NEXT ACADEMY

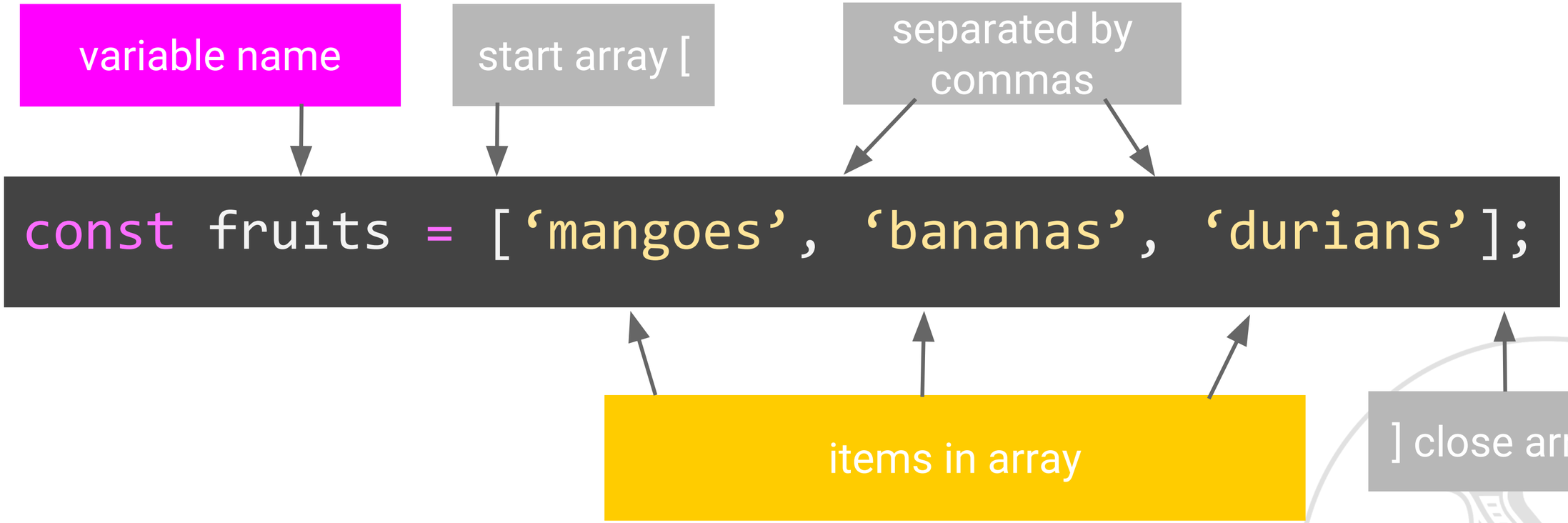
Introduction to Web Development

JavaScript Pt 2

arrays

data type

Anatomy of an array



creating an array

```
const fruits = [‘mangoes’, ‘bananas’, ‘durians’];
```

fruits

item	‘mangoes’	‘bananas’	‘durians’
index	0	1	2

creating an array

```
const things = [42, ‘hello’, -1, true];
```

things

item	42	‘hello’	-1	true
index	0	1	2	3

creating an array

```
const emptyArray = [];
```

emptyArray

item	undefined	undefined	undefined	...
index	0	1	2	...

getting an item

```
const fruits = [‘mangoes’, ‘bananas’, ‘durians’];

console.log(fruits[0]) //mangoes

console.log(fruits[1]) //bananas

console.log(fruits[2]) //durians

console.log(fruits[3]) //undefined
```

setting an item

```
const fruits = [‘mangoes’, ‘bananas’, ‘durians’];

console.log(fruits[1]) //bananas

fruits[1] = ‘papayas’

console.log(fruits[1]) //papayas
```

fruits

item	‘mangoes’	‘papayas’	‘durians’
index	0	1	2

array length

Number of items in array

```
const fruits = [‘mangoes’, ‘bananas’, ‘durians’, ‘lychees’];

console.log(fruits.length) //4
```

fruits

item	‘mangoes’	‘bananas’	‘durians’	‘lychees’
index	0	1	2	3

zero-indexing quiz!

Given the following array:

```
const fruits = [‘mangoes’, ‘bananas’, ‘dragonfruits’,
                ‘durians’, ‘limes’, ‘lychees’];
```

What is the index of

- mangoes?
- lychees
- dragonfruits?

adding items

myArray.push()

```
const fruits = [‘mangoes’, ‘bananas’, ‘durians’];

console.log(fruits.length); //3

fruits.push(‘guava’);

console.log(fruits.length) //4
```

removing last item

myArray.pop()

```
const fruits = [‘mangoes’, ‘bananas’, ‘durians’, ‘guava’];

console.log(fruits.length); //4

fruits.pop()

console.log(fruits.length) //3

console.log(fruits) //[‘mangoes’, ‘bananas’, ‘durians’]
```

removing item at position

myArray.splice()

```
const fruits = [‘mangoes’, ‘bananas’, ‘durians’, ‘guava’];

console.log(fruits.length); //4

fruits.splice(1, 1)

console.log(fruits.length) //3

console.log(fruits) //[‘mangoes’, ‘durians’, ‘guava’]
```

removing item at position

myArray.splice()

```
const fruits = [‘mangoes’, ‘bananas’, ‘durians’, ‘guava’];

console.log(fruits.length); //4

fruits.splice(0, 2)

console.log(fruits.length) //2

console.log(fruits) //[‘durians’, ‘guava’]
```

if-else

control flow

Concept: ‘if’

Do something if the value of a boolean expression (condition) is true. For example:

If today is Saturday, then I will eat ice-cream

Anatomy of ‘if’

If today is Saturday, then I will eat ice-cream

if	(condition is true)	then do stuff in block { }

```
if(today == “Saturday”){
  document.write(“Yum, ice-cream!”);
  iceCream = iceCream + 1;
}
```

‘if-else’

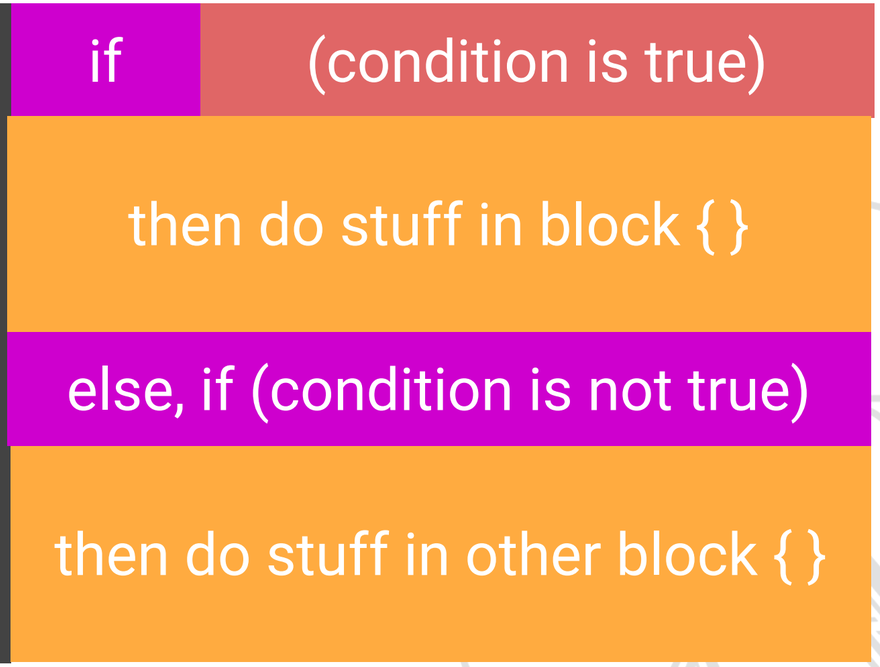
Do something if the value of a boolean expression is true. Otherwise, do another thing. For example:

If today is Saturday, then I will eat ice-cream. Else, I will eat cookies.

Anatomy of ‘if-else’

If today is Saturday, then I will eat ice-cream. Else, I will eat cookies.

```
if(today == “Saturday”){
  document.write(“Yum,  ice-cream!”);
}
else{
  document.write(“Crunchy cookies!”);
}
```



‘if-else if-else’

Like adding another if with a different condition.

If today is Saturday,

then I will eat ice-cream.

Else if today is Wednesday,

I will eat fried rice.

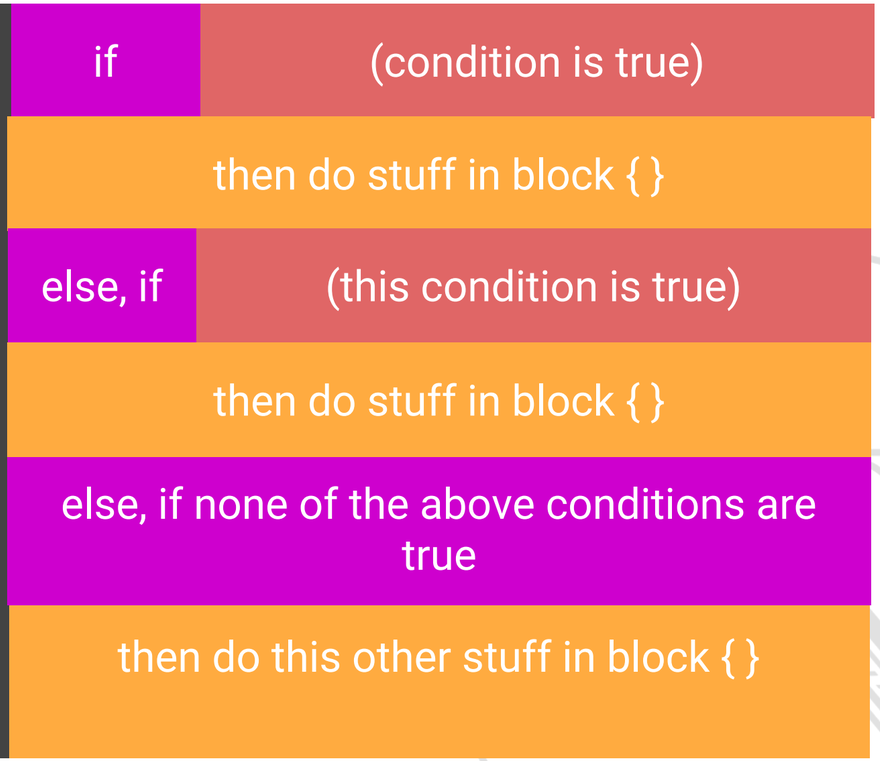
Else,

I will eat cookies :)

Anatomy of ‘if-else if-else’

If today is Saturday, then I will eat ice-cream. Else if today is Wednesday, I will eat chicken rice. Else, I will eat cookies.

```
if(today == “Saturday”){
   document.write(“Yum,  ice-cream!”);
}
else if (today == “Wednesday”){
     document.write(“Cluck cluck rice!”);
}
else{
   document.write(“Crunchy cookies!”);
}
```



Ordering matters

What happens when:

```
const cookies = 10;
if(cookies < 12){
    document.write("I still have enough cookies for you")
}
else if (cookies < 11){
    document.write("Not enough cookies :( ");
}
```

‘if-else if-else’ Caveats

If a condition is fulfilled, it will run the contents of { ... } and **stop** checking the other conditions.

Your conditions should start with the strictest one.

Truthy and Falsy

What if your condition is not a **boolean**?

```
const myString = ‘hello’;

if (myString) {

  console.log(‘hi’)

}
```

Truthy and Falsy values*

```
if (false) {}
if (null) {}
if (undefined) {}
if (0) {}
if (NaN) {}
if ('') {}
if ("") {}
```

*Anything that is not falsy is truthy

if-else exercise

TBA

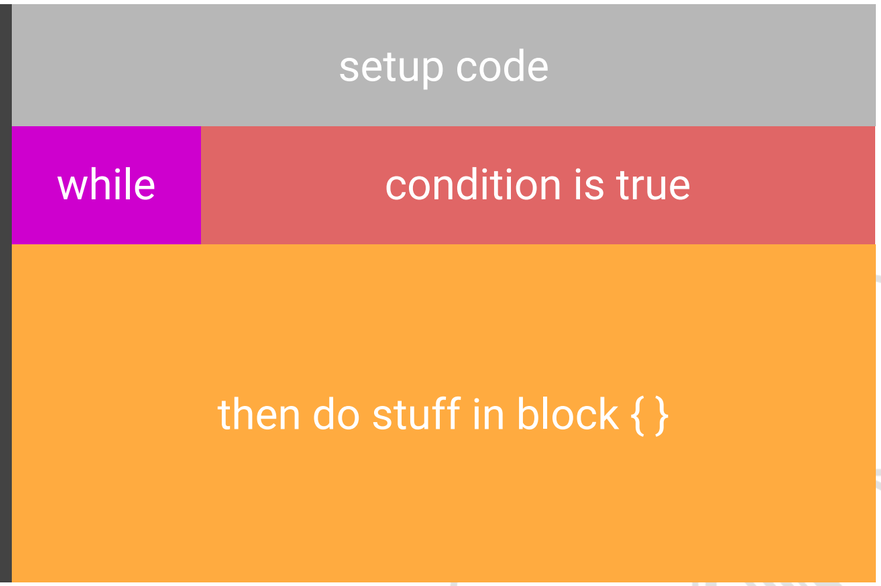
NEXT ACADEMY

loops

control flow

Anatomy of a while loop

```
let count = 0;
while(count < 10){
  console.log('round' + count);
  count = count + 1;
}
```



tracing loops

Looking at what's going on, step by step.

```
let scoops = 5;
while(scoops > 0){
  document.write('another scoop!<br/>')
  scoops = scoops - 1;
}
```

beware the infinite loop!

```
let count = 0;
while(count < 10){
  console.log(‘round’ + count);
  // counter not updated?!
}
```

using loops to find things

```
const pets = [‘nina’, ‘fluffy’, ‘kitty’, ‘mimi’];
let foundRoom = -1;
let i = 0;
while(i < pets.length){
  if(pets[i] == ‘kitty’){
    foundRoom = i;
  }
  i++;
}
console.log(‘found kitty at room: ’ + foundRoom);
```

breaking out of the loop

Will stop the loop.

Any other pets that comes after ‘kitty’ will not be checked.

```
while(i < pets.length){
  if(pets[i] == ‘kitty’){
    foundRoom = i;
    break;
  }
  i++
}
console.log(‘found kitty at room: ’ +
foundRoom);
```

Anatomy of a for loop

```
for(let i = 0; i < 10; i++){
  console.log('round' + i);
}
```

for

initialize; condition; update

code block {}

Closer look

only runs once

runs at the start of the loop

runs at the end of the loop

for	initialize counter;	check condition;	update counter

```
for(let i = 0; i < 10; i++){
  console.log('round' + i);
}
```

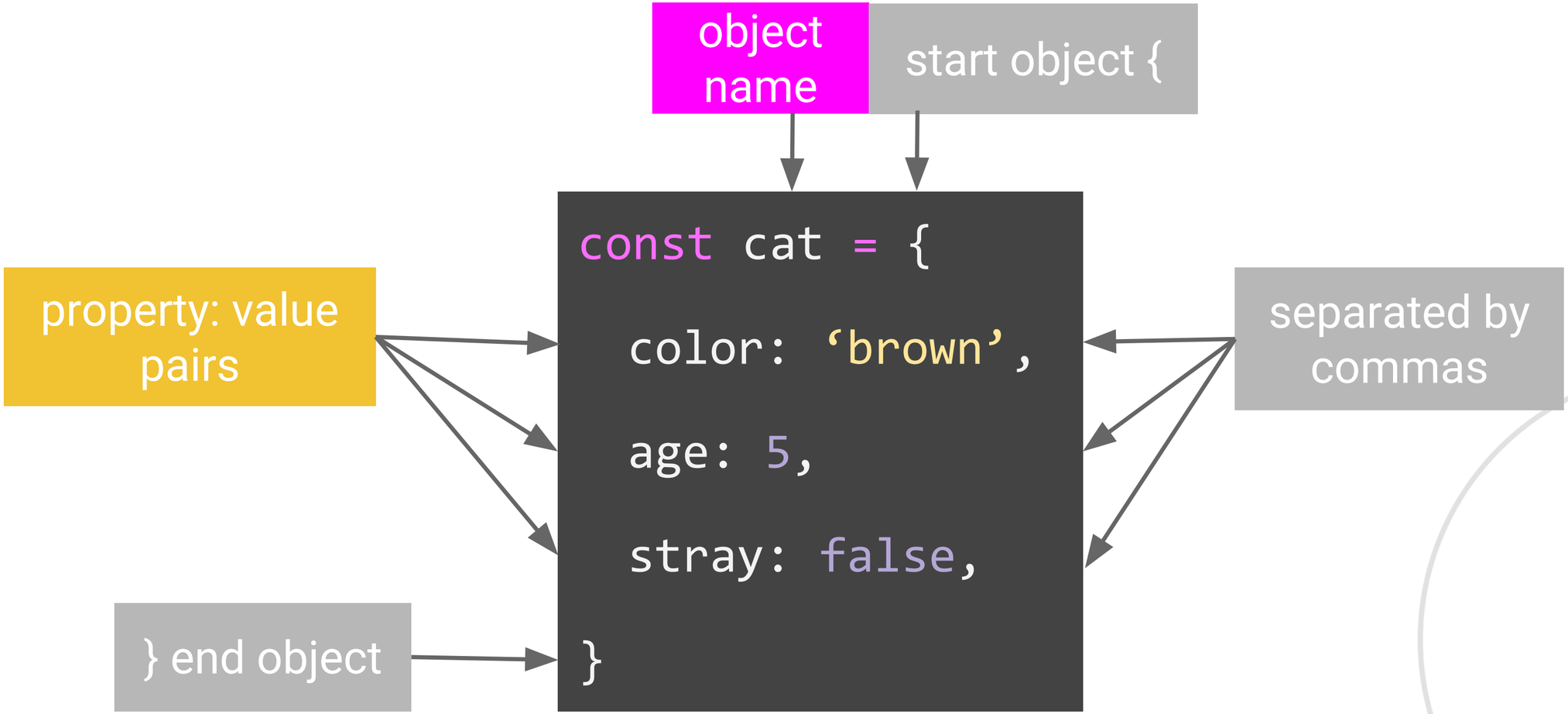
arrays with for-loops

```
const pets = [‘nina’, ‘fluffy’, ‘kitty’, ‘mimi’];
let foundRoom = -1;
for(let i = 0; i < pets.length; i++){
  if(pets[i] == ‘kitty’){
    foundRoom = i;
  }
}
console.log(‘found kitty at room: ’ + foundRoom);
```

Object

data type

Anatomy of an Object



getting

```
const peanut = {color: ‘brown’, age: 5, stray: false}

const butter = {color: ‘yellow’, age: 3, stray: true}

console.log(peanut[‘age’]); //5

console.log(butter[‘color’]); // ‘yellow’

console.log(peanut.age); //5

console.log(butter.stray); //true
```

setting

```
const peanut = { color: ‘brown’, age: 5, stray: false}

console.log(peanut[‘age’]); //5

peanut.age = 99;

peanut[‘age’] = 99;

console.log(peanut[‘age’]); //99
```

functions as properties?!

```
const peanut = {
  color: ‘peanut’,
  age: 5,
  stray: false,
  meow: function(){
    console.log(‘MEOW MEOW’);
  }
}
```

```
peanut.meow();

// logs ‘MEOW MEOW’
```