Java script and react

27 February 2024

20:13

W3school

* HTML : defines the content, CSS the layout, JavaScript the logic

-----------------------------------------------------------------------------------------

<!DOCTYPE html>

<html>

<body>

<h2>What Can JavaScript Do?</h2>

<p id="demo">JavaScript can change HTML content.</p>

<button type="button" onclick='document.getElementById("demo").innerHTML = "Hello JavaScript!"'>Click Me!</button>

</body>

</html>

---------------------------------------------------------------------------------------

* document.getElementById("demo").innerHTML = "Hello JavaScript"; // get HTML comp with id name demo and changes its content
* Both double and single quote works aspython
* document.getElementById("demo").style.fontSize = "35px"; // cane change the CSS style too
* Javascript and java are different
* Javascript is inserted between <script> tag
* -----------------------------------------------------------

<script>

document.getElementById("demo").innerHTML = "My First JavaScript";

</script>

---------------------------------------------------------------------

* Can be placed in <head> and <body> section
* Define and call a function

--------------------------------------------------------------

<!DOCTYPE html>

<html>

<head>

<script>

function myFunction() {

document.getElementById("demo").innerHTML = "Paragraph changed.";

}

</script>

</head>

<body>

<h2>Demo JavaScript in Head</h2>

<p id="demo">A Paragraph</p>

<button type="button" onclick="myFunction()">Try it</button>

</body>

</html>

---------------------------------------------------------------------------------------------

* Function can be placed in external files with .js and used <script src="myScript.js"></script> or <script src="https://www.w3schools.com/js/myScript.js"></script>
* JavaScript can be used to output in following ways : write In inner HTML, console, alert
* Semicolon separated statement
* Declare variable : let a, b, c;
* a= 10; //declares and assign

|  |  |
| --- | --- |
| **Keyword** | **Description** |
| var | Declares a variable |
| let | Declares a block variable |
| const | Declares a block constant |
| if | Marks a block of statements to be executed on a condition |
| switch | Marks a block of statements to be executed in different cases |
| for | Marks a block of statements to be executed in a loop |
| function | Declares a function |
| return | Exits a function |
| try | Implements error handling to a block of statements |

* Comments : same as c++, java
* Comma seperated assignement : let person = "John Doe", carName = "Volvo", price = 200;
* Default value of variables is undefined
* const define a const ref not a const val
* Make array, regex, function object always const :
* Operators
  + Arithmetic : same as c++ plus \*\* (exponential)
  + Assignment : same as c++

|  |  |
| --- | --- |
| **Operator** | **Description** |
| == | equal to |
| === | equal value and equal type |
| != | not equal |
| !== | not equal value or not equal type |
| > | greater than |
| < | less than |
| >= | greater than or equal to |
| <= | less than or equal to |
| ? | ternary operator |

* Logical : same as c++

|  |  |
| --- | --- |
| **Operator** | **Description** |
| typeof | Returns the type of a variable |
| instanceof | Returns true if an object is an instance of an object type |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Operator** | **Description** | **Example** | **Same as** | **Result** | **Decimal** |
| & | AND | 5 & 1 | 0101 & 0001 | 0001 | 1 |
| | | OR | 5 | 1 | 0101 | 0001 | 0101 | 5 |
| ~ | NOT | ~ 5 | ~0101 | 1010 | 10 |
| ^ | XOR | 5 ^ 1 | 0101 ^ 0001 | 0100 | 4 |
| << | left shift | 5 << 1 | 0101 << 1 | 1010 | 10 |
| >> | right shift | 5 >> 1 | 0101 >> 1 | 0010 | 2 |
| >>> | unsigned right shift | 5 >>> 1 | 0101 >>> 1 | 0010 | 2 |

* Datatype : String, Number, Bigint, boolean, undefined , Null, Symbol, Object(Array(list of python), object(dict of python), date)
* Same variable can hold different data as in python
* let x = BigInt("123456789012345678901234567890"); // to store big integers
* All javascript numbers are double

-------------------------------------------------

function myFunction(p1, p2) {

  return p1 \* p2;

}

------------------------------------------------

* Object properties can be accessed by [] or . Operator
* -----------------------------

const person = {

  firstName: "John",

  lastName : "Doe",

  id       : 5566,

  fullName : function() {

    return this.firstName + " " + this.lastName;

  }

};

--------------------------------------------------------

* This object same as c++

|  |
| --- |
| In an object method, this refers to the **object**. |
| Alone, this refers to the **global object**. |
| In a function, this refers to the **global object**. |
| In a function, in strict mode, this is undefined. |
| In an event, this refers to the **element** that received the event. |
| Methods like call(), apply(), and bind() can refer this to **any object**. |

* With new keyword : String , Number , Boolean become object. Avoid doing this as it complicates and slow things down

|  |  |
| --- | --- |
| **Event** | **Description** |
| onchange | An HTML element has been changed |
| onclick | The user clicks an HTML element |
| onmouseover | The user moves the mouse over an HTML element |
| onmouseout | The user moves the mouse away from an HTML element |
| onkeydown | The user pushes a keyboard key |
| onload | The browser has finished loading the page |

* let text = `He's often called "Johnny"`; // template string using `
* String methods: replace, trim, pad, substring, slice, Indexof, lastindexof, search etc.
* To pass regex, surround them /str/ instread of ""
* let text = `Welcome ${firstName}, ${lastName}!`; // toplace expression inside use (), even HTML can be placed
* For numeric operation like \* , / , JS tries to convert string to numeric
* Nan: not a number
* Number and its functions
* Arrays are like list in python and are objects in JS
* Array and its function
* const myArr = [[1,2],[3,4],[5,6]]; //2 d array
* numbers.forEach(myFunction); //array itteration
* Map() to apply same operation to all array element
* -----------------------------------------------------

const numbers1 = [45, 4, 9, 16, 25];

const numbers2 = numbers1.map(myFunction);

function myFunction(value, index, array) {

  return value \* 2;

}

-----------------------------------------------------------

* Similarly, filter() creates new array by filtering through
* Reduce to create a
* Other similar function are : every, some
* Few useful object and lib: Date, Math
* ?. : to check the null before accessing the property
* If, else, else if, break continue, , switch, default, for, while , do while : same as c++
* -----------------------------------------

for (key in object) { ///same "for of"

  // *code block to be executed*

}

-----------------------------------------------------------

* Break and continue can be used as goto label;
* Set, Maps(2d) : [] :
* Constructor keyword
* Regex : /*pattern*/*modifiers*;
* Try , catch, finally, throw
* Without let, var : variable becomes global
* Variables defined with let and const are hoisted to the top of the block, but not *initialized*.
* Hoisting: is JavaScript's default behavior of moving all declarations to the top of the current scope
* Use strict; //does not allow undeclared var use
* Lambda : let myFunction = (a, b) => a \* b;
* ---------------------

hello = function() {

  return "Hello World!";

}

-------------------------------------

class Car {

  constructor(name, year) {

    this.name = name;

    this.year = year;

  }

}

const myCar1 = new Car("Ford", 2014);

------------------------------------------------------

<script type="module">

import message from "./message.js";

</script>

------------------------------------------------------------

* To export add export keyword
* Debug in browser
* [JavaScript Reserved Words (w3schools.com)](https://www.w3schools.com/js/js_reserved.asp)
* Assignment op does not copy object but adress
* Direct creating constructor function creates class
* -----------------------------------------

function Person(first, last, age, eyecolor) {

  this.firstName = first;

  this.lastName = last;

  this.age = age;

  this.eyeColor = eyecolor;

  this.name = function() {

    return this.firstName + " " + this.lastName;

  };

}

--------------------------------------------------------------------

* Person.prototype.nationality = "English"; //to add new property to a class
* Iterable : must implement next()(return (next(val), done(bool)), to make for of work implement Symbol.iterator //returns next()
* Object class and its functions
* Adding () at the end of function def makes it a self invokable func
* JS function have a builtin object called arguments, with all details about arguments
* JS punction arguments are always pass by ref
* Call() of an object lets that function be called for other objects // similarly there is an apply()
* Bind() , used to lend function of one object to other
* Inheriance using extends, class Model extends Car // super() to access parent
* Static method to be called on object class
* Function can be passed as argument to other function and can be used as call back
* setInterval(myFunction, 1000); //to set timeout
* Promise (two callbacks one for success case one for failure)
* Adding async before a function makes it return promise, await can be used during call to wait for such promise/async func
* How HTML is separated in DOM, how DOM can be navigated and manipulated using JS

* 
* Dom has methods like : getElementById, getElementsByTagName, getElementsByClassName, createElement, removeChild, appendChild, replaceChild, write
* DOM has properties like innerHTML, attribute , style.Property
* One element like button etc can have multiple listener
* JS BOM(Browser object module) : this is to access window , screen, hsitory, location, timing , cookie etc. to change its properties like width etc
* Document/DOM can be accessed from BOM's window
* Use worker to run background task
* AJAX : async javascript and XML : updates web pages by fetching details from server
* Ajax can interact with php, database, asp,
* Jquery is kind of shorthand for document.getElementById etc .
* Various graphic library can be used to plot

React W3School

------------------------------------------------------

import React from 'react';

import ReactDOM from 'react-dom/client';

function Hello(props) {

return <h1>Hello World!</h1>;

}

const container = document.getElementById("root");

const root = ReactDOM.createRoot(container);

root.render(<Hello />);

-------------------------------------------------------

* React creates a virtual DOM, where it makes changes before making change to actual DOM
* Install npx and Node.js then run npx create-react-app <react app name>; cd <react app name> ; npm start; this will launch a webpage at localhost:3000
* Start at src/Index.js, app.js
* Calling react function(App) as below
* --------------------------------------

// Before

import ReactDOM from 'react-dom';

ReactDOM.render(<App />, document.getElementById('root'));

// After

import ReactDOM from 'react-dom/client';

const root = ReactDOM.createRoot(document.getElementById('root'));

root.render(<App />);

------------------------------------------------------

* To rendr HTML, react usage createRoot(HTML element) and render(){returns HTML element} // createroot detemines where element will be rendered
* JSX allow HTML tag inside javascript
* Use className instead of class in jSx
* const myElement =<h1>React is {5+5} times better with JSX</h1>; //expression inside jsx
* Use <div> </div> or <> </>to wrap multiple tab
* Class component // must extend React.Component and define render() returning html
* -------------------

class Car extends React.Component {

render() {

return <h2>Hi, I am a Car!</h2>;

}

}

* -----------------------------------
* Function component
* ----------------------------------------

function Car() {

return <h2>Hi, I am a Car!</h2>;

}

---------------------------------

* To use component root.render(<Car />);
* To pass props to component
* -------------------------

function Car(props) {

return <h2>I am a {props.color} Car!</h2>;

}

const root = ReactDOM.createRoot(document.getElementById('root'));

root.render(<Car color="red"/>);

-------------------------------------------

* A component can be used inside another componenet
* This.state{object} prop can be used to save the state
* Component state: Mounting, updating , unmounting
* Calls during mount
  1. constructor()
  2. getDerivedStateFromProps()
  3. render()
  4. componentDidMount()
* Calls during update
  1. getDerivedStateFromProps()
  2. shouldComponentUpdate()
  3. render()
  4. getSnapshotBeforeUpdate()
  5. componentDidUpdate()
* Calls during unmount : componentWillUnmount()
* onClick={shoot}  instead of onclick="shoot()" //both can be used, an builtin event object can also be passed
* Conditional rendering can be done using if , && and conditional ternary operator
* -----------------

function Goal(props) {

const isGoal = props.isGoal;

if (isGoal) {

return <MadeGoal/>;

}

return <MissedGoal/>;

}

return (

<>

<h1>Garage</h1>

{cars.length > 0 &&

<h2>

You have {cars.length} cars in your garage.

</h2>

}

</>

);

return (

<>

{ isGoal ? <MadeGoal/> : <MissedGoal/> }

</>

);

-----------------------------

* Map to add to list
* --------------------------

return (

<>

<h1>Who lives in my garage?</h1>

<ul>

{cars.map((car) => <Car brand={car} />)}

</ul>

</>

);

---------------------

* Using hook in func component
* ---------------------------\

import { useState } from 'react';

import ReactDOM from 'react-dom/client';

function MyForm() {

const [inputs, setInputs] = useState({});

const handleChange = (event) => {

const name = event.target.name;

const value = event.target.value;

setInputs(values => ({...values, [name]: value}))

}

const handleSubmit = (event) => {

event.preventDefault();

alert(inputs);

}

return (

<form onSubmit={handleSubmit}>

<label>Enter your name:

<input

type="text"

name="username"

value={inputs.username || ""}

onChange={handleChange}

/>

</label>

<label>Enter your age:

<input

type="number"

name="age"

value={inputs.age || ""}

onChange={handleChange}

/>

</label>

<input type="submit" />

</form>

)

}

const root = ReactDOM.createRoot(document.getElementById('root'));

root.render(<MyForm />);

---------------------------------------

* Similar approach for textarea and select
* Keep pages under src/pages
* Export default memo(Todos); //render todos only if they change
* React can use css styling too similar to HTML
* Hook
  + UseStatet : to set and get state
  + UseEffect : gets called on dependent change
  + useContext : to make global access of state
  + useRef:
  + useReducer
  + useCallback

Node.js w3schools

--------------------------------

* Download and install nodejs
* -----------------------------------------

var http = require('http');

http.createServer(function (req, res) {

res.writeHead(200, {'Content-Type': 'text/html'});

res.end('Hello World!');

}).listen(8080);

----------------------------------------------

* Init the js files through cmd "node myfirst.js"
* To import a module : var http = require('http');
* var fs = require('fs'); // have file handling function
* var url = require('url'); //url module,
* Other module are email ,db etc
* Node js module to server api(expressjs)
* ----------------------------------------

Const express = require('express');  
Const app = express();

app.get('/api/hello', (req, res) =>{  
 res.json({ message: 'Hello, API!'});  
});

app.listen(8000, () =>{  
 console.log('API server running on port 8000');  
});

---------------------------------------------------------------------

Apache web server

* Download and install
* Serves pages from var/www/html : put pages here to be served
* Same server can be used to create webserver