# WEB App Dev

[WEB App Dev 1](#_Toc12930)

[Web Scripts and Languages 1](#_Toc21386)

[Html 1](#_Toc25640)

[CSS 2](#_Toc32154)

[JS 8](#_Toc7185)

[React 16](#_Toc23093)

[WEB API 17](#_Toc1380)

[References 17](#_Toc31859)

[REST 18](#_Toc28958)

[Authentication and Authorization 18](#_Toc31365)

[Misc and Questions 18](#_Toc2381)

[MVC 19](#_Toc30830)

[Program.cs 19](#_Toc17228)

[Razor Pages 20](#_Toc6627)

[Middleware 20](#_Toc12473)

[Model 20](#_Toc7130)

[View 20](#_Toc3384)

[Controller 20](#_Toc7113)

[Routing 20](#_Toc5960)

## Web Scripts and Languages

### Html



#### Basic structure



#### Attributes

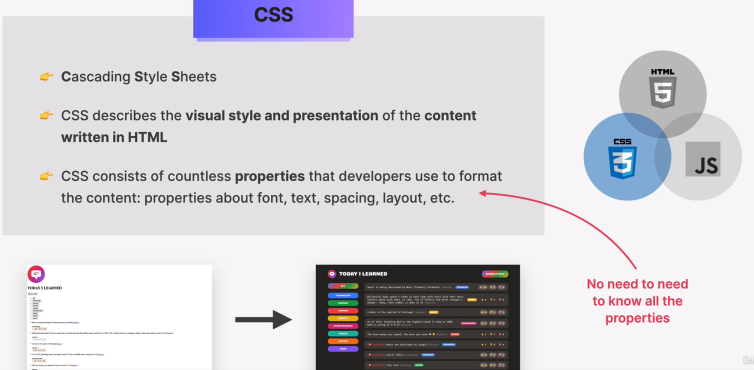
<img src="img\_girl.jpg" width="500" height="600">

Here **src**, **width** and **height** are known as attributes

#### Tags

List of tags: https://www.geeksforgeeks.org/html-tags-a-to-z-list/

### CSS





#### Ways to write CSS

Inline

<**p** style="color:#009900; font-size:50px;

             font-style:italic; text-align:center;">

        GeeksForGeeks

    </**p**>

Internal or embedded

 <**style**>

        .main {

            text-align: center;

        }

        .GFG {

            color: #009900;

            font-size: 50px;

            font-weight: bold;

        }

        .geeks {

            font-style: bold;

            font-size: 20px;

        }

    </**style**>

External

<**head**>

    <**link** rel="stylesheet" href="geeks.css" />

</**head**>

#### Selector

Dot operator is said to be class seletor. Use the while representing style for class



Below screenshot represents tag



Asterick represents **global selector**. This style will be applied for all elements.

#### Units

##### Px

Pixels are a unit of measurement commonly used in web design to define the size of various elements on a web page. The "px" unit represents a fixed-size, square area on a screen.

##### Fr

Fr is a fractional unit. Its an input that automatically calculates layout divisions when adjusting for gaps inside the grid.

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



grid-template-columns: 1fr 1fr 2fr 2fr;



##### Rem

Rem stands for root em - Relative to the font-size of the root element of the document

Em - Relative to the font-size of its nearest parent or the element itself

html {

font-size: 16px; /\* Set the base font size for the entire document \*/

}

body {

font-size: 1rem; /\* 1rem is equal to 16px in this example \*/

}

h1 {

font-size: 2rem; /\* 2rem is equal to 32px (2 times the base font size) \*/

}

p {

font-size: 1.2rem; /\* 1.2rem is equal to 19.2px (1.2 times the base font size) \*/

}

.container {

width: 50rem; /\* Width is set to 800px if the base font size is 16px (50 times the base font size) \*/

}

#### Pseudo class

Keywords that specify a special state of the selected elements. They allow you to style elements based on their state or position in the document. Pseudo-classes are denoted by a colon (:) followed by the pseudo-class name.

***li:nth-child(odd) {***

***background-color: #f0f0f0;***

***}***

***a:hover {***

***color: red;***

***}***

#### Property

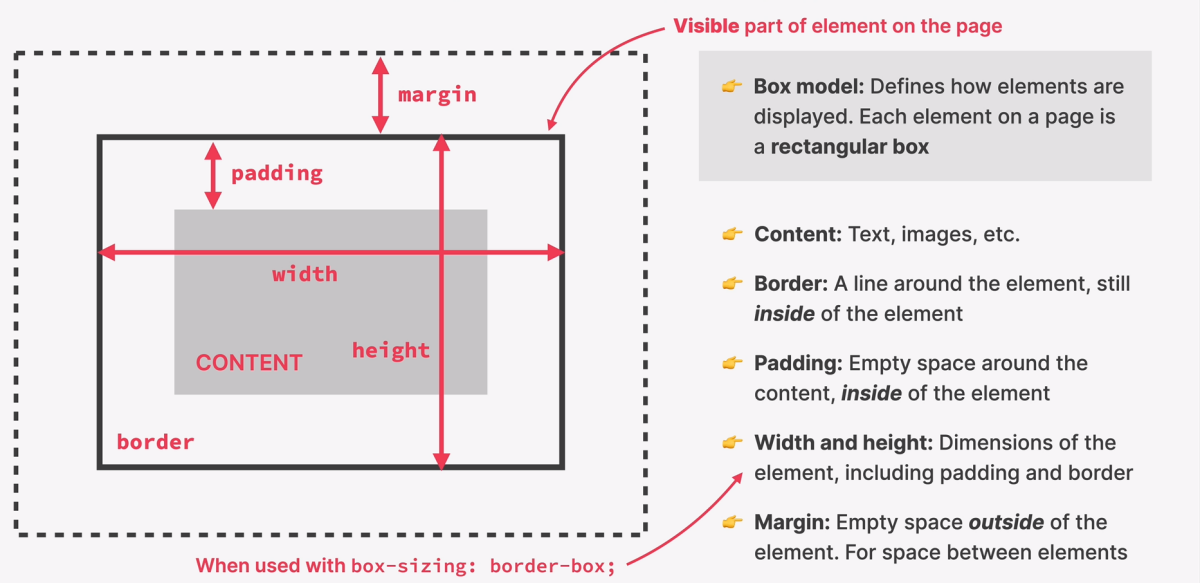
##### Transition property

***element {***

***transition: property duration timing-function delay;***

***}***

#### Box model



##### Property with multiple values

Padding: 16px 24px **(2)**

It can take values,

|  |  |  |
| --- | --- | --- |
| 2 | 3 | 4 |
| Top and bottom padding.  Left and right padding. | Top padding.  Left and right padding.  Bottom padding. | Top padding.  Right padding.  Bottom padding.  Left padding. |

Similarly for border, margin, etc

#### Block and inline elements

##### Block level

Starts on a new line and stretch the full width of their containing element. They create a "block" on the web page. Examples of block-level elements include <div>, <p>, <hx> , <ul>, <ol>, <li>, <table>, and others.

##### Inline level

Do not start on a new line and only take up as much width as necessary. They flow within the content and do not create a new "block." Examples of inline elements include <span>, <a>, <strong>, <em>, <img>

**Display: block** property can be used to change inline element to block level element

#### Layouts

##### Inline

For text

##### Block

For sections in a webpage

##### Table

For two-dimensional table data

##### Positioned

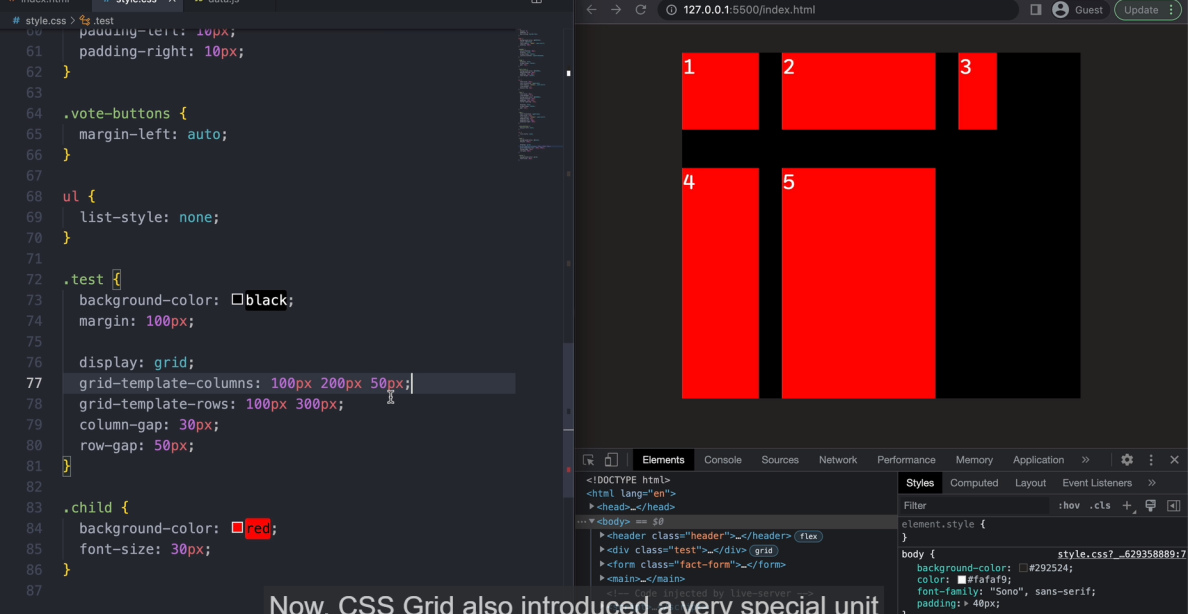
For explicit position of an element

##### Flexbox

This layout module makes it easier to design flexible responsive layout structure without using float or positioning.

##### CSS Grid

Refer test class in below screenshot for some of the example of grid related properties



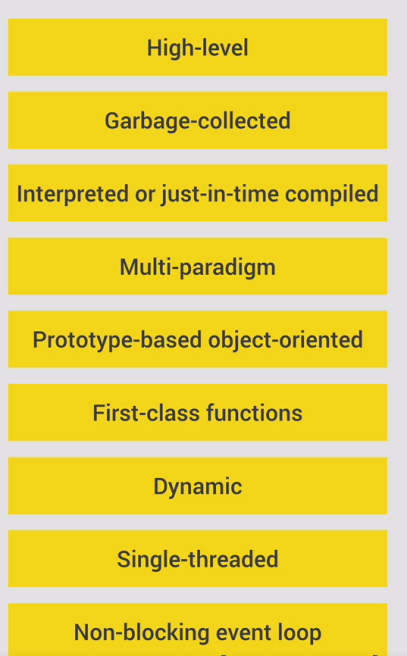
#### References

1. Use google fonts to download/get URL for required font style
2. For color palette: <https://tailwindcss.com/docs/customizing-colors>
3. Html emojis: <https://www.w3schools.com/charsets/ref_emoji_smileys.asp>
4. https://developer.mozilla.org/en-US/docs/Web/CSS

### JS

#### Definition





#### Strict mode

The "use strict"; directive is used in JavaScript to enable strict mode, which helps catch common coding errors and prevents the use of certain error-prone features.

*function myFunction() {*

*"use strict";*

*// The strict mode rules apply only to the code within this function.*

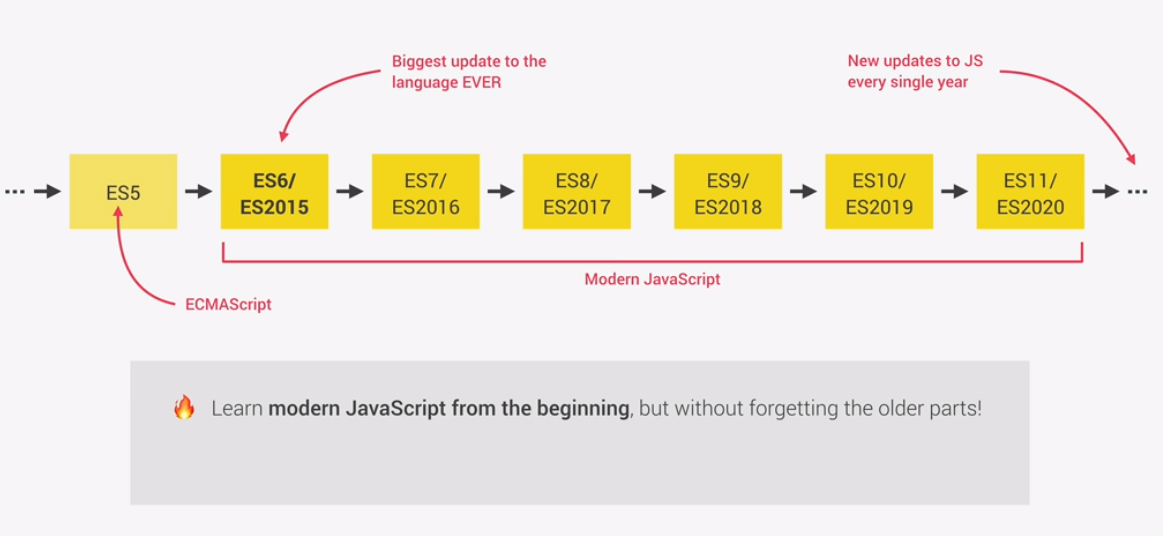
*// Your code here*

*var x = 10;*

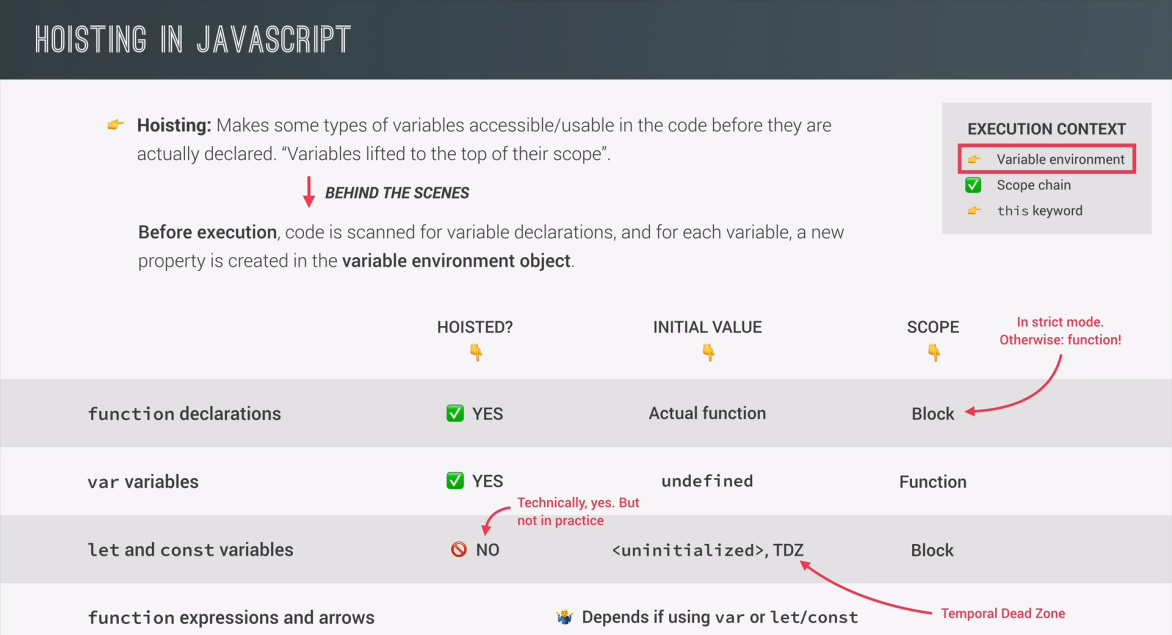
*console.log(x);*

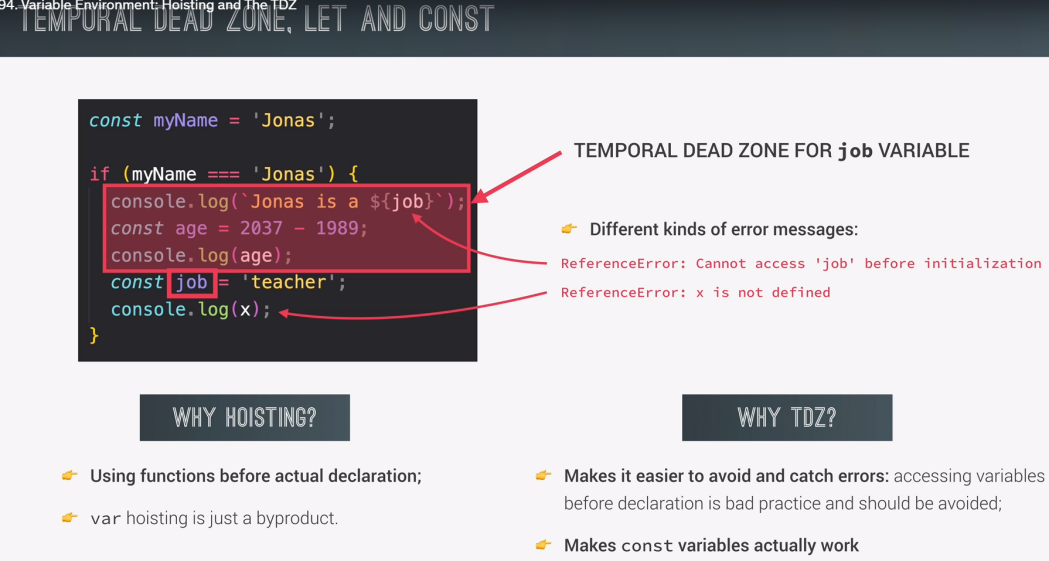
*}*

#### Versions

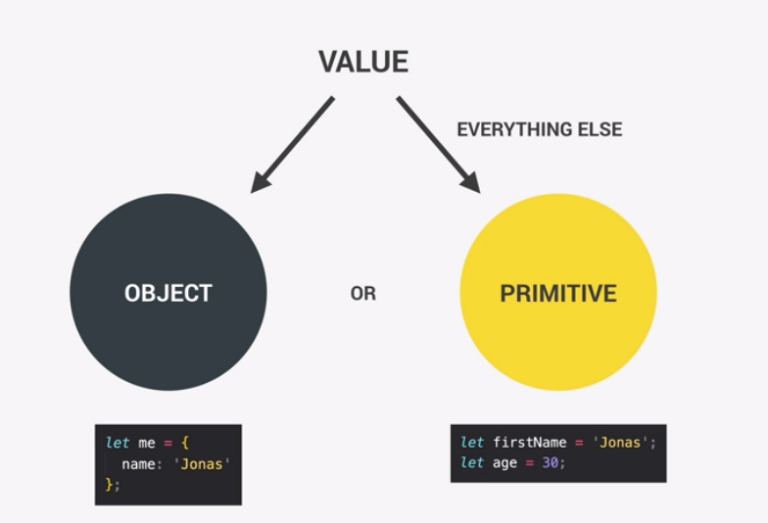


#### Hoisting





#### Types



##### Var,let, const

Age=30;//Variables without any keyword

Console.log(Age);//This will not throw any error

**Var**

Mutable

Used in older js version

Can be used before declare

Age=30;

Var Age;

**Let**

Used in modern JS

*Let age=30;*

*Age=31;//This works*

*Let withoutvalue; //This works*

**Const**

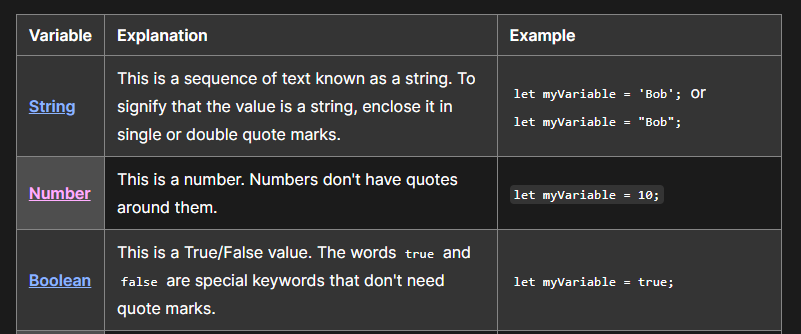
Immutable variable

*Const age=30;*

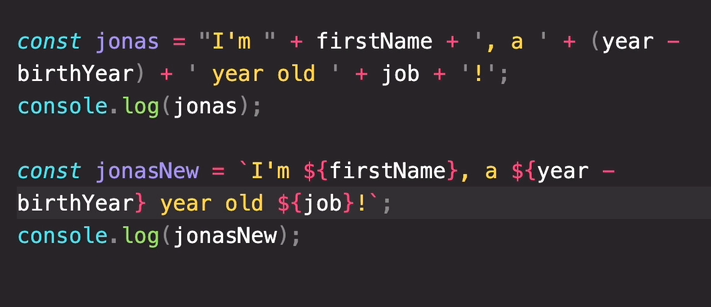
*Age=31;//This line will throw error*

*Const novalue;//This will throw error. Should have some variable.*

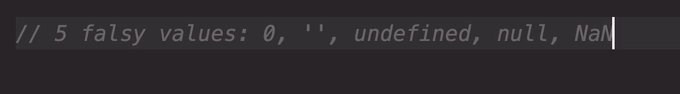
##### Primitive types



###### String



###### Bool



All other values will return true if we convert to bool

##### Non-primitive

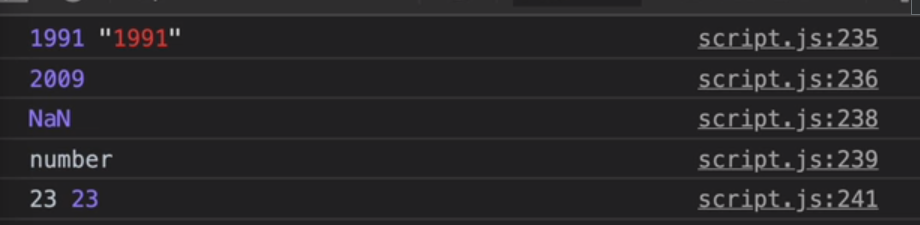
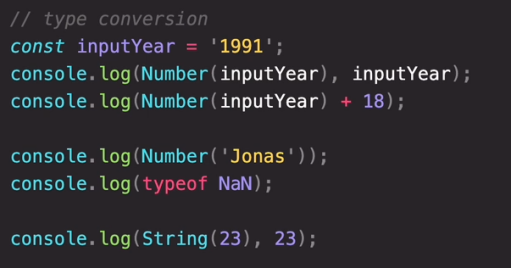
###### Array

###### Object

Used to store various keyed collections and more complex entities

##### Conversion and coercion

###### Conversion



###### Coercion

**(+) Converts number to string**

var num = "5";

var sum = num + 10; // sum is "510" (num is coerced into a string before addition)

**(-, /, \*) Converts string to number**

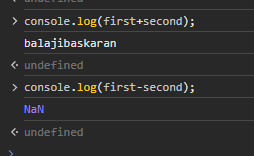
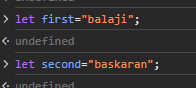
var result = "10" - 5;

console.log(result); // Output: 5 (string "10" is coerced into a number)

#### Operators

2\*\*3 represents 2 power 3

String addition and subtraction

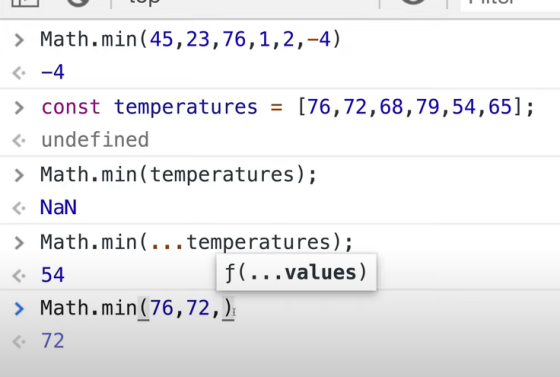


##### Spread Operator

Also known as … operator

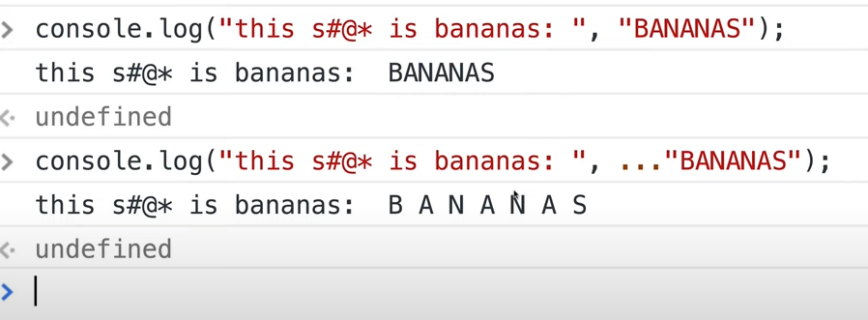
Can be used in

1. Function calls



Math.min(…) takes spread operator as parameter. So ***Zero or more numbers*** can be given as input***.***

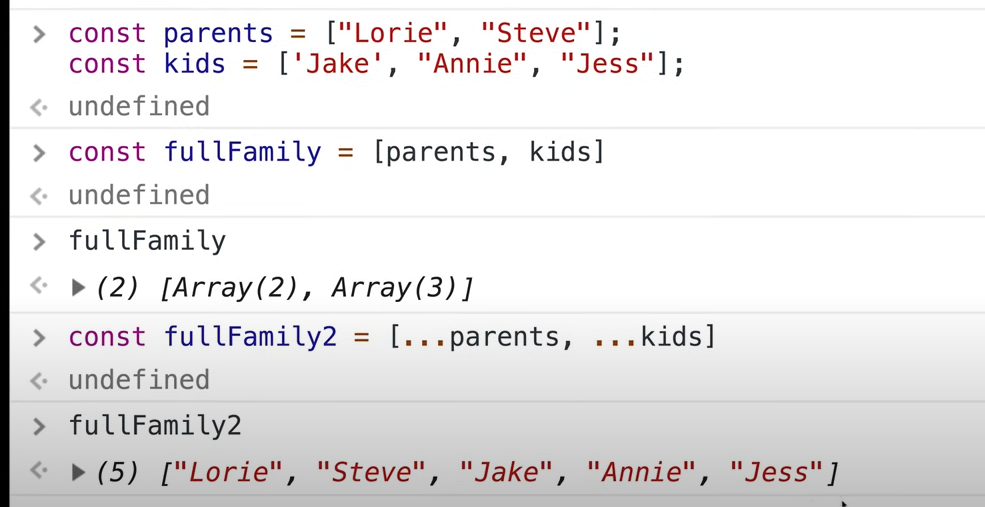




…”BANANAS” splits the string and passes each character as separate input.



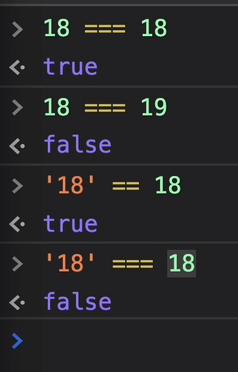
1. Array literals





1. Object literals

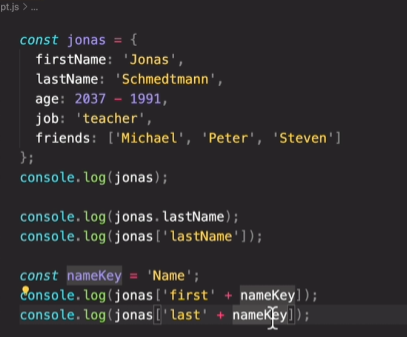
##### Equality operator



Always recommended to use ===

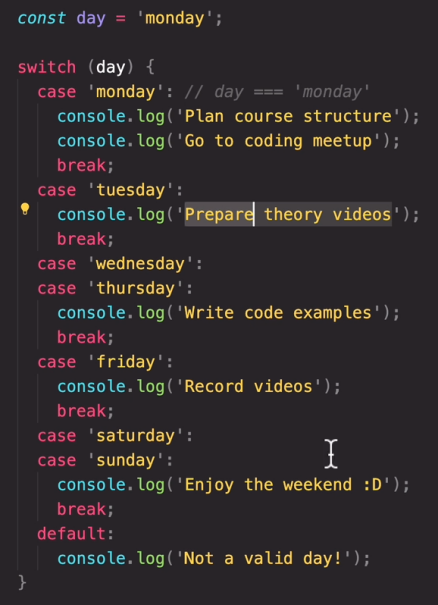
#### Dot and bracket notation

Dot and bracket notations are used to fetch the data from collection



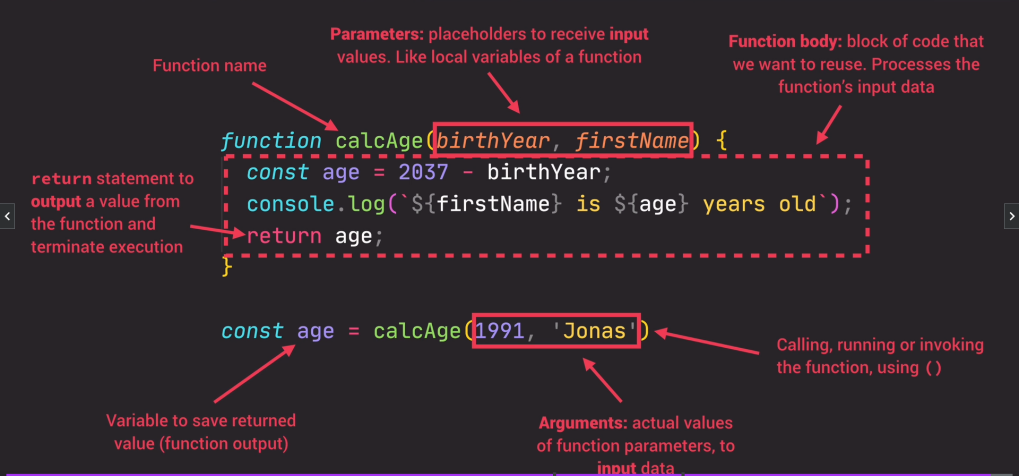
In above screenshot lastName is fetched using dot and bracket notation [‘lastname’]. String inside the bracket can be computed. Shown in last three lines.

#### Switch

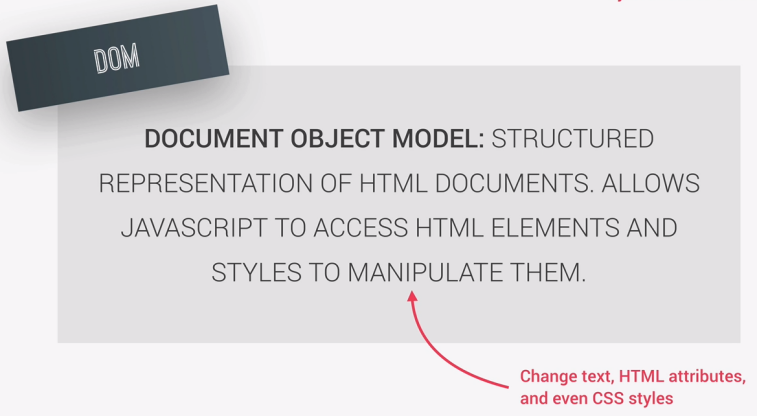


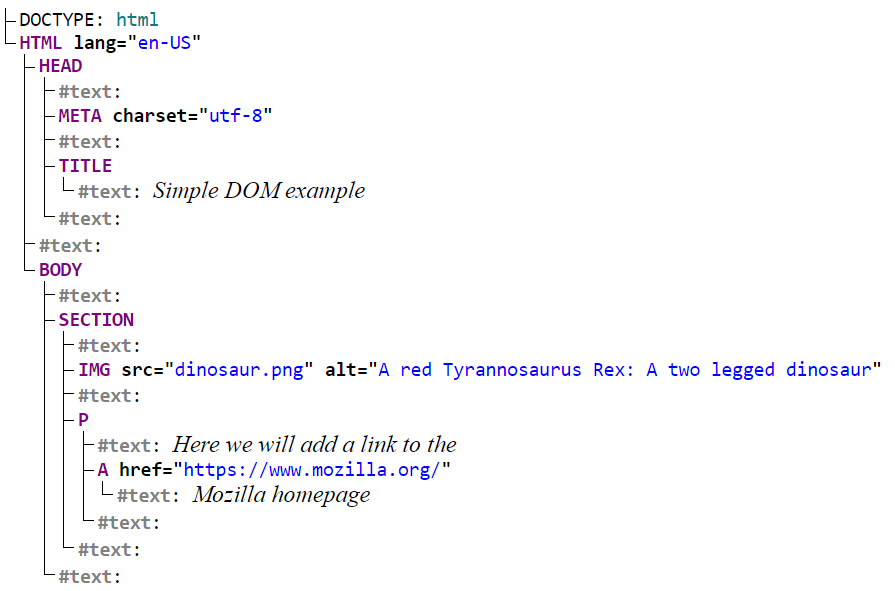
#### Function

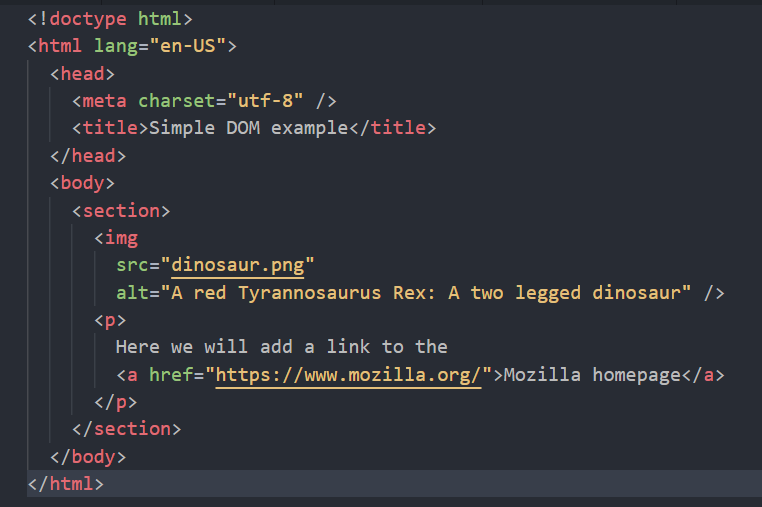
***Refer: DotnetDev\Webpages\Udemy\JsMyRef\function.js for code***



#### DOM







### React

#### Principles

##### Declerative vs Imperative

DOM Access is handled by React framework

##### Component Architecture

LEGO blocks

##### One way data flow

##### UI Library

#### Node

Provides runtime environment to run JS code out of browser

##### NVM

To switch the node versions

##### NPM vs NPX

Package manager and package execute. NPX is available from 5.2 and above, comes with NPM. NPX is used to run package binaries or commands without being installed globally.

#### JSX

JSX is a syntax extension for JavaScript that lets you write HTML-like markup inside a JavaScript file. Although there are other ways to write components, most React developers prefer the conciseness of JSX, and most codebases use it.

##### Rules

* Return a single root element
  + Group all tags under fragment<> </> or <Div>
* Close all the tags
* camelCase all most of the things!

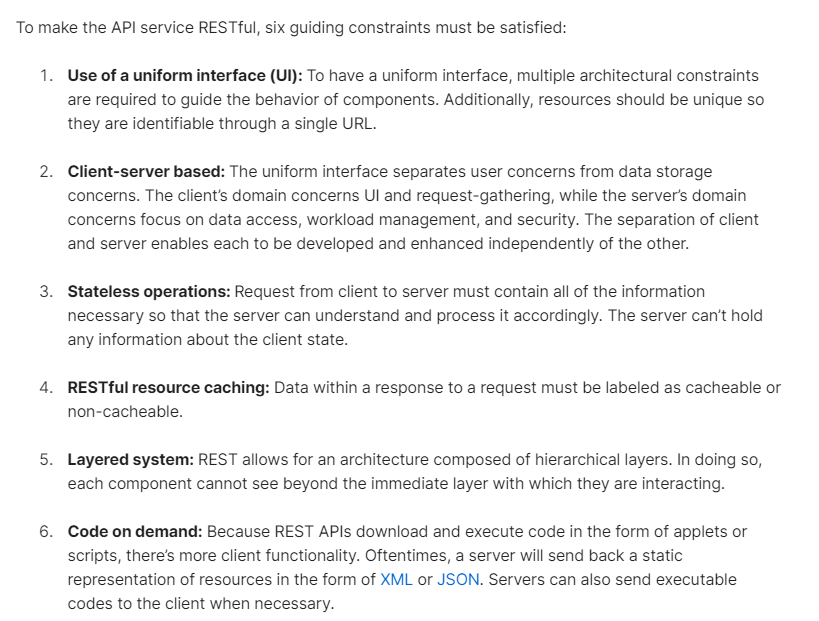
## WEB API

### References

Nick - Dometrain

### REST

#### Six constraints to make API services RESTful



Source: [Link](https://blog.postman.com/rest-api-examples/)

### Authentication and Authorization

### Misc and Questions

#### Interview Questions

#### Redis Output caching

Reference youtube video: <https://www.youtube.com/watch?v=_bg5dGnudPs>

## MVC

### Program.cs

Contains,

Services required by app are configured

Request handling pipeline -> Series of middleware components

#### Dependency injection (services)

***builder is a WebApplicationBuilder***

**builder.Services.**AddRazorPages();

**builder.Services.**AddControllersWithViews();

**builder.Services.**AddDbContext<RazorPagesMovieContext>(options =>

options.UseSqlServer(builder.Configuration.GetConnectionString("RPMovieContext")));

### Razor Pages

### Middleware

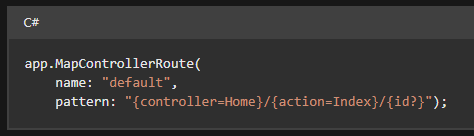
### Model

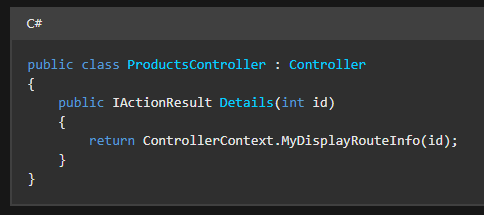
### View

### Controller

### Routing

#### Convention routing





Url: /Products/Details/5

{ controller = Products, action = Details, id = 5 }

#### Attribute routing