CIS 371 Web Application Programming
TypeScript V

Lecturer: Dr. Yong Zhuang

Recall

- Type Alias vs. Interface

- Class

- Functions as Arguments (to another Fn) Callback function

- Functions as Arguments: High-order functions

  - Array.sort()

  - Array.some()

  - Array.every()

Array.forEach(): inspect all shapes
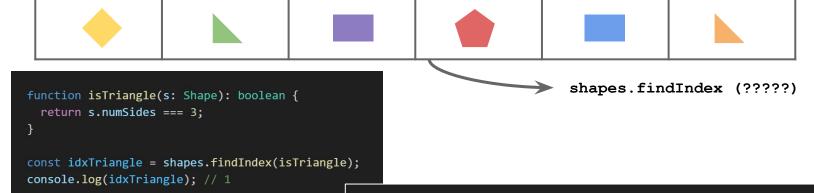

shapes.forEach (?????)

```
function printShape(s: Shape): void {
  console.log("# of sides", s.numSides);
}

shapes.forEach(printShape);
```

- Purpose: Executes a provided function once for each array element.
- Return value: *undefined.*

```
shapes.forEach(function (s: Shape): void {
  console.log("# of sides", s.numSides);
});                        // Anonymous function

shapes.forEach((s: Shape): void => {
  console.log("# of sides", s.numSides);
});                        // Anonymous fat arrow

shapes.forEach((s) => {
  console.log("# of sides", s.numSides);
});                        // No explicit type
```

Array.findIndex(): where is ...?


```typescript
function isTriangle(s: Shape): boolean {
  return s.numSides === 3;
}

const idxTriangle = shapes.findIndex(isTriangle);
console.log(idxTriangle); // 1
```

shapes.findIndex (?????)

- *Purpose: To find the index of the **first element** in the array that satisfies a provided testing function.*
- *Return value: the index of the **first element** in the array that passes the test. If **no elements** pass the test, it returns **-1**.*

```typescript
const idxTriangle = shapes.findIndex(function (s: Shape): boolean {
  return s.numSides === 3;
});

const idxTriangle = shapes.findIndex((s: Shape): boolean => {
  return s.numSides === 3;
});

const idxTriangle = shapes.findIndex((s: Shape): boolean => s.numSides === 3);

const idxTriangle = shapes.findIndex((s) => s.numSides === 3);
```
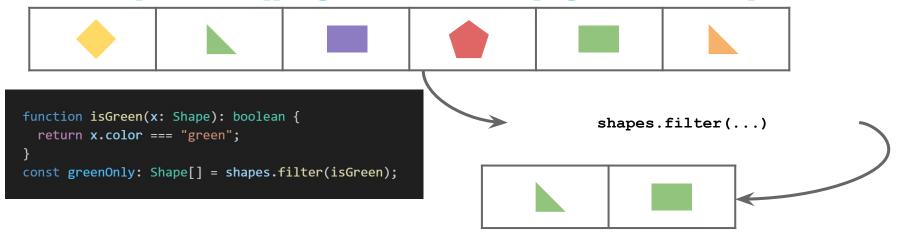
Array find() functions

- If you need the actual element that satisfies a condition in the array, use **find()**.
- If you need the index of the found element in the array that satisfies a condition, use **findIndex()**.
- If you need to find the index of a specific value in the array, use **indexOf()**. (It's similar to findIndex(), but checks each element for equality with the value instead of using a testing function.)
- If you need to determine whether an array includes a specific value, use **includes()**. Again, it checks each element for equality with the value instead of using a testing function.
- If you need to find if any element satisfies the provided testing function, use **some()**.

Array.filter(): give me only green shapes


```
function isGreen(x: Shape): boolean {
  return x.color === "green";
}
const greenOnly: Shape[] = shapes.filter(isGreen);
```
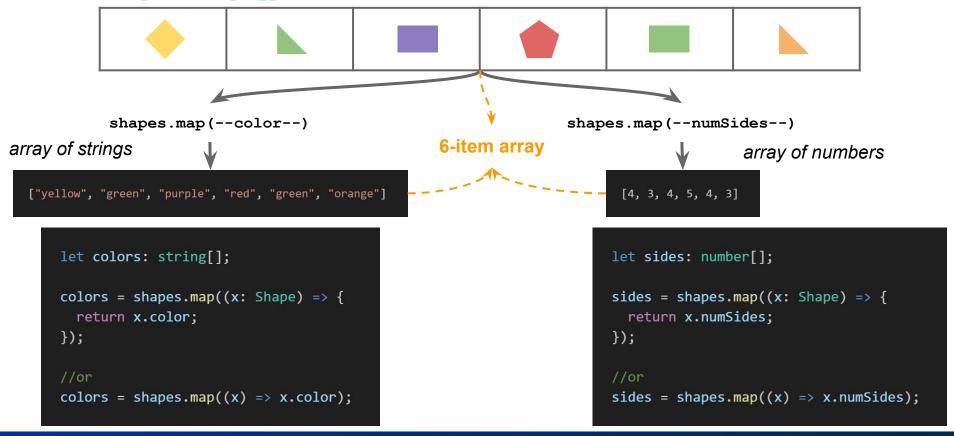
shapes.filter(...)

- *Purpose: creates a new array with all elements that pass the test implemented by the provided function.*
- *Return value: a new array with the elements that pass the test. If no elements pass the test, an empty array will be returned.*

```
const greenOnly = shapes.filter((shp: Shape): boolean => {
  return shp.color === "green";
});

const greenOnly = shapes.filter((x) => x.color === "green");
```

Array.map(): extract all colors/num sides

shapes.map(--color--)

array of strings

6-item array

shapes.map(--numSides--)

array of numbers

```
["yellow", "green", "purple", "red", "green", "orange"]
```

```
[4, 3, 4, 5, 4, 3]
```

```typescript
let colors: string[];

colors = shapes.map((x: Shape) => {
  return x.color;
});

//or
colors = shapes.map((x) => x.color);
```

```typescript
let sides: number[];

sides = shapes.map((x: Shape) => {
  return x.numSides;
});

//or
sides = shapes.map((x) => x.numSides);
```

Array.filter() & Array.map()

.filter()

```typescript
// Named
const numbers = [2, -30, 0, 17, 9, -11];
function isPos(x: number): boolean {
  return x > 0;
}
const out = numbers.filter(isPos);
console.log(out); // [2, 17, 19]
```

```typescript
// Fat arrow
const numbers = [2, -30, 0, 17, 9, -11];
const out = numbers.filter((x: number) => {
  return x > 0;
});
console.log(out); // [2, 17, 19]
```
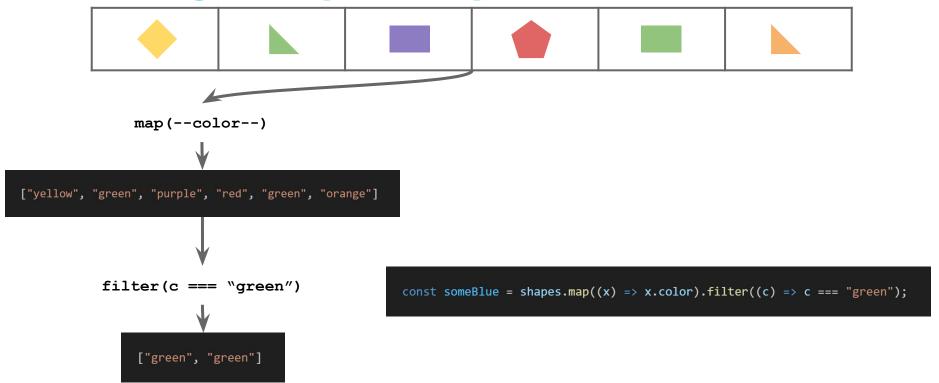
To create a new array with elements that pass a condition (or test) from the original array.

.map()

```typescript
// Function
const numbers = [2, -30, 0, 17, 9, -11];
function plus10(x: number): number {
  return x + 10;
}
const out = numbers.map(plus10);
console.log(out); // [12, -20, 10, 27, 19, -1]
```

```typescript
const numbers = [2, -30, 0, 17, 9, -11];
const out = numbers.map((x: number) => {
  return x + 10;
});
console.log(out); // [12, -20, 10, 27, 19, -1]
```

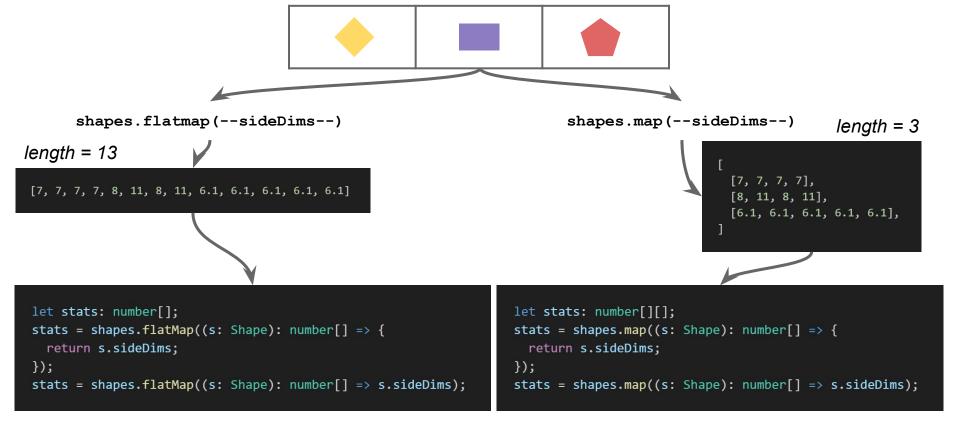*To create a new array by transforming each element in the original array.*

Chaining multiple Array functions

map(--color--)

```
["yellow", "green", "purple", "red", "green", "orange"]
```

filter(c === "green")

```
["green", "green"]
```

```
const someBlue = shapes.map((x) => x.color).filter((c) => c === "green");
```

Array.flatmap(): from one to many

`shapes.flatmap(--sideDims--)`

`shapes.map(--sideDims--)`

length = 3

length = 13

```
[7, 7, 7, 7, 8, 11, 8, 11, 6.1, 6.1, 6.1, 6.1, 6.1]
```

```
[
  [7, 7, 7, 7],
  [8, 11, 8, 11],
  [6.1, 6.1, 6.1, 6.1, 6.1],
]
```

```
let stats: number[];
stats = shapes.flatMap((s: Shape): number[] => {
  return s.sideDims;
});
stats = shapes.flatMap((s: Shape): number[] => s.sideDims);
```

```
let stats: number[][];
stats = shapes.map((s: Shape): number[] => {
  return s.sideDims;
});
stats = shapes.map((s: Shape): number[] => s.sideDims);
```

GRAND VALLEY STATE UNIVERSITY

Practical Use Case of flatmap()

```typescript
type Course = {
  name: string;
  credits: number;
  classList: Array<string>;
};
let allCourses: Array<Course> = [
  {
    name: "MTH101 Calculus",
    credits: 4,
    classList: [
      /* 25 student names */
    ],
  },
  {
    name: "HTM 203 Beer Brewing",
    credits: 2,
    classList: [
      /* 70 student names */
    ],
  },
];
```
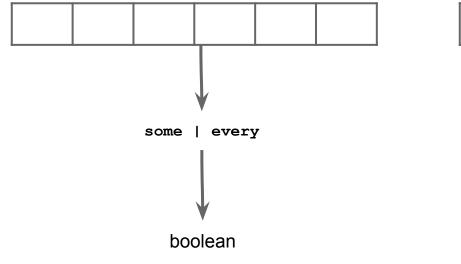
Find all students whose name begins with "Eli"

```typescript
const studentList = allCourses
  .flatMap((c: Course) => {
    return c.classList;
  })
  // you'll get 95 names from flatMap
  .filter((who: string) => {
    return who.startsWith("Eli");
  });
```
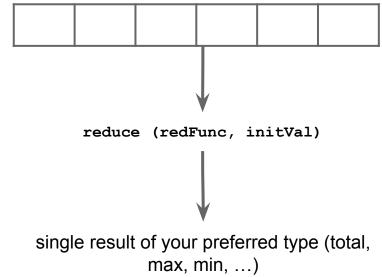
```typescript
const studentList = allCourses
  .flatMap((c: Course) => c.classList)
  // you'll get 95 names from flatMap
  .filter((who: string) => who.startsWith("Eli"));
```

GRAND VALLEY
STATE UNIVERSITY

Introducing: Array.reduce()

```
some | every
```

boolean

```
reduce (redFunc, initVal)
```

single result of your preferred type (total, max, min, …)

Not limited to only boolean output!!!

Array.reduce(): sum of values

```typescript
const scores = [23, -31, 17, 31, 19];
const computeSum = (accumulator: number, currentValue: number): number => {
  return accumulator + currentValue;
};


const totalScore = scores.reduce(computeSum);
console.log("Total ", totalScore); // Total 59
```

pos	accumulator	currentValue	return
1	23	-31	-8
2	-8	17	9
3	9	31	40
4	40	19	59

- *Acc is initialized from the first array item*
- *Work begins at position 1*

Array.reduce(): sum of values (with initial value)

```
const scores = [23, -31, 17, 31, 19];
const computeSum = (accumulator: number, currentValue: number): number => {
  return accumulator + currentValue;
};


const totalScore = scores.reduce(computeSum, 2000);
console.log("Total ", totalScore); // Total 2059
```

pos	accumulator	currentValue	return
0	2000	23	2023
1	2023	-31	1992
2	1992	17	2009
3	2009	31	2040
4	2040	19	2059

- *Acc is initialized from the initial value*
- *Work begins at position 0*

Array.reduce(): shortest river name(with initial value)

```typescript
const rivers = ["Amazon", "Mississippi", "Nile", "YangTze", "Yenisei"];
const shorterOf = (accumulator: string, currentValue: string): string => {
  if (currentValue.length < accumulator.length) return currentValue;
  else return accumulator;
};

const riverName = rivers.reduce(shorterOf, "Yellow");
console.log("Shortest ", riverName); // Nile
```

pos	accumulator	currentValue	return
0	Yellow	Amazon	Yellow
1	Yellow	Mississippi	Yellow
2	Yellow	Nile	Nile
3	Nile	YangTze	Nile
4	Nile	Yenisei	Nile

- *Acc is initialized from the provided value*
- *Work begins at position 0*

Array.reduce(): shortest river name(with initial value)

```
const rivers = ["Amazon", "Mississippi", "Nile", "YangTze", "Yenisei"];
const shorterOf = (accumulator: string, currentValue: string): string => {
  if (currentValue.length < accumulator.length) return currentValue;
  else return accumulator;
};


const riverName = rivers.reduce(shorterOf, "Roe");
console.log("Shortest ", riverName); // ?
```

pos	accumulator	currentValue	return
0	Roe	Amazon	Roe
1	Roe	Mississippi	Roe
2	Roe	Nile	Roe
3	Roe	YangTze	Roe
4	Roe	Yenisei	Roe

GRAND VALLEY
STATE UNIVERSITY

Array.reduce() with initial value

```
const rivers = ["Amazon", "Mississippi", "Nile", "YangTze", "Yenisei"];
const shorterLen = (accumulator: number, currentValue: string): number => {
  if (currentValue.length < accumulator) return currentValue.length;
  else return accumulator;
};
// Use 37 to initialize riverLen
const riverLen = rivers.reduce(shorterLen, 37);
console.log("Shortest ", riverLen); // 4
```

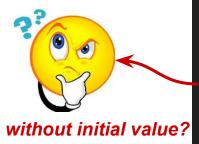| pos	accumulator(num)	currentValue(str)	return(num)
0	37	Amazon	6
1	6	Mississippi	6
2	6	Nile	4
3	4	YangTze	4
4	4	Yenisei	4

- *Type of acc and curr may be different*
- *Type of acc and type of initial value must match*
- *Type of acc determines the type of return*

Array.reduce()

```typescript
let myArray: Array<XYZ>;
```

```typescript
function myFunction(prev: XYZ, curr: XYZ): XYZ {
  // More code here
  return _____;
}
const result: XYZ = myArray.reduce(myFunction);
```

without initial value?

```typescript
function myFunction(prev: resultType, curr: XYZ): resultType {
  // More code here
  return _____;
}
const initValue: resultType = _____;
const result: resultType = myArray.reduce(myFunction, initValue);
```

GRAND VALLEY
STATE UNIVERSITY

Reduce: Array of objects

```
type River = {
  name: string;
  countries: Array<string>; // the river passes thru these countries
  lenInMiles: number; // river length in miles
};
```

```
const waters: Array<River> = [
  {
    name: "Amazon",
    countries: ["Brazil", "Columbia", "Peru"],
    lenInMiles: 4132,
  },
  { name: "Nile", countries: ["Egypt"], lenInMiles: 4388 },
  { name: "Mississippi", countries: ["US"], lenInMiles: 2340 },
  {
    name: "Mekong",
    countries: ["China", "Myanmar", "Laos", "Thailand", "Vietnam"],
    lenInMiles: 2703,
  },
  { name: "Ganges", countries: ["India", "Bangladesh"], lenInMiles: 1560 },
];
```

The name of the longest river?

```typescript
type River = {
  name: string;
  countries: Array<string>;
  lenInMiles: number;
};
```

```typescript
function lengthCompare(prev: River, curr: River): River {
  if (prev.lenInMiles > curr.lenInMiles) return prev;
  else return curr;
}
let winner: River;
winner = waters.reduce(lengthCompare);
console.log(winner.name);
```

```typescript
let winner: River;
winner = waters.reduce((prev: River, curr: River): River => {
  if (prev.lenInMiles > curr.lenInMiles) return prev;
  else return curr;
});
console.log(winner.name);
```

The longest mile?

```typescript
type River = {
  name: string;
  countries: Array<string>;
  lenInMiles: number;
};
```

```typescript
function compLength(prev: River, curr: River): River {
  if (prev.lenInMiles > curr.lenInMiles) return prev;
  else return curr;
}
let winner: River;
winner = waters.reduce(compLength);
console.log("Longest mile is", winner.lenInMiles);
```

```typescript
function compRivLen(prev: number, curr: River): number {
  if (prev > curr.lenInMiles) return prev;
  else return curr.lenInMiles;
}
let winner: number;
winner = waters.reduce(compRivLen, Number.MIN_VALUE);
console.log("Longest mile is", winner);
```

Goes through most countries?

```
type River = {
  name: string;
  countries: Array<string>;
  lenInMiles: number;
};
```

```
function countryCompare(prev: River, curr: River): River {
  if (prev.countries.length > curr.countries.length) return prev;
  else return curr;
}
let winner: River;
winner = waters.reduce(countryCompare);
console.log(winner.name);
```

```
let winner: River;
winner = waters.reduce((prev: River, curr: River): River => {
  if (prev.countries.length > curr.countries.length) return prev;
  else return curr;
});
console.log(winner.name);
```

GRAND VALLEY
STATE UNIVERSITY

Some examples

How many green shapes?

```
shapes.filter((s) => s.color === "green").length;
```

How many equilateral triangles?

```
shapes.filter(
  (s) =>
    s.numSides === 3 &&
    s.sideDims[0] === s.sideDims[1] &&
    s.sideDims[1] === s.sideDims[2]
).length;

shapes
  .filter((s) => s.numSides === 3)
  .filter(
    (s) => s.sideDims[0] === s.sideDims[1] && s.sideDims[1] === s.sideDims[2]
  ).length;
```

Largest perimeter?

```
shapes
  .map((shp) => {
    let perimeter = 0;
    // Compute perimeter
    return perimeter;
  })
  .reduce((acc: number, curr: number) => {
    if (acc > curr) return acc;
    else return curr;
  });
```