Comp 125 - Visual Information Processing

Spring Semester 2019 - Week 4 - Monday

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

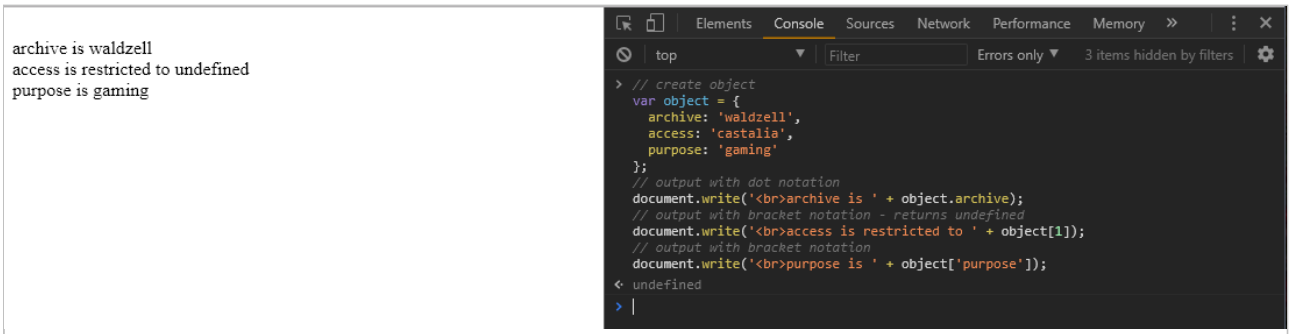
Output each answer to the document with a line break between each result.

- 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 Objects - example

```
// create object
var object = {
  archive: 'waldzell',
  access: 'castalia',
  purpose: 'gaming'
};

// output with dot notation
document.write('<br>archive is ' + object.archive);

// output with bracket notation - returns undefined
document.write('<br>access is restricted to ' + object[1]);

// output with bracket notation
document.write('<br>purpose is ' + object['purpose']);
```

JS Objects - example output



```
archive is waldzell
access is restricted to undefined
purpose is gaming
```

```
> // create object
  var object = {
    archive: 'waldzell',
    access: 'castalia',
    purpose: 'gaming'
  };
  // output with dot notation
  document.write('<br>archive is ' + object.archive);
  // output with bracket notation - returns undefined
  document.write('<br>access is restricted to ' + object[1]);
  // output with bracket notation
  document.write('<br>purpose is ' + object['purpose']);
<· undefined
> |
```

JS Object - example output

JS Objects - all keys

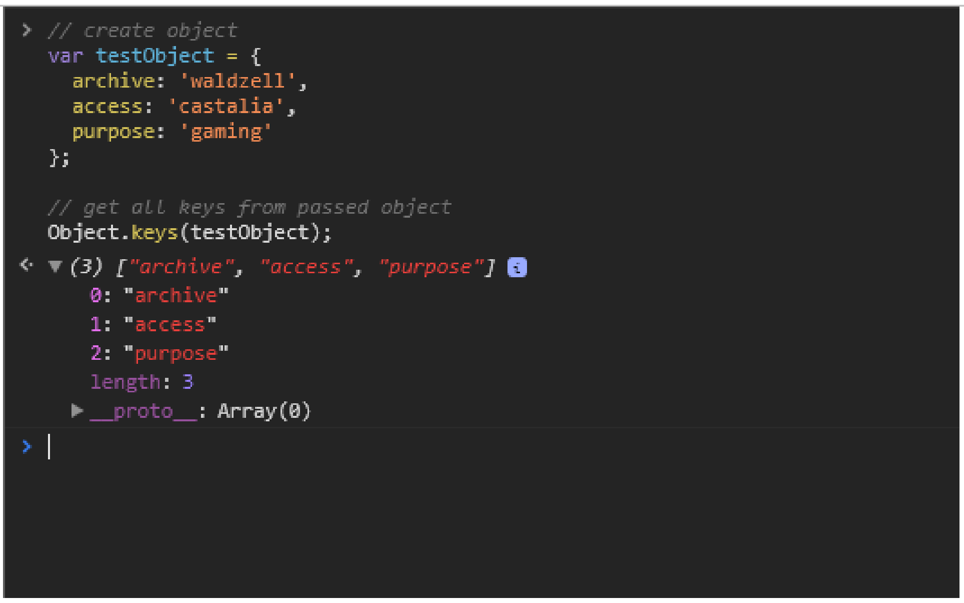
- access single values using a specific key
  - *dot or bracket notation...*
  - *JS provides method to access all keys in passed object*
  - *e.g. using `Object.keys()` method*

```
// create object
var testObject = {
  archive: 'waldzell',
  access: 'castalia',
  purpose: 'gaming'
};

// get all keys for passed object
Object.keys(testObject);
```

- `keys()` method returns an array of keys for `testObject`

JS Objects - all keys

get all keys from the passed object...



```
> // create object
  var testObject = {
    archive: 'waldzell',
    access: 'castalia',
    purpose: 'gaming'
  };

  // get all keys from passed object
  Object.keys(testObject);
<· ▼(3) ["archive", "access", "purpose"]
      0: "archive"
      1: "access"
      2: "purpose"
      length: 3
    ▶__proto__: Array(0)
> |
```

JS Object - get all keys

JS Objects - add values

- to add values to an object, we might need to start with an empty object

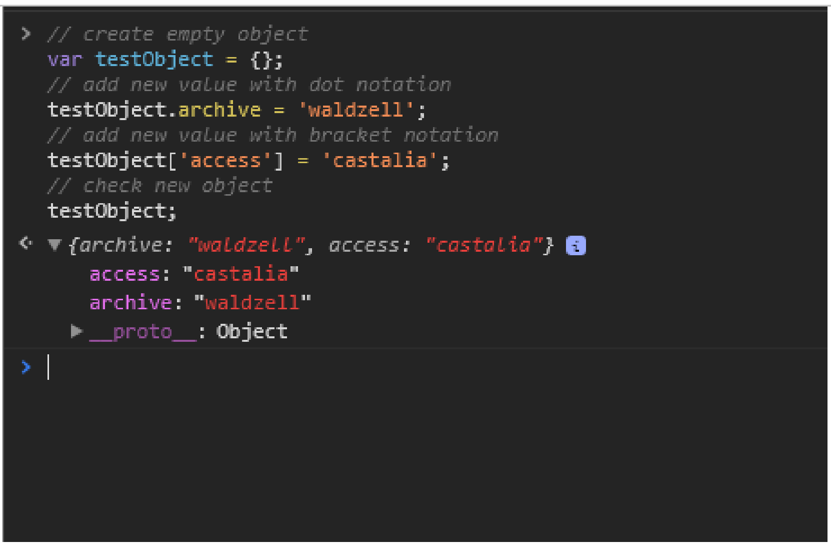
```
// create empty object
var testObject = {};
```

- uses same pattern as creating **array**
  - *{ } for object*
    - [ ] for array
  - *add single values to new object*

```
// create empty object
var testObject = {};
// add new value with dot notation
testObject.archive = 'waldzell';
// add new value with bracket notation
testObject['access'] = 'castalia';
```

JS Objects - add values

add some values to an empty object...



```
> // create empty object
  var testObject = {};
  // add new value with dot notation
  testObject.archive = 'waldzell';
  // add new value with bracket notation
  testObject['access'] = 'castalia';
  // check new object
  testObject;
<· ▼{archive: "waldzell", access: "castalia"}
      access: "castalia"
      archive: "waldzell"
    ▶ __proto__: Object
> |
```

JS Object - add some values

JS Objects - all values

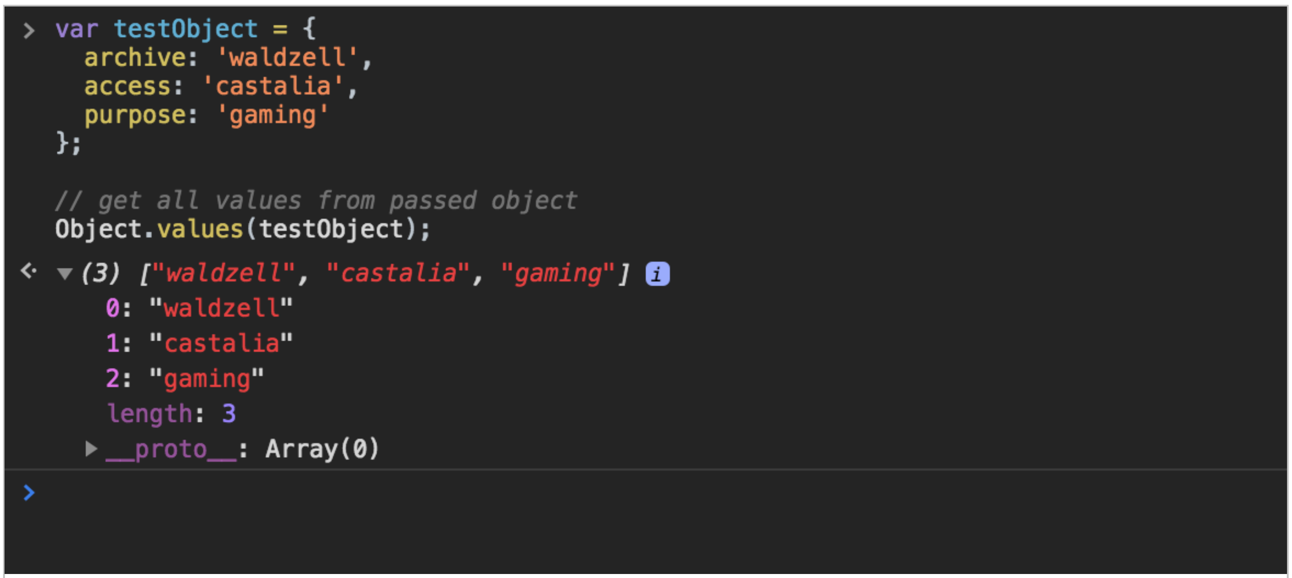
- JS provides method to access all values in passed object
  - *e.g. using `Object.values()` method*

```
// create object
var testObject = {
  archive: 'waldzell',
  access: 'castalia',
  purpose: 'gaming'
};

// get all values for passed object
Object.values(testObject);
```

- `value()` method returns an array of values for `testObject`

JS Objects - all values

get all values from the passed object...



```
> var testObject = {
    archive: 'waldzell',
    access: 'castalia',
    purpose: 'gaming'
  };

  // get all values from passed object
  Object.values(testObject);
<· ▼(3) ["waldzell", "castalia", "gaming"] i
      0: "waldzell"
      1: "castalia"
      2: "gaming"
      length: 3
    ▶ __proto__: Array(0)
>
```

JS Object - get all values

JS Objects - all entries

example 1

- JS provides method to access all entries in passed object
- e.g. using `Object.entries()` method
- return keys and values

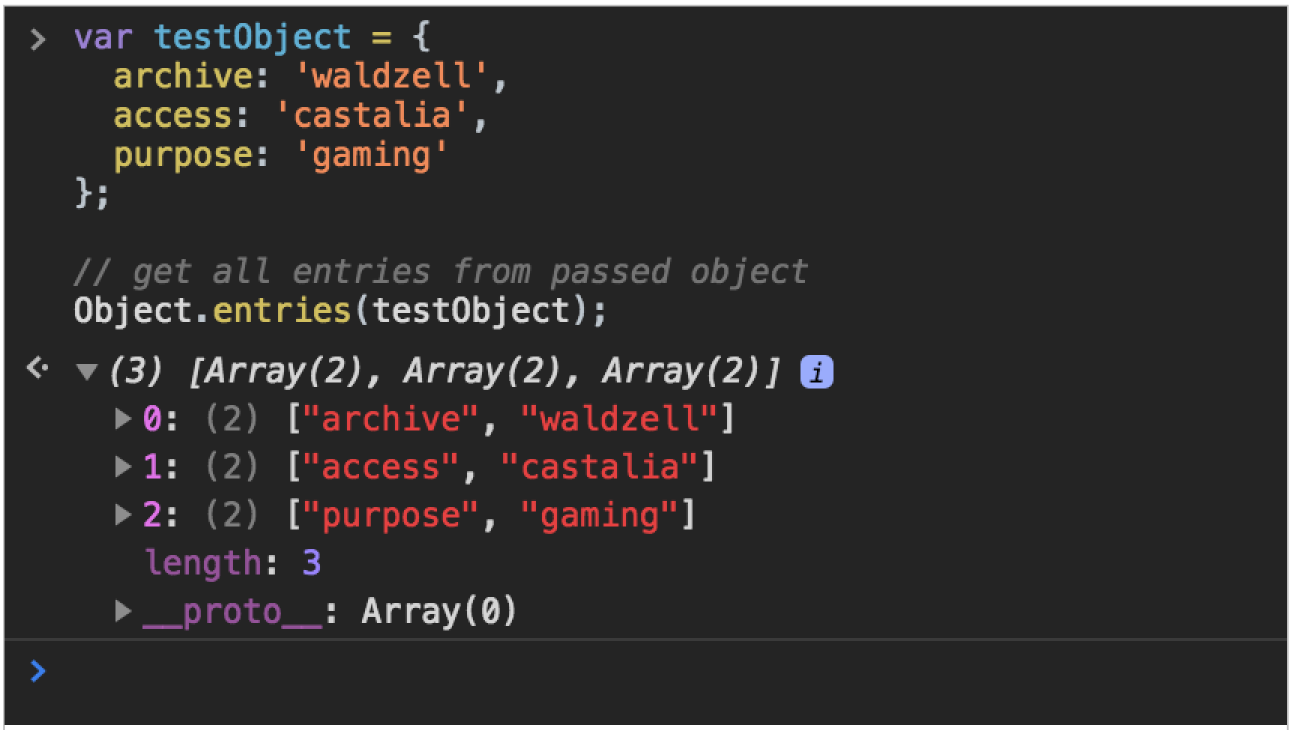
```javascript
// create object
var testObject = {
  archive: 'waldzell',
  access: 'castalia',
  purpose: 'gaming'
};

// get all entries for passed object
Object.entries(testObject);
```

- `entries()` method returns a multidimensional array of keys and values for `testObject`
- each inner array has key and values

JS Objects - all entries

example 1

get all entries from the passed object...



```
> var testObject = {
    archive: 'waldzell',
    access: 'castalia',
    purpose: 'gaming'
  };

  // get all entries from passed object
  Object.entries(testObject);
<· ▼(3) [Array(2), Array(2), Array(2)] i
    ▶0: (2) ["archive", "waldzell"]
    ▶1: (2) ["access", "castalia"]
    ▶2: (2) ["purpose", "gaming"]
     length: 3
    ▶__proto__: Array(0)
>
```

JS Object - get all entries

JS Objects - all entries

example 2

- `Object.entries()` method
  - *return keys and values*
  - *returns value regardless of data type*
  - *e.g. object, array values...*

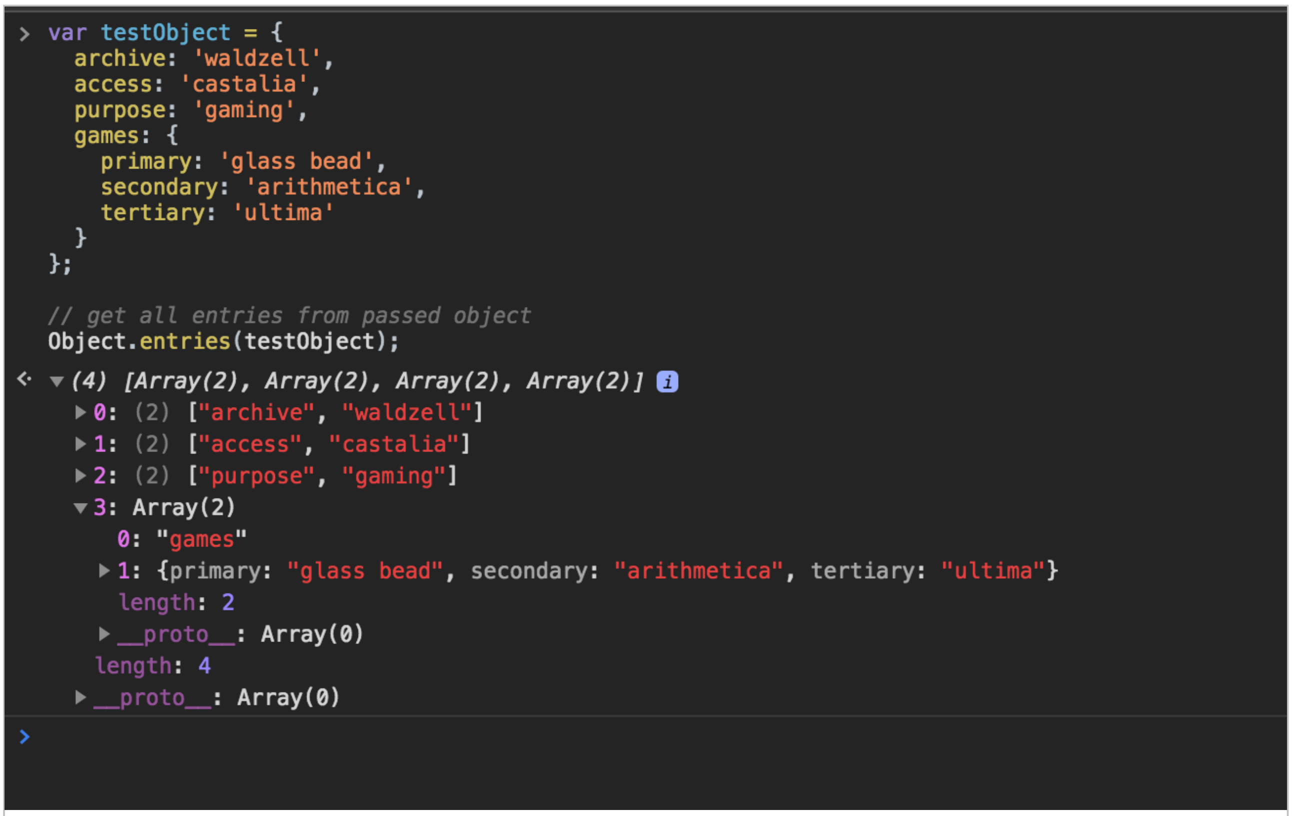
```
var testObject = {
  archive: 'waldzell',
  access: 'castalia',
  purpose: 'gaming',
  games: {
    primary: 'glass bead',
    secondary: 'arithmetica',
    tertiary: 'ultima'
  }
};

// get all entries from passed object
Object.entries(testObject);
```

- `entries()` method returns a multidimensional array of keys and values for `testObject`
- each inner array has key and values

JS Objects - all entries

example 2

get all entries from the passed object...



```
> var testObject = {
    archive: 'waldzell',
    access: 'castalia',
    purpose: 'gaming',
    games: {
      primary: 'glass bead',
      secondary: 'arithmetica',
      tertiary: 'ultima'
    }
  };

  // get all entries from passed object
  Object.entries(testObject);
<· ▼(4) [Array(2), Array(2), Array(2), Array(2)] i
   ▶0: (2) ["archive", "waldzell"]
   ▶1: (2) ["access", "castalia"]
   ▶2: (2) ["purpose", "gaming"]
   ▼3: Array(2)
      0: "games"
     ▶1: {primary: "glass bead", secondary: "arithmetica", tertiary: "ultima"}
      length: 2
     ▶__proto__: Array(0)
    length: 4
   ▶__proto__: Array(0)
>
```

JS Object - get all entries

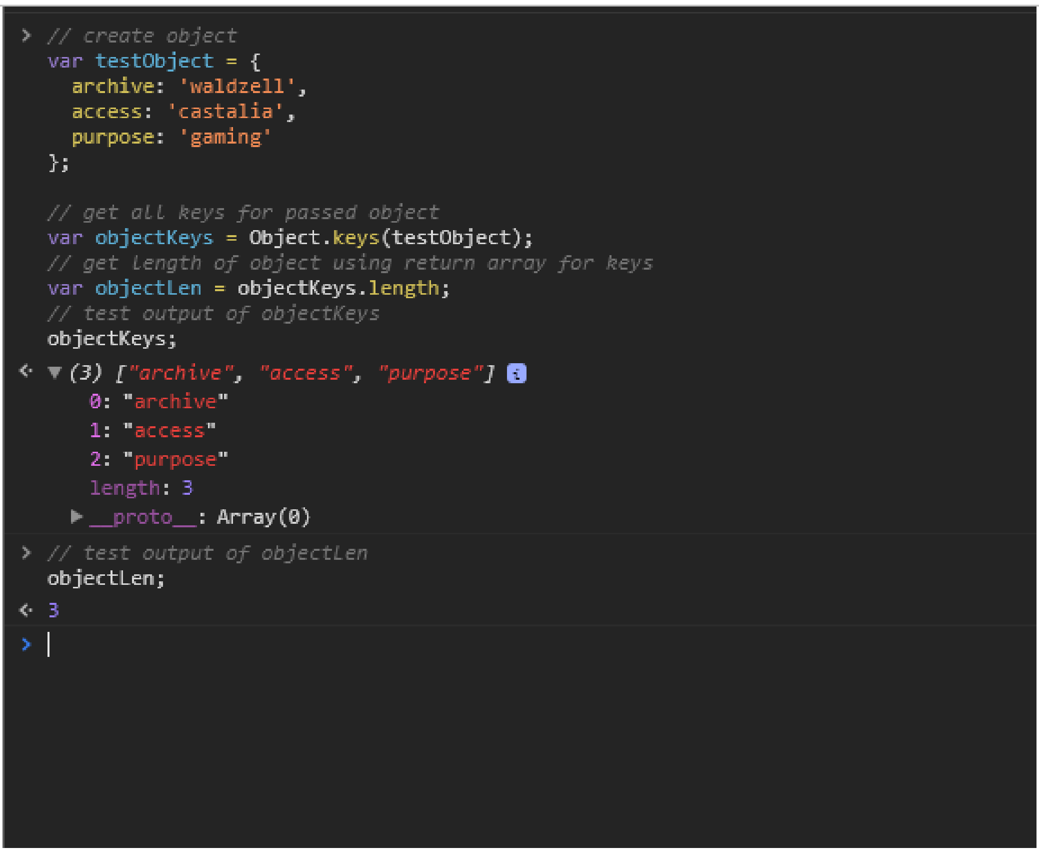
JS Objects - get length of object

- an object does not include its own `length` property
  - *but array includes the `length` property*
  - *we can use `keys()` method to get array of keys*
  - *then get `length` from keys array for passed object*

```
// create object
var testObject = {
  archive: 'waldzell',
  access: 'castalia',
  purpose: 'gaming'
};

// get all keys for passed object
var objectKeys = Object.keys(testObject);
// get length of object using return array for keys
var objectLen = objectKeys.length;
```

JS Objects - get length of object - v.1

use `keys()` and array `length` property...return `keys` array and length of object

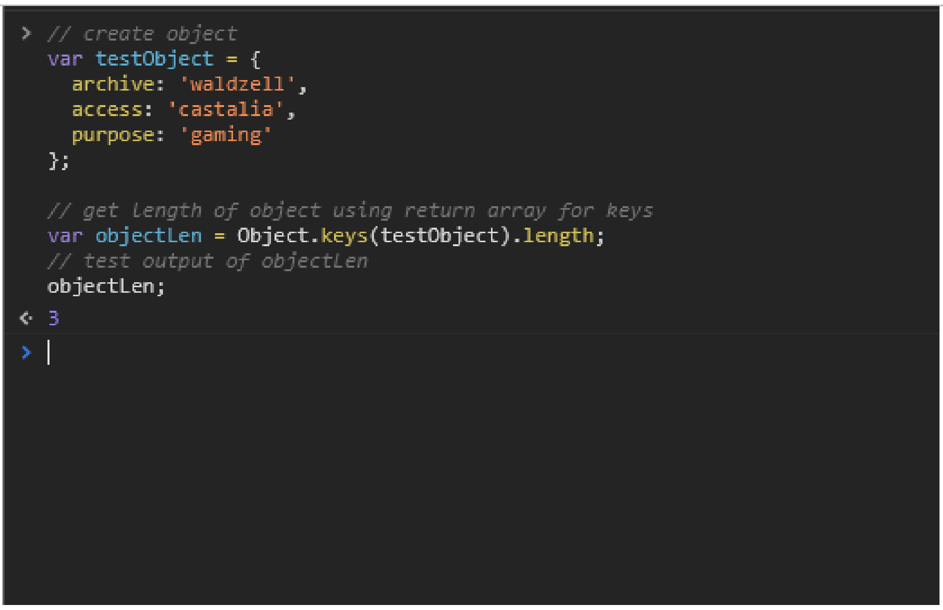


```
> // create object
  var testObject = {
    archive: 'waldzell',
    access: 'castalia',
    purpose: 'gaming'
  };

  // get all keys for passed object
  var objectKeys = Object.keys(testObject);
  // get length of object using return array for keys
  var objectLen = objectKeys.length;
  // test output of objectKeys
  objectKeys;
<· ▼(3) ["archive", "access", "purpose"] i
      0: "archive"
      1: "access"
      2: "purpose"
      length: 3
    ▶__proto__: Array(0)
> // test output of objectLen
  objectLen;
<· 3
> |
```

JS Object - get object length

JS Objects - get length of object - v.2

use `keys()` and array `length` property...only return length of object



```
> // create object
  var testObject = {
    archive: 'waldzell',
    access: 'castalia',
    purpose: 'gaming'
  };

  // get length of object using return array for keys
  var objectLen = Object.keys(testObject).length;
  // test output of objectLen
  objectLen;
<· 3
> |
```

JS Object - get object length

JS Objects - arrays as objects

- JS array an object that contains values, of any type, in numerically indexed positions
  - *store a number, a string...*
  - *array will start at index position 0*
  - *increments by **1** for each new value*
- arrays can also have properties
  - *eg: automatically updated **length** property*

```
var arrayA = [
  49,
  59,
  "Philae"
];
arrayA.length; //returns 3
```

- each value can be retrieved from its applicable index position,

```
arrayA[2]; //returns the string "Philae"
```

JS Objects - array structure

```
|-----------------------------------------|
|            |            |               |
|            |            |               |
|   0: 49    |   1: 59    |   2: "Philae" |
|            |            |               |
|            |            |               |
|-----------------------------------------|
```

JS Array

JS Objects - combine arrays and objects

- objects and arrays may also be combined in JavaScript
  - *an object in an array, array in object...*

```
// create array with object
var archives = [
    { name: 'waldzell', access: 'castalia', purpose: 'gaming' },
    { name: 'bodleian', access: 'oxford', purpose: 'research'}
];
```

- then access inner object

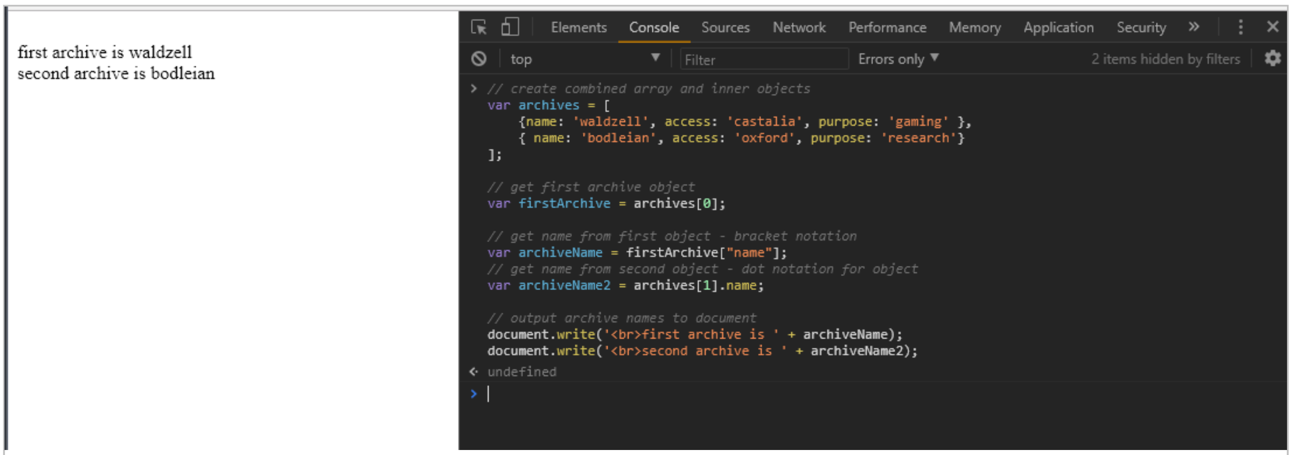
```
// get first archive object
var firstArchive = archives[0];
```

- then, we can get the name of the first archive, e.g.

```
// get name from first object - bracket notation
var archiveName = firstArchive["name"];
// get name from second object - dot notation for object
var archiveName2 = archives[1].name;
```

JS Objects - combine arrays and objects

combine arrays and objects...access inner values



```
first archive is waldzell
second archive is bodleian
```

```
> // create combined array and inner objects
  var archives = [
      {name: 'waldzell', access: 'castalia', purpose: 'gaming' },
      { name: 'bodleian', access: 'oxford', purpose: 'research'}
  ];

  // get first archive object
  var firstArchive = archives[0];

  // get name from first object - bracket notation
  var archiveName = firstArchive["name"];
  // get name from second object - dot notation for object
  var archiveName2 = archives[1].name;

  // output archive names to document
  document.write('<br>first archive is ' + archiveName);
  document.write('<br>second archive is ' + archiveName2);
< undefined
> |
```

JS - array and object combined

Fun exercise - using arrays

- create a new array, named **cities**, with the following values
  - *Paris, Marseille, Nice*
- add the following values to the end of the array
  - *Toulouse, Lyon*
- remove the fourth value from the array
- add the following values to the start of the array
  - *Cannes, Avignon*
- move the third value in the array to the end of the array
- move the fourth value in the array to the start of the array

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