# JavaScript Fundamentals

OLGUN DUTKAN

# Presentation Content

1. Introduction
2. Basic Concepts
3. Arrays
4. DOM Manipulation
5. Asynchronous Programming
6. Error Handling
7. ECMAScript
8. Conclusion

# 1. Introduction

1.1. What is JavaScript?
1.2. History of JavaScript
1.3. Why JavaScript is important
1.4. A "Hello world!" example

# 1.1. What is JavaScript?

JavaScript is a high-level, dynamic programming language that is widely used for creating interactive web pages and web applications.

All major web browsers have a dedicated JavaScript engine to execute the code on users' devices.

It supports both client-side and server-side scripting.

# 1.2. History of JavaScript

JavaScript was created in 1995 by Brendan Eich at Netscape Communications as a way to add interactive functionality to web pages. Originally called Mocha, then later renamed to LiveScript, it was eventually named JavaScript as a marketing ploy to capitalize on the popularity of Java at the time.

Netscape

# 1.2. History of JavaScript

JavaScript quickly became popular for its ability to create dynamic and interactive web pages. Its popularity continued to grow with the advent of AJAX (Asynchronous JavaScript and XML) in the early 2000s, which allowed for even more dynamic and responsive web applications.


the-year-of-major-updates.jpg

# 1.2. History of JavaScript

JavaScript was later standardized by ECMA International. These standards were published under the name ECMAScript. ECMAScript is a language that defines the core features of JavaScript, and JavaScript is developed in compliance with ECMAScript standards.

# 1.3. Why JavaScript is important

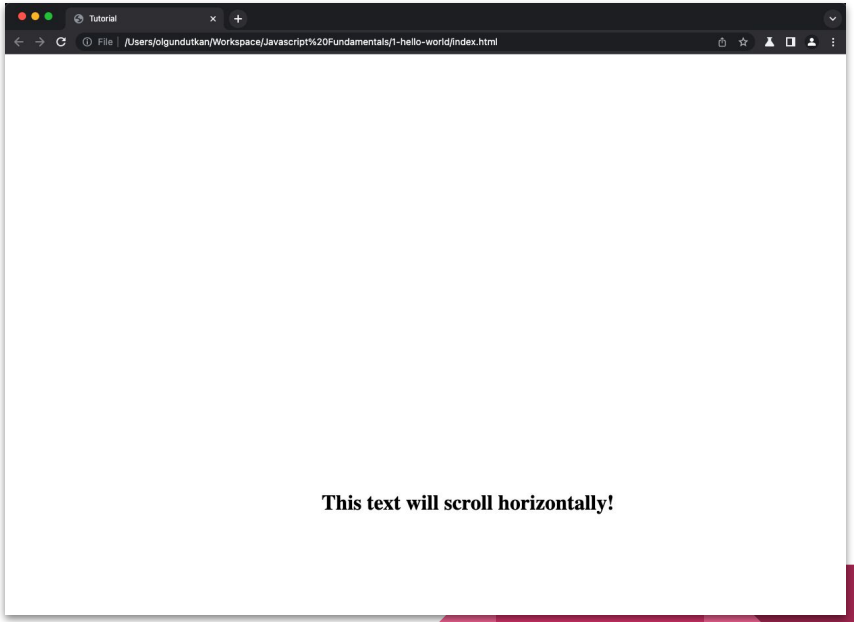
JavaScript is a programming language that adds interactivity and dynamism to web pages. It enables real-time updates of page content based on user interactions, provides data validation and form control, allows data manipulation and management, facilitates integration with various APIs, and enables the development of modern web applications. JavaScript is a fundamental tool in web development, enhancing web pages with interactivity and making them more engaging for users.

# 1.4. A "Hello world!" example

index.html × main.js style.css

1-introduction > 1-4-hello-world-example > index.html > ...

```html
<!DOCTYPE html>
<html>
  <head>
    <title>Tutorial</title>
    <link rel="stylesheet" href="./style.css" />
  </head>
  <body>
    <h1 id="scrolling-text">This text will scroll horizontally!</h1>
    <!-- Include the JavaScript file
      that contains the logic for scrolling the text horizontally -->
    <script src="./main.js"></script>
  </body>
</html>
```

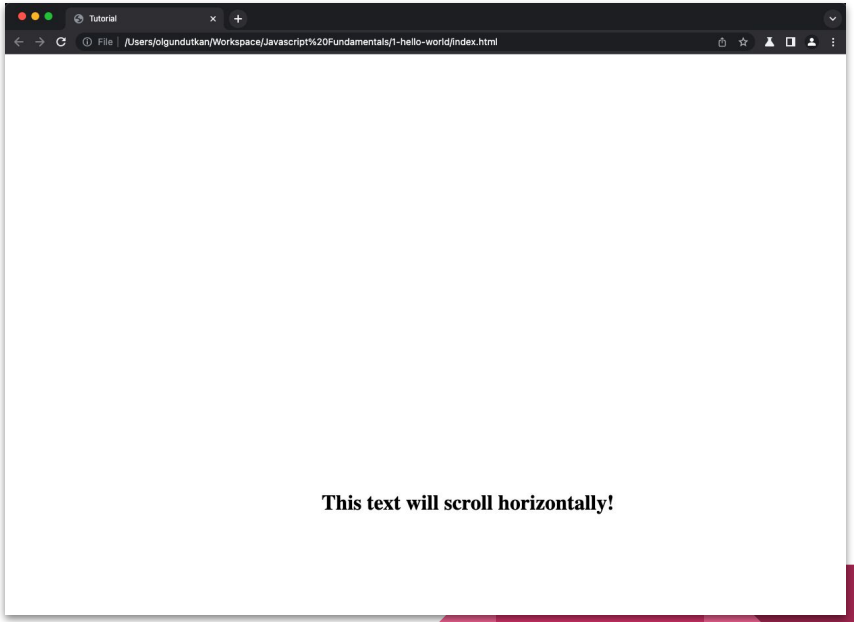


# 1.4. A "Hello world!" example

index.html | main.js | style.css

1-introduction > 1-4-hello-world-example > main.js > ...

```javascript
"use strict";

// Get the <h1> element with the ID "scrolling-text"
var h1 = document.getElementById("scrolling-text");

// Initialize the variable "left" to 0
// and the boolean "moveRight" to true
var left = 0;
var moveRight = true;

// Define a function called "moveText"
function moveText() {

  // Check if the text has reached the right edge of the screen
  if (left === window.innerWidth - h1.clientWidth) {
    moveRight = false;
  }
  // Check if the text has reached the left edge of the screen
  else if (left === 0) {
    moveRight = true;
  }

  // Move the text one pixel to the right or left depending
  // on the value of "moveRight"
  if (moveRight) {
    left++;
  } else {
    left--;
  }

  // Set the position of the <h1> element
  // to the current value of "left"
  h1.style.left = left + "px";
}

// Call the "moveText" function every 10 milliseconds
setInterval(moveText, 10);
```

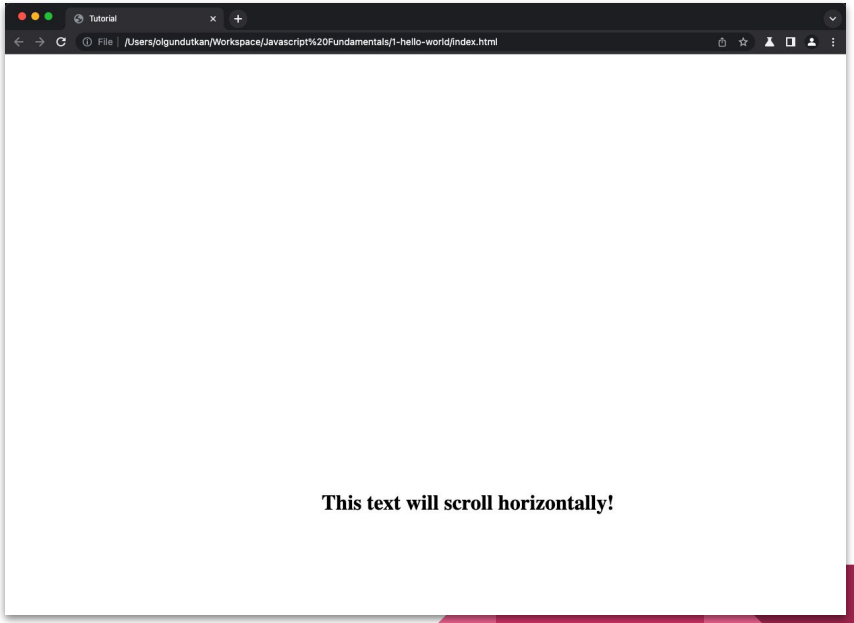


# 1.4. A "Hello world!" example

index.html | main.js | style.css

1-introduction > 1-4-hello-world-example > style.css > ...

```
#scrolling-text {
  position: absolute;
  top: 50%;
  left: 0;
  white-space: nowrap;
}

```



# 2. Basic Concepts

1.1. Variables and Data Types
1.2. Usage of Operators
1.3. Usage of Conditional
1.4. Usage of Loops

# 2.2. Variables and Data Types

1-primitive.js

2-basic-concepts > 2-2-variables-and-data-types > 1-primitive.js > ...

```javascript
// Primitive Data Types

var age = 34; // number

var name = "Olgun"; // string

var isValid = true; // boolean

var a; // undefined

var b = null; // null
```

2-reference.js

2-basic-concepts > 2-2-variables-and-data-types > 2-reference.js > ...

```javascript
// Reference Data Types

// object
var person = {
  name: "Olgun",
  age: 34,
  address: {
    city: "Ankara",
  },
};

// array
var numbers = [1, 2, 3, 4, 5];

// function
function sumOfTwoNumbers(num1, num2) {
  return num1 + num2;
}
```

# 2.3. Usage of Operators

1-arithmetic.js

2-basic-concepts > 2-3-usage-of-operators > 1-arithmetic.js > ...

```js
// Arithmetic Operators

var a = 10;
var b = 5;

// addition
var sum = a + b; // sum is 15

// subtraction
var difference = a - b; // difference is 5

// multiplication
var product = a * b; // product is 50

// division
var quotient = a / b; // quotient is 2
```

2-comparison.js

2-basic-concepts > 2-3-usage-of-operators > 2-comparison.js > ...

```js
// Comparison Operators

var a = 10;
var b = 5;

// equal to
console.log(a == b); // false

// not equal to
console.log(a != b); // true

// greater than
console.log(a > b); // true

// less than
console.log(a < b); // false

// greater than or equal to
console.log(a >= b); // true

// less than or equal to
console.log(a <= b); // false
```

# 2.3. Usage of Operators

3-equal-and-sameness.js

2-basic-concepts > 2-3-usage-of-operators > 3-equal-and-sameness.js > ...

```javascript
// Sameness

var a = 10;
var b = "10";

// equal to
console.log(a == b); // true

// sameness to
console.log(a === b); // false

// type check
console.log(typeof a); // number
console.log(typeof b); // string
```

# 2.3. Usage of Operators

4-logical.js

2-basic-concepts > 2-3-usage-of-operators > 4-logical.js > ...

```javascript
// Logical Operators

var age = 34;
var hasLicense = true;

// AND
if (age >= 18 && hasLicense) {
  console.log("You are eligible to drive.");
} else {
  console.log("You are not eligible to drive.");
}

var hasDegree = false;
var hasExperience = true;

// OR
if (hasDegree || hasExperience) {
  console.log("You are eligible for this job.");
} else {
  console.log("You are not eligible for this job.");
}

var isValid = true;

// NOT
if (!isValid) {
  console.log("This form is invalid.");
} else {
  console.log("This form is valid.");
}
```

5-assignment.js

2-basic-concepts > 2-3-usage-of-operators > 5-assignment.js > ...

```javascript
// Assignment Operators

var a = 10;
var b = 5;

// Basic assignment operator
a = b;
console.log(a); // Output: 5

// Addition assignment operator
a += b;
console.log(a); // Output: 10

// Subtraction assignment operator
a -= b;
console.log(a); // Output: 5

// Multiplication assignment operator
a *= b;
console.log(a); // Output: 25

// Division assignment operator
a /= b;
console.log(a); // Output: 5
```

# 2.4. Usage of Conditional

1-if.js

2-basic-concepts > 2-4-usage-of-conditional > 1-if.js > ...

```javascript
// if statement

var num = 10;

if (num > 0) {
  console.log("The number is positive.");
} else {
  console.log("The number is negative.");
}
```

2-switch.js

2-basic-concepts > 2-4-usage-of-conditional > 2-switch.js > ...

```javascript
// Switch statement

var day = "Monday";

switch (day) {
  case "Monday":
    console.log("Today is Monday.");
    break;
  case "Tuesday":
    console.log("Today is Tuesday.");
    break;
  case "Wednesday":
    console.log("Today is Wednesday.");
    break;
  case "Thursday":
    console.log("Today is Thursday.");
    break;
  case "Friday":
    console.log("Today is Friday.");
    break;
  case "Saturday":
    console.log("Today is Saturday.");
    break;
  case "Sunday":
    console.log("Today is Sunday.");
    break;
  default:
    console.log("Invalid day.");
    break;
}
```

# 2.5. Usage of Loops

1-for.js

2-basic-concepts > 2-5-usage-of-loops > 1-for.js > ...

```javascript
// For loop

for (var i = 0; i < 5; i++) {
  console.log(i);
}
```

2-while.js

2-basic-concepts > 2-5-usage-of-loops > 2-while.js > ...

```javascript
// While loop

var i = 0;
while (i < 5) {
  console.log(i);
  i++;
}
```

3-do-while.js

2-basic-concepts > 2-5-usage-of-loops > 3-do-while.js > ...

```javascript
// Do While loop

var i = 0;
do {
  console.log(i);
  i++;
} while (i < 5);
```

# 3. Arrays

1.1. Frequently Used Array Methods

# 1.1. Frequently Used Array Methods

- **push()**: adds one or more elements to the end of an array.
- **pop()**: removes and returns the last element of an array.
- **shift()**: removes and returns the first element of an array.
- **unshift()**: adds one or more elements to the beginning of an array.
- **splice()**: adds or removes elements from an array at a specified index.
- **slice()**: returns a new array containing a portion of the original array.

3-1-frequently-used-array-methods-1.js

3-arrays > 3-1-frequently-used-array-methods-1.js > ...

```js
// Frequently Used Array Methods 1

var numbers = [1, 2, 3, 4, 5];

numbers.push(6);
// numbers is now [1, 2, 3, 4, 5, 6]

numbers.pop();
// numbers is now [1, 2, 3, 4, 5]

numbers.unshift(0);
// numbers is now [0, 1, 2, 3, 4, 5]

numbers.splice(2, 1);
// numbers is now [0, 1, 3, 4, 5]

var slicedNumbers = numbers.slice(2, 4);
// slicedNumbers is [3, 4]
```

# 1.1. Frequently Used Array Methods

- **concat()**: merges two or more arrays into a new array.
- **indexOf()**: returns the index of the first occurrence of a specified element in an array.
- **forEach()**: executes a provided function once for each array element.
- **map()**: creates a new array by calling a provided function on each array element.
- **filter()**: creates a new array with all elements that pass a test implemented by a provided function.

3-1-frequently-used-array-methods-2.js

3-arrays > 3-1-frequently-used-array-methods-2.js > ...

```javascript
// Frequently Used Array Methods 2

var numbers = [0, 1, 3, 4, 5];

var fruits = ["apple", "banana", "orange"];

var combined = fruits.concat(numbers);
// combined is now ['apple', 'banana', 'orange', 0, 1, 3, 4, 5]

var orangeIndex = fruits.indexOf("orange");
// orangeIndex is 2

numbers.forEach((number) => console.log(number));
// logs each number in the numbers array

var doubledNumbers = numbers.map((number) => number * 2);
// doubledNumbers is [0, 2, 6, 8, 10]

var evenNumbers = numbers.filter((number) => number % 2 === 0);
// evenNumbers is [0, 4]

```

# 4. DOM Manipulation

1.1. Relationship Between HTML and JavaScript
1.2. Access to DOM Elements
1.3. Usage of Event Listeners

# 4.1. Relationship Between HTML and JavaScript

index.html

5-dom-manipulation > 5-1-relationship-between-html-and-javascript > index.html > ...

```html
<!DOCTYPE html>
<html>
  <head>
    <title>Increment/Decrement Example</title>
  </head>
  <body>
    <p id="counter">0</p>
    <button onclick="increment()">Increment</button>
    <button onclick="decrement()">Decrement</button>

    <script src="./app.js"></script>
  </body>
</html>
```

0

Increment Decrement

app.js

5-dom-manipulation > 5-1-relationship-between-html-and-javascript > app.js > ...

```javascript
"use-script";

// Define a variable called "number" and set its initial value to 0.
var number = 0;

// Get the HTML element with the ID "number"
// and store it in a variable called "counterElement".
var counterElement = document.getElementById("counter");

// Define a function called "increment" that will increase
// the value of the "number" variable by 1 and update
// the content of the "counterElement" with the new value.
function increment() {
  number++;
  counterElement.innerHTML = number;
}

// Define a function called "decrement" that will decrease
// the value of the "number" variable by 1 and update
// the content of the "counterElement" with the new value.
function decrement() {
  number--;
  counterElement.innerHTML = number;
}
```

# 4.2. Access to DOM Elements

- **getElementById(id):** This method returns the HTML element with the specified id attribute.
- **getElementsByClassName(className)**: This method returns a collection of HTML elements with the specified class name.
- **getElementsByTagName(tagName)**: This method returns a collection of HTML elements with the specified tag name.
- **querySelector(selector)**: This method returns the first HTML element that matches the specified CSS selector.
- **querySelectorAll(selector)**: This method returns a collection of HTML elements that match the specified CSS selector.

app.js

5-dom-manipulation > 5-2-access-to-dom-elements > app.js > ...

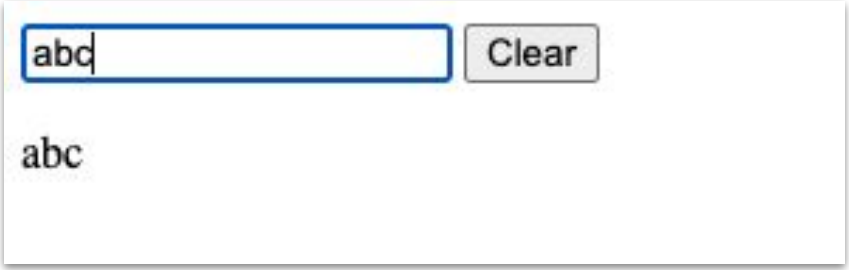
```js
"use script";

// Get element with the ID "title"
var elementById = document.getElementById("title");

// Get all elements with the class name "fruit"
var elementsByClassName = document.getElementsByClassName("fruit");

// Get all paragraph elements
var elementsByTagName = document.getElementsByTagName("p");

// Get the first element with the class name "fruit"
var firstFruitElementByClassName = document.querySelector(".fruit");

// Get all elements with the class name "fruit"
var fruitElementsByClassName = document.querySelectorAll(".fruit");
```

# 4.3. Usage of Event Listeners

index.html

5-dom-manipulation > 5-4-usage-of-event-listeners > index.html > ...

```html
<!DOCTYPE html>
<html>
  <head>
    <title>Event Listener Example</title>
  </head>
  <body>
    <!-- Input element to type text -->
    <input type="text" id="myInput" />
    <!-- Button element to clear the input field and output text -->
    <button id="my-button">Clear</button>
    <!-- Paragraph element to display the typed text -->
    <p id="output"></p>

    <script src="app.js"></script>
  </body>
</html>
```




app.js

5-dom-manipulation > 5-4-usage-of-event-listeners > app.js > ...

```javascript
"use script";

// Initialize a variable to store the new character typed in the input field
var newChar = "";

// Get the input element and assign it to the "input" variable
var input = document.getElementById("myInput");

// Get the output element and assign it to the "output" variable
var output = document.getElementById("output");

// Add an event listener to the input element that triggers when its value changes
input.addEventListener("input", function (event) {
  // Access the new value of the input field using the "event" object's "data" property,
  // which contains the character that was just typed
  newChar = event.data;

  // Append the new character to the end of the output element's current text
  output.innerText += newChar;
});

// Get the button element and assign it to the "myButton" variable
const myButton = document.querySelector("#my-button");

// Add an event listener to the button element that triggers when it's clicked
myButton.addEventListener("click", function () {
  // Reset the "newChar" variable to an empty string
  newChar = "";

  // Clear the input field by setting its value to an empty string
  input.value = newChar;

  // Clear the output text by setting its value to an empty string
  output.innerText = newChar;
});
```

# 6. Asynchronous Programming

# 7. ECMAScript

Closure a bir örnek

Variable referance a bir örnek

# 8. Conclusion

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

https://en.wikipedia.org/wiki/JavaScript