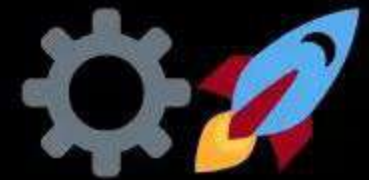


NAVIGATING THE JAVASCRIPT JUNGLE: A GUIDE TO ASYNC PROGRAMMING



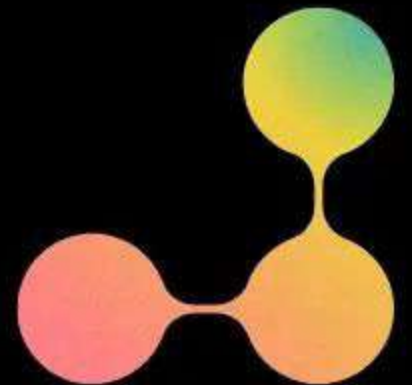
Tips for a async programming





PROMISES PRIMER

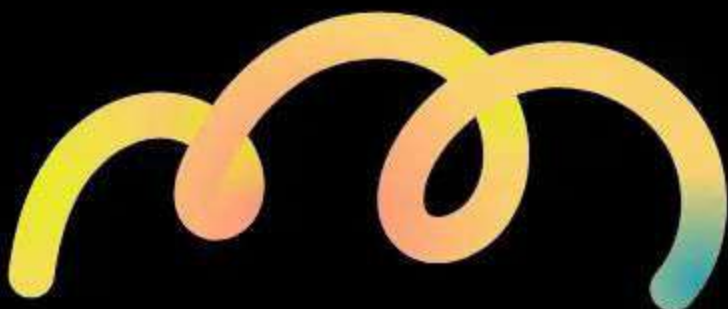
```
const fetchData = () => {  
  return new Promise((resolve, reject) => {  
    // Async operation  
    if (/* operation successful */) {  
      resolve(data);  
    } else { reject(error); }  
  });  
};
```





ASYNC/AWAIT ADVENTURES

```
const fetchData = async () => {  
  try {  
    const data = await fetch('api/data');  
    // Handle data }  
  catch (error) {  
    // Handle error  
  }  
};
```





CALLBACK CONUNDRUM

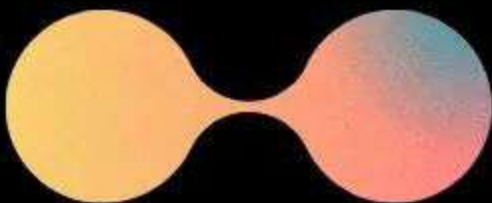
```
fetchData((data, error) => {  
  if (error) {  
    // Handle error  
  } else {  
    // Handle data  
  }  
});
```





CHAINING PROMISES BRILLIANCE

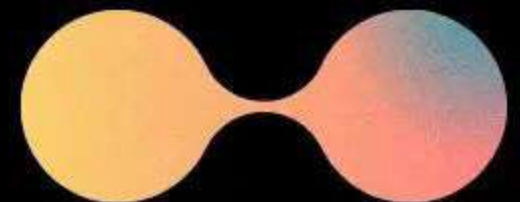
```
fetchData()  
  .then(response => response.json())  
  .then(data => {  
    // Handle data })  
  .catch(error => {  
    // Handle error });
```

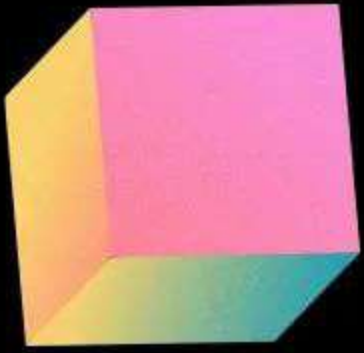




PARALLEL PROMISES POWER

```
Promise.all([fetchData1(), fetchData2()])  
  .then(results => {  
    // Handle results })  
  .catch(error => {  
    // Handle error });
```





**IF YOU LIKE THE
CONTENT,
DON'T FORGET TO
LIKE AND FOLLOW**

Tips for a async programming

