The real Flickr API

https://api.flickr.com/services/rest/?
method=flickr.photos.search&api_key=???
&tags=flowers&per_page=3&format=json

The api key ??? is an ID hard-coded into your app
that identifies it to Flickr – get it online.
Most APIs have this

Making a JSONp API call

- Add a script tag, with the src of the script containing
  the URL for the API call.
  - Make a script DOM element
  - Make up the script URL for the API call
  - Add it as the src property of the script element
  - Append the script to body in DOM
  - When browser executes the tag, it sends the URL to
    Flickr to retrieve the script
- Returned script:
  jsonFlickrAPI(
    "photos":
      
    "page":1,

Example real image URL

https://farm3.staticflickr.com/
2811/34211804241_32d196729e_m.jpg

Using the return data

- In function jsonFlickrAPI, need to...
  - Get the three image elements from DOM using
    getElementsByClassName
  - Get image data from the “data” object that
    jsonFlickrAPI got as input.
  - Loop through both image elements and image data,
    constructing URLs for the actual images from the ids,
    farms, and servers (farm is which Flickr server farm…
    there are apparently several)
  - Add each image’s URL as its src property

Response

- A little more complex…in our callback function
  jsonFlickrAPI(data), for instance, data.photos.photo
  contains:

"id":34211804241,
"owner":144222333@N02,
"secret":32d196729e,
"server":2811,
"farm":3,
"title":"Double Star","ispublic":1,"isfriend":0,"isfamily":0}
CSS using flexbox

Functions as values

- We know we can use functions as values in Javascript:

```
function f(x) { return x+2; }
var plus2=f;
plus2(5); // what does it return?
```

Adding an onclick value

- Say we want to highlight one of the images when we click on it.
- To add "onclick" from Javascript:

```
function highlight() { console.log("hi!"); }
var images = document.getElementsByClassName("flickrPhoto");
for (var i=0; i++; i<images.length) {
    image[i].onclick = highlight;
}
```

Notice...

- If we had said:

```
image[i].onclick = highlight();
```

...we’d be storing the return value of the function highlight() (undefined!) into the onclick property, not the function itself.

But which one to highlight?

- We’d like to pass the element index (first, second or third image) to the highlight() function.
- But we can’t change highlight() to take a parameter:

```
image[i].onclick = highlight(i);
```

...stores the result of highlight(i) into the onclick property, not the function itself.

Use new feature: closure

- Put highlight(i) into a new function that does not have any arguments.
- We’ll do two versions of this.

```
function makeOnClick(i, element) {
    function noarg() {
        highlight(i);
    }
    element.onclick = noarg;
}
```

- Notice we define a function inside another function.
Closure - version 1

```javascript
function makeOnClick(i, element) {
    function noarg() {
        highlight(i);
    }
    element.onclick = noarg;
}

// The function noarg calls highlight, which does have an argument
// noarg can be assigned to the onclick property
```

Closure - version 1

```javascript
for (i=0; i<3; i++) {
    makeOnClick(i, images[i])
}
```

Closure

- A closure is the context in which a function is defined.
- The closure of noarg is makeOnClick
- All the local variables of makeOnClick are available to noarg.
- The local variables of makeOnClick at the time noarg is defined are available to noarg

Closure - version 1

```javascript
function makeOnClick(i, element) {
    function noarg() {
        highlight(i);
    }
    element.onclick = noarg;
}

// The value of i is remembered when noarg is called!
```

Midterm problem

```javascript
function Weather(t, w) {
    this.fahrenheit = t;
    this.wind = w;
    this.celsius = function() {
        return (t-32)*5/9;
    }
}

// This works because the closure of this.celsius is Weather; it has access to the original version of t
```
Not the best solution

davisWeather = new Weather(77, 22);
davisWeather.celsius() // returns 25

davisWeather.fahrenheit = 86;
davisWeather.celsius() // returns?

Better solution

function Weather(t, w) {
    this.fahrenheit = t;
    this.wind = w;
    this.celsius = function() {
        return (this.fahrenheit-32)*5/9;
    }
}