

ARCH A4988

# Coding for Spatial Practices



<Sort, Filter & Search>

# Agenda

1. Review
2. Lecture: Sort, Filter & Search
3. Exercise

# Review

# Object Basics

## Iterating through an object

Like arrays, you can use a loop to iterate through an object.

```
let items = { a: 1, b: 2, c: 3 }
```

```
for (let key in items) {  
  console.log( items[key] );  
}
```

```
> 1
```

```
> 2
```

```
> 3
```

# Object Basics

## Iterating over an array of objects

Iterator functions apply a function to each element of an array.

`forEach()` takes a function as an argument and applied the called function to each element of an array.

```
let nums = [  
  { a: 1 },  
  { b: 2 },  
  { c: 3 }  
];  
  
function square (num) {  
  for (let key in num) {  
    console.log(num[key] * num[key])  
  }  
}
```

```
nums.forEach(square)
```

```
> 1  
> 4  
> 9
```

# Object Basics

## Putting array elements in order, sort,

The `sort()` function sorts data lexicographically assuming the data elements are strings.

If you need to sort data elements that are numbers, **you'll have to write an ordering function.**

```
let fruits = ['grapes',  
             'watermelon', 'apple']
```

```
fruits.sort();  
console.log( fruits );
```

```
> apple  
> grapes  
> watermelon
```

# Event Object

# The Event Object

## Accessing the Event Object

How do we gain access to the event object?

First, we need to pass the event object as a parameter.

It is best practice to use the word **event** as the parameter name.

# EVENT OBJECT

Alice was beginning to get very tired of sitting by her sister on the bank, and of having nothing to do. Once or twice she had peeped into the book her sister was reading, but it had no pictures or conversations in it, "and what is the use of a book," thought Alice, "without pictures or conversations?"

So she was considering in her own mind (as well as she could, for the day made her feel very sleepy and stupid), whether the pleasure of making a daisy-chain would be worth the trouble of getting up and picking the daisies, when suddenly a White Rabbit with pink eyes ran close by her.

There was nothing so very remarkable in that, nor did Alice think it so very much out of the way to hear the Rabbit say to itself, "Oh dear! Oh dear! I shall be too late!" But when the Rabbit actually took a watch out of its waistcoat-pocket and looked at it and then hurried on, Alice started to her feet, for it flashed across her mind that she had never before seen a rabbit with either a waistcoat-pocket, or a watch to take out of it, and, burning with curiosity, she ran across the field after it and was just in time to see it pop down a large rabbit-hole, under the hedge. In another moment, down went Alice after it!

[VIEW COMMENTS](#)

```
const viewComments = (event) => {  
  console.log(event)  
}
```

```
anchor.addEventListener('click',  
  viewComments);
```

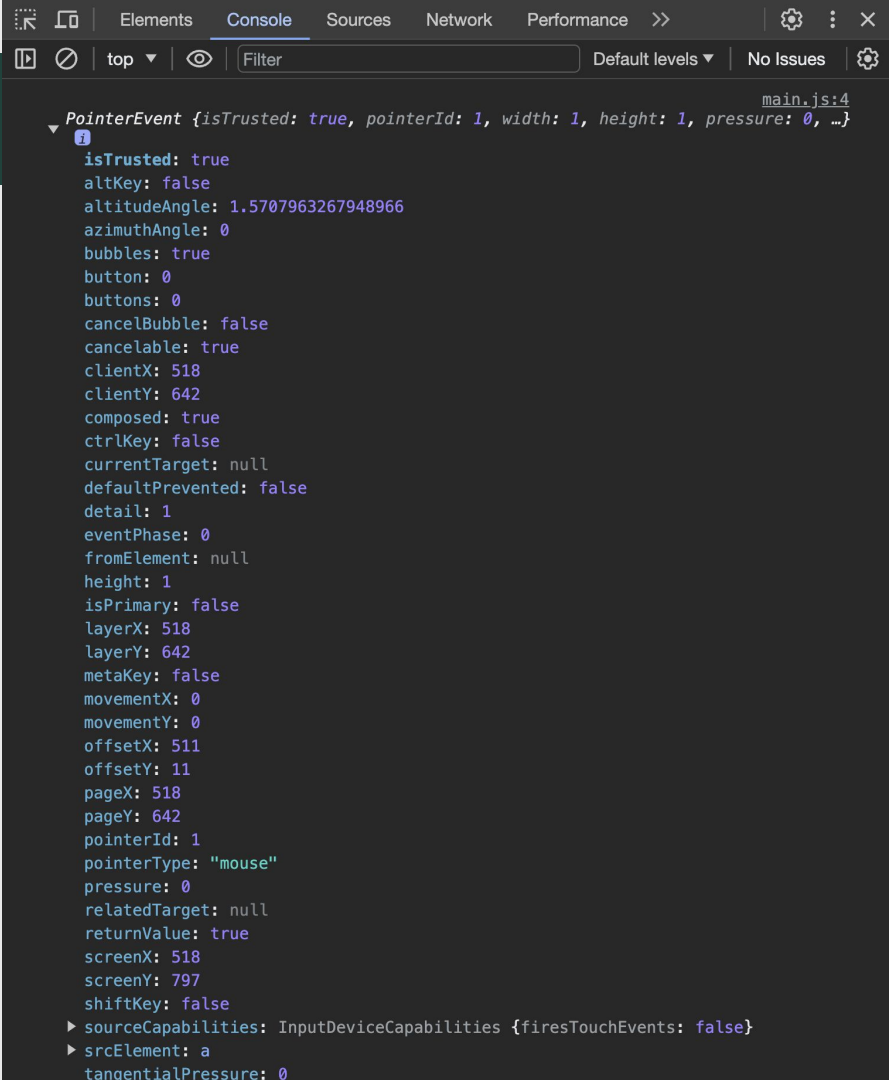


# The Event Object

## Accessing the Event Object

When you click on the View Comments link, you should see in the Developer Console, the event logged to the console.

Don't be overwhelmed by the number of properties!



# The Event Object

## Preventing Default Behavior

Some events, such as *clicking on a link* or *submitting a form*, are meant to take you to another page.

But, maybe you don't want to go to another page. Maybe, you want to fade in some comments.

```
<a href="#">View Comments</a>
```

```
let anchor = document.querySelector('a')
```

```
const viewComments = (event) => {  
  console.log(event)  
  event.preventDefault();  
  let comments =  
    document.querySelector('#comments');  
  comments.className = 'show-comments';  
}
```

```
anchor.addEventListener('click',  
  viewComments);
```

# The Event Object

## Preventing Default Behavior

You'll often use this method when you have *anchors* or *submit* buttons on a page that you want to provide with some JavaScript functionality, instead of having them take you to another page.

## EVENT OBJECT

Alice was beginning to get very tired of sitting by her sister on the bank, and of having nothing to do. Once or twice she had peeped into the book her sister was reading, but it had no pictures or conversations in it, "and what is the use of a book," thought Alice, "without pictures or conversations?"

So she was considering in her own mind (as well as she could, for the day made her feel very sleepy and stupid), whether the pleasure of making a daisy-chain would be worth the trouble of getting up and picking the daisies, when suddenly a White Rabbit with pink eyes ran close by her.

There was nothing so very remarkable in that, nor did Alice think it so very much out of the way to hear the Rabbit say to itself, "Oh dear! Oh dear! I shall be too late!" But when the Rabbit actually took a watch out of its waistcoat-pocket and looked at it and then hurried on, Alice started to her feet, for it flashed across her mind that she had never before seen a rabbit with either a waistcoat-pocket, or a watch to take out of it, and, burning with curiosity, she ran across the field after it and was just in time to see it pop down a large rabbit-hole, under the hedge. In another moment, down went Alice after it, and

[VIEW COMMENTS](#)

---

**Reader #1:** Great read!

---

**Reader #2:** One of my favorite books!

---

**Reader #3:** A rabbit with a watch.

---

# The Event Object

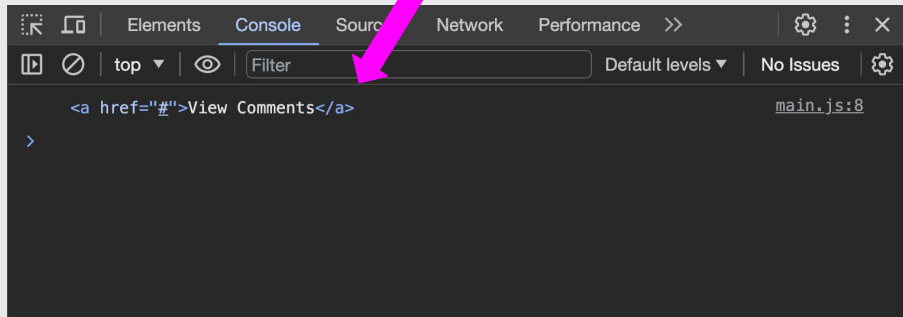
## target

Here we access the target of the event by using dot notation `event.target`. Then we log the target to the console.

Let's take a look at what we see in the console:

```
<a href="#">View Comments</a>
```

```
const viewComments = (event) => {  
  console.log(event.target)  
}
```



# The Event Object

## type

Sometimes, you may need to know what kind of event occurred. Find this using the `type` property.

You should see *The event type is click* in the developer console.

```
<a href="#">View Comments</a>
```

```
document.querySelector('a').addEventListener('click',  
viewComments);
```

```
const viewComments = (event) => {  
  let eventType = event.type;  
  console.log(The event type is:  
    ' + eventType)  
}
```

# Sort, Filter & Search

# Filter Method

## .filter()

The syntax for `filter()` resembles the following:

Here we have an array of numbers. Then, we apply a `filter()` to return all numbers that are greater than 7.

```
let numbers = [1, 3, 6, 8, 11];

const greaterThanSeven =
  numbers.filter(function(number) {
    return number > 7;
  });

console.log(greaterThanSeven);

// output
[8, 11]
```

# Filter Method

## .filter()

A common use case of `filter()` is with an array of objects through their properties.

`filter()` creates a new array with elements that meet the criteria. The original array remains in tact.

```
const words = ['spray', 'limit',  
               'elite', 'exuberant', 'destruction',  
               'present'];
```

```
const result =  
words.filter(function(word) {  
    return word.length > 6;  
});
```

```
console.log(result);
```

```
// output  
["exuberant", "destruction",  
 "present"]
```



# Filter Method

## .filter()

Here we have an array of creature objects. Then, we apply a `filter()` to return all creatures with a habitat that is equal to Ocean.

```
const creatures = [
  {name: "Shark", habitat: "Ocean"},
  {name: "Whale", habitat: "Ocean"},
  {name: "Lion", habitat: "Savanna"},
  {name: "Monkey", habitat: "Jungle"}
];
```

```
const aquaticCreatures =
  creatures.filter(function(creature) {
    return creature.habitat == "Ocean";
  });
```

```
console.log(aquaticCreatures);
```

```
// output
[ {name: "Shark", habitat: "Ocean"},
  {name: "Whale", habitat: "Ocean"} ]
```

# Sort Method

## .sort()

To effectively sort an array of objects we will need the `sort()` method and an accompanying compare function, `compareFn()`.

A compare function helps us to write our logic in the sorting of the array of objects.

```
let names = [  
  {name: 'Sara', age:24},  
  {name: 'John', age:24},  
  {name: 'Jack', age:25}  
];  
  
function compareFn(a, b){  
  // your logic in here  
}  
  
names.sort(compareFn)
```

# Sort Method

## Sorting Numbers

Here, the sort method is used to sort an array of objects by the *age property*.

To compare the *age property* we simply subtract them. If the difference is a *negative value* the order is changed. If it's a *positive value* the order remains the same.

```
let names = [  
  {name: 'Sara', age:24},  
  {name: 'John', age:22},  
  {name: 'Jack', age:27}  
];  
  
function compareAgeFn(a, b){  
  return a.age - b.age;  
}  
  
names.sort(compareAgeFn)  
  
// output  
[{name: "John", age: 22},  
 {name: "Sara", age: 24},  
 {name: "Jack", age: 27}]
```

# Sort Method

## Sorting Strings

Here, the sort method is sorting elements according to values returned by a custom compare function, `compareNameFn`.

If comparing two names results in 1, the order is changed. If comparing names results in -1 or 0, their order remains the same.

```
let names = [  
  {name: 'Sara', age:24},  
  {name: 'John', age:22},  
  {name: 'Jack', age:27}  
];  
  
function compareNameFn(a, b){  
  let comparison = 0;  
  if(a.name > b.name){  
    comparison = 1  
  } else if (a.name < b.name) {  
    comparison = -1  
  }  
  return comparison;  
}
```

```
names.sort(compareNameFn)
```

# Sort Method

## Custom compare function

The custom compare function returns a negative, zero, or positive value, depending on the arguments passed to it.

When the `sort()` function compares two values, it sends the values to the compare function, and sorts the values according to the returned (negative, zero, positive) value by the compare function.

# Search Using .includes()

## .includes()

The `includes()` method determines whether an array contains a certain value, the value for which you are searching.

```
const pets = ['cat', 'dog', 'bat'];  
  
console.log(pets.includes('cat'));  
// output: true  
  
console.log(pets.includes('at'));  
// output: false
```

# Exercises



# Iterate Over an Array of Objects



1. Iterate over this array of objects and print the name to the console.
2. Talk through the steps to render the name to the web page.
3. Now, render the name to the web page.

Instructions:

Go to the **starter\_code\_week\_13** folder and complete the **search\_names** exercise.

Adri  
Becky  
Chris  
Dillon  
Evan  
Frank  
Georgette  
Hugh  
Igor  
Jacoby  
Kristina  
Lemony  
Matilda  
Nile





## Guided Walk-Through: Search for a Name

8 minutes



1. Search for a particular name on the page by typing in the name in the input field.
2. Render the searched name to the web page.

search...

- Adri
- Becky
- Chris
- Dillon
- Evan
- Frank
- Georgette
- Hugh
- Igor
- Jacoby
- Kristina
- Lemony
- Matilda
- Nile



## Solo Exercise: Iterating over an array of objects

15 minutes



1. Iterate over this array of artist objects and print the artist name to the console.
2. Talk through the steps to render the artist name to the web page.
3. Now, render the artist name to the web page.

### **Starter code:**

<https://codepen.io/celestelyne/pen/vYgBbrp>

# Exercise

## Set Up

1. Iterate over this array of objects (in this case they are flowers)
2. Render the flowers to the web page
3. In the navigation bar, create four buttons: Yellow, Red, Blue and All
4. Filter the flowers by color so that clicking the yellow button only shows the yellow flowers.

### Filters

### Flowers

#### Tulip

yellow



#### Daffodil

yellow



#### Sunflower

yellow



#### Bluebell

blue

