Javascript Day 1

Prepreqs on machine

NodeJS atleast https://nodejs.org/en/blog/release/v10.13.0/

Chrome Browser

Visual Studio Code

For later days:

React Developer Tools

Redux Developer Tools

What will we require to develop web app applications

a) Platform or base layer that runs the application, i.e. the Web Browser or Chrome, so we need to understand the platform

b) Programming Language or Languages to express the logic of the application

   i) HTML and Javascript

c) Libraries that help us manage repetitive tasks and provide higher level concepts than the programming language and platform itself - Like React

d) Tooling to create and manage projects - create react app. This itself uses a platform called node and npm.

e) Tooling to edit the code and diagnose any problems with the running program - Visual Studio Code, React Developer Tools, Redux Developer Tools.

So lets dive in

How does a browser show a web page.

Lets find out.

What happens when we load a website

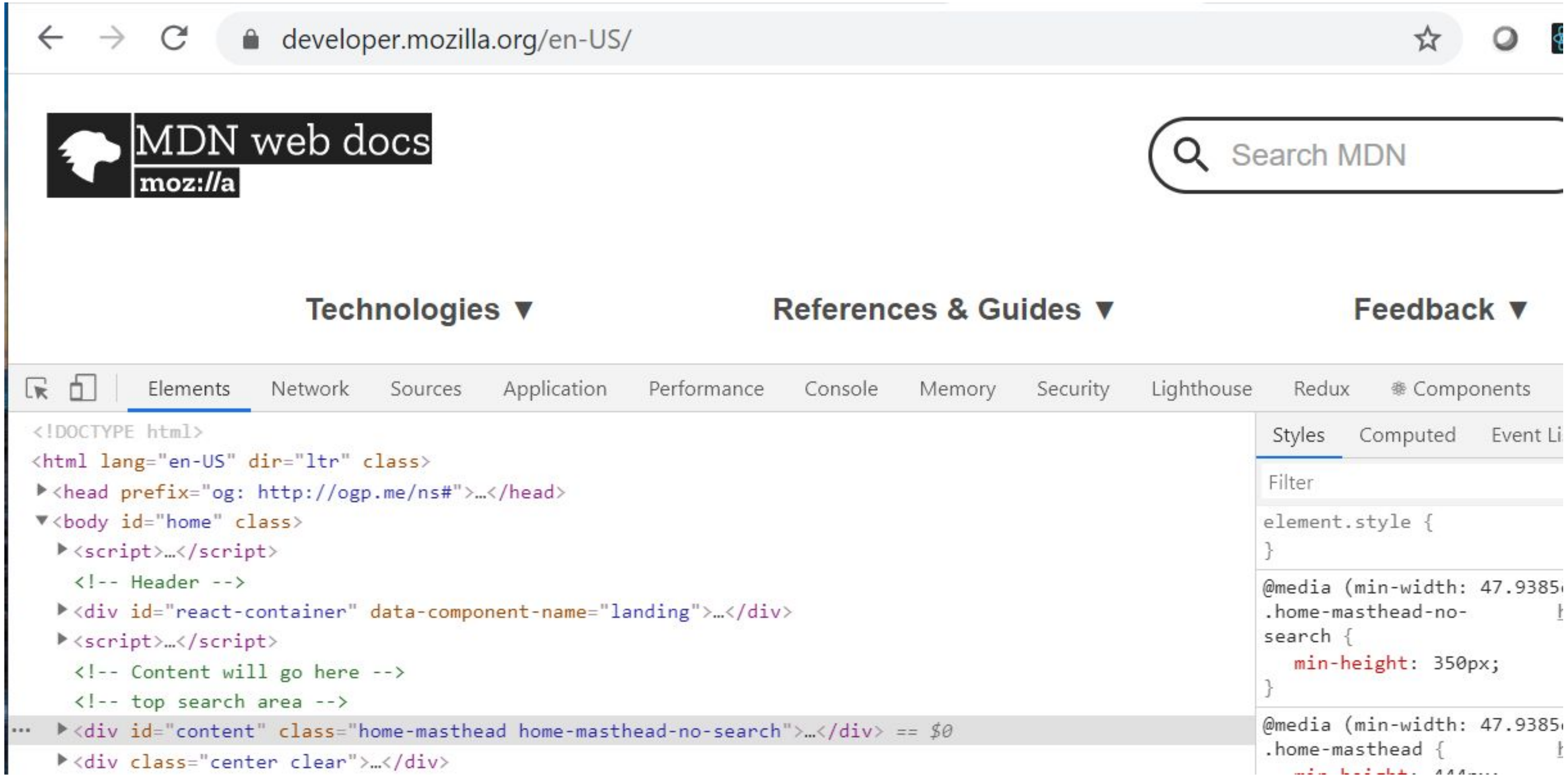
1. The browser goes to the DNS server, and finds the real address of the server that the website lives on (you find the address of the shop).
2. The browser sends an HTTP request message to the server, asking it to send a copy of the website to the client (you go to the shop and order your goods). This message, and all other data sent between the client and the server, is sent across your internet connection using TCP/IP.
3. If the server approves the client's request, the server sends the client a "200 OK" message, which means "Of course you can look at that website! Here it is", and then starts sending the website's files to the browser as a series of small chunks called data packets (the shop gives you your goods, and you bring them back to your house).
4. The browser assembles the small chunks into a complete website and displays it

How to look under the hood of what happens

Now we should be curious how this works. What are the details.

a) Open https://developer.mozilla.org/ in Chrome.
b) Right click anywhere in the content area
c) Click Inspect
d) This will open Chrome Developer Tools which lets us look under the hood of the website

How Developer tools looks to us



```
<!DOCTYPE html>
<html lang="en-US" dir="ltr" class>
  ▶<head prefix="og: http://ogp.me/ns#">…</head>
  ▼<body id="home" class>
    ▶<script>…</script>
     <!-- Header -->
    ▶<div id="react-container" data-component-name="landing">…</div>
    ▶<script>…</script>
     <!-- Content will go here -->
     <!-- top search area -->
... ▶<div id="content" class="home-masthead home-masthead-no-search">…</div> == $0
    ▶<div class="center clear">…</div>
```

```
element.style {
}
@media (min-width: 47.9385
.home-masthead-no-
search {
  min-height: 350px;
}
@media (min-width: 47.9385
.home-masthead {
```

So how to develop our own Web site.

a) Create a folder called jstraining
b) Fire up Visual Studio Code, and open this folder in VS code with File menu -> Open Folder
c) Create a file called index.html
d) In the edit type a exclamation and then tab to see the autogenerated code
e) Open integrated ter
f) Click view -> integrated terminal
g) This will open up the browser on 127.0.0.1:8080

Auto generated html code in index.html

```html
<!DOCTYPE html>
<html lang="en">
<head>
 <meta charset="UTF-8">
 <meta name="viewport" content="width=device-width, initial-scale=1.0">
 <meta http-equiv="X-UA-Compatible" content="ie=edge">
 <title>Document</title>
</head>
<body>
 </body>
</html>
```

Adding a script tag

Add a script tag

```
<script>
  console.log('Hello World')
</script>
```

Things to note : console is in global scope.

Output can be seen on the debug console.

Quick tour of javascript.

variables

A variable is a “named storage” for data. We can use variables to store .The statement below creates (in other words: *declares*) a variable with the name “message”:

```
let myvar;
```

we can check by console.log(myvar) or alert(myvar)

Now, we can put some data into it by using the assignment operator

```
let myvar = 5;

alert(myvar)
```

Javascript is dynamically typed

Variable identifiers do not have any declared type. Any type of value can be assigned to them.

```
let myvar = 5;

myvar = 'test'

myvar = true

yvar = null

myvar = Symbol()

myvar = { };
```

let vs const

`**const**` is a signal that the identifier won't be reassigned. `**let**` is a signal that the variable may be reassigned, such as a counter in a loop

```
const x = 10;

x = 12; // error
```

Javascript is weakly typed (coerces types)

2 + "4" is valid javascript

"4" + true

{} + ''

All of these are valid javascript

And basically **never** gives you a type error on any operation except on accessing Object properties.

Crazier examples

```
true + false

12 / "6"

"number" + 15 + 3

15 + 3 + "number"

[1] > null

"foo" + + "bar"

'true' == true

false == 'false'

null == ''
```

```
null == ''

!!"false" == !!"true"

[‘x’] == ‘x’

[] + null + 1

[1,2,3] == [1,2,3]

{}+[]+{}+[1]

!+[]+[]+![]

new Date(0) - 0

new Date(0) + 0
```

Data Types of Javascript

Six **Data Types** that are primitives, checked by typeof operator:

- undefined : `typeof instance === "undefined"`
- Boolean : `typeof instance === "boolean"`
- Number : `typeof instance === "number"`
- String : `typeof instance === "string"`
- BigInt : `typeof instance === "bigint"`
- Symbol : `typeof instance === "symbol"`

undefined

```
var a;
typeof a; //"undefined"

window.b;
typeof window.b; //"undefined"

a === undefined; // true
c === undefined; // Reference Error
```

The global **`undefined`** property represents the primitive value `undefined`

null and Objects

- null : `typeof instance === "object"`. Special primitive type having additional usage for it's value:
- Object : `typeof instance === "object"`. Special non-data but structural type for any constructed object instance also used as data structures: **new Object, new Array, new Map, new Set, new WeakMap, new WeakSet, new Date and almost everything made with new keyword;**

function Type

Functions return `typeof instance === "function"`.

function add(a,b) { return a + b }

typeof add returns "function"

undefined vs null

`undefined` is distinct from `null` which is also a primitive value representing the ***intentional*** **absence** of a value.

Similarity between `undefined` and `null` is they both coerce to false.

undefined is not stringified whereas null is stringified.

use null to denote absent value in your programs.

falsy and truthy values

A **falsy** (sometimes written **falsey**) value is a value that is considered false when encountered in a Boolean context.

There are 8 falsy values

All values except falsy are truthy values

These are falsy values

`false`	The keyword false
`0`	The number zero
`-0`	The number negative zero
`0n`	BigInt, when used as a boolean, follows the same rule as a Number. `0n` is *falsy*.
`""`	Empty string value
null	null - the absence of any value
undefined	undefined - the primitive value
NaN	NaN - not a number

examples of truthy - all evaluate to true

```
if (true)
if ({})
if ([])
if (42)
if ("0")
if ("false")
if (new Date())
if (-42)
if (12n)
if (3.14)
if (-3.14)
if (Infinity)
if (-Infinity)
```

Now lets put some html in the body here

```
 <div style="display: flex; flex-direction: column;
padding: 10px; width: 200px; height: 100px;
justify-content: space-between;">
   <div>
     <label for="quantity">x</label>
     <input id="x" type="number">
   </div>
   <div>
     <label for="quantity">y</label>
     <input id="y" type="number">
   </div>
   <button>+</button>
 </div>
 <div id="output">
 </div>
```

This creates 2 input elements that accept numbers
and an area to show the output.

Linking to a script from html

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

This script tag directs browser to load and execute this javascript file.

HTML and the DOM

Document Object Model -

Is a model(i.e. interfaces and methods) to create access and manipulate objects that refer to the HTML.

$0 is handy to inspect DOM in developer tools.

Some properties are common to all elements like event handlers.

onclick, onchange etc.

add.js

```
function handleXChanged(e) { // declare named function
 window.x = e.target.value;
}

function handleYChanged(e) {
 window.y = e.target.value;
};

document.getElementById('x').onchange = handleXChanged;
document.getElementById('y').onchange = handleYChanged;
```

Events in Javascript

events are actions or occurrences that happen in the system you are programming. e.g.

- The user clicking the mouse over a certain element or 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.

Full listing with tracking of input values

```html
<body>
 <div style="display: flex; flex-direction: column;
padding: 10px; width: 200px; height: 100px;
justify-content: space-between;">
   <div>
     <label for="quantity">x</label>
     <input  id="x" type="number">
   </div>
   <div>
     <label for="quantity">y</label>
     <input id="y" type="number">
   </div>
   <button id="add">+</button>
 </div>
 <div id="output">
 </div>
 <script src="add.js" />
</body>
```

```javascript
function handleXChanged(e) {
 window.x = e.target.value;
}

function handleYChanged(e) {
 window.y = e.target.value;
};

document.getElementById('x').onchange =
handleXChanged;
document.getElementById('y').onchange =
handleYChanged;
```

Tracking click and displaying output for add

```
function handleXChanged(e) {
 window.x = e.target.value;
}

function handleYChanged(e) {
 window.y = e.target.value;
};

function handleAddClick() {
 document.getElementById('output').innerHTML = Number(window.x) + Number(window.y)
}

document.getElementById('x').onchange = handleXChanged;
document.getElementById('y').onchange = handleYChanged;
document.getElementById('add').onclick = handleAddClick;
```

Javascript Objects

You access the properties of an object with a simple dot-notation:

```
objectName.propertyName
```

Or we can access with [ ] notation for dynamic property access

```
object[property]
```

Example of object literal

```
const person = {
 name: ['Bob', 'Smith'], // Array literal
 age: 32,
 gender: 'male',
 interests: ['music', 'skiing'],  // Array literal
 bio: function() {
   alert(this.name[0] + ' ' + this.name[1] + ' is ' + this.age + ' years old. He likes '
+ this.interests[0] + ' and ' + this.interests[1] + '.');
 },
 greeting: function() {
   alert('Hi! I\'m ' + this.name[0] + '.');
 }
};
```

Accessing properties and methods of object

```
person.name

person.name[0]

person.age

person.interests[1]

person.bio()

person.greeting()
```

JSON

Javascript Object Notation - is not a javascript literal but a data interchange format which is not dependent on language.

JSON has the following syntactical constraints:

- Object *keys* must be **strings** (i.e. a character sequence enclosed in **double** quotes `"`).
- The values can be either:
  - a string
  - a number
  - an (JSON) object
  - an array
  - `true`
  - `false`
  - `null`

When converting Javascript Objects to JSON

Based on those constraint

- all keys will be converted to string.
- undefined values and functions will be dropped.
- single quotes get converted to double quotes.

Possible improvements to code.

Instead of keeping state in window, we can keep state in a object called state.

const state = { };

// this is const as we won’t reassign the entire state. we will only change properties of the state.

e.g. state.x = e.target.value instead of window.x = e.target.value.

```javascript
const state = {x: null, y: null};

function handleXChanged(e) {
 state.x = e.target.value;
}

function handleYChanged(e) {
 state.y = e.target.value;
};

function handleAddClick() {
 document.getElementById('output').innerHTML = Number(state.x) + Number(state.y)
}

document.getElementById('x').onchange = handleXChanged;
document.getElementById('y').onchange = handleYChanged;
document.getElementById('add').onclick = handleAddClick;
```

Try the following on the console

```
JSON.stringify(person, null, 2)
```

This is a very common technique to analyze the objects in the application.

e.g.

```
JSON.stringify(state, null, 2)
```

Adding json data on the server

For learning ifsc data will be added.

https://raw.githubusercontent.com/siddharth-sharma/react-training/master/day1/ifsc.json

Copy this as ifsc.json into the jstraining folder.

Working with array data from a server.- add and link data.js

```
async function downloadData() {
 const res = await fetch('ifsc.json');
 window.ifscData = await res.json();
}

downloadData();
```

After this loads.

```
typeof ifscData

ifscData.length
```

Extract list of Banks from this data.

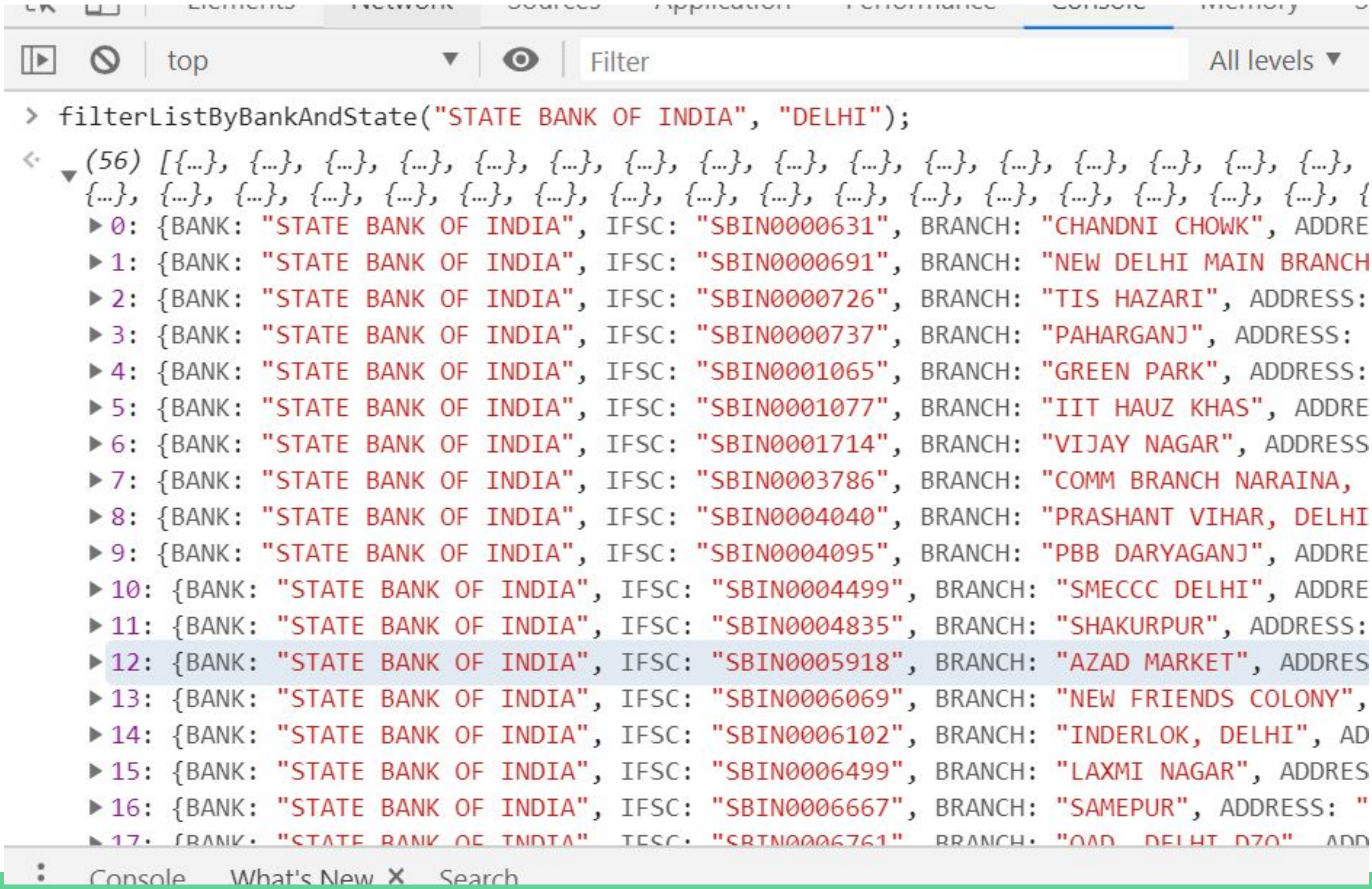
```
const bankList = new Set(ifscData.map(function(item) {

})
```

Calculate unique list of banks

```
const bankState = {};

function calculateUniqueBankList() {
 const fullBankList = ifscData.map(function(item){
   return item.BANK;
 })

 const bankSet = new Set(fullBankList)

 bankState.uniqueBankList = Array.from(bankSet);
}
```

Filter the list by criteria

```
function filterListByBank(bank) {
 return ifscData.filter(function(item) {
   return item.BANK === bank;
 })
}

function filterListByBankAndState(bank, state) {
 return ifscData.filter(function(item) {
   return item.BANK === bank && item.STATE === state;
 })
}
```



```
> filterListByBankAndState("STATE BANK OF INDIA", "DELHI");
<· ▼(56) [{…}, {…}, {…}, {…}, {…}, {…}, {…}, {…}, {…}, {…}, {…}, {…}, {…}, {…}, {…}, {…},
    {…}, {…}, {…}, {…}, {…}, {…}, {…}, {…}, {…}, {…}, {…}, {…}, {…}, {…}, {…}, {…}, {…}, {
    ▶0: {BANK: "STATE BANK OF INDIA", IFSC: "SBIN0000631", BRANCH: "CHANDNI CHOWK", ADDRE
    ▶1: {BANK: "STATE BANK OF INDIA", IFSC: "SBIN0000691", BRANCH: "NEW DELHI MAIN BRANCH
    ▶2: {BANK: "STATE BANK OF INDIA", IFSC: "SBIN0000726", BRANCH: "TIS HAZARI", ADDRESS:
    ▶3: {BANK: "STATE BANK OF INDIA", IFSC: "SBIN0000737", BRANCH: "PAHARGANJ", ADDRESS:
    ▶4: {BANK: "STATE BANK OF INDIA", IFSC: "SBIN0001065", BRANCH: "GREEN PARK", ADDRESS:
    ▶5: {BANK: "STATE BANK OF INDIA", IFSC: "SBIN0001077", BRANCH: "IIT HAUZ KHAS", ADDRE
    ▶6: {BANK: "STATE BANK OF INDIA", IFSC: "SBIN0001714", BRANCH: "VIJAY NAGAR", ADDRESS
    ▶7: {BANK: "STATE BANK OF INDIA", IFSC: "SBIN0003786", BRANCH: "COMM BRANCH NARAINA,
    ▶8: {BANK: "STATE BANK OF INDIA", IFSC: "SBIN0004040", BRANCH: "PRASHANT VIHAR, DELHI
    ▶9: {BANK: "STATE BANK OF INDIA", IFSC: "SBIN0004095", BRANCH: "PBB DARYAGANJ", ADDRE
    ▶10: {BANK: "STATE BANK OF INDIA", IFSC: "SBIN0004499", BRANCH: "SMECCC DELHI", ADDRE
    ▶11: {BANK: "STATE BANK OF INDIA", IFSC: "SBIN0004835", BRANCH: "SHAKURPUR", ADDRESS:
    ▶12: {BANK: "STATE BANK OF INDIA", IFSC: "SBIN0005918", BRANCH: "AZAD MARKET", ADDRES
    ▶13: {BANK: "STATE BANK OF INDIA", IFSC: "SBIN0006069", BRANCH: "NEW FRIENDS COLONY",
    ▶14: {BANK: "STATE BANK OF INDIA", IFSC: "SBIN0006102", BRANCH: "INDERLOK, DELHI", AD
    ▶15: {BANK: "STATE BANK OF INDIA", IFSC: "SBIN0006499", BRANCH: "LAXMI NAGAR", ADDRES
    ▶16: {BANK: "STATE BANK OF INDIA", IFSC: "SBIN0006667", BRANCH: "SAMEPUR", ADDRESS: "
    ▶17: {BANK: "STATE BANK OF INDIA", IFSC: "SBIN0006761", BRANCH: "OAD, DELHI DZO", ADD
```

Display options in a dropdown

```
Bank : <select id="bank"></select>

function fillBankListIntoSelect() {
 document.getElementById('bank')
 const bankSelect = document.getElementById('bank')

 bankState.uniqueBankList.forEach(function (element) {
   const newOption = document.createElement("OPTION");
   newOption.text = element;
   newOption.value = element;
   bankSelect.add(newOption)
 })
}
```

Arrow Functions - new more concise syntax.

```
function calculateUniqueBankList() {
 const fullBankList = ifscData.map(function(item){
   return item.BANK;
 })
```

```
becomes
```

```
const calculateUniqueBankList = () => {
 const fullBankList = ifscData.map(item => item.BANK);
 const bankSet = new Set(fullBankList)
 bankState.uniqueBankList = Array.from(bankSet);
}
```

this

value of `this` is determined by how a function is called (runtime binding)

```
function test() { console.log(this) }

test()

const myObject = {}

const bindTest = test.bind(myObject)

bindTest()
```

this inside a method call

```
const testObject = {
    foo: "a"
    test : function() {
        console.log(this)
    }
}
testObject.test();
```

```
// unnamed
let Rectangle = class {
 constructor(height, width) {
   this.height = height;
   this.width = width;
 }
};
console.log(Rectangle.name);
// output: "Rectangle"
```

Classes

In later versions of javascript ES2015 and above, Javascript has introduced classes.

```
class Rectangle {
 constructor(height, width) {
   this.height = height;
   this.width = width;
 }
}
const r = new Rectangle(5,5)
```

Must declare classes before using them

```
const p = new Rectangle(); // ReferenceError


class Rectangle {}
```

Class

```
class Rectangle {
 constructor(height, width) {
   this.height = height;
   this.width = width;
 }
 // Getter
 get area() {
   return this.calcArea();
 }
 // Method
 calcArea() {
   return this.height * this.width;
 }
}
```

Use arrow functions instead

```
class Rectangle {
 constructor(height, width) {
   this.height = height;
   this.width = width;
 }
 // Getter
 get area() {
   return this.calcArea();
 }
 // Method
 calcArea = () => {
   return this.height * this.width;
 }
}
```

Removing properties on an Object

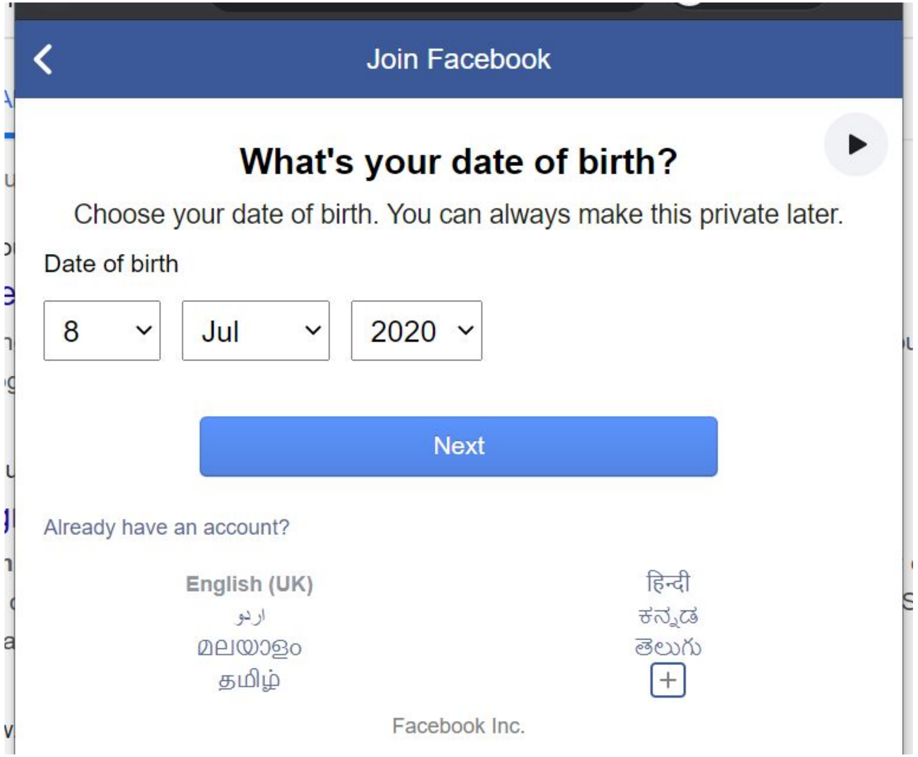
```
const Employee = {
 firstname: 'John',
 lastname: 'Doe'
};

console.log(Employee.firstname);
// expected output: "John"

delete Employee.firstname;

console.log(Employee.firstname);
// expected output: undefined
```

Exercise 1:

Record and upload atleast 10 minute video of inspecting any website of choice. Explain all the tabs you see and explain your understanding of whats going on in each tab.

Exercise 2

1) Create the following facebook page
2) Show Error in a red box if user is less than 13 years old.


Join Facebook

What's your date of birth?

Choose your date of birth. You can always make this private later.

Date of birth

8 Jul 2020

Next

Already have an account?

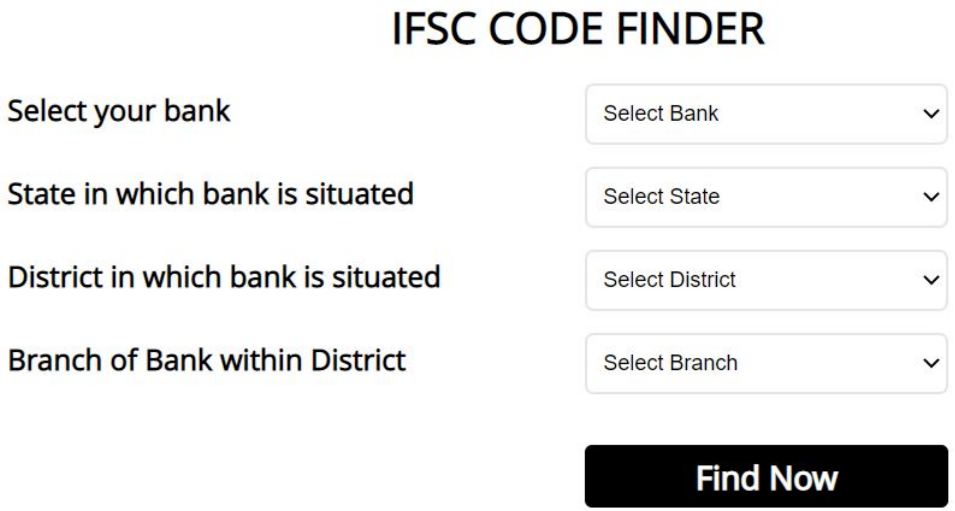
English (UK)
हिन्दी
اردو
ಕನ್ನಡ
മലയാളം
తెలుగు
தமிழ்
+

Facebook Inc.

Exercise 3: Implement IFSC Search Box

Using the following data
https://raw.githubusercontent.com/siddharth-sharma/react-training/master/day1/ifsc.json

implement
https://economictimes.indiatimes.com/wealth/ifsc-bank-code


IFSC CODE FINDER

Select your bank	Select Bank
State in which bank is situated	Select State
District in which bank is situated	Select District
Branch of Bank within District	Select Branch