# CIS 371 Web Application Programming

## TypeScript V

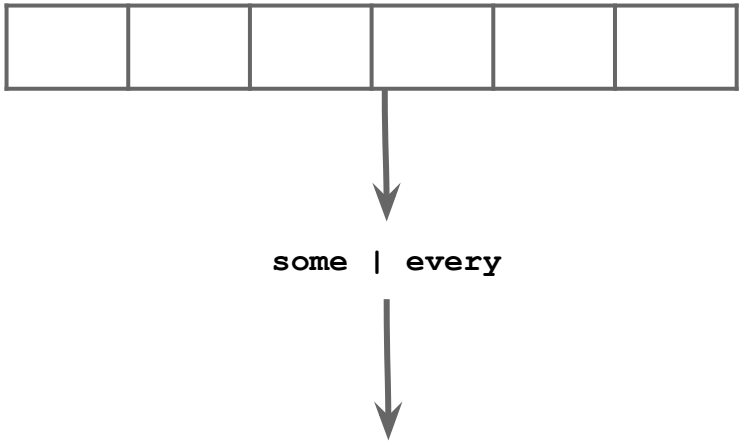

Lecturer: Dr. Yong Zhuang

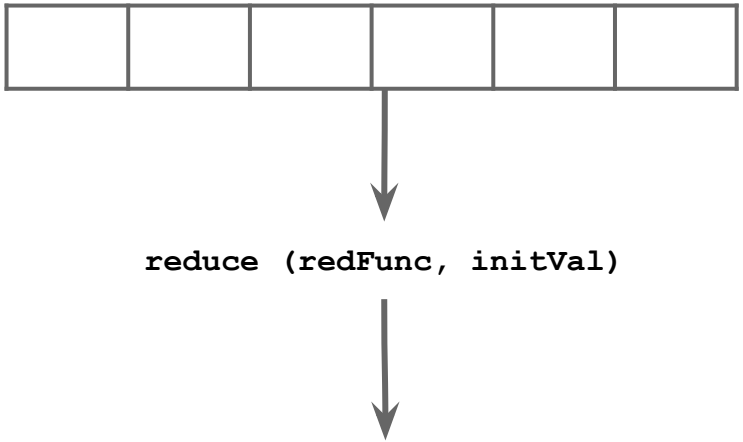
Based on the original version by Professor Hans Dulimarta.

# Recall

- Functions as Arguments (to another Fn) : High-order functions
  - Array.some()
  - Array.every()
  - Array.forEach()
  - Array.find(), Array.findIndex()
  - Array.filter()
  - Array.map() ,Array.flatmap()

# Introducing: Array.reduce()




***Not limited to only boolean output!!!***

# Array.reduce(): sum of values

```
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)

```
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 |

# 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()

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

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

*without initial value?*

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

# 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?

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

**Practice**

# The name of the longest river?

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

```
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);
```

**Practice**

```
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);
```

# What is the length of the longest river (in miles)?

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

**Practice**

# What is the length of the longest river (in miles)?

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

```
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);
```

**Practice**

```
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);
```

# Which river flows through the most countries?

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

**Practice**

# Which river flows through the 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);
```

**Practice**

```
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);
```

# 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;
  });
```