Module -2

Responsive web design

Javascript is a programming language which is used by several websites to make the web pages more interactive.

It was developed initially by Netscape

applications

→ client side validation

→ displaying pop up windows and dialogue boxes (alert, prompt, confirm).

→ Javascript was officially named in 1995, it was named due to strategic patnership between Netscape and Sunmicrosystems, the creators of java despite the name javascript and java are fundamentally different language they have different syntax and semantics.

→ Java is a compiled language used for building largescale application, while javascript code [within body tag, head tag and external javascript file] is a scripting language they provide interactivity to website

→ Scripts in the head are typically executed before the browser renders the html document in the body, this means that javascript will run before any element element on the page are visible.

→ Scripts placed at the end of body tag are executed after the browser as rendered the html document it this means that javascript code can access and manipulate

all html elements on the page.

Comments

Single line comment
multi line comment

Rules for naming variables
* Names must start with a letter (a-z, A-Z, _, $) after 1st letter we can use digits (0-9).
* Variables are case sensitive

Code

```
<html>
  <head> <title> JS example </title> </head>
  <body>
    <script>
      var x=5;
      function add(){
        var x=10;
        document.write(x);
      }
      add();
      document.write(x);
    </script>
  </body>
</html>
```

3 ways of declaring javascript variables:
var
let
const

The javascript 'let' statement is used to declare a variable with 'let' statement we can declare a variable which is scoped. This means that variable with 'let' is only accessible within the block of code in which it is defined.
ex:

```
{
  let x = 10;
}
// x cannot be accessed
```

The 'var' keyword is as a function scope. It means that if you define the variable using 'var' keyword you can access it throughout the function.
Ex:

```
function add()
{
  if(true)
  {
    var x=10;
    let y=20;
  }
  // y can't be accessed
  // x can be accessed
}
```

Variables declared with const keyword are the values which remains unchanged.

```
const
  const y; } incorrect
  y = 10;
  const y=10; // correct
```

document object model (DOM)

```
<html>
 <body>
    <h1> J
    <h1 id="abc"> Javascript and its
          fundamentals </h1>
    <script>
    const e= document.getElementById("abc");
    e.style.color="red";
    </script>
 </body>
</html>
```

When the web page is loaded the browser creates document object model of the page. The way the document content is accessed and modified is called document object model. When a web page is loaded into browser window it becomes a document object. After that javascript can access html elements and perform other operation on them.

DOM Methods

getElementById: It is a method to access html elements using "Id".

innerHTML : this property is used to get the content of the element and to modify the html element.

code

```
<html>
<body>
<p id="xyz"> Javascript and its fundamental
                </p>
<h1 id="abc">  </h1>
<script>
  const e = document.getElementById("xyz").
        innerHTML;
  document.getElementById("xyz")
  document.getElementById("abc").innerHTML=e;
</script>
</body>
</html>
```

code

```
<html>
 <body>
   <p id="abc" onclick=fun() > Doctor </p>
<script>
     function fun(){

  document.getElementById("abc").innerHTML=
              "apple";

   }
</script>
</body>
</html>
```

code (ticket booking)

```
<!DOCTYPE>
<html>
<head>
    <title> demo example </title>
</head>
<body>
    <h1> Online Booking </h1>
    <button onclick="bookticket()"> Book
    ticket </button>
<script>
    function bookticket() {
    let destination= prompt("enter your
destination:");
    if(!destination)
    {
      alert("booking cancelled");
      return;
    }
    let date = prompt("enter travel date:");
    if(!date)
    { if
      alert("please enter the travel
          date-");
    return;
    }
    let passengers = prompt("enter the
no. of passengers");
    if(!passengers || isNaN(passengers))
    {
      alert("please enter the number of
      passengers to book ticket");
      return; }
```

```
let c = confirm("confirm your bookings to ${destination} on ${date} for ${passengers}");

if (c) {
  alert("booking confirmed");
}
  else {
    alert("something went wrong, try again");
  }
}
</script>
</body>
</html>
```

Array

```
<html>
<body>
  <script>
    const cars = [2, "Volvo", "BMW"];
    document.write(cars + "</br>");
    document.write(cars[2] + "</br>");
    cars.push("MIT", "IIT");
    document.write(cars + "</br>");
    document.write(cars.length + "<br>");
    cars.pop();
    document.write(cars cars);
  </script>
</body>
</html>
```

even number or odd code

```
<html>
<body>
<script>
let n= alert
    let n= prompt("enter a number:");
    if(n<=100)
    {
        for(let i=1 ; i<=n ; i++)
        {
            if(i%2==0){
            document.write(i);
        }
    }
}
</script>
</body>
</html>
```

@ media query

code

```
<html>
<head> <title> media query example </title>
<style>
    @ media only screen and (max-width:
                                1000px) {
    body{
        background-color: Red;
    }
}
@ media only screen and (max-width: 1600px)
{
```

```
body{
    background-color: green; }
}
</style>
</head>
<body>
    <h1> Responsive web page </h1>
    <p> Resize the window </p>
</body>
</html>
```

Media queries : Media queries in CSS allows you to apply different styles based on devices screensize, resolution. They are mainly used to make websites responsive, ensuring a good layout on mobiles, tablets and desktops.

breakpoints: These are specific screen width at which CSS applies different styles to create a responsive design. Here are commonly used breakpoints:
Mobile :- max-width : 600px
tablet :- max-width : 601px and ~~max-width : 1023px~~
max-width : 1024px
desktop :- min-width : 1025 px

background size property determines how a background image fits inside an element. The possible values are auto, cover, contain.

Auto: The image retains its original size.
→ If the img is smaller than the ~~contain~~ container it won't strech to fit.
→ If the image is larger than the container only a portion will be visible.

cover:
The image scales to cover the ~~entire~~ entire container.

contain:
→ The image scales to fit completely inside the container.
→ the entire image will always be visible but there may be empty spaces.

eg: Imagine a ~~300px~~ 300px × 200px div and 500 × 500px image.
auto: The image stays 500 × 500px, so only part of it shows.
cover: The image resize to fill the div completely but some parts are cut off.
contain: The whole image is visible, but there might be empty space around it.

Date
Page

Bootstrap

bootstrap is the most popular html, css and javascript framework for developing responsive and mobile friendly website. It is absolutely free to download and use. It is front end framework used for easier and ~~faster~~ faster development. It also include javascript plugins.

2 ways to link bootstrap:

→ CDN link (online)

→ download and use (getbootstrap.com) offline

code (without bootstrap)

```
<html>
  <head>
    <title> Bootstrap </title>
    <style>
      .b {
          background-color: blue;
          color: red;
          border-radius: 10px;
          border: 2px solid black;
      }
    </style>
  </head>
  <body>
      <button class="b"> Click me </button>
  </body>
</html>
```

code

```
<html>
<head>
```

```
<title> Bootstrap </title>
<link href="https://cdn.jsdelivr.net/npm/
  bootstrap@3.3.0/dist/css ... >

<body>
  <div class="container" style="background
  -color: blue;"> I am inside a
        container </div>
  <div class="container-fluid" style=
  "background-color: green;">
        <h1> I am inside a container
    fluid </h1>
  </div>
  </body>
  </html>
```

<

code (button with bootstrap)

```
<html>
 <head>
    <title> bootstrap <title>
    <link href="          " rel="stylesheet">
</head>
<body>
    <button class="btn btn-primary">
    button1 </button>
    <button class="btn btn-danger">
      button2 </button>

<body>
 <div class="container">
  </body>
  </html>
```

bootstrap supports 2 classes:
container: it provides responsive fixed width container that adapts to different screensizes.
container-fluid: It provides full width container that span's the ~~entire~~ entire view port width

It is responsive layout structure that allps to organize contents in a flexible and efficient way. It is based on 12 column layout allowing developers to create ~~responsible~~ responsive designs that adjust to different screen sizes.

Key features of bootstrap grid system
1) container-Based Layout: The grid system

2) → Rows and columns: Rows (.row) act as wrappers for columns (.col-*).
3) 12-column Grid: The layout divides into 12 equal parts, allowing different column arrangements.
4) Responsive classes: Classes like .col-sm-*, .col-md-*, .col-lg-*, and .col-xl-* adjust column width based on screen size.

code (grid - system using bootstrap)

```
<html>
  <head>
    <title> Bootstrap grid </title>
    <link href="  " rel="stylesheet" >
  </head>
<body>
  <div class="container" >
    <div class="row" >
      <div class="col-md-4 bg-primary> col 1
      </div>
    <div class="col-md-4 bg-primary> col 2</div>
  <div class="col-md-4 bg-primary> col 3 </div>
 </div>
 <div class="row">
   <div class="col-sm-3 bg-warnings> Col A </div>
   <div class="col-sm-3 bg-danger> Col B </div>
 </div>
</body>
<html>
```

bootstrap classes

→ jumbotron
container
alert - alert - warning
container fluid
btn btn - primary (blue)
btn btn - success (green)
btn btn - danger (red)
pagination
page - item
list - group
list - group item
table table - bordered table - hover

→ Jumbotron is a large attention grabing component used to highlight important content. It usually features a big headline, some descriptive text and it calls a action button.

→ It is designed to be visually impactful, making it ideal for displaying key messages or promotions

code

```
<html>
<head>
  <meta name="viewport" content="width-devic
- width ">
  <link rel="stylesheet" href="style.css">
</head>
<body>
    <div class="container">
    <div class="jumbotron">
    <p> I am inside jumbotron </p>
    <button class="btn-btn-danger"> click here
        </button>
    </div>
    </div>
<div><table class="table-bordered table-hove
    <tr>
     <td> a </td>
     </tr>
     <tr>
      <td> b </td>
      </tr>
      <tr>
       <td> c </td>
      </td>
      </tr>
```

```
</lable>
</div>
<div class="container">
<ul class="pagination">
<li><a href="#">previous</a></li>
<li><a href="#">1</a></li>
<li><a href="#">2</a></li>
<li><a href="#">next</a></li>
</ul>
</div>
<ul class="list-group">
<li class="list-group-item">1</li>
<li class="list-group-item">2</li>
</ul>
</body>
</html>
```

→ form validation

```
<body>
<div>
<h2>Simple form validation</h2>
<form class="needs-validation" novalidate>
<div>
<label>Name</label>
<input type="text" class="form-control"
id="name" required>
<div class="invalid-feedback">please
enter your name:</div>
<button class="btn btn-primary" type="submi
-t">Submit</button>
<script>
(function(){
'use strict';
var forms= var form= document.querySelect
-or('.needs-validation');
```

```
form.addEventListener('Submit', function(event){
  if(!form.checkValidity()){
      event.preventDefault();
  }
  form.classList.add('was-Validated');
});
})();

</script>
</body>
```

AJAX (asynchronous Javascript and XML)
It is a technique used in web development to create dynamic and interactive web application. It allows web pages to update the data asynchronously without ~~reload~~ reloading entire page

→ asynchronous: request are sent to server in the background without distrupting user experience
→ javascript: the scripting language used to handle AJAX
→ XML: Originally used for data exchange but now Json is more commonly used.
→ Readystate: A property used to know the status of the request

Value	description
0	→ xhr object is created but yet to configured the request.
1	→ The request is configured but yet to send the request.
2	→ The request as been sent to server
3	→ Server is busy to process the request

4 → Request is processed and response is ready

Code

```
<html>
<body>
  <button> Click me </button>
  <script>
  let button = document.getElementByTagName("button")[0];
  button.addEventListner('click', () => {
    let xhr = new XMLHttpRequest();
    console.log('UNSENT', xhr.readyState);
    xhr.open('GET', './index.txt', true);
    console.log('OPEN', xhr.readyState);
    xhr.send();
    xhr.onload = () => {
      console.log('DONE', xhr.readyState);
    }
});
  </script>
  </body>
  </html>
```