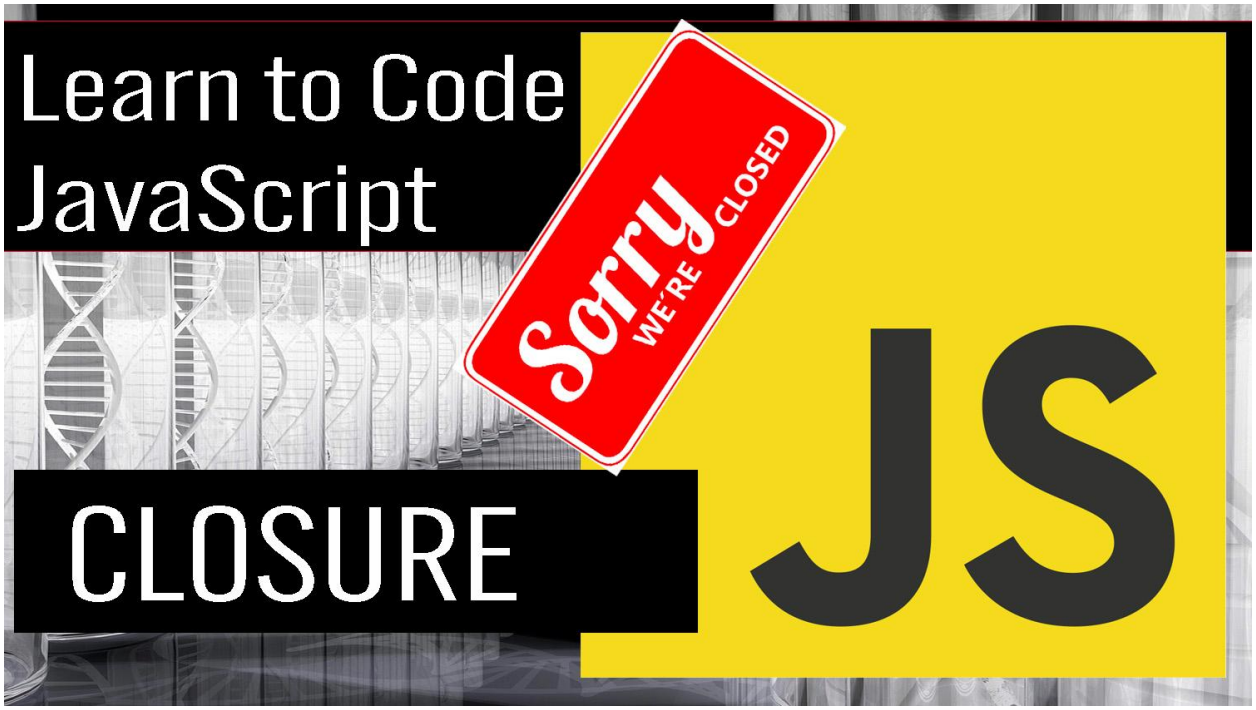


JavaScript Closure Explained

https://youtu.be/7vyHA0aODeU

A closure in JavaScript is a function that has access to variables in its parent scope, even after the parent function has returned. Closures are created when a function is defined inside another function, and the inner function retains access to the variables in the outer function's scope.

Here is an example of a closure in JavaScript:

code example

```
function outerFunction(x) {
    var innerVar = 4;
    function innerFunction() {
        return x + innerVar;
    }
    return innerFunction;
```

```
}

var closure = outerFunction(2);
console.log(closure()); // Output: 6
```

In this example, the innerFunction is a closure because it has access to the variable x and innerVar from the outerFunction even after outerFunction has returned.

A closure has three scope chains:

1. It has access to its own scope (variables defined between its curly braces {}).
2. It has access to the outer function's variables.
3. It has access to the global variables.

Closures are commonly used in JavaScript for a variety of tasks, such as:

- Implementing private methods and variables.
- Creating callback functions that retain access to variables from their parent scope.
- Creating and returning an object that has access to variables from its parent scope.

JavaScript closures are an important concept and it is important to understand how closures work in JavaScript. It is also important to be aware of the scope chain, and how closures interact with the scope chain.

```
<!DOCTYPE html>
<html>
<head>
  <title>Learn JavaScript</title>
```

```
</head>
<body>
  <h1>Complete JavaScript Course </h1>
  <div class="output"></div>
  <script src="app6.js"></script>
</body>
</html>
const val1 = 10;

function outerFun(x){
  const val2 = 10;
  function innerFun(){
     return x + val2 + val1;
  }
  return innerFun;
}

const val3 = outerFun(15);
console.log(val3());

for(let x=0;x<10;x++){
  console.log(outerFun(x+2)());
}
```

JavaScript Object Notation (JSON)

JavaScript Object Notation (JSON) is a lightweight data-interchange format that is easy for humans to read and write and easy for machines to parse and generate. It is based on a subset of the JavaScript Programming Language, Standard ECMA-262 3rd Edition - December 1999. JSON is a text format that is completely language independent but uses conventions that are familiar to programmers of the C-family of languages, including C, C++, C#, Java, JavaScript, Perl, Python, and many others. These properties make JSON an ideal data-interchange language.

https://youtu.be/wdoIV_09xAc

Here is an example of JSON data:

```
{
  "name": "Laurence Svekis",
  "age": 41,
  "address": {
    "street": "10 Main St",
    "city": "New York",
    "state": "NY",
    "zip": 10001
  },
  "phoneNumbers": [
    {
      "type": "home",
      "number": "212 123-1234"
    },
    {
      "type": "work",
      "number": "646 123-4567"
    }
  ]
}
```

JavaScript provides methods JSON.stringify() and JSON.parse() to convert between JSON and JavaScript objects.

Example of converting JavaScript object to JSON:

Code Example :

```
const object = { name: 'John Doe', age: 35 };
```

```
const json = JSON.stringify(object);
console.log(json);
Example of converting JSON to JavaScript object:
```

Code Example :

```
const json = '{"name":"John Doe","age":35}';
const object = JSON.parse(json);
console.log(object.name); // "John Doe"
```

In summary, JSON is a lightweight data interchange format that is easy for humans to read and write, and easy for machines to parse and generate. It is based on a subset of JavaScript and can be used with many programming languages. JavaScript provides built-in methods for converting between JSON and JavaScript objects.

There are several ways to get JSON data with JavaScript. One common method is to use the fetch() function to make an HTTP request to a server that returns JSON data. The fetch() function returns a promise that resolves to a response object, from which the JSON data can be extracted using the json() method.

Here is an example of how to get JSON data from a remote server:

```
fetch('https://api.example.com/data')
  .then(response => response.json())
  .then(data => {
    console.log(data);
  })
  .catch(error => {
    console.error('Error:', error);
```

```
  });
```

Another way to get JSON data is to load it from a local file using the XMLHttpRequest object or the fetch() function.

Here is an example of how to get JSON data from a local file:

```
var xhr = new XMLHttpRequest();
xhr.open('GET', 'data.json', true);
xhr.responseType = 'json';
xhr.onload = function() {
  if (xhr.status === 200) {
    console.log(xhr.response);
  }
};
xhr.send();
```

In summary, there are several ways to get JSON data with JavaScript, including using the fetch() function to make an HTTP request to a server that returns JSON data or by loading JSON data from a local file using the XMLHttpRequest object or the fetch() function. Once you have the data you can use json() to access the data.

Laurence Svekis

10 Main St
New York
NY
10001
home - (212 123-1234)
work - (646 123-4567)
work 2 - (343 133-4567)
{"name":"Laurence Svekis","age":41,"address":{"street":"10 Main St","city":"New York","state":"NY","zip":10001},"phoneNumbers":[{"type":"home","number":"212 123-1234"},{"type":"work","number":"646 123-4567"},{"type":"work 2","number":"343 133-4567"}]}

```
<!DOCTYPE html>
<html>
<head>
  <title>Learn JavaScript</title>
</head>
<body>
  <h1>Complete JavaScript Course </h1>
  <div class="output">Data</div>
  <script src="app7.js"></script>
</body>
</html>
```

```
const url = 'my1.json';
const output = document.querySelector('.output');
const dataSt = '{"name":"Laurence
Svekis","age":41,"address":{"street":"10 Main St","city":"New
York","state":"NY","zip":10001},"phoneNumbers":[{"type":"home","number
```

```
":"212 123-1234"},{"type":"work","number":"646
123-4567"},{"type":"work 2","number":"343 133-4567"}]}';
console.log(dataSt);
const dataObj = JSON.parse(dataSt);
console.log(dataObj);

output.addEventListener('click',getJsonData);

function getJsonData(){
  output.textContent = 'loading.....';
  fetch(url)
  .then(response => response.json())
  .then(data => {
     myOutput(data);
  })
  .catch(error => {
     console.error('Error:',error);
  })
}

function myOutput(data){
  let html = `<h1>${data.name}</h1>
<div>${data.address.street}</div>
<div>${data.address.city}</div>
<div>${data.address.state}</div>
<div>${data.address.zip}</div>
`;
data.phoneNumbers.forEach(el =>{
  html += `<small>${el.type} - (${el.number})</small><br>`;
```

```
})
html += JSON.stringify(data);
  output.innerHTML = html;
}
```

```
{
  "name": "Laurence Svekis",
  "age": 41,
  "address": {
     "street": "10 Main St",
     "city": "New York",
     "state": "NY",
     "zip": 10001
  },
  "phoneNumbers": [
     {
        "type": "home",
        "number": "212 123-1234"
     },
     {
        "type": "work",
        "number": "646 123-4567"
     },
     {
        "type": "work 2",
        "number": "343 133-4567"
     }
  ]
```

```
}
```

JavaScript Create Element List

The document.createElement() method in JavaScript is used to create a new HTML element with a specified tag name. The method takes a single argument, which is the tag name of the element to be created. For example, document.createElement("div") creates a new div element. The newly created element can be accessed and modified through the DOM API, such as adding content, attributes, and styles to the element. It can also be added to the document by using methods such as appendChild() or insertAdjacentHTML().

In the below example we will be creating a dynamic list, all the elements are created using JavaScript, adding the button for interaction when the user wants to add new people to the list.


```
const myArr = ['Laurence','Susan','Lisa'];
const output = document.querySelector('.output');
```

```
const btn = document.createElement('button');
btn.textContent = 'Add Person';
output.append(btn);

const myInput = document.createElement('input');
myInput.setAttribute('type','text');
myInput.value = 'Lawrence';
output.prepend(myInput);

const ul = document.createElement('ul');
output.append(ul);
build();

btn.addEventListener('click',addPerson);

function addPerson(){
  const newPerson = myInput.value;
  myArr.push(newPerson);
  adder(newPerson);
  console.log(myArr);
}

function adder(person){
  const li = document.createElement('li');
  li.textContent = person;
  ul.append(li);
}

function build(){
```

```
  myArr.forEach(ele => {
    adder(ele);
  })
}
```

Create an interactive table list of item object values from a JavaScript array.

Create a list of items within a table using JavaScript. Data is contained within an array with object values.

```
<!DOCTYPE html>
<html>
<head>
  <title>Learn JavaScript</title>
  <style>
    table{
      width:100%;
    }
    td:first-child{
      width:10%;
```

```
        }
        td:last-child{
            width:10%;
        }
        td{
            border: 1px solid #ddd;
        }
    </style>
</head>
<body>
    <h1>Learn JavaScript Course </h1>
    <div>
        <input type="text" id="addFriend" >
        <input type="button" id="addNew" value="Add New">
        <div class="output"></div>
    </div>
    <script src="app10.js"></script>
</body>
</html>

const myArr = [
    {name:'Laurence',score:0,id:1} ,
    {name:'Susan',score:0,id:2} ,
    {name:'Lisa',score:0,id:3}
];
const output = document.querySelector('.output');
const btn = document.querySelector('#addNew');
const addFriend = document.querySelector('#addFriend');
const tblOutput = document.createElement('table');
output.append(tblOutput);
```

```
addFriend.value = 'Laurence';
build();

btn.addEventListener('click',()=>{
   const myObj =   {name:addFriend.value,score:0,id:myArr.length+1} ;
   myArr.push(myObj );
   console.log(myArr);
   build();
})

function build(){
   tblOutput.innerHTML = '';
   myArr.forEach((ele,ind) =>{
      const tr = document.createElement('tr');
      tblOutput.append(tr);
      const td1 = document.createElement('td');
      td1.textContent = ele.id;
      tr.append(td1);
      const td2 = document.createElement('td');
      td2.textContent = ele.name;
      tr.append(td2);
      const td3 = document.createElement('td');
      td3.textContent = ele.score;
      tr.append(td3);
      tr.addEventListener('click',()=>{
         ele.score++;
         td3.textContent = ele.score;
      })
   })
```

```
}
```

How to Create Page Elements with JavaScript

How to Create Page Elements and make them Interactive with Event LIsteners

There are several ways to create page elements with JavaScript, including:

Using the document.createElement() method, which creates a new element with the specified tag name. For example, the following code creates a new div element:

```
let newDiv = document.createElement("div");
```

Using the innerHTML property to add HTML content to an existing element. For example, the following code adds a new p element to an existing div element with an id of "container":

```
let container = document.getElementById("container");
container.innerHTML += "<p>Hello World</p>";
```

Using the appendChild() method to add a new element as a child of an existing element. For example, the following code adds a new p element as a child of an existing div element with an id of "container":

```
let container = document.getElementById("container");
let newP = document.createElement("p");
```

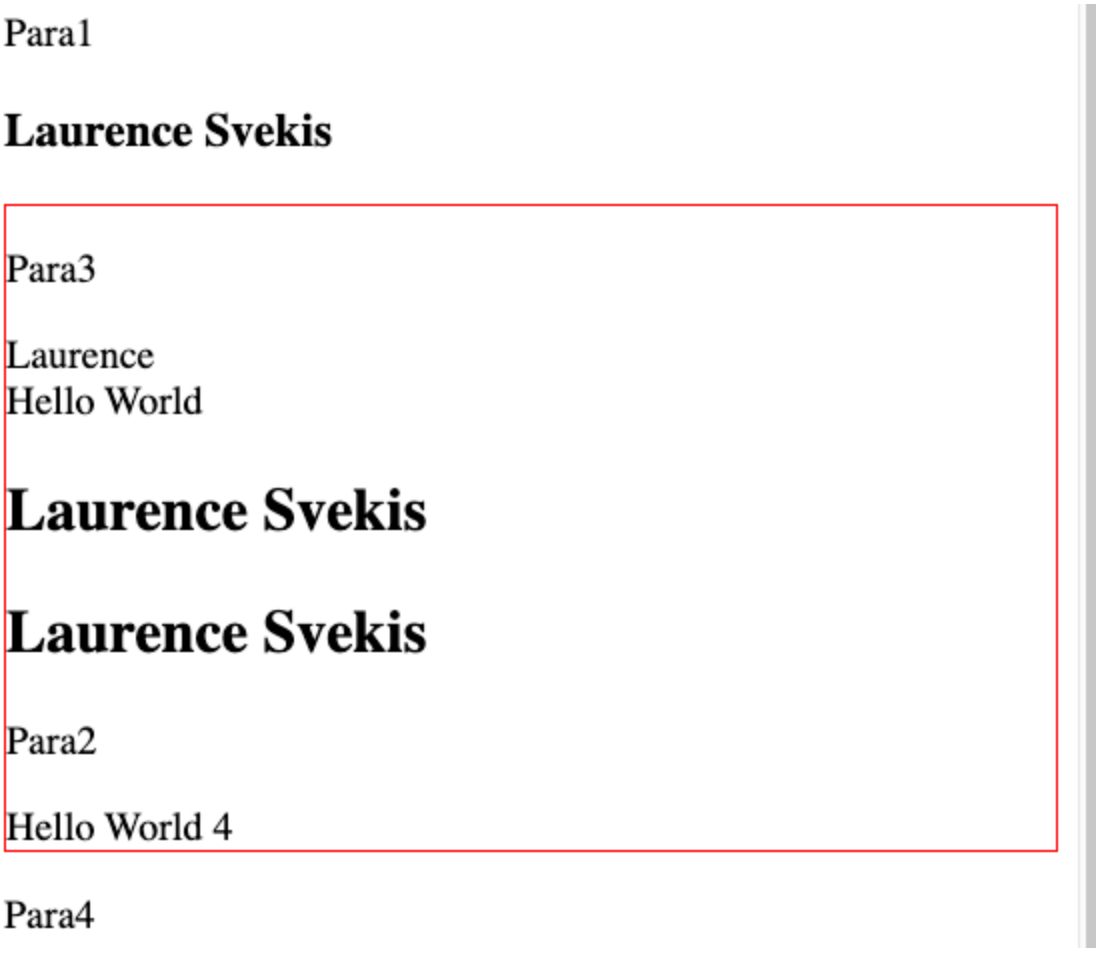
```
newP.innerHTML = "Hello World";
container.appendChild(newP);
```

Using the insertAdjacentHTML() method to insert HTML content at a specific position relative to an existing element. For example, the following code adds a new p element before an existing div element with an id of "container":

```
let container = document.getElementById("container");
container.insertAdjacentHTML("beforebegin", "<p>Hello World</p>");
```

You can also use any of the above methods to add CSS styles, classes and attributes to the newly created elements.

Coding Example of how to insert page content , html elements into your DOM page.




```
const ele1 = document.createElement('div');
ele1.textContent = 'My new element';
document.body.prepend(ele1);

const output = document.querySelector('.output');
output.innerHTML += '<div>Laurence</div>';
output.innerHTML += '<div>Hello World</div>';
output.style.border = '1px solid red';

const ele2 = document.createElement('h2');
ele2.innerHTML = 'Laurence Svekis';
const el = output.appendChild(ele2);
console.log(el);
```

```
const ele3 = document.createElement('h2');
ele3.innerHTML = 'Laurence Svekis';
const el2 = output.append(ele3);
console.log(el2);

output.insertAdjacentHTML('beforebegin','<p>Para1</p>');
output.insertAdjacentHTML('beforeend','<p>Para2</p>');
output.insertAdjacentHTML('afterbegin','<p>Para3</p>');
output.insertAdjacentHTML('afterend','<p>Para4</p>');

const ele4 = document.createElement('h3');
ele4.textContent = 'Laurence Svekis';
output.insertAdjacentElement('beforebegin',ele4);
output.insertAdjacentText('beforeend','Hello World 4');
```