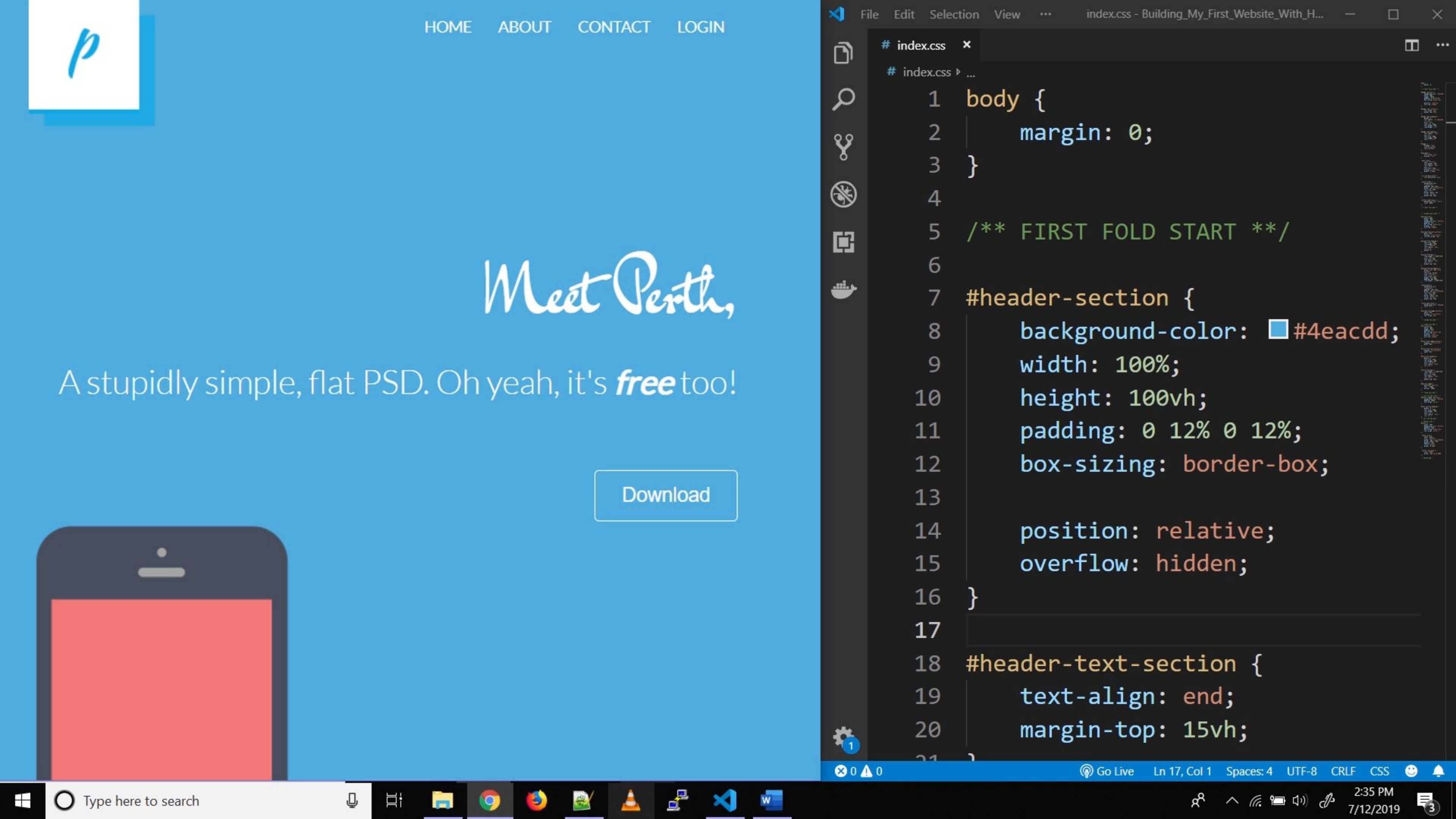
Introduction to JavaScript

Introduction to JavaScript

JavaScript is a light-weight, cross-platform, object-oriented programming language.

Together with HTML and CSS, JavaScript is one of the 3 core web development technologies.
HTML is used to show content on a webpage, CSS is used to add styles to that content and handling the layout. **JavaScript is used to add interactivity and dynamic effects** to that content. It can be used to manipulate HTML and CSS. For eg, showing a popup on a button click or showing page progress on scroll etc.

Whether you want to be a Frontend Developer (Vanilla JS, ReactJs, ReactNative) or a Backend Developer (MeteorJs, ExpressJs, NextJs). JavaScript is where you can START.



Meet Perth,

A stupidly simple, flat PSD. Oh yeah, it's ***free*** too!

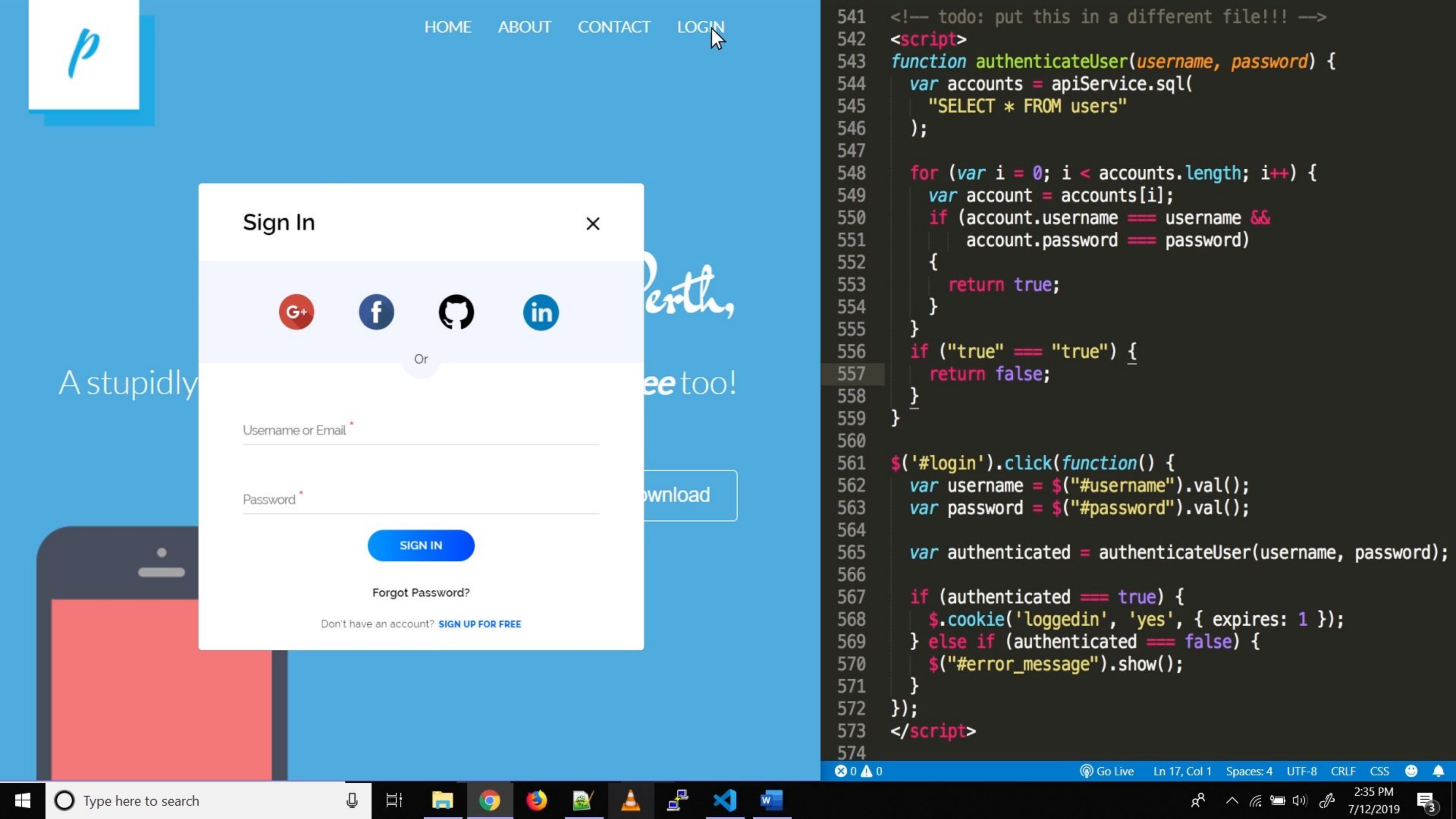
```css
body {
    margin: 0;
}

/** FIRST FOLD START **/

#header-section {
    background-color: #4eacdd;
    width: 100%;
    height: 100vh;
    padding: 0 12% 0 12%;
    box-sizing: border-box;

    position: relative;
    overflow: hidden;
}

#header-text-section {
    text-align: end;
    margin-top: 15vh;
}
```



```
<!-- todo: put this in a different file!!! -->
<script>
function authenticateUser(username, password) {
  var accounts = apiService.sql(
    "SELECT * FROM users"
  );

  for (var i = 0; i < accounts.length; i++) {
    var account = accounts[i];
    if (account.username === username &&
        account.password === password)
    {
      return true;
    }
  }
  if ("true" === "true") {
    return false;
  }
}

$('#login').click(function() {
  var username = $("#username").val();
  var password = $("#password").val();

  var authenticated = authenticateUser(username, password);

  if (authenticated === true) {
    $.cookie('loggedin', 'yes', { expires: 1 });
  } else if (authenticated === false) {
    $("#error_message").show();
  }
});
</script>
```

Building Blocks of Programming Language

These are some of the components or building blocks of a programming language:

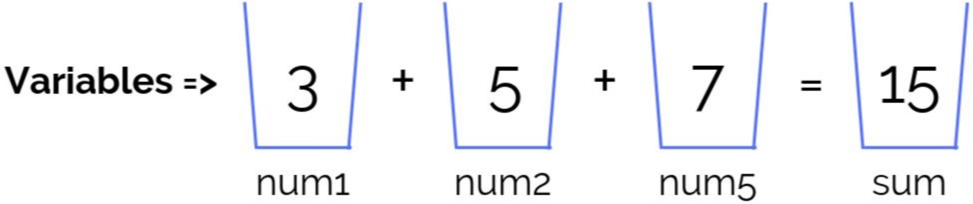
- Variables
- Data Types
- Operators
- Expressions

Variables

You can think of variables as a way to store some data for later use or in other words, think of them as containers which hold whatever data you put inside them.

Variables: Example

3 + 5 + 7 = 15

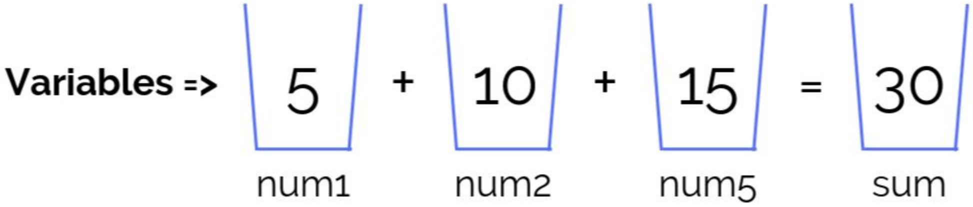


Variables => 3 + 5 + 7 = 15

num1 num2 num5 sum

Variables: Example

5 + 10 + 15 = 30



Variables => 5 + 10 + 15 = 30

num1 num2 num5 sum

How to Create a Variable?

Syntax:

```
var variable_name = value;
var variable_name;
```

For example,

```
var num1 = 10;
var mName = “John”;
```

Variable Naming Rules

Rules for naming variables.

- Names can contain letters, digits, underscores, and dollar signs.
- Names must begin with a letter.
- Names can also begin with $ and _.
- Names are case sensitive (y and Y are different variables).
- Reserved words (like JavaScript keywords) cannot be used as names.

Casing:

- Camel Case: totalSum (We use camel case for JS)
- Pascal Case: TotalSum
- Snake Case: total_sum

Variable: Practice Problems

Which of these variables have valid names? If a variable name is not valid then also explain the reason.

1. &alpha1
2. NUM1
3. DATE-5
4. TotalSum_
5. 12Byby
6. num1
7. super$$
8. Object
9. StringNew
10. $rankInClass

Operators

Operator are symbols used to perform some actions on variables. JavaScript supports many operators like:

- Arithmetic Operators
- Assignment Operators
- Comparison Operators: <, >, <=, >=, ==, ===, !=
- Conditional Operator: &&, || Ternary Operator ?=
- Logical Operators
- Type Operators: typeof and instanceof;

Expressions and Statements

Expression is a line of code which contains a valid combination of variables, values and operators and **generates** a value.

For example,

```
var num = a + b * c;
a+b+c;
a > b ? “A is larger” : “B is larger”
```

Statement is same as expression just that it **need not generate** a value.

For example,

```
for() {}
if() {}
```

Data Types

It is the type of value. JavaScript supports multiple data types:

Primitive data types:

- number - 90, 90.99
- string - “Apple”, “a”, ‘Apple’, ‘a’
- boolean - true or false
- null
- undefined
- Symbol

Data Types

Reference type

- Objects
- Arrays

Numbers

Both integers and decimal numbers are of type Number.

JavaScript Numbers are always 64-bit Floating Point where the number is stored in 52 bits (0 - 51), the exponent is stored in next 11 bits (52 - 62) and the last 1 bit (63) is for sign.

For example,

```
var num1 = 10;
var num2 = 10.3;
```

Note: Infinity and NaN are also of type numbers. You can check it by isFinite(val) and isNan(val);

Numbers: Practice Problems

Try to answer the following questions. Also, try to explain why do you think that your answer is correct. //Take 3mins for this

1. console.log(0 / 0)
2. console.log(0 / 5)
3. console.log(3 * 'a')
4. console.log('a' * 3)
5. console.log('b' + 4)
6. console.log(4 + 'b')
7. console.log(5 - 'c')
8. console.log('c' - 5)

Number Functions

We can convert our numbers to strings and strings numerals to numbers.

- variable.toString()
- parseInt(val)
- parseFloat(val)

We can also round-off these numbers

- variable.toFixed()
- variable.toFixed(2)

Numbers: Practice Problems

Try to answer the following questions. Also, try to explain why do you think your answer is correct. //Please complete this by 9:00

1. console.log(toString(100))
2. console.log(parseInt('44'))
3. console.log(parseInt('22.222'))
4. console.log(parseInt('Banana'))
5. console.log(parseFloat(12))
6. console.log(parseFloat(13.3))
7. var num1 = 10; console.log(num1.toFixed(2))
8. var num1 = 15.456789; console.log(num1.toFixed())
9. var num1 = 15.456789; console.log(number.toFixed(2))
10. var num1 = 15.456789; console.log(num1.toFixed(3))

Strings

A value written inside double quotes or single quotes is a string in JavaScript.

For example,
var mName = “John Lark”
var mProfession = ‘Cab Driver’

What if your string has double or single quotes inside it??
For example,
"I love "JavaScript"". Use \" escape sequence.
'I love 'JavaScript''. Use "\'
Another option is to switch the outer double or single quotes.

Strings: Practice Problems

Try to answer the following questions. Also, try to explain why do you think your answer is correct. //Take 2mins for this.

1. console.log('JavaScript is Cool")
2. console.log('I'm gonna learn JavaScript')
3. console.log("This is a \"test\" string")
4. console.log('This is another /'test/' string')
5. console.log("How about this string??")

Strings Functions

```
text1.toUpperCase()
text1.toLowerCase()
text1.concat(" ", text2, ...) //Concats two or more strings and returns the value.
str.trim()
str.charAt(0);

txt.split(",");        // Split on commas
txt.split(" ");         // Split on spaces
txt.split("|");
txt.split(") //Convert entire string to array of characters
```

Strings Functions: Practice Problems

Try to answer the following questions.

```
var str1 = 'Today is';
var str2 = '      a beautiful day     '
var str3 = ' In Hawaii.     '
Result = 'Today is a beautiful day In Hawaii.'
```

1. Use the str1, str2, str3 variables to create the Result string. Keep the extra spaces, lowercase and uppercase letters in mind.

```
var mStr = 'Mo Tu We Th Fr Sa Su'
```

2. Convert this string to an array which holds different days. Also, capitalize all the characters.

Strings Functions

str.length //Remember there are no parenthesis for length.

str.indexOf(subStr) //Returns -1 if not found. It is case-sensitive.

str.indexOf(subStr, fromPos) //fromPos matches the starting position of subStr.

str.lastIndexOf(subString) //Searches for the last occurence.

str.slice(start, end) //goes till position = end-1. If no value given for end then entire string is selected till the end. Also you can give negative values to start from end which starts from -1. For eg, mStr.slice(-8, -1);

str.substr(start, length)

Strings Functions: Practice Problems

Try to answer the following questions.

var mStr = "This is my test string to practice the JavaScript string function concepts. This is gonna be fun."

1. Find the length of the first sentence in the string.
2. Find the length of the second sentence in the string.
3. Find the first and last occurrence of the word "This".
4. Find the substring with length 12 from the START of the string mStr.
5. Find the substring with length 12 from the END of the string mStr.

NULL and Undefined

Undefined means a variable has been declared but has not been initialized or in another words not yet been assigned a value.

NULL is a value assigned to a variable, maybe just to represent that the variable holds no value.

NULL and Undefined: Practice Problems

```
var mNum;
console.log(mNum)
=================
var mNum = null
console.log(mNum)
```

1. What will be the result of first and second console.log().

```
console.log(undefined == null)
console.log(undefined === null)
```

2. What will be the result of first and second console.log().

DIALOGS

Dialogs in JavaScript

There are 3 different types of popups in JavaScript:

- window.alert(*text-message*)
- window.confirm(*text-message*)
- window.prompt(*text-message, default-value*)

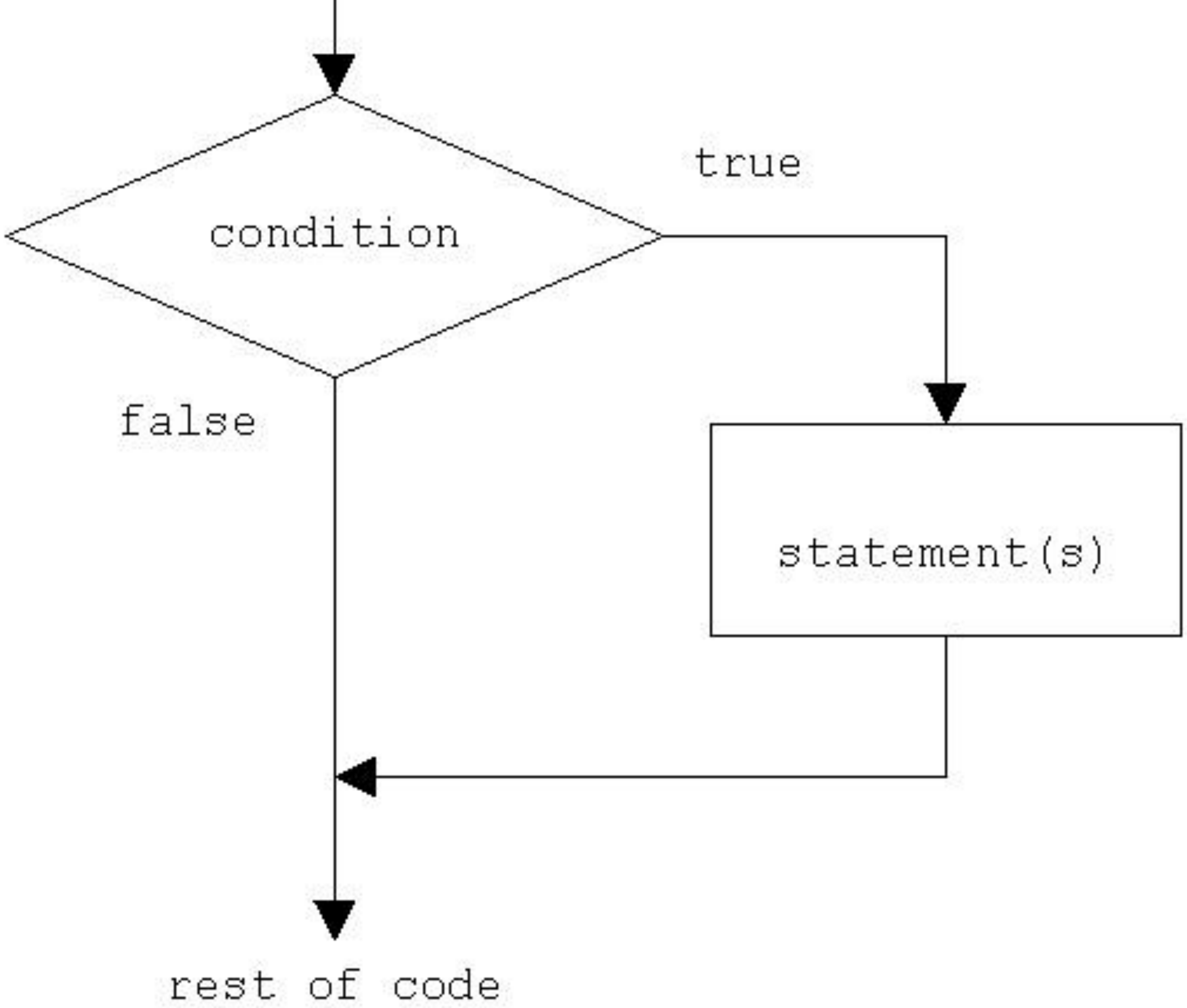
CONDITIONALS

If-else

```
if(num1 > num2) {
   //Code Block
}


if(num1 > num2) {} else {}


if(num1 > num2) {} else if {}
```


condition
true
false
statement(s)
rest of code

If-else (Practice Problems)

Answer the following questions.

Q.

var length=200;

var breadth=200;

Take these values of length and breadth of a rectangle and check if it is square or not. Change values and see if your code works or not.

If-else (Practice Problems)

Q.

var num1=5, num2=8, num3=20;

Write some code to check which number is greatest. Change values and see if your code works or not.

Q. Please finish this by 8:15 PM.

var numberOfItemPurchased = 8

A shop will give discount of 10% if the cost of purchased quantity is 1000 or more. Suppose, one unit will cost 100. Change the value of numberOfItemPurchased and print total cost for user.

If-else (Practice Problems)

Q. Please finish this by 7:42 PM

A school has following rules for grading system:

a. Below 25 - F

b. 25 to 45 - E

c. 45 to 50 - D

d. 50 to 60 - C

e. 60 to 80 - B

f. Above 80 - A

var enteredMarks = 77

Change value of enteredMarks and print the corresponding grade.

Switch Case

We use switch when based on certain known values we need to perform certain tasks.

```
switch(value) {
    case a:
        break;
    case b:
        break
    default:
}
```

Switch Case (Practice Problems)

Answer the following questions.

Q.

var monthNumber = 3;

Write a program to print name of month and number of days in a month.

Change the value of monthNumber and check if your code works fine.

Example:

monthNumber = 3

Output

Month = March

Total number of days = 31

Switch Case (Practice Problems)

Q. Please finish this by 8:09 PM.

var enteredAlphabet = 'd'

Write a program to check vowel or consonant using switch case. Change value of enteredAlphabet and check if your code works fine.

vowels: ‘a’, ‘e’, ‘i’, ‘o’, ‘u’, ‘A’, ‘E’, ‘I’, ‘O’, ‘U’

Example

enteredAlphabet: c

Output: 'c' is consonant

enteredAlphabet: a

Output: 'a' is vowel

Switch Case (Practice Problems)

Q.

var enteredNumber = 41

Write a program to check even or odd number using switch case. Change value of enteredNumber and check if your code works fine.

Example

enteredNumber: 41

Output: 41 is an odd number

OPERATORS

Arithmetic Operators

These are the arithmetic operators:

- +, // Addition
- -, // Subtraction
- *, // Multiplication
- /, // Division
- %, // Modulus
- ++, // Post and Pre Increment
- -- // Post and Pre Decrement

Arithmetic Operators (Practice Problems)

Try to answer the following questions without executing the code in codepen.

Q.

What will be printed in the console.

```
var num1 = 1 + 5;
var num2 = num1 / 4;
var num3 = 3 + 5;
var num4 = num3 / 4;
console.log(num2 + " " + num4);
```

Arithmetic Operators (Practice Problems)

Q. What will be printed in the console.

```
var a = 1;

var b = 2;

var c;

var d;

c = ++b;

d = a++;

c++;

b++;

++a;

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

Arithmetic Operators (Practice Problems)

Q. What will be printed in the console.

```
var input = 7;
var output1 = ++input + ++input + ++input;
var output2 = input++ + input++ + input++;
var output3 = input++ + ++input + input++;
console.log(output1 + ' ' + output2 + ' ' + output3);
```

Assignment Operators (Shorthand)

+=

var sum += value

This is a shorthand for var sum = sum + value

Similarly,

-=,

*=,

/=,

%=

Assignment Operators: Practice Problems

Try to answer the following questions without executing the code in codepen.

Q.

With x = 0, which of the following statements will produce a value of 1?

1. console.log(x++);

2. x = x + 1; console.log(x);

3. x += 1; console.log(x);

4. x =+ 1; console.log(x);

Assignment Operators: Practice Problems

Q.

With x = 1, which of the following statements will produce a value of 0?

1. console.log(x--);
2. x = x - 1; console.log(x);
3. x -= 1; console.log(x);
4. x =- 1; console.log(x);

Q.

What will be the result of following statements?

var num =+ 5 * 10 / 2; console.log(num);

Comparison Operators

Comparison Operators: ==, ===, !=, !==, <, <= , >, >=

Logical Operators: &&, ||, !

Ternary Operator: “ ? : ”

Comparison Operators: Practice Problems

Q.

What will be printed in the console?

```
var a = 10;
var b = 5;
var c = 12;
var e = 8;
var d;
d = parseInt((a * (c - b) / e + (b + c)) <= (e * (c + a) / (b + c) + a));
console.log(d);
if (d == 1){
        console.log((a * (c - b) / e + (b + c)));
} else {
        console.log((e * (c + a) / (b + c) + a));
}
```

Comparison Operators: Practice Problems

Q. Please finish this by 8:51 PM

What will be printed in the console?

```
var n = 2;
var p = 4;
var q = 5;
var w = 3;
if ( !((p * q) /n <= (q * w) + n/p ))
{
        console.log( ++p + w++ + " " + ++n);
}
else
{
        console.log(--p + q-- + " " + --n);
}
```

COERCION

Coercion and Types

Type coercion is conversion of a value from one type to another data type.

This can be done either programmatically by us or automatically by JavaScript. This results in two types of coercion:

- Implicit Type Coercion which is done automatically.
- Explicit Type Coercion which is done programmatically by us.

Implicit Type Coercion

Let's see some more example for conversion to string types

```
console.log(typeof('hola' + true))
console.log(typeof(null + 'hola' ))
console.log(typeof(5 + 'hola'))
```

Let's see some example for conversion to integer types

```
console.log(typeof(+'2'))
console.log(typeof(-'10'))
console.log(typeof(+'2'))
console.log(typeof(2 >= '2'))
```

Implicit Type Coercion

Let's see some examples for conversion to boolean types
if(true) {'apple'} else {'banana'}
if(10) {'apple'} else {'banana'} //anything that's not 0 or null or undefined or empty it's gonna be converted to true.

Full Boolean Reference: https://dorey.github.io/JavaScript-Equality-Table/

Implicit Coercion: Practice Problems

Q.

What will be printed in the console.

```
console.log(true + false)
console.log(12 / "6")
console.log("number" + 15 + 3)
console.log(15 + 3 + "number")
console.log(1 > null)
console.log("foo" + + "bar")
console.log('true' == true)
console.log(false == 'false')
console.log(null == '')
console.log(!!"false" == !!"true")
```

Explicit Type Coercion

Let's see some more example for conversion to string types

```
console.log(typeof(String(true)))
console.log(typeof(String(-98.13)))
console.log(typeof(String(null)))
console.log(typeof(String(undefined)))
```

Similarly, to convert a value to number datatype, we need to use the inbuilt Number() function. Let's try some examples.

```
console.log(typeof(Number('2')))
console.log(typeof(Number('-20.198')))
console.log(typeof(Number('apple')))
console.log(typeof(Number(true)))
console.log(typeof(Number(false)))
```

Explicit Type Coercion

Similarly, to convert a value to boolean datatype, we need to use the inbuilt Boolean() function.

Let's see some examples for conversion to boolean types

```
console.log(typeof(Boolean(1)))
console.log(typeof(Boolean(0)))
console.log(typeof(Boolean('abc')))
console.log(typeof(Boolean('')))
console.log(typeof(Boolean(null)))
console.log(typeof(Boolean(undefined)))
```

Explicit Coercion: Practice Problem

Q.

What will be printed in the console?

```
String(123)
String(-12.3)
String(null)
String(undefined)
String(true)
String(false)
```

Explicit Coercion: Practice Problem

Q.

What will be printed in the console?

```
Boolean('')
Boolean('Hello')
Boolean(0)
Boolean(200)
Boolean(-0)
Boolean(-200)
Boolean(NaN)
Boolean(null)
Boolean(undefined)
Boolean(false)
```

Explicit Coercion: Practice Problem

Q.

What will be printed in the console?

```
Number(null)
Number(undefined)
Number(true)
Number(false)
Number(" 12 ")
Number("-12.34")
Number("\n")
Number(" 12s ")
Number(123)
```

OBJECTS

Introduction to Objects

So far we have worked with data which of type Number or String or Boolean but what if our data is more complex like a collection of cars with their specification or the subject scores in midterm exams. We cannot store this kind of data in simple or in another word primitive data types. Lucky for us, JavaScript provides us with another data type to help us with these kinds of data, which is Objects.

Syntax:

```
{
    key: value
}
```

value ->

Properties: Number, String, Boolean, Object, Array

Method: Function

Introduction to Objects

For example,

```
var mCars = {
    p1: "350 kmph",
    gallardo: "320 kmph",
    veyron: "409 kmph",
    agera: "429 kmph"
}
console.log(mCars)
console.log(typeOf(mCars))
console.log(mCars.gallardo) //To access a value
console.log(mCars[‘gallardo’]) //To access a value
mCars.gallardo = ‘350 kmph’ //To update/add a value
```

Introduction to Objects

```
var mAgera = {
    name : "Agera",
    manufacturer : {
        "name": "Koenigsegg",
        "location": "Sweden"
     },
    topSpeed: 429,
    color: "black",
    spoilers: false,
    applyBrakes: function() { console.log("Applying Brakes");},
    getDetails: function() { console.log(this.name + “ with color “  + this.color) }
}
```

Objects: Practice Problem

Q. Please finish this by 9:22 PM
Create an object to represent an Athlete. The object should have following properties:

1. First Name
2. Last Name
3. Age
4. Sports
5. Experience
6. A method which prints the Full Name of the Athlete.
7. A method which prints all the details about the Athlete.

ARRAYS

Introduction to Arrays

Arrays are used to create lists. These are a special type of object which stores a list of data. Regular objects use keys to access data but arrays use position to access data.

For example,

```
var names = [‘John’, ‘Alice’, ‘Cara’, ‘Claire’];
```

```
var details = [‘John’, ‘Lark’, 30, ‘Race Driver’, ‘Danish’];
```

You can have any type of data in an array - Number, String, Boolean, Objects, Functions, Arrays.

Add/Delete Elements to Array

You can use position to add/update elements in an Array. But there is an issue if you add using position - It leaves holes in your array.

Two other options:

- You can resolve this by either using length.
- You can also use inbuilt push method to add elements at the end.
  mArr.push('Alan')

You can use mArr.pop(). It removes from end and returns the value.

Arrays Inbuilt Methods

mArr.sort() //sorts my list in ascending order
mArr.sort(function() {}) //Custom Sort
mArr.reverse() //inverts the list.

mArr.concat(mArr2) // Adds the passed array and returns a new array.
mArr.push(values) vs mArr.concat(mArr2)

mArr.splice(start-pos, num-of-items-to remove) // Removes items from a specific position and number of items to be removed. It updates the original array.

mArr.splice(start-pos, num-of-items-to remove, new-item1, new-item2, ...)
//Can also be used to add/replace items in the array.

Array sort with Custom Function

```
function compare(a, b) {
  var nameA = a.name.toUpperCase(); // ignore upper and lowercase
  var nameB = b.name.toUpperCase(); // ignore upper and lowercase

  if (nameA < nameB) {
    return -1; //When value is < 0 then a will come before b
  } else if (nameA > nameB) {
    return 1; //When value is > 0 then a will come after b
  }
  // a must be equal to b
  return 0; //When value is 0 then there will be no change in sequence.
}
```

mArr.sort(compare(a, b)) //Custom Sort

Array sort with Custom Function

Sort Numbers:

```
function compare(a, b) {
        return a.value - b.value; //ascending
}
```

```
mArr.sort(compare(a, b)) //Custom Sort
```

Arrays Inbuilt Methods: Practice Problems

```
var mObj = [{
    “name”: “John”,
    “score”: 98
},{
    “name”: “Alice”,
    “score”: 99
},{
    “name”: “Johnathan”,
    “score”: 90
}]
```

1. Sort the above array in ascending order based on the score.
2. Sort the above array in descending order based on the name.

Arrays Inbuilt Methods: Practice Problems

Q.

```
var mArr = [1,  2, 3, 4];
mArr.concat([5, 6, 7]);
mArr.concat(8, 9);
mArr.push(10, 11, 12);
console.log(mArr);
```

Q.

```
var mArr = [1, 2, 3, 4, 5, 6, 7, 8, 9];
```

1. Replace 3, 4 and 5 by 13, 14 and 15.
2. Remove 7, 8 and 9.

LOOPS

While Loop

Syntax:

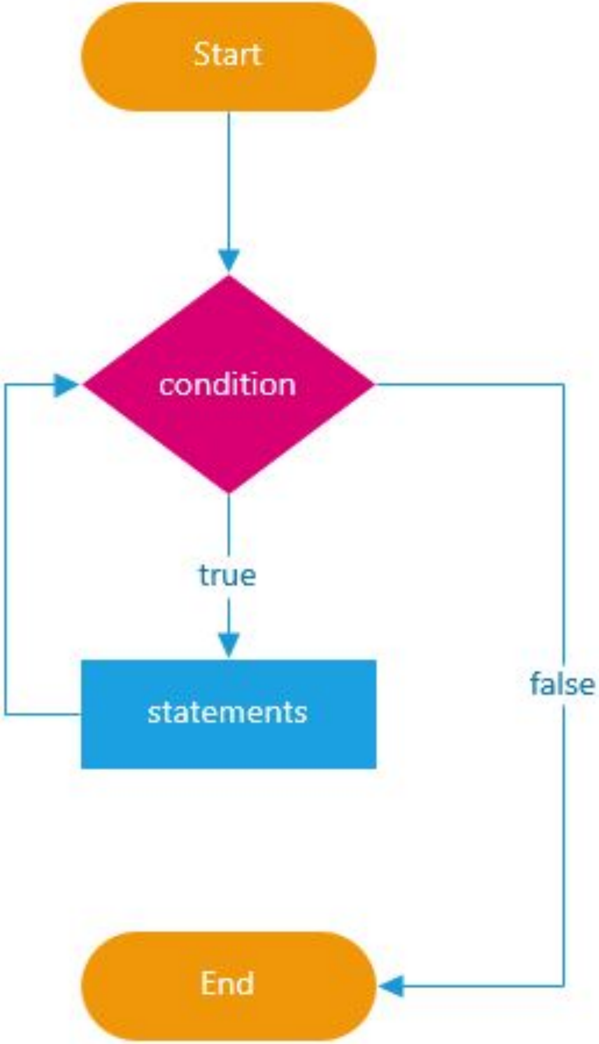
```
//initialization
while(condition) {
    //code that needs to be re-run

    //updation
}
```

Special Keywords:

Break: It takes the control flow outside the loop.

Continue: It skips the current iteration.



Start
condition
true
statements
false
End

For Loop

Syntax:

```
for (initialization; condition; post-expression) {
    // statements to re-run
}
```

Special Keywords:

Break: It takes the control flow outside the loop.

Continue: It skips the current iteration.

Loops: Practice Problems

Q. Find the smallest number from the following array: [13, 40, 10, 5, 1, 12, 24];

Q. Find the name of people who are not invited to the wedding from the following list

```
var invitationList = [
        {name: ‘Jacob’, invited: true},
        {name: ‘Alison’, invited: false},
        {name: ‘Winston’, invited: true},
        {name: ‘Lee’, invited: false},
        {name: ‘Bowry’, invited: true},
        {name: ‘Wan’, invited: false},
        {name: ‘Jovovich’, invited: true},
]
```

Loops: Practice Problems

Q. Add numbers from 1 to 100 until you get a sum more than or equal to 100.

Q. Find first 20 even numbers.

FUNCTIONS

Introduction to Functions

A function is used to put a piece of code together, so that it can be reused rather than rewriting it every time you need it.

Syntax:

```
function functionName(param1, param2, ..., paramN) {
    // block of code
}
```

To call the function we do function name followed by parenthesis. Inside the parenthesis you can pass arguments. JS doesn’t complains of we **pass extra/less arguments**. For example,

```
calculateSumOfTwoNum(10, 12);
var averageMarks = getTotalMarks(value1, value2, ...) / totalSubjects;
```

Make a Function Generate some Value

```
function getFullName (firstName, lastName) {
    return firstName + ' ' + lastName
}
```

This **return statement is used to return/generate** a value from a function. The **value can be of any data type** - Number, String, Boolean, Object, Array and Function.

Once a return statement is encountered no more lines of code are executed inside the function. In other words, a **function can only return once.**

Environment and Scope

Each variable has a scope, which is the part of the program in which the variable is visible. The variables defined outside of any function or block, the scope is the whole program - you can use these variables wherever you want. These are called **global variables**.

The variables created for function parameters or declared inside a function can be referenced only in that function, so they are known as **local variables**.

```
var num1 = 10; //num1 is a Global Variable
function sum(num2) {
    var total = num1 + num2;
    console.log(total);
}
console.log(num1)
```

Global Scope of num1

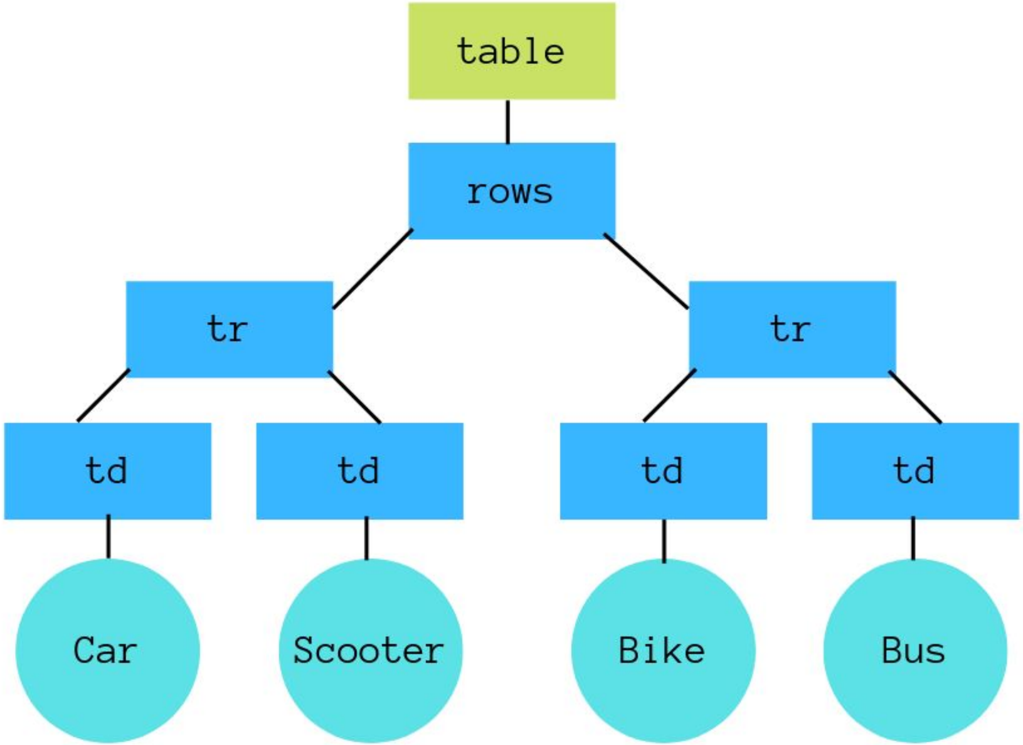
```
var num1 = 10; //num1 is a Global Variable
function sum(num2) {
    var total = num1 + num2; //total is a Local Variable
    console.log(total);                    Local Scope of num1
}
console.log(num1)                          Global Scope of num1
```

DOM

Document Object Model (DOM)

When a web page is loaded, the browser creates a Document Object Model of the page. It is the tree representation of the HTML document. In other words, the DOM is an object representation of the web page, which can be modified with JavaScript.

```
<table>
  <rows>
    <tr>
      <td>Car</td>
      <td>Scooter</td>
    </tr>
    <tr>
      <td>Bike</td>
      <td>Bus</td>
    </tr>
  </rows>
</table>
```



DOM Tree Structure

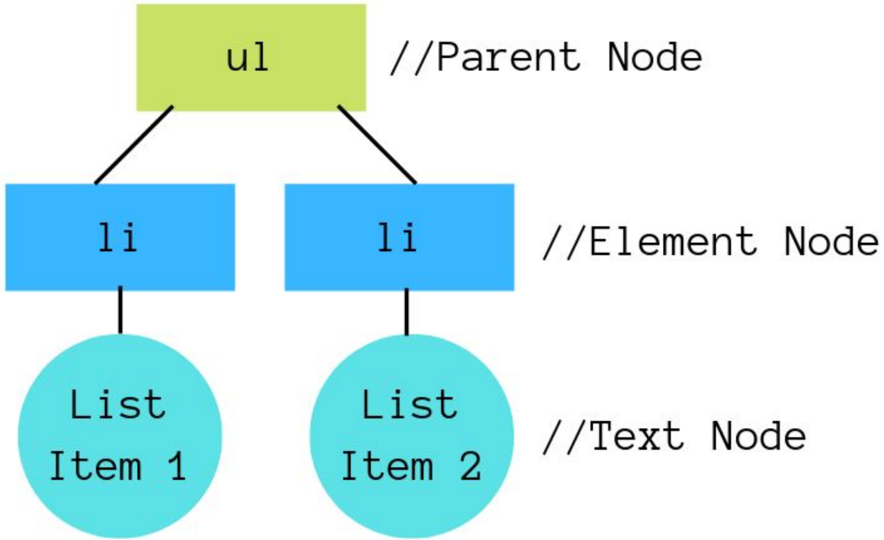
The top node of a tree is called as root node.

Every node has exactly one parent, except the root (which has no parent)

A node can have any number of children.

Siblings are nodes with the same parent.

```
<ul>
  <li>List Item 1</li>
  <li>List Item 2</li>
</ul>
```



ul //Parent Node

li li //Element Node

List Item 1 List Item 2 //Text Node

JavaScript and DOM

Now where does JavaScript fits in all this.

- JavaScript can add new HTML elements and attributes.
- JavaScript can change any HTML elements and attribute in the page
- JavaScript can change all the CSS styles in the page
- JavaScript can remove existing HTML elements and attributes
- JavaScript can add and listen to HTML events like onClick, onScroll etc.

Selecting HTML Elements using JS

document.getElementById("") => It returns the DOM object for the HTML element selected based on the “id” attribute.

document.getElementsByClassName("") => It returns a HTMLCollection list with DOM objects for the HTML elements selected based on the “class” attribute.

document.getElementsByTagName("") => It returns a HTMLCollection list with DOM objects for the HTML elements selected based on the “tag”.

Query Selectors

Query Selector allows you to use CSS-like selectors to select HTML elements. It is the new way in JavaScript to select HTML elements.

There are 2 such selectors:
The **document.querySelector()** selects the first item it finds. Say for example, you want to select the first item with a certain class. Instead of returning all the HTML elements matching that class, it returns only the first one.

The **document.querySelectorAll()** selects all the HTML elements matching the selector. Say for example, you want to select all the items with a class. You can use querySelectorAll()

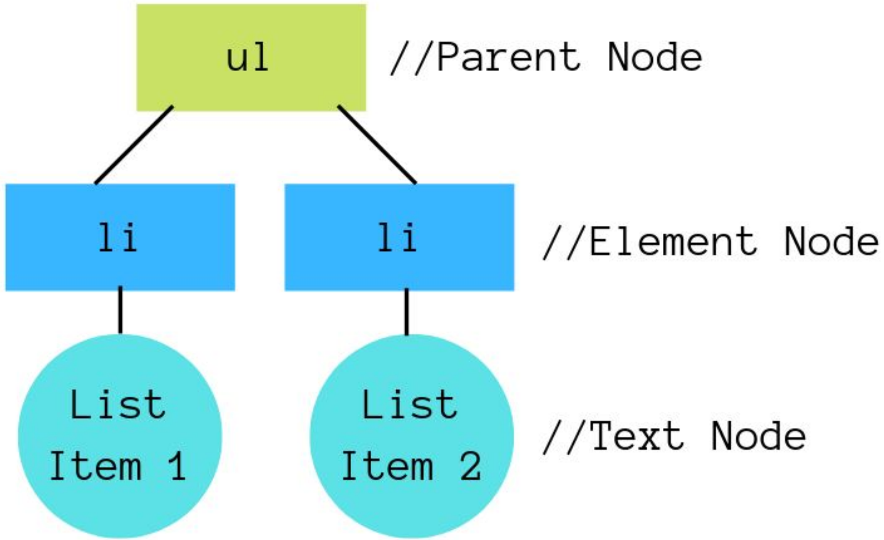
NOTE: Dont forget “#” and “.”

querySelectors vs getElementByX

Browser Support:
Because **querySelector** and **querySelectorAll** is **relatively new**, it is not supported by IE 7 and below. So, if your users will be using your website on IE7 and below then you should not use querySelector and querySelectorAll but instead use getElementById and getElementByClass.

Execution Speed:
getElementById() and getElementsByClassName() are more than twice as fast as querySelector() and querySelectorAll().
If you just want to select one element and that element has an ID then use getElementById.
Similarly, if you want to select multiple elements based on CSS selectors, then you can use querySelectorAll().

```
<ul>
  <li>List Item 1</li>
  <li>List Item 2</li>
</ul>
```



Creating New Elements Dynamically

```
document.createElement("tag-name");

document.createTextNode("The text you need");

parentNode.appendChild(childNode); //Appends the childNode as the last node.
```

Creating a Countdown Timer

HINT:

```
setInterval(function() {
   //Code to be executed at interval
}, time-interval)
```

Adding/Updating Styles

To update styles we can use the style property of DOM. Below is the syntax to add or update styles of an HTML element.

```
var selectedElement = document.getElementById(id);
selectedElement.style.{propertyName} = "New Value"
```

For eg,

```
selectedElement.style.width = "300px";
```

How do you update a list of selected elements??

NOTE: All the CSS property names get converted to camelCasing.

Adding/Updating Styles

Well, 2 things to notice here:

- All these **styles are added as inline-styles**. So if you inspect your HTML Elements, it's visible here.
- There's a small limitation, **you can only update those styles which can be written as inline-styles**. What this means is that you can update styles like color, margin, fontSize etc but you cannot update styles for pseudo classes and pseudo elements.

Adding/Updating Classes

You can use the className property of DOM to add/update the class. But there is a small limitation, it updates all the classes with the latest values.

To fix this issue you can use classList property provided by DOM.
To add a class you can do:

```
document.getElementById("MyElement").classList.add('MyClass');
```

To remove a class you can do:

```
document.getElementById("MyElement").classList.remove('MyClass');
```

Getting Style Values

There are 2 ways to get styles depending on how you have added styles.

- For inline styles, you can simply do selectedElement.style.inline_property_name.
- If you have styled elements using a CSS file then you need to use this method **window.getComputedStyle(selectedElement).property_name**.

Creating Elements from a List of Data

You can simply iterate the list and generate HTML elements based on the current array item.

EVENTS

What is an Event??

When some action happens in the webpage, that action is called event.

These are some examples of events:

- The user clicking the mouse over a certain element.
- The user hovering the cursor over a certain element.
- The user pressing a key on the keyboard.
- The user resizing or closing the browser window.
- A web page finishing loading.
- A form being submitted.
- A video/audio being played, or paused, or finished playing.

Event Listeners and Handlers

You can make your webpage react to these events. Say for eg, when someone clicks on a button. Show popup. or when someone clicks the escape button, hide all popups, or when video is finished playing then play the next video etc etc. **This reacting to events is called listening to events and the function that runs on the reaction is called event handler.**

How to listen to Events??

There are two ways you can listen to events.

- Using the event handler properties.
- Using addEventListener() method.

There are two ways we can handle events:

var selectedElement = getElementById('selector')/querySelector('selector')

Method 1: Using **Events Properties**

```
Syntax: selectedElement.{eventProperty} = function() {}
For example: selectedElement.onclick = function() {}
```

Method 2: Using **addEventListener()**.

```
Syntax: selectedElement.addEventListener({eventName},function() {})
For example: selectedElement.addEventListener('click', function() {})
```

Event Property vs addEventListener()

You can add same event multiple times to the same element using addEventListener() but it’s not possible with event property.

For eg,

```
selectedElement.onclick = function() {alert(‘Event 1’)}
selectedElement.onclick = function() {alert(‘Event 2’)} //Event 2 will override Event 1
```

```
selectedElement.addEventListener(‘click’, function() {alert(‘Event 1’)})
selectedElement.addEventListener(‘click’, function() {alert(‘Event 2’)}) //Both Event 1 and 2 alert will be shown.
```

NOTE: addEventListener doesn’t works in Internet Explorer < v9 but event property does.

ClassList Methods: contains() and toggle()

The **contains()** method returns true/false based on whether a class is present in the class list.

For eg,

```
selectedElement.classList.contains('MyClass')
```

The **toggle()** method on subsequent calls adds/removes a class from the classList.

For eg,

```
selectedElement.classList.toggle('MyClass')
```

NOTE: The classList property is supported only by modern browsers, it is not available in IE<v9

KEY EVENTS

Key Events

Using these events you can listen to key clicks.
There are 3 different events:

- keydown: ANY key is pressed.
- keyup: ANY key is released after the keydown event.
- keypress: ANY key except Alt, Shift, Ctrl, Fn, CapsLock is in pressed position.

Key Events: Event Object

When the event is keyboard event. It also gives access to some new properties:

- **e.key** - This key represents which keyboard key was clicked.
- **e.which** - It returns the ASCII code for the key pressed.
- **e.altKey** - This returns boolean based on whether alt key was pressed or not.
- **e.ctrlKey** - This returns boolean based on whether control key was pressed or not.
- **e.shiftKey** - This returns boolean based on whether shift key was pressed or not.

You can check codes for any key here => https://keycode.info/

FORM EVENTS

Form Events

These are some of the available form events:

- **change**: when the content of an input field is changed then this function is called or when the user selects a value from the dropdown etc. So basically, whenever the value changes for an input element then this change event is triggered. It is triggered once the element loses focus.
- **input**: It works same as change. The only difference is that it’s triggered as soon as the value is changed. It doesn’t waits for the element to lose focus.
- **focus**: The focus event is triggered when the input field is focused by the user.
- **blur**: The focus event is triggered when the input field loses focus.
- **submit**: The submit event is triggered when the submit button is clicked by the user.

Event Object: preventDefault()

When you click on this submit button the input field key value pairs are appended to the URL. This is the default behavior, say you did not this to happen. Instead of automatically updating the URL, you want to send this data to the backend using a REST API. Now, to avoid this default behavior, the event object has another method called **preventDefault().**

The “insertBefore()” Method

```
parentElement.insertBefore(childElement, insertBeforeWhichElement);
```

This method accepts 2 params, first is the newly created element and second is the sibling element before which you want to add you newly created element.

The “remove()” Method

To remove HTML elements from the DOM you can use the **remove()** method.
Syntax:
selectedElement.remove();

NOTE:
In IE<v9 the remove() method is not supported. You can use the removeChild() method.
Syntax:
parentElement.removeChild(childElement);

MOUSE EVENTS

Mouse Events

These are some of the available mouse events:

- **mousedown** event is triggered when either the left or right (or middle) mouse key is pressed.
- **mouseup** event trigger when either the left or right (or middle) mouse is released after the mousedown event.
- **click** event is triggered when the left mouse button is pressed and released, it requires the Mousedown and Mouseup event to happen before Click event. The normal expect trigger order: onMousedown >> onMouseup >> onClick
- **dblclick** event is triggered when the left mouse button is clicked twice.

Mouse Events

- **mouseenter** when the mouse enters an element.
- **mouseleave** when the mouse is moved out of an element.
- **mousemove** when the mouse is moved over the element.
- **contextmenu** when the context menu is opened, e.g. on a right mouse button click

Mouse Event Object Properties

- **clientX and clientY:** Relative to the top left of the physical screen/monitor. This reference point is below the URL bar and back button in the upper left.
- **pageX and pageY:** Relative to the top left of the fully rendered content area in the browser. This reference point is below the URL bar and back button in the upper left. It includes the scroll distance.
- **screenX and screenY:** Relative to the top left of the physical screen/monitor. It includes the URL Bar etc.

TIMING EVENTS

Timing Events

There are two timing methods available:

- **setTimeout**(function() {}, time-in-millisecs) //Executes the callback function after the specified time.
- **setInterval**(function() {}, time-in-millisecs) //Executes the callback function periodically after the specified time.

NOTE: To stop a timer function you can use the clearInterval() method.

Events Full Reference

https://developer.mozilla.org/en-US/docs/Web/Events

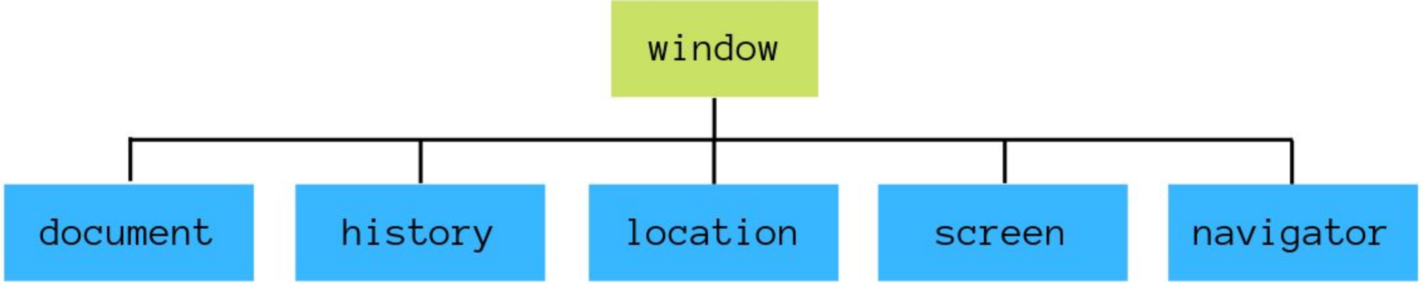
Please find 3 applications of all the events that we have learnt so far!!

BOM

Browser Object Model

BOM (Browser Object Model) represents the current browser window.
The main purpose of Browser Object Model is to manipulate the properties and methods which are associated with the browser window.

The Browser Object can be accessed using the **window** object.



window

document

history

location

screen

navigator

Browser Object Model

These are the main objects which are available inside the Browser Object Model:

- **document:** represents the webpage structure and content in a tree format, where the tree nodes represent different HTML elements and text content.
- **history:** holds the data about the URLs that have been visited by the user.
- **location:** contains information about the current URL.
- **screen:** holds information about the browser screen like width and height.
- **navigator:** contains information about the user's browser.

How to access the BOM Properties & Methods?

```
window.document.getElementById('topbar')

document.getElementById('topbar')



window.alert('This is window')

alert('This is without window')
```

The “innerWidth” and “innerHeight” Property

This window object also gives us access to some other properties like **innerWidth** and **innerHeight**

- **window.innerHeight:** the inner height of the browser window (in pixels). This is the current height of the part of browser window which shows web page content.
- **window.innerWidth:** the inner width of the browser window (in pixels)

NOTE: This property is not available in Internet Explorer with version 8 and less so for those you can use
document.body.clientHeight
document.body.clientWidth

Some methods provided by BOM

The window object also gives us access to some methods like:

- **setTimeout():** performs action after specified time.
- **setInterval():** performs an action periodically after specified time.
- **alert():** displays the alert box containing message with ok button.
- **confirm():** displays the confirm dialog box containing message with ok and cancel button.
- **prompt():** displays a dialog box to get input from the user.
- **open(url):** opens the entered URL in a new tab.
- **close():** closes the current window.

SCREEN OBJECT

Intro to Screen Object

The screen object holds information of the browser screen. It can be used to display screen width, height, colorDepth, pixelDepth etc.

- **screen.width:** property returns the width of the visitor's screen in pixels.
- **screen.availWidth:** property returns the width of the visitor's screen, in pixels, minus interface features like the Windows Taskbar.
- **screen.height:** property returns the height of the visitor's screen in pixels.
- **screen.availHeight:** property returns the height of the visitor's screen, in pixels, minus interface features like the Windows Taskbar.

NAVIGATOR OBJECT

Intro to Navigator Object

The window.navigator object contains information about the browser where your application is running.

userAgent: property returns the user-agent string. Every time your web browser makes a request to a website, it sends a HTTP Header called the "User Agent". The User Agent string contains information about your web browser name, operating system, device type and lots of other useful bits of information. You can parse this userAgent string to get information about the browser and operating system.

Intro to Navigator Object

Now, suppose your application needs to check whether the user has an active internet or not then you can use the **onLine** property of navigator object. The onLine property returns true if the browser is online:

The navigator object also gives us access to another property called **connection**. This connection property can be used to get current downlink speed and network type etc.

HISTORY OBJECT

Intro to History Object

The history object represents an array of URLs visited by the user. It gives us access to load previous and next page.

back() //loads the previous page.
forward() //loads the next page.

LOCATION OBJECT

Intro to Location Object

The **window.location** object can be used to get the details of the current page address (URL) and also to redirect the browser to a new page.

location.href: It returns the URL of the current page.
"https://jsbin.com/cuhoder/edit?js,console"

location.hostname: It returns the domain name.
"jsbin.com"

Intro to Location Object

location.pathname: It returns the pathname of the current page.
"/cuhoder/edit"

window.location.protocol: It returns the web protocol of the page.

location.search: It returns the search params in the URL.
"?js,console"

window.location.assign: It is used to load a new page in the same browser tab.
window.location.assign("https://www.google.com")

WEB STORAGE

HTML5 Web Storage

The HTML5's web storage feature lets you store some information locally on the user's computer, similar to cookies, but it is faster and much better than cookies.

There are two types of web storage, which differ in scope and lifetime:

- Local storage — The local storage uses the **localStorage** object to store data for your entire website on a permanent basis. That means the stored local data will be available on the next day, the next week, or the next year unless you remove it.
- Session storage — The session storage uses the **sessionStorage** object to store data on a temporary basis, for a single browser window or tab. The data disappears when session ends i.e. when the user closes that browser window or tab.

Local Storage

To access local storage of a browser you can use the localStorage object provided by window object. A different localStorage object is created for different host.

The data is stored in a key/value pair. The key identifies the name of the information (like 'first_name'), and the value is the value associated with that key (say John). **All the keys and values are stored as strings.**

```
localStorage.setItem(‘key-name’, value); //To store a value in local storage.
localStorage.getItem(‘key-name’) //To get a value from local storage.
localStorage.removeItem(‘key-name’) //To remove a value from local storage.
localStorage.clear() //To clear the local storage for the host.
```

Web Storage vs Cookies

The information stored in the web storage isn't sent to the web server as opposed to the cookies where data sent to the server with every request.

Also, where cookies let you store a small amount of data (nearly 4KB), the web storage allows you to store up to 5MB of data.

SCROLL EVENT

Scroll Event

It is triggered when the page is scrolled.

```
document.onscroll = function() {
    console.log(document.documentElement.scrollTop);
}
```

NOTE: For Safari you can use **document.body.scrollTop** because document.documentElement.scrollTop doesn’t works.

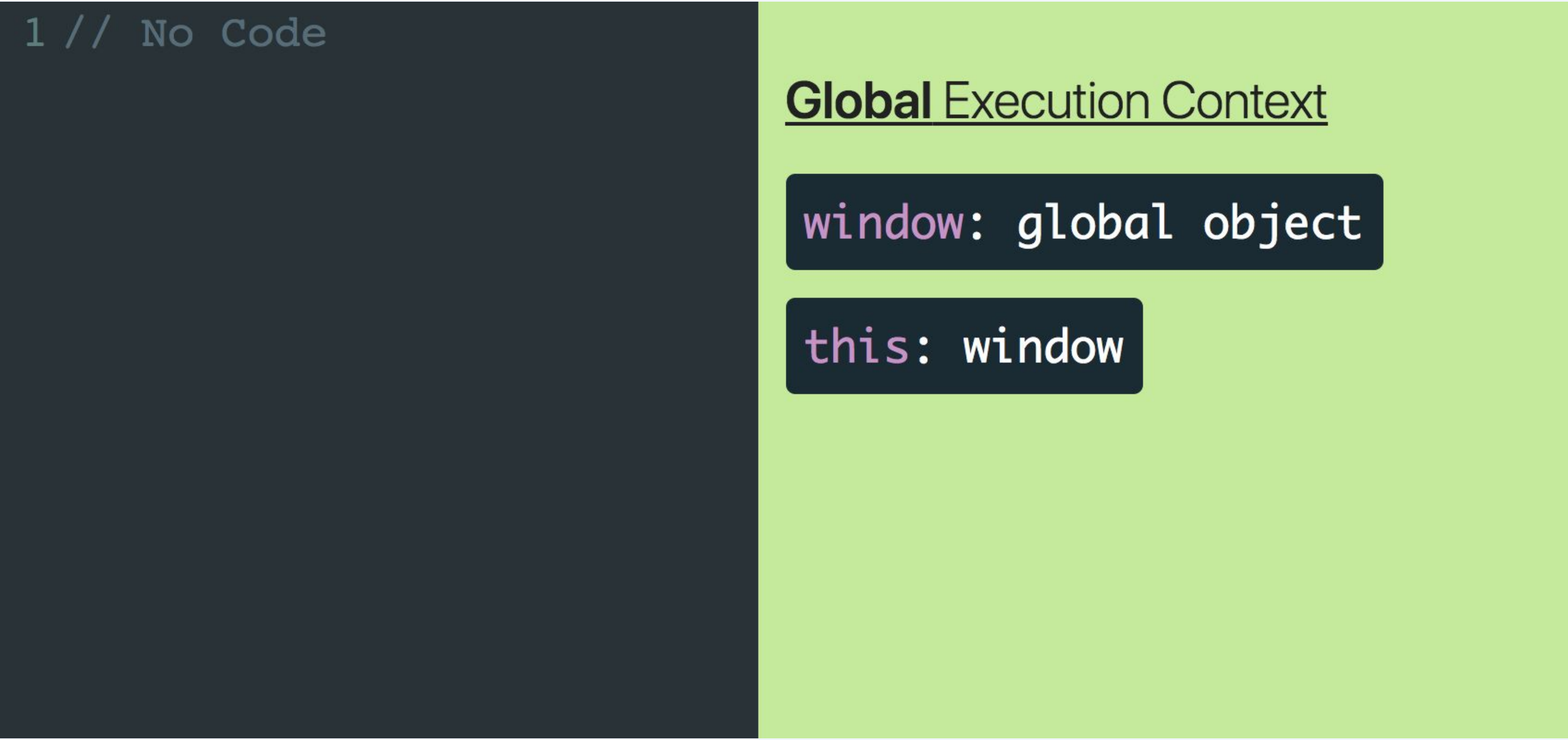
CONTROL FLOW

Execution Context

Execution context (EC) is defined as the environment in which JavaScript code is executed. The value of “this” object, variables, objects, and functions a piece of JavaScript code has access to constitutes its environment.

The first Execution Context that gets created when the JavaScript engine runs your code is called the “Global Execution Context”. Initially this Execution Context will consist of two things - a global object and a variable called this. this will reference the global object which will be window if you’re running JavaScript in the browser or global if you’re running it in a Node environment.

The “this” refers to the execution context from where the function is called.



```
// No Code
```

Global Execution Context

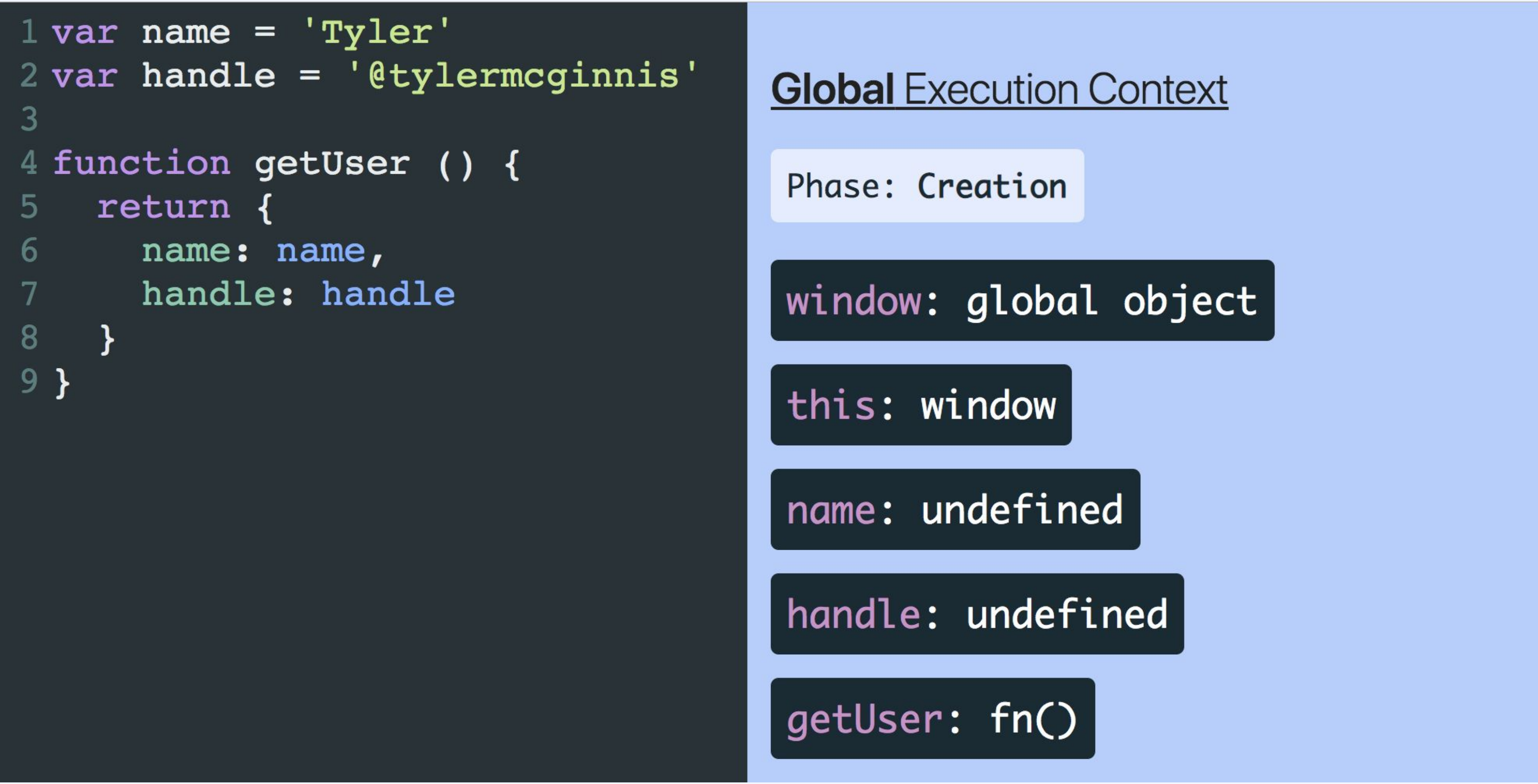
```
window: global object
```

```
this: window
```

Phases of Execution Context

There are 2 phases of Execution Context:

- **Creation Phase:** In the Creation phase, window and this are created, variable declarations (name and handle) are assigned a default value of undefined, and any function declarations (getUser) are placed entirely into memory.
- **Execution Phase:** In the Execution phase, the JavaScript engine starts executing the code line by line and assigns the real values to the variables already living in memory.



```
var name = 'Tyler'
var handle = '@tylermcginnis'

function getUser () {
  return {
    name: name,
    handle: handle
  }
}
```

Global Execution Context

Phase: Creation

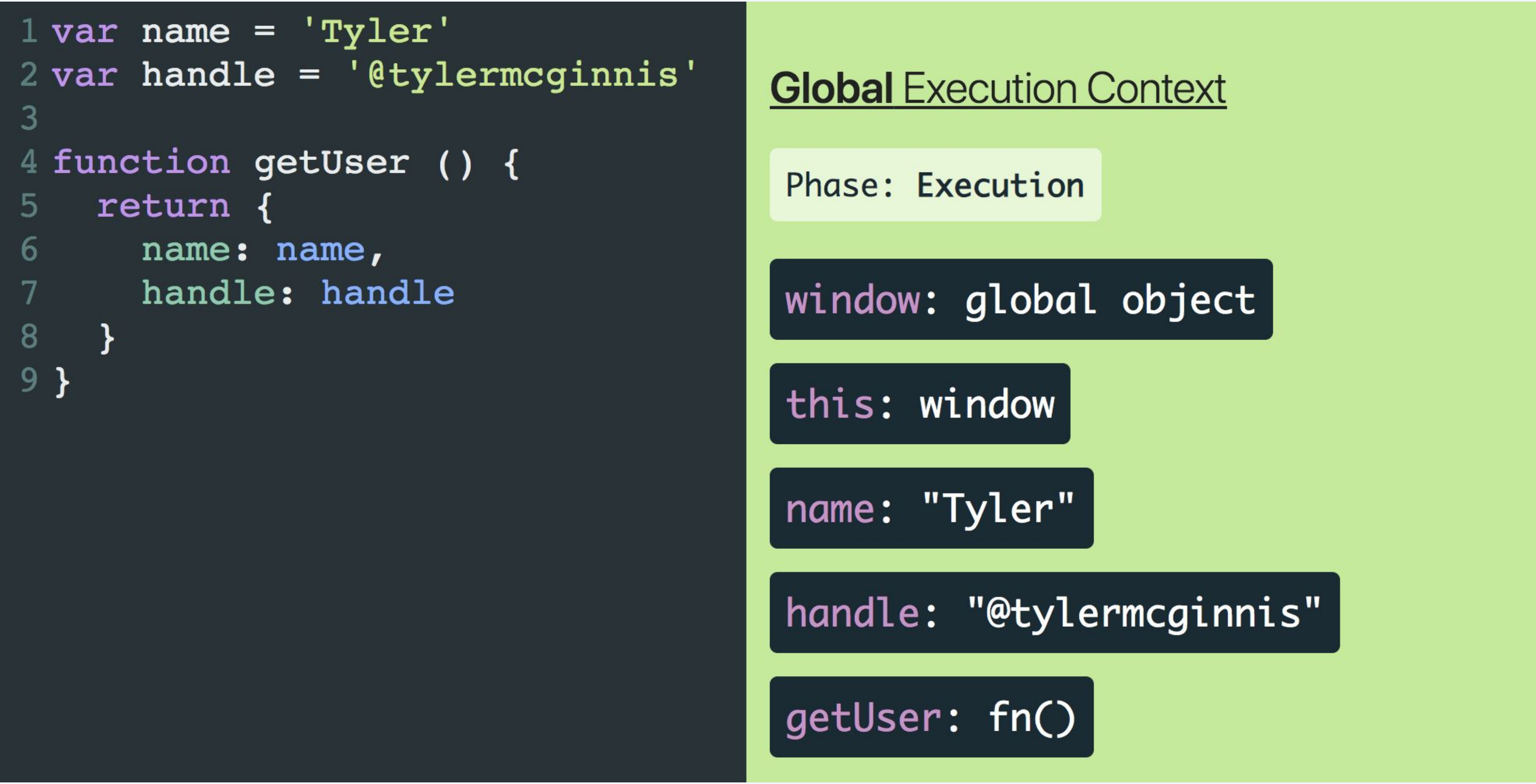
window: global object

this: window

name: undefined

handle: undefined

getUser: fn()



```
var name = 'Tyler'
var handle = '@tylermcginnis'

function getUser () {
  return {
    name: name,
    handle: handle
  }
}
```

Global Execution Context

Phase: Execution

```
window: global object
this: window
name: "Tyler"
handle: "@tylermcginnis"
getUser: fn()
```

JavaScript Visualizer

https://tylermcginnis.com/javascript-visualizer/

HOISTING

What is Hoisting??

In other programming languages you have to first declare the variable before you can use it.
Say for eg, PHP

```
$x
$x = "some-string"
```

But in javascript you can use it even before declaring it.

```
x = "some-string"
var x;
```

This mechanism where variables and function can be used before declaring them is called **Hoisting**.

Hoisting - Variable

Basically, JavaScript executes the code in two steps:

- Creation
- Execution

In the first step JavaScript analyses all the code and allocate the memory spaces for variables and functions. All the variables are assigned a value of undefined when declared first.

In the execution step it makes the assignments.

Hoisting - Variable

So if you try to print this num1 variable even before you assign it a value it will give you undefined.

```
console.log(num1) //Undefined
num1 = 5
console.log(num1) //5
var num1
```

Hoisting - Function

In the creation phase the functions definitions are moved to the memory.

So that’s why hoisting is possible with functions as well.

```
sum(5, 10);

function sum(num1, num2) {
  var total = num1 + num2;
  console.log('Sum => ' + total);
}
```

Hoisting - Function

But hoisting doesn't moves the function expressions to the top. Because it is treated as assignment and not declaration.

The code below will not work. It will give you an error.

```
sum();

var sum = function() {
  var total = num1 + num2;
  console.log('Sum => ' + total);
}
```

Hoisting

It is a bad practice to use variables before declaring them.

Hoisting is not a feature, it's just how JavaScript works behind the scenes. So, always ensure that you have declared your variables before you use them. It will help you avoid a lot of silent errors and unexpected behavior.

Hoisting

What do you think will be printed in the console and why?

```
var result = 6;
function sum() {
    console.log(result);
    var two = 2;
    var result = two + two;
    return result;
}
sum();
```

Sidebar/Navigation Drawer

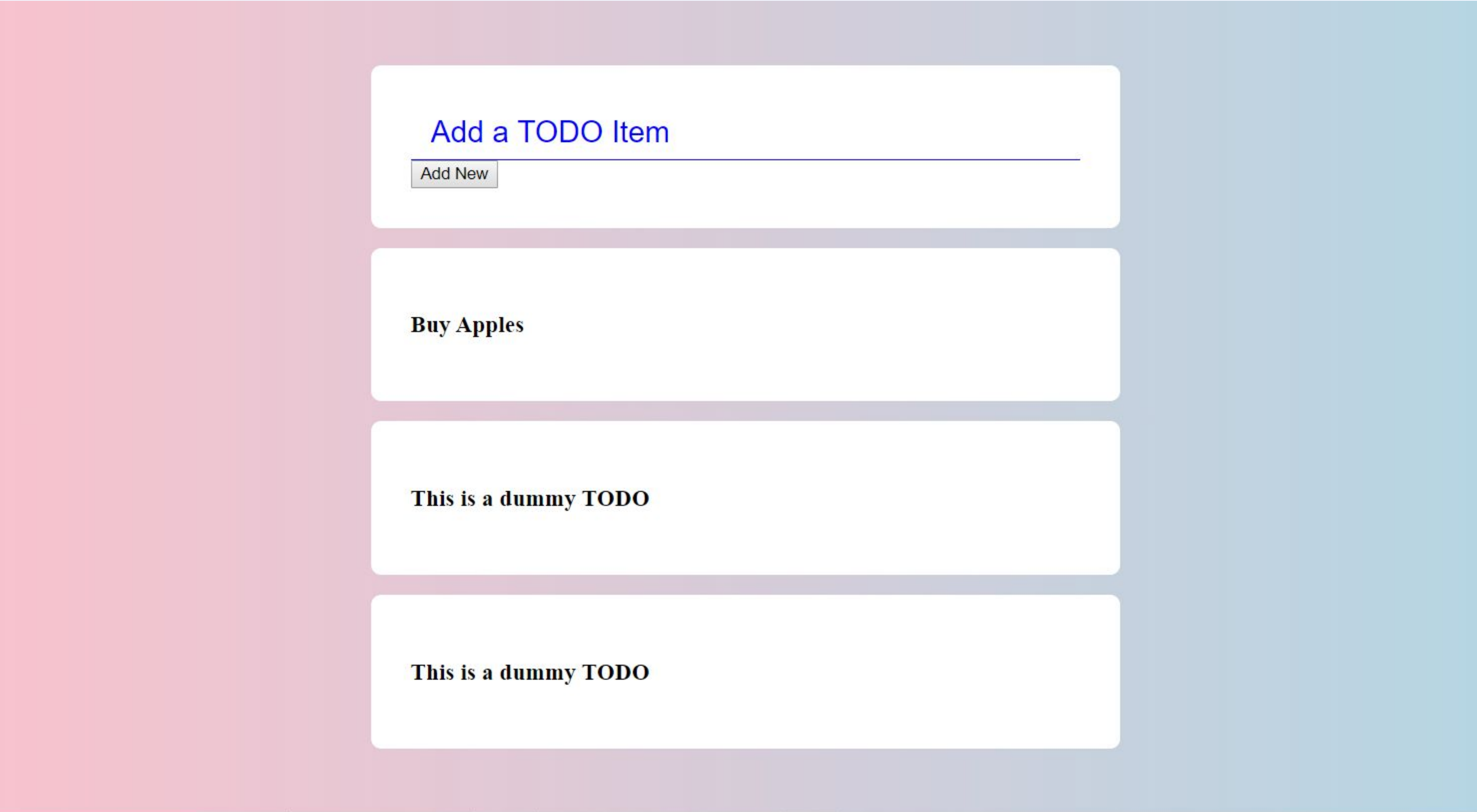
Sidebar

- The sidebar should be visible only on mobile screens.
- On hamburger click the sidebar should slide-in and slide-out.

Starter Code:
https://codepen.io/qaifikhan/pen/VwYjWMw?editors=1000

Finish this before tomorrow’s live session.

TODO App



Buy Apples

This is a dummy TODO

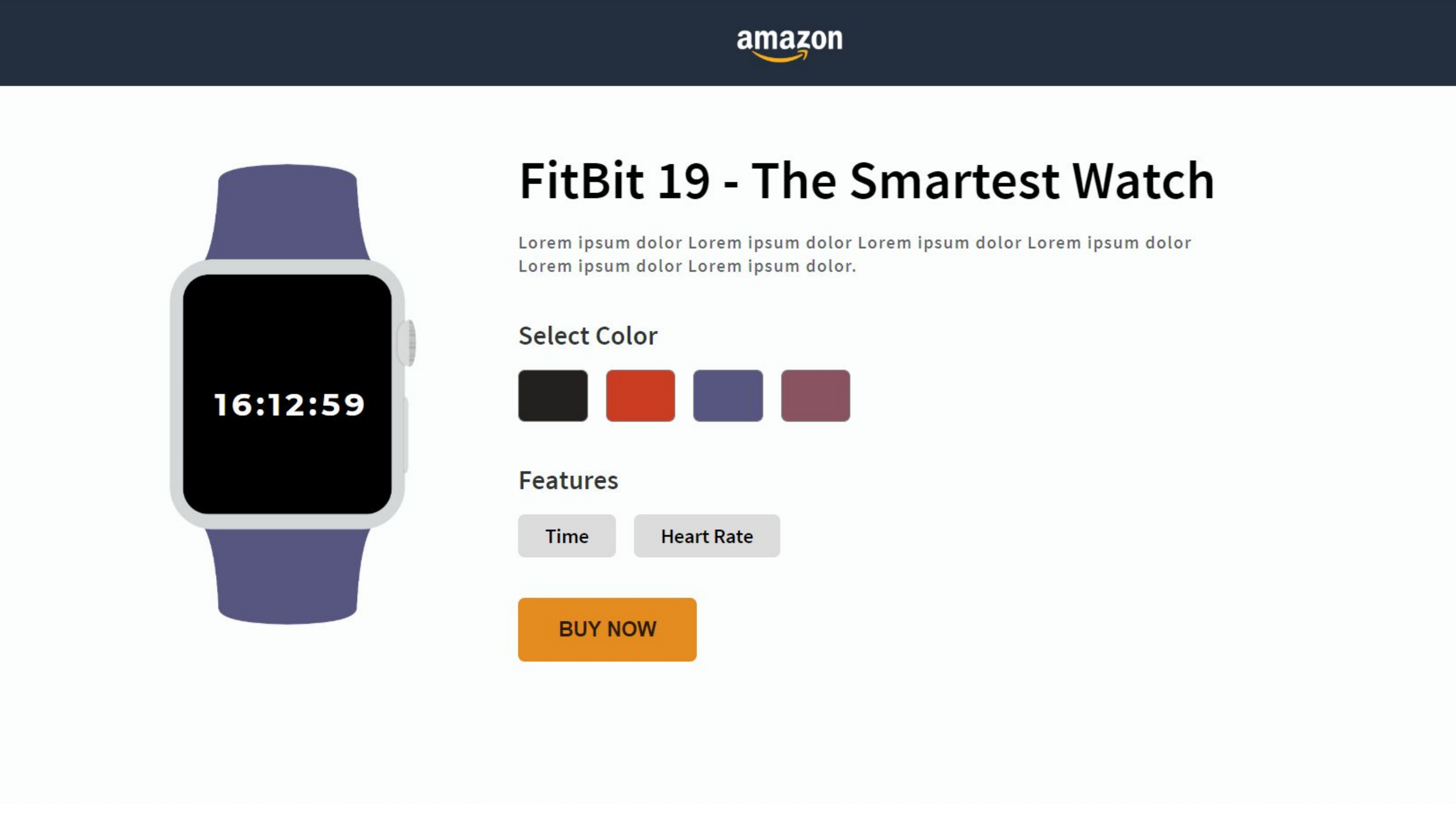
This is a dummy TODO

TODO App: Features

Create a TODO App with following features:

- There should be an input field to write the TODO item.
- On enter click the TODO item should be added as the first item in the TODO list.
- There should be an add button, on that button click the TODO item should be added as the first item in the TODO list.
- The field should give an error if the user tries to add a TODO item without writing something in the input box.
- Even when the webapp is closed, the data should be retained. When the user revisits the app, he/she should be able to see that data.
- The user should be able to update the TODO item.
- The TODO item should have created and updated date and time.

ASSIGNMENT-2: SMARTWATCH SHOWCASE



FitBit 19 - The Smartest Watch

Lorem ipsum dolor Lorem ipsum dolor Lorem ipsum dolor Lorem ipsum dolor Lorem ipsum dolor Lorem ipsum dolor.

Select Color

■ ■ ■ ■

Features

Smartwatch Showcase

Watch this video for problem statement:

https://www.edyoda.com/course/1496?episode_id=2494

GIT and GITHUB

GIT and Github

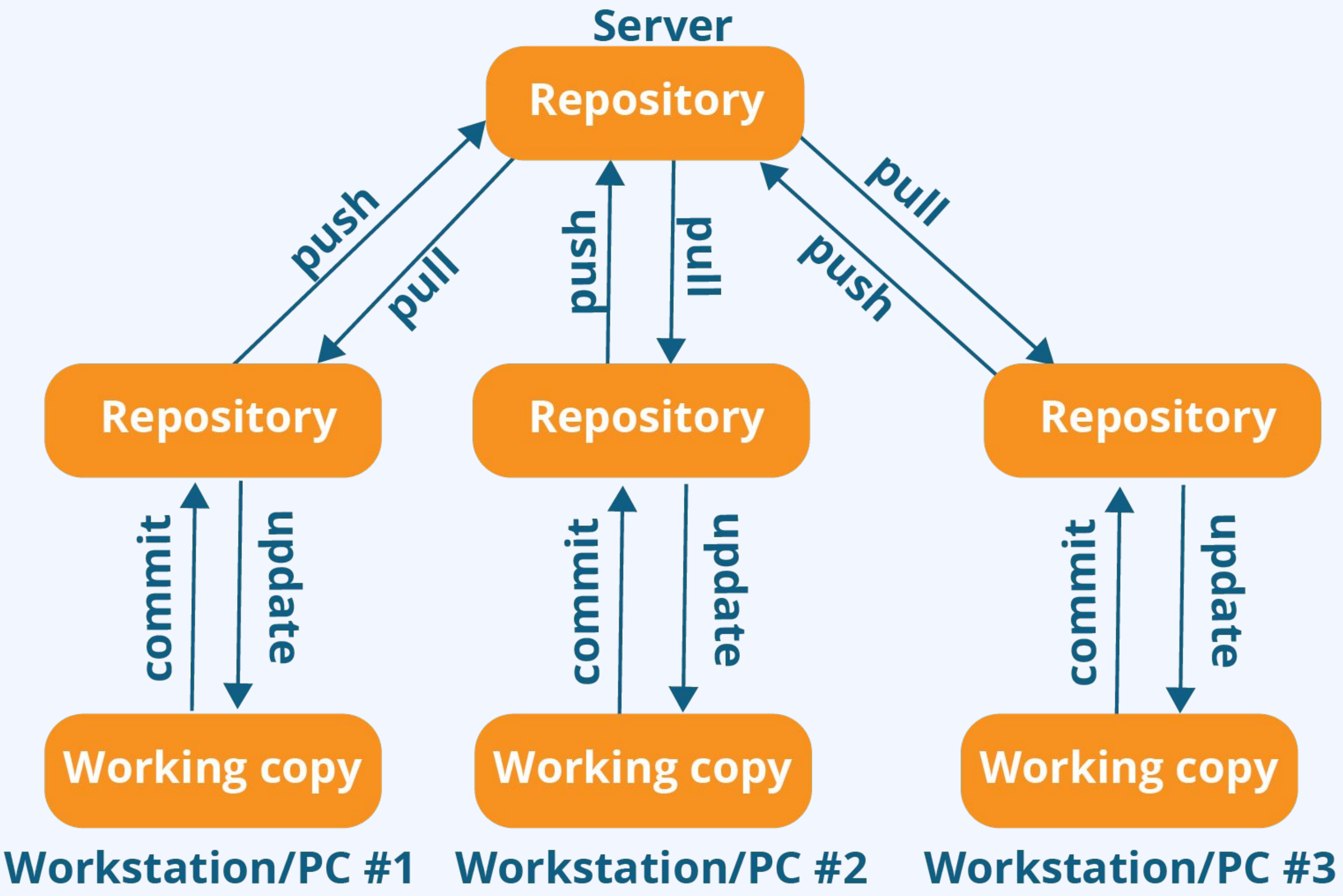
Version control systems are tools that help a software team manage changes to source code over time. Version control software keeps track of every modification to the code in a special kind of database.

Git is a distributed version-control system for tracking changes in source code during software development. It is designed for coordinating work among programmers, but it can be used to track changes in any set of files. It create a local repository and you can make changes in your local repository and push it to the remote repository where other developers can access those changes.

Repository is a data structure which stores the history of changes, versions of your files etc.

Git is the tool and GitHub is a hosting service for Git repositories.

Distributed version control system



Server

Repository

push

pull

push

pull

pull

push

Repository

Repository

Repository

commit

update

commit

update

commit

update

Working copy

Working copy

Working copy

Workstation/PC #1

Workstation/PC #2

Workstation/PC #3

GIT Installation

Download GIT on your system:
https://git-scm.com/downloads

Verify if installed properly:
Run the following command => git --version

Create an account on Github
Visit: https://www.github.com

Configure Git on your local system

```
git config --global user.name "Full Name"
git config --global user.email "your-email"
```

Create a new Repo on Github

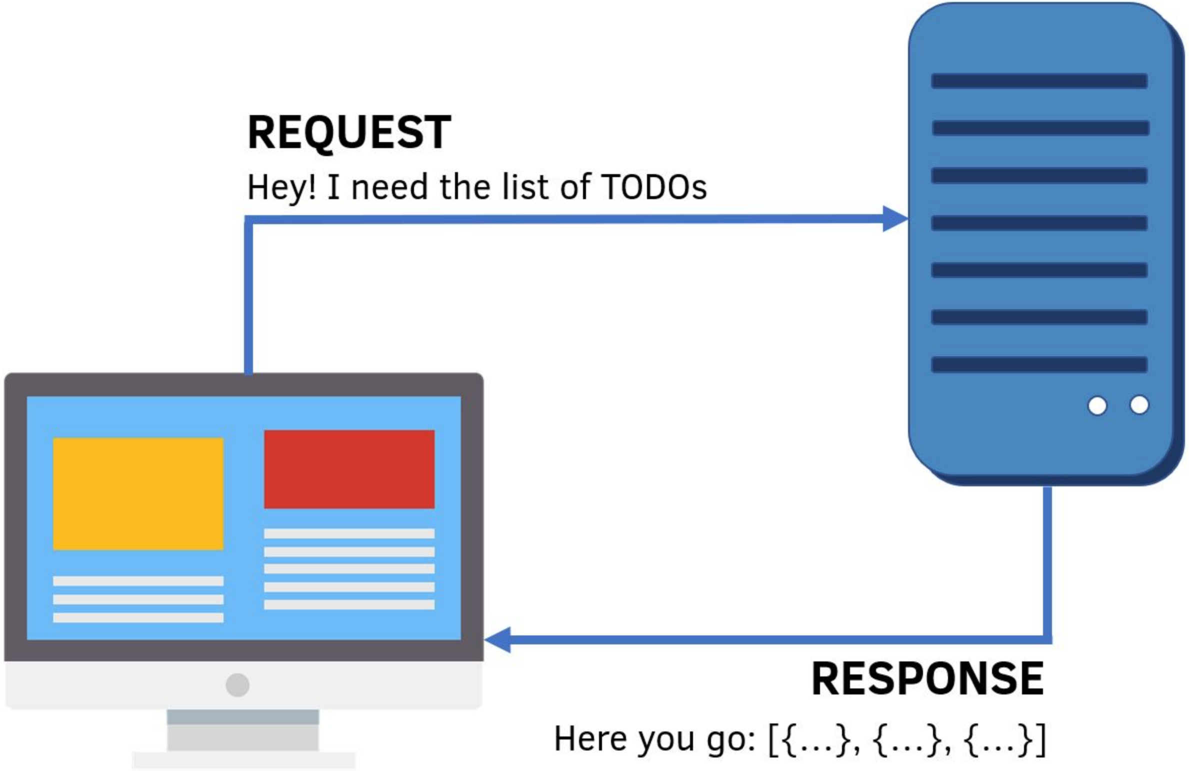
- Goto Github.
- Login to your account
- Click on create new repository.
- Follow the steps.

Updating an existing Repo

- Make changes in the files.
- Save all the changes.
- Use command prompt/gitbash and go to the project directory.
- Stage the changes by running command. **git add .** //This will stage all the files which means they are ready to be saved in the local repository.
- Commit the changes by running command **git commit -m “message”** //This will save the staged changes in the local repository.
- Push these changes to the remote repository by running command **git push**

CONNECTING TO BACKEND

FRONTEND	BACKEND
It refers to the client-side of the application	It refers to the server-side of the application.
It includes everything which is visible on the browser screen.	Backend is where all the data is handled. It sends data to show on the frontend. And it receives data from the frontend to store or update in the databases.
To develop frontend web applications you can use HTML, CSS, JS, React, Angular etc.	To develop backend applications you can use NodeJs, DotNet, PHP, JAVA etc.

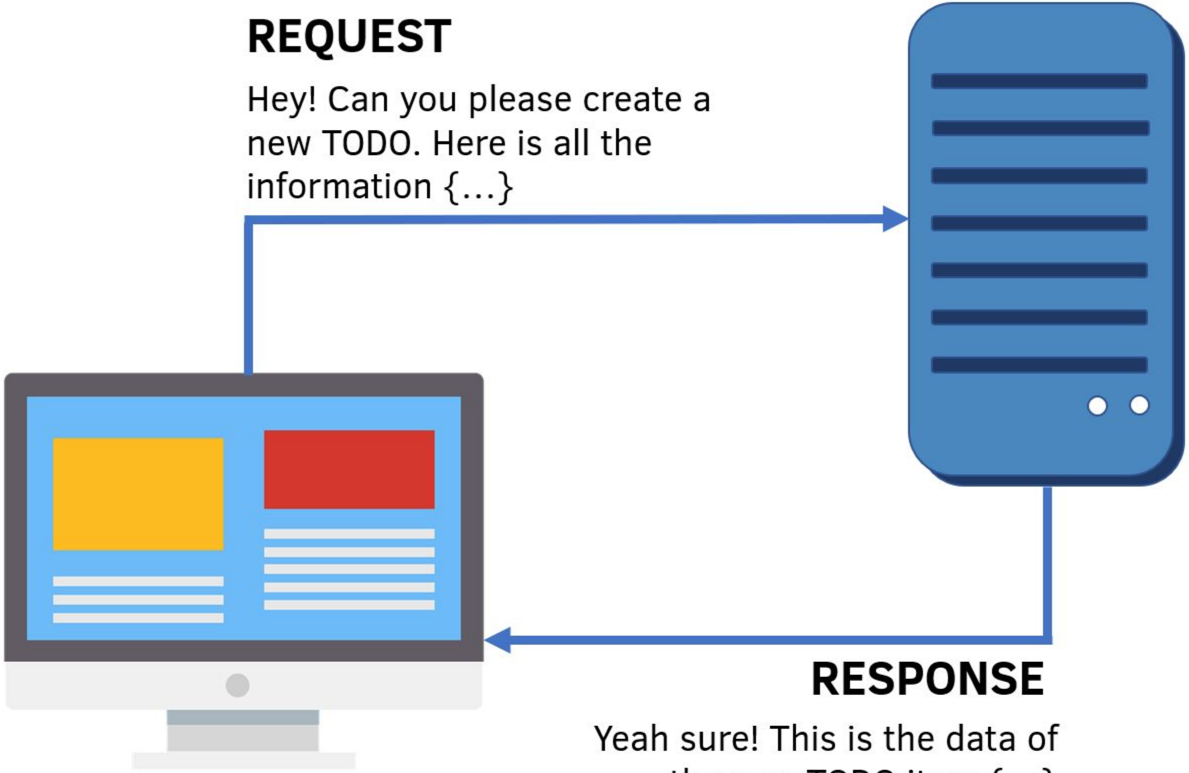


REQUEST

Hey! I need the list of TODOs

RESPONSE

Here you go: [{…}, {…}, {…}]



REQUEST

Hey! Can you please create a new TODO. Here is all the information {...}

RESPONSE

Yeah sure! This is the data of the new TODO item {...}

What is HTTP??

The requests are sent using HTTP. This HTTP stands for Hypertext Transfer Protocol. It is a protocol which allows the fetching of data from the server. It is the foundation of any data exchange on the Web. Clients and servers communicate by exchanging individual messages.

The messages sent by the client, in this case our Frontend Application, are called **requests** and the messages sent by the server as an answer are called **responses**.

HTTP Methods

The HTTP provides us with different methods for different type of requests:

- To get some data from the backend you can use **GET**.
- To create new data entries on the backend you can use **POST**.
- To update an existing data entry on the backend you can use **PUT**.
- To delete an existing date entry on the backend you can use **DELETE**.

There are a few other methods but these 4 are the major ones that you will use very often.

API Endpoints

To send request to your backend you will use a url which will point to your backend and will tell your backend how to handle the request. This url is called API Endpoint.
Say for example, you want to get TODO list from your backend.
API Endpoint: https://jsonplaceholder.typicode.com/todos
Method: GET

The part of the URL before todos points to your server. Now this second part, **/todos** tells your server to return todos list because a backend developer would have written some code to handle these urls or endpoints.
Just like on the frontend you handle different HTML pages for different URLs. Say /index.html loads index.html file or /contact.html loads contact html page.
Similarly, the backend developers write some code to handle different endpoints.

HTTP Response Codes

When the backend server sends a response. It sends a status code which tells whether the request was a success or it failed.

These are some common Response Status Codes:
200 - represents **Success** which means the request was successful.
400 - represents a **Bad Request** which means that the backend did not understand the request.
401 - represents **Unauthorized** which means that the user is not authorized to access the response data.
404 - represents **Not Found** which means that the url is not found by the backend.
500 - represents **Something Went Wrong** at the server.
There are many other HTTP response status code but these are the most common ones.

JSON

Intro to JSON

- JSON stands for JavaScript Object Notation.
- It is basically a text format that makes it easy to share data between Clients and Servers.
- Its origin is based on how JavaScript object works so that's why it looks like a JavaScript Object but remember **it is different from a JavaScript Object.**
- Regardless of the fact that it has its origin from JavaScript, it is widely used across many Languages like C, Ruby, Python, PHP etc.

Intro to JSON

For example, to create a JavaScript Object this is what we would do,

```
var mTodo = {
    message: 'Buy groceries',
    dueDate: '18th Jan 2019'
}
```

Now, to write the same thing as JSON, **you can wrap the key names between double quotes.**

Say we had an array of JSON Objects. This array of JSON Objects is called **JSON Array.**

Parsing JSON

One of the biggest benefits of using JSON Object is that it can be converted to a string and that string can be converted back to the JSON Object.

Now, for this purpose JSON provides us with 2 methods:

- JSON.stringify() - We pass the JSON Object inside the parenthesis and the method returns a stringified JSON or JSON string.
- JSON.parse() - We pass the JSON String inside the parenthesis and the method returns the JSON Object or JSON Array.

Values allowed in JSON

JSON Object accepts

- a string
- a number
- an object (JSON object)
- an array
- a boolean
- null

Unlike JavaScript Object, **JSON cannot hold two types of values which are undefined and functions.** If you try to convert a JSON Object into string it just ignores the keys with values undefined and functions.

AJAX

Intro to AJAX

AJAX is a short form for Asynchronous JavaScript And XML
AJAX allows web pages to be updated asynchronously by exchanging data with a web server behind the scenes. This makes it possible to update parts of a web page, without reloading the whole page.

Initially AJAX was used to send and receive XML because that was how the data was received from the backend in old days. But now it can also be used to send/receive JSON objects, which is pretty common these days.

Intro to AJAX

Now, if you remember I just mentioned a word asynchronously. Normally, the way JavaScript code is run is that it executes one line of code and then moves to next line and then the line after that and so on...
Say there is a line of code which takes 5 seconds to execute. This would just mean that JavaScript will have to wait 5 seconds before it can move on to execute more lines of code. Now, this is a bad user experience, your webpage will just freeze for 5 seconds and your user will not be able to do anything on the webpage. To avoid this, there is another way to execute a specific part of code such that the execution starts and it runs in the background, which allows rest of the code to be executed. And once the 5 second task is complete it let's JavaScript know that it has completed and a function can be called. This function is called Callback function.

Intro to AJAX

Now, you must be wondering what line of code might take 5 seconds to execute. When you send a request to the backend it takes time to bring back the response and for some requests it might even take upto 5 seconds or even more. That's why it is recommended to make Backend Requests as asynchronous.

XMLHttpRequest Object

AJAX uses a browser built-in XMLHttpRequest object to make requests to the backend server. This gave a simple and standard way to make HTTP requests from JavaScript to get content and update the HTML page.

To use the XMLHTTPRequest object first we need to create a new instance.
var xhttp = new XMLHttpRequest();

To define the url endpoint, the type of request this XMLHttpRequest object gives us access to a method called open
xhttp.open("GET", "*api-endpoint*", true); //It accepts 3 params, 1st is the request method, 2nd is the URL and 3rd accepts a boolean value to define whether the call is synchronous or asynchronous. It is recommended to make it asynchronous.

XMLHttpRequest Object

Now, to trigger this call we can use another method send.

xhttp.send();

This sends a request to the backend. To confirm it you can check it in network tab of the browser.

The “readyState” Property

The XMLHTTPRequest object gives us access to another property called readyState. The readyState property holds the status of the XMLHttpRequest.

These are the available Status for the request.
UNSENT- Represented by 0. It means the request has been initiated but open() not called yet.
OPENED- Represented by 1. It means that open() method has been called.
HEADERS_RECEIVED- Represented by 2. It means that send() has been called.
LOADING- Represented by 3. It means the backend is processing request.
DONE- Represented by 4. It means the request is completed and response is received.

The “onreadystatechange” Method

The XMLHTTPRequest also gives us access to another method called **onreadystatechange**. It executes a function when the readyState changes.

```
var xhttp = new XMLHttpRequest();
  xhttp.onreadystatechange = function() {
    if (this.readyState == 4) {
      console.log(this.responseText)
    }
  };
  xhttp.open("GET", "api-endpoint”, true);
  xhttp.send();
```

How to show a Progress Bar??

We can use readyState to know what’s the status of the request and handle it accordingly.

```
if (this.readyState === 3) {
 //.....Write code to show the HTML element which generates a Progress Bar
}

if(this.readyState === 4) {
 //Hide the progress bar and handle success/failure response.
}
```

Handling Success Response

This call might also fail for any XYZ reasons like the network connection failed, the server took too long to respond and the call expired, or something went wrong at the backend etc etc. To handle that we can use the status code of HTTP response.

To get access to status code of the response XMLHttpResponse gives us access to another property called **status**.

```
if (this.readyState == 4 && (this.status >== 200 && this.status < 300)) {
 //.....
}
```

Making a POST request

```
var dataToBeSentToBackend = {
 ...
}

xhttp.open("POST", "api-endpoint”, true);
xhttp.setRequestHeader("Content-Type", "application/json;charset=UTF-8");
xhhtp.send(JSON.stringify(dataToBeSentToBackend ));
```

ERROR HANDLING

The “try...catch” Statement

The **try...catch** statement consists of a try block, which contains one or more statements, and a catch block, containing statements that specify what to do if an exception is thrown in the try block.

In other words, you want the try block to succeed—but if it does not, you want control to pass to the catch block. **If any statement within the try block (or in a function called from within the try block) throws an exception, control immediately shifts to the catch block.** If no exception is thrown in the try block, the catch block is skipped.

The finally block executes after the try and catch blocks execute but before the statements following the try...catch statement.

The “try...catch” Statement

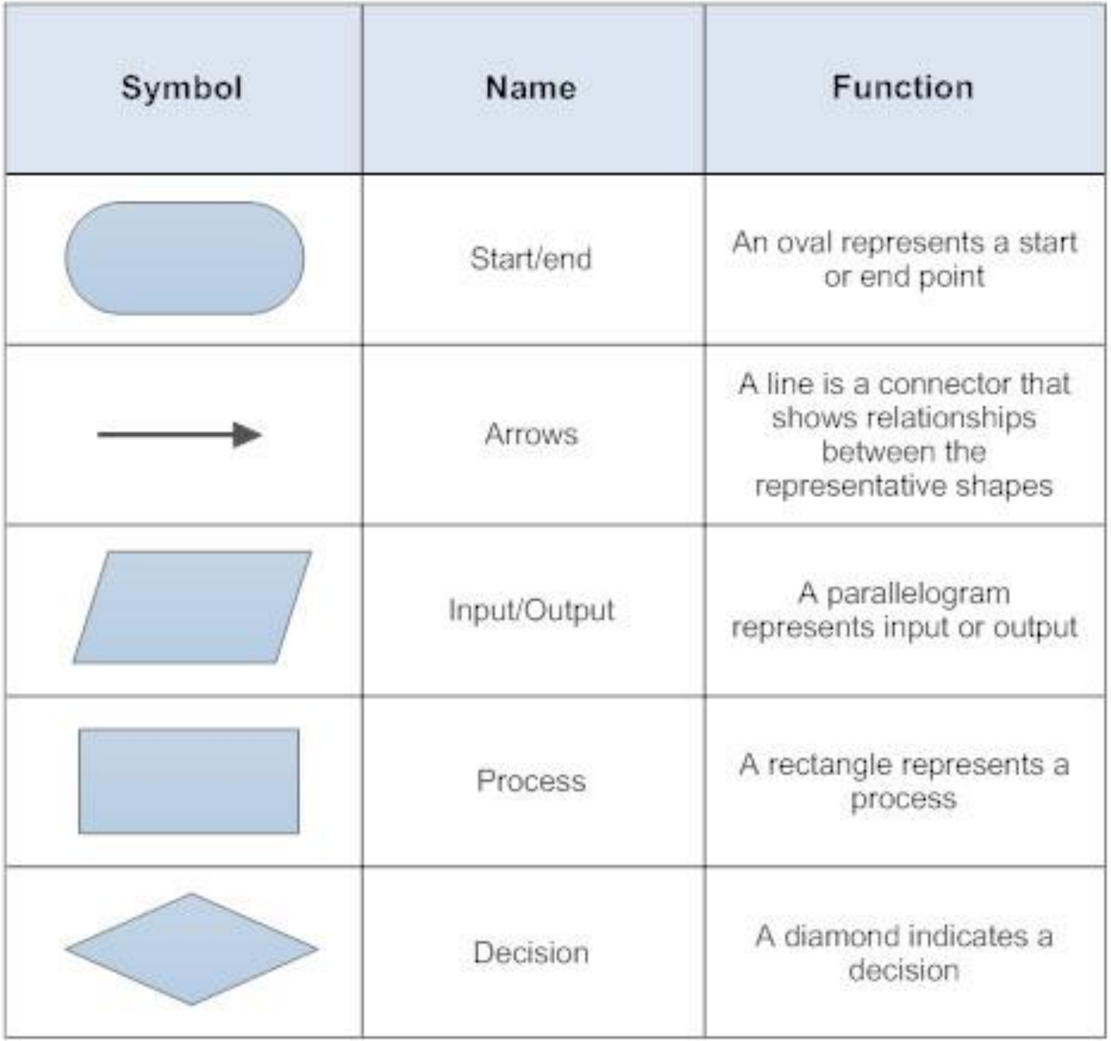
Syntax:

```
try {
    console.log(hola)
} catch(e) {
    console.log(e.toString())
} finally {
    console.log('Inside Finally');
}
```

HOMEWORK: Find practical applications when to use finally!!

NOTE: You can use the throw statement to throw/create a custom exception.

FLOWCHARTS

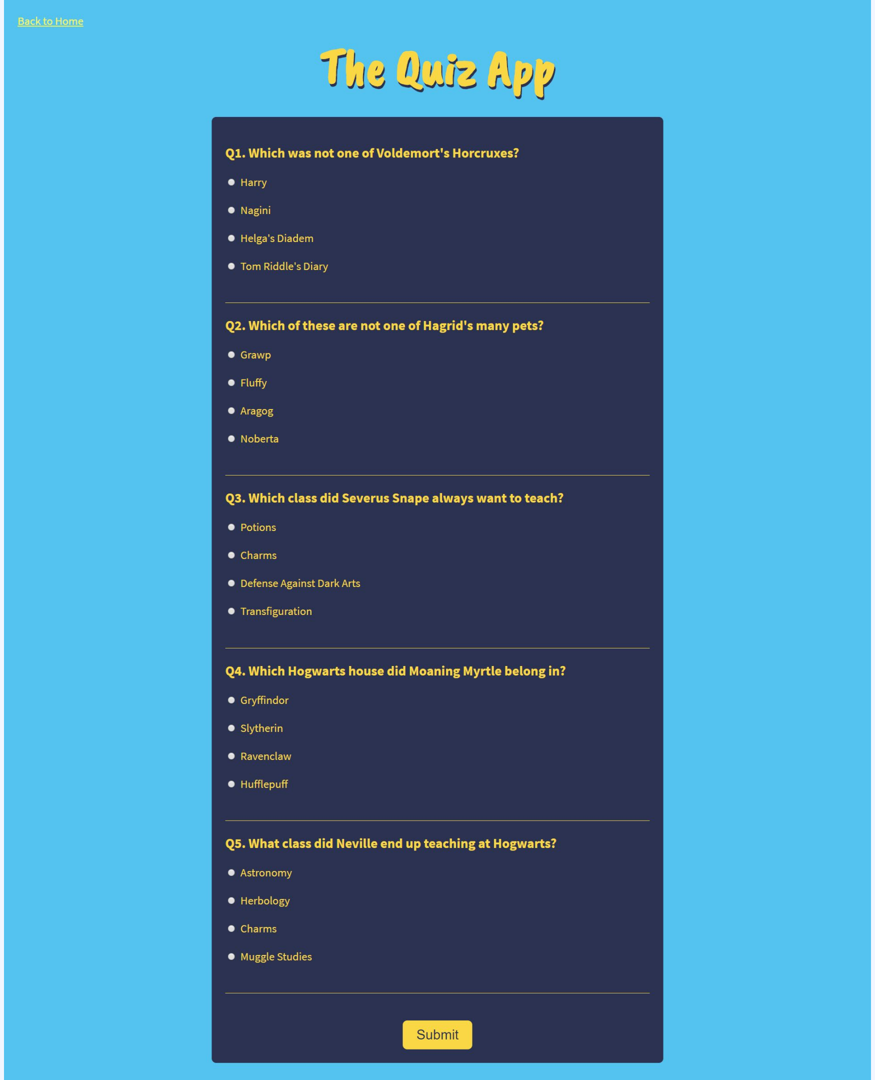


Symbol	Name	Function
⬭	Start/end	An oval represents a start or end point
→	Arrows	A line is a connector that shows relationships between the representative shapes
▱	Input/Output	A parallelogram represents input or output
▭	Process	A rectangle represents a process
◇	Decision	A diamond indicates a decision

Flow Thinking: Homework

Design flowchart for homepage and details page. Also, add implementation details.

Practice Problem: Quiz App



The Quiz App

Q1. Which was not one of Voldemort's Horcruxes?

○ Harry

○ Nagini

○ Helga's Diadem

○ Tom Riddle's Diary

Q2. Which of these are not one of Hagrid's many pets?

○ Grawp

○ Fluffy

○ Aragog

○ Noberta

Q3. Which class did Severus Snape always want to teach?

○ Potions

○ Charms

○ Defense Against Dark Arts

○ Transfiguration

Q4. Which Hogwarts house did Moaning Myrtle belong in?

○ Gryffindor

○ Slytherin

○ Ravenclaw

○ Hufflepuff

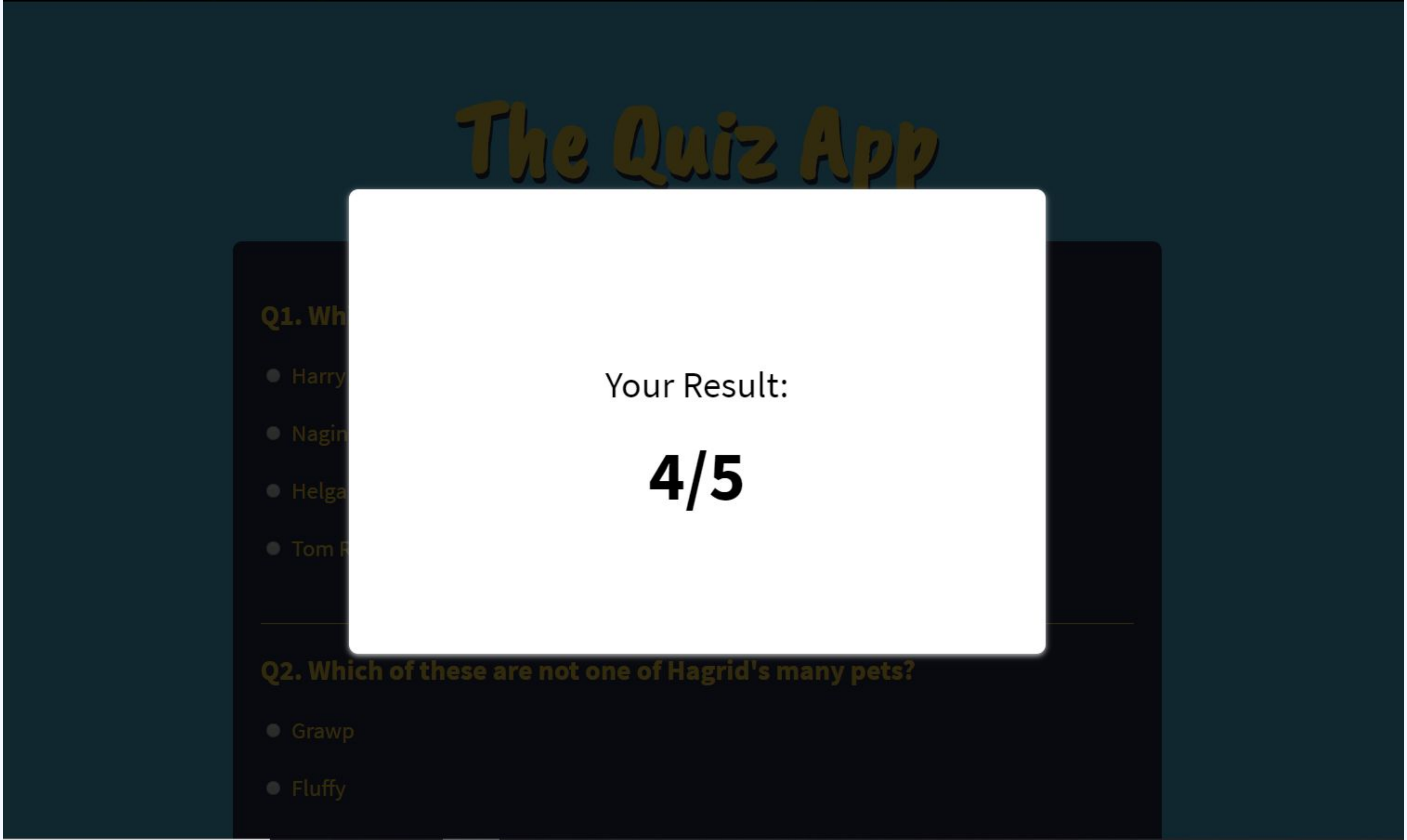
Q5. What class did Neville end up teaching at Hogwarts?

○ Astronomy

○ Herbology

○ Charms

○ Muggle Studies



The Quiz App

Q1. Wh

○ Harry

○ Nagin

○ Helga

○ Tom R

Your Result:

4/5

Q2. Which of these are not one of Hagrid's many pets?

○ Grawp

○ Fluffy

Practice Problem: Quiz App

Create a flowchart and implement the following functionalities.

Functionality:

- User should be able to select only one option for each question.
- When user hits the submit button, show the user his/her score out of 5.
- Each question holds 1 mark. There is not negative marking.

API Endpoint => http://5d76bf96515d1a0014085cf9.mockapi.io/quiz

jQuery

What is jQuery??

- It is a JavaScript library. It means that jQuery uses JavaScript under the hood.
- It is fast, lightweight, and feature-rich. It is very small is size which makes it lightweight and it comes with a lot of in-built features.
- It is based on the principle "write less, do more".
- If you plan to use it then you wouldn't have to worry about browser support because it is supported by all the major browsers like Chrome, Firefox, Safari, Internet Explorer, etc.

What can jQuery do??

These are the list of things that jQuery can help you with.

- HTML/DOM manipulation
- CSS manipulation
- Handling HTML Event
- Effects and animations
- Making AJAX Calls.

Advantages of jQuery

Let's talk about some advantages of using jQuery:

- **Save lots of time:** It provides you with tons on inbuilt functionalities that you can use directly use or else you would have to write it yourself using JavaScript.
- **Simplify common JavaScript tasks:** It helps you write the same functionality with lesser lines of code than JavaScript.

How to enable jQuery in your webapp??

Adding jQuery to your web app is very simple.

- Just search for jquery cdn on google.
- Visit the official jQuery website.
- Copy this script and you can paste it next to your other script tags.

```
<script
  src="https://code.jquery.com/jquery-3.4.1.min.js"
  integrity="sha256-CSXorXvZcTkaix6Yvo6HppcZGetbYMGWSFlBw8HfCJo="
  crossorigin="anonymous"></script>
```

And that's all now you can use all the jQuery features to develop your web app.

Syntax to use jQuery

This is the syntax to use jQuery.

```
$(selector).method()
```

- $ sign is used to access jQuery
- “selector” is used to find/select HTML elements.
- method() is used to perform some action on the selected element or elements.

Selectors and Filters

Selectors and Filters provide a way of finding and extracting information from our web page.

Selectors are used to select the HTML elements using CSS-like selectors.
For eg,
$('p') //This statement will select all the HTML elements with the paragraph tag and will return it as a list of objects.
$('#topbar')
$('header section')
$(‘.menu-item’)

We can also select elements by attributes. Say we wanted to select all input elements with type text. We can do something like this:
$("input[type=text]")

Selectors and Filters

Filters are used to refine the results returned from the selectors.

For eg,

$('p:first') //This :something is a filter which selects the first paragraph returned by $('p')

$('p').first() //The above line of code can also be written like this.

$('li').last() //say we wanted the last list item then we can use the last filter.

To select an element at a specific position you can use the eq filter

$('p').eq(1) //Return elem at position 2

The “ready()” Method

jQuery gives us access to a method called ready. What it does is, it checks whether the DOM tree is generated or not. Once the HTML elements are mounted in the browser DOM then we can perform any and all operations on them.

For that we can select the document object and use the ready method.

```
$(document).ready(function(){
    // JS Code goes here
});
```

This can also be written as

```
$(function(){// JS Code goes here });
```

Now even if we load our scripts in the head tag. It wouldn't give any error.

HOMEWORK: Find Vanilla JS equivalent for ready() method.

HOMEWORK: Find difference between load event and DOMContentReady

The “html()” and “text()” Method

We can use the html() method to get content of selected HTML element. This content can either be text or more HTML elements.

The syntax is,

To get values

$('selector').html()

To set values

$('selector').html(childElement)

Let's try an example to understand it better.

$('ul').html()

$('li').eq(2).text() //3rd element

Similarly, just to get just the text content we can use another method called text()

Creating and Updating HTML Elements

Vanilla JS:

```
  var para = document.createElement("p");
  var textNode = document.createTextNode(‘Hello World’)
  para.appendChild(textNode);        // Create text with DOM
```

Using jQuery:

```
$("<p>").text("Hello World");
```

Adding and Deleting HTML Elements

To append this to an HTML element we can use a jQuery method called append(). This method will add the newly created HTML element to the selected HTML element.

```
$("p").append(elem);
```

You can also append multiple elements using the same append method.
To pass multiple elements as arguments you can separate them by comma.

```
$("p").append(p1, p2, p3, p4, ....);
```

To append new HTML element to the top we can use the prepend() method

```
$("p").prepend(elem);
```

Similarly to remove an HTML element you can use the remove() method

```
$('#logo').remove();
```

HOMEWORK: Find jquery equivalent of insertBefore()

Adding and Updating CSS

To get CSS of an HTML element. We can use the css() method. It accepts the css property name as an argument and it returns the value of the first matched HTML element.
The syntax is as follows:

```
$('selector').css({property-name});
```

Similarly, to set or update styles we can use the same function and instead of passing one argument we will pass two. The first one will be the property-name and the second one will be it's value.

We can also set multiple styles using the same css method. Now instead of passing property name and value we pass an object with multiple property name and values as key-value pairs.

```
$("p").css({"background-color": "yellow", "font-size": "200%"});
```

Adding and Removing Classes

To add a new class we can use another method called addClass() and we pass the class name as an argument.

```
$("div").addClass("important");
```

Similarly, to remove a class we can use the removeClass method.

```
$("p").removeClass("blue");
```

Events

We can use the event name followed by parenthesis.

This is the syntax to handle events on selected HTML element or elements.

```
$('selector').event-name(function() {
    // Do something on event trigger
})
```

For eg,

let's try the click event.

```
$("btn-add").click(function(e){
    $(this)
});
```

Events

To add multiple events we can use on method

```
  $("input").on({
  'input' : function(e){
    console.log(e.target.value); //We can also use this to get current element.
    $(this).css("background-color", "yellow");
  },
  'blur': function(e){
    console.log(e.target.value);
    $(this).css("background-color", "green");
  }
})
```

AJAX Calls

jQuery also supports AJAX calls and it gives us access to methods to make HTTP calls.
Say you wanted to fetch some data from the backend. In this case you will make a GET call and for that jQuery has a method which is called GET.

This is the syntax
$.get(URL,callback).fail(callback);

The callback() function is called when the request is completed.
You don't have to create XMLHTTPrequest objects or listen to status changes.
jQuery handles everything for you.

AJAX Calls: POST

To make POST calls you can use the POST method.

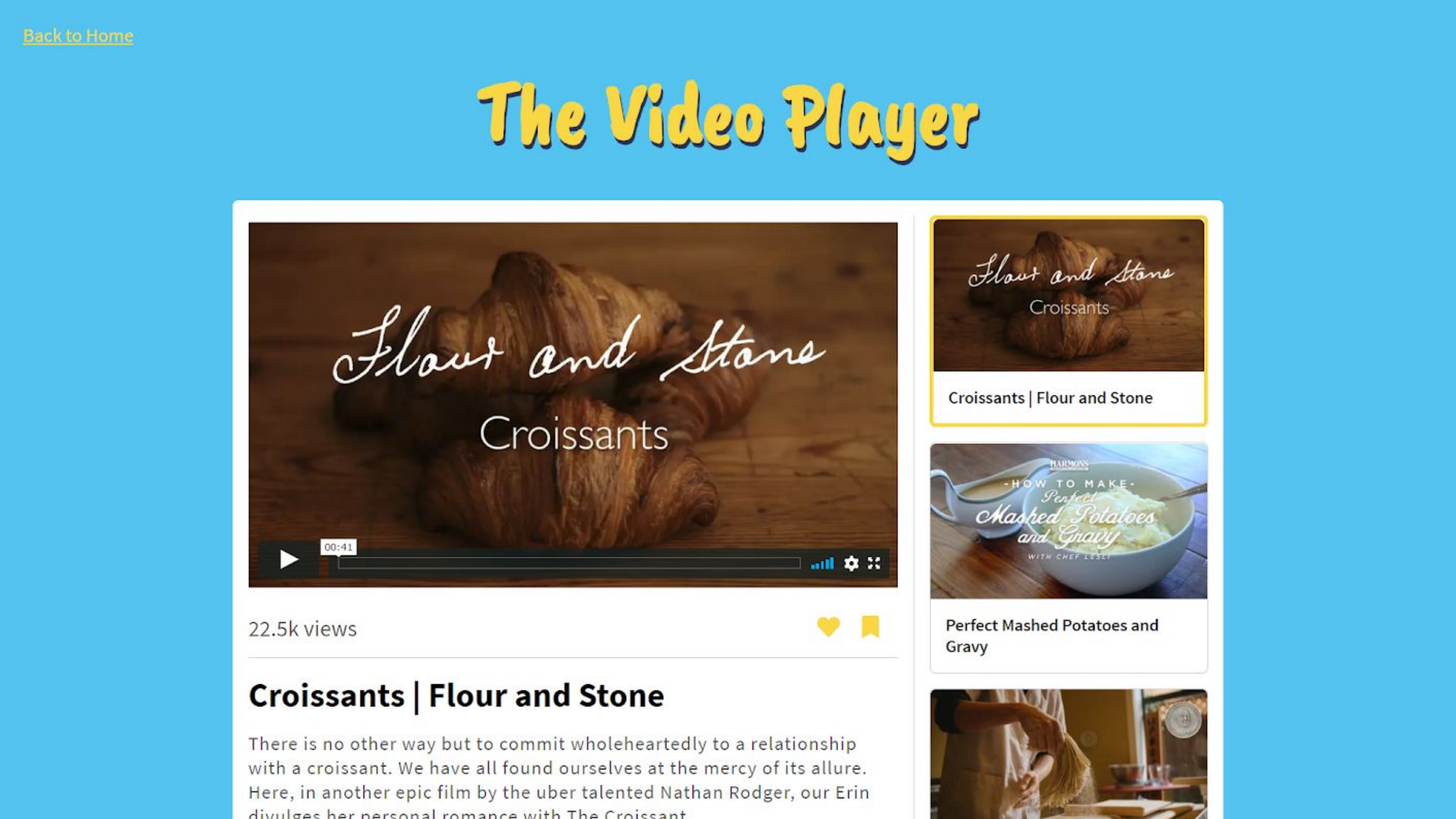
$.post(URL,data,callback); //It accept 3 parameters

- First, is the URL or the API Endpoint.
- Second, is the data that needs to be sent to the backend. You can directly pass JSON objects, you don’t have to convert it to string.
- Third is the callback function which is called when the request is completed.

AJAX Calls: PUT, DELETE etc

```
$.ajax({
  type: "PUT",
  url: "api-endpoint",
  data: {some-json-obj-or-array},
  success: function(response) {
      // Do stuff
  },
  error: function(request,status,errorThrown) {
      // There's been an error, do something with it!
      // Only use status and errorThrown.
      // Chances are request will not have anything in it.
  }
});
```

Practice Problem: Video Player



The Video Player

22.5k views

Croissants | Flour and Stone

There is no other way but to commit wholeheartedly to a relationship with a croissant. We have all found ourselves at the mercy of its allure. Here, in another epic film by the uber talented Nathan Rodger, our Erin divulges her personal romance with The Croissant

Application Data

List of Functionalities:

- When the web-app is opened the playlist is loaded from backed.
- The first video from the playlist is loaded in the details section.
- When user clicks on a video card from playlist, that video is loaded in the details section and the page is scrolled to the top.
- Currently, played video is highlighted in the playlist.
- When user clicks on Like Button the video gets liked/unliked based on the current status.
- When user clicks on Bookmark Button the video gets saved/unsaved based on the current status.

Application Data

Application Screenshot: https://i.imgur.com/aRpuIpm.png

Get Playlist: https://5d76bf96515d1a0014085cf9.mockapi.io/playlist

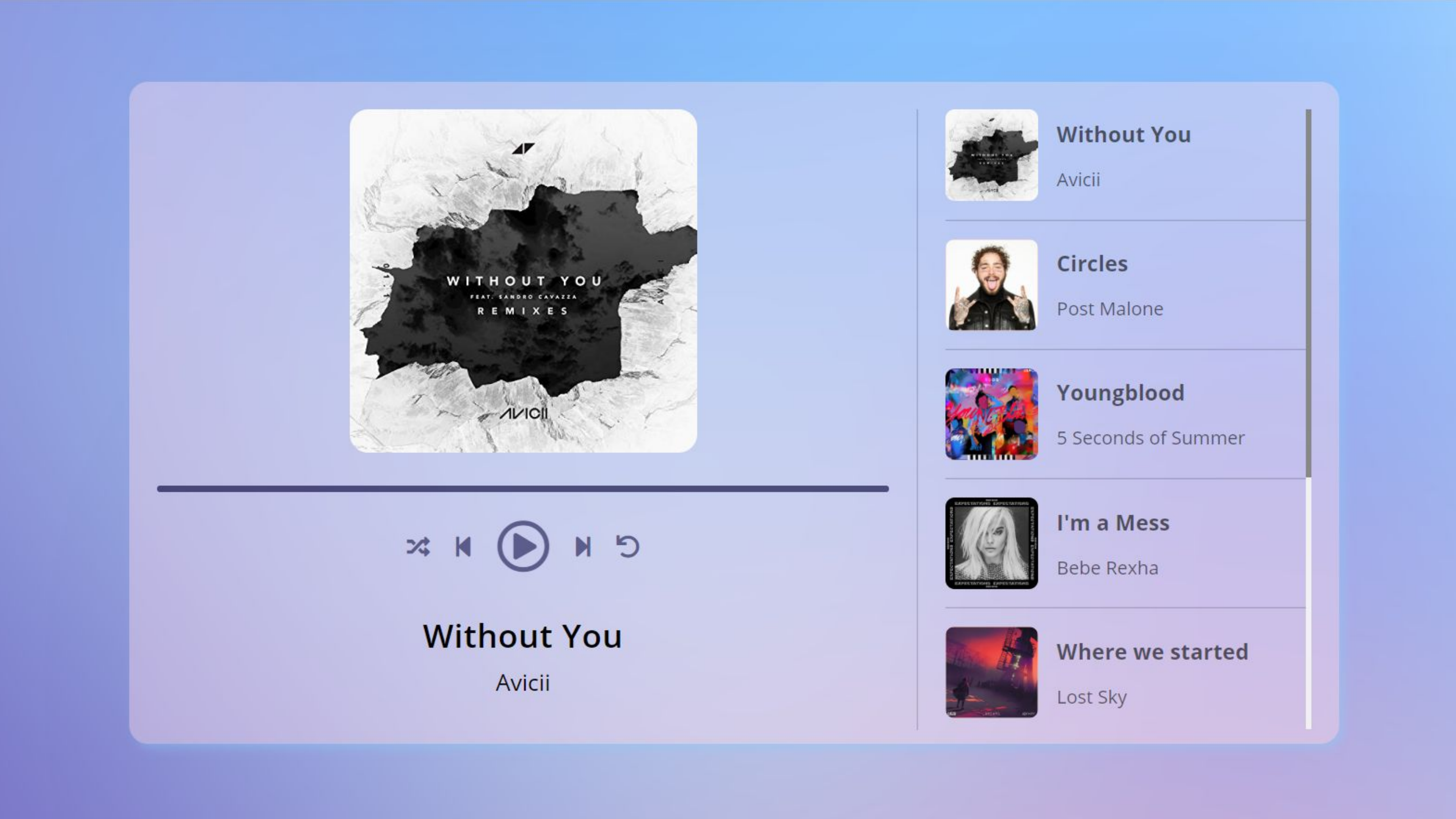
Get Video Details: https://5d76bf96515d1a0014085cf9.mockapi.io/video/1

LIKE UPDATE: https://5d76bf96515d1a0014085cf9.mockapi.io/video/1
{“isLiked”: true},{“isLiked”: false}

BOOKMARK UPDATE: https://5d76bf96515d1a0014085cf9.mockapi.io/video/1
{“isSaved”: true},{“isSaved”: false}

```
<iframe id="video-player" src="https://player.vimeo.com/video/190062231"
frameborder="0" webkitallowfullscreen mozallowfullscreen
allowfullscreen></iframe>
```

Practice Problem: Music Player



Without You

Avicii

Without You

Avicii

Circles

Post Malone

Youngblood

5 Seconds of Summer

I'm a Mess

Bebe Rexha

Where we started

Lost Sky

Music Player: Features

Here is the list of features for Music Player:

- By default the first song should loaded.
- When the user clicks on the play/pause button the song should play/pause and the icon should get updated.
- When a song finishes then the next song should be played automatically.
- On repeat icon click enable repeat track i.e., the same track should be played over and over.
- On shuffle click enable shuffling i.e., the next button should play a random track.
- Seek Functionality in progress bar.

Submit it before 9:00 PM tomorrow.

Music Player: https://reactmusicplayer-ab9e4.firebaseapp.com/

URL to get data for Music Player:

http://5dd1894f15bbc2001448d28e.mockapi.io/playlist

KEEP LEARNING!!