JavaScript DataView

Back to JS page

Table of Contents

- JavaScript DataView
  - Creating a DataView
    - Example 1
  - Reading and Writing Values
    - Example 2
  - Endianness (Byte Order)
    - Example 3
  - Mixed Types
    - Example 4
  - Encoding and Decoding a Record
    - Example 5
  - Document
  - Reference

What Is a DataView?

A `DataView` provides a low-level interface for reading and writing multiple number types in an `ArrayBuffer`.

Unlike typed arrays (like `Int32Array`), a `DataView`:

- Can read/write different data types in the same buffer
- Can start at any byte offset (not aligned to element size)
- Supports both big-endian and little-endian byte order

Creating a DataView

Create a `DataView` from an existing `ArrayBuffer`:

```
<!DOCTYPE html>
<html>
<body>

<h2>JavaScript DataView</h2>
<p>From w3schools.com, Experiment by Teeratus_R</p>

<h4>Creating a DataView</h4>
<p id="demo"></p>

<script>
const buffer = new ArrayBuffer(16);
const dataView = new DataView(buffer);

let text = "DataView created from ArrayBuffer(16)<br>";
text += "buffer: " + dataView.buffer.byteLength + " bytes<br>";
text += "byteOffset: " + dataView.byteOffset + "<br>";
text += "byteLength: " + dataView.byteLength;

document.getElementById("demo").innerHTML = text;
</script>

</body>
</html>
```

Example 1

Result View Example

Reading and Writing Values

DataView provides methods like `setInt32()`, `getFloat32()`, etc. to read/write values at specific byte offsets:

```
<!DOCTYPE html>
<html>
<body>

<h2>JavaScript DataView</h2>
<p>From w3schools.com, Experiment by Teeratus_R</p>

<h4>Reading and Writing Values</h4>
<p id="demo"></p>

<script>
const buffer = new ArrayBuffer(16);
const dv = new DataView(buffer);

// Write values at different offsets
dv.setInt32(0, 42);        // 4 bytes at offset 0
dv.setFloat64(4, 3.14159); // 8 bytes at offset 4
dv.setInt16(12, 255);      // 2 bytes at offset 12

// Read values back
let text = "Values written to DataView:<br><br>";
text += "getInt32(0): " + dv.getInt32(0) + " (4 bytes)<br>";
text += "getFloat64(4): " + dv.getFloat64(4) + " (8 bytes)<br>";
text += "getInt16(12): " + dv.getInt16(12) + " (2 bytes)<br><br>";

// Total bytes used: 4 + 8 + 2 = 14 bytes
text += "Total bytes used: 14 of " + buffer.byteLength;

document.getElementById("demo").innerHTML = text;
</script>

</body>
</html>
```

Example 2

Result View Example

Endianness (Byte Order)

Endianness refers to the byte order of multi-byte values. The `DataView` methods accept an optional `littleEndian` parameter:

- **Big-endian**: Most significant byte first (default)
- **Little-endian**: Least significant byte first

```
<!DOCTYPE html>
<html>
<body>

<h2>JavaScript DataView</h2>
<p>From w3schools.com, Experiment by Teeratus_R</p>

<h4>Endianness (Byte Order)</h4>
<p id="demo"></p>

<script>
const buffer = new ArrayBuffer(4);
const dv = new DataView(buffer);

// Write the same value with different endianness
dv.setInt32(0, 0x12345678, false);  // Big-endian (default)
const bigEndian = dv.getInt32(0, false);

dv.setInt32(0, 0x12345678, true);   // Little-endian
const littleEndian = dv.getInt32(0, true);

// Inspect individual bytes
const bytes = new Uint8Array(buffer);

let text = "Value: 0x12345678 (305419896)<br><br>";

text += "Big-endian bytes: ";
for (let b of bytes) text += "0x" + b.toString(16).toUpperCase() + " ";
text += "<br>";

dv.setInt32(0, 0x12345678, true);
text += "Little-endian bytes: ";
for (let b of bytes) text += "0x" + b.toString(16).toUpperCase() + " ";

document.getElementById("demo").innerHTML = text;
</script>

</body>
</html>
```

Example 3

Result View Example

Mixed Types

DataView allows mixing different data types in a single buffer:

```
<!DOCTYPE html>
<html>
<body>

<h2>JavaScript DataView</h2>
<p>From w3schools.com, Experiment by Teeratus_R</p>

<h4>Mixed Types</h4>
<p id="demo"></p>

<script>
const buffer = new ArrayBuffer(20);
const dv = new DataView(buffer);

// Write mixed data types
dv.setInt32(0, 2024);        // Year (4 bytes)
dv.setInt16(4, 12);          // Month (2 bytes)
dv.setInt16(6, 25);          // Day (2 bytes)
dv.setFloat32(8, 36.5);      // Temperature (4 bytes)
dv.setInt8(12, 72);          // Humidity % (1 byte)
dv.setInt8(13, 1);           // Active flag (1 byte)

// Read back
let text = "Mixed Data Types in DataView:<br><br>";
text += "Year: " + dv.getInt32(0) + "<br>";
text += "Month: " + dv.getInt16(4) + "<br>";
text += "Day: " + dv.getInt16(6) + "<br>";
text += "Temperature: " + dv.getFloat32(8) + "°C<br>";
text += "Humidity: " + dv.getInt8(12) + "%<br>";
text += "Active: " + (dv.getInt8(13) === 1);

document.getElementById("demo").innerHTML = text;
</script>

</body>
</html>
```

Example 4

Result View Example

Encoding and Decoding a Record

DataView is ideal for binary data formats like file headers or network packets:

```
<!DOCTYPE html>
<html>
<body>

<h2>JavaScript DataView</h2>
<p>From w3schools.com, Experiment by Teeratus_R</p>

<h4>Encoding and Decoding a Record</h4>
<p id="demo"></p>

<script>
// Define a record structure
function encodeRecord(id, score, name) {
  const buf = new ArrayBuffer(16);
  const dv = new DataView(buf);
  dv.setInt32(0, id);               // ID (4 bytes)
  dv.setFloat64(4, score);           // Score (8 bytes)
  for (let i = 0; i < name.length; i++) {
    dv.setUint8(12 + i, name.charCodeAt(i)); // Name chars
  }
  dv.setUint8(15, 0);               // Null terminator
  return buf;
}

function decodeRecord(buf) {
  const dv = new DataView(buf);
  const id = dv.getInt32(0);
  const score = dv.getFloat64(4);
  let name = "";
  for (let i = 12; i < 16; i++) {
    const ch = dv.getUint8(i);
    if (ch === 0) break;
    name += String.fromCharCode(ch);
  }
  return { id, score, name };
}

// Encode a record
const buffer = encodeRecord(101, 95.5, "Bob");
const decoded = decodeRecord(buffer);

let text = "Binary Record (16 bytes):<br><br>";
text += "Encoded:<br>";
const view = new Uint8Array(buffer);
for (let b of view) text += b.toString(16).padStart(2, "0") + " ";
text += "<br><br>";

text += "Decoded:<br>";
text += "ID: " + decoded.id + "<br>";
text += "Score: " + decoded.score + "<br>";
```

```
text += "Name: " + decoded.name;

document.getElementById("demo").innerHTML = text;
</script>

</body>
</html>
```

Example 5

Result View Example

Document

Document in project

You can Download PDF file.

Reference

- W3Schools JavaScript DataView