MODULE – 03

JavaScript

BASIC CONCEPT OF JAVASCRIPT

- Web pages made using only HTML are somewhat static with no interactivity and negligible user involvement.
- HTML tags are just instructions on document and the display of the document is dependent on the browser.
- Interactive pages cannot be built with only HTML, we need a programming language. So Netscape came out with a client-side language called as JavaScript.

WHAT IS JAVASCRIPT?

- JavaScript is the scripting language of the Web.
- JavaScript is used in Web pages to add functionality, validate forms, detect browsers, and much more.
- JavaScript is the most popular scripting language on the internet and works on all major browsers available like IE, Firefox, Chrome etc..

WHAT IS NEED OF JAVASCRIPT?

- For putting dynamic content into an HTML Page.
- For client side Validation.
- For storing and retrieving client's information in the form of Cookies.

HOW DOES JAVASCRIPT WORKS?

- When a JavaScript is inserted into an HTML document, the Internet browser will read the HTML and interpret the JavaScript.
- The JavaScript can be executed immediately, or at a later event.

EMBEDDING JS IN HTML

- In HTML you can use script tag for embedding javascript.
  - <SCRIPT>……</SCRIPT>

- Attributes for script tag
  - src :- Specifies the url of external script file
  - type :- Specifies the media type of the script
  - async, reffererpolicy & many more…

- Use the <NOSCRIPT> tag to specify alternate content for browsers that do not support JavaScript.

WHERE TO WRITE JS?

- Head Section
  - If you want the Scripts to be executed when they are called, or whenever a user action happens, put script tag in the head section.
- Body Section
  - If you want the Scripts to be executed when the page loads then put script tag in the body section
- External File
  - If you want to run the same JavaScript on several pages, without having to write the same script on every page, you can write a JavaScript in an external file.

WHERE TO WRITE JS?

- Head Section
  - If you want the Scripts to be executed when they are called, or whenever a user action happens, put script tag in the head section.
- Body Section
  - If you want the Scripts to be executed when the page loads then put script tag in the body section
- External File
  - If you want to run the same JavaScript on several pages, without having to write the same script on every page, you can write a JavaScript in an external file.

HELLO WORLD USING JS

- Let's implement the hello world program using JavaScript.

STATEMENT, SEMICOLON & COMMENTS

- Statements are syntax constructs and commands that perform actions.
  - alert('Hello to the'); alert('JS World');
- A semicolon may be omitted in most cases when a line break exists.

  alert("Hello")

  alert("World")
- Comments
  - //Single Line Comment
  - /* Multi Line Comment */

VARIABLES IN JS

- Why Variables?
- A variable is a "named storage" for data.
- A variable should be declared only once, repeated declaration cause an error (Exception :- var).
- Variables are case-sensitive.
- Variable name can't be a reserved keyword. Eg:- let, break, if, function etc.
- There are two limitations on variable names in JavaScript:
  - The name must contain only letters, digits, or the symbols $ and _.
  - The first character must not be a digit.
- Types of Variable
  - let
  - var
  - const

LET KEYWORD

- Variable Declaration
  - let name;
- Variable Assignment
  - name = “Neelkanth”;
- Variable Declaration with Assignment
  - let surname = “Patel”;
- Multiple Variables
  - let name = “Neelkanth”, surname=“Patel”;

VAR KEYWORD

- The var keyword is used in all JavaScript code from 1995 to 2015.
- There are subtle differences between let and var, we'll cover them in detail later.
- Variable Declaration
  - var name;
- Variable Assignment
  - var = "Neelkanth";
- Both Declaration & Assignment
  - var age = 27

CONST KEYWORD

- To declare a constant (unchanging) variable, use const instead of let.
  - const myBirthday = '18.04.1995;

- There is a widespread practice to use constants as aliases for difficult-to-remember values that are known prior to execution.
- Such constants are named using capital letters and underscores.

DATATYPES

- A value in JavaScript is always of a certain type. For example, a string or a number.
- There are eight basic data types in JavaScript.
- We can put any type in a variable.
  - For example, a variable can at one moment be a string and then store a number:

NUMBER

- The number type represents both integer and floating point numbers.
- There are many operations for numbers, e.g. multiplication *, division /, addition +, subtraction -, etc.
- Besides regular numbers, there are so-called “special numeric values” which also belong to this data type: Infinity, -Infinity and NaN.

BIGINT

- In JavaScript, the “number” type cannot safely represent integer values larger than (253-1) (that’s 9007199254740991), or less than -(253-1) for negatives.
- BigInt type was recently added to the language to represent integers of arbitrary length.
- A BigInt value is created by appending n to the end of an integer:

STRING

- A string in JavaScript must be surrounded by quotes.
- In JavaScript, there are 3 types of quotes.
  - Double quotes: "Hello".
  - Single quotes: 'Hello'.
  - Backticks: `Hello`.
- Double and single quotes are “simple” quotes. There’s practically no difference between them in JavaScript.
- Backticks are “extended functionality” quotes. They allow you to embed variables and expressions into a string by wrapping them in ${…}.

BOOLEAN

- The boolean type has only two values: true and false.
- This type is commonly used to store yes/no values: true means “yes, correct”, and false means “no, incorrect”.

NULL

- The special null value does not belong to any of the types described above.
- It forms a separate type of its own which contains only the null value.
- It's just a special value which represents "nothing", "empty" or "value unknown".

UNDEFINED

- The special value undefined also stands apart. It makes a type of its own, just like null.
- The meaning of undefined is “value is not assigned”.
- If a variable is declared, but not assigned, then its value is undefined:

OBJECTS

- The object type is special.
- All other types are called “primitive” because their values can contain only a single thing (be it a string or a number or whatever).
- In contrast, objects are used to store collections of data and more complex entities.

TYPEOF

- The typeof operator returns the type of the argument. It's useful when we want to process values of different types differently or just want to do a quick check.
- A call to typeof x returns a string with the type name:

OPERATORS

CONDITIONALS

- If statement
  - The if(…) statement evaluates a condition in parentheses and, if the result is true, executes a block of code.
- Else statement
  - The if statement may contain an optional “else” block. It executes when the condition is falsy.
- Else if Ladder
  - Sometimes, we’d like to test several variants of a condition. The else if clause lets us do that.

DIALOGS

- There are basically three types of dialogs that we will use in JavaScript, namely :-
  - Alert
    - It shows a message and waits for the user to press “OK”.
    - alert(“You are directing to Website");
  - Confirm
    - The function confirm shows a modal window with a question and two buttons: OK and Cancel.
    - The result is true if OK is pressed and false otherwise.
    - confirm(“Are you 18 years old”);
  - Prompt
    - prompt(“Enter your age”, [default]);
    - It shows a modal window with a text message, an input field for the visitor, and the buttons OK/Cancel.

LOOPS

- Loops are a way to repeat the same code multiple times.
- There are 3 basic loops :-
  - While

    ```
    while (condition) {
     // code }
    ```
  - Do…while

    ```
    do {
     // loop body
    } while (condition);
    ```
  - For

    ```
    for (begin; condition; step) {
     // … loop body … }
    ```

LOOPS – CONTINUE & BREAK

- Normally, a loop exits when its condition becomes falsy.
- But we can force the exit at any time using the special break directive.

- The continue directive is a "lighter version" of break. It doesn't stop the whole loop. Instead, it stops the current iteration and forces the loop to start a new one (if the condition allows).

FUNCTION

- Functions are the main "building blocks" of the program. They allow the code to be called many times without repetition.
- To create a function we can use a function declaration.

```
function name(parameter1, parameter2, ... parameterN) {
  ...body...
}
```

FUNCTION VARIABLES

- Local variables
  - A variable declared inside a function is only visible inside that function.
- Outer Variables or Global Variables
  - Variables declared outside of any function.
  - Global variables are visible from any function.
  - If a same-named variable is declared inside the function then it *shadows* the outer one.

FUNCTION PARAMETERS

- We can pass arbitrary data to functions using parameters.
- When a value is passed as a function parameter, it's also called an *argument*.

```
function name(parameter1, parameter2, ... parameterN) {
  ...body...
}
name(2,"String",true)
```

- Default Parameters
  - If a function is called, but an argument is not provided, then the corresponding value becomes undefined.
  - To avoid this we use default parameter for function.
  - function getName(name="No name Provided") {….}

FUNCTION RETURN VALUE

- A function can return a value back into the calling code as the result.
- There may be many occurrences of return in a single function.
- Syntax:-

```
function name(….){
        return “Hello”;
}
```

ARRAYS

- To store ordered collections in javascript, we use special data structure named Array.
- There are two syntaxes for creating an empty array:
  - let arr = new Array();
  - let arr = [];
- For array initialization we use
  - Colors = [“Red", “Green", “Blue"];
- Array elements are numbered, starting with zero.
- We can get an element by its number in square brackets.

ARRAYS FUNCTIONS

- The total count of the elements in the array is its length.
  - Color.length
- Array Loops
  - For of loop
    - The for..of give access to the value of the current element.
  - For In loop
    - The for..of give access to the index of the current element.

ARRAYS FUNCTIONS

- at keyword is used for retrieving values from particular array.
  - In other words, arr.at(i):
    - is exactly the same as arr[i], if i >= 0.
    - for negative values of i, it steps back from the end of the array.
- There are some other common methods namely push, pop, shift, unshift

ARRAYS FOR EACH

- The arr.forEach method allows to run a function for every element of the array.

```
arr.forEach(function(item, index, array) {
  // ... code
});
```

STRINGS

- In JavaScript, the textual data is stored as strings. There is no separate type for a single character.
- String with backtics are special. They allow multiline spaning.

STRINGS - SPECIAL CHARACTERS

- There are some special character used in string. Some of them are:-
  - \n
  - \t
  - \”
  - \’
  - \\

STRINGS - FUNCTION

- String length
  - The length property gives the string length.
- String chatAt
  - To get a character at certain index, use square brackets [index] or call the method str.charAt(index).
- Changing the case
  - Methods toLowerCase() and toUpperCase() change the case.
- Get Substring
  - String.substring(start, end) between start and end indexes

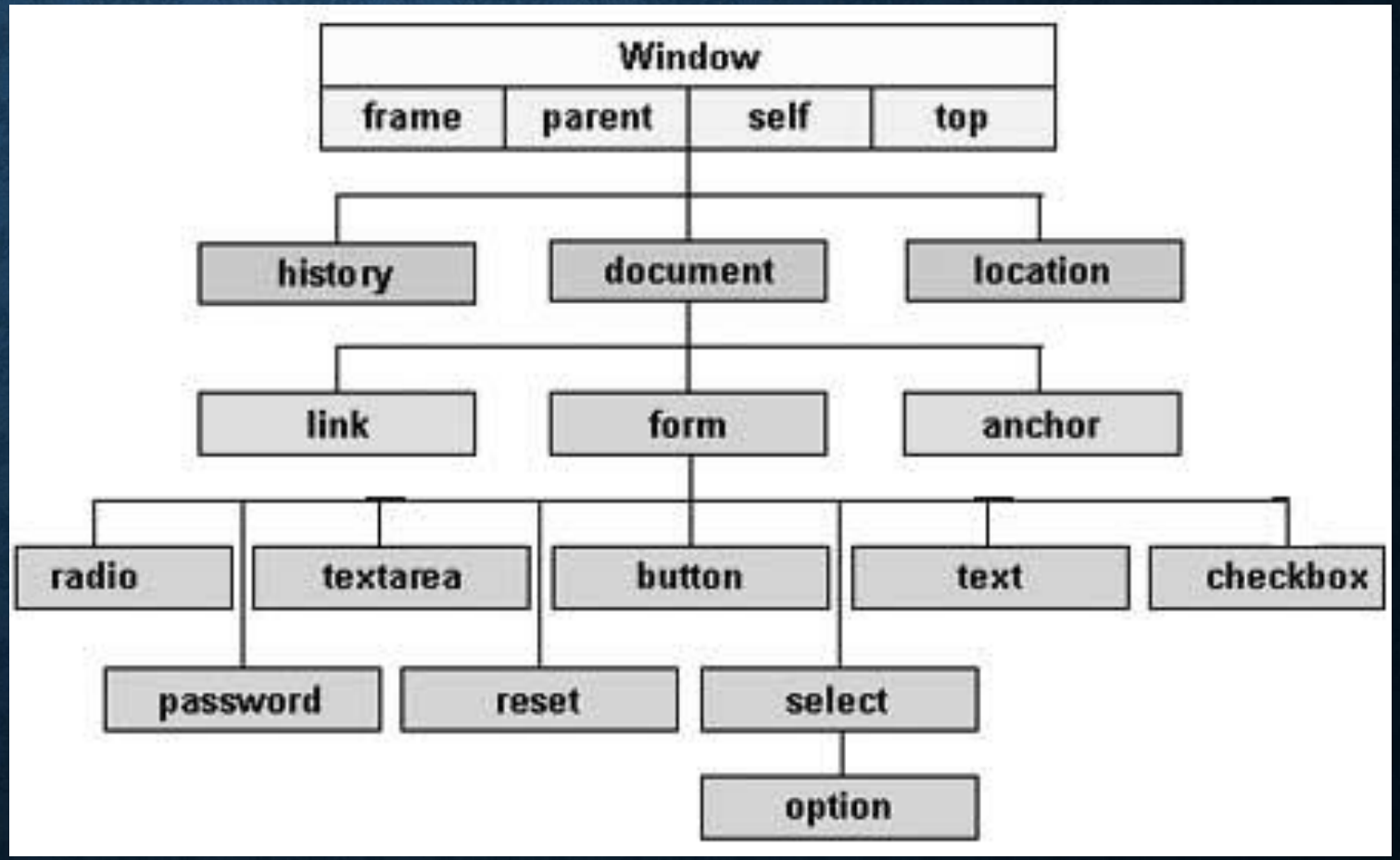
STRINGS

- Strings are immutable
  - Strings can’t be changed in JavaScript. It is impossible to change a character.
- Comparing strings
  - strings are compared character-by-character in alphabetical order.
    - But, a lowercase letter is always greater than the uppercase.

HASH TABLE

DOCUMENT OBJECT MODEL

- The Document Object Model, or DOM for short, represents all page content as objects that can be modified.
- The document object is the main "entry point" to the page. We can change or create anything on the page using it.

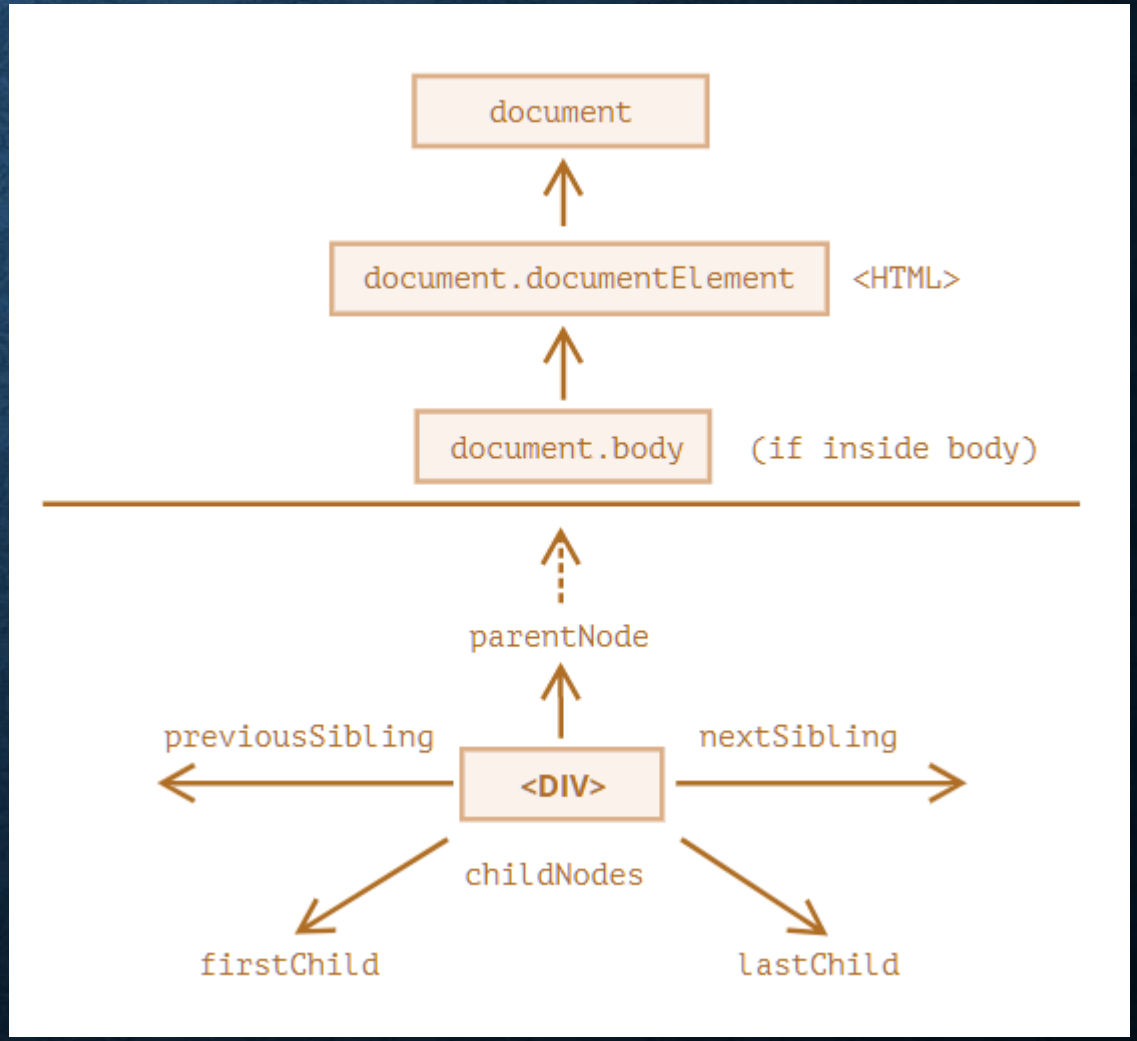


DOM TREE

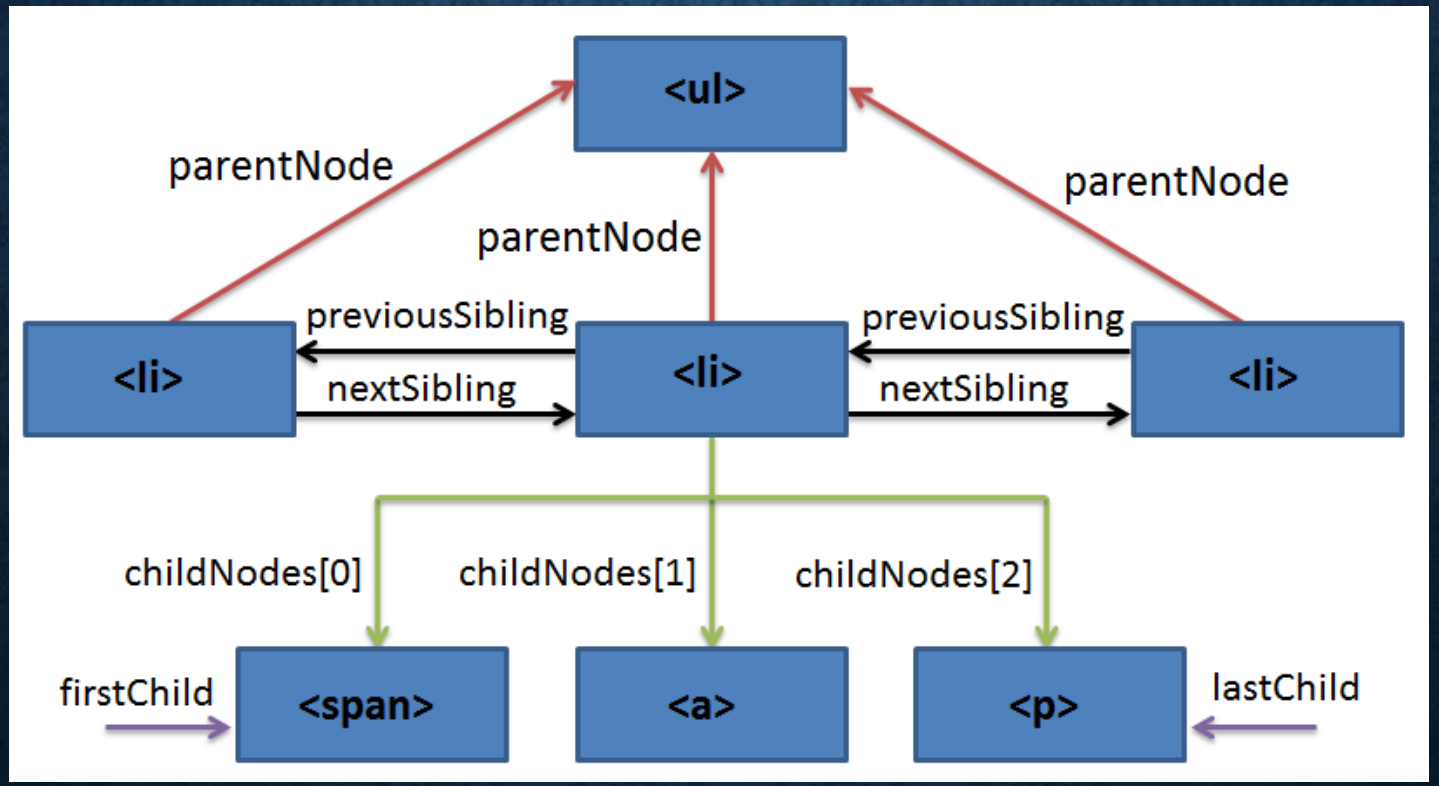
- The backbone of an HTML document is tags.
- According to the Document Object Model (DOM), every HTML tag is an object. Nested tags are “children” of the enclosing one. The text inside a tag is an object as well.
- All these objects are accessible using JavaScript, and we can use them to modify the page.
- Comments & whitespaces are also part of DOM.

DOM TRAVERSAL

- The topmost tree nodes are available directly as document properties.
- The topmost document node is document.documentElement. That's the DOM node of the <html> tag.
- Another widely used DOM node is the <body> element – document.body.
- The <head> tag is available as document.head.

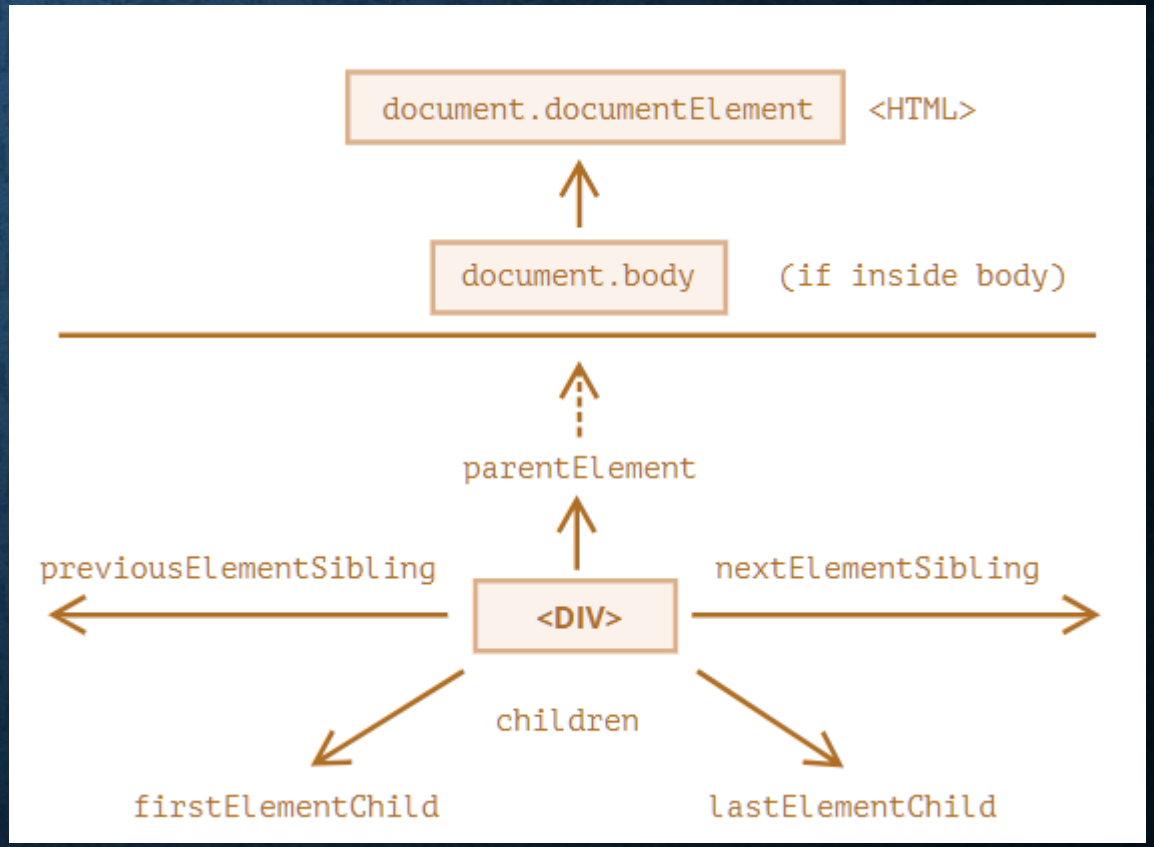


DOM TRAVERSAL



DOM ELEMENT TRAVERSAL

- Navigation properties listed till now refer to all nodes. For instance, in childNodes we can see both text nodes, element nodes, and even comment nodes if they exist.
- The links are similar to those given above, just with Element word inside:
  - children – only those children that are element nodes.
  - firstElementChild, lastElementChild – first and last element children.
  - previousElementSibling, nextElementSibling – neighbor elements.
  - parentElement – parent element.



SEARCHING DOM: getElement*

- document.getElementById
  - If an element has the id attribute, we can get the element using the method document.getElementById(id), no matter where it is in the page.
- getElementsByTagName(tag)
  - getElementsByTagName(tag) looks for elements with the given tag and returns the collection of them. The tag parameter can also be a star "*" for “any tags”.
- getElementsByClassName(className)
  - getElementsByClassName(className) returns elements that have the given CSS class.
- getElementsByName(name)
  - getElementsByName(name) returns elements with the given name attribute, document-wide. Very rarely used.

SEARCHING DOM: querySelector*

- querySelectorAll(css)
  - querySelectorAll(css) returns all elements inside elem matching the given CSS selector.
- querySelector(css)
  - querySelector(css) returns the first element for the given CSS selector.
- matches(css)
  - matches(css) does not look for anything, it merely checks if elem matches the given CSS-selector. It returns true or false.
- closest(css)
  - closest(css) looks for the nearest ancestor that matches the CSS-selector. The element itself is also included in the search.

REMEMBER

- Only document.getElement*/querySelector*, or anyElem.getElement*/querySelector*
  - The method getElementById can be called only on document object. It looks for the given id in the whole document.
  - Other all can be on any element.

DOM INNER & OUTER HTML

- The innerHTML property allows to get the HTML inside the element as a string.
- We can also modify it. So it's one of the most powerful ways to change the page.

- The outerHTML property contains the full HTML of the element. That's like innerHTML plus the element itself.
- unlike innerHTML, writing to outerHTML does not change the element. Instead, it replaces it in the DOM.