1 **HTML APIs**

JavaScript

2 **APIs (Application Programming**

- An API (Application Programming Interface) is a set of methods and tools used for building software applications
- These tools are pre-programmed so that the developer can use them without having to recreate them (”without rebuilding the wheel”)

3 **Adding an API to a Website**

- Use a <script> tag with the src attribute assigned the URL address of the API source to add it to a website
- Format:

  <script src="*apiUrl*"></script>
- Example:

  <script src="http://maps.googleapis.com/maps/api/js"></script>

4 **The Google Maps API**

- The Google Maps API contains a JavaScript library of components for customizing maps and information on those maps
- It can be added easily to any web page within a <script> tag as follows:

  <script src="http://maps.googleapis.com/maps/api/js"></script>

5 **Setting Map Properties (Page 1)**

- Create an *objectVariable* using *name*: *value* pairs to define the properties for the map
- Format:

```
var objectVariable =
{
    center: new google.maps.LatLng(latitude, longitude),
    zoom: int,
    mapTypeId: google.maps.MapTypeId.CONSTANT
};
```

6 **Setting Map Properties (Page 2)**

- Create an *objectVariable* using *name*: *value* pairs to define the properties for the map (*con.*):
  - The center property specifies where to center the map using the google.maps.LatLng constructor to specify the coordinates in the order—latitude, longitude
  - The zoom property specifies the zoom level for the map (zero (0) shows a map of the Earth fully zoomed out and higher zoom levels zoom to a closer resolution)

7 **Setting Map Properties (Page 3)**

- Create an *objectVariable* using *name*: *value* pairs to define the properties for the map (*con.*):
  - The mapTypeId property specifies map type to display using the google.maps.MapTypeId class constants:
    - ROADMAP—normal, default 2D map
    - SATELLITE—photographic map only
    - HYBRID—photographic map + roads and city names
    - TERRAIN—topographical map with mountains, rivers, etc.

8 **Setting Map Properties (Page 4)**

- Example:

```
var mapProperties =
{
    center: new google.maps.LatLng(51.508742, -0.120850),
    zoom: 8,
    mapTypeId: google.maps.MapTypeId.ROADMAP
};
```

9 **Creating the Map Object (Page 1)**

- The google.maps.Map constructor creates the map object and places it in the Web document
- Format:

  var *mapObject* = new google.maps.Map(*container*, *mapPropertiesObject*);
  - The *container* is the HTML element that will "contain" the map (often a <div> block)
  - The *mapPropertiesObject* is the properties object variable that was previously defined

10 **Creating the Map Object (Page 2)**

- Example:

```
var map = new google.maps.Map(document.getElementById("googleMap"),
  mapProperties);
```

11 **Try It Out**

- apis1.htm
- api1a.htm

12 **Markers**

- A marker is an "overlay" object that notes a position on a map
  - Usually it will be the center as specified by longitude and latitude of the map object
- Involves two steps:
  - Create (instantiate) a marker object and set its properties
  - Call the setMap() method for the marker which inserts it into the map

13 **Instantiate a Marker Object (Page 1)**

- Call the google.maps.Marker constructor to create the marker object
- Format:

```
var markerObject = new google.maps.Marker(
{
    position: latitudeLongitudeObject
});
```

  - A *latitudeLongitudeObject* is assigned to the position property which specifies location on map where the point of *markerObject* will be displayed

14 **Instantiate a Marker Object (Page 2)**

- Example 1:

```
var marker = new google.maps.Marker(
{
    position: new google.maps.LatLng(51.5074, -0.1255)
});
```

  - Values 51.5074 and -0.1255 are latitude and longitude

15 **Instantiate a Marker Object (Page 3)**

- Example 2:

```
var marker = new google.maps.Marker(
{
    position: myCenter     // 'myCenter' is a position
                           //     (latitude/longitude) object
});
```

16 **Method setMap() for a Marker**

- The setMap() method for the marker object inserts it into map
  - This operation takes place after the map object already has been instantiated
- Format:

```
markerObject.setMap(mapObject);
```

- Example:

```
marker.setMap(map);
```

17 **Markers: animation Property (Page 1)**

- Markers have an animation property that defines the marker's behavior
- Valid values for the constant assigned to the animation are:
  - DROP—marker drops from top of map and comes to rest at its final location
  - BOUNCE—marker bounces up and down

18 **Markers: animation Property (Page 2)**

- The animation property is set using constants from the google.maps.Animation class
- Format:

animation: google.maps.Animation.*ANIMATION_CONSTANT*
  - Valid *ANIMATION_CONSTANT*s are DROP and BOUNCE

19 **Markers: animation Property (Page 3)**

- Example:

```
var marker = new google.maps.Marker(
{
    position: myCenter,
    animation: google.maps.Animation.BOUNCE
});
```

20 **Try It Out**

- api1b.htm

21 **An Image as the Marker**

- The icon property can be set for a marker object to substitute an image for the default Google API marker
- Format:

  icon: "*imageFile.extension*"
- Example:

```
var marker = new google.maps.Marker(
{
    position: myCenter,
    icon: "pointer.png"
});
```

22 **Try It Out**

- api1c.htm

23 **Circles**

- A circle is an "overlay" object that draws a circle on a map
- Involves two steps:
  - Create (instantiate) circle object and set its properties
  - Call the setMap() method for the circle which inserts it into the map

24 **Instantiate a Circle Object (Page 1)**

- Call the google.maps.Circle constructor to create the circle object
- Format:

  var *circleObject* = new google.maps.Circle(

  {

    (*circle properties and values*)

  });

25 **Instantiate a Circle Object (Page 2)**

- Some circle object properties:
  - center—the google.maps.LatLng (latitude…longitude) of the center location of the circle
  - radius—measured in meters on the map
  - strokeColor—hexadecimal color of border as a string, e.g. "#FFFFFF"
  - strokeOpacity—transparency of border from 0.0 (0%) to 1.0 (100%)
  - strokeWeight—border thickness in pixels

26 **Instantiate a Circle Object (Page 3)**

- Some circle object properties (*con.*):
  - fillColor—hexadecimal color of circle as a string, e.g. "#FFFFFF"
  - fillOpacity—transparency of circle from 0.0 (0%) to 1.0 (100%)

27 **Instantiate a Circle Object (Page 4)**

- Example:

```
var myCircle = new google.maps.Circle(
{
   center: myCenter,
   radius: 50,
   strokeColor: "#000099",
   strokeOpacity: 0.8,
   strokeWeight: 2,
   fillColor: "#000099",
   fillOpacity:0.4
});
```

28 **Method setMap() for a Circle**

- The setMap() method for the circle object inserts it into map
  - This operation takes place after the map object already has been instantiated
- Format:

  circleObject.setMap(*mapObject*);
- Example:

  myCircle.setMap(map);

29 **InfoWindow**

- A infoWindow is an “overlay” object that appears as a text message in a rectangular box usually above a marker object
- Involves two steps:
  - Create (instantiate) infoWindow object and set its properties
  - Call the setMap() method for the infoWindow which inserts it into the map

30 **Instantiate an InfoWindow Object (Page 1)**

- Call the google.maps.InfoWindow constructor to create the infoWindow object

- Format:
  ```
  var infoWindowObject = new google.maps.InfoWindow(
  {
     content: "textString"
  });
  ```
  - ▫ The *textString* is the text message that appears in the infoWindow box

31 **Instantiate an InfoWindow Object (Page 2)**

- Example:
  ```
  var infoWindow = new google.maps.InfoWindow(
  {
      content: "Hello New York"
  });
  ```

32 **Method setMap() for an InfoWindow**

- The setMap() method for the infoWindow object inserts it into map
  - ▫ This operation takes place after the map object already has been instantiated
- Format:
  infoWindowObject.setMap(*mapObject*, *markerObject*);
  - ▫ The *markerObject* is the marker object to which the infoWindow is attached
- Example:
  infoWindow.setMap(map, marker);

33 **Try It Out**

- api1d.htm

34 **Polyline**

- A polyline is an "overlay" object that draws a line through a series of coordinates in sequence
- Involves two steps:
  - ▫ Create (instantiate) the polyline object and set its properties
  - ▫ Call the setMap() method for the polyline which inserts it into the map

35 **Instantiate a Polyline Object (Page 1)**

- Call the google.maps.Polyline constructor to create the polyline object
- Format:
  ```
  var polylineObject = new google.maps.Polyline(
  {
      path: pathArrayObject,
     (other polyline properties and values)
  });
  ```
  - ▫ The *pathArrayObject* is an array that consists of a series of LatLng (latitude…longitude) objects

36 **Instantiate a Polyline Object (Page 2)**

- Some other polyline object properties:
  - strokeColor—hexadecimal color of border as a string, e.g. "#FFFFFF"
  - strokeOpacity—transparency of border from 0.0 (0%) to 1.0 (100%)
  - strokeWeight—border thickness in pixels

37 **Instantiate a Polyline Object (Page 3)**

- Example:

```
var flightPath = new google.maps.Polyline(
{
   path: myTrip,
   strokeColor: "#0000FF",
   strokeWeight: 3
});
```

38 **Polyline Coordinates (Page 1)**

- Array coordinates for the polyline path consist of a series of LatLng (latitude…longitude) objects
- Format:

  var *pathArrayObject* = [*latLngObject1*, *latLngObject2*, *latLngObject3*, …];

39 **Polyline Coordinates (Page 2)**

- Example:

```
var newYork=new google.maps.LatLng(40.7128, -74.0059);
var houston=new google.maps.LatLng(29.7604, -95.3698);
var sanFrancisco=new google.maps.LatLng(37.7749, -122.4194);
var myTrip = [newYork, houston, sanFrancisco];
```

40 **Method setMap() for a Polyline**

- The setMap() method for the polyline object inserts it into map
  - This operation takes place after the map object already has been instantiated
- Format:

  polylineObject.setMap(*mapObject*);
- Example:

  flightPath.setMap(map);

41 **The addDomListener() Method**

- A DOM event of google.maps.event that runs a function after the Web document loads
- Can be used to draw the map once the document has loaded
- Format:

  google.maps.event.addDomListener(window, "load", *functionName*);
- Example:

```
google.maps.event.addDomListener(window, "load", drawMap);
```

42 **Try It Out**

- api2.htm

43 **Polygon**

- A polygon is an “overlay” object that draws a line through a series of coordinates in sequence
- It is different from a polyline in that it is designed to define a region within a “closed loop”
- Involves two steps:
  - Create (instantiate) the polygon object and set its properties
  - Call the setMap() method for the polygon which inserts it into the map

44 **Instantiate a Polygon Object (Page 1)**

- Call the google.maps.Polygon constructor to create the polygon object
- Format:

var *polygonObject* = new google.maps.Polygon(
{
    path: *pathArrayObject*,
  (*other polygon properties and values*)
});

45 **Instantiate a Polygon Object (Page 2)**

- Some other polygon object properties:
  - strokeColor—hexadecimal color of border as a string, e.g. "#FFFFFF"
  - strokeOpacity—transparency of border from 0.0 (0%) to 1.0 (100%)
  - strokeWeight—border thickness in pixels
  - fillColor—hexadecimal color of polygon as a string, e.g. "#FFFFFF"
  - fillOpacity—transparency of polygon from 0.0 (0%) to 1.0 (100%)

46 **Instantiate a Polygon Object (Page 3)**

- Example:

```
var flightPath = new google.maps.Polygon(
{
  path: myTrip,
  strokeColor: "#0000FF",
  strokeWeight: 3
});
```

47 **Polygon Coordinates (Page 1)**

- Array coordinates for the polygon path consist of a series of LatLng (latitude…longitude) objects
- Format:

var *pathArrayObject* = [*latLngObject1*, *latLngObject2*, *latLngObject3*, …, *latLngObject1*];
  - The *pathArrayObject* array for a polygon should *end* with the same LatLng (latitude…longitude) object that *began* it to complete the loop

48 **Polyline Coordinates (Page 2)**

- Example:

```
var newYork=new google.maps.LatLng(40.7128, -74.0059);
var houston=new google.maps.LatLng(29.7604, -95.3698);
var sanFrancisco=new google.maps.LatLng(37.7749, -122.4194);
var myTrip = [newYork, houston, sanFrancisco, newYork];
```

49 **Method setMap() for a Polygon**

- The setMap() method for the polygon object inserts it into map
  - This operation takes place after the map object already has been instantiated
- Format:

  polygonObject.setMap(*mapObject*);
- Example:

  flightPath.setMap(map);

50 **Try It Out**

- api2a.htm

51 **Google Map API Events (Page 1)**

- JavaScript responds to interactions within Google maps by generating events specific to elements within the mao
- Every Google Maps JavaScript API object responds to a number of events
- Event listeners for those events are registered by calling the addListener() method
  - Specifies an event to listen for and a method to call when the event occurs

52 **Google Map API Events (Page 2)**

- Some of the events to which the Google Maps API responds are:
  - center_changed
  - click
  - dblClick
  - drag
  - mouseMove
  - resize
  - zoomChanged

53 **Google Map API Events (Page 3)**

- Method addListener() is a member of the google.maps.event package
- Format:

  google.maps.event.<u>addListener</u>(*mapElement*, "*event*", function() { … });

54 **Google Map API Events (Page 4)**

- Example:

```
google.maps.event.addListener(marker, "click", function()
{
    map.setZoom(15);
    map.setCenter( marker.getPosition() );
});
```

  - Calls the specified method when an object variable named “marker” is clicked

55 **The click Event**

- Occurs (fires) when a Google Maps API object is clicked
- Format:
  google.maps.event.<u>addListener</u>(*mapElement*, "<u>click</u>", function() { ... });

56 **The setZoom() Method**

- For a map object zooms the map to the value of the *int* argument
- Format:
  mapObject.setZoom(*int*);
- Example:
  map.setZoom(15);
  - The *int* property specifies the zoom level for the map (zero (0) shows a map of the Earth fully zoomed out and higher zoom levels zoom to a closer resolution)

57 **The setCenter() Method**

- For a map object sets the center of the map to the location of the *positionObject*
- Format:
  mapObject.setCenter(*positionObject*);
- Example:
  map.setCenter(marker.getPosition());

58 **The getPosition() Method**

- For Google Map API map elements *gets* (returns) the location (latitude...longitude) of the object
- Format:
  mapElement.getPosition()
- Example:
  map.setCenter(marker.getPosition());

59 **Try It Out**

- api3.htm

60 **The center_changed Event**

- Occurs when the center of the map is moved (usually when it is dragged)

- Format:
  google.maps.event.addListener(*mapElement*, "center_changed", function() { … });

61 **The setTimeout() Method**

- A window method that calls a function after a specified number of milliseconds
- Format:
  window.setTimeout(function() { … }, *milliseconds*)
- Example:

```
window.setTimeout( function()
{
   map.panTo( marker.getPosition() );
},
3000);
```

62 **The panTo() Method**

- For Google Map API map elements returns the map to location of a LatLog (latitude…longitude) position object
- Format:
  mapElement.panTo()
- Example:
  map.panTo(marker.getPosition());
-

63 **Try It Out**

- api3a.htm

64 **The open() Method**

- For instantiated Google Map API map elements “opens” the object (makes it visible)
- Required arguments are:
- Format:
  mapElement.open(*mapObject*, *positionObject*)
  - The *mapObject* in which it will open
  - The *positionObject* where the object will display
- Example:
  infoWindow.open(map, marker);
-

65 **Try It Out**

- api3b.htm

66 **Controls (Page 1)**

- The Google Maps API has a set of controls for manipulating the map when it is displayed
- The default controls are:

  - Zoom—displays a slider or "+/-" buttons to control the zoom level of the map
  - Pan—displays pan control for panning the map
  - MapType—lets a user toggle between map types (roadmap and satellite)
  - Street View—displays “Pegman” icon which can be dragged onto map to enable “Street View”
  -

67 **Controls (Page 2)**

- Other controls include:
  - Scale—displays a map scale element
  - Rotate—displays a small circular icon for rotating map
  - Overview Map—displays a thumbnail map reflecting the current map view within a wider area

68 **Disabling All Controls (Page 1)**

- To disable all default controls for a map, set the disableDefaultUI property for the map properties to true
- Format:

```
disableDefaultUI: true
```

69 **Disabling All Controls (Page 2)**

- Example:

```
var mapProperties =
{
   center: new google.maps.LatLng(lat, lng),
   zoom: 12,
   disableDefaultUI: true,
   mapTypeId: google.maps.MapTypeId.ROADMAP
};
```

70 **Try It Out**

- api4.htm

71 **Enabling Controls (Page 1)**

- Making controls visible and/or invisible is implemented through map properties of type boolean
- Format:

controlName: true/false

72 **Enabling Controls (Page 2)**

- Example:

```
var mapProperties =
{
   center: new google.maps.LatLng(lat, lng),
   zoom: 12,
   panControl: true,
   zoomControl: true,
   mapTypeControl: true,
   streetViewControl: true,
   overviewMapControl: true,
   mapTypeId: google.maps.MapTypeId.ROADMAP
};
```

73 **Try It Out**

- api4a.htm

74 **The mapTypeControl Object (Page 1)**

- The mapTypeControl lets user toggle between map types (roadmap and satellite)
- The mapTypeControlOptions field from class google.maps.MapTypeControlStyle can be set to specify the format of this control
  - HORIZONTAL_BAR—displays one button for each map type (default)
  - DROPDOWN_MENU—selects map types from a drop-down menu

75 **The mapTypeControl Object (Page 2)**

- Formats:
  google.maps.MapTypeControlStyle.HORIZONTAL_BAR
  google.maps.MapTypeControlStyle.DROPDOWN_MENU

76 **The mapTypeControl Object (Page 3)**

- Example:
  google.maps.MapTypeControlStyle.HORIZONTAL_BAR
  google.maps.MapTypeControlStyle.DROPDOWN_MENU