Thread of Execution

JS single-threaded language

Single sequential flow of control

It is a synchronous language with asynchronous capabilities

A thread has a call stack & memory

How Async JS Works

If fetching data from a server, the operations might take a while.

setTimeout & clearTimeout

```
setTimeout(function () {
	console.log('Hello from callback');
},2000);

console.log('Hello from global scope');
// Global will show first no matter what since setTimeout is going to be put on the queue task, waiting for other tasks to be finished before running

function changeText() {
	document.querySelector('h1').textContent = 'Hello from callback';
}
const timerId = setTimeout(changeText, 3000);
document.querySelector('#cancel').addEventListener('click',()=>{
	clearTimeout(timerId)
})
```

setInterval & clearInterval

```
const intervalID = setInterval(myCallback, 1000);

function myCallback(){
	console.log(Date.now());
}

let intervalID;
```

```
function startChange(){
	if(!intervalID){
		intervalID= setInterval(changeColor, 1000);
	}
}
function changeColor() {
	if(document.body.style.backgroundColor !== 'black'){
		document.body.style.backgroundColor = 'black';
		document.body.style.color = 'white';
	}
	else{
		document.body.style.backgroundColor = 'white';
		document.body.style.color = 'black';
	}
}
function stopChange() {
	clearInterval(intervalID);
}
document.getElementByID('start').addEventListener('click', stopChange);
document.getElementByID('stop').addEventListener('click', stopChange);
```

generate radom hex color

```
Math.floor(Math.random() * 16777215).toString(16));
```

Callbacks

```
function toggle(e){
	e.target.classList.toggle('danger');
}

document.querySelector('button').addEventListener('click', toggle);

const posts = [
{title: 'Post One', body: 'This is post one'},
{title: 'Post Two', body: 'This is post two'}
]

function createPost(post, cb) {
	setTimeout(()=>{
		posts.push(post);
		cb();
	}, 2000);
}
```

```
function getPosts() {
        setTimeout(() => {
                posts.forEach(function (post) {
                        const div = document.createElement('div');
                        div.innerHTML = `<strong>${post.title}</strong> - ${post.body}`;
                        document.querySelector('#posts').appendChild(div);
                });
        }, 1000);
}

createPost({title: 'Post Three', body: 'This is post three'}, getPosts);
```

HTTP Requests

GET	**fetch/retrieve data from the server**
POST	send data to the server
PULL & PATCH	update data on a server
DELETE	delete data from a server

```
const xhr = new XMLHttpRequest();

xhr.open('GET','./movies.json');

// readyState
// 0 - request not initialized
// 1 - server connection established
// 2 - request received
// 3 - processing request
// 4 - request finished and response is ready
xhr.onreadystatechange = function () {
   if(this.readyState === 4 && this.status === 200){
      // console.log(JSON.parse(this.responseText));
      const data = JSON.parse(this.responseText);

      data.forEach(movie => {
         const li = document.createElement('li');
         li.innerHTML = `<strong>${movie.title}</strong> - ${movie.year}`;
         document.querySelector('#results').appendChild(li);
      });
   }
}
```

```
xhr.send();
```

Callback Hell

```
function getData(endpoint,cb) {
    const xhr = new XMLHttpRequest();

    xhr.open('GET',endpoint);
    xhr.onreadystatechange = function() {
        if(this.readyState === 4 && this.status === 200){
            cb(JSON.parse(this.responseText));
        }
    }

    setTimeout(()=>{
        xhr.send();
    }, Math.floor(Math.random()*3000)+1000);
}
getData('./movies.json', (data)=>{
    console.log(data);
    getData('./actors.json', (data)=>{
        console.log(data);
        getData('./directors.json', (data)=>{
            console.log(data);
        });
    });
});
```

Promises

```
// Create a promise
const promise = new Promise((resolve, reject) => {
    //Do some async task
    setTimeout(() =>{
        console.log('Async task complete');
        resolve();
    },1000)
});

promise.then(()=>{
    console.log('Promise consumed...')
});
```

```
const getUser = new Promise((resolve, reject) => {
    //Do some async task
    setTimeout(() =>{
        let error = false;
        if(!error){
            resolve({name:'John', age:30});
        }
        else{
            reject('Error: Something went wrong');
        }
    },1000)
});

getUser
    .then((user)=>console.log(user))
    .catch((error)=>console.log(error))
    .finally(()=>console.log('The promise has been resolved or rejected'));

console.log('Hello from global scope');
```

Callback to Promise Refactor

```
function createPost(post){
    return new Promise((resolve, reject)=>{
        setTimeout(()=>{
            let error = false;
            if(!error){
                posts.push(post);
                resolve()
            }
            else{
                reject('Something went wrong...');
            }
        }, 2000);
    });
}

function getPosts(){
    setTimeout(()=>{
        posts.forEach(function (post) {
            const div = document.createElement('div');
            div.innerHTML = `<strong>${post.title}</strong> - ${post.body}`;
            document.querySelector('#posts').appendChild(div);
        })
    }, 1000);
```

```
}

function showError(error){
    const h3 = document.createElement('h3');
    h3.innerHTML= `<strong>${error}</strong>`;
    document.getElementById('posts').appendChild(h3);
}

createPost({title: 'Post Three', body:'This is post 3'})
    .then(getPosts)
    .catch(showError);
```

Promise Chaining

```
const promise = new Promise((resolve, reject) =>{
    setTimeout(()=>{
        let error = false;
        if(!error){
            resolve({name:'John', age:30});
        }
        else{
            reject('Error: Something went wrong');
        }
    }, 1000);
});

promise
    .then((user)=>{
        console.log(user);
        return user.name;
    })
    .then((name)=>{
        console.log(name);
        return name.length;
    })
    .then((length)=>{
        console.log(length);
    })
    .catch((error)=> {
        console.log(error)})
        return 123
    .then((ca)=>{
        console.log(ca);
    })
```

promise.all()

```
function getData(endpoint) {
    return new Promise((resolve, reject)=>{

        const xhr = new XMLHttpRequest();

        xhr.open('GET',endpoint);
        xhr.onreadystatechange = function() {
            if(this.readyState === 4){
                if(this.status === 200){
                    resolve(JSON.parse(this.responseText));
                }
                else{
                    reject('Error: Something went wrong');
                }
            }
        }

        setTimeout(()=>{
            xhr.send();
        }, Math.floor(Math.random()*3000)+1000);
    });
}
const moviesPromise = getData('./movies.json');
const actorsPromise = getData('./actors.json');
const directorsPromise = getData('./directors.json');

const dummyPromise = new Promise((resolve,reject)=>{
    resolve('Hello world');
});

Promise.all([moviesPromise, actorsPromise, directorsPromise, dummyPromise])
    .then((data)=>{
        console.log(data);
    })
    .catch((error)=>console.log(error));
```

Fetch API

```
// Fetching a JSON file
fetch('./movies.json')
    .then((response)=>{
    return response.json();
    })
```

```
    .then((data)=>{
        console.log(data);
    })

// Fetching a txt file
fetch('./test.text')
    .then((response)=>response.text())
    .then((data)=>console.log(data));

// Fetching from an API
fetch('<https://api.github.com/users>')
    .then((response)=>response.json())
    .then((data)=>(document.querySelector('h1').textContent = data.login));
```

Fetch Options

```
function createPost({title, body}){
    fetch('<https://jsonplaceholder.typicode.com/posts>',{
        method: 'POST',
        body: JSON.stringify({
            title,
            body
        }),
        headers:{
            'Content-Type':'application/json',
            token: 'abc123'
        }
    })
    .then(res => res.json)
    .then(data => console.log(data));

}
createPost({title:'My Post', body:'This is my post'})
```