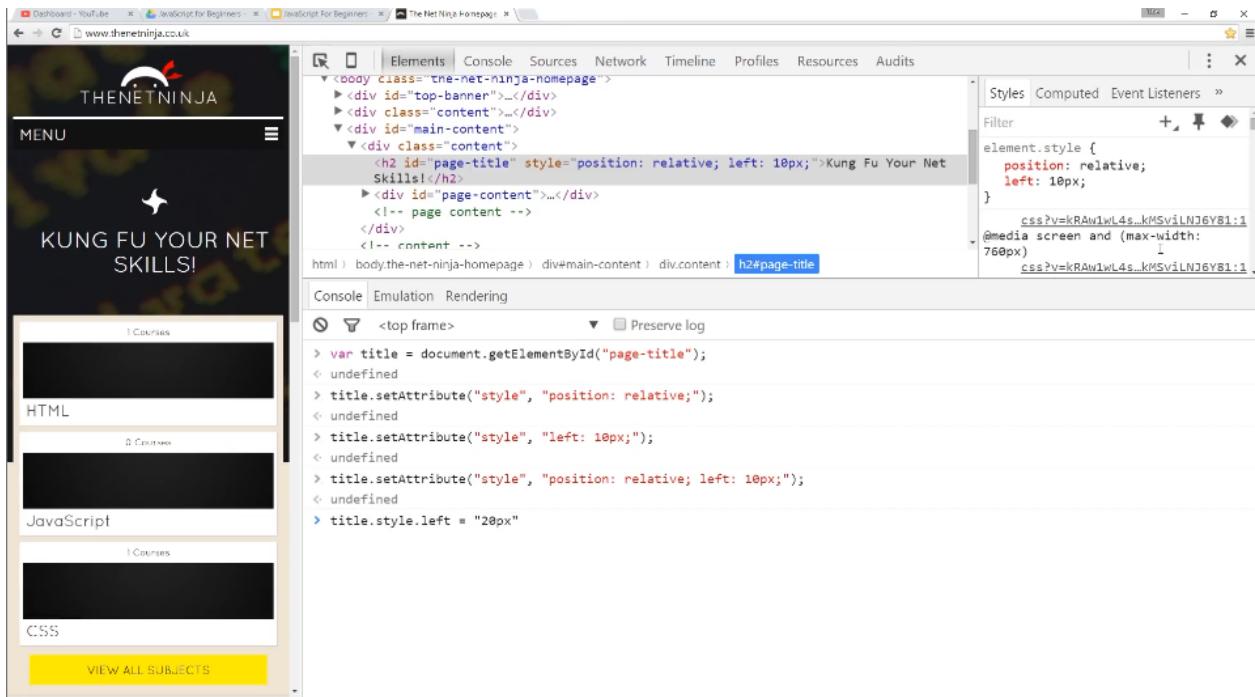
```
h2#page-title {
  position: relative;
}
@media screen and (max-width: 760px) {
  h2#page-title {
    left: 10px;
  }
}
```

```
> var title = document.getElementById("page-title");
< undefined
> title.setAttribute("style", "position: relative;");
< undefined
> title.setAttribute("style", "left: 10px;")
```

This screenshot shows the same homepage after the CSS update. The 'KUNG FU YOUR NET SKILLS!' banner has shifted 10 pixels to the left. The developer tools interface is identical to the first screenshot, with the Styles panel showing the modified CSS rule and the browser's console tab displaying the applied JavaScript code.

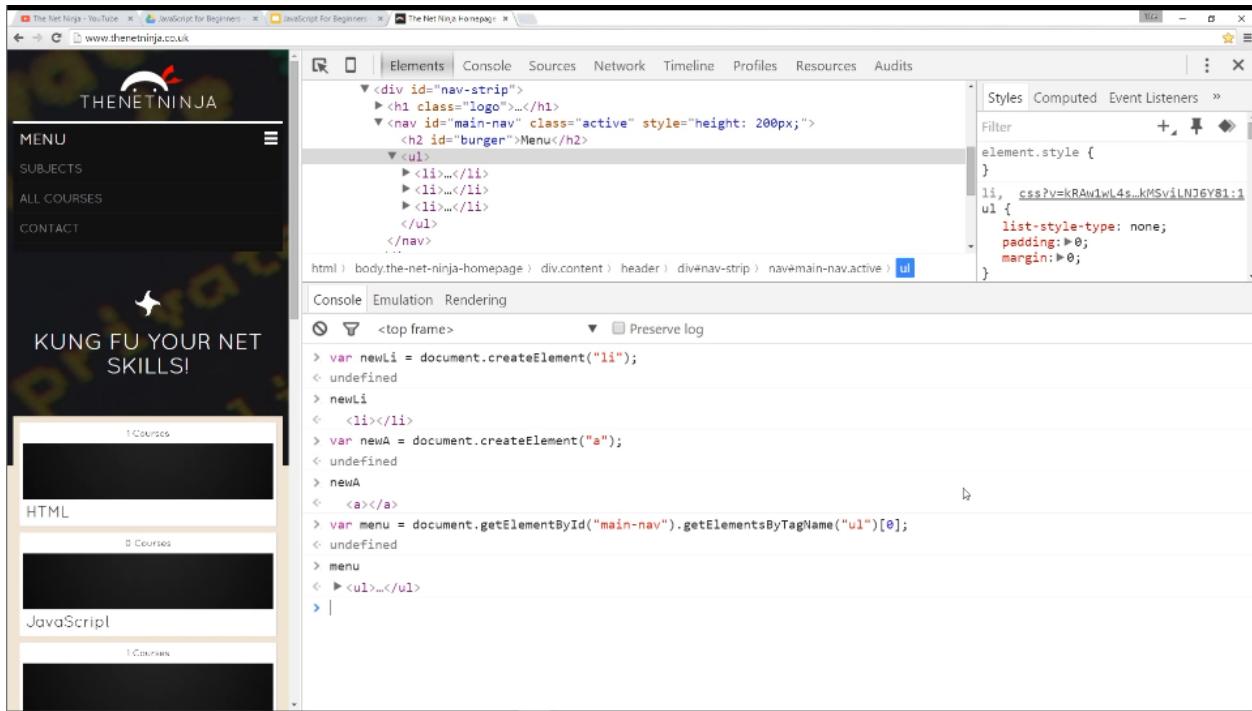
```
h2#page-title {
  left: 10px;
}
@media screen and (max-width: 760px) {
  h2#page-title {
    left: 10px;
  }
}
```

```
> var title = document.getElementById("page-title");
< undefined
> title.setAttribute("style", "position: relative;");
< undefined
> title.setAttribute("style", "left: 10px;")
```

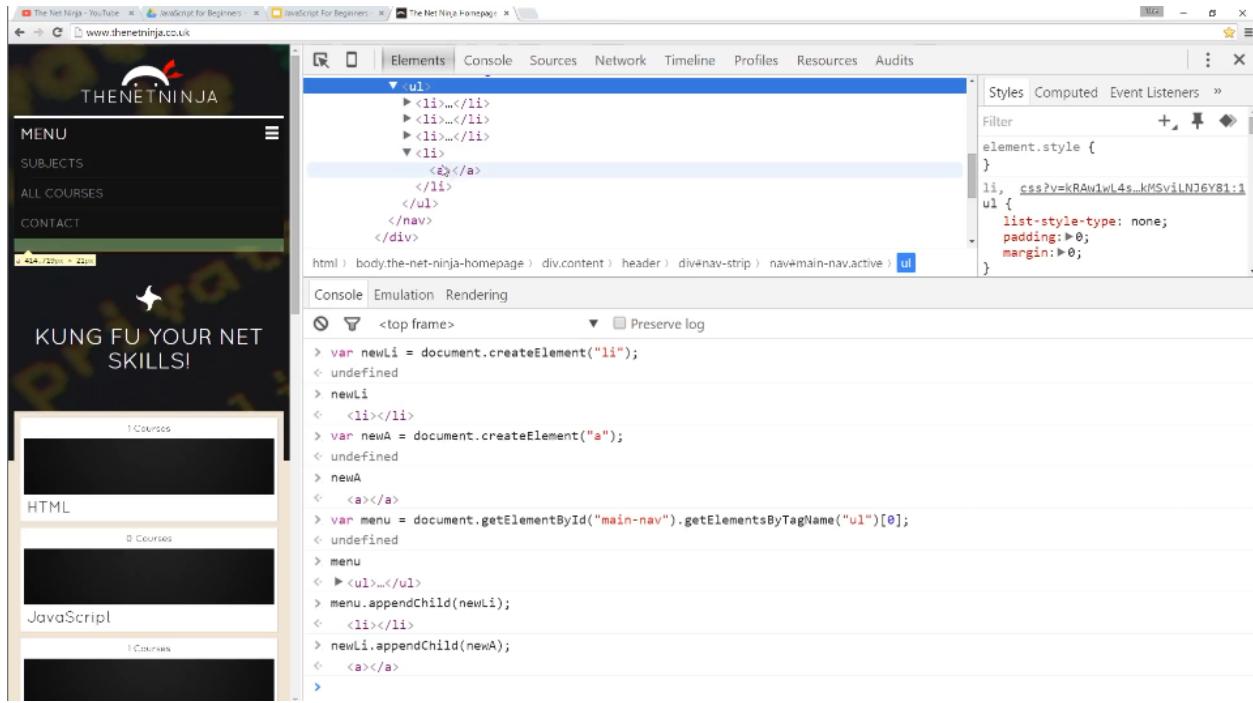


37. Adding Elements to the DOM

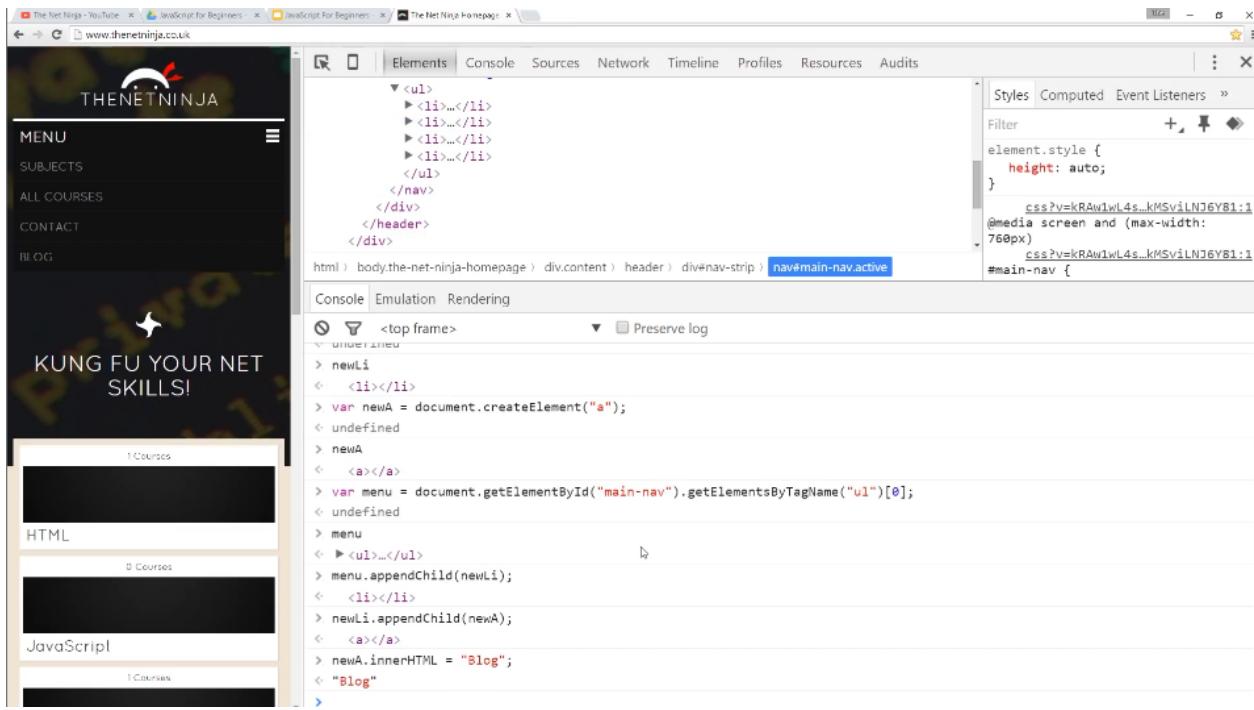
Step 1: Get the Data where to add “new Li”



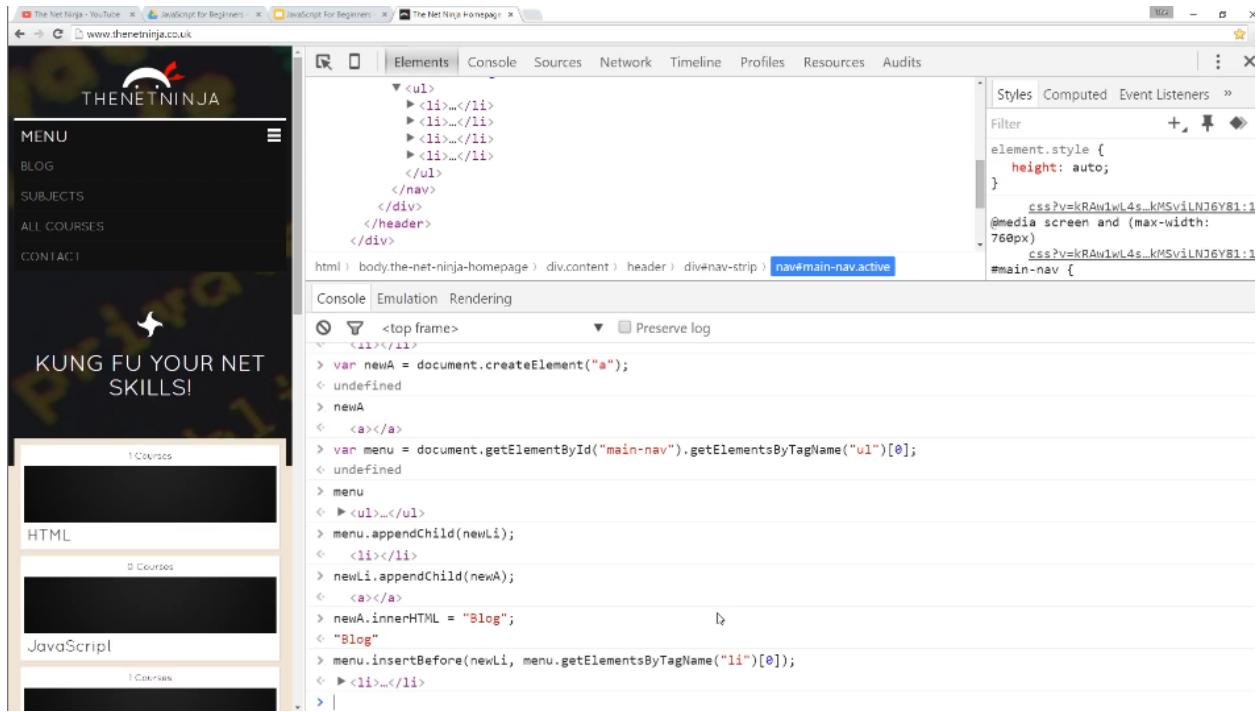
Step 2: Adding the “new link”



→Here “Li” is inserted at bottom & If the styles height is modified to “auto” it is visible in page



→Here “Li” is inserted at the Top



The screenshot shows the Chrome DevTools Elements tab with the DOM tree for the homepage. The main navigation bar is represented by a `ul` element. A new `li` element, containing the text "Blog", has been inserted as the first child of this `ul` element. The CSS styles for the main navigation are visible on the right side of the DevTools interface.

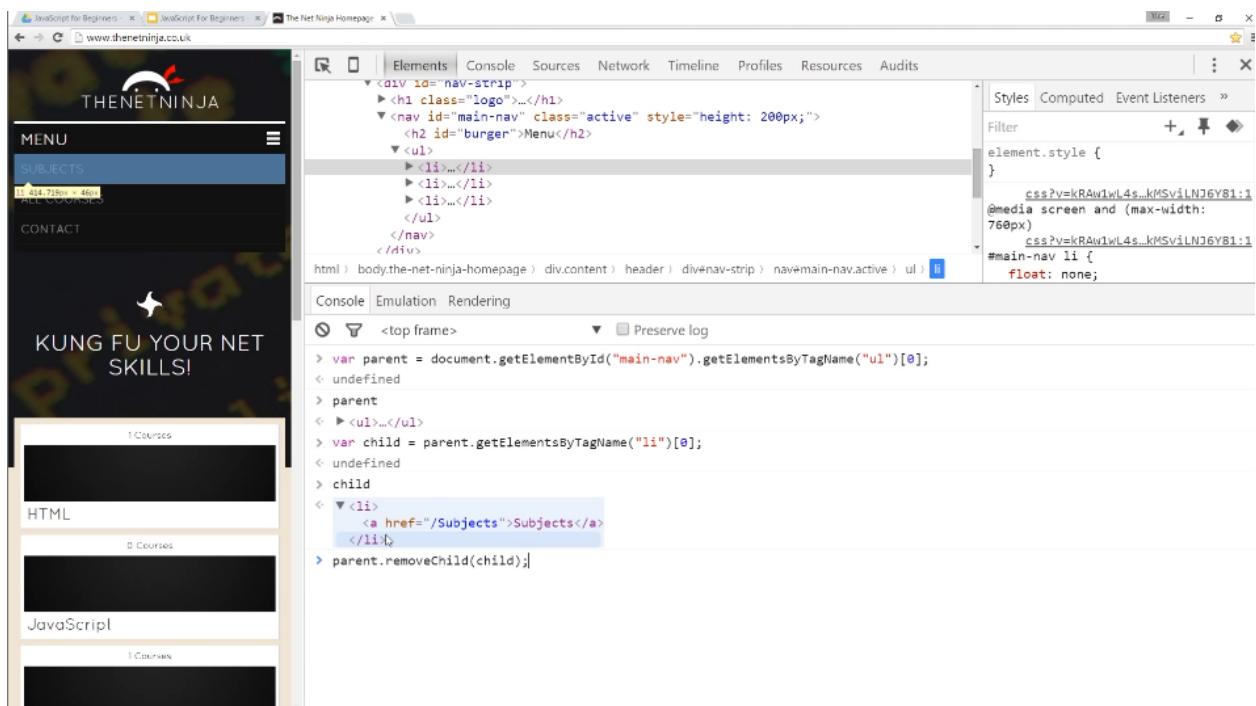
```
<ul>
  <li>...</li>
  <li>...</li>
  <li>...</li>
  <li>...</li>
  <li>...</li>
</ul>
</nav>
</div>
</header>
</div>
```

```
html > body.the-net-ninja-homepage > div.content > header > div>nav-strip > nav>main-nav.active
```

```
Styles Computed Event Listeners »
Filter + ⚙️
element.style {
  height: auto;
}
css?v=kRAv1wL4s..kMSviLNJ6YB1:1
@media screen and (max-width: 768px)
css?v=kRAv1wL4s..kMSviLNJ6YB1:1
#main-nav {
```

```
Console Emulation Rendering
< top frame> ▾ <1><2>/<1>
> var newA = document.createElement("a");
< undefined
> newA
< <a></a>
> var menu = document.getElementById("main-nav").getElementsByTagName("ul")[0];
< undefined
> menu
< > <ul>...</ul>
> menu.appendChild(newA);
< <li></li>
> newLi.appendChild(newA);
< <a></a>
> newA.innerHTML = "Blog";
< "Blog"
> menu.insertBefore(newLi, menu.getElementsByTagName("li")[0]);
< > <li>...</li>
> |
```

38. Removing Elements from the DOM



The screenshot shows the Chrome DevTools Elements tab with the DOM tree for the homepage. The `ul` element under the main navigation now has fewer children than before, indicating that the "Subjects" item has been removed. The CSS styles for the main navigation are visible on the right side of the DevTools interface.

```
<div id="nav-strip">
  <h1 class="logo">...</h1>
  <nav id="main-nav" class="active" style="height: 200px;">
    <h2 id="burger">Menu</h2>
    <ul>
      <li>...</li>
      <li>...</li>
      <li>...</li>
    </ul>
  </nav>
</div>
```

```
html > body.the-net-ninja-homepage > div.content > header > div>nav-strip > nav>main-nav.active > ul
```

```
Styles Computed Event Listeners »
Filter + ⚙️
element.style {
}
css?v=kRAv1wL4s..kMSviLNJ6YB1:1
@media screen and (max-width: 768px)
css?v=kRAv1wL4s..kMSviLNJ6YB1:1
#main-nav li {
  float: none;
```

```
Console Emulation Rendering
< top frame> ▾ <1><2>/<1>
> var parent = document.getElementById("main-nav").getElementsByTagName("ul")[0];
< undefined
> parent
< <ul>...</ul>
> var child = parent.getElementsByTagName("li")[0];
< undefined
> child
< <li>
  <a href="/Subjects">Subjects</a>
</li>
> parent.removeChild(child);
```

→Removed “Subjects”

The screenshot shows the Net Ninja homepage with the developer tools DevTools open. The 'Subjects' menu item is selected in the Elements tab's DOM tree. The browser console contains the following JavaScript code:

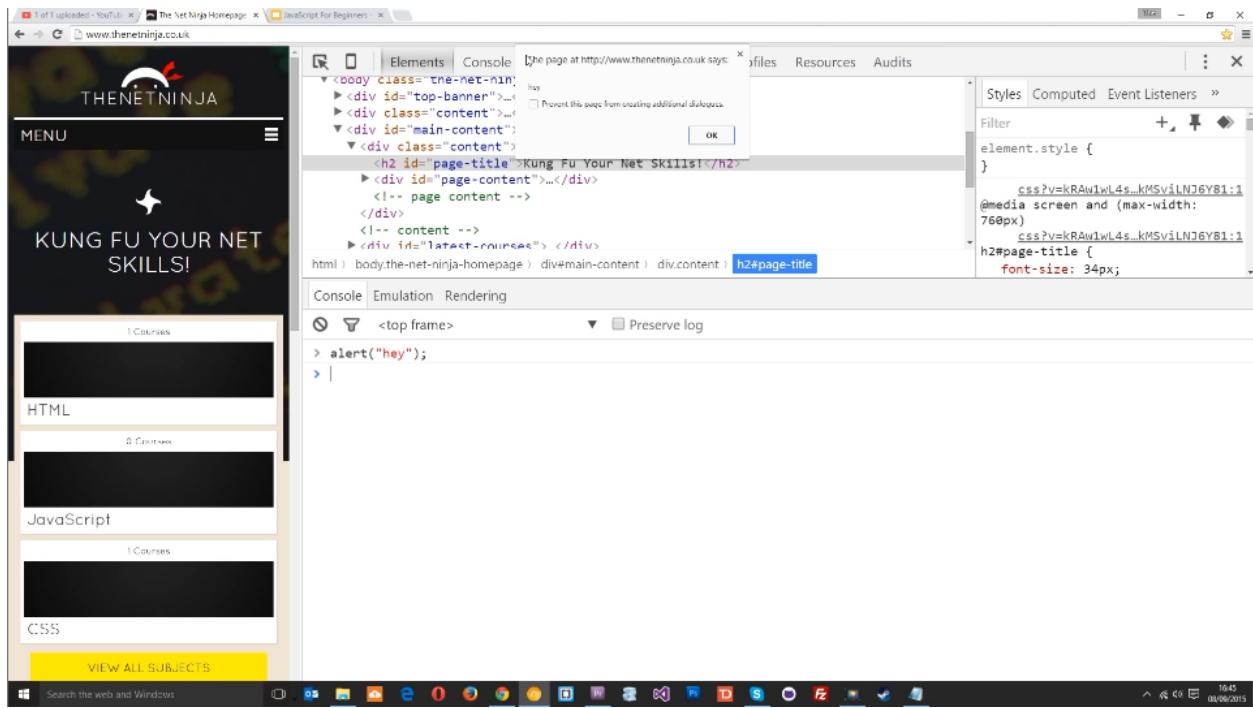
```
> var parent = document.getElementById("main-nav").getElementsByTagName("ul")[0];
< undefined
> parent
< > <ul>
> var child = parent.getElementsByTagName("li")[0];
< undefined
> child
< <li>
< <a href="/Subjects">Subjects</a>
</li>
> var removed = parent.removeChild(child);
< undefined
> parent.appendChild(removed)
< > <li>
```

→added back to parent.

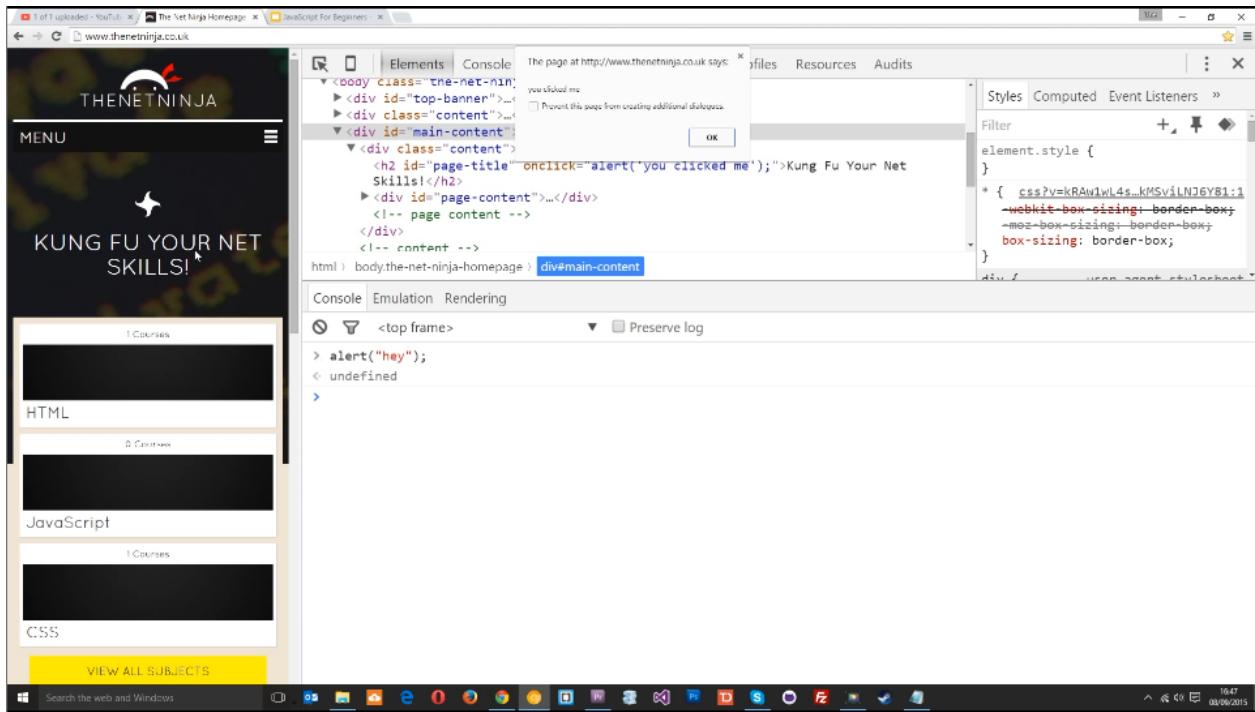
The screenshot shows the Net Ninja homepage with the developer tools DevTools open. The 'Subjects' menu item is now visible in the 'main-nav' ul in the DOM tree. The browser console contains the same JavaScript code as the previous screenshot, demonstrating the addition of the removed item back to its parent.

39. Introduction to JavaScript Events

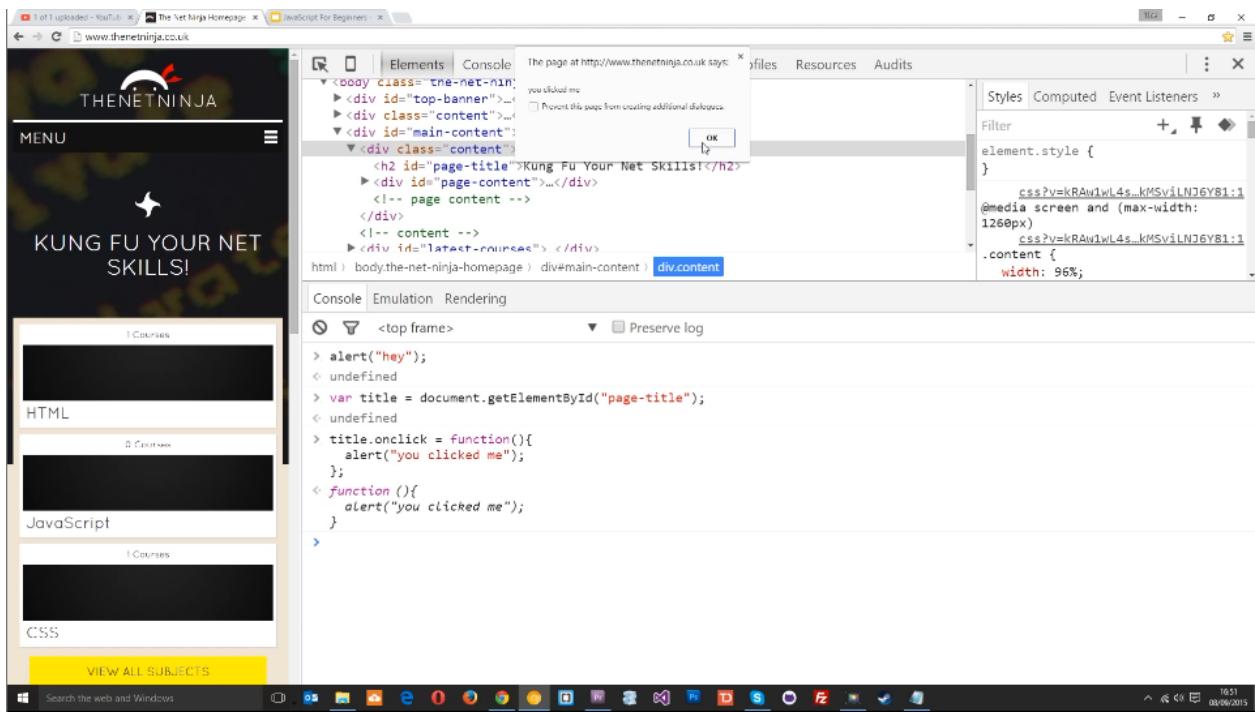
→ Dialog box appear at Top of the page



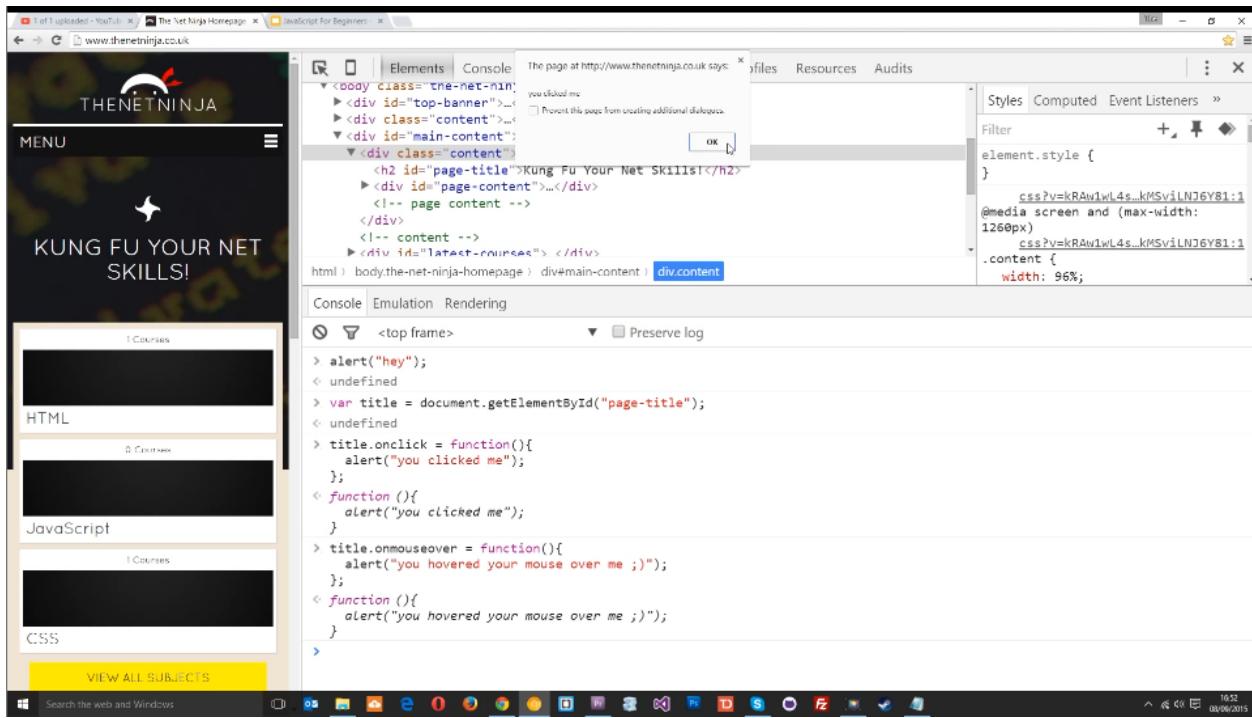
→Event is added (“onclick”)



→If I click on the “KUNG FU YOUR NET SKILLS!” it will show the dialog box



→ If I drag the mouse on the “KUNG FU YOUR NET SKILLS!” it will show the dialog box



40. The onClick Event

index.html

```
<!DOCTYPE html>
<html lang="en">

<head>
    <meta charset="utf-8">
    <title>JavaScript for Beginners</title>
    <link href="style.css" type = "text/css" rel = "stylesheet">
</head>
<body>

    <div id="content">
```

```
<p>The ICC Cricket World Cup is an international cricket tournament held every four years. It is organized by the International Cricket Council (ICC) and features teams from around the world competing for the prestigious title. </p>
```

```
<p>The Cricket World Cup 2023 is scheduled to be the 13th edition of the tournament. It is set to be hosted by India, with matches expected to be played in various cities across the country. The tournament will bring together the top cricketing nations to compete in a series of One Day International (ODI) matches</p>
```

```
<p>The qualification process for the Cricket World Cup usually involves teams earning qualification spots through various regional tournaments and rankings. The final tournament typically consists of 10 teams competing in a round-robin format, followed by knockout stages leading up to the final match.</p>
```

```
</div>
```

```
<a id ="show-more">Show More</a>
```

```
<script src="test.js"></script>
```

```
</body>
</html>
```

style.css

```
body{
    background: #ccc;
}

#content{
    width: 400px;
    background: #fff;
    padding: 20px;
    padding-top: 0;
    font-family: calibri;
```

```
font-size: 18px;
color: #444;
margin: 0 auto;

max-height: 270px;
overflow: hidden;

/* set transitions up. */
-webkit-transition: max-height 0.7s;
-moz-transition: max-height 0.7s;
transition: max-height 0.7s;
}

#content.open{
    max-height: 1000px;

    /* set transitions up. */
    -webkit-transition: max-height 0.7s;
    -moz-transition: max-height 0.7s;
    transition: max-height 0.7s;
}

#show-more{
    background: #1594e5;
    color: #fff;
    font-family: calibri;
    display: block;
    width: 140px;
    font-size: 24px;
    text-transform: uppercase;
    padding: 10px;
    text-align: center;
    margin: 20px auto;
    cursor: pointer;
}
```

test.js

```
var content = document.getElementById("content");
var button = document.getElementById("show-more");

button.onclick = function() {

    if(content.className == "open"){
        //shrink the box
        content.className = "";
        button.innerHTML = "Show More";
    }else{
        //expand the box
        content.className = "open";
        button.innerHTML = "Show Less";
    }
};
```

O/P:



The screenshot shows a web browser window with the URL <http://127.0.0.1:5500/index.html>. The page content is as follows:

The ICC Cricket World Cup is an international cricket tournament held every four years. It is organized by the International Cricket Council (ICC) and features teams from around the world competing for the prestigious title.

The Cricket World Cup 2023 is scheduled to be the 13th edition of the tournament. It is set to be hosted by India, with matches expected to be played in various cities across the country. The tournament will bring together the top cricketing nations to compete in a series of One Day International (ODI) matches.

The qualification process for the Cricket World Cup usually involves teams earning qualification spots through various regional tournaments and rankings. The final tournament typically consists of 10 teams competing in a round-robin format, followed by knockout stages leading up to the final match.

SHOW LESS

The browser's address bar and toolbar are visible at the top, and a vertical scrollbar is on the right side of the page.

41. Window onLoad Event

→ If the “script” is placed at the head part then it won’t load the script
That is why introduce “**window.onload**”

index.html

```
<!DOCTYPE html>
<html lang="en">

<head>
    <meta charset="utf-8">
    <title>JavaScript for Beginners</title>
    <link href="style.css" type = "text/css" rel = "stylesheet">

    <script src="test.js"></script>

</head>
<body>
```

```

<div id="content">
    <p>The ICC Cricket World Cup is an international cricket tournament held every four years. It is organized by the International Cricket Council (ICC) and features teams from around the world competing for the prestigious title. </p>

    <p>The Cricket World Cup 2023 is scheduled to be the 13th edition of the tournament. It is set to be hosted by India, with matches expected to be played in various cities across the country. The tournament will bring together the top cricketing nations to compete in a series of One Day International (ODI) matches</p>

    <p>The qualification process for the Cricket World Cup usually involves teams earning qualification spots through various regional tournaments and rankings. The final tournament typically consists of 10 teams competing in a round-robin format, followed by knockout stages leading up to the final match.</p>
</div>

<a id ="show-more">Show More</a>

</body>
</html>

```

test.js

```

function setUpEvents(){

    var content = document.getElementById("content");
    var button = document.getElementById("show-more");

    button.onclick = function()  {

        if(content.className == "open"){
            //shrink the box
            content.className = "";
        }
    }
}

```

```

        button.innerHTML = "Show More";
    }else{
        //expand the box
        content.className = "open";
        button.innerHTML = "Show Less";
    }
};

}

window.onload = function() {
    setUpEvents();
};

}

```

→there is no change in the “style.css”

O/P:



The screenshot shows a web browser window with the URL <http://127.0.0.1:5500/index.html>. The page content is as follows:

The ICC Cricket World Cup is an international cricket tournament held every four years. It is organized by the International Cricket Council (ICC) and features teams from around the world competing for the prestigious title.

The Cricket World Cup 2023 is scheduled to be the 13th edition of the tournament. It is set to be hosted by India, with matches expected to be played in various cities across the country. The tournament will bring together the top cricketing nations to compete in a series of One Day International (ODI) matches.

The qualification process for the Cricket World Cup usually involves teams earning qualification spots through various regional tournaments and rankings. The final tournament typically consists of 10 teams competing in a round-robin format, followed by knockout stages leading up to the final match.

[SHOW LESS](#)

The browser's address bar and toolbar are visible at the top, and a sidebar on the right contains links like 'Maps', 'Gmail', 'WhatsApp', 'YouTube', 'hrms | IRIS', 'GATE CS Preparation...', 'LeetCode', 'Must Do Coding Qu...', 'DSA self placed cour...', and 'Other bookmarks'.

42. JavaScript Timers

index.html

```
<!DOCTYPE html>
<html lang="en">

<head>
    <meta charset="utf-8">
    <title>JavaScript for Beginners</title>
    <link href="style.css" type = "text/css" rel = "stylesheet">
</head>
<body>

    <div id="message">
        <p>THIS IS A REEEEEAAAALITY IMPORTANT NOTIFICATION</p>

        <p>PIE IS NICE</p>
    </div>

    <script src="test.js"></script>

```

```
</body>
</html>
```

style.css

```
body{
    background: #eee;
}

#message{
    width: 400px;
    background: #ddaaaa;
    padding: 20px;
    font-family: calibri;
    font-size: 18px;
    color: #444;
    margin: 0 auto;
    text-align: center;
    border: 1px solid #ff7777;
    font-weight: bold;
    opacity: 0;

    /* set transitions up. */
    -webkit-transition: opacity 0.7s;
    -moz-transition: opacity 0.7s;
    transition: opacity 0.7s;
}

#message.show{
    opacity: 1;

    /* set transitions up. */
    -webkit-transition: opacity 0.7s;
    -moz-transition: opacity 0.7s;
    transition: opacity 0.7s;
}
```

test.js

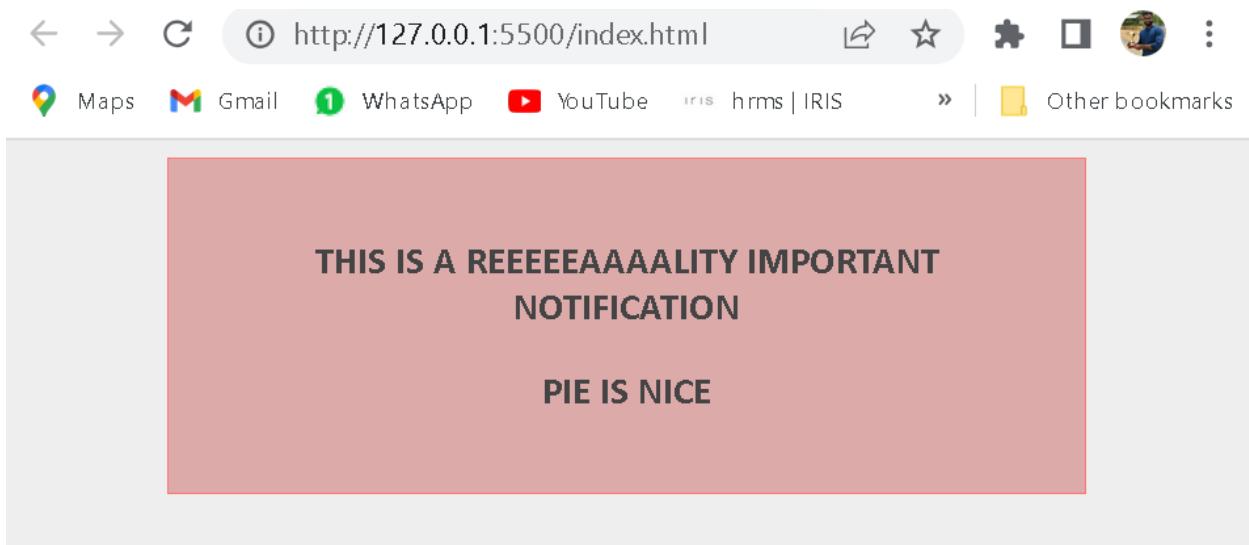
```
var myMessage = document.getElementById("message");

function showMessage() {

    myMessage.className = "show";
}

setTimeout(showMessage, 3000);
```

O/P:



index.html

```
<!DOCTYPE html>
<html lang="en">

<head>
    <meta charset="utf-8">
    <title>JavaScript for Beginners</title>
    <link href="style.css" type = "text/css" rel = "stylesheet">
</head>
<body>
```

```
<div id="colour-changer"></div>

<script src="test.js"></script>

</body>
</html>
```

style.css

```
body{
    background: #eee;
}

#message{
    width: 400px;
    background: #ddaaaa;
    padding: 20px;
    font-family: calibri;
    font-size: 18px;
    color: #444;
    margin: 0 auto;
    text-align: center;
    border: 1px solid #ff7777;
    font-weight: bold;
    opacity: 0;

    /* set transitions up. */
    -webkit-transition: opacity 0.7s;
    -moz-transition: opacity 0.7s;
    transition: opacity 0.7s;
}

#message.show{
    opacity: 1;

    /* set transitions up. */
    -webkit-transition: opacity 0.7s;
```

```

-moz-transition: opacity 0.7s;
transition: opacity 0.7s;
}

#colour-changer{
width: 200px;
height: 100px;
margin: 30px auto;
border: 1px solid #000;
background: #fff;

-webkit-transition: background 0.7s;
-moz-transition: background 0.7s;
transition: background 0.7s;
}

```

test.js

```

var colourChanger = document.getElementById("colour-changer");
var colours = ["red", "blue", "green", "pink"];
var counter = 0;

function changeColour() {

    if(counter >= colours.length){ //repeating the cycle
        counter = 0;
    }

    colourChanger.style.background = colours[counter];
    counter++;

}

var myTimer = setInterval(changeColour, 3000); //3000ms or 3sec

colourChanger.onclick = function() {

    clearInterval(myTimer);
    colourChanger.innerHTML = "Timer stopped";
}

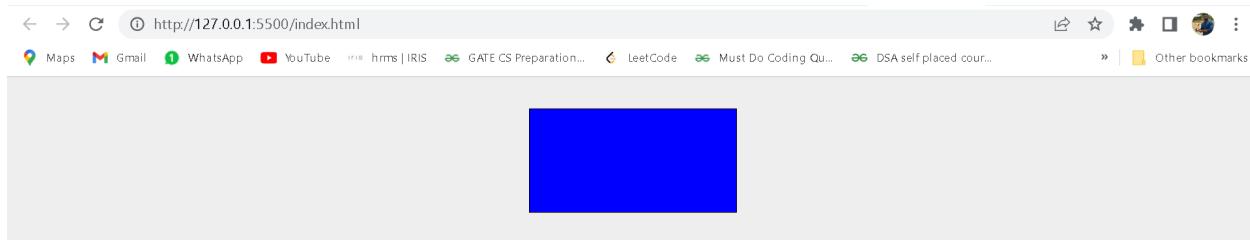
```

```
};
```

O/P:



→ after every 3sec it will change the colour



43. Accessing Form Elements

index.html

```
<!DOCTYPE html>
<html lang="en">

<head>
    <meta charset="utf-8">
    <title>JavaScript for Beginners</title>
    <link href="style.css" type = "text/css" rel = "stylesheet">
</head>
<body>

    <form id="my-form" name="myForm" action="#">
        <label for="name">Name:</label>
```

```

<input type="text" name="name"><br>
<input type="checkbox" name="biking" value="biking">Biking<br>
<input type="checkbox" name="skiing" value="skiing">Skiing<br>
<input type="checkbox" name="driving" value="driving">Driving<br>
<label for="colours">Fav colour:</label>
<select name="colour" id="">
    <option>Red</option>
    <option>Blue</option>
    <option>Green</option>
</select>
<input type="submit" name="submit" value="submit">
</form>
<script src="test.js"></script>

</body>
</html>

```

→there is no change in the test.js & style.css

The screenshot shows the Brackets IDE interface with the file 'index.html' open. The code in index.html is as follows:

```

<!DOCTYPE html>
<html lang="en">
    <head>
        <meta charset="utf-8" />
        <title>JavaScript for Beginners</title>
        <link href="style.css" type="text/css" rel="stylesheet">
    </head>
    <body>
        <form id="myForm" name="myForm" action="#">
            <label for="name">Name:</label>
            <input type="text" name="name"><br/>
            <label>Hobbies:</label><br/>
            <input type="checkbox" name="biking" value="biking">Biking<br/>
            <input type="checkbox" name="skiing" value="skiing">Skiing<br/>
            <input type="checkbox" name="driving" value="driving">Driving<br/>
            <label for="colour">Fav colour:</label>
            <select name="colour">
                <option>Red</option>
                <option>Blue</option>
                <option>Green</option>
            </select>
            <input type="submit" name="submit" value="Submit">
        </form>
        <script src="test.js"></script>
    </body>
</html>

```

To the right of the code editor is a browser window showing the form with the following values:

- Name: ninja
- Hobbies: Biking, Skiing, Driving
- Fav colour: Blue

The browser's developer tools are open, showing the following details:

- Elements** tab: Shows the DOM structure of the page.
- Console** tab: Displays the JavaScript console output, which includes the following code and its execution results:

```

> document.forms.myForm;
< [object HTMLFormElement]>
> var myForm = document.forms.myForm;
< undefined>
> myForm
< [object HTMLFormElement]>
> myForm.name
< [object HTMLInputElement]>
> myForm.colour.value
< "Blue">
> myForm.name.onfocus = function(){
    myForm.name.style.border = "4px solid pink";
};
< function ()>
> myForm.name.onblur = function(){
    myForm.name.style.border = "none";
};
< function ()>

```

O/P:

The screenshot shows a browser window with the URL `http://127.0.0.1:5500/index.html`. The page contains a form with fields for Name, Fav colour, and checkboxes for Biking, Skiing, and Driving. A 'submit' button is present. The developer tools console tab is active, displaying the following code and its execution stack:

```
<form id="my-form" name="myForm" action="#">  
myForm.name  
> myForm.name.value  
< ''  
> myForm.name.value  
< 'raghava'  
> myForm.colour.value  
< 'Red'  
> myForm.name.onfocus = function() {  
}  
< f () {  
}  
> myForm.name.onfocus = function() {  
    myForm.name.style.border = "4px solid pink";  
};  
< f () {  
    myForm.name.style.border = "4px solid pink";  
}  
>
```

← → ⌂ ⓘ http://127.0.0.1:5500/index.html

Maps Gmail WhatsApp YouTube IRIS hrms | IRIS 213 5 1850 Other bookmarks

Name: **raghava**

Biking
 Skiing
 Driving

Fav colour: **Red** ▾

submit

Elements Console 1,915 Issues: 1850 65

Default levels ▾

- ▶ 343 messages
- ▶ 2 user messages
- ▶ 250 errors
- ▶ 76 warnings
- ▶ No info
- ▶ 17 verbose

```
< ''
> myForm.name.value
< 'raghava'
> myForm.colour.value
< 'Red'
> myForm.name.onfocus = function() {
    }
< f () {
}
> myForm.name.onfocus = function() {
    myForm.name.style.border ="4px solid pink";
};
< f () {
    myForm.name.style.border ="4px solid pink";
}
> myForm.name.onblur = function() {
    myForm.name.style.border ="none";
};
< f () {
    myForm.name.style.border ="none";
}>
```

44.Very Simple Form Validation

index.html

```
<!DOCTYPE html>
<html lang="en">

<head>
    <meta charset="utf-8">
    <title>JavaScript for Beginners</title>
    <link href="style.css" type = "text/css" rel = "stylesheet">
</head>
<body>

<form id="my-form" name="myForm" action="#">
    <label for="name">Name:</label>
    <input type="text" name="name"><br>
    <label>Hobbies:</label><br>
    <input type="checkbox" name="biking" value="biking">Biking<br>
    <input type="checkbox" name="skiing" value="skiing">Skiing<br>
    <input type="checkbox" name="driving" value="driving">Driving<br>
    <label for="colour">Fav colour:</label>
    <select name="colour">
        <option>Red</option>
        <option>Blue</option>
        <option>Green</option>
    </select>
    <input type="submit" name="submit" value="Submit">
</form>

<div id="message" style="color: red;"></div>

<script src="test.js"></script>
</body>
</html>
```

style.css

```
body{
    background: #eee;
}

#message{
    width: 400px;
    font-family: calibri;
    font-size: 18px;
    color: #444;
    font-weight: bold;

    /* set transitions up. */
    -webkit-transition: opacity 0.7s;
    -moz-transition: opacity 0.7s;
    transition: opacity 0.7s;
}
```

test.js

```
var myForm = document.getElementById("my-form");
var message = document.getElementById("message");

myForm.onsubmit = function() {

    if(myForm.name.value == "") {
        message.innerHTML = "please enter a name";
        return false;
    } else{
        message.innerHTML = "";
        return true;
    }
};
```

O/P:

http://127.0.0.1:5500/index.html

Name:

Hobbies:

Biking
 Skiing
 Driving

Fav colour:

http://127.0.0.1:5500/index.html?name=Gudiwada+Raghava&colour=Blue&submit=Submit#

Name:

Hobbies:

Biking
 Skiing
 Driving

Fav colour:

← → ⌛ ⓘ http://127.0.0.1:5500/index.html

Maps Gmail WhatsApp YouTube IRIS hrms | IRIS GATE

Name:

Hobbies:

Biking
 Skiing
 Driving

Fav colour:

please enter a name

45. JavaScript Libraries

JAVASCRIPT LIBRARIES

- General purpose
- Animation
- Form enhancement
- Video
- Many more...

<https://jquery.com/> →Download jQuery click

Go to the “Using jQuery with a CDN”

<https://releases.jquery.com/> →jQuery

index.html

```
<!DOCTYPE html>
<html lang="en">

<head>
    <meta charset="utf-8">
    <title>JavaScript for Beginners</title>
    <link href="style.css" type = "text/css" rel = "stylesheet">
</head>
<body>

    <div id="content">
        <p>read the text message</p>
        <p>grab me</p>
    </div>

    <script src="https://code.jquery.com/jquery-1.12.4.min.js"></script>
    <script src="test.js"></script>
</body>
</html>
```

test.js

```
var myPara = document.getElementById("content").getElementsByTagName("p")[5];
var myPara2 = $("#content p:last-child");
```

→add class

The screenshot shows a browser window with the URL `http://127.0.0.1:5500/index.html`. The page content consists of six `p` elements with the text "read the text message" repeated five times and "grab me" once. The developer tools are open, specifically the Elements and Issues panels.

Elements Tab: Shows the DOM structure. The last `p` element has a class attribute set to "test".

```
<p>read the text message</p>
<p>read the text message</p>
<p>read the text message</p>
<p>read the text message</p>
<p class="test">grab me</p>
```

Issues Panel: Shows 812 issues with 773 errors and 39 warnings.

- 62 messages
- 2 user mes...
- 20 errors
- 37 warnings
- No info (selected)
- 5 verbose

```
> var myPara2 = $("#content p:last-child");
< undefined
> myPara2.addClass("test");
< n.fn.init [p.test, prevObject: n.fn.init
  > (1), context: document, selector: '#content p:last-child']
>
```

→remove class

The screenshot shows a browser window with the URL `http://127.0.0.1:5500/index.html`. The page content contains six `p` elements with the text "read the text message" and one `p` element with the text "grab me". The developer tools are open, displaying the following panels:

- Elements Panel:** Shows the DOM structure with the `<script src="https://code.jquery.com/jquery-1.12.4.min.js">` tag.
- Console Panel:** Shows the following JavaScript code and its execution results:

```
var myPara2 = $("#content p:last-child");
myPara2.addClass("test");
myPara2.removeClass("test");
```
- Issues Panel:** Shows 914 issues with the following details:
 - 76 messages
 - 3 user mes...
 - 24 errors
 - 46 warnings
 - No info (selected)
 - 6 verbose

→FadeOut

The screenshot shows a browser window with the URL `http://127.0.0.1:5500/index.html`. The page content consists of five identical paragraphs: "read the text message". The browser's developer tools are open, specifically the Elements, Console, and Issues panels.

Elements Panel: Shows the DOM structure of the page. It includes the following code:

```
<p>read the text message</p>
<p>read the text message</p>
<p>read the text message</p>
<p>read the text message</p>
<p class style="display: none;">grab me</p>
</div>
<script src="https://code.jquery.com/jquery-1.12.4.min.js">
```

Console Panel: Shows the following command and its result:

```
myPara2.removeClass("test");
```

Issues Panel: Shows 1,067 issues with the following details:

- 110 messages
- 4 user messages
- 33 errors
- 68 warnings
- No info (selected)
- 9 verbose

Specific error details shown in the Issues panel:

- `n.fn.init [p.test, prevObject: n.fn.init(1), context: document, selector: '#content p:last-child']`
- `n.fn.init [p, prevObject: n.fn.init(1), context: document, selector: '#content p:last-child']`
- `n.fn.init [p, prevObject: n.fn.init(1), context: document, selector: '#content p:last-child']`

→FadeIn

The screenshot shows a browser window with the URL `http://127.0.0.1:5500/index.html`. The page content consists of six `p` elements with the text "read the text message" repeated five times and "grab me" once. Below the page content, the browser's developer tools are open:

- Elements Panel:** Shows the DOM structure with the last `p` element having a `class="style='display: block;'"` attribute.
- Console Panel:** Shows the following JavaScript code:

```
myPara2.removeClass('test');
n.fn.init [p, prevObject: n.fn.init(1),
  context: document, selector: '#content p:last-child']
myPara2.fadeOut();
n.fn.init [p, prevObject: n.fn.init(1),
  context: document, selector: '#content p:last-child']
myPara2.fadeIn();
n.fn.init [p, prevObject: n.fn.init(1),
  context: document, selector: '#content p:last-child']
```
- Issues Panel:** Displays 1,113 issues with the following breakdown:
 - 119 messages
 - 4 user messages
 - 36 errors
 - 73 warnings
 - No info (selected)
 - 10 verbose

→add css

The screenshot shows a browser window with developer tools open. The address bar shows the URL `http://127.0.0.1:5500/index.html`. The developer tools interface includes tabs for Elements, Console, Sources, Network, Performance, and Memory. The Issues panel is visible on the right, displaying 242 issues. One specific issue is highlighted: `myPara2.css({position: "relative", color: "red"});` with a warning icon. The code editor shows a red message: `<p>read the text message</p>` followed by a CSS rule: `<p style="position: relative; color: red;">grab me</p>`.

→Animate

The screenshot shows a browser window with developer tools open. The address bar shows the URL `http://127.0.0.1:5500/index.html`. The developer tools interface includes tabs for Elements, Console, Sources, Network, Performance, and Memory. The Issues panel is visible on the right, displaying 519 issues. One specific issue is highlighted: `myPara2.css({position: "relative", color: "red"});` with a warning icon. The code editor shows a red message: `<p>read the text message</p>` followed by a CSS rule: `<p style="position: relative; color: red; left: 50px;">grab me</p>`. Additionally, there is a JavaScript animation call: `myPara2.animate({left: "50px"});`

46 What to Study Next

JAVASCRIPT LIBRARIES

- jQuery
- MooTools
- Modernizr

AJAX AND JSON

- AJAX IS ESSENTIALLY A PART OF JAVASCRIPT
- Allows JavaScript to communicate with the server without having to leave the web page
- Great example is when you zoom in on Google Maps
- JSON is just a way of organizing data & works well with AJAX

EXPLORING JAVASCRIPT FURTHER

- Lots in JavaScript that we haven't covered:
 - Regular expressions & advanced validation
 - OOP (object orientated programming)
 - Sliders, drag and drop, animated menu's