JavaScript의 DOM(Document Object Model) vs BOM(Browser Object Model)

어떤 코드가 DOM에 해당되느냐, BOM에 해당되느냐에 따라서 HTML page와 Browser가 각기 주도권을 주고 받는 과정이 달라져 코드 실행 결과가 예상과 달라지기도 한다.

☑ BOM (Browser Object Model)

Browser Object Model(BOM)에 대한 공식 표준은 없다. 최신 Browser는 JavaScript 상호 작용에 대해 (거의) 동일한 method와 property를 구현했기 때문에 BOM의 method 및 property 라고 하는 경우가 많다.
Browser Object Model(BOM)은 Browser에 대한 모든 내용을 담고있는 Object 이다. Browser에 관한 정보를 제공하거나 Browser의 모양을 제어하도록 제공되는 object 들이다.

❖ Window Object

window object는 모든 Browser에서 지원된다. Browser의 window를 나타낸다. 모든 global JavaScript object, function 및 variable은 자동으로 window object의 member가 된다.
global variable은 window object의 property 이다. global function은 window object의 method 이다.
HTML DOM의 document object 조차도 window object의 property 이다. 다음은 같은 의미이다.

```
window.document.getElementById("header");
document.getElementById("header");
```

대표적으로 아래와 같은 Object들이 존재한다.

❖ Window Object Property

- ☞ location (window의 Location object를 반환한다) (url address와 관련)
- ☞ navigator (window의 Navigator object를 반환한다)
- ☞ history (window에 대한 History object를 반환한다)
- ☞ screen (window에 대한 Screen object를 반환한다)
- ☞ document (window의 Document object를 반환한다)
- ☞ string, boolean, object, number, function, array 등의 data type

Window Object Properties

Property	Description
closed	Returns a boolean true if a window is closed.
console	Returns the Console Object for the window. See also The Console Object.
defaultStatus	Deprecated.
document	Returns the Document object for the window. See also The Document Object.
frameElement	Returns the frame in which the window runs.
frames	Returns all window objects running in the window.
history	Returns the History object for the window. See also The History Object.
innerHeight	Returns the height of the window's content area (viewport) including scrollbars
innerWidth	Returns the width of a window's content area (viewport) including scrollbars
length	Returns the number of <iframe> elements in the current window
localStorage	Allows to save key/value pairs in a web browser. Stores the data with no expiration date
location	Returns the Location object for the window. See also the The Location Object.
name	Sets or returns the name of a window
navigator	Returns the Navigator object for the window. See also The Navigator object.
opener	Returns a reference to the window that created the window
outerHeight	Returns the height of the browser window, including toolbars/scrollbars
outerWidth	Returns the width of the browser window, including toolbars/scrollbars
pageXOffset	Returns the pixels the current document has been scrolled (horizontally) from the upper left corner of the window
pageYOffset	Returns the pixels the current document has been scrolled (vertically) from the upper left corner of the window
parent	Returns the parent window of the current window
screen	Returns the Screen object for the window See also The Screen object

Window Object Methods

Method	Description
alert()	Displays an alert box with a message and an OK button
atob()	Decodes a base-64 encoded string
blur()	Removes focus from the current window
btoa()	Encodes a string in base-64
clearInterval()	Clears a timer set with setInterval()
clearTimeout()	Clears a timer set with setTimeout()
close()	Closes the current window
confirm()	Displays a dialog box with a message and an OK and a Cancel button
focus()	Sets focus to the current window
getComputedStyle()	Gets the current computed CSS styles applied to an element
getSelection()	Returns a Selection object representing the range of text selected by the user
matchMedia()	Returns a MediaQueryList object representing the specified CSS media query string
moveBy()	Moves a window relative to its current position
moveTo()	Moves a window to the specified position
open()	Opens a new browser window
print()	Prints the content of the current window
prompt()	Displays a dialog box that prompts the visitor for input
requestAnimationFrame()	Requests the browser to call a function to update an animation before the next repaint
resizeBy()	Resizes the window by the specified pixels
resizeTo()	Resizes the window to the specified width and height
scroll()	Deprecated. This method has been replaced by the scrollTo() method.
scrollBy()	Scrolls the document by the specified number of pixels
scrollTo()	Scrolls the document to the specified coordinates
setInterval()	Calls a function or evaluates an expression at specified intervals (in milliseconds)
setTimeout()	Calls a function or evaluates an expression after a specified number of milliseconds
stop()	Stops the window from loading

여기에서 Window Object는 모든 BOM object들의 ancestor(조상) 즉, 최상위 Object(global object)이다. 모든 Object를 다 포함하고 있기 때문에 코딩을 할 때는 굳이 표기하지 않고 생략이 가능하다.

하나의 window(열린 창)마다 하나의 Window Object가 생성된다. 또한 사용자가 선언한 global variable(전역 변수)들도 모두 Window Object 안에 등록된다. (단, local variable(지역변수) 즉, function 안에서 선언한 변수들은 그 function 안에서만 쓸 수 있다.)

따라서 위에서 적은 대표적 Object들 말고도 굉장히 많은 property, collection, method들이 존재한다.

```
window.open("sURL", "sWindowName", "sFeature");
window.close();
```

그 중에서도 많이 사용하는 것은 위와 같이 window 창을 열고 닫는 method 이다.

☑ DOM(Document Object Model)

DOM(Document Object Model)은 web document 용 programming interface 이다. program이 document structure, style 및 content를 변경할 수 있도록 page를 나타낸다. DOM은 document를 node와 object로 나타낸다. 그렇게 하면 programming language가 page와 상호 작용할 수 있다.

web page는 Browser window에 표시되거나 HTML source로 표시될 수 있는 document 이다. 두 경우 모두 동일한 document 이지만 DOM(Document Object Model) 표현을 통해 조작할 수 있다. web page의 Object-Oriented 표현으로 JavaScript와 같은 scripting language로 수정할 수 있다.

DOM(Document Object Model)은 web page를 나타내는 HTML과 같은 document structure를 memory에 표시하여 web page를 script 또는 programming language에 연결한다. HTML, SVG 또는 XML document를 object로 modeling 하는 것이 핵심 JavaScript language의 일부가 아니지만 일반적으로 JavaScript를 나타낸다.

DOM은 logical tree가 있는 document를 나타낸다. tree의 각 분기는 node로 끝나고 각 node는 object를 포함한다. DOM method를 사용하면 tree에 프로그래밍 방식으로 액세스할 수 있다. 그것들을 사용하여 document의 structure, style 또는 content를 변경할 수 있다.

node에는 event handler가 연결되어 있을 수도 있다. event가 trigger 되면 event handler가 실행된다.

Web Page가 load 되면 Browser는 page의 Document Object Model(DOM)을 생성한다. HTML DOM model은 Object Tree로 구성된다. Document Object Model(DOM)은 Document에 대한 모든 내용을 담고 있는 Object이다. HTML tag를 dynamic(동적)으로 제어하기 위해 반드시 알아두어야 할 개념이다.

Object Modle을 통해 JavaScript는 dynamic HTML을 생성하는 데 필요한 모든 기능을 얻을 수 있다.

- ☞ JavaScript는 page의 모든 HTML element를 변경할 수 있다.
- ☞ JavaScript는 page의 모든 HTML attribute를 변경할 수 있다.
- ☞ JavaScript는 page의 모든 CSS style을 변경할 수 있다.
- ☞ JavaScript는 기존 HTML element 및 attribute를 제거할 수 있다.
- ☞ JavaScript는 새로운 HTML element 및 attribute을 추가할 수 있다.
- ☞ JavaScript는 page의 모든 기존 HTML event에 반응할 수 있다.
- ☞ JavaScript는 page에서 새로운 HTML event를 생성할 수 있다.

Browser는 HTML page를 load 하는 과정에서 HTML tag들을 각기 하나의 Object로 만든다. 따라서 DOM은 HTML tag 당 하나씩 있으며, name은 tag name과 같다.

예를 들어 ... 로 구성된 element는 p object, 로 구성된 요소는 div object 이런 식이다.

이와 같이 HTML page의 각 element를 object화한 것이 HTML DOM Object이다.

❖ DOM Object는 5종류의 속성으로 구성된다.

- ☞ property : DOM Object의 member variable. HTML tag의 attribute 반영
- ☞ method : DOM Object의 emember function수. HTML tag를 제어
- ☞ event listener : HTML tag에 작성된 eventListerner()를 그대로 가진다.
- ☞ collection: 정보를 집합적으로 표현하는 일종의 배열. 예를 들어 childeren collection은 DOM Object의 모든 child DOM Object에 대한 address를 가진다.
- ☞ css style : style property를 통해 HTML tag에 적용된 CSS style sheet에 접근 가능

❖ DOM Tree

HTML tag는 포함관계(즉, parent-child 관계)가 존재한다. Browser는 HTML page를 load 하면서 이 포함관계에 따라 DOM Object들을 Tree 구조로 만드는데, 이게 바로 DOM Tree 이다.

DOM Tree의 최상위 Object는 document Object이다. 그러나 document Object는 DOM Object가 아니다. document Object는 BOM Object로 보아야 하기 때문이다. 이 때문에 document Object에는 CSS style sheet가 없어 CSS style property 등과 연결되어 있지 않고, document.style.color 등의 접근은 잘못된 코드이므로 오류가 발생한다.

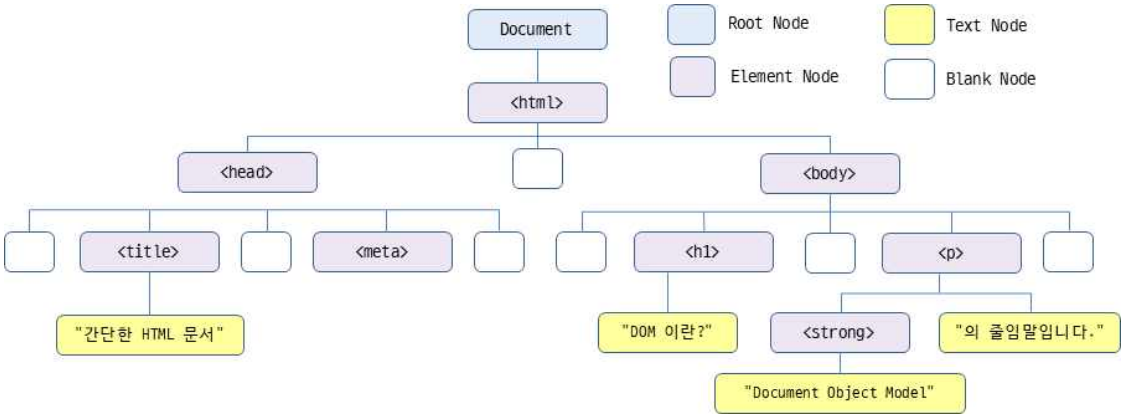
```
document.open();
document.close();
```

document Object에서도 open(), close() 함수가 존재한다. 그러나 window 창 자체를 열고 닫았던

window Object의 open(), close()와는 달리 document Object의 open(), close()는 Browser 내 HTML text를 지워버리고(open()), 새로운 내용을 쓴 뒤(document.write())에 Object를 닫아주는(close()) 역할을 한다. 따라서 document.open(); 을 실행한 후 Browser의 개발 도구의 소스를 보면 깨끗해진 모습을 볼 수 있다.

DOM Tree를 구성하는 Object 하나를 Node(노드)라고 한다. Tree에서 위쪽은 parent node(부모 노드), 아래쪽을 child node(자식 노드) 라고 한다. Document를 제외하고 최상단에 있는 html은 root node가 된다.
DOM은 Tree 구조의 Data Structure 이다. 이 하나의 Object를 Node 라고 부른다. Tree에서 위쪽은 parent node, 아래쪽을 child node 라고 한다. Document를 제외하고 최상단에 있는 <html>은 Root Node 가 된다.

- ☞ Root Node : 전체 document를 가리키는 Document Object 이며 document로 참조할 수 있다.
- ☞ Element Node : HTML element를 가리키는 Object(=> element Object) 이다.
- ☞ Text Node : text를 가리키는 Object(=> text Object) 이다.
- ☞ Blank Node :



☑ DOM과 JavaScript

거의 모든 예제와 마찬가지로 앞의 짧은 예제는 JavaScript 이다. 즉, JavaScript로 작성되었지만 DOM을 사용하여 document와 해당 element에 액세스 한다.
DOM은 programming language가 아니지만, DOM이 없으면 JavaScript language에는 web page, HTML document, SVG document 및 component 부분에 대한 model이나 개념이 없다.
document 전체, head, document 내의 table, table header, table cell 내의 text 및 document의 기타 모든 element는 해당 document에 대한 DOM의 일부이다. DOM과 JavaScript와 같은 scripting lnaguage를 사용하여 모두 액세스하고 조작할 수 있다.

DOM은 JavaScript language의 일부가 아니라 Web site를 구축하는 데 사용되는 Web API 이다. JavaScript는 다른 context에서도 사용할 수 있다.
예를 들어 Node.js는 Computer에서 JavaScript program을 실행하지만 다른 API set를 제공하며, DOM API는 Node.js runtime의 핵심 부분이 아니다.

DOM은 특정 program language와 독립적으로 설계되어, document의 structural representation(구조적 표현)을 일관된 단일 API에서 사용할 수 있다. 대부분의 Web 개발자가 JavaScript를 통해서만 DOM을 사용하더라도 다음 Python 예제에서 볼 수 있듯이 DOM 구현은 모든 language에 대해 build 할 수 있다.

```
# Python DOM example
import xml.dom.minidom as m

doc = m.parse(r"C:\Projects\Py\chap1.xml")
doc.nodeName          # DOM property of document object
p_list = doc.getElementsByTagName("para")
```

☑ 기본 data type

이 page에서는 다양한 object와 type을 간단한 용어로 설명한다. 그러나 API 주위에 전달되는 다양한 data type이 있다는 사실을 알고 있어야 한다. 참고로 DOM을 사용하는 대부분의 코드는 HTML document 조작을 중심으로 돌아가기 때문에 DOM의 node를 element로 참조하는 것이 일반적이지만 엄밀히 말하면 모든 node가 element는 아니다.

다음 표에서는 이러한 data type에 대해 간략하게 설명한다.

Data Type (Interface)	Description
Document	member가 document type의 object를 반환하는 경우(예: element의 ownerDocument property가 해당 element가 속한 document를 반환) 이 object는 root document서object 자체이다. DOM document Reference에서 document object를 참조하도록 한다.
Node	document 내에 있는 모든 object는 일종의 node 이다. HTML document에서 object는 element node 일 수 있지만, text node 또는 attribute node 일 수도 있다.
Element	element type은 node를 기반으로 한다. DOM API의 member가 반환한 element 또는 type element의 node를 참조한다. 예를 들어 document.createElement() method가 node에 대한 object reference를 반환한다고 말하는 대신, 이 method가 DOM에서 방금 생성된 element를 반환한다고 말한다. element object는 DOM Element interface와 더 기본적인 Node interface를 구현하며, 둘 다 이 reference(참조)에 함께 포함되어 있다. HTML document에서 element는 HTML DOM API의 HTMLElement interface와 특정 종류의 element(예: <table> element의 경우 HTMLTableElement)의 기능을 설명하는 다른 interface에 의해 더욱 향상된다.
NodeList	nodeList는 document.querySelectorAll() method에서 반환되는 종류와 같은 element의 array 이다. nodeList의 item은 다음 두 가지 방법 중 하나로 index에 의해 액세스 된다. · list.item(1) · list.[1] 이 둘은 동등하다. 첫 번째에서 item()은 nodeList object의 single method 이다. 후자는 일반적인 array syntax를 사용하여 list의 두 번째 항목을 가져온다.
Attr	attribute이 member에 의해 반환되면(예: createAttribute() method에 의해) attribute에 대한 특별한(작지만) interface를 노출하는 object reference 이다. attribute는 element와 마찬가지로 DOM의 node 이지만 거의 사용하지 않을 수 있다.
NamedNodeMap	NamedNodeMap은 array와 비슷하지만 item은 name이나 index로 액세스 된다. 후자의 경우는 list에서 특정 order가 없기 때문에 단순히 enumeration(열거)하기 위한 편의일 뿐이다. namedNodeMap에는 이를 위한 item() method가 있으며, namedNodeMap에서 item을 add(추가) 및 remove(제거) 할 수도 있다.

또한 염두에 두어야 할 몇 가지 일반적인 용어 고려사항이 있다. 예를 들어 모든 Attr node를 element로 reference 하고, DOM node array를 nodeList로 reference 하는 것이 일반적이다. document 전체에 걸쳐 소개되고 사용되는 이러한 term(용어)와 기타 용어를 찾을 수 있다.

☑ DOM interface

이 가이드는 DOM 계층 구조를 조작하는 데 사용할 수 있는 object와 실제 things에 관한 것이다. 이러한 작동 방식을 이해하는 것이 혼란스러울 수 있는 많은 지점이 있다. 예를 들어 HTML form element를 나타내는 object는 HTMLFormElement interface에서 name property를 가져오지만, HTMLElement interface에서 className property를 가져온다. 두 경우 모두 원하는 property는 해당 form objecct에 있다.

그러나 object와 object체가 DOM에서 구현하는 interface 간의 relationship(관계)는 혼란스러울 수 있으므로, 여기에서는 DOM specification(사양)의 실제 interface와 이러한 interface를 사용할 수 있는 방법에 대해 약간 설명하려고 한다.

❖ interface 및 object

많은 object는 여러 다른 interface에서 차용한다. 예를 들어 table object는 createCaption 및 insertRow와 같은 method를 포함하는 특수 HTMLTableElement interface를 implements(구현) 한다.
그러나 HTML element 이기도 하기 때문에 table은 DOM Element Reference에서 설명한 Element interface를 implements(구현) 한다.
마지막으로 HTML element는 DOM에 관한 한 HTML 또는 XML page의 object model을 구성하는 node tree의 node 이기도 하므로, table object 또한 Element가 파생되는 보다 기본적인 Node interface를 implements(구현) 한다.

다음 예에서와 같이 table object에 대한 참조를 얻을 때 일상적으로 이러한 세 가지 interface를 object에 대해 알지 못하는 사이에 상호 교환 가능하게 사용한다.

```
const table = document.getElementById("table");
const tableAttrs = table.attributes;       // Node/Element interface

for (let i = 0; i < tableAttrs.length; i++) {
  // HTMLTableElement interface: border attribute
  if(tableAttrs[i].nodeName.toLowerCase() == "border")
    table.border = "1";
}
// HTMLTableElement interface: summary attribute
table.summary = "note: increased border";
```

❖ DOM의 Core interfaces

DOM에서 가장 일반적으로 사용되는 interface를 나열한다. 아이디어는 이러한 API가 여기서 수행하는 작업을 설명하는 것이 아니라, DOM을 사용할 때 매우 자주 보게 될 method와 property의 종류에 대한 아이디어를 제공하는 것이다.

document 및 window object는 일반적으로 DOM programming에서 가장 자주 사용하는 interface의 object 이다. 간단히 말해서 window object는 Browser와 같은 것을 나타내며 document object는 document 자체의 root 이다.
Element는 일반 Node interface에서 inheritance(상속)되며 이 두 interface스는 함께 개별 element에 사용하는 많은 method와 property를 제공한다. 이러한 element에는 이전 섹션의 table object 예제에서와 같이 해당 element가 보유하는 table data 종류를 처리하기 위한 특정 interface가 있을 수도 있다.

다음은 DOM을 사용하는 Web 및 XML page scripting의 일반적인 API list 이다.

- ☞ document.querySelector(selector)
- ☞ document.querySelectorAll(name)
- ☞ document.createElement(name)
- ☞ parentNode.appendChild(node)
- ☞ element.innerHTML
- ☞ element.style.left
- ☞ element.setAttribute()
- ☞ element.getAttribute()
- ☞ element.addEventListener()
- ☞ window.content
- ☞ GlobalEventHandlers/onload
- ☞ window.scrollTo()

❖ Element.innerHTML

Element property innerHTML은 element 내에 포함된 HTML 또는 XML markup을 가져오거나 설정한다.
element의 content를 바꾸지 않고 document에 HTML을 삽입하려면 insertAdjacentHTML() method를 사용한다.

☞ Value

element의 descendant(자손)에 대한 HTML serialization(직렬화)를 포함하는 DOMString 이다. innerHTML value를 설정하면 element의 모든 descendant(자손)이 remove(제거)되고, htmlString string에 제공된 HTML을 parsing 하여 생성된 node로 대체된다.

☞ 사용 참고 사항

innerHTML property는 page가 처음 로드된 이후 변경된 사항을 포함하여 page의 현재 HTML source를 검사하는 데 사용할 수 있다.

☞ element의 HTML content 읽기

innerHTML을 읽으면 user agent가 element의 descendant(자손)으로 구성된 HTML 또는 XML 조각을 serialization(직렬화) 한다. 결과 string이 반환된다.

```
let contents = myElement.innerHTML;
```

참고로 반환된 HTML 또는 XML 조각은 element의 현재 content를 기반으로 생성되므로 반환된 조각의 markup 및 formatting 이 original page markup과 일치하지 않을 수 있다.

☞ element의 content 바꾸기

innerHTML value를 설정하면 element의 기존 content를 새 coontent로 쉽게 바꿀 수 있다.
참고로 삽입할 string에 잠재적으로 malicious content(악성 콘텐츠)가 포함될 수 있는 경우 이는 security risk 이다. user-supplied data를 삽입할 때 삽입하기 전에 content를 삭제하기 위해 항상 Element.SetHTML()을 대신 사용하는 것을 고려해야 한다.
예를 들어 document의 body attribute content를 지워 document의 전체 내용을 지울 수 있다.

```
document.body.innerHTML = "";
```

이 예는 문서의 현재 HTML markup을 가져오고 "<" chracter를 HTML entity "<"로 대체하여 기본적으로 HTML을 raw text로 변환한다. 그런 다음 <pre> element로 wapping 된다. 그런 다음 innerHTML의 value가 이 새 string으로 변경된다. 결과적으로 document content는 page의 전체 소스 코드 표시로 대체된다.

```
document.documentElement.innerHTML = "<pre>" +
         document.documentElement.innerHTML.replace(/</g,"&lt;") +
           "</pre>";
```

☞ 운영 세부 정보

innerHTML의 value를 설정하면 정확히 어떻게 될까? 그렇게 하면 user agent가 다음 단계를 따르게 된다.

① 지정된 value는 document type에 따라 HTML 또는 XML로 parsing 되어 new element에 대한 new DOM node set를 나타내는 DocumentFragment object가 생성된다.

② content가 대체되는 element가 <template> element인 경우 <template> element의 content attribute는 1단계에서 생성된 new DocumentFragment로 대체된다.

③ 다른 모든 element의 경우 element의 content가 new DocumentFragment의 node로 대체된다.

☞ element에 HTML 추가

innerHTML value를 설정하면 element의 기존 content에 new content를 추가할 수 있다.
예를 들어 기존 list(<ul>)에 new list item(<li>)을 추가할 수 있다.

[HTML]

```
<ul id="list">
  <li><a href="#">Item 1</a></li>
  <li><a href="#">Item 2</a></li>
  <li><a href="#">Item 3</a></li>
</ul>
```

[JavaScript]

```
const list = document.getElementById("list");

list.innerHTML += `<li><a href="#">Item ${list.children.length + 1}</a></li>`;
```

innerHTML을 사용하여 html element를 추가하면(예: el.innerHTML += "link") 이전에 설정한 event listener가 제거된다. 즉, 그런 식으로 HTML element를 추가한 후에는 이전에 설정된 event listener를 수신할 수 없다.

JavaScript JSON(JavaScript Object Notation)

JavaScript Object를 String으로 표현하는 Data Format 이다.
JavaScript object literal과 유사하지만 object가 아닌 string 일 뿐이며, 이를 이용해 다른 Domain과 data를 주고 받을 수 있게 된다.

JSON은 XML 등 다른 데이터 포맷에 비해 간결하며, 손쉽게 JavaScript Object로 변환할 수 있으므로, Web Application에서 Server - Client 간의 데이터 교환에 주로 JSON을 사용한다.

☑ JSON 표기법

JSON은 JavaScript Object와 유사하지만 표기법이 조금 다르다. 표기법이 간결하긴 하지만, syntax(문법)에 예민하므로 정확히 사용하는 것이 중요하다.

```
{
    "name" : "Semyung",
    "age" : 33,
    "gender" : "male",
    "location" : "Jecheon",
    "marriage" : false,
    "friends" : ["ICT","SmartIT","SMU"]
}
```

☞ key는 반드시 double quotation(큰따옴표)로 둘러싸야 한다.
☞ value 값은 string, number, boolean, object, null 이 올 수 있다.
☞ key-value는 : (콜론) 기호를 이용해 구분하며, key-value 쌍은 , (쉼표) 기호로 구분한다.

☑ Serialization(직렬화) vs Deserialization(역직렬화)

☞ JavaScript Object <-> JSON string 으로 변환할 수 있다.

☞ Serialization(직렬화)는 컴퓨터 메모리 상에 존재하는 Object를 string 으로 변환하는 것을 말한다.

☞ Deserialization(역직렬화) 또는 Parsing은 string을 JavaScript Object로 반환하는 것을 말한다.

☞ 이러한 과정을 통해 데이터를 다양한 Domain 간에 주고 받을 수 있게 되는 것이다.

❖ JavaScript Object를 JSON string 으로 변경 (Serialization)

☞ JSON.stringify(JavaScript Object) method를 사용

JavaScript Object를 JSON string data로 변경할 수 있다.

```
const smu = {
  name : 'smart',
  age : 33,
  gender : 'male',
  location : 'Jecheon',
  marriage : false,
  friends : ['ICT','SmartIT','SMU']
}

const serialize_json = JSON.stringify(smu);

console.log("Serialize 결과 타입 : ", typeof serialize_json);
console.log("Serialize 결과 : ", serialize_json);

//Serialize 결과 타입 :  string
//Serialize 결과 :
{"name":"smart","age":33,"gender":"male","location":"Jecheon","marriage":false,"friends":["ICT",
"SmartIT","SMU"]}
```

❖ JSON string 을 JavaScript Object 로 변경 (Deserialization)

☞ JSON.parse() method를 사용

JSON string data를 JavaScript Object로 변경할 수 있다.

```
const deserialzie_object = JSON.parse(serialize_json);

console.log("Deserialize 결과 타입 : ", typeof deserialzie_object);
console.log("Deserialize 결과 : ", deserialzie_object);

//Deserialize 결과 타입 :  object
//Deserialize 결과 :  {
  name: 'smart',
  age: 33,
  gender: 'male',
  location: 'Jecheon',
  marriage: false,
  friends: [ 'ICT', 'SmartIT', 'SMU' ]
}
```