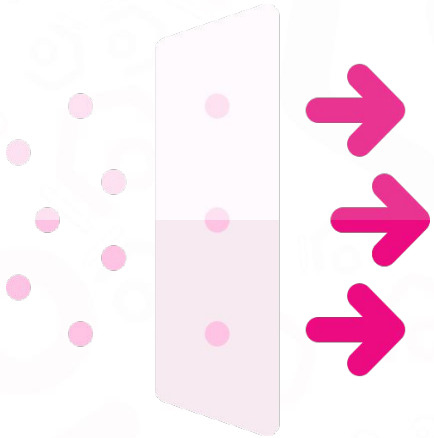
CodingLab
Powered by BK

Bootstrap



Filter

Introduction to Filter

Duration: 30 minutes
Q&A: 5 minutes by the end of the lecture

Filter

When working with arrays, we frequently wish to select a specific subset of the elements; for example, **only the even numbers**, or **only the people over 21.**

The filter abstraction allows us to accomplish this.

CodingLab Powered by RBK

Filter

```
function olderThan30(people) {



}
```

```
var people = [
  {name: “Alyssa”, age: 22},
  {name: “Ben”, age: 36},
  {name: “Lem”, age: 42},
  {name: “Eva”, age: 19},
  {name: “Louis”, age: 33}
];
```

Let’s consider two examples of **filtering** certain elements from an array. In our examples, we’ll use an array of objects representing people to find **the people older than 30** and **the people with three-letter names.**

CodingLab Powered by RBK

Filter

```
function olderThan30(people) {




}
```

```
var people = [
  {name: “Alyssa”, age: 22},
  {name: “Ben”, age: 36},
  {name: “Lem”, age: 42},
  {name: “Eva”, age: 19},
  {name: “Louis”, age: 33}
];
```

Our first example will involve getting just the people with an age greater than thirty.

CodingLab Powered by RBK

Filter

```
function olderThan30(people) {




}
```

```
var people = [
  {name: “Alyssa”, age: 22},
  {name: “Ben”, age: 36},
  {name: “Lem”, age: 42},
  {name: “Eva”, age: 19},
  {name: “Louis”, age: 33}
];

olderThan30(people);
/* =>
[
  {name: “Ben”, age: 36},
  {name: “Lem”, age: 42},
  {name: “Louis”, age: 33}
]
*/
```

If we were to invoke our function `olderThan30` with `people` as an argument, the output we would expect to get is shown in bold in the blue box.

CodingLab Powered by RBK

Filter

```
function olderThan30(people) {

  each          (people, function(person) {


  });

}
```

```
var people = [
  {name: “Alyssa”, age: 22},
  {name: “Ben”, age: 36},
  {name: “Lem”, age: 42},
  {name: “Eva”, age: 19},
  {name: “Louis”, age: 33}
];

olderThan30(people);
/* =>
[
  {name: “Ben”, age: 36},
  {name: “Lem”, age: 42},
  {name: “Louis”, age: 33}
]
*/
```

Our strategy will involve iterating over **each** person in the **people** parameter...

CodingLab Powered by RBK

Filter

```
function olderThan30(people) {

  each          (people, function(person) {
    if    (person.age > 30) {

    }
  });

}
```

```
var people = [
  {name: “Alyssa”, age: 22},
  {name: “Ben”, age: 36},
  {name: “Lem”, age: 42},
  {name: “Eva”, age: 19},
  {name: “Louis”, age: 33}
];

olderThan30(people);
/* =>
[
  {name: “Ben”, age: 36},
  {name: “Lem”, age: 42},
  {name: “Louis”, age: 33}
]
*/
```

...and then checking if each person’s age is greater than thirty. **Q:** If the person’s age is greater than 30, we need to **keep** the `person` object -- what needs to be added to this function so that we can accomplish this?

CodingLab Powered by RBK

Filter

```
function olderThan30(people) {
  var acc = [];
  each          (people, function(person) {
    if    (person.age > 30) {

    }
  });
  return acc;
}
```

```
var people = [
  {name: “Alyssa”, age: 22},
  {name: “Ben”, age: 36},
  {name: “Lem”, age: 42},
  {name: “Eva”, age: 19},
  {name: “Louis”, age: 33}
];

olderThan30(people);
/* =>
[
  {name: “Ben”, age: 36},
  {name: “Lem”, age: 42},
  {name: “Louis”, age: 33}
]
*/
```

A: We need an **accumulator** to store the people that should be kept! First we’ll add a variable called `acc` to represent the accumulator...

CodingLab Powered by RBK

Filter

```
function olderThan30(people) {
  var acc = [];
  each          (people, function(person) {
    if    (person.age > 30) {
      acc.push(person);
    }
  });
  return acc;
}
```

```
var people = [
  {name: “Alyssa”, age: 22},
  {name: “Ben”, age: 36},
  {name: “Lem”, age: 42},
  {name: “Eva”, age: 19},
  {name: “Louis”, age: 33}
];

olderThan30(people);
/* =>
[
  {name: “Ben”, age: 36},
  {name: “Lem”, age: 42},
  {name: “Louis”, age: 33}
]
*/
```

A: ...and then we’ll use `push` to store the matching `person` object into the accumulator.

CodingLab Powered by RBK

Filter

```
function olderThan30(people) {
  var acc = [];
  each          (people, function(person) {
    if    (person.age > 30) {
      acc.push(person);
    }
  });
  return acc;
}
function threeLetterNames(people) {




}
```

```
var people = [
  {name: “Alyssa”, age: 22},
  {name: “Ben”, age: 36},
  {name: “Lem”, age: 42},
  {name: “Eva”, age: 19},
  {name: “Louis”, age: 33}
];
```

Let’s now tackle the problem of filtering **just the people with three-letter names.**

CodingLab Powered by RBK

Filter

```
function olderThan30(people) {
  var acc = [];
  each          (people, function(person) {
    if    (person.age > 30) {
      acc.push(person);
    }
  });
  return acc;
}
function threeLetterNames(people) {




}
```

```
var people = [
  {name: “Alyssa”, age: 22},
  {name: “Ben”, age: 36},
  {name: “Lem”, age: 42},
  {name: “Eva”, age: 19},
  {name: “Louis”, age: 33}
];

threeLetterNames(people);
/* =>
[
  {name: “Ben”, age: 36},
  {name: “Lem”, age: 42},
  {name: “Eva”, age: 19}
]
*/
```

As before, the desired result of invoking `threeLetterNames` on the `people` array is shown in the blue box.

CodingLab Powered by RBK

Filter

```
function olderThan30(people) {
  var acc = [];
  each          (people, function(person) {
    if    (person.age > 30) {
      acc.push(person);
    }
  });
  return acc;
}
function threeLetterNames(people) {
  var acc = [];

  return acc;



}
```

```
var people = [
  {name: “Alyssa”, age: 22},
  {name: “Ben”, age: 36},
  {name: “Lem”, age: 42},
  {name: “Eva”, age: 19},
  {name: “Louis”, age: 33}
];

threeLetterNames(people);
/* =>
[
  {name: “Ben”, age: 36},
  {name: “Lem”, age: 42},
  {name: “Eva”, age: 19}
]
*/
```

Based on the `olderThan30` function, we know that our function will need an **accumulator** to store the matching people...

CodingLab Powered by RBK

Filter

```
function olderThan30(people) {
  var acc = [];
  each          (people, function(person) {
    if    (person.age > 30) {
      acc.push(person);
    }
  });
  return acc;
}
function threeLetterNames(people) {
  var acc = [];
  each          (people, function(person) {
  });


  return acc;
}
```

```
var people = [
  {name: “Alyssa”, age: 22},
  {name: “Ben”, age: 36},
  {name: “Lem”, age: 42},
  {name: “Eva”, age: 19},
  {name: “Louis”, age: 33}
];

threeLetterNames(people);
/* =>
[
  {name: “Ben”, age: 36},
  {name: “Lem”, age: 42},
  {name: “Eva”, age: 19}
]
*/
```

...we also know that our function will need to iterate over all of the `people` to determine which ones should be kept...

CodingLab Powered by RBK

Filter

```
function olderThan30(people) {
  var acc = [];
  each          (people, function(person) {
    if    (person.age > 30) {
      acc.push(person);
    }
  });
  return acc;
}
function threeLetterNames(people) {
  var acc = [];
  each          (people, function(person) {
    if    (                          ) {
      acc.push(person);
    }
});
  return acc;
}
```

```
var people = [
  {name: “Alyssa”, age: 22},
  {name: “Ben”, age: 36},
  {name: “Lem”, age: 42},
  {name: “Eva”, age: 19},
  {name: “Louis”, age: 33}
];

threeLetterNames(people);
/* =>
[
  {name: “Ben”, age: 36},
  {name: “Lem”, age: 42},
  {name: “Eva”, age: 19}
]
*/
```

...and finally, we can also infer that we'll need an **`if`** statement to determine if the `person` should be added to the accumulator.

CodingLab Powered by RBK

Filter

```
function olderThan30(people) {
  var acc = [];
  each          (people, function(person) {
    if    (person.age > 30) {
      acc.push(person);
    }
  });
  return acc;
}
function threeLetterNames(people) {
  var acc = [];
  each          (people, function(person) {
    if    (         ??????????       ) {
      acc.push(person);
    }
});
  return acc;
}
```

```
var people = [
  {name: “Alyssa”, age: 22},
  {name: “Ben”, age: 36},
  {name: “Lem”, age: 42},
  {name: “Eva”, age: 19},
  {name: “Louis”, age: 33}
];

threeLetterNames(people);
/* =>
[
  {name: “Ben”, age: 36},
  {name: “Lem”, age: 42},
  {name: “Eva”, age: 19}
]
*/
```

Q: What condition should we use to determine if this person should be kept?

CodingLab Powered by RBK

Filter

```
function olderThan30(people) {
  var acc = [];
  each          (people, function(person) {
    if    (person.age > 30) {
      acc.push(person);
    }
  });
  return acc;
}
function threeLetterNames(people) {
  var acc = [];
  each          (people, function(person) {
    if    (person.name.length === 3) {
      acc.push(person);
    }
});
 return acc;
}
```

```
var people = [
  {name: “Alyssa”, age: 22},
  {name: “Ben”, age: 36},
  {name: “Lem”, age: 42},
  {name: “Eva”, age: 19},
  {name: “Louis”, age: 33}
];

threeLetterNames(people);
/* =>
[
  {name: “Ben”, age: 36},
  {name: “Lem”, age: 42},
  {name: “Eva”, age: 19}
]
*/
```

A: If the person’s name has a `length` of three, we want to keep that `person`.

CodingLab Powered by RBK

Filter

```
function olderThan30(people) {
  var acc = [];
  each          (people, function(person) {
    if    (person.age > 30) {
      acc.push(person);
    }
  });
  return acc;
}
function threeLetterNames(people) {
  var acc = [];
  each          (people, function(person) {
    if    (person.name.length === 3) {
      acc.push(person);
    }
});
 return acc;
}
```

```
var people = [
  {name: “Alyssa”, age: 22},
  {name: “Ben”, age: 36},
  {name: “Lem”, age: 42},
  {name: “Eva”, age: 19},
  {name: “Louis”, age: 33}
];
```

Now, let’s take a look at these two functions. **Q:** What are the differences and similarities between the two functions?

CodingLab Powered by RBK

Filter

```
function olderThan30(people) {
  var acc = [];
  each          (people, function(person) {
    if    (person.age > 30) {
      acc.push(person);
    }
  });
  return acc;
}
function threeLetterNames(people) {
  var acc = [];
  each          (people, function(person) {
    if    (person.name.length === 3) {
      acc.push(person);
    }
});
 return acc;
}
```

```
var people = [
  {name: “Alyssa”, age: 22},
  {name: “Ben”, age: 36},
  {name: “Lem”, age: 42},
  {name: “Eva”, age: 19},
  {name: “Louis”, age: 33}
];
```

A: The **only** difference between the two is the ***condition*** that we check! Let’s see what we can do to extract this pattern into its own *abstraction*.

CodingLab Powered by RBK

Filter

```
function olderThan30(people) {
  var acc = [];
  each          (people, function(person) {
    if    (person.age > 30) {
      acc.push(person);
    }
  });
  return acc;
}
function threeLetterNames(people) {
  var acc = [];
  each          (people, function(person) {
    if    (person.name.length === 3) {
      acc.push(person);
    }
});
 return acc;
}
```

```
var people = [
  {name: “Alyssa”, age: 22},
  {name: “Ben”, age: 36},
  {name: “Lem”, age: 42},
  {name: “Eva”, age: 19},
  {name: “Louis”, age: 33}
];

function filter(                    ) {




}
```

First, we'll create a function called **`filter`**. **Q:** What's are some similarities between the `olderThan30` and `threeLetterNames` functions that we can extract into the `filter` function?

CodingLab Powered by RBK

Filter

```
function olderThan30(people) {
  var acc = [];
  each          (people, function(person) {
    if    (person.age > 30) {
      acc.push(person);
    }
  });
  return acc;
}
function threeLetterNames(people) {
  var acc = [];
  each          (people, function(person) {
    if    (person.name.length === 3) {
      acc.push(person);
    }
  });
  return acc;
}
```

```
var people = [
  {name: “Alyssa”, age: 22},
  {name: “Ben”, age: 36},
  {name: “Lem”, age: 42},
  {name: “Eva”, age: 19},
  {name: “Louis”, age: 33}
];

function filter(                    ) {
  var acc = [];









  return acc;
}
```

A: One of these is the **accumulator** variable, acc.

CodingLab Powered by RBK

Filter

```
function olderThan30(people) {
  var acc = [];
  each          (people, function(person) {
    if    (person.age > 30) {
      acc.push(person);
    }
  });
  return acc;
}
function threeLetterNames(people) {
  var acc = [];
  each          (people, function(person) {
    if    (person.name.length === 3) {
      acc.push(person);
    }
  });
  return acc;
}
```

```
var people = [
  {name: “Alyssa”, age: 22},
  {name: “Ben”, age: 36},
  {name: “Lem”, age: 42},
  {name: “Eva”, age: 19},
  {name: “Louis”, age: 33}
];

function filter(array           ) {
  var acc = [];




  return acc;
}
```

A: We’re also going to need to iterate over an array, which means that we need an array parameter...

CodingLab Powered by RBK

Filter

```
function olderThan30(people) {
  var acc = [];
  each          (people, function(person) {
    if    (person.age > 30) {
      acc.push(person);
    }
  });
  return acc;
}
function threeLetterNames(people) {
  var acc = [];
  each          (people, function(person) {
    if    (person.name.length === 3) {
      acc.push(person);
    }
  });
  return acc;
}
```

```
var people = [
  {name: “Alyssa”, age: 22},
  {name: “Ben”, age: 36},
  {name: “Lem”, age: 42},
  {name: “Eva”, age: 19},
  {name: “Louis”, age: 33}
];

function filter(array           ) {
  var acc = [];
  each(array, function(element) {



  });
  return acc;
}
```

A: ...that we can then perform some iteration over using each. **Q:** What else is similar?

CodingLab Powered by RBK

Filter

```
function olderThan30(people) {
  var acc = [];
  each          (people, function(person) {
    if    (person.age > 30) {
      acc.push(person);
    }
  });
  return acc;
}
function threeLetterNames(people) {
  var acc = [];
  each          (people, function(person) {
    if    (person.name.length === 3) {
      acc.push(person);
    }
  });
  return acc;
}
```

```
var people = [
  {name: “Alyssa”, age: 22},
  {name: “Ben”, age: 36},
  {name: “Lem”, age: 42},
  {name: “Eva”, age: 19},
  {name: “Louis”, age: 33}
];

function filter(array           ) {
  var acc = [];
  each(array, function(element) {
    if (                    ) {

    }
  });
  return acc;
}
```

A: Next, we see that we always check a condition using the `if` statement...

CodingLab Powered by RBK

Filter

```
function olderThan30(people) {
  var acc = [];
  each          (people, function(person) {
    if    (person.age > 30) {
      acc.push(person);
    }
  });
  return acc;
}
function threeLetterNames(people) {
  var acc = [];
  each          (people, function(person) {
    if    (person.name.length === 3) {
      acc.push(person);
    }
  });
  return acc;
}
```

```
var people = [
  {name: “Alyssa”, age: 22},
  {name: “Ben”, age: 36},
  {name: “Lem”, age: 42},
  {name: “Eva”, age: 19},
  {name: “Louis”, age: 33}
];

function filter(array           ) {
  var acc = [];
  each(array, function(element) {
    if (                    ) {
      acc.push(element);
    }
  });
  return acc;
}
```

A: ...and then we always use **push** to add the element into the accumulator if the condition is `true`.

CodingLab Powered by RBK

Filter

```
function olderThan30(people) {
  var acc = [];
  each          (people, function(person) {
    if    (person.age > 30) {
      acc.push(person);
    }
  });
  return acc;
}
function threeLetterNames(people) {
  var acc = [];
  each          (people, function(person) {
    if    (person.name.length === 3) {
      acc.push(person);
    }
  });
  return acc;
}
```

```
var people = [
  {name: “Alyssa”, age: 22},
  {name: “Ben”, age: 36},
  {name: “Lem”, age: 42},
  {name: “Eva”, age: 19},
  {name: “Louis”, age: 33}
];

function filter(array           ) {
  var acc = [];
  each(array, function(element) {
    if (    ??????????    ) {
      acc.push(element);
    }
  });
  return acc;
}
```

Now that we have extracted all of the common parts, we can turn our attention to the **differences** between the two functions. **Q:** What can we use to test if an **`element`** should be added to the accumulator?

CodingLab Powered by RBK

Filter

```
function olderThan30(people) {
  var acc = [];
  each          (people, function(person) {
    if    (person.age > 30) {
      acc.push(person);
    }
  });
  return acc;
}
function threeLetterNames(people) {
  var acc = [];
  each          (people, function(person) {
    if    (person.name.length === 3) {
      acc.push(person);
    }
  });
  return acc;
}
```

```
var people = [
  {name: “Alyssa”, age: 22},
  {name: “Ben”, age: 36},
  {name: “Lem”, age: 42},
  {name: “Eva”, age: 19},
  {name: “Louis”, age: 33}
];

function filter(array, predicate) {
  var acc = [];
  each(array, function(element) {
    if (predicate(element)) {
      acc.push(element);
    }
  });
  return acc;
}
```

A: A function! We call this function a **predicate**, because that’s the term we use to refer to a function that returns either `true` or `false`.

CodingLab Powered by RBK

Filter

```
function olderThan30(people) {
  var acc = [];
  each          (people, function(person) {
    if    (person.age > 30) {
      acc.push(person);
    }
  });
  return acc;
}
function threeLetterNames(people) {
  var acc = [];
  each          (people, function(person) {
    if    (person.name.length === 3) {
      acc.push(person);
    }
  });
  return acc;
}
```

```
var people = [
  {name: “Alyssa”, age: 22},
  {name: “Ben”, age: 36},
  {name: “Lem”, age: 42},
  {name: “Eva”, age: 19},
  {name: “Louis”, age: 33}
];

function filter(array, predicate) {
  var acc = [];
  each(array, function(element) {
    if (predicate(element)) {
      acc.push(element);
    }
  });
  return acc;
}
```

Now, let’s rewrite our two functions using `filter`!
Q: What are some things that we can replace/change?

CodingLab Powered by RBK

Filter

```
function olderThan30(people) {
  var acc = [];
        filter(people, function(person) {
    if    (person.age > 30) {
      acc.push(person);
    }
  });
  return acc;
}
function threeLetterNames(people) {
  var acc = [];
        filter(people, function(person) {
    if    (person.name.length === 3) {
      acc.push(person);
    }
  });
  return acc;
}
```

```
var people = [
  {name: “Alyssa”, age: 22},
  {name: “Ben”, age: 36},
  {name: “Lem”, age: 42},
  {name: “Eva”, age: 19},
  {name: “Louis”, age: 33}
];

function filter(array, predicate) {
  var acc = [];
  each(array, function(element) {
    if (predicate(element)) {
      acc.push(element);
    }
  });
  return acc;
}
```

First, let’s replace **each** with **filter**.

CodingLab Powered by RBK

Filter

```
function olderThan30(people) {

        filter(people, function(person) {
    if    (person.age > 30) {
      acc.push(person);
    }
  });

}
function threeLetterNames(people) {

        filter(people, function(person) {
    if    (person.name.length === 3) {
      acc.push(person);
    }
  });

}
```

```
var people = [
  {name: “Alyssa”, age: 22},
  {name: “Ben”, age: 36},
  {name: “Lem”, age: 42},
  {name: “Eva”, age: 19},
  {name: “Louis”, age: 33}
];

function filter(array, predicate) {
  var acc = [];
  each(array, function(element) {
    if (predicate(element)) {
      acc.push(element);
    }
  });
  return acc;
}
```

Next, let’s get rid of the **acc** variable, because `filter` does the accumulation for us.

CodingLab Powered by RBK

Filter

```
function olderThan30(people) {

        filter(people, function(person) {
    if    (person.age > 30) {

    }
  });

}
function threeLetterNames(people) {

        filter(people, function(person) {
    if    (person.name.length === 3) {
    }
  });
}
```

```
var people = [
  {name: “Alyssa”, age: 22},
  {name: “Ben”, age: 36},
  {name: “Lem”, age: 42},
  {name: “Eva”, age: 19},
  {name: “Louis”, age: 33}
];

function filter(array, predicate) {
  var acc = [];
  each(array, function(element) {
    if (predicate(element)) {
      acc.push(element);
    }
  });
  return acc;
}
```

This also means that we can get rid of `acc.push`, because this is also taken care of by `filter`.

CodingLab Powered by RBK

Filter

```
function olderThan30(people) {

        filter(people, function(person) {
    if    (person.age > 30) {
    }
  });

}
function threeLetterNames(people) {

        filter(people, function(person) {
    if    (person.name.length === 3) {
    }
  });
}
```

```
var people = [
  {name: “Alyssa”, age: 22},
  {name: “Ben”, age: 36},
  {name: “Lem”, age: 42},
  {name: “Eva”, age: 19},
  {name: “Louis”, age: 33}
];

function filter(array, predicate) {
  var acc = [];
  each(array, function(element) {
    if (predicate(element)) {
      acc.push(element);
    }
  });
  return acc;
}
```

Now, notice that we use the **return value** of the **predicate** to determine whether or not the element should be “kept”. **Q:** What change do we need to make to the functions passed to `filter`?

CodingLab Powered by RBK

Filter

```
function olderThan30(people) {

        filter(people, function(person) {
    return person.age > 30;

  });
}

function threeLetterNames(people) {

        filter(people, function(person) {
    return person.name.length === 3;

  });
}
```

```
var people = [
  {name: “Alyssa”, age: 22},
  {name: “Ben”, age: 36},
  {name: “Lem”, age: 42},
  {name: “Eva”, age: 19},
  {name: “Louis”, age: 33}
];

function filter(array, predicate) {
  var acc = [];
  each(array, function(element) {
    if (predicate(element)) {
      acc.push(element);
    }
  });
  return acc;
}
```

Now, notice that we use the **return value** of the **predicate** to determine whether or not the element should be “kept”. **Q:** What change do we need to make to the functions passed to `filter`?

CodingLab Powered by RBK

Filter

```
function olderThan30(people) {

        filter(people, function(person) {
    return person.age > 30;

  });
}

function threeLetterNames(people) {

        filter(people, function(person) {
    return person.name.length === 3;

  });
}
```

```
var people = [
  {name: “Alyssa”, age: 22},
  {name: “Ben”, age: 36},
  {name: “Lem”, age: 42},
  {name: “Eva”, age: 19},
  {name: “Louis”, age: 33}
];

function filter(array, predicate) {
  var acc = [];
  each(array, function(element) {
    if (predicate(element)) {
      acc.push(element);
    }
  });
  return acc;
}
```

A: We need to **`return`** `true` or `false` depending on whether or not the person should be “kept”!

CodingLab Powered by RBK

Filter

```
function olderThan30(people) {

  return filter(people, function(person) {
    return person.age > 30;

  });

}

function threeLetterNames(people) {

  return filter(people, function(person) {
    return person.name.length === 3;
  });
}
```

```
var people = [
  {name: “Alyssa”, age: 22},
  {name: “Ben”, age: 36},
  {name: “Lem”, age: 42},
  {name: “Eva”, age: 19},
  {name: “Louis”, age: 33}
];

function filter(array, predicate) {
  var acc = [];
  each(array, function(element) {
    if (predicate(element)) {
      acc.push(element);
    }
  });
  return acc;
}
```

Finally, in order to get the result of invoking `filter`, we'll need to `return` its result.

CodingLab
Powered by BK

That's it

For Filter