JavaScript Coding Examples

Block Scope Declare Variables

```
Laurence Laurence Laurence
```

```
Mike Jane Mike
```

```
Mike Jane Laurence
```

```
var user1 = 'Laurence';
let user2 = 'Laurence';
const user3 = 'Laurence';
console.log(`${user1} ${user2} ${user3}`);
if(true){
  user2 = 'Jane';
  var user1 = 'Mike';
  //let user2 ='Mike';
```

```
  const user3 = 'Mike';
  //user3 = 'Jane';
  console.log(`${user1} ${user2} ${user3}`);
}
//user2 = 'Laurence';
//user3 = 'Laurence';
console.log(`${user1} ${user2} ${user3}`);
```

For of vs in iterable Items

Use the ”**for in statement”** for object property names, and index values.
Use the **“for of statement”** for iterable items like arrays to return the values contained within the items.

```
const arr = [1,2,3,4,5];
const str = 'Laurence';
const obj = {first:'Laurence',last:'Svekis'};

for(let val in str){
  console.log(str[val]);
}

for(const letter of str){
  //console.log(letter);
}

((...arg)=>{
  for(const val of arg){
     //console.log(val);
  }
```

```
})(1,2,3,4);



(function(){
  for(const val of arguments){
    // console.log(val);
  }
})(1,2,3,4,5,6,7,8);




for(let i=0;i<arr.length;i++){
  //console.log(arr[i]);
}
for (const val of arr){
 // console.log(val);
}

for (let val of arr){
  val++;
 // console.log(val);
}
```

JavaScript Countdown using Interval


```
    <div class="output"></div>
    <button>Start</button>
```

```
const btn = document.querySelector('button');
const output = document.querySelector('.output');
output.style.color = 'black';

btn.addEventListener('click',()=>{
  let counter = 11;
    const countdown = setInterval(()=>{
    counter--;
    output.innerText = counter;
    output.style.color = 'red';
    if(counter < 1){
        output.style.color = 'black';
        output.innerText = 'Blast Off!';
        clearInterval(countdown);
    }
  },1000);
})
```

JavaScript Object Literals

```
const obj1 = new Object();
obj1.first = 'Laurence';
obj1['last'] = 'Svekis';
obj1.fullName = function(){
  return `${this.first} ${this.last}`;
}

const obj2 = {
  first : 'Laurence',
  last : 'Svekis',
  fullName : function(){
     return `${this.first} ${this.last}`;
  },
  full : ()=>`${obj2.first} ${obj2.last}`
};
console.log(obj1.fullName());
console.log(obj2.fullName());
console.log(obj2.full());
```

JavaScript Instance Objects

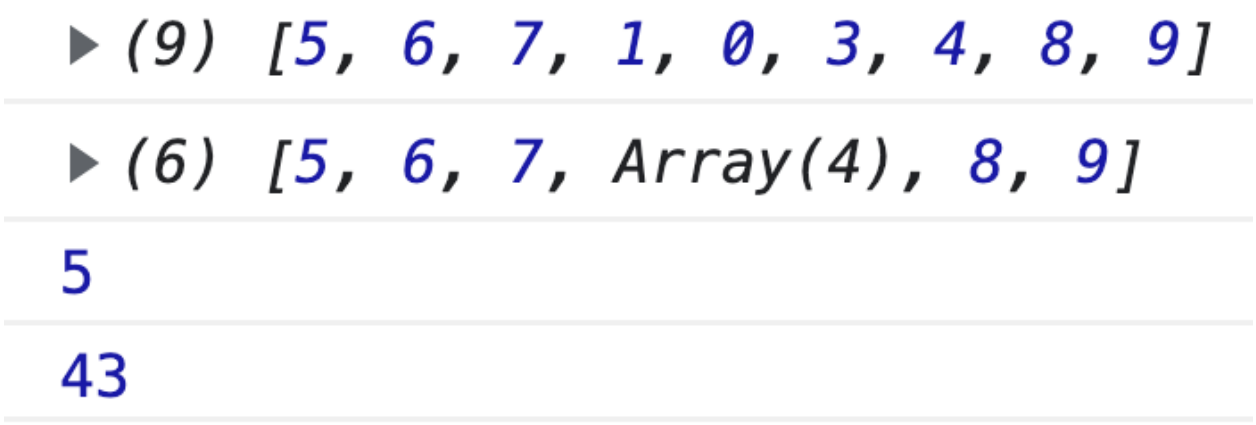
```
const Person = function(){
  let id = Math.floor(Math.random()*1000);
  ///console.log('Function Run');
  this.getId = function(){
     return id;
  }
```

```
  this.makeId = function(){
     id = Math.floor(Math.random()*1000);
     return id;
  }
  this.setId = function(newId){
     id = newId;
     return id;
  }
}

const peop1 = new Person();
const peop2 = new Person();
const holder = [];
for(let i = 0;i<10;i++){
  holder.push(new Person());
  console.log(holder[i].getId());
}

peop1.makeId ();
peop1.setId (55);
console.log(peop1.getId());
```

JavaScript Spread Operator syntax




```
const arr1 = [1,2,3,4];
const arr4 = [...arr1];
arr1[1] = 0;
const arr2 = [5,6,7,...arr1,8,9];
const arr3 = [5,6,7,arr1,8,9];
arr1[2] = 0;
console.log(arr2);
console.log(arr3);

function adder(a,b,c,d){
  return a + b + c + d;
}
function adder1(){
  let total = 0;
  for(const val of arguments){
     total += val;
  }
  return total;
}
```

```
console.log(adder(...arr1));
console.log(adder1(...arr2));
```