Unit - II

Functions and Objects

2.1 Functions

ஜாவாஸ்கிரிப்ட்டில், ஒரு function என்பது ஒரு குறிப்பிட்ட பணியைச் செய்யும் அல்லது மதிப்பைக் கணக்கிடும் குறியீட்டின் மறுபயன்பாட்டு தொகுதி ஆகும். Functionகள் codeஐ modularize செய்ய அனுமதிக்கின்றன, மேலும் ஒழுங்கமைக்கப்பட்ட, படிக்கக்கூடிய மற்றும் பராமரிக்க கூடியவையாக தருகிறது.

Function Basics

ஜாவாஸ்கிரிப்ட்டில் ஒரு function *“function”* என்ற Keyword பயன்படுத்தி வரையறுக்கப்படுகிறது, அதன்பின் ஒரு தனித்துவமான function பெயர், parameterகளின் பட்டியல் (காலியாக இருக்கலாம்) மற்றும் curly பிரேஸ்களுக்குள் கொடுக்கப்பட்ட ஒரு statement block இருக்கும்.

Syntax

```
function function_name()
{

}
```

Example

```
<!doctype HTML>
<html>
  <head>
        <title>Function Basics</title>
  </head>
  <body>
    <h1>Function Basics</h1>
  <script>
        function star()
        {
                document.write("*****");
        }
        star();
        document.write("<h1>Next Stars</h1>");
        star();
  </script>
  </body>
</html>
```

Passing Parameters

ஜாவாஸ்கிரிப்ட்டில் தகவல் function அனுப்பப்படும் போது, parameterகள் பயன்படுத்தப்படுகின்றன. ஒரு parameter என்பது functionல் பயன்படுத்தப்படும் ஒரு variable அல்லது மதிப்பு. Parameterகள் எப்போதும் ஜாவாஸ்கிரிப்டில் மதிப்பின் மூலம் அனுப்பப்படும். Functionகுள் செய்யப்படும் எந்த மாற்றமும் functionக்கு வெளியே பிரதிபலிக்காது. பல parameter மதிப்புகள் கமா(,) மூலம் பிரிக்கப்படுகின்றன.

Syntax

```
function function_name(parameter_list)
{

}
```

Example

```
<script>
  function star(n)
  {
```

```
            for(let i=0;i<n;i++)
                    document.write("*");
        }
        star(10);
        document.write("<h1>Next Stars</h1>");
        star(5);
      </script>
```

Return Statement

ஒரு function அதன் அழைப்பு இடத்திற்கு திரும்ப வேண்டிய மதிப்பைக் குறிப்பிட "*return*" அறிக்கை பயன்படுத்தப்படுகிறது. Function செயல்பாட்டிலிருந்து வெளியேறி மதிப்பை வழங்க வேண்டும் என்று return statement குறிக்கிறது.

Syntax

```
return value;
```

Example

```
<script>
  function star(n)
  {
        let ch="";
        for(let i=0;i<n;i++)
                ch += "*"
        return ch;
  }
  document.write(star(1));
  document.write("<h1>Next Stars</h1>");
  document.write(star(5));
</script>
```

Scope of Variable

Scope of variable programகுள் ஒரு குறிப்பிட்ட variableன் அணுகலைக் குறிக்கிறது. இரண்டு வகையான scopeகள் உள்ளன

1. Local
2. Global

Example of variable scope

```
<script>
  function msg()
  {
        let mymsg="JS Scope";
  }
  document.write("<h3>"+mymsg+"</h3>");
</script>
```

msg() function உள்ளே mymsg variable அறிவிக்கப்படுவதால் மேலே உள்ள ஸ்கிரிப்ட் எந்த outputடையும் காட்டாது, மேலும் அந்தச் functionக்கு வெளியே அதை access செய்ய முடியாது.

Local Scope

ஒரு function அல்லது blockன் உள்ளே அறிவிக்கப்பட்ட ஒரு variable local scopeஐ கொண்டுள்ளது. அந்த function அல்லது அது அறிவிக்கப்பட்ட blockக்குள் மட்டுமே இதை access செய்ய முடியும்.

Example

```
<script>
  function msg()
  {
        let mymsg="JS Scope";
  }
```

```
  function display()
  {
        document.write(mymsg);//Not accessible as mymsg is declared
  }                                  //in msg() function
  msg();
  display();
 </script>
```

Global Scope

எந்தவொரு function அல்லது blockன் வெளியே அறிவிக்கப்பட்ட ஒரு variable global scopeஐ கொண்டுள்ளது. Global variableகள் codeன் எந்தப் பகுதியிலிருந்தும், functionன் உள்ளேயும் வெளியேயும் அணுகலாம்.

Example

```
 <script>
  let mymsg = "Global Message";
  function msg()
  {
        document.write(mymsg+"<br>");//mymsg accessible as it is
  }                                            //a global variable

  function display()
  {
        document.write(mymsg);//mymsg accessible as it is
  }                                  //a global variable
  msg();
  display();
 </script>
```

Variable Naming for Scope

ஜாவாஸ்கிரிப்டில், variableகள் பயன்படுத்தப்படும் நோக்கத்தையும் சூழலையும் பிரதிபலிக்கும் அர்த்தமுள்ள மற்றும் விளக்கமான பெயர்களைக் கொடுப்பது முக்கியம். Global மற்றும் Local variableகளுக்கு ஒரே பெயரைப் பயன்படுத்த முடியாது. இதை மனதில் வைத்து, நோக்கத்தின் அடிப்படையில் variablகளுக்கு பெயரிடுவதற்கான சில வழிகாட்டுதல்கள் இங்கே உள்ளன.

- Global variableகள் 'g' முன்னொட்டாக இருக்க வேண்டும்.
- Local variableகள் அது பயன்படுத்தப்படும் function பெயருடன் முன்னொட்டாக இருக்க வேண்டும்.

Example

```
 <script>
  let gstu = 45;
  function exam()
  {
   examstu = 38;
   document.write("Student allowed for exam: "+ examstu+"<br>");
  }
  document.write("<h1>JavaScript Scope</h1>");
  exam();
  document.write("Total Student Strength: "+ gstu+"<br>");
 </script>
```

Inner Function and Closure

ஒரு inner function, nested function என்றும் அழைக்கப்படுகிறது, இது மற்றொரு functionகுள் வரையறுக்கப்பட்ட ஒரு function ஆகும். Inner functionகள் வெளிப்புற functionன் variableகள் மற்றும் parameterகளுக்கான accessஐ கொண்டுள்ளன.

Syntax

```
function outerFunction()
{
 Function Code
 function innerFunction()
 {
  Function Code
```

```
      }
    }
```

ஒரு function மற்றொரு functionக்குள் வரையறுக்கப்படும்போது ஒரு closure function உருவாக்கப்படுகிறது, மேலும் வெளிப்புற function execute செய்யப்பட்ட பிறகும் inner function அந்த வெளிப்புற functionன் variableகளுக்கான accessஐ தக்க வைத்துக் கொள்ளும்.

Example

```
<script>
      function counter() {
        let count = 0;

        function inc() {
              console.log(++count) ;
        }
        return inc;
      }

      let increment = counter();
      increment();
      increment();
</script>
```

இந்த எடுத்துக்காட்டில், 'counter()' ஆனது 'inc' inner functionஐ return செய்கிறது. `increment()' function அழைக்கப்படும் போது, அது outer scopeன் 'count' variableக்கான accessஐ கொண்டுள்ளது, மேலும் function callகளுக்கு இடையில் variable கிடைக்கும்.

Function as Object

ஜாவாஸ்கிரிப்டில் அனைத்தும் object. *new* keyword மற்றும் Function() objectஐ பயன்படுத்தி functionஐ object போன்ற வரையறுக்க முடியும். Function objectல் parameterகள் உள்ளன, அவை function argumentகள் மற்றும் function body வரையறுக்கப் பயன்படுகின்றன. இதில் பூஜ்ஜியம் அல்லது அதற்கு மேற்பட்ட function argumentகளைக் கொண்டிருக்கலாம், அவை ஆரம்ப parameterகளாக வழங்கப்படுகின்றன. கடைசி parameter function body ஆகும்.

Syntax

```
var variable = new Function(“arg1”, “arg2”, “function body");
```

Example

```
<script>
  let myfun = new Function("a","b","let c; c=a+b; alert(c);");
   myfun(2,3);
</script>
```

Function Literals

தனியான function declaration பயன்படுத்தாமல், ஒரு expressionல் நேரடியாக ஒரு function வரையறுக்க Function Literals பயன்படுத்தப்படுகின்றன. இது function keywordஐ பயன்படுத்தி உருவாக்கப்படுகிறது, அதைத் தொடர்ந்து functionன் பெயர் (விரும்பினால்), parameterகள் மற்றும் function body கொடுக்கப்பட்டுள்ளது.

Example

```
<script>
 let check = function test(a){
                               if(a>18)
                                 document.write("Voting allowed");
                               else
```

```
                    document.write("Voting not allowed");
                 };
  check(23);
 </script>
```

Anonymous Functions

Anonymous function என்பது பெயர் இல்லாத ஒரு function ஆகும். Anonymous functionகள் *function* expressionகளைப் பயன்படுத்தி உருவாக்கப்படுகின்றன மற்றும் குறுகிய காலத்திற்கு ஒரு function தேவைப்படும் இடங்களில் பயனுள்ளதாக இருக்கும் மற்றும் வேறு எங்கும் அதை மீண்டும் பயன்படுத்த முடியாது.

Example

```
</script>
        var button = document.getElementById("myButton");
        button.addEventListener("click", function() {
          alert("Button clicked!");
        });
</script>
```

இந்த எடுத்துக்காட்டில், button கிளிக் நிகழ்விற்கான எச்சரிக்கை செய்தியைக் காட்ட ஒரு anonymous function உருவாக்கப்பட்டது. இந்த function button கிளிக்குகளில் செயல்படுத்தப்படுகிறது, மற்ற இடங்களில் இதை அழைக்க முடியாது.

Static Variables

ஜாவாஸ்கிரிப்ட்டில் மற்ற மொழிகளைப் போல built-in static variableகள் இல்லை. ஜாவாஸ்கிரிப்டில் functionகளை objectகளாகப் பயன்படுத்துவதன் மூலம் static variableகளை உருவாக்கலாம். புள்ளி (.) குறியீட்டைப் பயன்படுத்தி functionஐ objectடுடன் நேரடியாக இணைக்கப்பட்ட ஒரு variable static variableஆக செயல்படுகிறது.

Example

```
 <script>
        function sum(x)
        {
                sum.total = sum.total + x;
                document.write(sum.total+"<br>");
        }
        sum.total = 0;
        sum(10);
        sum(5);
        sum(1);
 </script>
```

இந்த எடுத்துக்காட்டில், variable *total* function objectஉடன் sum.total என இணைக்கப்பட்டுள்ளது. இது variableஐ static variableஆக பயன்படுத்த அனுமதிக்கிறது.

Advanced Parameter Passing

ஜாவாஸ்கிரிப்ட் functionகள் objectகளால் ஆனவை, அவற்றுடன் தொடர்புடைய சில properties மற்றும் methodகள் உள்ளன. சில முக்கியமான properties

- **length**: ஒரு functionல் கொடுக்கப்பட்ட parameterகளின் எண்ணிக்கையை வழங்கும் Read only propertyஆகும்.
- **arguments[]** array: functionக்கு மாறுபட்ட எண்ணிக்கையிலான argumentகள் அனுப்பப்படும் போது இந்த array பயன்படுத்தப்படுகிறது.

Example

```
<script>
        function sum()
        {
          let tot=0;
          document.write("<br>Function Arguments: "+sum.length);
          if(arguments.length>0)
          {
           for(let i=0;i<arguments.length;i++)
           {
            tot += arguments[i];
           }
           document.write("<br>Sum of arguments is "+tot);
          }
        }
        sum(1,1,4,5);
</script>
```

மேலே உள்ள எடுத்துக்காட்டில், function *sum*() பூஜ்ஜிய parameterகளைக் கொண்டிருப்பதால் sum.length 0 ஐ வழங்கும். Function *sum()*க்கு அனுப்பப்படும் *argument*களைப் பெற array பயன்படுத்தப்படுகிறது.

Advanced Function Properties and Methods

ஜாவாஸ்கிரிப்ட் function objectக்கு caller property உள்ளது. Functionன் callerஐ அடையாளம் காண இந்த முறை பயன்படுத்தப்படுகிறது.

Example

```
<script>
        function rect(l,b)
        {
          area(l,b);
        }
        function squar(l)
        {
          area(l,l);
        }

        function area(l,b)
        {
          if(area.caller==rect)
            alert("Area of Rectangle: "+(l*b));
           else if(area.caller == squar)
            alert("Area of Square: "+(l*b));
        }

        squar(10);
        rect(3,2);
</script>
```

மேலே உள்ள எடுத்துக்காட்டில், function *area* இரண்டு வெவ்வேறு functionகளான rect மற்றும் squar ஆகியவற்றிலிருந்து அழைக்கப்படுகிறது. area functionன் *caller* property, area function அழைக்கப்படும் function பெயரை வழங்குகிறது.

Recursive Function

ஒரு recursive function தன்னைத்தானே அழைக்கும் ஒரு function ஆகும். Recursive functionகள் பொதுவாக ஒரு பணியை சிறிய problemகளாக பிரிக்கக்கூடிய பணிகளுக்குப் பயன்படுத்தப்படுகின்றன.

Example

```
<script>
  function odd(x)
  {
   if(x==0)
   {
          return;
   }
   else
```

```
  {
        if((x%2)!=0)
       document.write("<br>"+x);
        odd(--x);
  }
 }
 odd(10);
</script>
```

இந்த எடுத்துக்காட்டில் function odd() 0 க்கும், functionக்கு argumentஆக கொடுக்கப்பட்ட எண்ணுக்கும் இடையில் உள்ள ஒற்றைப்படை எண்ணைக் காட்ட மீண்டும் மீண்டும் அழைக்கப்படுகிறது.

2.2 Objects

ஜாவாஸ்கிரிப்ட் என்பது "object-based" ப்ரோக்ராம்மிங் மொழியாகும். Object-based மொழிகள் objectகளின் கருத்தையும், object-oriented ப்ரோக்ராம்மிங்ன் சில அம்சங்களையும் ஆதரிக்கின்றன. Objectகளுக்கு அவற்றுடன் தொடர்புடைய properties மற்றும் methods உள்ளன.

Objects in JavaScript

ஜாவாஸ்கிரிப்டில் உள்ள objectகள் data typeகளை பிரதிநிதித்துவப்படுத்துவது முதல் Document Object Model (DOM) வழியாக HTML ஆவணங்களை கையாளுதல் வரை, browserஉடன் இடைமுகப்படுத்துவது வரை மற்றும் பல வேறுபட்ட பாத்திரங்களை வகிக்கிறது. ஜாவாஸ்கிரிப்டில் உள்ள பொருள்கள் நான்கு குழுக்களாக வகைப்படுத்தப்பட்டுள்ளன.

1. **User-defined**: இந்த objectகள் புரோகிராமரால் உருவாக்கப்பட்டது.
2. **Native**: இந்த objectகள் ஜாவாஸ்கிரிப்ட் மொழியால் வழங்கப்படுகின்றன.
3. **Host**: இந்த objectகள் ஜாவாஸ்கிரிப்ட் மொழியின் பகுதியாக இல்லை, ஆனால் அவை ஹோஸ்ட் (உலாவி) மூலம் ஆதரிக்கப்படுகின்றன.
4. **Document**: இந்த objectகள் Document Object Modelன் ஒரு பகுதியாகும்.

Type	Example	Governing Standards
User-Defined	student, phone	N/A
Native	Array, Date	ECMAScript
Document	Image, HTMLInputElement	W3C DOM
Host	window, navigate	HTML5

Example

```
<script>
        let student = new Object();//User-Defined Object
        student.name = "Anbu";
        document.write("<h3>"+student.name+"</h3>");

        let today = new Date();//Native Object
```

```
        document.write("<h3>"+today+"</h3>");
        //Document Object
        let pagetitle = document.getElementById("title").innerHTML;
        document.write("<h3>"+pagetitle+"</h3>");

        let width = window.innerWidth;//Host Object
        document.write("<h3>"+width+"</h3>");

 </script>
```

Fundamentals

ஒரு *object* என்பது primitive types, functions, மற்றும் other objectகளின் வரிசைப்படுத்தப்படாத தரவுகளின் தொகுப்பாகும். ஒரு குறிப்பிட்ட பணிக்கு தேவையான அனைத்து data மற்றும் behavior ஒரே இடத்தில் கொடுக்கப்பட்டுள்ளது. பெரிய மற்றும் சிக்கலான programகளை எழுதும் போது objectகள் பயனுள்ளதாக இருக்கும்.

Object Creation

ஒரு Object *constructor* மற்றும் *new* Keywordஉடன் உருவாக்கப்படுகிறது. Constructor என்பது ஒரு சிறப்பு வகை function ஆகும், இது ஒரு புதிய objectஐ அது எடுக்கும் memoryஐ துவக்குவதன் மூலம் பயன்பாட்டிற்கு தயார் செய்கிறது. ஜாவாஸ்கிரிப்ட்டில் நேட்டிவ் objectகள் அவற்றின் constructorஐ பயன்படுத்தி உருவாக்கப்படுகின்றன மற்றும் user-defined objectகள் Object() கன்ஸ்ட்ரக்டரைப் பயன்படுத்தி உருவாக்கப்படுகின்றன.

Example

```
 <script>
   let student = new Object();//User-Defined Object
   student.name = "Anbu";
   document.write("<h3>"+student.name+"</h3>");

   let today = new Date();//Native Object
   document.write("<h3>"+today+"</h3>");
</script>
```

Object Literals

ஜாவாஸ்கிரிப்ட் object உருவாக்கத்திற்கான literal syntax ஆதரிக்கிறது. Syntax சுருள் பிரேஸ்களைக் கொண்டுள்ளது, இது கமாவால் பிரிக்கப்பட்ட property-value ஜோடிகளின் பட்டியலைக் கொண்டுள்ளது. Property-value ஜோடிகள் property பெயர், அதைத் தொடர்ந்து ஒரு colon மற்றும் அதன் மதிப்பை என்ற வகையில் கொடுக்கப்படுகிறது.

Syntax

```
    {
      property: value,
      property: function(){}
    };
```

Example

```
 let car = {
          brand: "Honda",
          model: "Civic",
          drive: function() {
                document.write("<h1>You are Driving
            "+this.brand+" "+this.model+"</h1>");
            }//Function End
        };
car.drive();
```

மேலே உள்ள எடுத்துக்காட்டில், object literal syntax பயன்படுத்தி ஒரு *car* object உருவாக்கப்படுகிறது.

Object Destruction and Garbage Collection

ஜாவாஸ்கிரிப்ட் objectகள் browserல் memory பயன்படுத்துகின்றன, ஜாவாஸ்கிரிப்ட்டில் ஒரு பொருள் உருவாக்கப்பட்டவுடன், interpreter தானாகவே memoryஐ ஒதுக்குகிறது. ஒரு objectக்கு ஒதுக்கப்பட்ட memoryஐ அதன் பயன்பாட்டிற்குப் பிறகு வெளியிடும் செயல்முறை "*Garbage Collection*" என்று அழைக்கப்படுகிறது. Garbage Collection செயல்முறையை JavaScript தானாகவே நிர்வகிக்கிறது. நிரலில் ஒரு object இனி அணுகாதபோது, objectல் ஆக்கிரமிக்கப்பட்ட memoryஐ interpreter வெளியிடுகிறது.

Properties

Objectகளில் property என்பது dataவை சேமிக்கப் பயன்படுத்தப்படும் பெயராகும். புள்ளி (.) ஆபரேட்டர் அல்லது அடைப்புக்குறி குறியீட்டைப் [] பயன்படுத்தி object property அணுகப்படுகிறது.

Example

```
 <script>
        let car = {
        brand: "Honda",
        model: "Civic",
        drive: function() {document.write("<h1>You are Driving "+this.brand+" "+this.model+"</h1>");}
        };
        car.drive();
        alert(car["brand"]+"\n"+car.model);
 </script>
```

மேலே உள்ள எடுத்துக்காட்டில், கார் ஆப்ஜெக்ட் பிராண்ட் மற்றும் மாடலின் பண்புகளை டாட் அல்லது பிராக்கெட் ஆபரேட்டரைப் பயன்படுத்தி அணுகலாம்.

Object are Reference Types

ஜாவாஸ்கிரிப்ட்டில் data typeகள் primitive அல்லது reference வகைகளாகும். ஒரு primitive வகை நேரடியாக தரவுகளை variableல் சேமிக்கிறது. ஒரு reference வகை உண்மையான மதிப்பை variableல் சேமிக்காது, அது உண்மையான தரவைக் கொண்ட memoryகான reference சேமிக்கிறது.

Example

```
 <script>
        let x,y;
        x = 10;
        y = x;
        x = 12;
        document.write("<h1>Value of x: "+x+"</h1>");
        document.write("<h1>Value of y: "+y+"</h1>");
        let car = {
                brand: "Honda",
                model: "Civic",
              drive: function() {
                    document.write("<h1>You are Driving "+this.brand+" "+this.model+"</h1>");}
              };
        car.drive();
        mycar = car;
        mycar.brand = "Tata";
```

```
        mycar.model = "Altroz";
        car.drive();
 </script>
```

மேலே உள்ள எடுத்துக்காட்டில் primary types *x* மற்றும் *y* நேரடியாக மதிப்புகளை சேமிக்கிறது, எனவே variable *x* இன் மதிப்பு 12 ஆக மாற்றப்படும் போது அது variable *y* இல் உள்ள மதிப்பைப் பாதிக்காது. *car* மற்றும் *mycar* ஆகிய objectகள் குறிப்பு வகைகளாகும், *mycar* objectல் உள்ள மதிப்பு மாற்றப்படும்போது, objectகள் அதே memory location குறிப்பிடுவதால், அது ஆப்ஜெக்ட் *car* பிரதிபலிக்கிறது.

Passing Objects to Functions

ஜாவாஸ்கிரிப்டில் objectகள் reference typeகளாகும், ஏனெனில் அவை உண்மையான dataவிற்கான reference குறிப்பிடுகின்றன. Functionகளுக்கு objectகளை parameterகளாக அனுப்பும் போது, functionல் உள்ள object கையாளுதல் கவனமாக செய்யப்பட வேண்டும், ஏனெனில் இது அழைப்பு இடத்திலும் மதிப்புகளை மாற்றும்.

Example

```
 <script>
        let data = {
        x: 10,
        y: 20
        };

        function swap_obj(obj)
        {
         let tmp;
         tmp = obj.x;
         obj.x = obj.y;
         obj.y = tmp;
        }
        swap_obj(data);
        document.write("<h4>Value of Object x: "+data.x+"</h4>");
        document.write("<h4>Value of Object y: "+data.y+"</h4>");
 </script>
```

மேலே உள்ள எடுத்துக்காட்டில், object *data swap_obj()* functionக்கு parameterஆக அனுப்பப்படும் போது, functionன் உள்ளே உள்ள objectன் x மற்றும் y properties உள்ள மதிப்புகளின் இடமாற்றம் செய்யப்பட்டது functionக்கு வெளியே பிரதிபலிக்கிறது.

Comparing Objects

ஜாவாஸ்கிரிப்ட்டில் இரண்டு objectகளை ஒப்பிடுவதற்கு comparison ஆபரேட்டர் பயன்படுத்தப்படும்போது அது உண்மையில் objectன் referenceஐ ஒப்பிடுகிறது, objectன் உள்ளடக்கத்தை அல்ல. அதாவது, இரண்டு objectகளும் ஒரே நினைவக இருப்பிடத்தைக் குறிப்பிடுகின்றனவா இல்லையா என்பதை comparison ஆபரேட்டர் சரிபார்க்கிறது.

Example

```
 <script>
   let str1 = "Hai";
   let str2 = "Hai";
   document.write(str1==str2);
   document.write("<br>");

   let strobj1 = new String("hai");
   let strobj2 = new String("hai");
```

```
  document.write(strobj1===strobj2);
</script>
```

மேலே உள்ள எடுத்துக்காட்டில், *str1* மற்றும் *str2* மாறிகள் stringஐ primitive typeகச் சேமிக்கின்றன, எனவே comparison ஆபரேட்டர் மதிப்புகளை ஒப்பிட்டு *"true"* வழங்குகிறது. *strobj1* மற்றும் *strobj2* மாறிகள் string மதிப்பை ஒரு string object சேமிக்கின்றன, எனவே comparison ஆபரேட்டர் objectகளின் memory referenceஐ ஒப்பிடுகிறது, மதிப்பை அல்ல, இது எப்போதும் *"false"* என்பதையே தரும்.

Prototype Based Objects

பெரும்பாலான ஆப்ஜெக்ட் சார்ந்த programming languagகளில் ஒவ்வொரு objectல் உள்ள குறியீட்டை விவரிக்க classகள் பயன்படுத்தப்படுகின்றன. இந்த class compile செய்யும் நேரத்தில் வரையறுக்கப்படுகிறது மற்றும் ஒருமுறை எழுதியவுடன் அதை மாற்ற முடியாது. ஜாவாஸ்கிரிப்ட்டில் calss குறியீடு இல்லை, எனவே ஏற்கனவே உள்ள objectகளின் பண்புகளை மாற்றியமைக்க முடியும். ஜாவாஸ்கிரிப்ட் ஒரு *prototype-based* object-oriented மொழி. ஒவ்வொரு objectக்கும் ஒரு prototype உள்ளது, இது objectன் அனைத்து நிகழ்வுகளிலும் பகிரப்படும் ஒரு property சேர்க்க பயன்படுகிறது.

Example

```
 <script>
   function student()
   {
     this.name =  "Anbu";
     this.dept =  "CE";
   }
   let stuobj = new student();
   console.log(stuobj.name);

   student.prototype.age = 19;

   let stuobj1 = new student();
   console.log(stuobj1.age);
   console.log(stuobj.age);
 </script>
```

மேலே உள்ள எடுத்துக்காட்டில், *stuobj* objectக்கு properties *name* மற்றும் *dept* உள்ளது. Objectன் prototypeஐ பயன்படுத்தி, ஒரு புதிய property age சேர்க்கப்படுகிறது, இது ஏற்கனவே உள்ள object *stuobj* மற்றும் புதிதாக உருவாக்கப்பட்ட object *studobj1* இரண்டிலும் கிடைக்கிறது.

Constructors

ஒரு கன்ஸ்ட்ரக்டர் என்பது constructor ன் புதிய நிகழ்வை உருவாக்கப் பயன்படும் ஒரு சிறப்பு functionஆகும். ஒரு constructor அழைக்கப்படும் போது, interpreter புதிய objectக்கு memory ஒதுக்குகிறார். இந்த keywordஐ பயன்படுத்தி objectன் propertyகளை constructorரால் அணுக முடியும்.

Syntax

```
function Constructor()
{
```

```
}
let obj = new Constructor();
```

Example

```
 <script>
   function Car(mdl,prc)
   {
     this.model = mdl;
     this.price = prc;
   }

   let honda = new Car("Civic","10 Lakhs");
   document.write("<h1>Car: "+honda.model+" +honda.price+"</h1>")
 </script>
```

மேலே உள்ள எடுத்துக்காட்டில், இரண்டு parameterகளை கொண்டு, *this* keyword பயன்படுத்தி *model* மாதிரி மற்றும் *price*க்கு மதிப்புகளை ஒதுக்கும் ஒரு Car constructor உருவாக்கப்பட்டது.

Inheritance

Inheritance என்பது பெற்றோரிடமிருந்து ஒரு குழந்தைக்கு propertyக்களை அனுப்ப அனுமதிக்கும் கருத்தாகும். ஜாவாஸ்கிரிப்ட்டில், prototypeகளைப் பயன்படுத்தி inheritance செயல்படுத்தபடுகிறது .

Example

```
 <script>
  function Bike()
  {
  }
  Bike.prototype.brand = "Yamaha";
  let myBike = new Bike();

  function SuBike()
  {
    this.cylinder = 4;
  }
  SuBike.prototype = Bike.prototype;

  mySuBike = new SuBike();
  alert( myBike.brand);
  alert(mySuBike.brand);
  alert(mySuBike.cylinder);
 </script>
```

மேலே உள்ள எடுத்துக்காட்டில் ஒரு *Bike* object *brand* propertyஉடன் உருவாக்கப்பட்டது. Property *cylinder* ஒரு புதிய பொருள் *SuBike* உருவாக்கப்பட்டது. *SuBike* objectன் prototypeப் பண்புகளைப் பயன்படுத்தி, *Bike* objectன் propertyகளைப் பெறுகிறது. இப்போது *SuBike* constructorப் பயன்படுத்தி உருவாக்கப்பட்ட பொருள் *brand* மற்றும் *cylinder* ஆகிய இரண்டிற்கும் அணுகலைக் கொண்டுள்ளது.

Overriding

ஜாவாஸ்கிரிப்ட் overriding ஆதரிக்கிறது. ஜாவாஸ்கிரிப்ட்டில் ஒரே பெயரில் பல functionகள் வரையறுக்கப்படும்போது கடைசியாக வரையறுக்கப்பட்டவை முன்பு வரையறுக்கப்பட்ட functionகளை மீறும். ஒவ்வொரு முறையும் function அழைக்கப்படும் போது, கடைசியாக வரையறுக்கப்பட்ட

ஒன்று செயல்படுத்தப்படும். ஜாவாஸ்கிரிப்டில் built-in object methodகளும் override செய்யலாம்.

Example

```
<script>
  function sum(a,b,c)
  {
        return (a+b+c);
  }
  function sum(a,b)
  {
        return (a+b);
  }
  function alert(msg)
  {
        document.write("<br>New Message is "+msg);
  }
  document.write(sum(10,20,30));
  alert("Welcome");
</script>
```

மேலே உள்ள எடுத்துக்காட்டில், *sum* functionன் முதல் வரையறை மூன்று parameterகள் மற்றும் இரண்டாவது வரையறை இரண்டு parameterகள் கொண்டது. *sum* function அழைக்கப்படும் போது அது எப்போதும் இரண்டு parameterகள் கொண்டுள்ள இரண்டாவது வரையறை அழைக்கும். இது overriding எனப்படும். இதேபோல் built-in *alert* method overriding செய்யப்படுகிறது, இது வலைப்பக்கத்தில் செய்தியைக் காட்டுகிறது.

2.3 Arrays

ஒரு array என்பது ஜாவாஸ்கிரிப்டில் உள்ள data structuresஆகும், இது ஒரு variable பல மதிப்புகளைச் சேமிக்கப் பயன்படுகிறது. எண்கள், ஸ்ட்ரிங், objectகள் மற்றும் பிற arrayகள் உட்பட பல்வேறு வகையான தரவுகளை arrayல் வைத்திருக்க முடியும். ஜாவாஸ்கிரிப்டில் உள்ள arrayன் சில முக்கிய பண்புகள்

1. **Ordered Collection**: ஒரு arrayல் உள்ள ஒவ்வொரு elementஐயும் அதன் index 0, 1 மற்றும் பலவற்றிலிருந்து அணுகலாம்.
2. **Dynamic Size**: Runtimeல் arrayன் அளவை மாற்றலாம்.
3. **Multiple Data Types**: Arrayகள் வெவ்வேறு தரவு வகைகளின் கூறுகளைக் கொண்டிருக்கலாம்.

ஜாவாஸ்கிரிப்ட்டில் ஒரு arrayஆனது, *literal syntax* அல்லது *Array()* constructor பயன்படுத்தி வரையறுக்கப்படுகிறது.

Array Literal []

ஒரு arrayஐ உருவாக்க array literalஆனது சதுர அடைப்புக்குறிகளை [] பயன்படுத்துகிறது. இது குறிப்பிட்ட மதிப்புகளுடன் ஒரு arrayஐ துவக்க பயன்படுகிறது. Arrayன் நீளம் என்பது literalல் கொடுக்கப்பட்ட argumentகளின் எண்ணிக்கை.

Syntax

```
let a = [val1,val2,...valn];
```

Example

```
<script>
  let a = [1,2,3];
  document.write(a);
</script>
```

Array Constructor

Array() கன்ஸ்ட்ரக்டரைப் பயன்படுத்தி ஒரு array உருவாக்கலாம். Arrayஐ துவக்க, constructorன் argumentகள் பயன்படுத்தப்படுகின்றன. ஒற்றை எண் argument குறிப்பிடப்பட்டால், அது arrayன் நீளத்தை வரையறுக்கிறது.

Syntax

```
let a = new Array(val1,val2,...valn);
```

Example

```
<script>
  let a = new Array(1,2,3);
  let b = new Array(10);//define size of the array
  document.write(a);
  document.write("<br>");
  document.write(b);
</script>
```

Accessing array elements

ஜாவாஸ்கிரிப்ட் arrayல் உறுப்புகள் index value பயன்படுத்தி அணுகப்படுகின்றன. Index value 0 முதல் arrayLength-1 வரை இருக்கும். அமைக்கப்படாத indexஐ அணுகுவது *"undefined"* என்று மதிப்பை தரும்.

Adding, Changing and Removing array elements

ஜாவாஸ்கிரிப்ட்டில் ஏற்கனவே உள்ள arrayல் ஒரு array elementஐ சேர்க்க, புதிய index value பயன்படுத்தப்படுகிறது, இது அந்த indexல் மதிப்பைச் சேர்க்கும். Array elementன் மதிப்பை மாற்றுவது, அந்த indexல் மதிப்பை மறுஒதுக்கீடு செய்வதன் மூலம் செய்யப்படுகிறது. ஒரு elementஐ நீக்குவது *delete* முறையைப் பயன்படுத்தி செய்யப்படுகிறது.

Example

```
<script>
  let a = [1,2,3];
  document.write("<h3>"+a[1]+"</h3>");//accessing

  document.write("<h3>"+a[3]+"</h3>");
  a[3] = 34;//changing
  document.write("<h3>"+a[3]+"</h3>");
  document.write("<h3>"+a+"</h3>");
  delete a[2];//removing
  document.write("<h3>"+a+"</h3>");
</script>
```

Important array Properties

ஜாவாஸ்கிரிப்டில் உள்ள arrayன் சில முக்கியமான properties மற்றும் methods

- **Length**: இந்தப் property அணிவரிசையின் நீளத்தை வழங்குகிறது.
- **Push**: இந்த method வரிசையின் முடிவில் argumentஐ சேர்க்கிறது.
- **Pop**: இந்த method arrayலிருந்து கடைசி elementஐ நீக்கி, திரும்பப் பெறுகிறது.

- **Shift**: இந்த method arrayயிலிருந்து முதல் உறுப்பை நீக்கி, திருப்பித் தருகிறது.

Example

```
<script>
  let a = [1,2];
  document.write("<h3>Length: "+a.length+"</h3>");
  a.push(23);
  document.write("<h3>Array: "+a+" Pop: "+a.pop()+"</h3>");

  document.write("<h3>Array: "+a+" Shift"+a.shift()+"</h3>");

  document.write("<h3>Final Array: "+a+"</h3>");
</script>
```

Manipulating Arrays

ஜாவாஸ்கிரிப்ட் array கையாளுதலைச் செய்யப் பயன்படுத்தப்படும் சில முக்கியமான methodகள்

- **concat()**: இந்த method இரண்டு arrayகளை இணைக்கப் பயன்படுகிறது மற்றும் ஒருங்கிணைந்த arrayஐ வழங்குகிறது.
- **join('ch')**: கொடுக்கப்பட்ட எழுத்தான 'ch' ஐப் பயன்படுத்தி arrayன் elementகளை ஒரு stringல் இணைக்க இந்த method பயன்படுத்தப்படுகிறது.
- **reverse()**: இந்த method arrayல் உள்ள உறுப்புகளை தலைகீழாக மாற்றுகிறது.
- **slice(start, end)**: இந்த method index *start* முதல் index *end* வரை உள்ள elementகளை கொண்ட ஒரு arrayஐ வழங்குகிறது.
- **sort()**: இந்த method arrayன் உள்ளடக்கத்தை ஏறுவரிசையில் வரிசைப்படுத்துகிறது.

Example

```
<script>
  let a = [1,2];
  let b = [3,4];
  a = a.concat(b);
  document.write("<h3>Array A: "+a+"</h3>");

  let str = b.join('-');
  document.write("<h3>String: "+str+"</h3>");

  a.reverse();
  document.write("<h3>Array A: "+a+"</h3>");

  let c = a.slice(0,2);
  document.write("<h3>Slice C: "+c+"</h3>");
</script>
```

Output

```
Array A: 1,2,3,4

String: 3-4

Array A: 4,3,2,1

Slice C: 4,3
```

Multidimensional Arrays

Multidimensional array என்பது ஒரு arrayஐ அதன் elementகளாகக் கொண்ட ஒரு array ஆகும். Multidimensional arrayல் உள்ள elementகளை அணுக, பல indexகள் பயன்படுத்தப்படுகின்றன.

Syntax

```
let array =[[],[]];
```

Example

```
<script>
  let a = [[1,2],[3,4]];
  document.write("<h3>Array Element Row1 Col2: "+a[0][1]+"</h3>");
</script>
```

Extending array with Prototypes

ஜாவாஸ்கிரிப்டில் built-in objectகளின் செயல்பாடுகளை prototypeகளைப் பயன்படுத்தி மாற்றியமைக்க முடியும். Array objectன் எந்த நிகழ்விலும் பயன்படுத்தக்கூடிய custom methodகளை உருவாக்க இது உதவுகிறது.

Syntax

```
Array.prototype.methodname = function(){ };
```

Example

```
<script>
  // Extending the Array prototype with a custom method
  Array.prototype.sum = function() {
    let total = 0;
    for (let i = 0; i < this.length; i++) {
        total += this[i];
    }
    return total;
  };

  // Using the custom method on an array
  let numbers = [1, 2, 3, 4, 5];
  let result = numbers.sum();
  document.write("<h2>Sum of Array: "+result+"</h2>");
</script>
```

மேலே உள்ள எடுத்துக்காட்டில், Array objectன் prototype propertyஐ பயன்படுத்தி, custom method sum சேர்க்கப்படுகிறது, இது arrayல் உள்ள அனைத்து elementகளின் மொத்தத்தைக் கண்டறியும்.

2.4 Date

Date object ஜாவாஸ்கிரிப்டில் தேதி மற்றும் நேரத்தை கையாளும் methodகளின் தொகுப்பைக் கொண்டுள்ளது. ஜவ்ஸகிரிப்ட்டில் dateஐ "*epoc*", ஜனவரி 1, 1970 முதல் மில்லி விநாடிகளின் எண்ணிக்கையாக கணக்கிடுகிறது. கிளையன்ட் இயந்திரத்தின் தேதி மற்றும் நேரத்தை JavaScript பயன்படுத்துகிறது, கிளையன்ட் இயந்திரம் தவறான தேதி மற்றும் நேரத்தைக் கொண்டிருந்தால் அது பிரதிபலிக்கும். வாரத்தின் நாட்கள் மற்றும் ஆண்டின் மாதங்கள் பூஜ்ஜியத்தில் தொடங்கி கணக்கிடப்படுகின்றன. ஞாயிறு 0, திங்கள் 1, ..., சனிக்கிழமை 6. ஜனவரி 0, ..., டிசம்பர் 11.

Creating Dates

ஜாவாஸ்கிரிப்ட்டில், Date() கன்ஸ்ட்ரக்டரைப் பயன்படுத்தி தேதிகள் உருவாக்கப்படுகின்றன. Constructor தேதி உருவாக்கத்திற்கான optional parameterகளை ஏற்றுக்கொள்கிறது.

Syntax

```
let today = Date();
```

Optional Parameters

- "month dd, yyyy hh:mm:ss" - குறிப்பிட்ட *month*, நாள்(*dd*), ஆண்டு(*yyyy*), மணிநேரம்(*hh*),

நிமிடம்(*mm*) மற்றும் second(*ss*) ஆகியவற்றைக் கொண்டு Date objectஐ உருவாக்குகிறது.

- milliseconds - epochற்குப் பிறகு மில்லி விநாடிகளின் முழு எண்ணாகக் குறிப்பிடப்படும் தேதியுடன் Date object உருவாக்குகிறது.
- yyyy, mm, dd - ஆண்டு (*yyyy*), மாதம் (*mm*) மற்றும் நாள் (*dd*) ஆகியவற்றிற்கான முழு எண் மதிப்புகளுடன் Date object உருவாக்குகிறது.

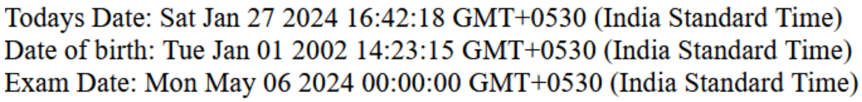
Example

```
<script>
  let today = new Date();
  document.write("<br>Todays Date: "+today);

  let dob = new Date("Jan 01, 2002 14:23:15");
  document.write("<br>Date of birth: "+dob);

  let examdate = new Date(2024, 04, 06);
  document.write("<br>Exam Date: "+examdate);
</script>
```

Output



```
Todays Date: Sat Jan 27 2024 16:42:18 GMT+0530 (India Standard Time)
Date of birth: Tue Jan 01 2002 14:23:15 GMT+0530 (India Standard Time)
Exam Date: Mon May 06 2024 00:00:00 GMT+0530 (India Standard Time)
```

Manipulating Dates

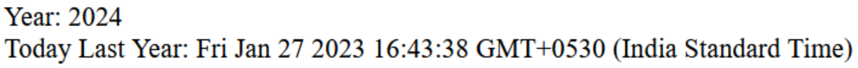
JavaScript இல் Date object படிக்க முடியாத மில்லி விநாடிகளாக மதிப்புகளை சேமிக்கிறது. ஜாவாஸ்கிரிப்ட் கையாளுதலுக்கான மனிதன் எளிதாக புரிந்துகொள்ளும் வடிவங்களை வழங்கும் methodகளைப் பயன்படுத்துகிறது. JavaScript ஆனது தேதி புலங்களைப் படிக்கவும் எழுதவும் *get* மற்றும் *set* methodகளை வழங்குகிறது.

Part of Date	Get Method	Set Method
Year	getFullYear()	setFullYear(Year)
Date	getDate()	setDate(Date)
Month	getMonth()	setMonth(Month)
Day	getDay()	--
Hour	getHours()	setHours(hour)
Minutes	getMinutes()	setMinures(minutes)
Seconds	getSeconds()	setSeconds(seconds)
Milliseconds	getMilliseconds()	setMilliseconds(millisec)

Example

```
<script>
  let today = new Date();
  document.write("<br>Year: "+today.getFullYear());
  today.setFullYear("2023");
  document.write("<br>Today Last Year: "+today);
</script>
```

Output



```
Year: 2024
Today Last Year: Fri Jan 27 2023 16:43:38 GMT+0530 (India Standard Time)
```

Converting Dates to Strings and Strings to dates

ஜாவாஸ்கிரிப்டில் தேதி என்பது stringஆக மாற்றப்பட வேண்டிய ஒரு பொருளாகும், அது ஒரு வலைப்பக்கத்தில் காட்டவும், databaseல் சேமிக்கவவும் மற்றும் குக்கீகளில் சேமிக்கவும் பயன்படுகிறது. தேதியை string மாற்ற JavaScript toString() முறை பயன்படுத்தப்படுகிறது.

Databaseல் சேமிக்கப்பட்ட மற்றும் text inputட்டிலிருந்து பெறப்பட்ட தேதிகள் ஜாவாஸ்கிரிப்ட் தேதி objectஆக மாற்றப்பட வேண்டும், இதனால் அதை நிரலில் கையாள முடியும். Stringஐ *millisecond* மாற்றும் *parse()* method மூலம் மாற்றம் செய்யப்படுகிறது, இது *Date()* constructorஐ பயன்படுத்தி தேதி பொருளாக மாற்றப்படுகிறது.

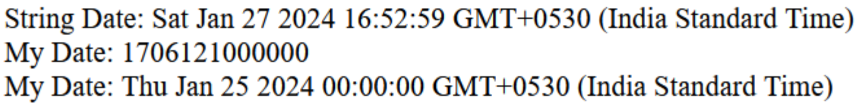
Syntax

```
let myDate = Date.parse(“12/14/2008”);
```

Example

```
<script>
  let today = new Date();
  document.write("<br>String Date: "+today.toString());
  let str = "01/25/2024";
  let myDate = Date.parse(str);
  document.write("<br>My Date: "+myDate);
  let myDateNew = new Date(myDate);
  document.write("<br>My Date: "+myDateNew);
</script>
```

Output



```
String Date: Sat Jan 27 2024 16:52:59 GMT+0530 (India Standard Time)
My Date: 1706121000000
My Date: Thu Jan 25 2024 00:00:00 GMT+0530 (India Standard Time)
```

2.5 Math

Math object அடிப்படை எண்கணித ஆபரேட்டர்களைக் காட்டிலும் மிகவும் சிக்கலான கணிதச் செயல்பாடுகளை அனுமதிக்கும் operationகளின் தொகுப்பைக் கொண்டுள்ளது. கணிதப் பொருளுக்கு constructor இல்லை. ஒரு objectஐ உருவாக்காமல் இதைப் பயன்படுத்தலாம்

Syntax

```
Math.method();
```

Math Methods

பொதுவாகப் பயன்படுத்தப்படும் சில ஜாவாஸ்கிரிப்ட் கணித methodகள் கீழே பட்டியலிடப்பட்டுள்ளன.

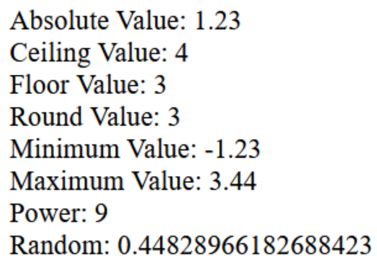
Method	Description
abs()	***Sign*** இல்லாமல் கொடுக்கப்பட்ட எண்ணின் முழுமையான மதிப்பு.
floor()	மிகப்பெரிய முழு எண் மதிப்பு, கொடுக்கப்பட்ட எண்ணை விட குறைவாக அல்லது சமமாக இருக்கும்.

ceil()	ஒரு சிறிய முழு எண் மதிப்பு, கொடுக்கப்பட்ட எண்ணை விட அதிகமாகவோ அல்லது சமமாகவோ இருக்கும்.
round()	கொடுக்கப்பட்ட எண்ணின் நெருங்கிய முழு எண் மதிப்பு.
min()	கொடுக்கப்பட்ட எண்ணின் சிறிய முழு எண் மதிப்பு
max()	கொடுக்கப்பட்ட எண்களின் குறைந்தபட்ச மதிப்பு.
pow()	அடித்தளத்தின் மதிப்பை அடுக்கு சக்திக்கு வழங்கும்.
random()	0 (உள்ளடங்கியது) மற்றும் 1 (பிரத்தியேகமானது) இடையே ரேண்டம் எண்ணை தருகிறது.

Example

```
<script>
  let a = -1.23;
  let b = 3.44
  document.write("<br>Absolute Value: "+Math.abs(a));
  document.write("<br>Ceiling Value: "+Math.ceil(b));
  document.write("<br>Floor Value: "+Math.floor(b));
  document.write("<br>Round Value: "+Math.round(b));
  document.write("<br>Minimum Value: "+Math.min(a,b));
  document.write("<br>Maximum Value: "+Math.max(a,b));
  document.write("<br>Power: "+Math.pow(3,2));
  document.write("<br>Random: "+Math.random());
</script>
```

Output



```
Absolute Value: 1.23
Ceiling Value: 4
Floor Value: 3
Round Value: 3
Minimum Value: -1.23
Maximum Value: 3.44
Power: 9
Random: 0.44828966182688423
```

Random Numbers

Math.random() முறையானது 0 மற்றும் 1 க்கு இடையில் ஒரு எண் மதிப்பை வழங்குகிறது. *m* முதல் *n* வரையிலான வரம்பில் random எண்ணை உருவாக்க, பின்வரும் சூத்திரம் பயன்படுத்தப்படுகிறது.

Math.round(Math.random()*(n-m))+m

Example

```
<script>
  let rand = Math.round(Math.random()* (10-1))+1;
  document.write("<br>Random Number between 1 - 10: "+rand);
</script>
```

Number Object

ஜாவாஸ்கிரிப்ட் Number object என்பது primitive எண் தரவு வகையைக் குறிக்கப் பயன்படுத்தப்படும் built-in objectஆகும். குறிப்பிட்ட பணிகளைச் செய்யப் பயன்படுத்தப்படும் அதன் சொந்த methodகள் உள்ளன.

Syntax

```
let a = new Number(value);
```

Methods

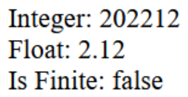
Method	**Description**

toString()	எண்ணை stringஆக மாற்றுகிறது.
Number.parseFloat(str)	ஒரு stringஐ மிதக்கும் புள்ளி எண்ணாக மாற்றுகிறது.
Number.parseInt(str)	ஒரு stringஐ முழு எண்ணாக மாற்றுகிறது
Number.isFinite()	கொடுக்கப்பட்ட மதிப்பு வரையறுக்கப்பட்ட எண்ணாக உள்ளதா என்பதைச் சரிபார்க்கிறது

__Example__

```
<script>
  let a = Number.parseInt("202212");
  let b = Number.parseFloat("2.12");
  let c = 1/0;
  document.write("<br>Integer: "+a);
  document.write("<br>Float: "+b);
  document.write("<br>Is Finite: "+Number.isFinite(c));
</script>
```

__Output__



```
Integer: 202212
Float: 2.12
Is Finite: false
```

2.5 Regular Expression

ஒரு Regular Expression ("regex" அல்லது "regexp") என்பது ஒரு தேடல் வடிவத்தை உருவாக்கும் எழுத்துகளின் வரிசையாகும். இது மின்னஞ்சல் முகவரி, கடவுச்சொற்கள் அல்லது மொபைல் எண்கள் போன்றவற்றை பொருத்திப்பார்க்க பயன்படுகிறது. Regular Expressionகள் JavaScript 1.2 இல் அறிமுகப்படுத்தப்பட்டன. ஜாவாஸ்கிரிப்ட்டில் Regular Expressionகளை இரண்டு வழிகளில் உருவாக்கலாம். இது RegExp கன்ஸ்ட்ரக்டர் மூலம் உருவாக்கப்படலாம் அல்லது patternஐ இணைக்க முன்னோக்கி சாய்வுகளை (/) பயன்படுத்தி உருவாக்கலாம். *test()* methodஐ பயன்படுத்தி pattern சரிபார்க்கப்படுகிறது.

__Syntax__

```
let regex = new RegExp('pattern');
let regex = /pattern/;
```

__Example__

```
<script>
  let pattern = new RegExp("www");
  //let pattern = /https:www/;
  let str = "www.google.com";
   document.write("<br>Comparing Pattern : "+pattern.test(str));
</script>
```

__Creating Patterns__

சிறப்பு எழுத்து வரிசைகளைப் பயன்படுத்தி patternகள் உருவாக்கப்படுகின்றன. ஒவ்வொரு சிறப்பு எழுத்துக்கும் ஒரு அர்த்தம் உள்ளது, இது patternஐ உருவாக்க பயன்படுகிறது. இந்த எழுத்துக்கள் meta characterகள் என்று அழைக்கப்படுகின்றன. பின்வரும் meta character வகைகளை பயன்படுத்தி patternகள் உருவாக்கப்படுகின்றன

- Positional Indicators
- Escape Codes
- Repetition Quantifiers

- Grouping
- Character Classes
- Alternatives

Positional Indicators

இந்த மெட்டா எழுத்துகள், stringல் எந்த இடத்தில் பேட்டர்ன் பொருத்தப்பட வேண்டும் என்பதைக் குறிப்பிடப் பயன்படுகிறது. நிலையை குறிக்கும் இரண்டு எழுத்துக்கள் உள்ளன. கேரட் சின்னம் (^) stringன் தொடக்கத்தில் செய்ய வேண்டிய பொருத்தத்தைக் குறிப்பிடுகிறது மற்றும் டாலர் சின்னம் ($) stringன் முடிவில் செய்ய வேண்டிய பொருத்தத்தைக் குறிப்பிடுகிறது.

Example

```
<script>
  let pattern = /^www/;
  let str = "www.google.com";
  document.write("<br>Comparing Pattern : "+pattern.test(str));
</script>
```

மேலே உள்ள எடுத்துக்காட்டில், கேரட் சின்னத்தை (^) பயன்படுத்தி stringன் தொடக்கத்தில் "www" என்ற பேட்டர்ன் பொருந்தும்படி குறிப்பிடப்பட்டுள்ளது.

Escape Codes

Escape codeகள் அதன் உண்மையான மெட்டா கேரக்டர் அர்த்தம் இல்லாமல் பேட்டர்னில் ஒரு சிறப்பு எழுத்து தேவைப்படும் போது பயன்படுத்தப்படும். மெட்டா எழுத்தின் முன் பின்சாய்வு (/) வைப்பதன் மூலம் இது செய்யப்படுகிறது. எஸ்கேப் குறியீடுகளின் மாதிரிகள் \[, \], \(, \), \{, \}, \|, \?, *, \+, \\, இவை எழுத்துப்பூர்வ எழுத்துகளுடன் பொருந்தும் [,], (,) , {, }, |, ?, *, +, \.

Example

```
<script>
  let pattern = /^\+91/;
  let str = "+91 9993300011";
  document.write("<br>Comparing Pattern : "+pattern.test(str));
</script>
```

மேலே உள்ள எடுத்துக்காட்டில், பேட்டர்னுக்கு அதன் மெட்டா கேரக்டர் அர்த்தம் இல்லாமல் + சின்னம் தேவைப்படுகிறது, எனவே பேட்டர்னில் ஒரு பின்சாய்வு அதன் முன் வைக்கப்படுகிறது.

Repetition Quantifiers

ஒரு கேரக்டர், குரூப் அல்லது கேரக்டர் கிளாஸ் ஒரு patternல் எத்தனை முறை திரும்பத் திரும்ப வேண்டும் என்பதைக் குறிப்பிட, ரிப்பீஷன் குவாண்டிஃபையர்கள் பயன்படுத்தப்படுகின்றன. Repetition quantifierகள் கீழே உள்ள அட்டவணையில் கொடுக்கப்பட்டுள்ளன.

Quantifiers	Description
n+	Matches any string that contains at least one n
n*	Matches any string that contains zero or more occurrences of n

n?	Matches any string that contains zero or one occurrences of n
n{X}	Matches any string that contains a sequence of X number of n's
n{X,Y}	Matches any string that contains a sequence of X to Y number of n's
n{X,}	Matches any string that contains a sequence of at least X number of n's

Example

```
 <script>

   let pattern = /^w*/;
   let str = "www.";

   document.write("<br>Comparing Pattern : "+pattern.test(str));
 </script>
```

மேலே உள்ள எடுத்துக்காட்டில், stringல் உள்ள 'w' என்ற எழுத்தை பூஜ்ஜியம் அல்லது அதற்கு மேற்பட்ட நிகழ்வுகளுடன் பொருந்தக்கூடிய repetition quantifier * பயன்படுத்தப்படுகிறது.

Grouping

Patternன் ஒரு பகுதியை ஒற்றை அலகாகக் கருதுவதற்கு grouping பயன்படுத்தப்படுகிறது. இது அடைப்புக்குறிகளைப் பயன்படுத்தி குறிப்பிடப்படுகிறது (). Groupingற்குப் பிறகு repetition quantifier வைக்கப்படும்போது அது groupன் உள்ளடக்கங்களுக்குப் பயன்படுத்தப்படும்.

Example

```
 <script>

   let pattern = /^w*.(tn)+/;
   let str = "www.tngptc.com";

   document.write("<br>Comparing Pattern : "+pattern.test(str));
 </script>
```

மேலே உள்ள எடுத்துக்காட்டில், 'tn' stringன் ஒற்றை அலகு ஒன்றை உருவாக்க, grouping பயன்படுத்தப்படுகிறது, அதைத் தொடர்ந்து repetition quantifier + பொருந்தும் பொது 'tn' stringல் ஒரு முறையாவது இருக்க வேண்டும் என்பதைக் குறிப்பிடுகிறது.

Character Classes

Character Classகள் சதுர அடைப்புக்குறிகளைப் பயன்படுத்தி குறிப்பிடப்படுகின்றன []. சதுர அடைப்புக்குறிக்குள் குறிப்பிடப்பட்ட எழுத்துக்களின் குழுவிலிருந்து எந்த ஒரு எழுத்தையும் பொருத்த இது பயன்படுகிறது. மதிப்புகளின் வரம்பைக் குறிப்பிட, 0-9 எண்களுக்கு கோடு (-) பயன்படுத்தப்படுகிறது மற்றும் a-z மற்றும் A-Z எழுத்துக்களுக்கு பயன்படுத்தப்படுகின்றன.

Example

```
 <script>

   let pattern = /^\+91 [9876][0-9]{9}/;
   let str = "+91 6993300011";
```

```
  document.write("<br>Comparing Pattern : "+pattern.test(str));
</script>
```

மேலே உள்ள எடுத்துக்காட்டில், 9,8,7 அல்லது 6 இல் தொடங்கும் 10 இலக்க மொபைல் எண்ணுடன் பொருத்த ஒரு பேட்டர்ன் உருவாக்கப்பட்டுள்ளது மற்றும் 0-9 வரம்பிலிருந்து மீதமுள்ள 9 இலக்கங்கள் கொடுக்கப்பட்டுள்ளன.

Alternatives

குழாய் சின்னத்தை (|) பயன்படுத்தி அல்டெர்னட்டிவ் வழங்கப்படுகிறது. இது பல உருப்படிகளின் தருக்க OR ஐக் குறிக்கப் பயன்படுகிறது.

Example

```
<script>

  let pattern = /^w{3}\.[0-9a-zA-Z]*\.(com|co.in|in)$/;
  let str = "www.google.in";

  document.write("<br>Comparing Pattern : "+pattern.test(str));
</script>
```

மேலே உள்ள எடுத்துக்காட்டில், stringன் இறுதியில் "com", "co.in" அல்லது "in" ஐப் பொருத்துவதற்கு, groupingல் alternative பயன்படுத்தப்படுகிறது.

RegExp Object

JavaScript இல் RegExp ஆப்ஜெக்ட்டைப் பயன்படுத்தி regular expression உருவாக்கப்படுகிறது. இது பின்வரும் முறைகளைக் கொண்டுள்ளது

- **test()**: இந்த method stringஐ வாதமாகப் பொருத்தி சரி அல்லது தவறு என வழங்கும்
- **Substring**: தொகுக்கப்பட்ட patternன் பகுதியுடன் பொருந்தக்கூடிய ஒரு சரத்தைப் பிரித்தெடுக்க இது பயன்படுகிறது. இது $1 - $9 பயன்படுத்தி காட்டப்படுகிறது.
- **compile()**: இந்த method ஏற்கனவே உள்ள regular expression புதியதாக மாற்ற பயன்படுகிறது.
- **exec()**: இந்த method stringல் ஒரு பொருத்தத்தை சோதிக்கவும் மற்றும் ஒரு arrayயை திரும்பவும் பயன்படுத்தப்படுகிறது. Length property, திரும்பிய arrayன் நீளத்தை வழங்கும் மற்றும் index property stringல் எந்த நிலையில் pattern பொருந்தியது என்பதை வழங்கும்.
- **toString()**: இந்த method regular expressionஐ stringஆக மாற்றுகிறது.

Example

```
<script>

  let pattern = RegExp("[9876][0-9]{7}[12]{2}");
  let str = "6993300012";
  document.write("<br>Comparing Pattern : "+pattern.test(str));

  let ptrn1 = RegExp("^(0[1-9]{2,4}) ([1-9][0-9]{6,7})");
  let phone = "0452 2673631 ";

  document.write("<br>Comparing Pattern : "+ptrn1.test(phone));
  document.write("<br>STD Code : "+RegExp.$1);
```

```
pattern.compile("[9876][0-9]{6}[3]{3}");
let ph1 = "9893487333";
document.write("<br>Comparing Phone Number1 : "+pattern.test(ph1));

  let res = ptrn1.exec(phone);
  document.write("<br>Array Length : "+res.length);
  document.write("<br>Array Element: "+res[1]);
  document.write("<br>Index of string match: "+res.index);

  document.write("<br>Convert RegExp to String: "+ptrn1.toString());
 </script>
```

Output

```
Comparing Pattern : true
Comparing Pattern : true
STD Code : 0452
Comparing Phone Number1 : true
Array Length : 3
Array Element: 0452
Index of string match: 0
Convert RegExp to String: /^(0[1-9]{2,4}) ([1-9][0-9]{6,7})/
```

String Methods for Regular Expression

String object regular expression பயன்படுத்தும் நான்கு முறைகளைக் கொண்டுள்ளது. அவை கீழே கொடுக்கப்பட்டுள்ளன

- **search()**: இந்த முறை பொருந்தக்கூடிய substringன் indexஐ வழங்குகிறது.
- **split()**: இந்த method ஒரு stringஐ sub-stringகளாகப் பிரித்து, கொடுக்கப்பட்ட regular expressionன் அடிப்படையில் ஒரு arrayயை வழங்குகிறது.
- **replace()**: இந்த method கொடுக்கப்பட்ட stringஐ regular expressionஉடன் பொருந்தும் முதல் இடத்தில மாற்றியமைக்கும்.
- **match()**: இந்த method regular expressionஉடன் பொருந்தக்கூடிய stringஐ வழங்குகிறது.

Example

```
<script>

  let pattern = RegExp("[0]{3}");
  let str = "6993000111";
  document.write("<br>Search Pattern : "+str.search(pattern));

  let dt = "21-01/2015";
  let exp = /[-|\/]/g;
  let arr = dt.split(exp);
  document.write("<br>Date Array: "+arr[1]);

  let dt1 = dt.replace(exp,'_');
  document.write("<br>New Date: "+dt1);

  let yr = dt.match(/2[0-9]/);
  document.write("<br>Year: "+yr[1]);
</script>
```

Output

```
Search Pattern : 4
Date Array: 01
New Date: 21_01_2015
Year: undefined
```