Create form??? https://www.oreilly.com/videos/html-css-and/9781838551278/9781838551278-video2\_33/

Position video review

[Introduction to float (linkedin.com)](https://www.linkedin.com/learning/css-essential-training-3/introduction-to-float?autoplay=true&resume=false&u=2097252)

apply position: absolute; to a container element, it will be positioned relative to its closest positioned ancestor element, regardless of the position value of its children.

The position: relative; property on the child element only affects the position of the element itself and its descendants, not its parent.

* Notice the background color of the padding vs the margin.
* Note: Margin does not contribute to the total size of the element since it adds space around the element

But it does affect the amount of space that the element takes up.

**only content and border can be assigned color**

Headings should have Arial, Helvetica, sans-serif font and be [center-aligned](https://developer.mozilla.org/en-US/docs/Web/CSS/text-align).

h1 {

font-family: Arial, Helvetica, sans-serif;

text-align: center;

}

1 rem: relative to font size of the root element

1 em: relative to font size of the element.

|  |  |  |
| --- | --- | --- |
| **Selector** | **Example** | **Example description** |
| Descendant selector  [element element](https://www.w3schools.com/cssref/sel_element_element.asp) | div p | Selects all <p> elements inside <div> elements |
| Child selector  [element > element](https://www.w3schools.com/cssref/sel_element_gt.asp) | div > p | Selects all <p> elements where the parent is a  <div> element |
| Adjacent sibling selector  [element + element](https://www.w3schools.com/cssref/sel_element_pluss.asp) | div + p | Selects all <p> elements that are placed  immediately **after** <div> elements |
| General sibling selector  [element1 ~ element2](https://www.w3schools.com/cssref/sel_gen_sibling.asp) | p ~ ul | Selects every <ul> element that are preceded by a <p> element |

[Pseudo-class Selectors](https://www.w3schools.com/css/css_pseudo_classes.asp)

Style elements based on their **state**

pseudo-classes:

:link

:visited

:focus

:hover

:active

:checked

:first-child

[Pseudo-element Selectors](https://www.w3schools.com/css/css_pseudo_elements.asp)

Style a **specific part** of the **selected element**(s):

Examples:

::after

::before

::first-line

::first-letter

::selection

**Make all the links inside a list item green in unvisited state, red when visited, and pink when hover.:**

Note: a:hover MUST come after a:link and a:visited in the CSS definition in order to be effective!

a:link {

color: green;

text-decoration: none;

}

/\* Style for visited link \*/

a:visited {

color: red;

text-decoration: none;

}

/\* Style for link on hover \*/

a:hover {

color: pink;

text-decoration: none;

}

**Make the first letter of the first list item bold and 120% in size.**

ul li:first-child::first-letter {

font-weight: bold;

font-size: 120%;

}

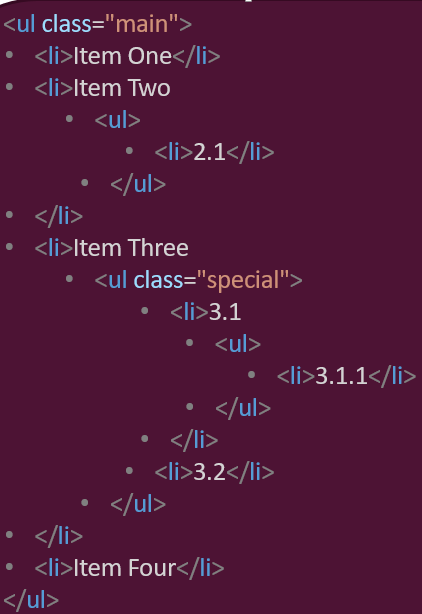
**Using attribute selector, give a yellow background to the “BCIT homepage” hyperlinked text**

a[title="BCIT homepage"] {

background-color: yellow;

}

**ul.main > li:nth-child(3) \* { color: green; }**



Position?

Float?

**flex-**grow (default: 0)

flex-shrink (default: 1)

flex: 1 0 100%;

flex-basis: 200px;

flex-grow: 1;

order: 2;

/\* with grid \*/

/\* body {

display: grid;

grid-template-columns: 1fr 3fr 1fr;

}

header {

background-color: aqua;

grid-column: 1/-1;

}

nav {

background-color: olivedrab;

}

main {

background-color: gray;

grid-column: 2/3;

grid-row: 2/3;

}

aside {

background-color: green;

}

footer {

background-color: plum;

grid-column: 1/-1;

}

footer ul {

display: flex;

list-style-type: none;

justify-content: center;

gap: 15px;

} \*/

More than 600px

Less than 600px

body {

background-color: antiquewhite;

}

nav {

border: 2px solid chocolate;

display: flex;

justify-content: space-around;

}

@media (min-width: 600px) {

nav {

flex-direction: column;

align-items: center;

/\* margin-right: 15px; \*/

/\* padding-right: 45px;

padding-left: 45px; \*/

/\* padding: 0 45px;

\*/

}

body {

/\* display: flex; \*/

display: grid;

grid-template-columns: 1fr 2fr;

gap:15px;

}

|  |  |
| --- | --- |
| *let* | when the value is needs to change at some point when you run your code.The value is mutable. |
| *const* | a variables type and value is intended to stay the same or when the value is immutable |

**Loop**

Write a piece of code to log to console powers of 2 (from 2 to 1024).

**Using for loop**

let result = 1; for (let i = 0; i < 10; i++) { result \*= 2; }

**Using while loop**

let result = 1;let i = 0;while (i < 10) { result \*= 2; i++; }console.log(result); VM1308:7 1024

**do/while Loop**

Write a piece of code using do/while loop to get user’s name and show an alert with a welcome message including user’s name. Keep asking user for an input if a number is entered instead of a string.

let name;

do {

name = prompt('Enter your name:');

} while (isNaN(name) === false);

alert(`Welcome ${name}!`);

The isNaN() function determines whether a value is NaN when converted to a number.

**Accessing Array Elements**

random[2][1]

What if I access an index that doesn’t have anything stored?

Ex: random[10]  
>>undefined

**Adding/Removing Array Elements**

**let fruits = ['Apple', 'Banana'];**

Use a built-in array method to

Add ‘Orange’ to the end of the array fruits.push('orange');

Remove the last element of the array fruits.pop();

Add ’Mango’ to the front of the array fruits.unshift('mango');

Remove the first element of the array fruits.shift();

Look up some other array methods like splice(), indexof(), slice(), etc.

array.splice (start,delete,element1,element2,...)

array.indexOf(item, start)

**Array – Activity**

let sequence = [1,1,2,3,5,8,13];

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

if (sequence[i] %2 !==0) { console.log(sequence[i]); }

}

OR

let sequence = [1,1,2,3,5,8,13];

for (let item in sequence) {

if (sequence[item] %2 !==0) { console.log(sequence[item]); }

}

OR

let sequence = [1,1,2,3,5,8,13];

for (let item of sequence) {

if (item %2 !==0) {

console.log(item); }

}

**JavaScript** [**Objects**](https://developer.mozilla.org/en-US/docs/Learn/JavaScript/Objects/Basics)

To access values of object properties:

*objectName.propertyName* e.g. person.age

*objectName["propertyName"]* e.g. person[“age”] ------ This is useful when the propertyName is a variable

**console.log this statement using the person object defined above:**

”my name is Bob Smith. I am interested in music and skiing”

**const person = {**

**name: ['Bob', 'Smith'],**

**age:32,**

**gender: 'male',**

**interest: ['music','skiing'],**

**}**

**console.log(`My name is ${person.name[0]} ${person.name[1]}. I am interested in ${person.interest[0]} and ${person.interest[1]}.`);**

**Loop through object’s properties**

**Write a loop to print out all the keys and values of this object:**

**e.g. person’s name is Celeste**

for (let key in person) {

console.log(`person's ${key} is ${person[key]}`);

}

**Write a piece of code to iterate through the “students” array and print out the number of student’s in CIT program with GPA above 3.**

let student1 = {

ID: "A00022",

GPA: 3.0,

program: "CIT"

};

let student2 = {

ID: "A01000",

GPA: 3.1,

program: "CST"

};

let student3 = {

ID: "A00114",

GPA: 3.2,

program: "CIT"

};

let students = [student1, student2, student3];

**let students = [student1, student2, student3];**

**let count = 0; // variable to keep track of the count**

**for (let i = 0; i < students.length; i++) {**

**if (students[i].program === "CIT" && students[i].GPA > 3) {**

**count++;**

**}**

**}**

**console.log(`Number of students in CIT program with GPA above 3: ${count}`);**

**for in loop**for (let key in students) {

if (students[key].program === "CIT" && students[key].GPA > 3) {

count++;

}

}

**for loop**

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

if (students[i].program === "CIT" && students[i].GPA > 3) {

count++;

}

**for of loop**

for (let student of students) {

if (student.program === "CIT" && student.GPA > 3) {

count++;

}

**Lab 7**

Create a JS file and save it in a separate folder. Add a line in course.html page to load and run the JS file (don’t forget the appropriate attribute). In your JS file:

Define a constant array called courseList. This array should have at least 3 items in it. Each item should be in the format of a JS object: with code and name properties. e.g. { code: ”ACIT 1620” , name: ” Web Fundamental Technologies”}.

Get a 4-digit number from the user e.g.“1620”. Prompt the user again if invalid (e.g. a non-numeric string or a numeric one with different number of digits) data is entered.

Write a loop to iterate through courseList array and check if the "code" property of any of the items in the array contains the provided number by the user.

If yes, log to console “Yes I am taking the course: ACIT 1620 – Web Fundamental Technologies” (use template literal to generate this string)

If not, add a new object to the array with code property equal to the value user entered, and null for the name property. Log the updated array to the console.

const courseList = [

{

code: "ACIT 1620",

name: "Web Fundamental Technologies"

},

{

code: "ACIT 1515",

name: "Scripting for IT"

},

{

code: "ACIT 1420",

name: "Database Systems"

}

];

let input;

do{

input = prompt("Enter a course code: ");

} while(isNaN(input) === true || input.length !==4);

let courseFound = false;

for (let i of courseList){

if (i.code.includes(input)){

console.log(`Yes I am taking the course: ${i.code} - ${i.name}`);

courseFound = true;

break;

}

}

if (!courseFound) {

courseList.push({ code: input, name: null });

console.log(`Added course ${input} successful`);

}

If you try to call greet(Dan) without quotes, the JavaScript interpreter will interpret Dan as a variable name and look for a variable named “Dan" to use as the argument value for the name parameter. If there is no such variable defined in the current scope, it will result in a ReferenceError.

**Hoisting – Function Declaration vs Expression**

In JavaScript, function expressions declared with const or let are not hoisted,

**Function parameters – More Than One**

**accept another parameter ”time” which is** [**default**](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Functions/Default_parameters) **to ”day”**

let greet = (name, time = "day") => { console.log(`Good ${time} ${name}!`) }

What happens if you call the function with only one value?

The function will then execute the console.log() statement with the default value of time

What happens if you call the function with too many variables? (more than declared)

If you call the greet function with more arguments than declared in the function signature, the additional arguments will be ignored, and the function will only use the first two arguments as specified in the function declaration.

**This version of the greet function can handle any number of arguments after the name and time parameters.**

let greet = (name, time = "day", ...args) => {

console.log(`Good ${time} ${name} ${args.join(" ")}!`)

}

**return statement**

Prompt user to enter a value. Write a function that takes one parameter as radius of a circle and return the circles area. Call the area calculating function with the value entered by user. Show an alert message with both values e.g. “The area of a circle with radius 2 is 12.56”

let calArea = (r)=>{

const pi = 3.14159;

let area = pi \* r \*\* 2;

alert(`The area of a circle with radius ${r} is ${area.toFixed(2)}`);

return area;

};

let r = prompt('enter a radius:');

calArea(r);

**JavaScript Object – Review**

Properties containing a function definition are **object’s methods**.

To invoke a method on the car object:

car.move()

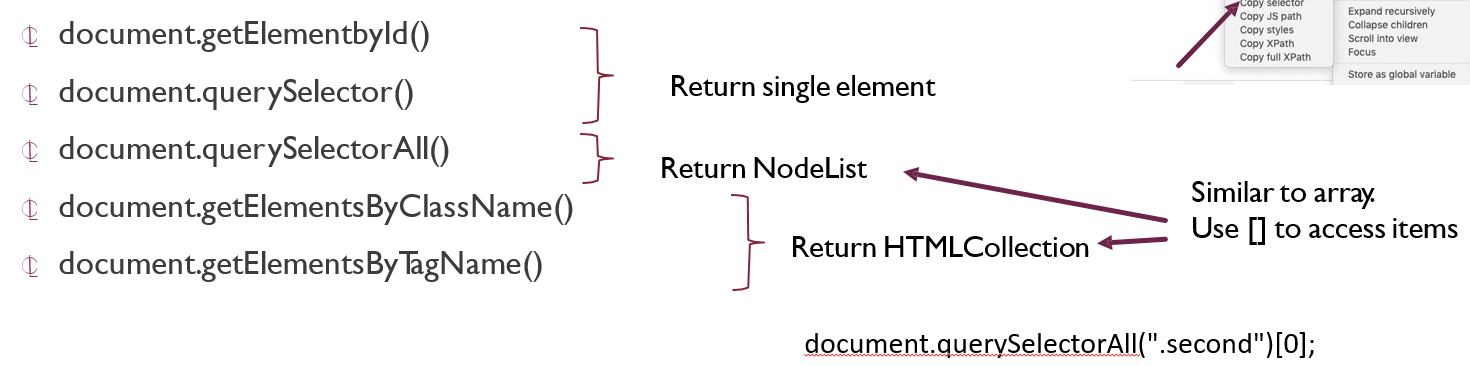
document.getElementbyId()

document.querySelector()

document.querySelectorAll()

document.getElementsByClassName()

document.getElementsByTagName()



How do we access the first paragraph?

document.getElementsByTagName("p")[0];

document.querySelector("p")

How about the second paragraph?

document.querySelectorAll(".second")[0];

document.getElementsByClassName("second")[0];

document.querySelector('p.second')

How about the <h2>?

document.querySelector("h2.second");

**document.getElementbyId() –** [**textContent**](https://developer.mozilla.org/en-US/docs/Web/API/Node/textContent)

**Update the area calculation function to access DOM elements and update their text according to their id:**

* **Update your code to use document.querySelector().**

**<body>**

**<h2>Let's try some JavaScript</h2>**

**<p id="radius">Value provided by the user</p>**

**<p id="result">Show area or Error</p>**

**</body>**

const calArea = (r) =>{

const pi = 3.14159;

const area = pi \* r \*\* 2;

const firstline = document.querySelector('#radius');

const secondline = document.querySelector('#result');

if (isNaN(r)){

firstline.textContent = `Value provided by the use is ${r}`;

secondline.textContent = `Error`;

}

else{

firstline.textContent = `Value provided by the use is ${r}`;

secondline.textContent = `Area based on the radius ${r} is ${area}`;

}

return area;

}

**Activity - Add DOM elements**

**Write a function that gets an array of strings and populate the unordered list with class=“shopping” with the array elements.**

**appendChild() -> adds a node to the end of the list of children of a specified parent node.**

**You can hard-code the array in the code.**

function populateList(shoppingList) {

// 1. select and store a ref to the ul -> querySelector

const ulList = document.querySelector(".shopping");

// 2. loop through the shoppingList -> for loop

for (let item of shoppingList) {

// 3. create an li -> createElement

const newLi = document.createElement("li");

// 4. append the li to the ul -> parent.appendChild(child)

ulList.appendChild(newLi);

// 5. update the textContent of the li with array item

newLi.textContent = item;

}

}

let myList = ["bread", "cheese", "green pepper"];

populateList(myList);

Changing [Element.classList](https://developer.mozilla.org/en-US/docs/Web/API/Element/classList) by add(), remove(), replace(), and toggle()

**Activity – Access and Change classes**

**.squareList {**

**list-style-type: square;**

**}**

**.circleList {**

**list-style-type: circle;**

**}**

const ulList = document.querySelector(".shopping");

function squareListMaker() {

ulList.classList.add("squareList");

ulList.classList.remove("circleList");

}

squareListMaker();

**DOM – Updating attributes**

**Write a JS function to find the img tag with id=“shoppingCart” and update its src, alt, width, and height attribute.You could use this link as the src:** [**https://cdn-icons-png.flaticon.com/512/263/263142.png**](https://cdn-icons-png.flaticon.com/512/263/263142.png)

function updateShoppingCart() {

const imgSource = document.querySelector("img#shoppingCart")

imgSource.setAttribute("src", "https://cdn-icons-png.flaticon.com/512/263/263142.png")

imgSource.setAttribute("alt", "shopping cart image")

imgSource.setAttribute("width", "100px")

imgSource.setAttribute("height", `${(100 \* 308) / 512}px`);

}

**DOM – Updating** [**styles**](https://www.w3schools.com/jsref/dom_obj_style.asp)

**Write a JS function to find all <li> elements, check if their text contains word “green”. If so, change their text color to green.**

function makeListGreen() {

// 1.find all the li s -> array-like structure

const listItems = document.querySelectorAll(".shopping > li");

// 2.loop through the array

// for (let i =0 ; i <listItems.length; i++)

// {

// let item = listItems[i]

// }

for (let item of listItems) {

// 3. find the textContent of each item of the array

let text = item.textContent;

//4. use .includes() to check if the text contains green

if (text.includes("green")) {

//5. update the style of that element to green text

// item.style.color = "green";

item.classList.add("greenText");

}

}

}

makeListGreen();

**Write a function, to be executed and handle the events when the event happens (in our example, displaying a message)**

//1. get a reference to the button -> querySelector

const button = document.querySelector("#clickMe");

//2. define a function to alert user

function alertUser() {

alert("you clicked!");

// button.removeEventListener("click", alertUser);

}

//3. add the function as event listener ##Update the previous activity to display the alert only the first time that the button is clicked

button.addEventListener("click", alertUser, { once: true });

**Change the background of the page to pink when the button clicked**

|  |
| --- |
| function changeBGPink() { |
|  | // document.body.style.backgroundColor = "pink"; |
|  | // document.body.style = "background-color:pink" |
|  | document.body.classList.add("pinkBG"); |
|  | } |
|  | button.addEventListener("click", changeBGPink); |

**Change the button’s text to say “clicked!” and then change the text on the button back to “Click Me!” when the button is pressed again.**

|  |
| --- |
| function changeText(event) { |
|  | console.log(event); |
|  | // if it says Click Me! change it to clicked |
|  | //else change it to Click Me! |
|  |  |
|  | if (button.textContent === "Click Me!") { |
|  | button.innerHTML = "clicked!"; |
|  | } else { |
|  | button.textContent = "Click Me!"; |
|  | } |
|  | } |
|  | button.addEventListener("click", changeText); |

**Prevent default??**

theButton.addEventListener(

\_\_\_"click"\_\_\_ , function(e){e. \_\_\_preventDefault();\_\_\_ document.body. \_\_\_style.background\_\_\_

= "blue";

});

mainHeading.addEventListener("click", () => { console.log("you clicked!");

});

~~mainHeading.addEventListener("click", function () => { console.log("you clicked!"); });~~

mainHeading.addEventListener("click", function () { console.log("you clicked!"); });

function logClick()

{

console.log("you clicked!");

};

mainHeading.addEventListener("click", logClick);

**add event handlers on the <ul> element to capture interactions with the li elements?**

**<ul id="parent-list">**

**<li id="post-1">Item 1</li>**

**<li id="post-2">Item 2</li>**

**<li id="post-3">Item 3</li>**

**</ul>**

if (event.target.tagName === "LI") {

// Code to handle click event on <li> element

console.log("Clicked on: ", event.target.textContent);

}

});

**anonymous functions are not considered equal to each other, even if their implementations are identical.**

What happens after running this code?

theElement.addEventListener('click', function myEventListeningFunction() {

console.log('howdy');

});

theElement.removeEventListener('click', function myEventListeningFunction() {

console.log('howdy');

});

Question options:

theElement will continue to listen to events(correct)

**A certain HTML element has a class "show" defined, and you can remove this class by calling .classList.remove("show"). What other call can you use instead?**

.classList.toggle("show")

**Which one(s) is(are) correct definition of a function? (select all that applies)**

Question options:

const makeNoise = function () {

console.log("Pling!");

};

() => {console.log("Pling!");}

makeNoise() {

console.log("Pling!");

}

function makeNoise() {

console.log("Pling!");

}