

Comp 125 - Visual Information Processing

Spring Semester 2018 - week 3 - friday

Dr Nick Hayward

Fun exercise - using variables and operators

- calculate the **number of seconds in an hour**
- using the **number of seconds in an hour**, calculate the **number of seconds in a day**
- using **number of seconds in a day**, calculate the **number of seconds in a year**
- using **number of seconds in a year**, calculate the **number of seconds in your current age** in years, e.g. 22 years

Output each answer to the document with a line break between each result.

- please signup for a CodePen account - <https://codepen.io/>
 - *use for writing and testing assignment*
 - *send URL to completed PEN for assignment - use private message to TA*

JS Data Structures - arrays - multi-dimensional access

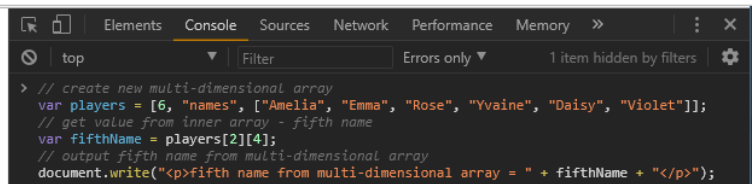
- then access value in an inner array using familiar pattern of index positions, e.g.

```
// create new multi-dimensional array  
var players = [6, "names", ["Amelia", "Emma", "Rose", "Yvaine", "Daisy", "Violet"]];  
// get value from inner array - fifth name  
var fifthName = players[2][4];
```

JS Data Structures - arrays - multi-dimensional access

access the inner array of a multi-dimensional array...

fifth name from multi-dimensional array = Daisy



```
> // create new multi-dimensional array
var players = [6, "names", ["Amelia", "Emma", "Rose", "Yvaine", "Daisy", "Violet"]];
// get value from inner array - fifth name
var fifthName = players[2][4];
// output fifth name from multi-dimensional array
document.write("<p>fifth name from multi-dimensional array = " + fifthName + "</p>");
```

JavaScript - array access

JS Data Structures - arrays - using arrays

- JavaScript provides two common options for working with and using arrays
 - ***properties*** and ***methods***
- **properties** tell us about the array
 - e.g. *length*
 - *length* property includes both defined and undefined values
- **methods** help us modify, update, or return a new array
 - e.g. *push()*, *unshift()*, *pop()*, *shift()*...

JS Data Structures - arrays - using arrays

check array object and property length...

```
> // use array literals to create new array
var players = ["Amelia", "Emma", "Daisy", "Yvaine"];
// update player name
players[3] = "Rose";
// add new player to the array
players[5] = "Violet";
// check updated array
players;
< ▼ (6) ["Amelia", "Emma", "Daisy", "Rose", empty, "Violet"] ⓘ
  0: "Amelia"
  1: "Emma"
  2: "Daisy"
  3: "Rose"
  5: "Violet"
  length: 6
  ▶ __proto__: Array(0)
> // check length of players array
players.length;
< 6
> |
```

JavaScript - array property - length

JS Data Structures - arrays - push() & unshift()

- add a value to the start or end of an array
- push () - adds new value to end of an array
 - *auto-increments array index, e.g.*

```
var places = [];  
places.push("Waldzell");
```

- unshift () - adds new value to start of an array
 - *increments each index position by 1 for existing values, e.g.*

```
var places = ["Waldzell"];  
places.unshift("Mariafels");
```

JS Data Structures - arrays - push() & unshift()

push () a value to the end of an array, and then unshift () a value at the start of an array...

```
> // define variable for new array
var places = [];
// push new value to end of array
places.push("Waldzell");
places;
< ▾ ["Waldzell"] ⓘ
  0: "Waldzell"
  length: 1
  ▶ __proto__: Array(0)
> places.unshift("Mariafels");
< 2
> places;
< ▾ (2) ["Mariafels", "Waldzell"] ⓘ
  0: "Mariafels"
  1: "Waldzell"
  length: 2
  ▶ __proto__: Array(0)
> |
```

JavaScript - arrays - push and unshift

JS Data Structures - arrays - pop() & shift()

- we can also remove items from the end or start of an array
- pop() - removes the last value from an array, e.g.

```
var places = ["Waldzell", "Mariafels", "Castalia"];  
places.pop();
```

- shift() - removes the first value from an array, e.g.

```
var places = ["Waldzell", "Mariafels"];  
places.shift();
```

JS Data Structures - arrays - pop()

pop() a value from the end of an array...

```
> // define variable for new array
var places = [];
// push new value to end of array
places.push("Waldzell");
places;
< ▶ ["Waldzell"]
> places.unshift("Mariafels");
< 2
> places;
< ▶ (2) ["Mariafels", "Waldzell"]
> // pop the last item from the array
places.pop();
< "Waldzell"
> // check places array
places;
< ▼ ["Mariafels"] ⓘ
  0: "Mariafels"
  length: 1
  ▶ __proto__: Array(0)
> |
```

JavaScript - arrays - pop

JS Data Structures - arrays - shift()

shift() a value from the start of an array...

```
> // create array with places
var places = ["Waldzell", "Mariafels", "Castalia"];
// remove first item from array with shift()
places.shift()
< "Waldzell"
> // check places array
places;
< ▼ (2) ["Mariafels", "Castalia"] ⓘ
  0: "Mariafels"
  1: "Castalia"
  length: 2
  __proto__: Array(0)
> |
```

JavaScript - arrays - shift

JS Data Structures - arrays - adding/combining

- also combine two or more arrays to create a new combined, single array
- to perform this task, JavaScript provides the `concat ()` method
- e.g. we might need to combine two existing arrays

```
var playersOne = ["Amelia", "Yvaine", "Emma"];  
var playersTwo = ["Daisy", "Violet", "Rose", "Clementine"];  
var allPlayers = playersOne.concat(playersTwo);
```

- values from `playersOne` array will be added to start of new array `allPlayers`
 - *values from `playersTwo` array will be added to end of new array*

JS Data Structures - arrays - adding/combining

combine two arrays to create a new array of values...

```
> // create first array of values
var playersOne = ["Amelia", "Yvaine", "Emma"];
// create second array of values
var playersTwo = ["Daisy", "Violet", "Rose", "Clementine"];
// combine arrays to create a new array
var allPlayers = playersOne.concat(playersTwo);
// check output
allPlayers;
< ▼ (7) ["Amelia", "Yvaine", "Emma", "Daisy", "Violet", "Rose", "Clementine"] ⓘ
  0: "Amelia"
  1: "Yvaine"
  2: "Emma"
  3: "Daisy"
  4: "Violet"
  5: "Rose"
  6: "Clementine"
  length: 7
  ▶ __proto__: Array(0)
> |
```

JavaScript - arrays - concat

JS Data Structures - arrays - adding/combining

combine two arrays to create a new array of values...change order

```
> // create first array of values
var playersOne = ["Amelia", "Yvaine", "Emma"];
// create second array of values
var playersTwo = ["Daisy", "Violet", "Rose", "Clementine"];
// combine arrays to create a new array
var allPlayers = playersTwo.concat(playersOne);
// check output
allPlayers;
< ▼ (7) ["Daisy", "Violet", "Rose", "Clementine", "Amelia", "Yvaine", "Emma"] ⓘ
  0: "Daisy"
  1: "Violet"
  2: "Rose"
  3: "Clementine"
  4: "Amelia"
  5: "Yvaine"
  6: "Emma"
  length: 7
  ▶ __proto__: Array(0)
> |
```

JavaScript - arrays - concat

JS Data Structures - arrays - find index

- also find index position of a known value in a given array
 - using the method `indexOf()`
- e.g. we need to find index position of value Daisy

```
var playersAll = ["Amelia", "Yvaine", "Emma", "Daisy", "Violet", "Rose", "Clementine"];  
var indexPosn = playersAll.indexOf("Daisy");
```

- returned index position will be 3
 - if value cannot be found in array, return will be `-1`
 - if value appears more than once in array
 - `indexOf()` will return first index position of value

JS Data Structures - arrays - find index

check index position of a value in an array...

```
> // create first array of values
var playersAll = ["Amelia", "Yvaine", "Emma", "Daisy", "Violet", "Rose", "Clementine"];
// check index of Daisy
var indexPosn = playersAll.indexOf("Daisy");
// check indexPosn value
indexPosn;
< 3
> // check if Beatrice is in the array
playersAll.indexOf("Beatrice");
< -1
> |
```

JavaScript - arrays - indexOf