



JavaScript and JSON

By Prof. Vaibhavi Patel



Outline



- JavaScript:
 - JS Data types and variable
 - Arrays
 - Operators
 - Conditionals
 - Functions
 - Objects
 - JS Events, and JS Forms
- JSON: Introduction to JSON, JSON Syntax, JSON data types, JSON Objects, JSON Arrays, JSON Parse, JSON HTML.



Introduction

- JavaScript is one of the **3 languages** all web developers **must** learn:
- 1. **HTML** to define the content of web pages
- 2. **CSS** to specify the layout of web pages
- 3. **JavaScript** to program the behavior of web pages



Introduction

- JavaScript is a client side scripting language.
- It is a lightweight, dynamic computer programming language, used to make web pages alive.
- It is used to programmatically perform actions within the page.
- When JavaScript was created, it was initially called “LiveScript”.
- But Java was a very popular language at that time, so it was decided that positioning a language as a “younger brother” of Java would help.
- JavaScript can execute not only in the browser, but also on the server.
- JavaScript has evolved greatly as a language and is now used to perform a wide variety of tasks.



Advantages of JavaScript

The merits of using JavaScript are –

- **Less server interaction** – You can validate user input before sending the page off to the server. This saves server traffic, which means less load on your server.
- **Immediate feedback to the visitors** – They don't have to wait for a page reload to see if they have forgotten to enter something.
- **Increased interactivity** – You can create interfaces that react when the user hovers over them with a mouse or activates them via the keyboard.
- **Richer interfaces** – You can use JavaScript to include such items as drag-and-drop components and sliders to give a Rich Interface to your site visitors.



What Can In-Browser JavaScript Do?

- If JavaScripts are used in any websites, then it should **not be given any low level CPU permissions** like switching off the CPU etc. That is why JavaScript is made with extremely safe permissions that does not have any permission to access low level CPU usage.
- JavaScript can **add** new HTML and **change** existing HTML from DOM.
- It can even **react** to any events (actions).
- It can also manage the **AJAX** requests (GET or POST request)
- JavaScript can **get and set** cookies and use local storage.



Introduction

- Dynamic HTML (DHTML)
- Makes possible a Web page to react and change in response to the user's actions

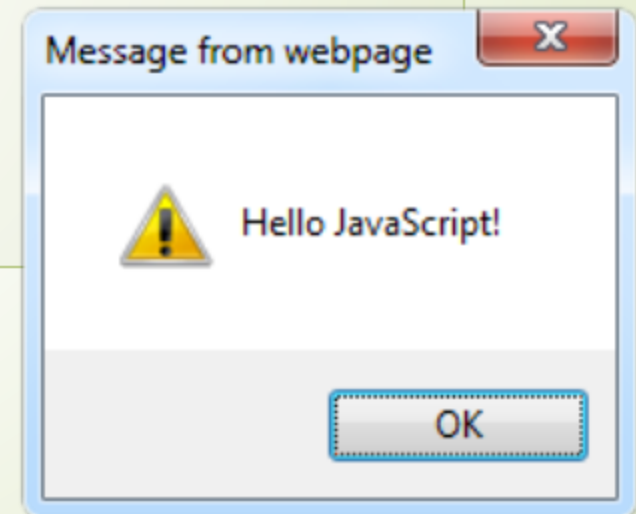
$\text{DHTML} = \text{HTML} + \text{CSS} + \text{JavaScript}$

first-script.html

```
<html>

<body>
  <script type="text/javascript">
    alert('Hello JavaScript!');
  </script>
</body>

</html>
```



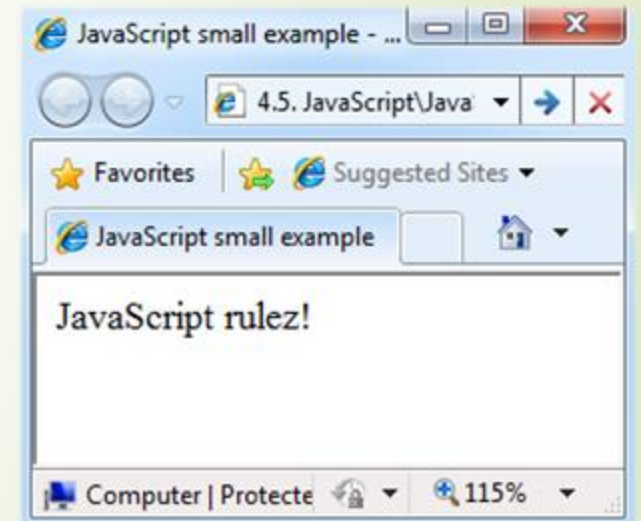
The HTML <script> Tag

- The JavaScript code can be placed in:
 - <script> tag in the head
 - <script> tag in the body – not recommended
- External files, linked via <script> tag the head. Files usually have .js extension

<script src="scripts.js" type="text/javascript"></script>

small-example.html

```
<html>  
<body>  
  <script type="text/javascript">  
    document.write('JavaScript rulez!');  
  </script>  
</body>  
</html>
```



<noscript> Tag

- The HTML The <noscript> tag is used to provide an alternate content for users that have disabled scripts in their browser or have a browser that doesn't support client-side scripts:

- `<script>`
.....
`</script>`
`<noscript>`

Sorry, your browser does not support JavaScript!

`</noscript>`



Identifier

- Same as Java/C++ except that it allows an additional character – '\$'.
- Contains only 'A' – 'Z', 'a' – 'z', '0' – '9', '_', '\$'
- First character cannot be a digit
- Case-sensitive
- Cannot be reserved words or keywords



JS data types

- ▶ Variables are typeless
- ▶ Numbers (integer, floating-point)
- ▶ Boolean (true / false)
- ▶ String type – string of characters

```
var myName = "You can use both single or double  
quotes for strings";
```

- ▶ Arrays

```
var my_array = [1, 5.3, "aaa"];
```

- ▶ Associative arrays (hash tables)

```
var my_hash = {a:2, b:3, c:"text"};
```



Array

- An array is represented by the **Array** object. To create an array of N elements, you can write

```
var myArray = new Array(N);
```

- Index of array runs from 0 to N-1.
- Can store values of different types
- Property "**length**" tells the # of elements in the array.
- Consists of various methods to manipulate its elements. e.g., **reverse()**, **push()**, **concat()**, **pop()**

Arrays Operations and Properties

- Declaring new empty array:

```
var arr = new Array();
```

- Declaring an array holding few elements:

```
var arr = [1, 2, 3, 4, 5];
```

- Appending an element / getting the last element:

```
arr.push(3);  
var element = arr.pop();
```

- Reading the number of elements (array length):

```
arr.length;
```

- Finding element's index in the array:

```
arr.indexOf(1);
```




Strings

- A string variable can store a sequence of alphanumeric characters, spaces and special characters.
- A string can be enclosed by a pair of single quotes (') or double quote (").
- Use escaped character sequence to represent special character (e.g.: \")



String Operations

- The + operator joins strings

```
string1 = "fat ";  
string2 = "cats";  
alert(string1 + string2); // fat cats
```

- What is "9" + 9?

```
alert("9" + 9); // 99
```

- Converting string to number:

```
alert(parseInt("9") + 9); // 18
```



Everything is Object

- Every variable can be considered as object

```
var test = "some string";  
alert(test[7]); // shows letter 'r'  
alert(test.charAt(5)); // shows letter 's'  
alert("test".charAt(1)); //shows letter 'e'  
alert("test".substring(1,3)); //shows 'es'
```

```
var arr = [1,3,4];  
alert (arr.length); // shows 3  
arr.push(7); // appends 7 to end of array  
alert (arr[3]); // shows 7
```

typeof operator

```
var x = "hello", y;  
alert("Variable x value is " + typeof x );  
alert("Variable y value is " + typeof y );  
alert("Variable z value is " + typeof z );
```

- An unary operator that tells the type of its operand.
 - Returns a string which can be "number", "string", "boolean", "object", "function", "undefined"..
- An array is internally represented as an object.



Standard Popup Boxes

- ▶ Alert box with text and [OK] button
 - ▶ Just a message shown in a dialog box:

```
alert("Some text here");
```

- ▶ Confirmation box
 - ▶ Contains text, [OK] button and [Cancel] button:

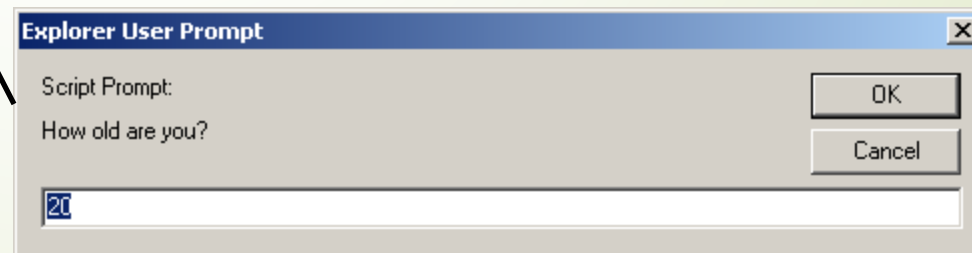
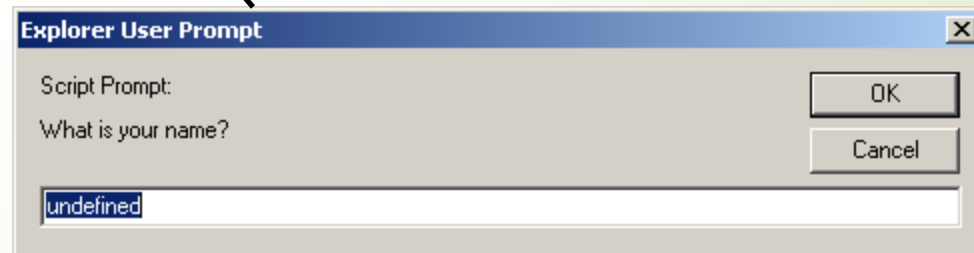
```
confirm("Are you sure?");
```

- ▶ Prompt box
 - ▶ Contains text, input field with default value:

```
prompt ("enter amount", 10);
```

alert(), confirm(), and prompt()

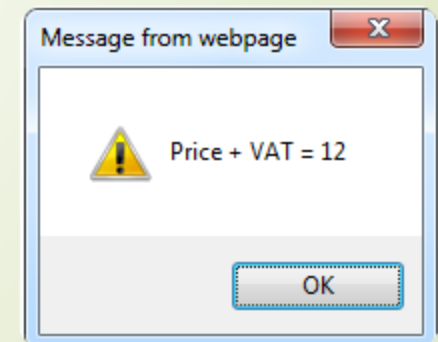
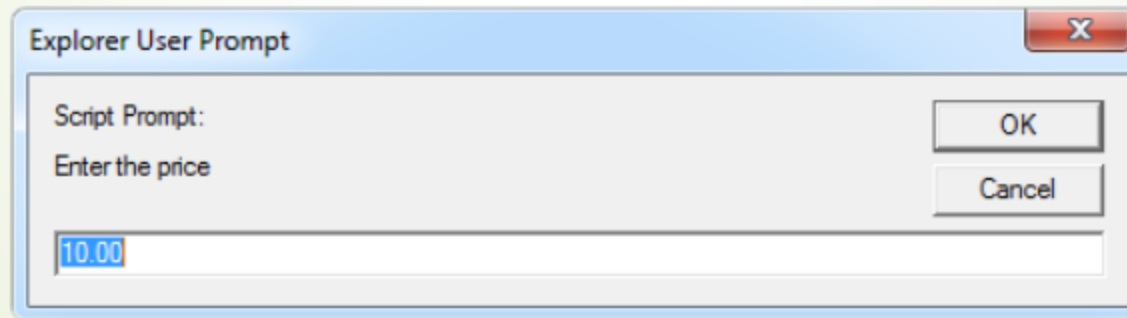
```
<script type="text/javascript">  
alert("This is an Alert method");  
confirm("Are you OK?");  
prompt("What is your name?");  
prompt("How old are you?", "20");  
</script>
```



JavaScript Prompt – Example

➤ prompt.html

```
price = prompt("Enter the price", "10.00");  
alert('Price + VAT = ' + price * 1.2);
```





Operators

1) Arithmetic operators

- $+$, $-$, $*$, $/$, $\%$

- Post/pre increment/decrement

- $++$, $--$



Operators

2) Comparison operators

Symbol	Meaning
>	Greater than
<	Less than
>=	Greater than or equal to
<=	Less than or equal to
==	Equal
!=	Not equal

- ==, != (Strictly equals and strictly not equals)
 - i.e., Type and value of operand must match / must not match



== VS ===

```
// Type conversion is performed before comparison
var v1 = ("5" == 5);    // true

// No implicit type conversion.
// True if only if both types and values are equal
var v2 = ("5" === 5);   // false

var v3 = (5 === 5.0);   // true

var v4 = (true == 1);   // true (true is converted to 1)

var v5 = (true == 2);   // false (true is converted to 1)

var v6 = (true == "1"); // true
```



3) Logical Operators

! – Logical NOT

&& – Logical AND

|| – Logical OR

4) Assignment operators

=, +=, -=, *=, /=, %=



5) Bitwise operators

Operator	Name	Description
&	AND	Sets each bit to 1 if both bits are 1
	OR	Sets each bit to 1 if one of two bits is 1
^	XOR	Sets each bit to 1 if only one of two bits is 1
~	NOT	Inverts all the bits
<<	Zero fill left shift	Shifts left by pushing zeros in from the right and let the leftmost bits fall off
>>	Signed right shift	Shifts right by pushing copies of the leftmost bit in from the left, and let the rightmost bits fall off
>>>	Zero fill right shift	Shifts right by pushing zeros in from the left, and let the rightmost bits fall off



Conditional Statements

If you want to perform different actions for different decisions, you can use conditional statements in your code to do this.

- In JavaScript we have the following conditional statements:
- Use **if** to specify a block of code to be executed, if a specified condition is true
- Use **else** to specify a block of code to be executed, if the same condition is false
- Use **else if** to specify a new condition to test, if the first condition is false
- Use **switch** to specify many alternative blocks of code to be executed

Conditional Statements

➤ Syntax

```
if (condition) {  
    // block of code to be executed if  
    the condition is true  
}
```

```
if (condition) {  
    // block of code to be executed if the  
    condition is true  
} else {  
    // block of code to be executed if the  
    condition is false  
}
```

```
if (condition1) {  
    // block of code to be executed if  
    condition1 is true  
} else if (condition2) {  
    // block of code to be executed if the  
    condition1 is false and condition2 is true  
} else {  
    // block of code to be executed if the  
    condition1 is false and condition2 is false  
}
```

Conditional Statements

- Use the switch statement to select one of many code blocks to be executed.

Syntax

```
switch(expression) {  
    case x:  
        // code block  
        break;  
    case y:  
        // code block  
        break;  
    default:  
        // code block  
}
```



Looping Statement

if you want to run the same code over and over again, each time with a different value.

- “for” Loops
- “while” Loops
- “do ... while” Loops
- “break” statement
- “continue” statement



for-Looping Statement

```
for (statement 1; statement 2; statement 3)  
{  
    // code block to be executed  
}
```

Statement 1 is executed (one time) before the execution of the code block.

Statement 2 defines the condition for executing the code block.

Statement 3 is executed (every time) after the code block has been executed.



while-Looping Statement

The while loop loops through a block of code as long as a specified condition is true.

➤ Syntax

➤ `while (condition) {`
 // code block to be executed
}



do/while-Looping Statement

- do/while loop is a variant of the while loop.
- This loop will execute the code block once, before checking if the condition is true, then it will repeat the loop as long as the condition is true.

Syntax

```
do {  
    // code block to be executed  
}  
while (condition);
```



Break and continue

- The break statement "jumps out" of a loop.
- The break statement ends the loop
- The continue statement "jumps over" one iteration in the loop.
- The continue statement breaks one iteration (in the loop), if a specified condition occurs, and continues with the next iteration in the loop.



Function in javascript

- A JavaScript function is a block of code designed to perform a particular task.
- A JavaScript function is executed when "something" invokes it (calls it).
- A JavaScript function is defined with the function keyword, followed by a name, followed by parentheses ().
- Function names can contain letters, digits, underscores, and dollar signs (same rules as variables).
- The parentheses may include parameter names separated by commas: (parameter1, parameter2, ...)
- The code to be executed, by the function, is placed inside curly brackets: { }



Functions



// A function can return value of any type using the
// keyword "return".

// The same function can possibly return values
// of different types

```
function foo (p1) {  
    if (typeof(p1) == "number")  
        return 0;    // Return a number  
    else  
        if (typeof(p1) == "string")  
            return "zero"; // Return a string
```

// If no value being explicitly returned
// "undefined" is returned.

```
}
```

```
foo(1);        // returns 0  
foo("abc");    // returns "zero"  
foo();         // returns undefined
```



Functions

```
// "arguments" is a local variable (an array) available  
// in every function  
// You can either access the arguments through parameters  
// or through the "arguments" array.
```

```
function sum ()  
{  
    var s = 0;  
    for (var i = 0; i < arguments.length; i++)  
        s += arguments[i];  
    return s;  
}
```

```
sum(1, 2, 3);           // returns 6  
sum(1, 2, 3, 4, 5);     // returns 15  
sum(1, 2, "3", 4, 5);   // returns ?
```



Object in JavaScript

- JavaScript variables are containers for data values.

```
var car="fiat";
```

- Objects are variables too. But objects can contain many values.

```
var car = {type: "Fiat", model: "500", color: "white"};
```

The values are written as **name:value** pairs (name and value separated by a colon) called properties.



Object in JavaScript

➤ Accessing Object Properties:

You can access object properties in two ways:

objectName.propertyName

or

objectName["propertyName"]



Object in JavaScript

- Objects can also have **methods**.
- Methods are **actions** that can be performed on objects.
- You access an object method with the following syntax:
objectName.methodName()



Object in JavaScript

- Creating objects using **new Object()**

```
var person = new Object();
```

```
// Assign fields to object "person"  
person.firstName = "John";  
person.lastName = "Doe";
```

HTML Events

- An event occurs as a result of some activity
- When JavaScript is used in HTML pages, JavaScript can "react" on these events

e.g.:

- A user clicks on a link in a page
- Page finished loaded
- Mouse cursor enter an area
- A preset amount of time elapses
- A form is being submitted

HTML Events



- Events are an important part of JavaScript programming, you can write scripts that run in response to the actions of the user, even after the Web page has been opened.

HTML Events

- Event Handler – a segment of codes (usually a function) to be executed when an event occurs
- We can specify event handlers as attributes in the HTML tags.
- The attribute names typically take the form "onXXX" where XXX is the event name.

e.g.:

- `<button onClick="alert('bye Bye')">say bye</button>`

HTML Events

- JavaScript code is often several lines long. It is more common to see event attributes calling functions.

```
<button onclick="displayDate()">The time  
is?</button>
```



Common HTML Events

Event	Description
onChange	The value of the text field, textarea, or a drop down list is modified
onClick	A link, an image or a form element is clicked once
onDbClick	The element is double-clicked
onMouseDown	The user presses the mouse button
onLoad	A document or an image is loaded
onSubmit	A user submits a form
onReset	The form is reset
onUnLoad	The user closes a document or a frame
onResize	A form is resized by the user

Using the Onclick Event Handler

When the user clicks a radio button, the click event is initiated and the onclick event handler instructs the browser to run a JavaScript command to change the background color of the Web page.

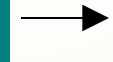
```
<p>Change background color to:</p>
<form>
<input type="radio" name="colors" onclick="document.bgColor='red'">Red <br>
<input type="radio" name="colors" onclick="document.bgColor='blue'">Blue<br>
<input type="radio" name="colors" onclick="document.bgColor='green'">Green
</form>
```

JavaScript commands

Change background color to:

☐ Red
☐ Blue
☐ Green

initial Web page



users clicks
the red
button

Change background color to:

☒ Red
☐ Blue
☐ Green

users clicks
the blue
button

Change background color to:

☐ Red
☒ Blue
☐ Green

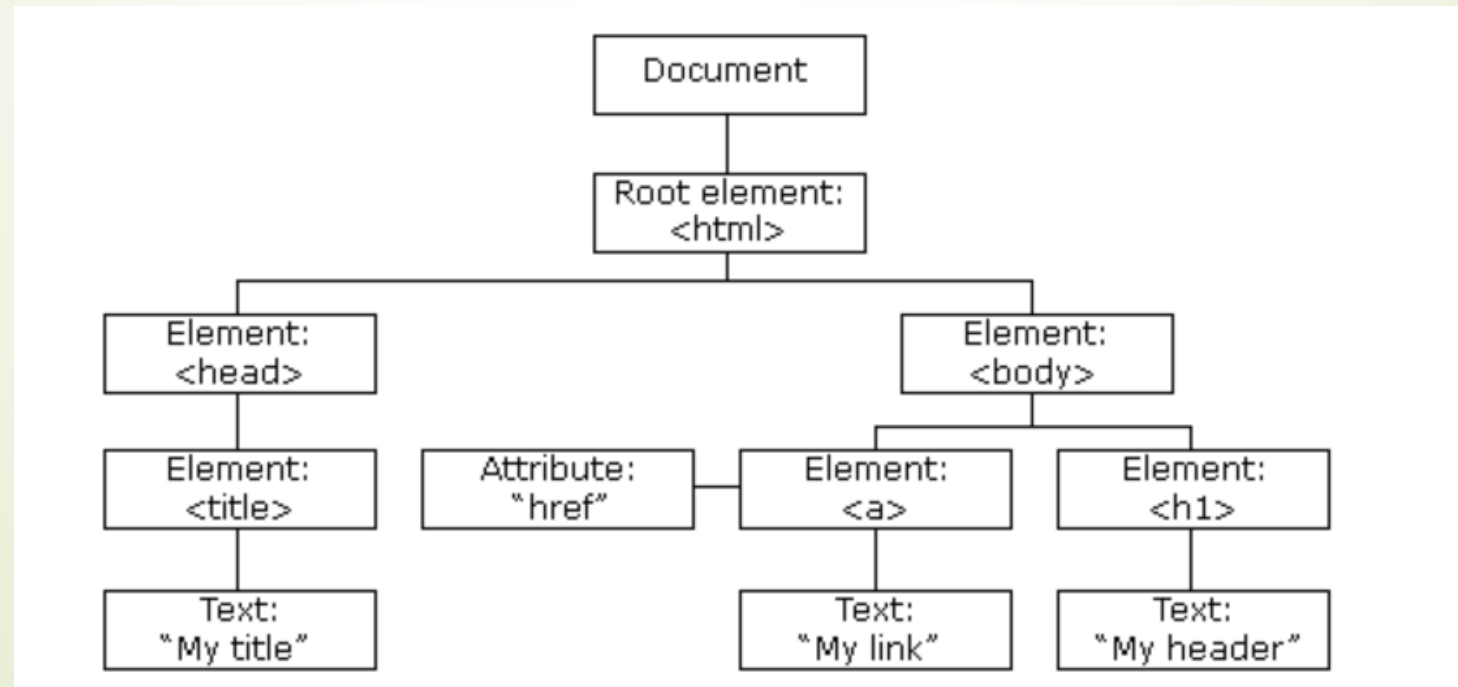
users clicks
the green
button

Change background color to:

☐ Red
☐ Blue
☒ Green

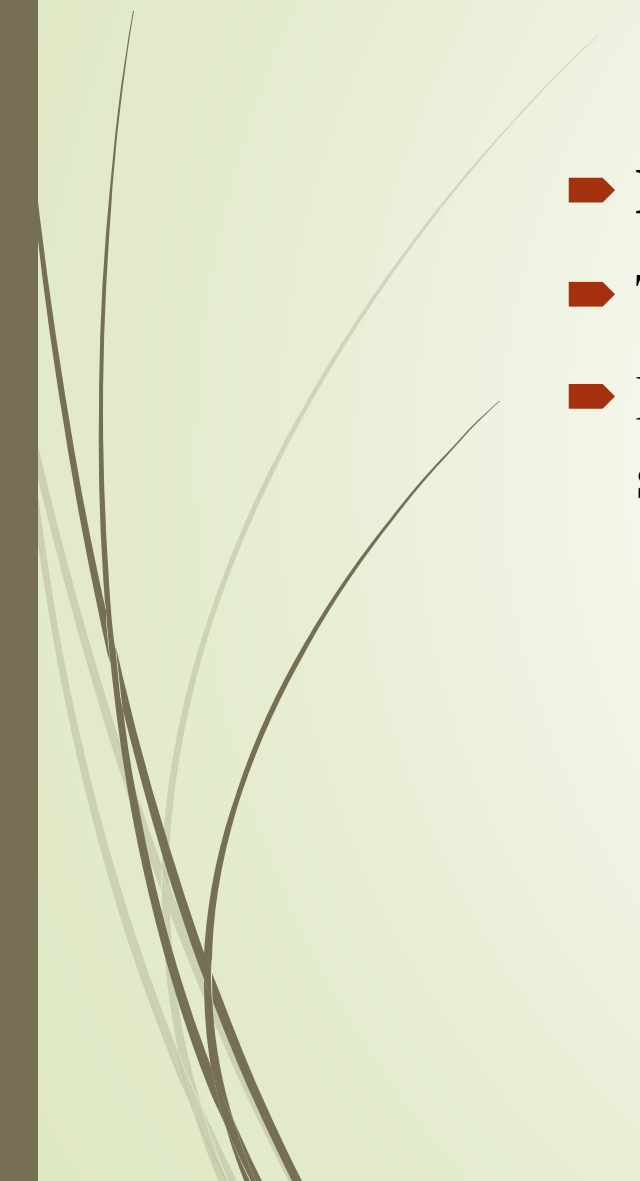
DOM (Document Object Model)

- When a web page is loaded, the browser creates a **Document Object Model** of the page.
- The **HTML DOM** model is constructed as a tree of **Objects**:





DOM (Document Object Model)

- **Document** object is the owner of all other objects in your web page.
 - The document object represents your web page
 - If you want to access any element in an HTML page, you always start with accessing the document object.
- 



Accessing Elements

- Access elements via their ID attribute

```
var elem = document.getElementById("some_id")
```

- Via the name attribute

```
var arr = document.getElementsByName("some_name")
```

- Via tag name

```
var imgTags = document.getElementsByTagName("img")
```



DOM (Document Object Model)

- With the object model, JavaScript gets all the power it needs to create dynamic HTML:

JavaScript can change all the HTML elements in the page

JavaScript can change all the HTML attributes in the page

JavaScript can change all the CSS styles in the page

JavaScript can remove existing HTML elements and attributes

JavaScript can add new HTML elements and attributes

etc...




Example

- Following JavaScript example writes "Hello JavaScript!" into an HTML element with id="demo":

[RUN\(change_content.html\)](#)


- Following JavaScript example display Date and Time when you click on button:

[RUN\(date_time.html\)](#)



JavaScript can change HTML attribute values.

➡ [RUN\(imoji.html\)](#)




JavaScript can change the style of an HTML element

➡ [RUN\(change_style.html\)](#)



JavaScript Can Hide HTML Elements

➡ [RUN\(hide_elements.html\)](#)



JavaScript can show hidden HTML elements.

➡ [RUN\(show_hidden.html\)](#)



form

- HTML form validation can be done by JavaScript.

[RUN\(formdemo.html\)](#)

A decorative graphic on the left side of the slide. It features a solid red arrow pointing to the right, positioned horizontally. Behind the arrow and extending upwards and to the right are several thin, dark, curved lines that sweep across the frame.

JSON



JSON



- JSON: JavaScript Object Notation.
- JSON is a syntax for storing and exchanging data.
- JSON is a lightweight data-interchange format.
- It is easy for humans to read and write.
- It is easy for machines to parse and generate.
- JSON is a text format that is completely language independent. These properties make JSON an ideal data-interchange language.
- The file type for JSON files is ".json"

JSON Syntax

- Data is in name/value pairs

A name/value pair consists of a field name (in double quotes), followed by a colon, followed by a value

- Data is separated by commas
- Curly braces hold objects
- Square brackets hold arrays

```
{ "name": "John" }
```




➤ **In JSON**, string values must be written with double quotes:

```
{ "name": "John" }
```

In JavaScript, you can write string values with double or single quotes:

```
{ name: 'John' }
```



➤ **In JSON**, *values* must be one of the following data types:

- a string
- a number
- an object (JSON object)
- an array
- a boolean
- null

➤ **In JavaScript** values can be all of the above, plus any other valid JavaScript expression, including:

- a function
- undefined



➤ JSON Numbers

- Numbers in JSON must be an integer or a floating point.

```
{ "age":30 }
```

➤ JSON Objects

- Values in JSON can be objects.

```
{"employee":{ "name":"John", "age":30, "city":"New York"
}}
```



➤ JSON Arrays

- Values in JSON can be arrays.

```
{"employees": [ "John", "Anna", "Peter" ]}
```

➤ JSON Booleans

- Values in JSON can be true/false.

```
{ "sale": true }
```



Exchanging Data

- When exchanging data between a browser and a server, the data can only be text.
- JSON is text, and we can convert any JavaScript object into JSON, and send JSON to the server.
- We can also convert any JSON received from the server into JavaScript objects.
- This way we can work with the data as JavaScript objects, with no complicated parsing and translations.



Sending Data

- If you have data stored in a JavaScript object, you can convert the object into JSON using **JSON.stringify()** function , and send it to a server.
- [RUN\(json_send.html\)](#)

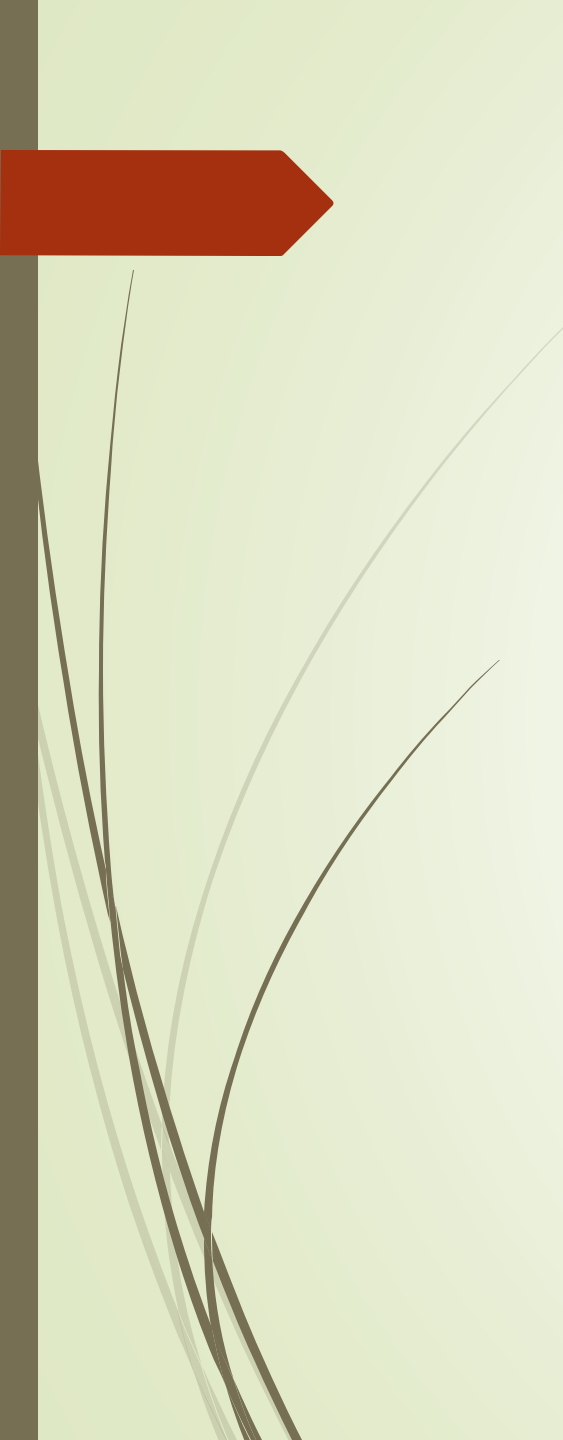
Receiving Data

- If you receive data in JSON format, you can convert it into a JavaScript object using **JSON.parse()** function.
- [RUN\(json_receive.html\)](#)



JSON vs XML

- JSON doesn't use tags.
- JSON is shorter
- JSON is quicker to read and write
- JSON can use arrays
- XML has to be parsed with an XML parser. JSON can be parsed by a standard JavaScript function.
- For applications, JSON is faster and easier than XML



JSON Example

```
{ "employees": [  
  { "firstName": "John", "lastName": "Doe" },  
  { "firstName": "Anna", "lastName": "Smith" },  
  { "firstName": "Peter", "lastName": "Jones" }  
]}
```

XML Example

```
<employees>  
  <employee>  
    <firstName>John</firstName> <lastName>Doe</lastName>  
  </employee>  
  <employee>  
    <firstName>Anna</firstName> <lastName>Smith</lastName>  
  </employee>  
  <employee>  
    <firstName>Peter</firstName> <lastName>Jones</lastName>  
  </employee>  
</employees>
```



Thank you